POLITECNICO DI TORINO

Master's Degree in ICT for Smart Societies

Master's Degree Thesis

Machine learning on the edge for Industry 4.0 applications

Supervisors

Prof. Lorenzo BOTTACCIOLI

Prof. Daniele MAZZEI

Candidate

Anastasiya ISGANDAROVA

July 2021

Abstract

Nowadays, billions of connected devices are installed in vehicles, home appliances, and industries. The Internet of Things (IoT) enables to connect heterogeneous electronic devices efficiently and continuously, can analyze information from these devices, and make decisions based on this information. It creates the possibility to transfer data over a network without human-to-human or human-to-machine interaction.

The phenomenon of the IoT provides many opportunities in the manufacturing sector to manage the maintenance of industrial machines. Industrial IoT is believed to transform the traditional concept of the factory into fully integrated manufacturing systems. Industrial machines can be equipped with many devices and interact with the outside world. Machine Learning has been effectively applied to various process optimization, monitoring, and control applications in manufacturing and predictive maintenance in different industries.

With the advent of Industry 4.0, Industrial IoT has been increasingly used in the acquisition of critical production data and turn this data into valuable insights of efficiency of the manufacturing operations. In particular, production monitoring gives real-time insights using sensors and provides early warning notification of the equipment conditions. By using IoT sensors, one can mitigate machine damage or production slowdowns. Once the machine failure is detected, highly experienced machine operators can either retrofit or repair this machine. Retrofitting means rejuvenating the existing machine with an IoT solution by adding new features or improving the existing ones.

The thesis work is primarily concerned with the retrofitting of machines to infer the machine state using power consumption analysis through machine learning on the edge algorithms. The hypothesis we start with is that the power consumption profile can be used as a fingerprint of the different machines' activities.

We considered the main application fields of Industry 4.0 and analyzed which enabling technologies are applied in industrial use cases to solve Industry 4.0 problems. In particular, we developed two taxonomies, one for enabling technologies in the Industry 4.0 world and the second for classifying Industry 4.0 problems. The taxonomies were accomplished by reviewing the current state-of-the-art of

Industry 4.0 challenges based on academic papers, white papers, and use cases in industrial enterprises. This information is crucial for future strategic planning in the industrial sector, both for process optimization and for product innovation in different industries.

In order to deeply understand the production monitoring Industry 4.0 problem, we have considered two case studies, in the first case study we took into account power consumption data of several home appliances inside a house and the second one was considering the state of coffee machine based on power consumption data. It was hard to organize the data acquisition stage within the factory due to the Covid19 restrictions. For this reason, we have built an ad-hoc dataset of power consumption collecting data from simplified machines, such as a coffee machine and dishwasher. Moreover, the power consumption profiles of these simplified machines were representative enough of the industrial ones.

The first use case included the acquisition of home appliance power consumption data. The approach involved recording the energy consumption of the respective device and evaluating the measured data using machine learning algorithms and tools. Machine learning algorithms were used to understand the power consumption profile given only non-invasive data. IoT solutions used to solve retrofitting problems should be non-invasive to simplify the implementation of the solution and the maintenance processes.

In the second use case, we studied a coffee machine, on which multiple sensors were mounted. We used a Deep Learning model to infer the machine's state based on time-series power consumption data of the coffee machine. Since it would be adaptable to any kind of industrial machine, the aim was to keep the method as generic as possible. The results of these use cases can be used in future industrial developments.

Acknowledgements

I would like to say “thank you” to all the people who have helped me with my thesis; Professor Bottaccioli and Professor Mazzei for their guidance and assistance during the entire master thesis project. A warm thank you to Karim Hamdy, who inspired me with his deep knowledge in microcontrollers and treated me as a friend, and the entire working group at “Zerynth” for their hospitality while I was an intern in their office.

This thesis would not have been possible without the support of my family. The most important “thank you” goes to my mother Olga, who was always near to encourage me even being far away. A special thank goes to my beloved sister Tanuwka, who motivated me every day during this challenging period and listening to my messages even during her internship in the nuthouse. Thanks to my grandparents for writing me long lovely messages given their speed.

I want to thank all my friends who have been with me during all these years abroad; “thank” goes to my dear Seva and Elya with whom I have shared all difficulties and who were always ready to help especially by sharing with me a video of Zyuma to improve my mood. “Thank” goes to Leyla, for being the first person apart from my family, who lived with me for two years being both my roommate and friend (it is not easy, I know). “Thank you” for being always ready to listen to my exams’ stories and to eat Chinese food. A big “grazie mille” goes to Manu who was has been near throughout all this time and gave me all his master-chief receipts to taste with the phrase “mangi come un uomo”.

Thanks to all my friends who I have met along with my master’s degree for their support and encouragement throughout my studies.

Table of Contents

List of Tables

List of Figures

Acronyms

AI

Artificial Intelligence

ML

Machine Learning

IoT

Internet of Things

DBSCAN

Density-Based Spatial Clustering of Applications with Noise

ZDM

Zerynth Device Manager

FOTA

Firmware Over The Air

API

Application Programming Interface

PCA

Principal Component Analysis

LLE

Locally Linear Embedding

IRQ

Interquartile Range Method

KNN

K-Nearest Neighbors

DT

Decision Tree

LDA

linear Discriminant Analysis

LR

Logistic Regression

SVM

Support Vector Machine

ANN

Artificial Neural Network

IWSN

Industrial Wireless Sensor Network

RF

Random Forest

DL

Deep Learning

RNN

Recurrent Neural Network

LSTM

Long Short-Term Memory

GRU

Gated Recurrent Unit

Chapter 1

Introduction

The fourth Industrial Revolution, also known as I4.0, is a new concept that involves technological innovations in industrial processes. In Germany, the term “Industry 4.0” was coined when the government and private sector, led by Bosch, formed a research group to improve German manufacturing’s competitiveness in the global market [1]. This term covers all manufacturing domains and includes advanced manufacturing technologies that collect, optimize, and deploy data.

Smart factories are equipped with sensors, embedded software, and robotics solutions that collect and analyze data for better decision-making. Industry 4.0 technologies assist businesses in managing production planning and scheduling, capacity usage, maintenance, and energy conservation. The focus of Industry 4.0 is to combine production, information technology and the internet. The advantages of digitalization range from increased manufacturing performance to the deployment of new products and services. By exploiting new production models based on cloud-interconnected systems and big data management, mass manufacturing will become more efficient. Over the last few years, Industry 4.0 has developed as a promising framework for integrating and extending manufacturing processes at both intra- and inter-organizational levels [2]. The Industry 4.0 concepts are proposed to enable companies to have more flexible manufacturing processes and to analyze huge amount of data in real time, resulting in better strategic and operational decision-making [3]. The methodologies and ideas in the context of "Industry 4.0" are represented in the disciplines of electrical engineering, business administration, computer science, business and information systems engineering, and mechanical

engineering.

The goal of the thesis is production monitoring without intervention by industrial machines. We tried to investigate non-invasive industrial power consumption analysis driven by Machine Learning (ML) on the edge as a source of information for the understanding of what machines are doing to provide data for production monitoring. The challenge is to train ML algorithms by using only non-invasive data.

We considered two different use cases to focus on the investigation of non-invasive industrial power usage: one for consumption monitoring and the other for machine state analysis. The first use case aims to address the first problem, which is the state of power consumption. Using untagged data from household appliances, an unsupervised ML model was trained to distinguish between the various types of power consumption activities. Because of existing technologies, the model can be easily moved to the edge. The second use case aims to figure out what a coffee machine is doing based on its power consumption and infer the machine's state. The data acquisition for the second case of coffee machine was done inside the Zerynth company, thanks to the Zerynth Platform.

Zerynth provides a way for developers, system integrators, and businesses to enable IoT for their products. Zerynth, based in Pisa, Italy, offers global support through a comprehensive network of European partners. Zerynth's IoT experts assist with the creation of end-to-end IoT products, from hardware to firmware, cloud, and end-user interface.

We are interested in their industrial IoT solutions and kits, which help machines collect data and digitize the entire industrial process. Thanks to the Zerynth Platform for simplifying and accelerating the development of IoT applications and many successful use cases in the optimization of industrial processes, Zerynth was chosen as a company to implement the thesis.

The thesis work is organized as follows:

- Chapter 1 – introduces the problem we are addressing and our objectives: Non-Invasive Industrial Data Analysis powered by ML on the edge. In this chapter, we introduce Zerynth, the company in which the experimental part of the thesis was implemented, and give an overview of the suggested solution.

- Chapter 2 – introduces the research that has been done in the Industry 4.0

environment based on academic papers, business use cases, and white papers to establish two taxonomies for industry 4.0 problems and enabling technologies used to solve them, as well as the relationship between them.

- Chapter 3 – introduces a real need for product monitoring in the Industry 4.0 environment and demonstrates some research work done in this area using ML algorithms.
- Chapter 4 – discusses the Methodologies used to solve the problems. It provides theoretical knowledge of the ML algorithms used in the first use case and an overview of the neural network used in the second use case. The detailed description of the Zerynth platform, on which the data acquisition was done, is also given in this chapter.
- Chapter 5 – explains the generation of data, how data was acquired for the first use case of power consumption of home appliances, and the ML algorithms that were used to identify the power consumption profile of home appliances.
- Chapter 6 – explains how data was generated and acquired for the second use case of coffee machine state analysis, and the DL model that was trained for this use case.
- Chapter 7 - the thesis work is completed by highlighting the most relevant findings and making recommendations for future research.

Chapter 2

A Taxonomy for Industry 4.0 problems

A question that many experts ask nowadays is **what are the typical problems that can be addressed by Industry 4.0?**

Industry 4.0 allows comprehensive real-time tracking of operations, allowing for real-time data collection, monitoring, and maintenance; it introduces a new or substantially improved method of production, service provision, mode of supply, storage, distribution, or a significant enhancement of business support activities. One of the challenges faced by Industry 4.0 is managing the large amounts of data that are generated by analyzing production data and coordinating the findings with customer information systems.

Big data analysis may enable managers to detect defects, faults, and shortcomings in the manufacturing process at an early stage; optimize automation processes and conduct trend analyses, use resources more efficiently, and perform predictive maintenance. Predictive maintenance is used to identify equipment failures before they occur, increasing manufacturing efficiency.

Given this wide range of potential enhancements enabled by Industry 4.0 applications, another question arises: **What are the 4.0 key enabling technologies?**

In the Industry 4.0 environment, the European Commission identifies nine main enabling technology trends (Figure 2.1.) and investigates possible technological and economic benefits for manufacturers and production equipment suppliers [4], [5]. We used the enabling technologies provided by European Commission as guidelines

for our research, for technology' categorization. These enabling technologies are

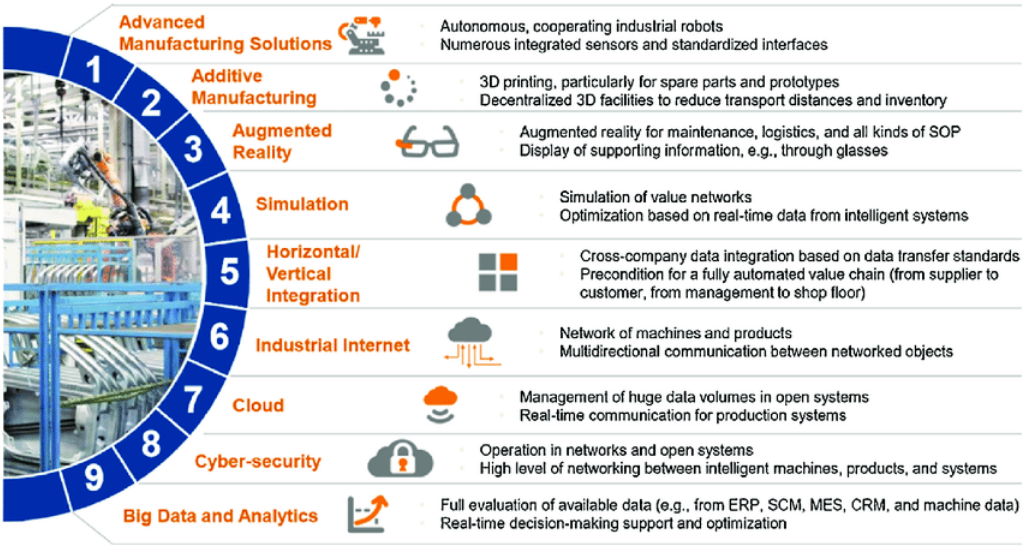


Figure 2.1: Key nine enabling technology trends. The image was taken from [5].

Advanced Manufacturing Solutions, Additive Manufacturing, Augmented Reality, Simulation, Horizontal/Vertical Integration, Industrial Internet, Cloud, Cybersecurity, Big Data, and Analytics; their existence helps in turning the manufacturing process into a completely digitized and intelligent one.

Moreover, according to the Microsoft report [6], critical technology drivers for IoT success in the coming years are AI, Edge Computing, and 5G. Companies should be aware of these enabling technologies' capabilities and they should have a detailed technology roadmap and framework of how they could embed AI in their strategies and businesses.

From the industry point of view, AI technologies are methods and procedures that allow technical systems to perceive their environments, solve problems independently, find new ways of solving challenges, make decisions, and, most importantly, learn from experience to become more proficient at completing tasks and actions.

The AIoT paradigm, which combines AI with IoT , has gained popularity in recent years. It's an innovative computing approach that can contribute to the creation of more scalable, effective, and secure I4.0 solutions [7].

To help companies to use enabling technologies, we tried to answer the following questions: **1) Which of these Industry 4.0 addressable problems and which of these Industry 4.0 key enabling technologies have been implemented**

and used in industrial use cases? 2) Is it possible to create a taxonomy that categorizes I4.0 problems and enabling technologies? 3) Which is the relationship between I4.0 problems and enabling technologies?.

2.1 4.0 Taxonomy Building

To answer these questions, there is a need to look beyond the scientific articles. As a result, we conducted research using academic papers, white papers, business use cases and consulting company reports as sources. Two taxonomies were developed, one based on enabling technologies used in the Industry 4.0 environment, and the other for classifying I4.0 problems. The two taxonomies are connected by the use of white paper, academic papers and business use cases. Each use case was regarded as a representative of one or more Industry 4.0 problems, from which we extracted the I4.0 enabling technologies that were used to solve them.

2.2 Industry 4.0 problems

First of all, we can divide the I4.0 innovation problems into two main categories: Product Innovation and Process Optimization.

- **Product Innovation** – aims at building a new generation of products that are IoT connected, allowing businesses to move towards a service-based solution. Product Innovation in the I4.0 environment also aims at improving the product's usability by making it easier to use and understand.
- **Process Optimization** – aims at improving the efficiency of production processes. Industrial processes can be optimized by improving the equipment efficiency, workforce, and supply chain. It changes the production processes and may not necessarily have an explicit impact on the final output while increasing productivity and reducing production costs.

From Product Innovation we can derive other subcategories of I4.0 problems:

- **Product Servitization** – is the innovation of an organization's processes and capabilities to better create mutual value through a shift from the sale of Product Service Systems rather than products.

- **Usability Improvement** – is a product usability improvement by designing a user interface, with usability testing, targeting customers.
- **After-Sales Support** – any service provided after a customer has purchased a product.
- **Inventory Management** – is a systematic approach to sourcing, storing, and selling inventory, both raw materials (components) and finished goods (products).
- **Cost and Number of Parts Reduction** – is the achievement of a real and permanent reduction in the unit cost of products manufactured.

From Process Optimization we can derive subcategories of I4.0 problems:

- **Equipment Efficiency Improvement**
  - **Real-time Production Monitoring and Supervision** – is a constant stream of data about the overall health and performance of assets and production lines. Real-time alerts can assist operators in quickly addressing any issues before they become major failures that can cost plants time and money. Operators can make faster data-driven decisions to ensure optimal performance when they possess appropriate information.
  - **Predictive Maintenance** – detects equipment failures early, allowing store managers to prepare spare equipment and avoid downtime. The IoT enables warehouse managers to save money by reducing downtime and machine repair costs.
  - **Smart Scheduling** – The adoption of Smart Scheduling will allow industries to benefit from the cumulative experience in their fields (for instance, facilitating the selection of appropriate rescheduling strategies).
  - **Vertical Interconnection and Integration (between departments in a factory)** – involves the procedure of acquiring business operations within the same production vertical.
- **Worker security improvement and accident prevention**
  - **Smart PPE (Personal Protection Equipment)** – such as safety shoes, earplugs, and protective eyewear has always been critical in protecting the wearer from one or more occupational safety and health (OSH)

hazards. PPE is required for the person to do his/her job safely or with less risk of injury. It is important to provide a high level of protection for workers.

  - **Worker Attention and Mental State Monitoring** – it is essential to improve working conditions in the plant or warehouse based on real-time temperature, humidity, and other data, to quickly detect the presence of gasses, radiation are the main factors that influenced mental state monitoring.

- **Worker routine and operation optimization**

  - **Time and Method Smart Measurement** – Scheduling is the process of arranging and controlling workloads via shift work scheduling and data analytics.

- **Supply Chain**

  - **Horizontal interconnection and integration (between different actors of the supply chain)** – refers to the supply chain's IT systems and flows.

  - **Smart warehouse** – IoT solutions for warehouse management provide real-time data on product locations, transportation conditions, packaging integrity, and more. Thanks to real-time updates, store managers can ensure that no inventory is lost during transportation and that supply chain vendors manage deliveries responsibly.

  - **Supply chain transparency and reliability improvement** – is the fundamental process of increasing visibility, traceability, and transparency throughout the supply chain by collecting and exchanging information and communicating it to authorized internal and external stakeholders. The purpose of enhancing supply chain reliability is to meet consumer demand as quickly and efficiently as feasible.

2.3 Industry 4.0 needs and enabling technology identification

Table 2.1 represents a list of sources that can be used to build a taxonomy of both problems and enabling technologies. As we can see from the table, white papers, academic papers, and business use cases were all considered sources for the development of the two taxonomies.

Table 2.1: List of articles for development of a Taxonomy.

Article Name	Article Type	Publication Date
AI in production [8]	white paper	March 2019
Optimization of machining process [9]	academic paper	March 2019
AI for defect detection [10]	business use case	March 2020
IoT Impacts the Supply Chain [11]	white paper	October 2019
Smart Scheduling [12]	academic paper	August 2018
Process Discovery: Capturing the Invisible [13]	academic paper	March 2010
Condition monitoring in I4.0 [14]	academic paper	2019
Industry 4.0 in the tobacco industry [15]	academic paper	February 2020
Logistic Industry Innovations with IoT [16]	white paper	September 2019
ML in Demand Forecasting [17]	business use case	June 2020
Edge - Computing [18]	business use case	August 2020
Predictive Maintenance using ML [19]	business use case	

Continued on next page

Table 2.1 – continued from previous page

Article Name	Article Type	Publication Date
5G Heart of Industry 4.0 [20]	white paper	
Real-Time Production Performance Monitoring [21]	business use case	
Smart Warehouse System [22]	business use case	
Blockchain-enabled IoT shipment tracking system [23]	business use case	
Vehicle Tracking and Fleet Management [24]	business use case	
Smart Tracking Solution [25]	white paper	
Reducing Complexity with Simplicity [26]	academic paper	August 2015
IoT Order Fulfillment [27]	white paper	
Asset Tracking [28]	business use case	
Predictive Maintenance [29]	business use case	
Predictive Analytics [30]	business use case	
IoT in Retail [31]	business use case	
Remote Measurements Control[32]	busines use case	
IT plus OT convergency for your automation solution [33]	white paper	
Data driven technology for efficiency in energy intensive industries [34]	white paper	
Improving stores efficieny using clients shopping times [35]	white paper	

Continued on next page

Table 2.1 – continued from previous page

Article Name	Article Type	Publication Date
Smart PPE and IoT to improve workplace safety [36]	white paper	
Cost Management For After-Sales Supply Chain [37]	white paper	
IoT Retail with Consumer Smartphone detection [38]	business use case	
IoT Industry Solution [39]	white paper	
Tracking System [40]	white paper	
Product Service System for aerospace maintenance [41]	academic paper	May 2012
Energy sources optimization [42]	white paper	
Predictive maintenance and production optimisation[43]	white paper	
Leveraging IoT Framework [44]	academic paper	September 2020

2.4 Industry 4.0 enabling technologies

Out of the key nine enabling technological trends identified by the European Commission in the Introduction section, the research focused on Advanced Manufacturing Solutions, Horizontal/Vertical Integration, Industrial Internet, Cloud, Big data, and Analytics. According to the information gathered, these five enabling technologies are critical in resolving Industry 4.0 problems. Table 2.2 lists all the technologies investigated in the literature which are considered as enabling technologies of Industry 4.0. The taxonomy was derived using the main categories identified by the European Commission.

Many of the nine essential enabling technologies that form the foundation for Industry 4.0 are currently used in manufacturing. However, with the usage of Industry 4.0, it will change production: optimized cells will join together as a fully integrated, automated, and optimized production flow, transforming traditional

production relationships among suppliers, producers, and customers — as well as between humans and machines. The list of enabling technologies that were considered when creating a taxonomy of enabling technologies are [5]:

- **Big Data and Analytics** – is the collection and comprehensive evaluation of data from multiple sources that will become a standard to support real-time decision making.
- **Industrial Internet** – devices will be enriched with embedded computing and connected using standard technologies.
- **Cloud** - machine data and functionality will be deployed to the cloud, enabling more data-driven services for production systems.
- **Horizontal/Vertical Integration** – functions from the enterprise to the shop floor level are not fully integrated. With Industry 4.0, companies and departments will become much more cohesive, universal data-integration networks evolve.
- **Advanced Manufacturing Solutions** – the greater connectivity and interaction of Industry 4.0–capable machines and systems in their factories, manufacturing-system suppliers will have to expand the role of IT in their products.

Table 2.2: List of enabling technologies.

Enabling Technologies	
Horizontal/Vertical Integration	1. DataBases (a) SQL DB (b) Non SQL DB (c) Time series DB i. InfluxDB ii. Prometheus iii. Graphite 2. Blockchain 3. Connectivity (a) GSM/4G/5G (b) MQTT, Node-Red (c) REST API and Webhook (d) RFID/NFC (e) Bluetooth (f) LPWAN

Continued on next page

Table 2.2 – continued from previous page

Enabling Technologies	
Industrial Internet	1. Industrial IOT (a) Industrial communication protocols i. Ethernet Protocols ii. Fieldbus Protocols iii. Wireless Protocols (b) Industrial Gateway and data acquisition device

Continued on next page

Table 2.2 – continued from previous page

Enabling Technologies	
Big Data and Analytics	1. Data science (a) Data Visualization i. Grafana ii. Kibana iii. Metabase (b) Data Analytics i. Data Lake and Warehouse ii. Data Mining iii. All-inclusive tools A. Tableau B. PowerBI C. Elastic Stack D. SAP BI E. QlikView/Qlik Sense iv. Data Engines A. Apache Hadoop B. Apache Kafka C. Apache Spark 2. IoT (a) Signal Processing 3. AI (a) Machine Learning (b) Deep Learning (c) Reinforcement Learning (d) Continuous Learning (e) Computer Vision (f) Natural Language Processing

Continued on next page

Table 2.2 – continued from previous page

Enabling Technologies	
Cloud	1. Container Technology (a) Docker (b) Kubernates (c) Terraform 2. Serverless programming (a) AWS Lambda functions (b) Azure functions 3. Device Management (a) Zerynth Device Manager (b) AWS IoT Device Management (c) Azure IoT Hub (d) WinCC OA IoT OPA 4. Cloud Data Storage (a) AWS S3 (b) Google Cloud Storage (c) Microsoft Azure Storage 5. Edge Computing (a) AWS green grass (b) Azure Edge IoT (c) Custom solutions based on docker swarm (d) Multi-access edge computing (MEC) (e) AWS Wavelength

Continued on next page

Table 2.2 – continued from previous page

Enabling Technologies	
Advanced Manufacturing Solutions	1. Embedded Computing (a) Arduino (b) STM32 (c) ESP32 (d) FRGA 2. Sensors (hardware) 3. Signal Processing 4. Blockchain 5. Connectivity (a) GSM/4G/5G (b) MQTT, Node-Red (c) REST API and Webhook (d) RFID/NFC (e) Bluetooth (f) LPWAN

2.5 Relation of Industry 4.0 problems and enabling technologies

We created a relationship map between problems and enabling technologies based on the two taxonomies indicated above, where a specific I4.0 problem was resolved in an industrial use case employing a specific I4.0 enabling technology.

The generated taxonomy can be used as a search engine at the following link: `https://docs.google.com/spreadsheets/d/1JyHiKv_kBnENtA8yGMV3Sn9x05hKOLJwf49LwdYiEU0/edit?usp=sharing`.

For example, in the article "Optimization of Machining Process" [9], we can notice that this is an I4.0 process optimization problem that can be solved using Big Data, Cloud, and Horizontal/Vertical Integration as enabling technologies. The Figure 2.2 represents the branch of the taxonomy graph for this article.

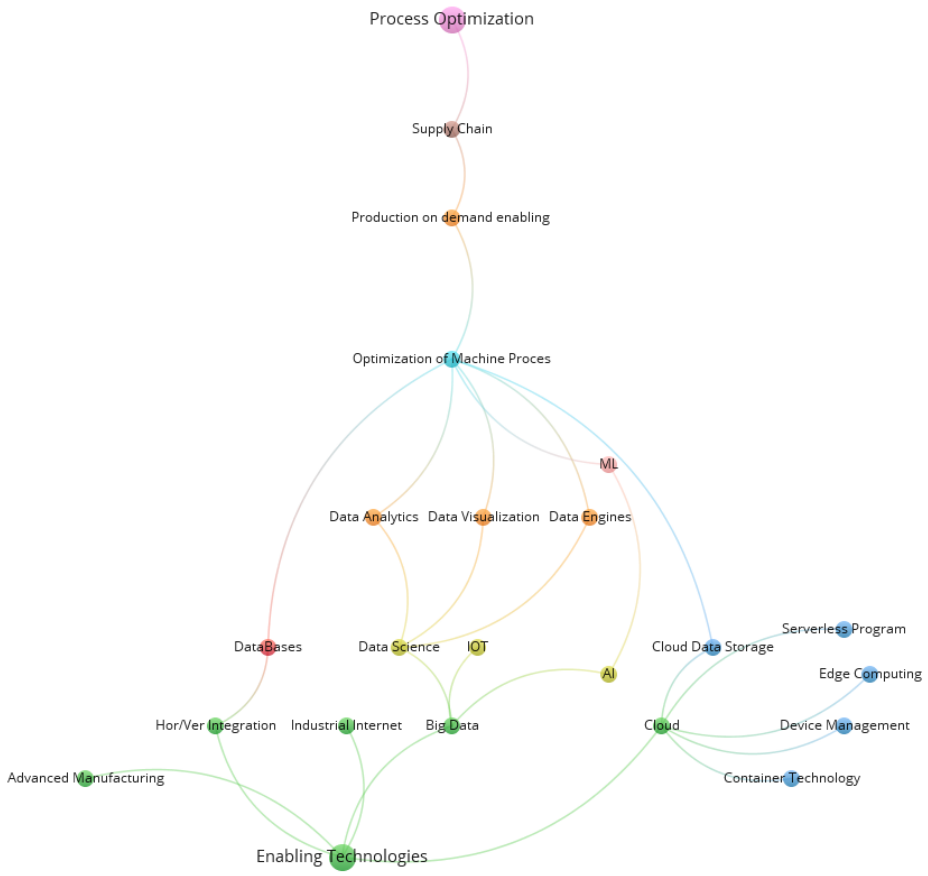


Figure 2.2: The branch of taxonomy's graph

Unfortunately, we were unable to identify a use case for every problem that exists in the Industry 4.0 world; while, it is worth pointing out that out of considered papers Real-time Production Monitoring Analysis, Predictive Maintenance, Smart Scheduling, Smart warehouse, Cost, and several parts/component reduction, Supply

chain transparency and reliability improvement are said to be the most frequently addressed problems in the Industry 4.0 environment. According to considered business use cases, white papers, and research articles, the most commonly used I4.0 enabling technologies are Time-series Databases, Industrial Communication Protocols, Cloud Data Storage, Data Analytics, Data Visualization, ML, and Sensors.

Table 2.3 represents the ranking list of the most commonly used I4.0 enabling technologies and problems in the industrial sector. The rank was created by ordering problems and enabling technologies based on the number of articles associated with them.

Table 2.3: The rank of the most used enabling technologies and problems found in articles.

Enabling Technologies	**Number of articles**	**Problems**	**Number of articles**
Time series Database	20	Smart warehouse	7
Data Visualization and Dashboarding	18	Real-time Production monitoring and analysis	6
Cloud Data Storage	18	Cost and number of parts/-component reduction	6
Data Analytics	17	Smart Scheduling	5
Sensors	16	Supply chain transparency and reliability improvement	5
Machine Learning	14	Predictive maintenance	4
Connectivity	12		
Signal Processing	11		
Industrial communication protocols	9		

To enhance the 4.0 transformation, it is crucial to provide both the academic and industrial world with a clear picture of I4.0 problems and enabling technologies used to solve them. This taxonomy is the initial step in this direction, setting the way for further research targeted at better understanding how the I4.0 world is evolving.

After completing some exploratory work, we focused this thesis work on process optimization, one of the two main categories of I4.0 innovation problems. We considered Product Monitoring to be one of the subcategories of the I4.0 process optimization problem. It is a challenging manufacturing issue. This paper focuses

on how to enhance it using I4.0 technologies, namely by the use of Machine Learning and Deep Learning. After careful consideration of I4.0 problems and enabling technologies that are employed to solve them, we decided to consider two different use cases for the demands of production monitoring and power consumption analysis by using ML algorithms and DL neural network as an enabling technology.

Chapter 3

State of the Art

Migration to Industry 4.0 may be less costly and more incisive with the application of the retrofitting concept, which is the fastest and most cost-effective solution to boost equipment efficiency, reduce production costs and increase industry connectivity. There are two main ways to upgrade an industrial machine: rebuild (the major and some minor components of a machine are replaced, and a revamped machine operates like a new one) and retrofit (new technology or features are added to the machine).

Retrofitting can do more than just update an industrial machine; it can also rejuvenate the entire industrial park, allowing for a continual updating of knowledge about the role of each asset in the process. Rebuilding or retrofitting existing equipment has numerous advantages. First, a repair or retrofit is less expensive than purchasing a new industrial machine. Second, any operational faults are resolved with a rebuild, and the machine's life cycle is extended. However, not all existing industrial machines can be rebuilt or retrofitted, so it is crucial to understand your options for each specific machine.

Production monitoring provides you with data on the entire manufacturing cycle, points out any faults at any step, and helps prevent shipping delays. It collects data directly from the production line to cover a wide range of performance criteria, including product faults, machine downtime, and more. It allows production managers to make real-time adjustments or design solutions to specific problems or customer requirements.

Manufacturers are rushing to replace obsolete equipment to accomplish their

Industry 4.0 goals. With the contribution of ML algorithms, accurate fault occurrence can lead to minimizing downtime and a cost-effective maintenance schedule. The data should be accurately interpreted to identify the various faults at the production level and to correct them as soon as possible to increase efficiency. In the meantime, given the resource-constrained nature of IoT-based devices, ML at the edge can provide advanced prediction skills and effective resource management.

Daniel Ramotsoela et al.[45] presented a comparison of ML algorithms for anomaly detection on industrial wireless sensor networks (IWSNs), including K-Nearest Neighbor, SVM, ANN, and hybrid approaches. Anomaly detection techniques can detect abnormal changes in the environment where IWSNs are deployed. They can be used in a variety of applications that require prior knowledge, and computation overhead should be taken into consideration in the scheme design. Over a third of the non-parametric schemes examined in the study required little or no prior knowledge, while less than a third included practical considerations. Parametric methods do not apply to many IWSN applications since they require the distribution function to be known in advance and to remain constant during the network's lifespan. ML techniques are more robust, but they come at a high cost in terms of computational overhead. In literature, practical feasibility has been neglected, and it should be paid more attention to in the future.

Ali Azadeh et al.[46] proposed a unique flexible algorithm for classifying centrifugal pump conditions. To perform accurate classification in the manufacturing field, they considered ANN, SVM with genetic algorithms and SVM with particle swarm optimization. The goal of this research was to develop a reliable and effective method for monitoring the status of a centrifugal pump to detect two types of faults. Flow, temperature, suction pressure, discharge pressure, velocity, and vibration are the major features used to monitor the system's condition. These features have been classified by the proposed integrated algorithm, which is appropriate for corrupted and noisy data. They calculated the performance of the proposed flexible algorithm in noisy situations to demonstrate the robustness of the algorithm. The results showed that the support vector classifier improves when combined with a genetic algorithm and particle swarm optimization.

Patel and Giri [47] investigated the use of a Random Forest (RF) classifier to diagnose multi-class mechanical faults in an induction motor bearing. An accelerometer sensor was used to capture the bearing vibration data; statistical

features from the bearing vibration signal were calculated and classified using RF classifiers for four class problems. The purpose of the study was to evaluate the efficiency of RF in fault diagnosis. The outcomes were compared to those obtained by an ANN. According to the findings, RF outperformed the neural network approach in terms of performance and accuracy, and it performed well on large datasets.

Muhammad Syafrudin et al.[48] considered IoT-based sensors, big data processing, and a hybrid prediction model as a solution for providing an efficient real-time monitoring system of the manufacturing process. The characteristics of IoT-generated sensor data from the manufacturing process include real-time, large amounts, and unstructured data. The big data processing platform uses Apache Kafka as a message queue, Apache Storm is a real-time processing engine, and NoSQL MongoDB to store sensor data. The hybrid prediction model consists of Density-Based Spatial Clustering of Applications with Noise (DBSCAN)-based outlier detection and RF classification. It was used to remove outliers in sensor data and to provide fault detection throughout the manufacturing process. DBSCAN was used to separate outliers from typical sensor data and RF was used to predict faults based on the sensor data. In comparison to the other models, the results demonstrated that the proposed hybrid prediction model is effective and accurate.

The above-mentioned experimental results are likely to improve product monitoring during manufacturing, lowering the risk of unanticipated losses caused by faults. The focus of the thesis is to monitor the power consumption profile of the coffee machine and home appliances by using Machine Learning and Deep Learning techniques. We considered previous work done in this area in order to understand better which algorithms were used.

With the advancement in embedded devices, in terms of processing power, energy storage, and memory capacity, the ability to derive significant value from having ML on the edge has increased. Low-power and connected systems, including sensors, are used in homes, vehicles, and industries. Historically, ML algorithms have been run on servers and high-performance machines; however, putting ML inference on edge devices holds a lot of promise.

If the model has been trained and the coefficients have been determined, they are stored in program memory and the ML algorithms can be executed on a device with limited RAM. ML algorithms are used in a variety of areas, including smart

cities, health care, automotive, and the industrial environment. They can detect suspicious activity in agriculture, smart homes, and healthcare by collecting data from multiple sensors or machines.

Han et al. [49] introduced the role of socio-economic factors in forecasting energy use. By estimating the daily peak and total load and developing a data-driven model for the prediction of household characteristics in residential buildings, they investigated the significance of socio-economic factors for residential customers. Their experimental analysis consisted of over 3800 households providing consumption data for 1.5 years. To evaluate the main features, they used a random forest algorithm to examine the collected dataset. Their results showed that the proposed random forest method outperforms a Support Vector Machine -based method in accurate prediction and scalability.

Carolina et al. [50] concentrated on the heating of Danish dwellings with heat pumps, analyzing daily load profiles of power consumption using the K-Means method. Their research discovered that socio-economic characteristics are critical in supporting better demand-side management. The research was done based on hourly recordings from 139 dwellings and two main clusters were identified for both weekends and weekdays. The main cluster has a constant load profile and a minor one with variations during the day. The difference between two head load patterns is due to building characteristics like house area, building year, floor heating and so on. The results demonstrate that the heating load profiles of dwellings with heat pumps vary according to the external load conditions.

The prediction model of energy consumption of household appliances has been proposed by Lei Xiang et al. [51]. In the paper, the data has been filtered to exclude non-predictive parameters and is used for training. Five prediction models were trained using repeated cross-validation and were evaluated in the testing set. They used SVM, K-Nearest Neighbor, Random Forest, Long Short-Term Memory network, and Extreme Random Forest. The analysis of five algorithms shows that the deep learning method has advantages in the prediction of energy consumption of house appliances.

John E.Seem [52] studied a pattern recognition algorithm for deciding which days of the week have similar power consumption profiles. The irregular energy consumption data was removed from the clusters using outlier analysis algorithms, both univariate and multivariate. A pattern recognition algorithm was used to

determine the days of the week with a similar consumption of energy. Amber et al. [53] compared prediction capabilities of five different intelligent system techniques (Multiple Regression, Genetic Programming, Artificial Neural Network, Deep Neural Network, and Support Vector Machine) by predicting consumption of electricity of an administration building located in the UK. The data was collected for five years, taking into account five different parameters, such as solar radiation, temperature, wind speed, humidity, and the weekday index. The weekday index is an important parameter that is used to differentiate between weekdays and weekends. Four years of data are used for training the models and to predict data for the fifth year. The results show that the Artificial Neural Network performs better than all other techniques. The Deep Neural Network requires a large amount of data for training and the dataset in the research was limited. However, even with a limited dataset, the performance of the Deep Neural Network was almost the same as in other methods.

Chapter 4

Methodology explanation

The idea of this chapter is to provide an overview of the proposed system architecture as well as to introduce the machine learning and deep learning techniques that were used to analyze the non-invasive power consumption profile in this thesis work. According to the proposed approach, we need to acquire physical power usage statistics from home appliances or a coffee machine. The challenge of the ML application in manufacturing is the acquisition of relevant data.

This chapter provides a more detailed discussion of the Zerynth platform. For both use cases indicated in the introduction section, the Zerynth platform was utilized to collect data. Figure 4.1 depicts the workflow of our process: starting from the data acquisition stage, we were able to train ML models to identify power consumption profiles.

Data was collected using 4ZeroBox, an easy-to-configure tool for data acquisition. Through an Ethernet cable, the gathered data was sent to the LAN of the Zerynth company's local servers. The data was sent to Zerynth Device Manager, running on the local servers, for further elaboration. Data from ZDM can be transferred to a dashboard for future visualization, depending on the viewer's specific needs.

Due to the untagged gathered dataset, we used unsupervised ML algorithms to explore interrelationships among a collection of patterns by grouping them into homogeneous clusters. After labeling the data, we used supervised ML algorithms to determine which class the power consumption data belonged to.

To give the reader a better understanding of the thesis work, the next subsections of this chapter will outline the main theoretical background and tools that we

employ in this work.

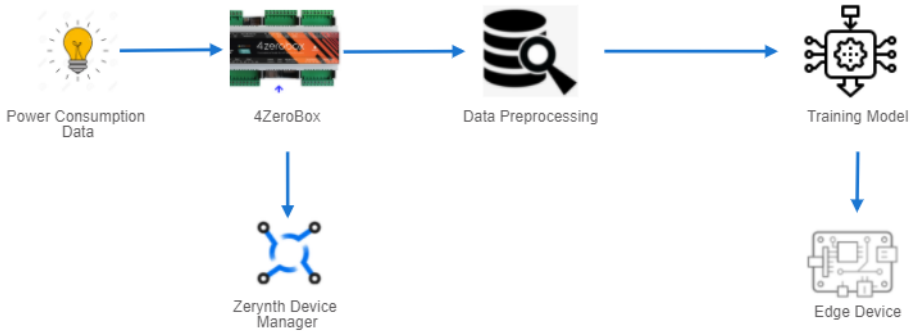


Figure 4.1: Graphical depiction of the workflow.

4.1 Machine Learning

ML is a branch of Artificial Intelligence (AI) that focuses on developing applications that learn from experience (data) and improve their accuracy over time, much as humans do.

It's difficult to describe ML precisely, but (Arthur Samuel, 1959) described it as follows:

"ML is the field of study that gives computers the ability to learn without being explicitly programmed."

ML algorithms are trained to find patterns and features from the data to produce reliable and informed results. It is the ability to adapt to new data in a self-contained and iterative manner. The more accurate the model is, the better the algorithm is. If the predicted outcome does not match what was intended, the algorithm is retrained until the desired output is found. It allows the ML algorithm to learn on its own and generate the best possible response, which will improve in accuracy over time.

To create a ML program, an engineer should first gather a substantial set of training data, which is then fed into a special type of algorithm, which discovers the rules without being aware of all the complexities.

The ML algorithm builds a model of the system based on the given dataset during the training phase. This data is used in the model to make predictions, which is referred to as an inference. To create a ML model, one should use the following four basic steps:

- Selecting and preparing a training dataset.
- Choosing an algorithm to run on the training dataset.
- Training the algorithm to create the model.
- Using and improving the model.

Complex ML algorithms can be run on even the smallest microcontrollers thanks to advances in microprocessor and algorithm design. Embedded ML is the application of ML to embedded systems. Jeff Bier coined the acronym BLERP to describe the core benefits of implementing ML on embedded devices [54] :

- **Bandwidth** - ML algorithms on the edge devices can extract meaningful information from data that would be otherwise inaccessible due to the bandwidth constraints.
- **Latency** - ML models on the edge can respond in real-time to inputs, which would not be possible in case of dependence on network latency.
- **Economics** - embedded ML systems avoid the costs of transmitting data over a network and processing it in the cloud.
- **Reliability** - systems which are controlled by ML on the edge models are inherently more reliable than those which depend on a connection with the cloud.
- **Privacy** - when data is processed on an embedded system and is never transmitted to the cloud, the user privacy is protected.

ML has been divided into the following primary categories [55] :

- **Supervised ML** - trains itself on a labeled dataset. It requires less training data than other ML models, and the training process is simplified since the model's results can be compared to labeled data.

- **Unsupervised ML** - learning unlabeled data and uses algorithms to extract meaningful features which are needed to label and classify the data.
- **Semi-Supervised ML** - it is a medium between supervised and unsupervised learning. A smaller labeled dataset is used to guide classification and feature extraction from a larger unlabeled dataset during the training phase.
- **Reinforcement Learning** - the algorithm discovers data through a process of trial and error and then decides what action results in higher rewards.

4.1.1 Unsupervised Machine Learning

Pattern recognition is performed without the use of a target attribute in Unsupervised ML algorithms. These learning techniques are used for learning complex, highly non-linear models with millions of parameters to use a large amount of unlabeled data [56]. Unsupervised methods can detect potentially interesting and new cluster structures in a dataset without requiring a training set that includes a priori knowledge of objects' class labels as input. The most important unsupervised ML algorithms are as follows [55] :

- **Clustering** – the primary goal is to group similar instances into clusters. Examples: K-Means, DBSCAN, Hierarchical Cluster Analysis.
- **Anomaly detection and novelty detection** – the goal is to learn what "normal" data looks like and use it to detect abnormal instances. Examples: One-class SVM, Isolation Forest.
- **Visualization and dimensionality reduction** – the goal is to preserve as much structure as possible and understand how the data is organized. Examples: Principal Component Analysis (PCA), Kernel PCA, Locally - Linear Embedding (LLE), t - distributed Stochastic Neighbor Embedding.
- **Association rule learning** – the goal is to dig into large amounts of data and discover the relations between attributes. Examples: Apriori, Eclat.

There are several unsupervised learning algorithms to choose from, but clustering was chosen for this project because it seems to be the best due to the absence of a label for the given dataset and the necessity of an exploratory analysis of the data itself.

Clustering Algorithm

The primary goal of Clustering Algorithms is to group a set of objects according to their similarities. The distance between two objects is used to calculate their similarity. The higher the similarity between two objects, the shorter the distance between them. The main distinction in the clustering techniques can be done between [57]:

- hierarchical clustering - "a set of nested clusters is organized as a hierarchical tree".
- partitional clustering - "a division of data objects into non - overlapping subsets (clusters) such that each data is in exactly one cluster".

Other distinctions between clusters are exclusive or non - exclusive (in the second case, points could belong to different clusters at the same time), fuzzy or non - fuzzy (in the fuzzy, each point has a coefficient with a value between 0 and 1 and the weight of the probability of that point to belong to any cluster), partial or complete (in some cases, only part of the data is taken into the consideration) and heterogeneous or homogeneous (clusters with different sizes and shapes). There are some types of clusters that exist:

- **well - separated clusters** – any point in a cluster is closer to all those in the same cluster than to those which are in any other cluster.
- **center - based clusters** – any object in a cluster is closer to the "center" of a cluster: "center" is the average of all cluster's points.
- **contiguous - based clusters** – a point in a cluster is closer to one or more other points in the same cluster than to any point not in the cluster itself.
- **density - based clusters** – a cluster is a dense region of points and different dense regions of points are separated by non - dense ones.
- **shared property/conceptual clusters** – clusters are created by sharing some common property or points that have the same particular concept.
- **clusters defined by an objective function** – clusters are created to minimize or maximize an objective function.

K-Means clustering

The K-Means Algorithm is an iterative method proposed by Stuart Lloyd at Bell Labs in 1957 as a technique for pulse-code modulation; in 1965, Edward W.Forgy published the same algorithm, that's why K-Means is known as Lloyd-Forgy. It is a clustering algorithm for dividing n observations into k clusters. This algorithm generates the above-mentioned center-based clusters. A centroid (the average of all the points inside the cluster) or a medium (the most representative point of a cluster) may be the cluster's center. Algorithm 1 shows an implementation of the K - Means clustering algorithm.

Algorithm 1: K - Means Algorithm

```
Select K points as the initial centroids;
while Centroids do not change do
    Form K clusters by assigning all points to the nearest centroid;
    Recompute the centroid of each cluster;
end
```

Since the mean squared distance between the instances and their closest centroid can only go down at each step, the algorithm is guaranteed to converge after a certain number of iterations.

Clustering Evaluation Techniques

Some methods, such as the evaluation of the number of squared errors (SSE), the evaluation of performance using the silhouette index, and the elbow procedure, can be used to evaluate a k-means clustering algorithm. The error is defined as the distance of each sample to their closest cluster center and the SSE is computed as 4.1:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2) \tag{4.1}$$

A set of N samples X is divided into K disjoint clusters C using the K-Means algorithm. Each cluster is described by the mean μ_j of the samples in the cluster. The objective of the K-Means algorithm is to choose centroids that minimize the SSE. The ideal situation is to have a small number of clusters and a small SSE value at the same time.
The silhouette index can be used to assess the consistency of clusters within clusters. For data point $i \in C_i$ in the cluster C_i two values a_i and b_i are calculated:

- a(i) – the average distance between its own cluster and any other one belonging to its same cluster.

- b(i) – the smallest average distance from i to all points in any other cluster, apart from the one to which i actually belongs.

The a_i and b_i can be calculated as:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq i} d(i, j) \tag{4.2}$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{4.3}$$

C_k is any cluster to which i does not belong. The silhouette index for the data point i can be calculated as

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}, \text{if } |C_i|>1 \tag{4.4}$$

or as

$$s(i) = 0, \text{if } |C_i| = 1 \tag{4.5}$$

The silhouette value ranges from -1 to 1, with a high value indicating a good match between a point and the cluster it belongs to. For values of k, the average silhouette method calculates the average silhouette of observations. The elbow method's goal is to determine the optimum "k" number of clusters based on the SSE. The idea is to pick a "k" where SSE begins to flatten out and form an elbow. The overall value of SSE can not be improved by adding another cluster.

4.1.2 Supervised Machine Learning

The primary goal of supervised ML is to learn a target function that will be used to predict a class's values. The best scenario is the algorithm that can correctly determine the class labels for the unseen dataset. The learning algorithm can "reasonably" generalize from the training data to the unknown situation. Models are trained using a labeled dataset, where the model learns about the type of data, and then this model is tested based on the test dataset and predicts the output.

Classification and regression problems are two fields where supervised learning is useful.

- *Classification* – is the task of learning qualitative categories from the input. The output of the model is defined in a discrete space. Here, we formalize the problem of binary classification, which can be easily extended to the multiclass problem iterating the procedure. Given a training set of observations $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, the binary classification problem consists into estimating a function $f(x_i) = y_i$.
- *Regression* – is the task of learning a quantitative outcome variable from the input. The output of the model is defined in a continuous space. Formally, given a set of observations $x_1, x_2, ..., x_n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, the binary classification problem consists into estimating a function $f(x_i) = y_i$.

In this work, we focus on five supervised ML algorithms.

Logistic Regression

Logistic Regression (LR) is a powerful and well-established method for supervised classification and is commonly used to represent the occurrence or non-occurrence of an event. It's a classification problem extension of the linear regression model. It helps to find the probability that a new instance belongs to a particular class; the result ranges from 0 to 1. The multinomial logistic regression model is a generalization of the logistic regression model that can be used to model a categorical variable with more than two values. The Logistic Regression model computes a weighted sum of the input features and outputs the logistic of the result. The logistic function is a sigmoid function that returns a number between 0 and 1.

Linear Discriminant Analysis

Linear Discriminant Analysis is a method for determining a linear combination of features that characterizes or distinguishes two or more groups of objects. The aim of LDA is to model group differences and project features from a higher-dimensional space into a lower-dimensional space. It can be implemented in three steps:

- First step – is to calculate the separability between different classes (distance between the mean of different classes).

- Second step – is to calculate the distance between the mean and sample of each class.
- Third step – is to construct the lower-dimensional space that maximizes the between-class variance and minimizes the within-class variance.

k-Nearest Neighbors Algorithm

The K-Nearest Neighbors Algorithm (KNN) is a supervised ML algorithm that is used to solve classification and regression problems. It is simple and easy to implement. It consists of two stages: the first is determining the nearest neighbors ("k" is the number of closest neighbors), and the second is determining the class using those neighbors. For a data record that should be classified, its "k" nearest neighbors are retrieved, and this forms the data sample's neighborhood. The dataset is split into two parts: a training dataset for training the algorithm and a testing data set for testing the algorithm's output on new data. For the same dataset, different classification results can be obtained by selecting the K-nearest data points. Finally, the data point is assigned to the class that contains the majority of the "k" data samples. It doesn't work well with high-dimensional data because calculating distances in each dimension is difficult, and with categorical features because calculating distances between dimensions with categorical features is difficult.

Decision Tree Classification Algorithm

Decision Tree (DT) algorithms can perform both classification and regression tasks. The aim is to learn basic decision rules from data features to build a model that predicts the value of a target variable. A decision tree begins with a single node, branches into potential outcomes, leading to additional nodes. The split criterion: Gini index or Entropy, is used to make decisions at these nodes. Each leaf in the decision tree corresponds to one class or its probability. The classification algorithm branches to the child node, where the testing and branching phase are repeated until the lead node are reached. The decision nodes are responsible for splitting the data according to the decision rules, while the leaves are the outcomes, i.e. the prediction. Figure 4.2 shows a toy example of a decision tree for a classification problem.

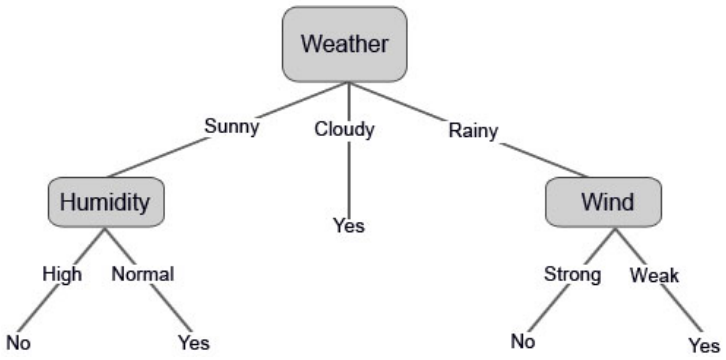


Figure 4.2: A toy example of decision tree for a playing tennis decision based on weather conditions taken from [58]. The task can be declined as a classification problem. The decision tree is separated into two levels: the first is based on the attribute "weather", and the second is based on the attributes "humidity" and "wind".

The prediction by using the DT method is fast and works with high-dimensional data. However, due to the tree's continuous growth before the data is separated, there is a risk of overfitting. It reduces the training error but increases the test error.

Support Vector Machine

A SVM is a powerful and versatile ML model that performs linear and nonlinear classification, regression, and outlier detection. It distinguishes between two or more classes of data by defining an optimal hyperplane in multidimensional space that divides them. A margin is a gap between the two lines on the nearest class points; support vectors are the data closest to the hyperplane that will be removed, causing the hyperplane to be redefined. The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. It is possible to find the optimal marginal hyperplane by generating a hyperplane that best segregates the groups. The kernel trick is a function that takes a low-dimensional input space and converts it to a higher-dimensional one, which is what the SVM algorithm does for a non-linear separation problem. It helps in building a more accurate classifier.

The linear classification function of SVM separates the training data into two classes with a linear separating hyperplane. Figure 4.3 represents data of two

classes, where the green squares belong to the negative class and the blue circles to the positive one. The goal of SVM is to put a linear boundary in the middle of two classes. The distance between two imaginary lines is the maximum margin [59]. The training data represented by

$$(x_1, y_1), \ldots, (x_l, y_l), \quad x \in \mathbb{R}^n, y \in \{+1, -1\} \tag{4.6}$$

can be divided by a hyperplane equation:

$$(w \times x) + b = 0 \tag{4.7}$$

The separation of hyperplane can be explaned by using the following form:

$$y_i[w \times x_i + b] \geq 1 \quad i = 1,2,\ldots,l \tag{4.8}$$

The problem of finding the optimal hyperplane can be transferred to the problem

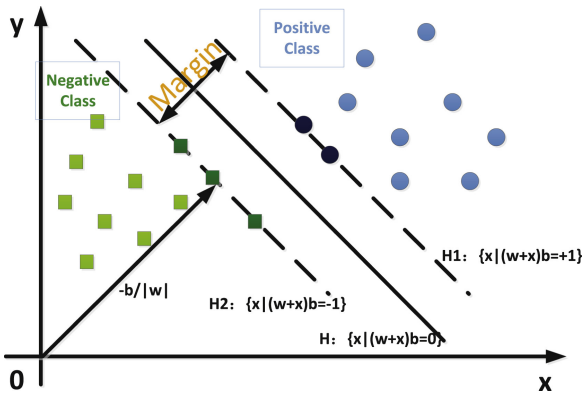


Figure 4.3: Two classes classification using SVM. This image was taken from [59].

of finding minimum of the function:

$$\Phi(w) = \frac{1}{2}\|w\|^2 = \frac{1}{2}(w \times w) \tag{4.9}$$

The optimal hyperplane can be found by solving a quadratic programming problem. In a non-linear case, we should always take into consideration the nonlinearities or noise (the source of non-linear data). It is hard to separate with a straight line. In this case, we use Kernel functions that transform non-linear spaces into linear ones. It transforms data into a new dimension that may be classified.

4.1.3 Overfitting and underfitting

Overfitting means that the ML model performs well on the training data while fitting poorly on the testing set (unseen dataset). It is a general problem in ML that can't be totally avoided. It occurs due to different reasons either the limits of the training dataset, which can have a limited size or include plenty of noises, or the constraints of the algorithms that require too many parameters or too complex. Instead of learning the discipline hidden behind the data, overfitted models that try to memorize all of the data take into account even inevitable noise on the training dataset. Underfitting occurs when the model is too simple to learn the underlying structure of the data. Some techniques exist to prevent overfitting [60]:

- **Early - Stopping** – is used to avoid the phenomenon "learning speed slow-down". The algorithm's accuracy stops improving after a certain number of iterations or even gets worse. If we stop learning before this point, it's underfitting; otherwise, after that point it is overfitting. The aim is to determine when to stop training to prevent both scenarios.
- **Expansion of the training data** – an expanded data set can improve the accuracy of the prediction to a great extent. The enormous size of data will increase the training time.
- **Cross Validation** – the idea is to use initial training data to generate multiple train-test splits and then use them to tune the model. The data is split into k subsets, which are called folds. The aim is to train the algorithm on k-1 folds iteratively, with the remaining fold serving as the test set. The overall estimation error is the average test error of iterations.
- **Regularization** – to select only the useful features and remove the useless ones or to minimize the features' weights that influenced a little bit on the final classification. The regularization's amount to apply during learning can be controlled by a hyperparameter (a parameter of a learning algorithm).

4.2 Deep Learning

Deep Learning (DL) is a subset of ML and of AI. The relationship between AI, ML and DL is shown in Figure 4.4. With DL, the algorithms do not need to be told

about the important features, since it is possible to find them from data by using a "neural network". AI allows computers to mimic human behavior.

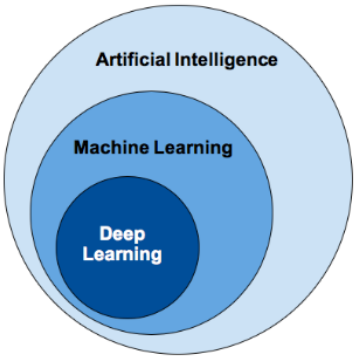


Figure 4.4: Relationship between AI, ML and DL. The image was taken from [61].

An artificial neuron, similarly to the biological one, consists of a set of input (dendrites) and a single output (axon). The neuron's main objective is to process the x_i inputs using a weighted sum $\sum(x_i w_i)$ add a given b bias, which serves to increase or decrease the activation function input and apply an activation function ϕ. The neuron applies the following mathematical operations in the inputs x_i to calculate the output y:

$$y = \phi(\sum_i(\omega_i x_i) + b) \tag{4.10}$$

Figure 4.5 represents the architecture of a single neuron. When all neurons in a layer are connected to every neuron in the previous layer, it is called a fully connected layer or a dense layer. The final layer produces a value (or values) that is either the predicted value in case of a regression problem or the predicted label (or the probability to belong to each label of target data) in case of the classification problem.

Layered neural networks are the most widely used. Figure 4.6 shows an example of a layered structure neural network. Hidden layers are intermediate layers between the input and output layers and are where all the computation is done. It is as if each of the hidden layers focuses on a particular feature of input data to recognize a very complex pattern. Every layer except the output one includes a bias neuron and is fully connected to the next layer. The number of hidden layers is a parameter

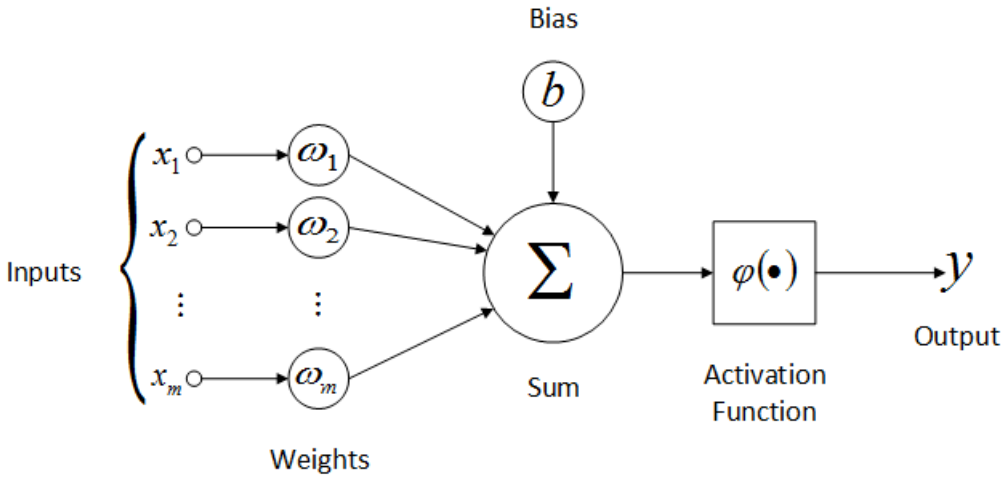


Figure 4.5: Single neuron model. Figure was taken from [62].

that can be set based on the task's difficulty.

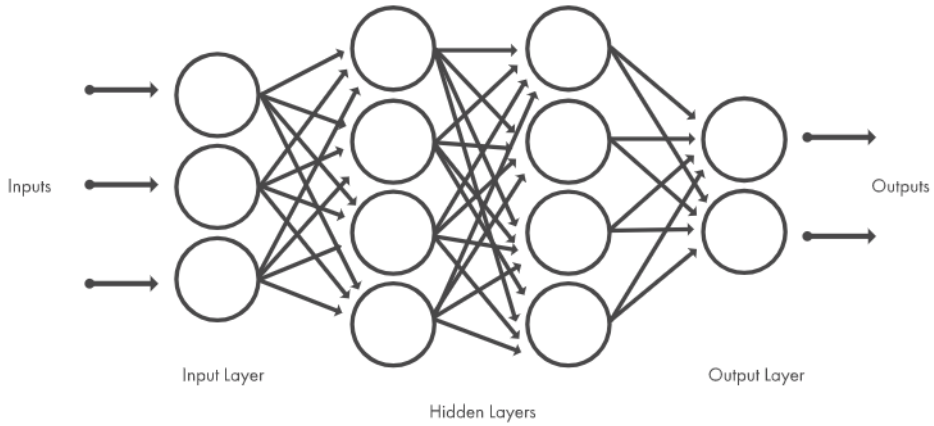


Figure 4.6: Typical neural network architecture. Figure was taken from [63].

Activation function

The activation function defines the output of a neural network. It is a fundamental component in Neural Networks. The basic process carried out by a neuron in a neural network is represented in Figure 4.7. The function is attached to each neuron in the network and determines whether or not it should be activated, based

Figure 4.7: The process implemented by a neuron in a neural network. The image was taken from [64].

on whether or not each neuron's input is relevant to the model's prediction. The activation functions can be divided into two types: Linear Activation Function and Non-Linear Activation Function.

For this thesis, we used the "softmax" activation function. The softmax function can be used to express a probability distribution over a discrete variable with n potential values. Softmax functions are used as the output of a classifier, to represent the probability distribution over n different classes. The softmax function returns a vector of values that sum to 1.0, which can be interpreted as class membership probability. The standard softmax function defines $\sigma : \mathbb{R}^k \to [0,1]^K$ is defined by the formula [65]:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K. \tag{4.11}$$

It applies the standard exponential function to each element z_i of the input vector z and then normalizes these values by dividing by the sum of all these exponentials. The denominator is a normalizing term made up of the sum of the numerators, ensuring that all node outputs add up to one.

Optimizer

Adam is an optimization algorithm that has been designed specifically for training deep neural networks. It can be used to update network weights iteratively based on training data instead of the traditional stochastic gradient descent approach. This algorithm computes individual adaptive learning rates for various parameters from estimations of first and second moments of the gradients. The detailed description of the Adam algorithm is represented [66].

Hyper-parameters

The learning process is greatly influenced by hyper-parameters. They have an impact on the model that is created and its ability to predict outcomes. The following are the hyper-parameters:

- **Batch size** – number of training samples used in one single iteration of the training process. The smaller is the batch size value the better is the accuracy obtained for the model that requires a larger number of updates, which means larger computations and a slower learning process. At the end of the batch, the predictions are compared to the expected output variables and an error is calculated.

- **Epoch** – number of times that the learning algorithm will work through the entire training dataset during the training session. There are one or more taches in an epoch. The number of epochs is often large allowing the learning algorithm to run until the error from the model has been sufficiently minimized. Good model performance can be achieved exploiting several epochs.

- **Learning rate** – it controls the speed at which Neural Network learning parameters are updated. Choosing the learning rate is challenging because a value that is too small may result in a long training process that could get stuck in a local minima without improvement; whereas a value too large may result in an unstable training process, exceeding weights and bias optimal value and the algorithm to fail to converge.

- **Number of hidden layers** – If data is less complex and is having fewer dimensions or features then neural networks with 1 to 2 hidden layers would work. If data has a lot of dimensions or features, you can use 3 to 5 hidden layers to get an optimum solution. It should be kept in mind that increasing the number of hidden layers increases the model complexity and choosing hidden layers in two digits may sometimes lead to overfitting.

- **Number of neurons per layer** – Using too few neurons in the hidden layers will result in something called underfitting, while too many neurons in the hidden layers may result in overfitting. The number of hidden neurons should be $\frac{2}{3}$ the input layer's size, plus the size of the output layer. It provides a

starting point for you to consider. The selection of an architecture for your neural network will come down to trial and error.

In all hyper-parameters there is an intrinsic trade-off between performance level and computational costs: high accuracy values require ideal settings.

4.2.1 Recurrent Neural Network

A Recurrent Neural Network (RNN) [67] is a broad class of Neural Network that is designed to deal with temporal sequences. The neurons of RNN send each other feedback signals.

Figure 4.8 represents the basic scheme of the RNN. An input of the RNN is not only the current input instance but also background information preserving the description of the previous inputs instances, called hidden state [67]. The output of some of the hidden layers is saved into the hidden state and recursively fed back as input for the successive input instance. The cost function during the training

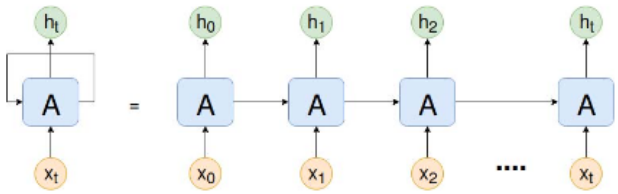


Figure 4.8: Basic scheme of the RNN (left) and unrolled version of the same RNN (right). The input sequence x_t is sent to the network sequentially. The network block A produces the output h_t, which depends not only from the input, but also from the state of the network block A at time t. The state information transmission is represented by the arrows going from A a time step t to A at time step $t + 1$. The image was taken from [68].

phase has to flow from output to input to update the weights. This procedure is called backpropagation.

In the RNN case, the error should flow not only back to the input layer, but also back in time, since the part of the input is coming from the hidden state. Generally, the weights are set to random values near zero. The gradient flowing back has an exponential decay, being multiplied for values near zero at every step back in time. Weight updates become negligible and the network becomes difficult to train. Moreover, the learning time arises because the error vanished as it gets propagated

back. This problem is called the vanishing gradient problem. A detailed overview of the vanishing gradient problem and the methods employed to overcome it are given in [69].

Long Short-Term Memory

Hochreiter and Schmidhuber [70] proposed the Long Short-Term Memory (LSTM) type of RNN model to overcome the problem of vanishing gradients. It decouples the cell state and the output, allowing the gradient to flow back unchanged since the cell state allows information to flow through it substantially unmodified [71]. Although the accuracy of LSTM is higher than that of other algorithms, the training time of LSTM is longer.

Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is a special case of LSTM. It has a shorter training time than traditional LSTM and has fewer parameters than LSTM due to the lack of an output gate. Figure 4.9 shows the structure of gated recurrent unit network, where $z(t)$ is an update gate vector, $r(t)$ is a reset gate vector. Similar to the gates within LSTMs, the reset and update gates control how much and which information to retain. There are two input features at each time: $x(t)$ - is the input vector and $h(t-1)$ is the previous output vector. The output of each gate can be obtained through logical operation and nonlinear transformation of input. More information about GRU is written in [72].

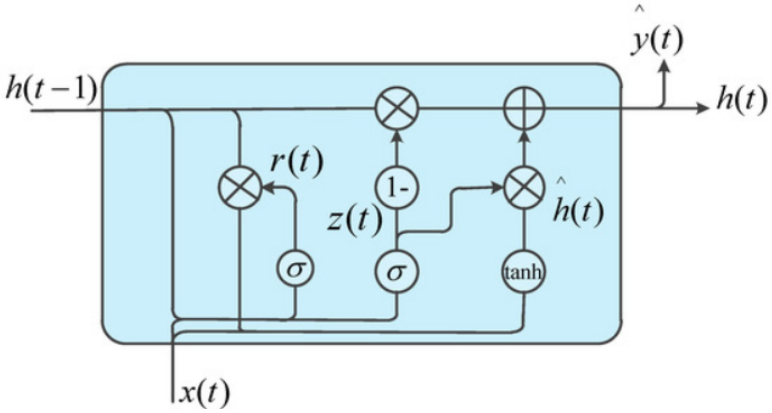


Figure 4.9: The structure of gated recurrent unit (GRU) network. The image was taken from [72].

4.3 Machine Learning on the Edge Platform

4.3.1 TensorFlow

TensorFlow is an open-source ML library that was first released to the public in 2015 with the slogan "An Open Source Machine Learning Framework for Everyone" after being created by Google's Brain Team for internal use. One of the primary needs, when TensorFlow was released, was the ability to train models and run them in desktop environments [73]. Google launched TensorFlow 2.0 in late 2019, with a significant update that made it more user-friendly, leading to renewed interest in the ML community.

TensorFlow Lite comes with everything you need to convert TensorFlow models and run them on mobile, embedded, and Internet of Things devices. It was designed to make ML on the edge more convenient, rather than requiring data to be sent back and forth from a server. It enhances the following characteristics [74]:

- **Latency** - there's no round-trip to a server
- **Connectivity** - no need for the internet connection
- **Privacy** - no data needs to leave the device
- **Power consumption** - power-hungry network connections

The straightforward training workflow is represented in Figure 4.10, and the model is ready to be trained and used on embedded devices thanks to the TensorFlow Lite converter [75].

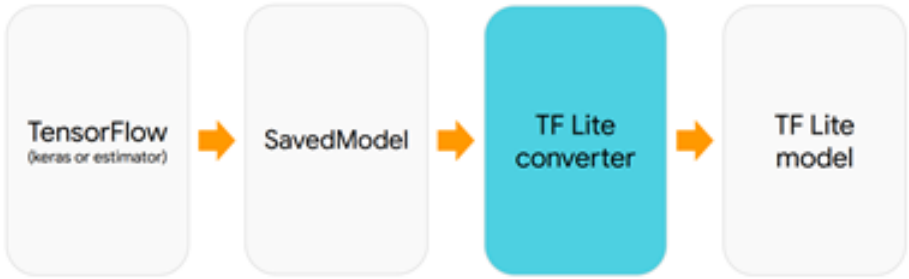


Figure 4.10: Training workflow by using TensorFlow Lite converter.

Next step is to proceed with the inference workflow as shown in Figure 4.11.

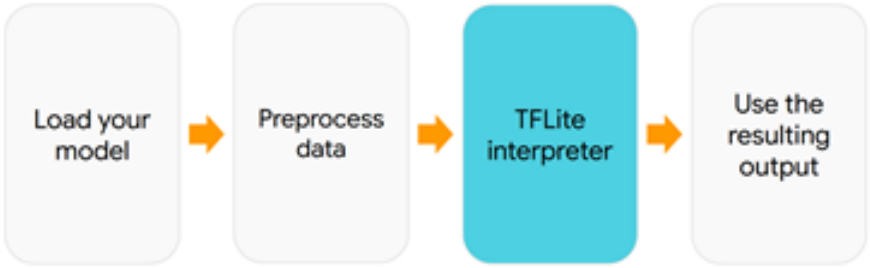


Figure 4.11: Workflow by using TensorFlow Lite interpreter.

This procedure generates a list of C++ 11 source code files that can be used in projects.

4.3.2 Keras

Keras is a Python-based deep learning API that runs on the TensorFlow ML platform. The core data structures of Keras are layers and models. The Sequential model, which consists of a linear stack of layers, is the simplest type of model. Keras has the following characteristics [76]:

- **Simple** – Keras reduces developer cognitive load, allowing you to concentrate on the most important aspects of the problem.

- **Flexible** – Keras follows the progressive disclosure of complexity's principle. It means simple workflows should be quick and easy; while, complicated ones should be accessible via a clear path that builds on previous knowledge.

- **Powerful** – Keras provides organizations and companies with industry-strength performance and scalability.

Keras supports convolutional and recurrent neural networks, as well as common utility layers like dropout, batch normalization, and pooling, in addition to standard neural networks.

4.3.3 Edge Impulse

Edge Impulse is a leading ML development framework for edge devices that is available for free to developers and enterprises. Edge Impulse's primary purpose is to assist software developers, engineers, and domain experts in solving real-world problems through ML on edge devices without having to be experts in this field [77].

An entire cycle of the developing ML model on the edge device is represented in Figure 4.12. An impulse is designed after the data is collected and the training set is in place. It takes raw data, cuts it into smaller windows, uses signal processing blocks to extract features, and then uses a learning block to classify new data. Signal processing blocks are used to make raw data easier to process by returning the same values for the same input while learning blocks learn from previous experiences.

Once all of the data have been processed, it's time to start training a neural network. Neural networks are algorithms that can learn to recognize patterns in their training data and are loosely modeled after the human brain.

The internal state of the neurons is gradually tweaked and optimized during training so that the network transforms its input in precisely the right ways to generate the correct output. This is accomplished by feeding a sample of training data into the network, determining how far the network's output differs from the correct response and adjusting the neurons' internal states to increase the likelihood of a correct answer being generated next time. This results in a trained network when repeated thousands of times.

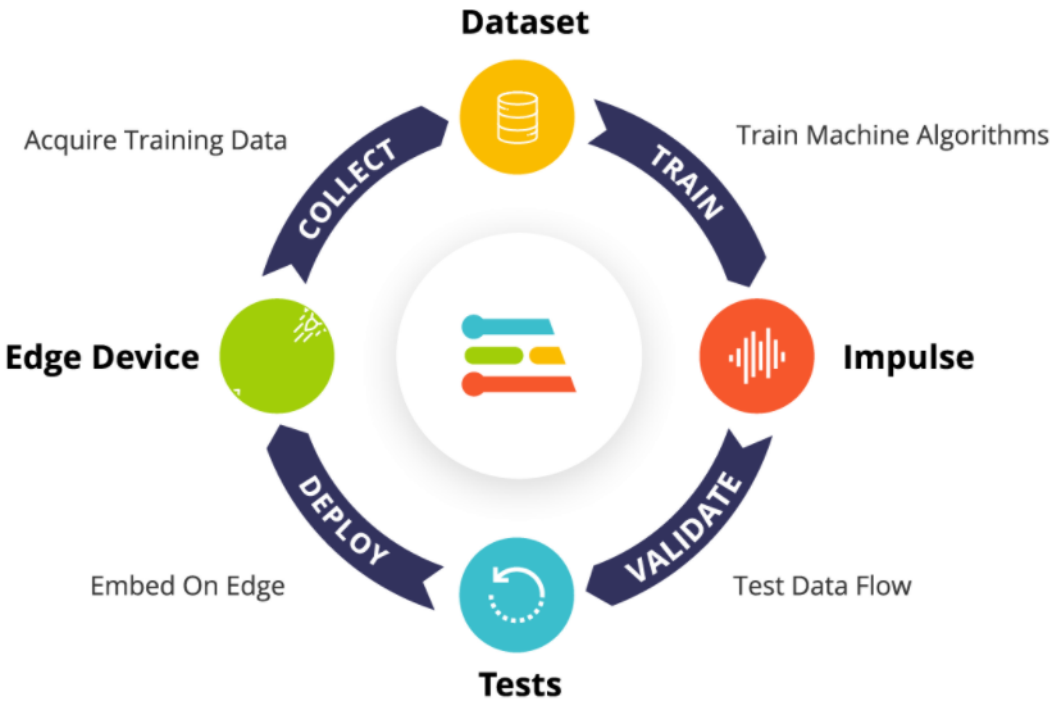


Figure 4.12: The entire cycle of the ML model development in Edge Impulse. This image was taken from [77].

The default neural network architecture given by Edge Impulse will work for the current project, but you can define your own. Data scientists' tools like TensorFlow and Keras can be used to import custom neural network code.

20% of the training data is set aside for validation at the start of the training. This means that rather than being used to train the model, it is now used to assess its performance.

If the number of cycles is too low, the network will not learn much from the training dataset; on the other hand, if the number of cycles is too high, the network will begin to memorize the training data and perform poorly on the unknown dataset. The last option is known as overfitting.

Before implementing the model in the real world, it's critical to test it on previously unseen data. This prevents the model from learning to overfit the training data, which is a common problem.

By making frequent changes to the model to boost its efficiency on the test dataset, the model may begin to overfit on the test dataset as well; to prevent this, new data should be added to the test dataset.

This model can be deployed back to the device once the impulse has been designed, trained, and verified. Edge Impulse can package up the entire impulse into a single C++ library that can be included in the embedded applications. It is possible to repeat this process until the best solution to real-world problems is discovered [77].

4.4 Zerynth Platform and Hardware

Zerynth is a platform designed to simplify and accelerate the development of IoT applications and offers developers, system integrators, and businesses a way to enable IoT for their products. This platform includes a software development environment that supports embedded Python and C programming languages, compatible hardware modules, an IoT device manager, and a cloud service to link everything.
The Zerynth SDK serves as a gateway to the Zerynth platform, which includes the Zerynth OS and Zerynth Device Manager (ZDM). Zerynth OS is a multithreaded Real-Time OS for 32-bit microcontrollers that supports Python and C programming and has a tiny footprint (low Flash and RAM requirements). ZDM is a device and data management service that makes it simple to register, organize, monitor, and remotely manage IoT devices at scale in a safe and scalable manner.

4.4.1 4ZeroBox

4ZeroBox is a versatile data acquisition unit intended for use as a machine-to-cloud interface in industrial machines. It's a modular hardware electronic unit with a powerful 32-bit microcontroller (240 MHz, 4MB Flash, 521KB SRAM) and numerous onboard features including a DIN-rail mountable case with industrial-grade sensor channels, support for Wi-Fi, Ethernet, LoRa, Bluetooth, CAN, RS485, RS232, SD Card, JTAG, I2C, SPI.

Two onboard MikroBus sockets enable the 4ZeroBox to be expanded with hundreds of MikroElektronika click boards.

Zerynth is the official programming framework for 4ZeroBox. It allows programming the 4ZeroBox applications in Python (or hybrid C/Python) and includes a compiler, a debugger, and an editor, alongside tutorials and example projects for an easy learning experience.

The 4ZeroBox enables the user to choose the most appropriate installation technique when adapting it to the particular industrial environment. There are three different installation modes:

- **PLC mode:** Acquisition of digital data from PLC via digital ports and protocols like Siemens S7, OPC UA, Ethernet/IP, etc.
- **RETROFITTING mode:** Acquisition in parallel to the PLC using the equipped sensors or installing new ones.
- **HYBRID mode:** The two options merged for obtaining the best from your machines.

The user can configure the hardware features of the 4ZeroBox as well as the required credentials, parameters, and libraries for programming the 4ZeroBox and connecting it to the ZDM. 4ZeroBox can be connected via USB to any PC or mini PC running Windows or Ubuntu to begin developing your application and receive debug information via RS232 using the same media. **TouchKey Click** has four capacitive pads powered by TTP224 is a touchpad detector integrated circuit. It operates with either a 3.3V or a 5V power supply (the response time is just 100mS at fast mode and 200mS at low power modes). Each pad on the board receives an interrupt signal: OUTA, OUTB, OUTC, and OUTD (in place of default mikroBUS™ RST, AN, PWM, and INT pins respectively). TouchKey click is used as a replacement for standard mechanical switches and buttons.

4.4.2 Zerynth OS

Zerynth OS is a multithreaded Real-Time Operating System that allows code reuse on a wide range of 32-bit microcontrollers while maintaining a limited footprint. The Zerynth Virtual Machine is the core of Zerynth. It was designed from scratch to bring Python to the embedded world with support for multi-thread and cross-board compatibility. Figure 4.13 represents the Zerynth OS's structure. Zerynth supports all of Python's high-level most used features, including modules, classes,

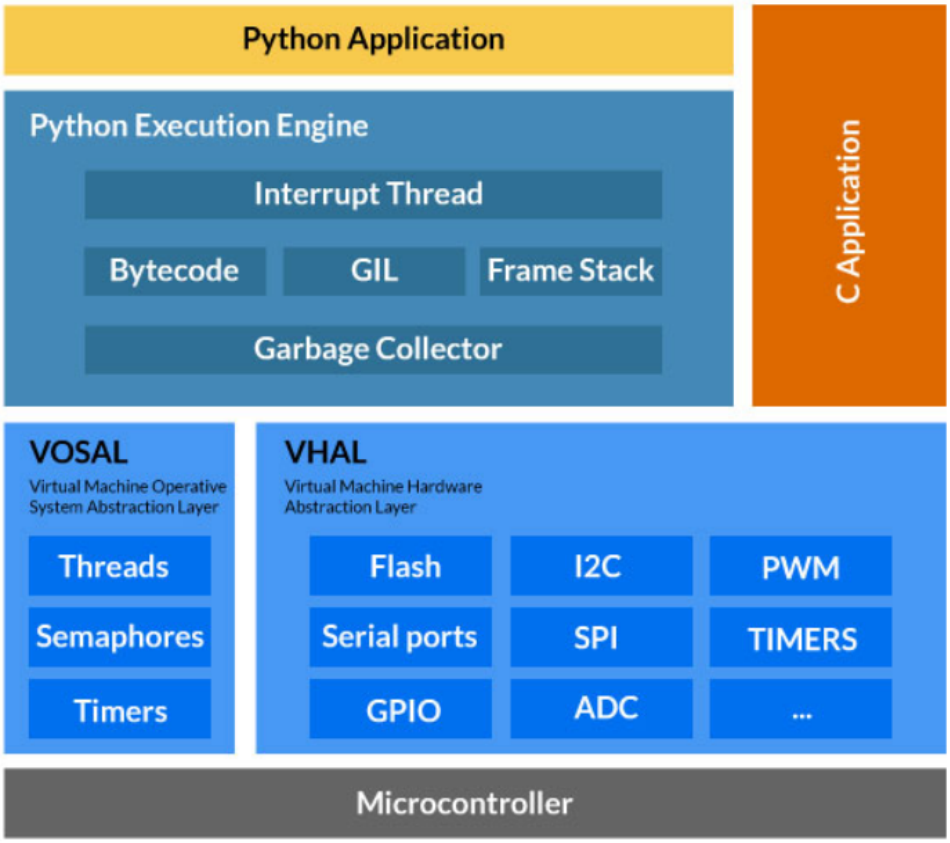


Figure 4.13: The structure of Zerynth OS. This image was taken from [78].

multithreading, callbacks, timers, and exceptions, as well as certain hardware-specific features such as interrupts, PWM, digital I/O, etc. Since the primary goal of Zerynth VM is to make Python usable in the IoT environment, some Python features were removed because they were too resource-intensive, and non-Python features were added because they were more functional in the embedded setting. Python Object size has been reduced, true multi-threading with priorities has been implemented, with each thread acting as an RTOS thread with its memory and priority. To carry sequences of 16 bits integers, new data structures such as shorts and short-array have been implemented.

4.4.3 Zerynth Device Manager

ZDM is a device and data management service that helps developers create scalable, secure, and reliable IoT solutions faster. The list of the key advantages of using ZDM is:

- Devices onboarding and provisioning enabled with gold-standard security practices.
- Devices lifecycle control allows complex tasks like remote procedure calls and firmware over-the-air updates (FOTA) through the REST of the APIs.
- Data Management for storage, aggregation, and feeding of data to the final IoT Application.
- Event management with the integrated alarming engine.
- Easy integration with third-party services and frontends, including all the major cloud service providers.

ZDM allows the management of IoT devices as well as the collection and aggregation of the produced data. Device management is critical in a wide range of manufacturing, consumer, and commercial applications. A list of ZDM's principal applications can be found here:

- **Industrial Applications** – ZDM allows a user to easily connect diverse industrial device fleets and manage, monitor, and troubleshoot connected devices deployed across multiple industrial industrial sites. It enables users

to monitor IoT devices to detect any unusual behaviors across their fleet and take appropriate mitigation measures, such as deploying a hotfix.

- **Consumer Applications** – The user can monitor the speed at which its firmware-over-the-air (OTA) updates are sent to consumer devices already in the field, and submit an update to ensure that devices are running the latest software and to push bug fixes or firmware updates to patch security vulnerabilities and boost the device's functionality.

- **Commercial Applications** – ZDM enables you to organize your devices into a hierarchical structure based on their purpose, security requirements, or any other category relevant to your business. It's possible to keep track of any connected device's status and take action on several devices at once.

The main features of ZDM are as follows:

- **Security** – ZDM handles the complicated and delicate provisioning process by ensuring each device's cryptographic identity, encrypting all communications via custom certificates, hardening the TLS stack, and speeding it up where possible.

- **Device-independent** – Due to its extensive software base, ZDM can work with any device, including products powered by Zerynth OS and related libraries, as well as CPU-based SBC, computers, software applications, and the ZDM-Client Python library.

- **Data collection** – ZDM distinguishes between user devices and the data they generate. Devices can send data and events to the ZDM, which are then tagged, synchronized, stored, and sent to other services via Webhook, MQTT, REST API, and a variety of other connectors.

- **Evenets collection** – Data and events are handled independently by ZDM. Data are values provided by connected sensors, while events are alerts sent by the device to activate specific logic functions.

- **Device Control** – ZDM manages all aspects of the device's life cycle, including firmware execution, contacting the device and retrieving job execution status notifications, and performing diagnostic queries on devices.

- **FOTA update** – FOTA updates are particular category of jobs that are available by default in the ZDM device library for Zerynth OS. FOTA jobs are created to ensure that firmware updates are secure.

4.4.4 Zerynth Module ZM1

The Zerynth Module 1 is a dedicated customized version of the ESP32-WROOM-32SE Module. The CPU clock frequency is 80 - 120 MHz, with an embedded 16 MB of SPI Flash memory [79].

ESP-32 is designed to provide the best power and RF performance in a wide range of applications and power scenarios, demonstrating robustness, flexibility, and reliability. With about 20 external modules, ESP32 is a fully integrated solution for Wi-Fi and Bluetooth IoT applications. ESP32 integrates an antenna switch, power amplifier, low-noise receive amplifier, filters, and power management modules [80]. The following hardware is needed for the development of ESP-32 applications:

- An ESP32 board
- USB cable - USB A/ micro USB B
- Computer running Windows, Linux or macOS

It is possible to choose to either download and install the following software:

- Toolchain to compile code for ESP32
- Build tools - CMake and Ninja to build a full Application for ESP32
- ESP-IDF that essentially contains Application Programming Interface (API) (software libraries and source code) for ESP32 and scripts to operate the Toolchain

or to get through the onboarding process using the Eclipse Plugin or VS Code Extension for IDE (integrated development environment) [80]. Figure 4.14 depicts the entire developing applications process for the ESP32.

4.4.5 Grafana

Grafana is a free and open-source data visualization, monitoring, and analysis platform. It can be used to analyze data and monitor long-term patterns and

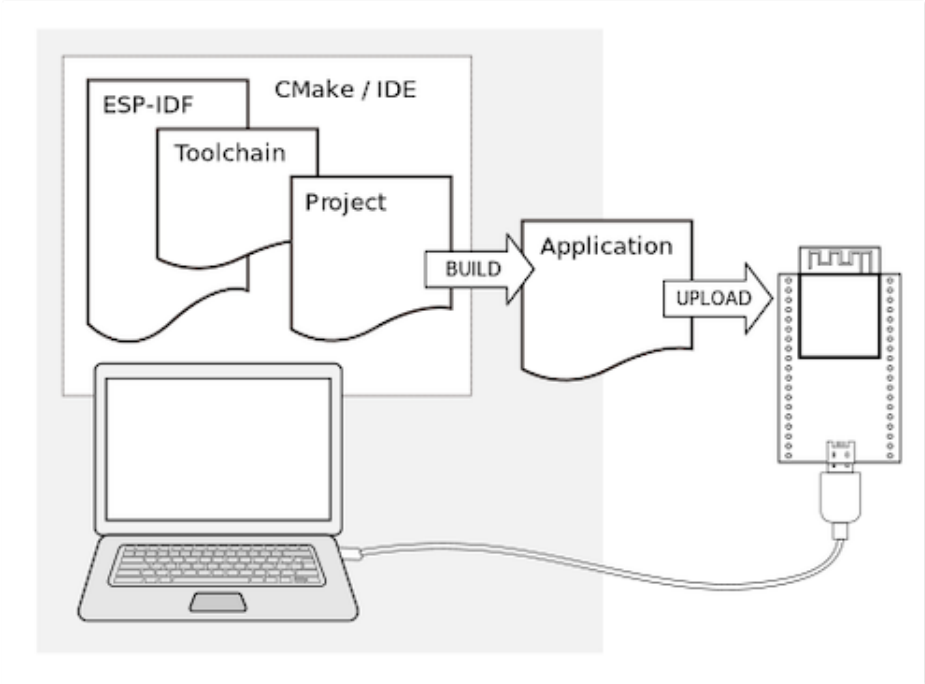


Figure 4.14: Development of applications for ESP32. This image was taken from [80].

trends, and it can be accessed through a series of dynamic and reusable dashboards that users build and share with business and technical teams.

Grafana offers a wide variety of visualization tools, including heatmaps, graphs, and histograms. Grafana offers a variety of visualization options to assist you in comprehending your data. In the same graph, different data sources can be combined. It enables you to query, visualize, alert on, and understand your metrics regardless of their location.

Chapter 5

Home Appliances Use Case

In this use case, we implemented a ML model aimed at understanding the state of the power consumption profile of home appliances.

Domestic appliances that are deployed in our everyday life have different duty cycles and are possible to categorize based on the operational state or working principle of home appliances into the following categories: two-state appliances (have only two states, either ON or OFF at a given time, such as a toaster, table lamp, etc.), multi-state appliances (have more than two operating states, ON, OFF and standby modes, such as a washing machine, dishwasher and stove burner), continuously - varying power appliances (don't consume constant power, such as light dimmers and power tools.), permanent consumer appliances (remain permanently ON, such as smoke detectors and telephone sets.). These variations are due to the different power consumption of each device and each performance mode. Data acquisition is the process of collecting aggregated load measurements from household appliances at a specified time interval to distinguish distinct load trends. When you turn an appliance on and off, the power level changes and a new steady power level begins. This is characterized by a change in power [81].

5.1 Data Acquisition

This work focuses on data collected from home appliances by the Zerynth company in Pisa, with the 4ZeroBox serving as a data acquisition tool plugged into the house. The Grafana visualization tool can be used to visualize real-time power

consumption. Grafana is connected with ZDM, and then the ZDM API is used to retrieve 4ZeroBox data. This is the system that allows data collection for the given use case. The 4ZeroBox has some sensors attached to it to obtain environmental values and power consumption measurements. The collected data is tagged with the terms "env" and "power": "env" refers to data about environmental values such as humidity, pressure, and temperature, and "power" refers to data about power consumption. The data is collected every second and sent to the cloud every two minutes. The data gathered for this project covers the period from October 8, 2020, to December 16, 2020.

5.2 Data Overview

The power consumption csv file contains data corresponding to the "power" tag and consists of eight columns:

- **timestamp** – is the timestamp of when device sent data to the cloud.
- **device ID** – the device's id, the same for all recorded samples.
- **device Name** – the device's name, the same for all recorded samples.
- **pow1** – is the overall power consumption.
- **pow2** – is the power consumption of home appliances (dish washer, microwave, bathroom heater, air dryer, coffee machine, oven).
- **pow3** – is the power consumption of garage appliances (washing machine, dryer, waste water pump, freezer)
- **ts** – is a timestamp in different format from the one in column "Timestamp".
- **tag** – tagged as a "power" in power consumption file.

Figure 5.1 represents the columns of power consumption csv file.

Timestamp	Device id	Device name	Tag	pow1	pow2	pow3	ts
2020-10-29T20:52:00+00:00	dev-5bqxb2br4pku	device_power_consumption	power	597.862605	549.372108	88.852530	1.604005e+09
2020-10-29T20:54:00+00:00	dev-5bqxb2br4pku	device_power_consumption	power	595.177162	548.279794	88.990615	1.604005e+09
2020-10-29T20:58:00+00:00	dev-5bqxb2br4pku	device_power_consumption	power	571.912581	546.022620	56.224551	1.604005e+09
2020-10-29T21:00:00+00:00	dev-5bqxb2br4pku	device_power_consumption	power	572.403343	548.420867	56.131955	1.604005e+09
2020-10-29T21:02:00+00:00	dev-5bqxb2br4pku	device_power_consumption	power	570.977357	546.217072	55.678232	1.604005e+09
2020-10-29T21:04:00+00:00	dev-5bqxb2br4pku	device_power_consumption	power	569.616190	544.272548	57.428305	1.604005e+09

Figure 5.1: The power consumption file's content.

The environmental measurement csv file contains data corresponding to the "env" tag and consists of eight columns such as timestamp, device ID, device Name, ts and tag as the ones in the power consumption file. The differences are the following columns:

- **hum** – value of humidity in %H.
- **pres** – value of pressure in hPa.
- **temp** – value of temperature in °C.

Figure 5.2 represents the columns of environmental measurement csv file.

Timestamp	Device id	Device name	Tag	hum	pres	temp	ts
2020-10-23T20:42:00+00:00	dev-5bqxb2br4pku	device_power_consumption	env	57.308464	1019.642667	25.011333	1.603486e+09
2020-10-23T20:44:00+00:00	dev-5bqxb2br4pku	device_power_consumption	env	57.284574	1019.538403	25.010756	1.603486e+09
2020-10-23T20:46:00+00:00	dev-5bqxb2br4pku	device_power_consumption	env	57.265104	1019.446417	25.011750	1.603486e+09
2020-10-23T20:48:00+00:00	dev-5bqxb2br4pku	device_power_consumption	env	57.265999	1019.438667	25.008667	1.603486e+09
2020-10-23T20:58:00+00:00	dev-5bqxb2br4pku	device_power_consumption	env	57.209101	1019.581651	25.023119	1.603487e+09
2020-10-23T21:00:00+00:00	dev-5bqxb2br4pku	device_power_consumption	env	57.251994	1019.628000	25.021000	1.603487e+09

Figure 5.2: The environmental measurement file's content.

5.3 Processing data

In most cases, the dataset contains noise and missing feature values and requires a pre-processing stage. To deal with missing data, various researchers have proposed a variety of techniques. Hodge & Austin [82] conducted a study of current methods for detecting outliers (noise). They distinguished the advantages and disadvantages of the outlier identification methods. There is no single universal outlier detection approach, so the developer should select an algorithm that is suitable for their dataset. An outlier is an observation that is different from the other observations. Outliers may occur as a result of a reading error, a machine fault, or a manual error. The presence of outliers can result in biased or invalid findings, which can affect the analysis and subsequent processing.

5.3.1 Data Cleaning

Cleaning data is the method of removing outliers or modifying data that is incorrect, meaningless, or has anomalous values to prepare it for analysis. The Interquartile Range Method (**IQR**) was used to identify the outliers. The difference between the 75th and 25th percentiles is used to determine the IRQ, which is a statistical measure of dispersion. "In statistics, a percentile is a score below which a given percentage of scores in its frequency distribution falls or a score at or below which a given percentage falls." [83].

The boxplot is a standardized graphical representation of the distribution of data based on a five-number summary: the minimum (Q_0) - the lowest data point excluding any outliers, the maximum (Q_4) - the largest data point excluding any outliers, the sample median (Q_2) - the middle value of the dataset, the first and third quartiles. It is used to detect any anomaly or outlier and to identify data distribution. They can be drawn either horizontally or vertically. The interquartile range is a key component of the boxplot's representation (is the distance between the upper and lower quartiles). The 25th percentile is the first quartile also known as the lower quartile of the median of the lower half of the dataset (Q_1) and the 75th percentile is the third quartile (Q_3) also known as the upper quartile or the median of the upper half of the dataset. The IRQ is calculated in the following way:

$$\text{IQR} = Q_3 - Q_1 \tag{5.1}$$

The box plots have lines extending from the boxes (whiskers) indicating variability outside the upper and lower quartiles. The vertical lines above and below the rectangle are called whiskers. They span from the first data point greater than Q1 - 1.5*IQR to the last data point less than Q3 + 1.5*IQR. All other observed points outside of these limits are outliers. Figure 5.3 represents the box plot.

Data cleaning was performed before using the dataset for training the ML model. Figure 5.4a represents the box plot of "pow2" column data, which is the power consumption of home appliances before using of the IRQ method (outliers are shown by circles in the figure). Figure 5.4b represents the box plot after using the IRQ method.

Out of 41039 number of observations of the energy consumption of home appliances, 4251 were outliers. The data should be normalized or standardized to

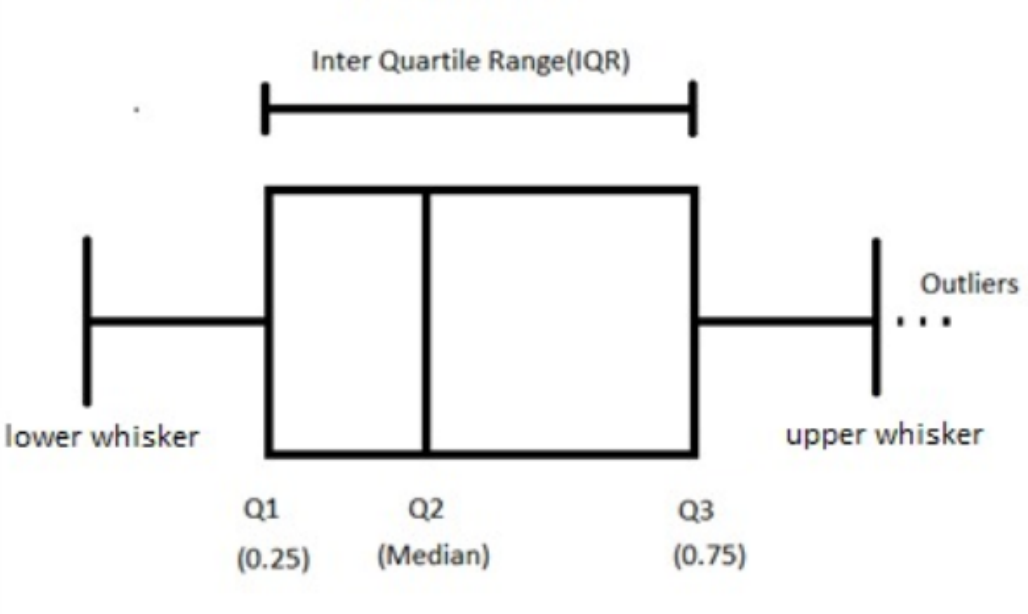
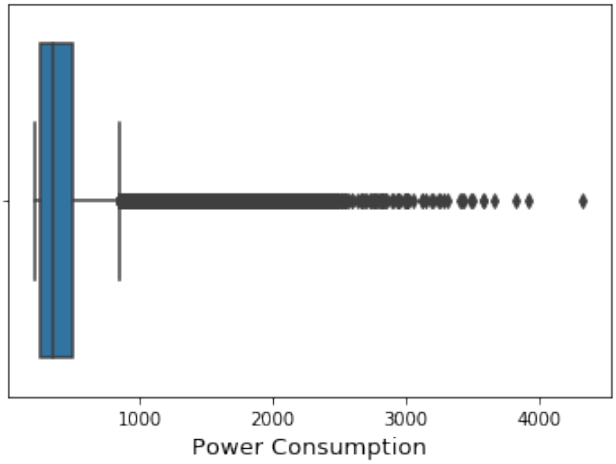
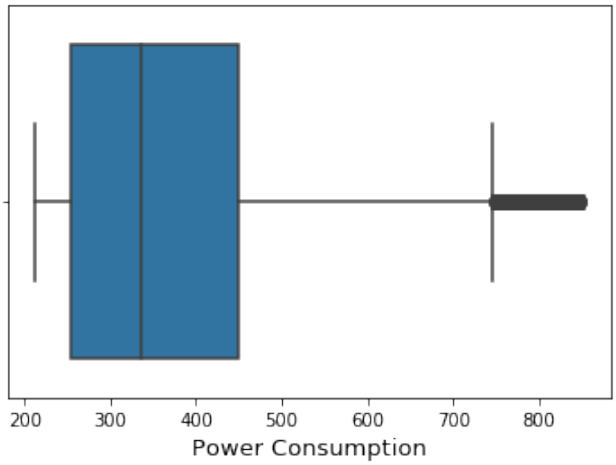


Figure 5.3: The box plot.



(a) The box plot with outliers



(b) The box plot without outliers

Figure 5.4: The box plots of home appliances dataset with and without outliers.

avoid reliance on the measurement units chosen. Normalization is a technique that is frequently used in the data preparation process of ML algorithms. It is a scaling technique to change the values of numeric columns in the dataset to use a common scale, so values are shifted or rescaled so that all values are within the range of 0 and 1. Normalizing data samples means assigning equal weight to all attributes. It helps avoid attributes with initially wide ranges from outweighing attributes with initially smaller ranges by using distance-based methods.

Data normalization can be accomplished in a variety of ways. MinMaxScaler was chosen as the technique to normalize the data for this work. The Min-Max method transforms features by scaling them to the [0,1] or [-1, 1] range. The composition of the data determines the target range. The formula for a min-max of [0,1] is calculated as :

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{5.2}$$

where x is an original value, x' is the normalized one, min, max equals to the feature range. This min-max transformation is used as an alternative to zero mean, unit variance scaling. It subtracts the minimum value from the feature and then divides it by the range (difference between the original minimum and the original maximum). It is a good practice to use other scaling techniques with the MinMaxScaler:

- **Fit the scaler using available training data.** – the usage of training data to calculate the minimum and maximum observable values. This is done by using the $fit()$ function.
- **Apply the scale to training data.** – usage of the normalized data to train ML model. This is done by using the $transform()$ function.
- **Apply the scale to data going forward.** – prepare new data to make future predictions.

MinMaxScaler doesn't change meaningfully the information embedded in the original data. Prior to implementing the K-Means clustering algorithm, the data was normalized using MinMaxScaler() from the sklearn.preprocessing package [84], which contains several utility functions and transformer classes available in sklearn to convert raw features into something more suitable for the downstream estimator's representation.

5.4 Results

5.4.1 K-Means Clustering Algorithm

The sklearn implementation of the K-means clustering algorithm has been performed on a dataset of home appliances' power consumption. The algorithm is described in the Methodology section. This algorithm clusters data by dividing samples into "n" equal-variance groups and minimizing the SSE. Within clusters, the less variance there is, the more homogeneous the data points are within the same cluster.

The elbow method was described in the Methodology section as a method that gives us an idea of what a good "k" number of clusters would be, based on the SSE between data points and their assigned clusters' centroids. It is possible to determine the correct number of clusters after evaluation of SSE for different values of "k" and usage of the kneed locator library to decide which part of the graph is an elbow. One should choose a few clusters so that adding another cluster does not improve the overall SSE. According to the result of elbow method in Figure 5.5, the number of clusters that should be used for the K-means model is five.

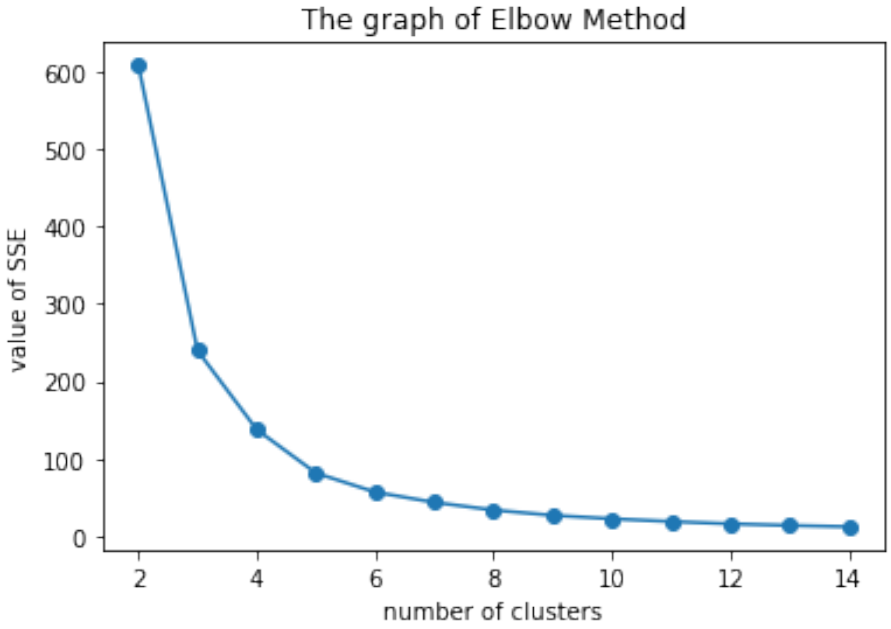


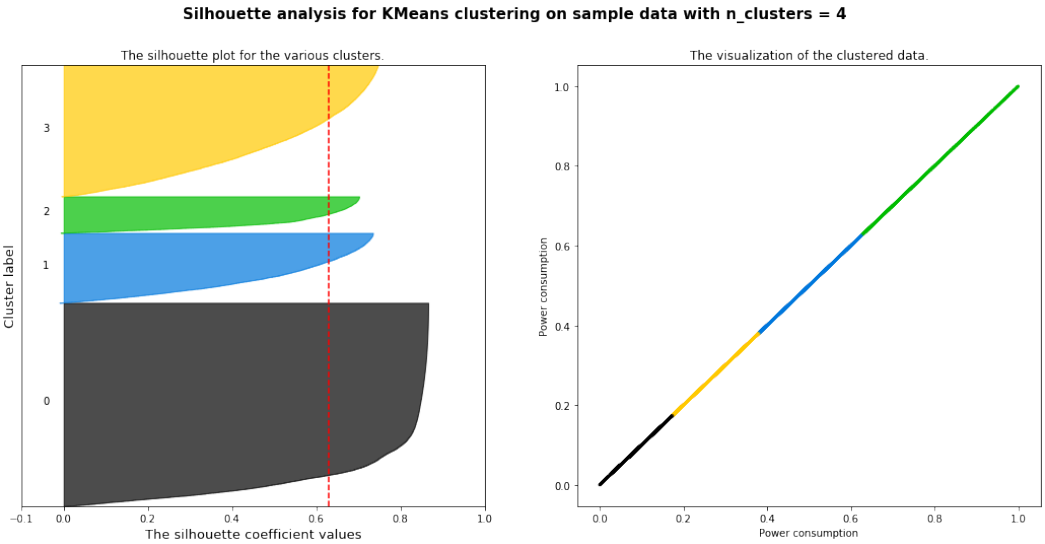
Figure 5.5: The result of the elbow method for power consumption of home appliances.

A silhouette index is the second metric that provide some insight into the "k" number of clusters. It is calculated for each sample of different clusters and is used to evaluate how many samples are close to each other (degree of separation between clusters).

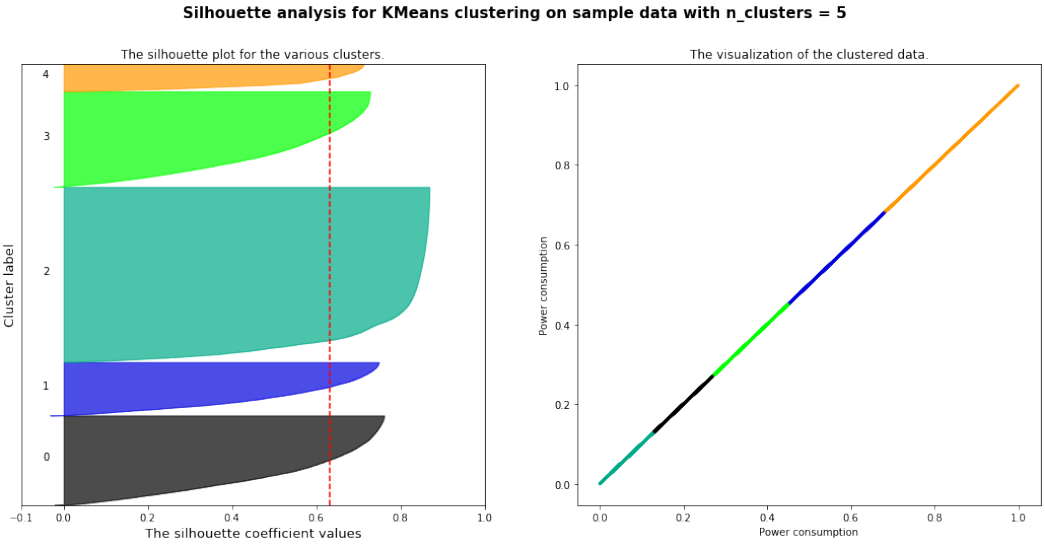
More in detail, a silhouette method discussed in the Methodology section. The coefficient of silhouette takes values in the interval [-1,1]; "0" means that clusters are overlapping, "1" means that the clusters are very dense and nicely separated, "-1" means that data belonging to clusters may be wrong/incorrect. Figure 5.6 represents the values of silhoutte per number of clusters after the implementation of K-Means algorithm.
Figure 5.6 shows the average silhouette score is 0.62 (n_cluster=4) and 0.64 (n_cluster=5). It is worth pointing out that when n_cluster = 5, the average silhouette score is high. Figure 5.6b shows that cluster "0" has the greatest amount of the samples.

After considering the elbow method and silhouette score analysis results, the number of clusters for the K-Means algorithm was set at five. Figure 5.7 represents the output results of K-Means Clustering algorithm choosing five as number of clusters. The output of the K-Means algorithm was three columns: timestamp, power consumption of home appliances, and a cluster corresponding to power consumption. Given that we have a dish washer, a microwave, a bathroom heater, an air dryer, a coffee machine, and an oven, the number of five clusters may be a good choice. We were unable to identify the power consumption of each appliance because the data was unlabeled, but we were able to identify five clusters (five power consumption trends).

The biggest disadvantage of the K-Means algorithm is the requirement of a predefined number of clusters. This might be hard to know beforehand. Another disadvantage of K-Means is that it is prone to having outliers.



(a) n_clusters = 4



(b) n_clusters = 5

Figure 5.6: The plot of average silhouette value over number of clusters for power consumption of home appliances.

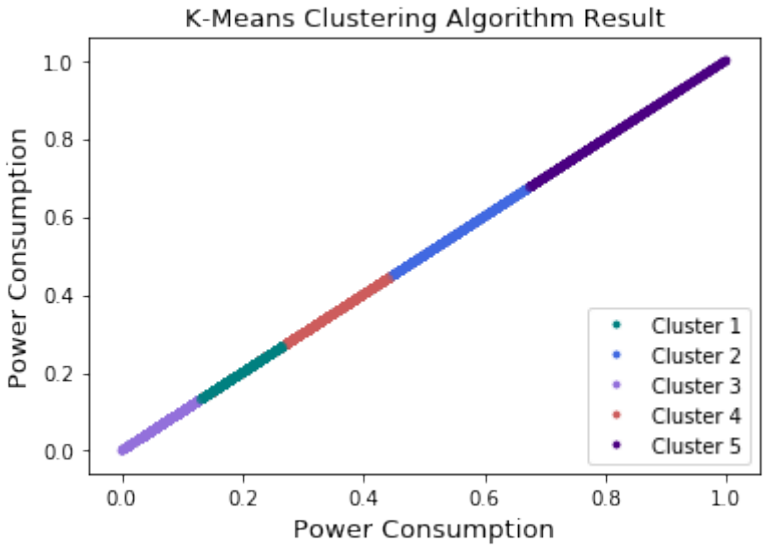


Figure 5.7: The plot of K-Means Algorithm Results distributing power consumption data over five clusters.

5.4.2 Training Machine Learning Model

Scikit-Learn includes several functions to split the dataset into multiple subsets in different ways. The dataset of home appliance power consumption is divided into two parts: 70% for training and evaluating the model, and 30% for testing.

The next step is to build data models and estimate their accuracy on unseen (test) data using statistical methods. The stratified 10-fold cross-validation is used to estimate model accuracy. The dataset is split into ten parts, with nine being trained and one being tested and the process is repeated for all combinations of train-test splits. The K-fold cross-validation was addressed in the Methodology section. Stratified means that each dataset's split aims to have the same distribution of examples by class as the entire training dataset. The metric of "accuracy" was used to evaluate each model. This is a percentage calculated by dividing the number of correctly estimated instances by the total number of instances in the dataset and multiplying it by 100.

Five different algorithms were chosen as a mixture of simple linear (LR and LDA) and nonlinear (DT, KNN, and SVM) algorithms to train the ML model. A brief description of these algorithms is given in section 4.3.2. The most important aspect that has to be considered during an analysis of these algorithms is their

accuracy. Table 5.1 gives information about the accuracy estimations for each model.

Table 5.1: The accuracy on different algorithms.

Algorithm	**Accuracy**
Logistic Regression	85.3830%
Linear Discriminant Analysis	98.2873%
Decision Tree	99.9961%
K-Nearest Neighbor	99.9845%
Support Vector Machine	99.9728%

Logistic Regression has the lowest estimated accuracy rate, at about 0.85 or 85%. Since each algorithm was evaluated ten times, there is a set of accuracy measurements for each algorithm (via 10 fold-cross validation). In terms of performance, the output of the five ML algorithms that were evaluated is not significantly different. The effective performances were obtained from the K-Nearest Neighbor and Decision Tree algorithms.

The Grid Search method was used to tune the parameters of the algorithms for optimal efficiency. It is a process for finding the optimal hyperparameters for a model, resulting in the "accurate" predictions, by searching through a manually specified subset of the algorithm's hyperparameters space. It takes all possible combinations of those parameters, evaluates the results using cross-validation and the scoring metric that is provided. The output of the Grid Search is the best parameters for the dataset.

The Grid Search was used to improve the models' accuracy. Table 5.2 represents the results after Grid Search implementation on the five above-mentioned algorithms, as well as the best parameters for each algorithm and the mean cross-validation accuracy using the best parameters.
The accuracy of the Logistic Regression Algorithm has been significantly improved from 85% to 97% due to the usage of the algorithm with the best parameters. Taking into consideration Table 5.1 and Table 5.2 it is worth pointing out that K-Nearest Neighbor and Decision Tree are the ones that are more "accurate".

Figure 5.8 shows the results of y_test (true class label of the test dataset) and y_predicted (predicted class label). From Figure 5.8a and Figure 5.8b it is possible to notice that y_test and y_predicted clusters are almost the same, given that

Table 5.2: The accuracy of different algorithms.

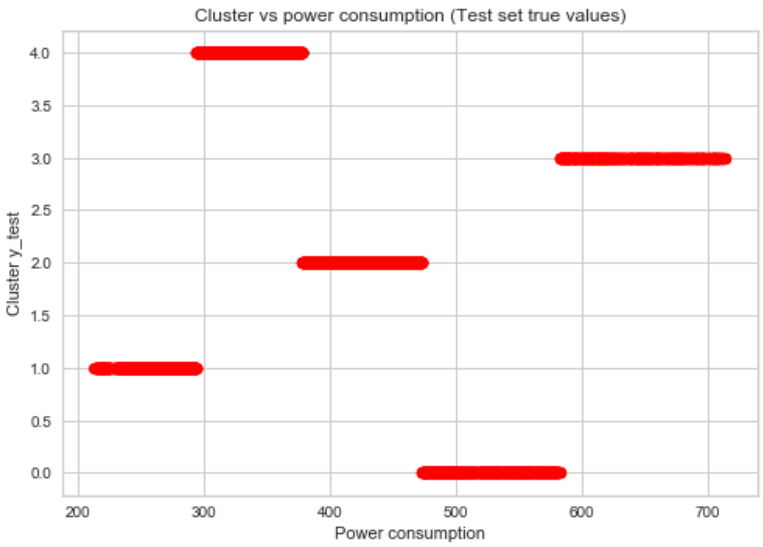
Algorithm	**Best Parameters**	**Accuracy with Best Parameters**	**Mean Cross Validation Accuracy**
LR	'class_weight':'balanced','solver':'lbfgs'	97.2184%	85.3713%
LDA	'solver':'svd'	98.3419%	98.2874%
KNN	'algorithm':'auto','leaf_size':5, 'n_neighbors':2, 'weights': 'distance'	99.9819%	99.9845%
SVM	'class_weight':None, 'degree':3, 'shrinking': False	99.9456%	99.9767%
DT	'class_weight':'balanced','presort':True, 'min_samples_split':2, 'criterion':'entropy', 'splitter':'random'	99.9456%	99.9961%

the accuracy of the K-Nearest Neighbor Algorithm with the best parameters was 99.98%.

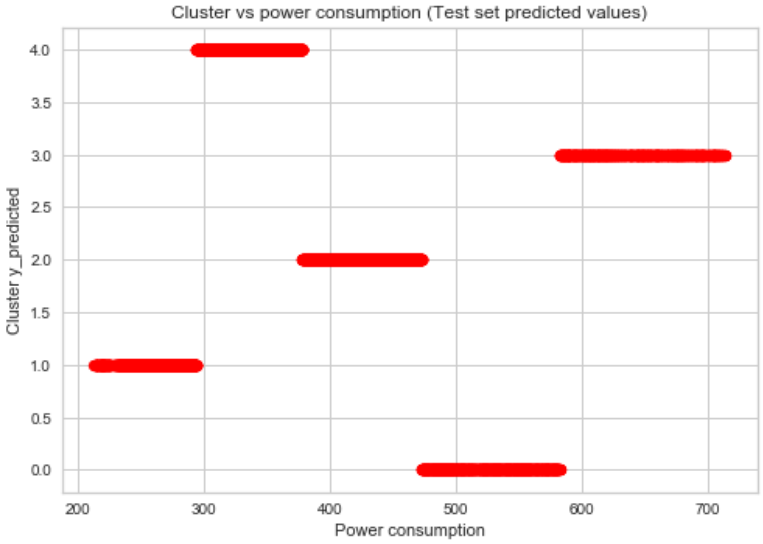
A better way to evaluate the performance of a classifier is to consider the confusion matrix, precision, recall, and f1 score. The confusion_matrix and classification_report methods of the sklearn.metrics can be used to calculate these metrics. The confusion matrix is a specific table layout that has different combinations of predicted and actual values. The diagonal elements show the number of points for which the predicted label is equal to the true label, while off-diagonal elements are the ones that are mislabeled by the classifier. Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class. The higher the values on the confusion matrix's diagonal, the better, indicating a large number of accurate predictions.

Figure 5.9 represents the confusion matrix's heatmap of the K-Nearest Neighbor classifier. From Figure 5.9 we can see that only one sample out of 11.035 was labeled as "3", instead of being labeled as "1". Given that only one sample was incorrectly predicted, the K-Nearest Neighbor classifier's accuracy of 99.98% is reasonable.

The classification report is run on the test dataset to evaluate how well the model is doing on the new data. It builds a text report with a representation of



(a) y_test true clusters



(b) y_predicted clusters

Figure 5.8: The plot of y_test true clusters vs y_predicted clusters for power consumption of home appliances.

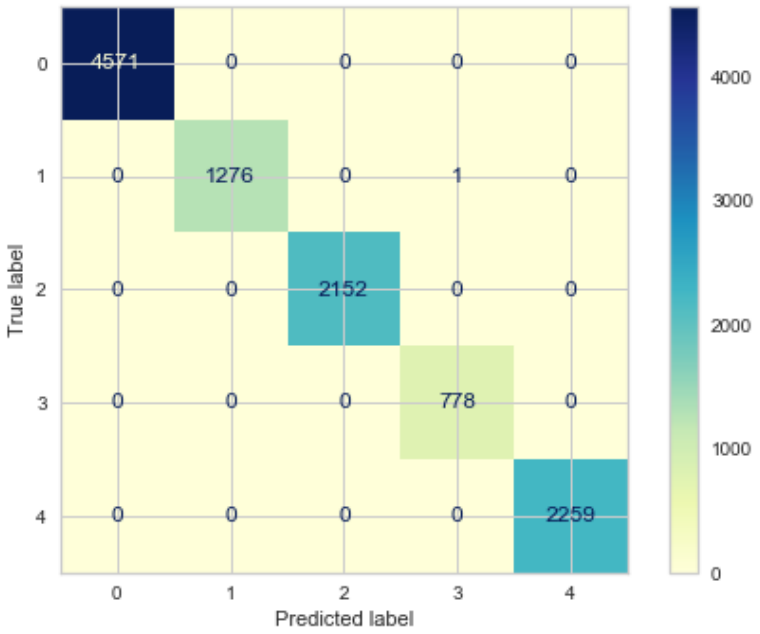


Figure 5.9: The confusion matrix of K-Nearest Neighbor classifier.

the main classification metrics for each class.

Figure 5.10 represents the classification report for the K-Nearest Neighbor classifier; the value of precision for class "3" and the value of recall for class "1" is 0.999 due to the one wrong identified sample from the dataset. All the remaining values for the other classes are equal to one since they were predicted correctly.

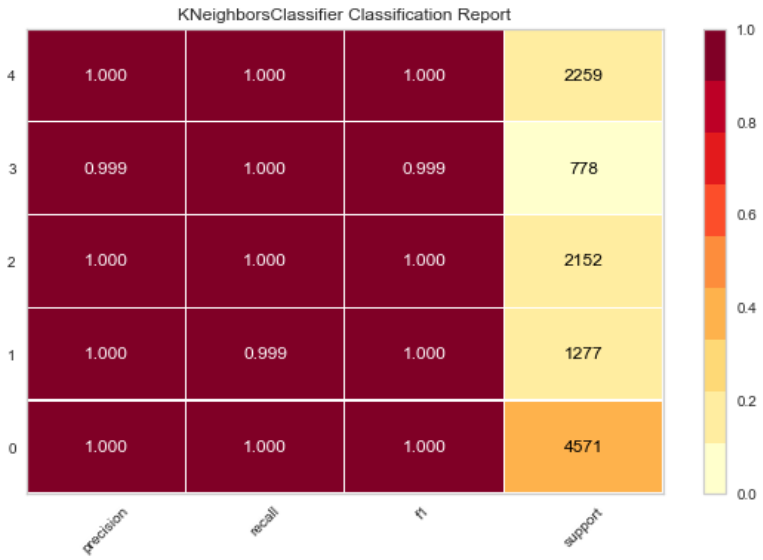


Figure 5.10: The classification report of K-Nearest Neighbor classifier.

There are four ways to check if the predictions are right or wrong:

- **True Negative (TN)** – the actual class is negative and estimated negative.
- **True Positive (TP)** – the actual class is positive and estimated positive.
- **False Negative (FN)** – the actual class is positive but estimated negative.
- **False Positive (FP)** – the actual class is negative but estimated positive.

Using this terminology the metrics are defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.3}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.4}$$

$$\text{F1 score} = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \tag{5.5}$$

- **Precision** – is a measure of a classifier's exactness.
- **Recall** – is the measure of the model to identify True Positives correctly.
- **F1 score** – is a weighted harmonic mean of precision and recall. A good F1-score is an indicative of a good precision and a good recall value.

- **Support** – is the number of actual occurrences of the class in the specified dataset.

As a result of the conducted tests, comparing the performances of all five algorithms, namely Logistic Regression, Linear Discriminant Analysis, Decision Tree, K-Nearest Neighbor and SVM algorithms, the K-Nearest Neighbor and the Decision Tree are better in their accuracy. Looking at the research from a wider perspective, the accuracy of all five algorithms surpassed 85%. The Grid Search is also done to improve the accuracy of the algorithms at a global level.

Chapter 6

Coffee Machine Use Case

In this chapter, we are going to discuss the implementation of the coffee machine use case. The goal is to build a DL model using Neural Networks to predict what the label of the pressed button is. We monitored a coffee machine for several days and we'd like to know what it's doing depending on the amount of power it consumes. A user always finds the coffee-maker ready-to-use because the appliance is permanently working. It is necessary to keep track of every coffee machine's state to have an idea of the coffee machine's power consumption profile.

6.1 Data Acquisition

The data acquisition stage is used to analyze data from different perspectives and to extract the hidden patterns from the dataset. The current experimental data is the real-time consumption profile of coffee machine inside the Zerynth company. We acquired coffee-maker power consumption data between the 10th of March and the 27th of March , 2021. The data was collected at a sampling frequency of four samples per second, and the acquisition timestamps were available.

The following components made up the data acquisition stage's system architecture:

1. 4ZeroBox, a data acquisition device for IoT, has been installed on a coffee machine in a non-invasive way to acquire power consumption data of the different working states of the coffee machine. On each event, the 4ZeroBox board sends information including: timestamp, the power consumed in Wh, the state in which

the machine is set, and so forth.

2. To trigger a new event, a TouchKey click with four capacitive pads has been placed into the 4ZeroBox's mikroBUS socket.

3. The 4ZeroBox was connected through Ethernet cable to LAN of the local servers of the Zerynth company. ZDM was running on the local servers.

4. ZDM was running on Beelink Mini PC and MQTT was used as a communication protocol to send samples to ZDM. It allows the management of IoT devices and collects and aggregates the data produced.

Because the coffee machine is not utilized on weekends, the data collection procedure mentioned above is used to estimate the coffee machine's operating mode during working days.

6.2 Data Overview

In the first step of the use case, power usage, timestamp, and pressed button were all taken into account. There are five columns in the dataset for the coffee machine's state analysis.The data acquisition output csv file is organized as follows:

- **timestamp_in** – the unit timestamp of when the device sent data to ZDM.
- **b** – indicates the pressed button.
  - "0" – no button pressed
  - "1" – button "A" pressed
  - "2" – button "B" pressed
  - "4" – button "C" pressed
  - "8" – button "D" pressed
  - "3,5,7,. . . " – "binary" combinations of two or more pressed button.
- **p** – instantaneous power consumption.
- **s** – seconds part of the unix timestamp recorded by the device.
- **us** – microseconds part of the unix timestamp recorded by the device.

Figure 6.1 gives an idea of the collected data, representing only first five rows of the acquired dataset.

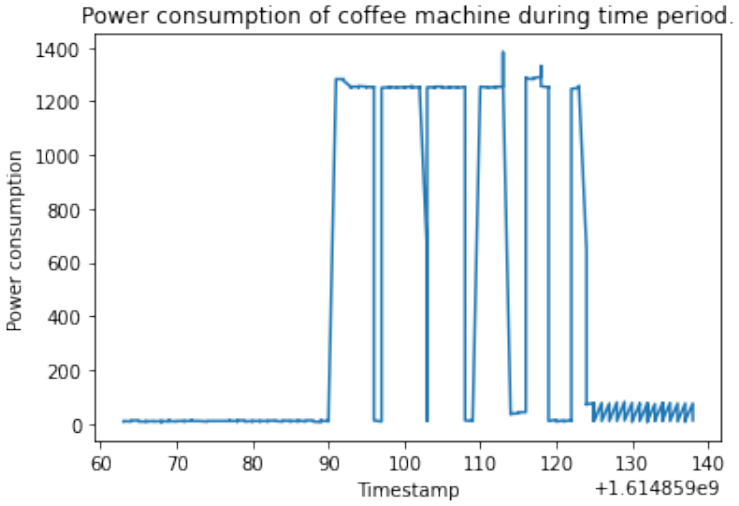
	timestamp_in	b	p	s	us
0	2021-03-04 11:55:39+00	0	8.13	1614858938	528498
1	2021-03-04 11:55:39+00	0	8.13	1614858938	778434
2	2021-03-04 11:55:39+00	0	8.13	1614858939	28418
3	2021-03-04 11:55:39+00	0	10.16	1614858939	278424
4	2021-03-04 11:55:40+00	0	8.13	1614858939	528517

Figure 6.1: The sample of dataset collected from coffee machine use case.

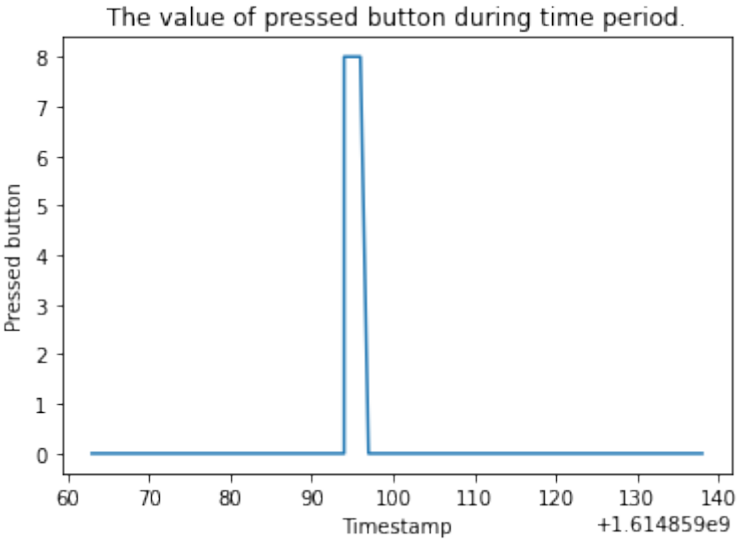
6.3 Data Analysis

To detect the start of an event, we looked at the value of the pressed button. In particular, when a new event starts, the value of the pushed button changes. From the analysis, we immediately noticed the unreliability of the pressed button, as there was a need to click simultaneously on the button on TouchKey, which is positioned on top of 4ZeroBox, and the coffee machine itself, so it was strictly dependent on human-machine interaction. It happens when someone touched a button, but did not press on it.

Figure 6.2 represents the coffee machine's power consumption and the value of the pressed button over time when 4ZeroBox sends data to ZDM. There was strange behavior when focusing on the power consumption data and looking at Figure 6.2a. Due to the presence of small noise readings, it is possible to figure out brief variations and troughs in power consumption over a few seconds. When the coffee machine's button is pressed to drink coffee, the value of the pressed button changes from "0" to "8" as shown in Figure 6.2b. However, the pressed button's value should remain "8" until the completion of the coffee-making process, when power consumption will again be close to zero. It is possible to see how noisy power consumption measurements affect the value of the pressed button.



(a) value of power consumption



(b) value of pressed button

Figure 6.2: The plot of coffee machine power consumption and corresponding pressed button values during period of time.

6.3.1 Peak Detection

Before starting to train the DL model, it is needed to identify and analyze peaks in the time-series of power consumption data. Peaks are useful topological features of a time-series because they highlight significant events such as a sudden increase in power use.

When the button is pressed, the coffee-making process starts, so the value of power consumption increases and lasts for a while. The value of power consumption is reduced when the coffee preparation is completed.

After the peaks have been identified, the analysis of these peaks entails many tasks, including determining the periodicity of the peaks, forecasting the time of recurrence, and estimating the value of the next peak. The important question in this use case is how to detect a peak given the noise in the data and the fact that peaks occur at different amplitudes and scales.

We began by plotting the Figure 6.3 that gives an overview of the time-series dataset, including the pushed button labels. 686 of the 886 samples in Figure 6.3 have the label 0 (grey color), indicating that no button was pressed. It's worth pointing out that the power consumption value is close to zero for a vast amount of time given that the button is not pressed.

We used the pandas dataframe.rolling() function for peak detection in time-series data, which helps calculate new values throughout each row of the dataframe. The concept of rolling window calculation is commonly applied to time-series data. A window of size "k" denotes the "k" number of observations used to calculate statistics at one time. After an empirical evaluation, we chose a window length of 75 and a power threshold value of 300. Then, we rolled maximum and minimum power consumption values with a window length of 75 to find the maximum and minimum in each window. We generated a moving windows pandas series, which is a one-dimensional labeled array, representing the difference between the maximum and minimum power consumption within a window size of 75.

We set a power threshold value of 300 and compared the moving window array values to the threshold one. If the value of a moving window array is below the power threshold, the value of the moving window is set to 0. If it is above the power threshold, the new value of moving windows is set to 1. Figure 6.4 represents the value of the moving window after applying the power consumption threshold and the position (index) of this value in the moving window array.

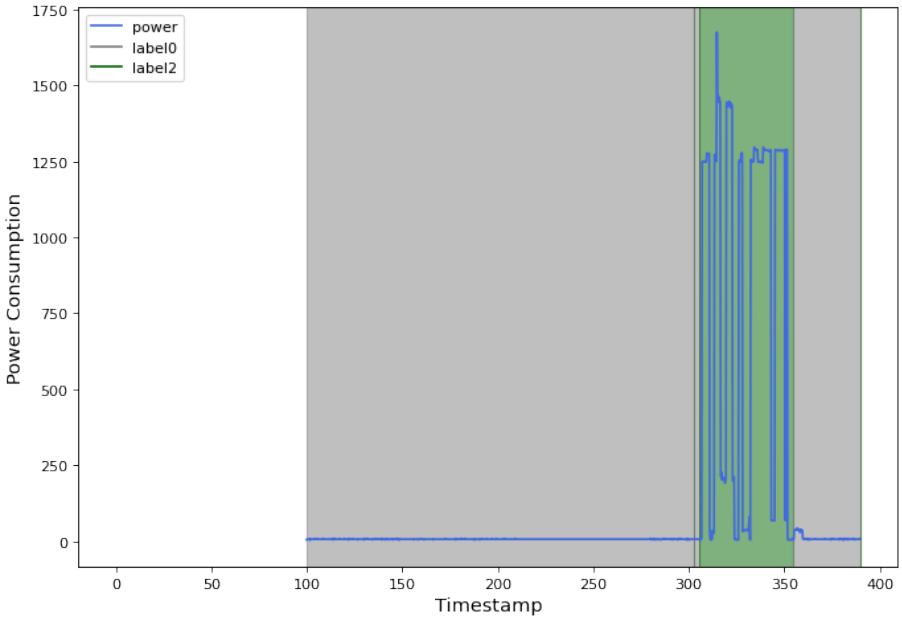


Figure 6.3: The sample of the time-series dataset given labeled button and power consumption.

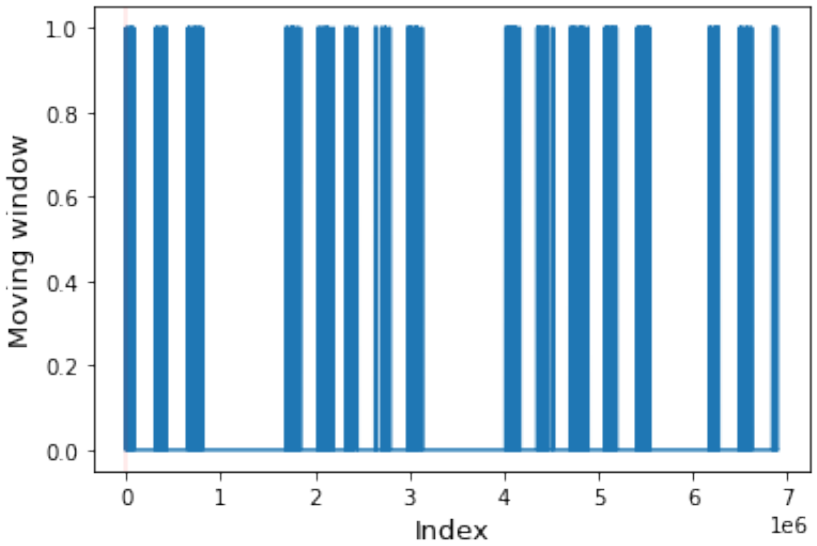


Figure 6.4: The value of moving window array with corresponding index.

The starting and ending points of the peak in time-series data were determined using a moving window array. We iterated through the moving window array, and

if the moving window value was equal to 1, we considered it a new starting point for the peak time-series; if it became equal to 0, the peak time-series ended.

From the time-series data, we were able to identify the peak starting and ending indices. These indices were utilized to create a new dataset that included peak time-series and label buttons. Figure 6.5 represents the peak of power consumption values with label two, while the red line shows the peak's starting point. As can be seen from the plot, we also analyzed non-peaked time-series power consumption data with label 0 before and after peak values. For the training of the DL model, we are also interested in the state of the coffee machine when no button is pressed.

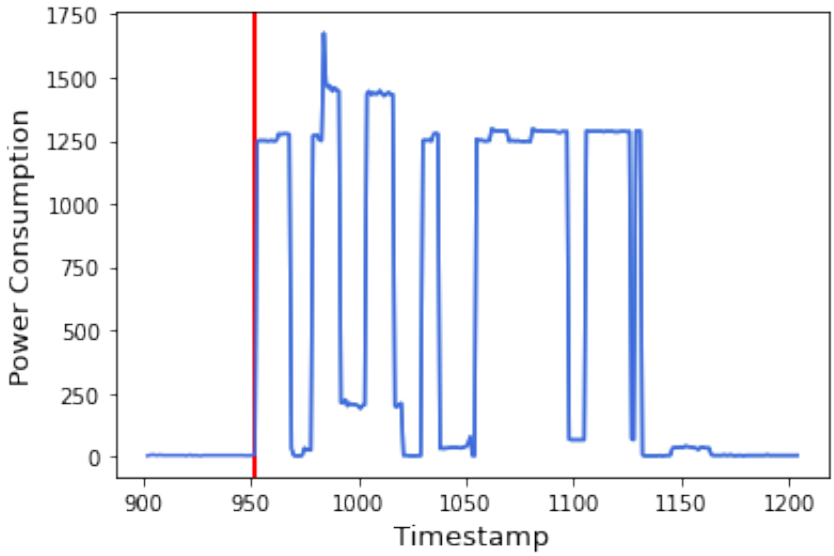


Figure 6.5: The first power consumption peak value when button "B" pressed.

6.4 Results

6.4.1 Training Deep Learning Model

The length of the new time-series peak dataset without label "0" is 354. We included some non-peaked time-series (200 samples) to be used for the future training of the DL model. To avoid dealing with values that can bias the output of the DL model due to too high or too low values, we normalized the time-series peak dataset by using the "maximum-minimum" normalization technique. Each label

in the dataset was mapped to a binary vector by the one-hot-encoding technique. We shuffled the input dataset and split it into a 10% validation dataset and a 90% training dataset. While tuning model hyper-parameters, validation data was used to provide an unbiased evaluation of the model fit for the training dataset.

Hyper-parameters (described in the Methodology section) determine the behavior of the Neural Network and its capacity to learn patterns from the training dataset.

Table 6.1 shows the parameters that were used for the first trial.

Table 6.1: The parameters used in the first step.

Batch size	Epochs	Activation Function	Hidden States
64	50	softmax	100

An advantage of using TensorFlow and Keras is that they make it easy to create models. We created a sequential model by passing a list of layer instances to the constructor. Figure 6.6 represents the building function of DL model. In compiling, we configured the model with the Adam optimizer and the categorical_crossentropy loss function.

```
input_dim, output_dim, n_features = train_x.shape[1], train_y.shape[1], 1
activation, loss, optimizer = 'softmax', 'categorical_crossentropy', 'adam'
hidden_states, epochs, batch_size = 100, 50, 64

model = Sequential()
model.add(GRU(hidden_states, input_shape=(input_dim, n_features)))
model.add(Dense(output_dim, activation=activation))
model.compile(loss=loss, optimizer=optimizer, metrics=['accuracy'])
model.build()
```

Figure 6.6: TensorFlow: build model function.

To create a GRU model, we need to add the GRU layer. Figure 6.7 represents the scheme of the neural network model. As we can notice, there is one GRU layer.

After training a DL model over 50 epochs, we got the training accuracy equals to 68.14% and validation accuracy to 62.82%. Validation accuracy is less than training one, which makes sense given that with training data the model is already familiar with. Figure 6.8 represents the accuracy of the training and validation dataset over epochs.

We used the Grid Search method to improve Neural Network performance. In

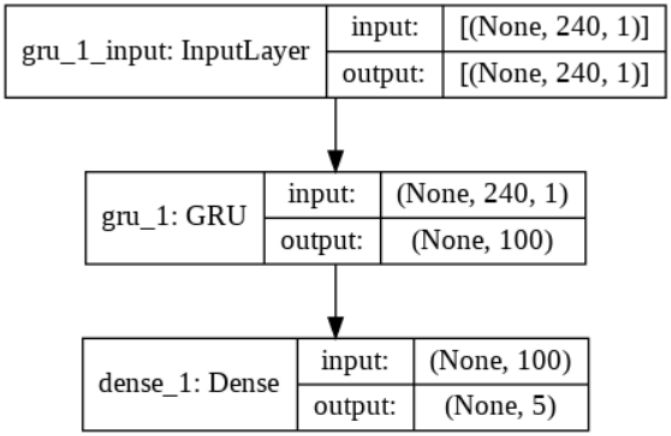


Figure 6.7: Plot of Neural Network Model Graph

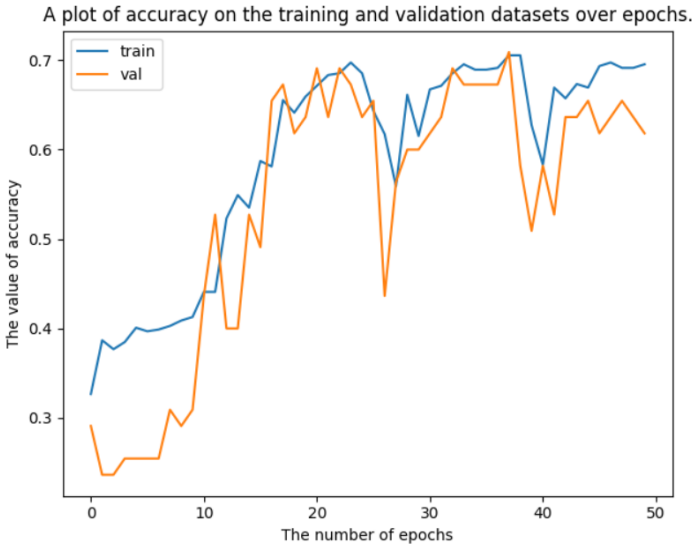


Figure 6.8: The plot of accuracy on validation and training data.

particular, we tuned various hyper-parameters to understand which ones were best for the problem. Table 6.2 lists all hyper-parameters that have been varied during the search for the best model. Hidden_dims parameter in the Table 6.2 is the number of neurons in this hidden layer. Both the batch size (64) and the Adam optimizer remained the same. It is worth figuring out that the number of possible configurations was 48.

We computed accuracy scores from each set of parameters using Grid Search,

Table 6.2: The hyperparameters used in Grid Search.

Epochs	Learning Rates	RNN Layer Types	Hidden_dims
[250, 500, 1000, 1200]	[0.01, 0.001, 0.0005]	[GRU, LSTM]	[50,100]

taking into account 48 alternative combinations, and the best combination was retrained. Figure 6.9 represents the best found configuration after the Grid Search implementation. According to Figure 6.9, 17 combination of 48 was the best one.

The next step was to train the DL model again by using the best-found hyper-parameters. Validation and training accuracy have both increased:

- Training Accuracy - 77.56% with best hyper-parameters, 68.14% without.
- Validation Accuracy - 72.73% with best hyper-parameters, 62.82% without.

```
best_config = configs[17]
best_config
```

```
{'opt_type': tensorflow.python.keras.optimizer_v2.adam.Adam,
 'lr': 0.001,
 'n_epoch': 250,
 'rec_type': tensorflow.python.keras.layers.recurrent_v2.GRU,
 'hidden_dim': 100,
 'dropout': 0,
 'rec_dropout': 0}
```

Figure 6.9: The best configuration parameters.

As shown in Figure 6.9, we can see that GRUs trains faster and performs better than LSTMs on a small dataset. Despite having a small dataset of 554 samples, the GRU model was able to improve both training and validation accuracy. Our goal was to build a DL model to predict the label of the pressed button, we discovered during data analysis (peak detection) that the data was insufficient to forecast the labels. For future work, different RNN could be explored, and it would be better to have more data samples to test the created model.

Chapter 7

Conclusion

In this work, we considered production monitoring as an Industry 4.0 problem and its importance in the Industry 4.0 environment. The goal of the work was to monitor production without intervention by real industrial machines and to provide a taxonomy for both Industry 4.0 problems and enabling technologies that were used to solve them. This has been feasible thanks to the research done on academic articles, use cases implemented within industrial companies, and white papers.

Industry 4.0 enables companies to incorporate their customers' needs and expectations into their development and production processes in novel ways, such as by sharing direct data with their machinery. It also simplifies machine data analysis, increasing productivity and reducing production process failures. The challenge is to create new business models, services, and products that take advantage of the potential for humans and machines to collaborate and maximize the benefits of that collaboration. To fuel the 4.0 transformation, it's important to give both the academic and industrial worlds a clear picture of what's possible and what's needed to make it happen. This taxonomy is the first step in the proper direction. This exploratory work will help with the explanation of the 4.0 environment and open the door for further research aimed at a deeper understanding of how the Industry 4.0 world is evolving.

As previously stated, deploying ML algorithms on IoT devices minimizes network congestion by allowing computations to be performed close to data sources while also reducing power consumption for continuous wireless transmission to cloud servers. The goal of this research was to consider two use cases: one for monitoring

the power usage of home appliances and the other for analyzing the state of a coffee machine. We used unsupervised ML methods to label data in the first use case, and then investigated the implementation of five supervised ML algorithms to identify power consumption profiles of home appliances. All five algorithms represent good accuracy for the considered use case. In the second use case, we studied the coffee machine's state analysis. After creating the dataset, we performed training of DL model by using RNN, namely LSTM and GRU. For this use case, GRU performed better, improving both training and validation accuracy. For future work, it is essential to have more peak time-series data samples to test the model and be able to distinguish the various tasks in the coffee machine based on the obtained results.

In the end, one of the possible future works is to continue this research on Industry 4.0 problems and enabling technologies by creating a dashboard with a search engine to query the taxonomy and analyze statistics about it. Another way to continue the research is to survey the companies that are adopting Industry 4.0 technologies to assess the current state of smart factories. The project's future implementation could include developing a real-time microcontroller-based system to monitor factory production.The state of the art is focused on ML systems deployed on edge devices, providing a comparison between the ML algorithms implementable in edge computing. In addition, the process of bringing ML to the edge was described in detail for future deployment. It is feasible to define a set of compatible hardware and software to implement ML on-the-edge based on the requirements (privacy, energy consumption, computational complexity).

Bibliography

[1] H. Lasi, P. Fettke, H. G. Kemper, T. Feld, and M. Hoffmann. «Industry 4.0». In: *Business & information systems engineering* 6.4 (2014), pp. 239–242 (cit. on p. 1).

[2] L. D. Xu, E. L. Xu, and L. Li. «Industry 4.0: state of the art and future trends». In: *International Journal of Production Research* 56.8 (2018), pp. 2941–2962 (cit. on p. 1).

[3] M. E. Porter and J. E. Heppelmann. «How smart, connected products are transforming competition». In: *Harvard business review* 92.11 (2014), pp. 64–88 (cit. on p. 1).

[4] Ron Davies. *Industry 4.0 Digitalisation for productivity and growth.* Sept. 2015 (cit. on p. 4).

[5] M. Rüßmann, M. Lorenz, P. Gerbert, and e.t.c. *Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries.* Apr. 2015 (cit. on pp. 4, 5, 12).

[6] Microsoft News Center. *Microsoft announces IoT Signals research report on state of IoT adoption.* July 2019 (cit. on p. 5).

[7] Richard Quinnell. *AI and the IoT merge to form the AIoT.* 2021 (cit. on p. 5).

[8] E. Charalambous, R. Feldmann, G. Richter, and C. Schmitz. «AI in production: A game changer for manufacturers with heavy assets». In: (Mar. 2019) (cit. on p. 9).

[9] M. Azadi Moghaddam and F. Kolahan. «Modeling and optimization of the electrical discharge machining process based on a combined artificial neural network and particle swarm optimization algorithm». In: *Scientia Iranica. Transaction B, Mechanical Engineering* 27.3 (2020), pp. 1206–1217 (cit. on pp. 9, 18).

[10] Serhii Maksymenko. «AI - based visual inspection for defect detection». In: *Mobidev* (Mar. 2020) (cit. on p. 9).

[11] Digiteum. *How IoT Impacts the Supply Chain*. Oct. 2019 (cit. on p. 9).

[12] D.A.Rossit, F.Tohmé, and M.Frutos. «Industry 4.0: Smart Scheduling». In: *International Journal of Production Research.* 57.12 (2019), pp. 3802–3813 (cit. on p. 9).

[13] Wil Van der Aalst. «Process Discovery: Capturing the Invisible». In: *IEEE Computational Intelligence Magazine* 5.1 (2010), pp. 28–41. DOI: 10.1109/MCI.2009.935307 (cit. on p. 9).

[14] Ż. Tomasz, M. Tomasz, K. Jacek, M. Madera, and J. Sęp. «Condition monitoring in Industry 4.0 production systems - the idea of computational intelligence methods application». In: *Procedia CIRP* 79 (2019), pp. 63–67. URL: https://doi.org/10.1016/j.procir.2019.02.012 (cit. on p. 9).

[15] T. Latinovic, C. Barz, A. POP, and P. Petrică POP. «FMEA ANALYSIS AS SUPPORT TO INDUSTRY 4.0 IN THE TOBACCO INDUSTRY». In: *Annals of the Faculty of Engineering Hunedoara* 18.1 (2020), pp. 83–87 (cit. on p. 9).

[16] Digiteum. *Logistic Industry Innovations with IoT*. Sept. 2019 (cit. on p. 9).

[17] Liudmyla Taranenko. «Machine Learning in demand forecasting for retail». In: *Mobidev* (June 2020) (cit. on p. 9).

[18] STL Partners. *STL Partners' Edge Insights Service*. 2020. URL: https://stlpartners.com/edge-computing/edge-insights-service/ (cit. on p. 9).

[19] LACROIX Electronics. *Predictive maintenance at LACROIX Electronics*. URL: https://www.st.com/content/dam/artificial-intelligence/edge-ai/stmicroelectronics-stlivedays-low-power-predictive-maintenance-Lacroix-marketing-presentation-2171.pdf (cit. on p. 9).

[20] Merck. *5G: POWERING A CONNECTED FUTURE*. URL: https://www.merckgroup.com/en/research/science-space/envisioning-tomorrow/smarter-connected-world/5g.html (cit. on p. 10).

[21] Zerynth. *Real-Time Production Performance Monitoring* (cit. on p. 10).

[22] Zerynth. *Smart Warehousing* (cit. on p. 10).

[23] Zerynth. *Blockchain-enabled IoT shipment tracking system* (cit. on p. 10).

[24] Zerynth. *Vehicle Tracking and Fleet Management* (cit. on p. 10).

[25] Libelium. *Smart Tracking* (cit. on p. 10).

[26] A.C. Valdeza, P. Braunera, A.K. Schaara, A. Holzingerb, and M. Zieflea. «Reducing Complexity with Simplicity - Usability Methods for Industry 4.0». In: vol. 9. 2015, p. 14 (cit. on p. 10).

[27] Particle. *Order fulfillment*. URL: https://www.particle.io/iot-order-fulfillment/ (cit. on p. 10).

[28] Particle. *An all-in-one asset tracking solution*. URL: https://www.particle.io/solutions/iot-asset-tracking/ (cit. on p. 10).

[29] Microsoft. *Predictive maintenance with the intelligent IoT Edge* (cit. on p. 10).

[30] Ferran Garcia Pagans. *Predictive Analytics Using Rattle and Qlik Sense*. Packt Publishing, June 2015. ISBN: 9781784395803 (cit. on p. 10).

[31] EDI Consortium. *IoT in Retail*. Mar. 2019 (cit. on p. 10).

[32] EDI Consortium. *Remote Measurements Control*. Mar. 2019 (cit. on p. 10).

[33] Siemens. *SIMATIC WinCC OA IOT Suite* (cit. on p. 10).

[34] Ivan Miljanić. «Data driven technology for efficiency in energy intensive industries». In: *REACH* (Nov. 2020) (cit. on p. 10).

[35] Ivan Miljanić. «Improving stores efficiency using clients shopping times». In: *REACH* (Nov. 2020) (cit. on p. 10).

[36] Insight SiP. *Smart PPE and IoT to improve workplace safety* (cit. on p. 11).

[37] FlashGlobal. *How to Improve Inventory Cost Management For After-Sales Supply Chain*. Aug. 2018 (cit. on p. 11).

[38] Libelium. *Smart Retail IoT Solution* (cit. on p. 11).

[39] Libelium. *Industry 4.0 IoT Solution* (cit. on p. 11).

[40] Particle. *The particle tracking system* (cit. on p. 11).

[41] H. Zhu, J. Gao, D. Li, and D. Tang. «A Web-based Product Service System for aerospace maintenance, repair and overhaul services». In: *Computers in Industry* 63 (2012), pp. 338–348 (cit. on p. 11).

[42] Ivan Miljanić. «Energy sources optimization». In: *REACH* (Nov. 2020) (cit. on p. 11).

[43] Ivan Miljanić. «Predictive maintenance and production optimisation in industry». In: *REACH* (Nov. 2020) (cit. on p. 11).

[44] I.S. Brito, L. Murta, N. Loureiro, P.R.D. Pacheco, and P. Bento. *Leveraging IoT Framework to Enhance Smart Mobility: The U-Bike IPBeja Project.* IGI Global, 2020, pp. 166–185 (cit. on p. 11).

[45] D. Ramotsoela, A. Abu-Mahfouz, and G. Hancke. «A Survey of Anomaly Detection in Industrial Wireless Sensor Networks with Critical Water System Infrastructure as a Case Study». In: *Sensors* 18.8 (2018), p. 2491 (cit. on p. 22).

[46] A.Azadeh, M.Saberi, A.Kazem, V.Ebrahimipour, A.Nourmohammadzadeh, and Z.Saberi. «A flexible algorithm for fault diagnosis in a centrifugal pump with corrupted data and noise based on ANN and support vector machine with hyper-parameters optimization». In: *Applied Soft Computing* 13.3 (2013), pp. 1478–1485 (cit. on p. 22).

[47] R. K. Patel and V.K. Giri. «Feature selection and classification of mechanical fault of an induction motor using random forest classifier.» In: *Perspectives in Science* 8 (2016), pp. 334–337 (cit. on p. 22).

[48] M. Syafrudin, G. Alfian, N. L. Fitriyani, and J. Rhee. «Performance analysis of IoT-based sensor, big data processing, and machine learning model for real-time monitoring system in automotive manufacturing». In: *Sensors* 18.9 (2018), p. 2946 (cit. on p. 23).

[49] Y. Han, X. Sha, E. Grover-Silva, and P. Michiardi. «On the impact of socio-economic factors on power load forecasting.» In: IEEE International Conference on Big Data, 2014, pp. 742–747 (cit. on p. 24).

[50] C. M. R.do Carmo and T. H. Christensen. «Cluster analysis of residential heat load profiles and the role of technical and household characteristics». In: *Energy and Buildings* 125 (2016), pp. 171–180 (cit. on p. 24).

[51] L. Xiang, T. Xie, and W. Xie. «Prediction model of household appliance energy consumption based on machine learning». In: vol. 1453. IOP Publishing Ltd, 2020 (cit. on p. 24).

[52] John E. Seem. «Pattern recognition algorithm for determining days of the week with similar energy consumption profiles». In: *Energy and Buildings* 37 (2005), pp. 127–139 (cit. on p. 24).

[53] K.P. Amber, R. Ahmad, M.W. Aslam, A. Kousar, M. Usman, and M. S. Khan. «Intelligent techniques for forecasting electricity consumption of buildings». In: *Elsevier* 157 (2018), pp. 886–893 (cit. on p. 25).

[54] Jeff Bier. *AI and Vision at the Edge*. 2020. URL: https://www.eetimes.com/ai-and-vision-at-the-edge/ (cit. on p. 28).

[55] Aurelien Geron. *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow*. 2019 (cit. on pp. 28, 29).

[56] M. Khanum, T. Mahboob, W. Imtiaz, H. A. Ghafoor, and R. Sehar. «A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance.» In: *International Journal of Computer Applications* 119.13 (2015) (cit. on p. 29).

[57] E. Baralis and T. Cerquitelli. *Clustering fundamentals*. URL: https://dbdmg.polito.it/wordpress/wp-content/uploads/2017/02/24e-DMClustering_2x.pdf (cit. on p. 30).

[58] Peter Waiganjo Wagacha. «Induction of decision trees». In: (2003) (cit. on p. 35).

[59] Z. Yin and J. Hou. «Recent advances on SVM based fault diagnosis and process monitoring in complicated industrial processes». In: *Neurocomputing* 174 (2016), pp. 643–650 (cit. on p. 36).

[60] Xue Ying. «An Overview of overfitting and its solutions». In: vol. 1168. 2. 2019. DOI: http://dx.doi.org/10.1088/1742-6596/1168/2/022022 (cit. on p. 37).

[61] Mark Robins. «The Difference Between Artificial Intelligence, Machine Learning and Deep Learning». In: (May 2020) (cit. on p. 38).

[62] D. Oliveira, F. Araújo, B. Barros, and e.t.c. «A System Based on Artificial Neural Networks for Automatic Classification of Hydro-generator Stator Windings Partial Discharges.» In: *Microwaves, Optoelectronics and Electromagnetic Applications.* (Sept. 2017) (cit. on p. 39).

[63] MathWorks. *What Is a Neural Network?* (Cit. on p. 39).

[64] Medium. *Activation Functions (Linear/Non-linear) in Deep Learning* (cit. on p. 40).

[65] Wikipedia. *Softmax function* (cit. on p. 40).

[66] D.P. Kingma and Jummy Ba. «Adam: A method for stochastic optimization». In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 40).

[67] Z. C. Lipton, J. Berkowitz, and C. Elkan. «A critical review of recurrent neural networks for sequence learning». In: *arXiv preprint arXiv:1506.00019* (June 2015) (cit. on p. 42).

[68] S.K.Saksena, B. Navaneethkrishnan, S. Hegde, and e.t.c. «Towards Behavioural Cloning for Autonomous Driving». In: *2019 Third IEEE International Conference on Robotic Computing (IRC)*. 2019, pp. 560–567 (cit. on p. 42).

[69] Sepp Hochreiter. «The vanishing gradient problem during learning recurrent neural nets and problem solutions». In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (1998), pp. 107–116 (cit. on p. 43).

[70] S. Hochreiter and J. Schmidhuber. «Long short-term memory». In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 43).

[71] M. Sundermeyer, R. Schluter, and H. Ney. «LSTM neural networks for language modeling». In: *Thirteenth annual conference of the international speech communication association.* 2012 (cit. on p. 43).

[72] Y. Wang, W. Liao, and Y. Chang. «Gated recurrent unit network-based short-term photovoltaic forecasting». In: *Energies* 11.8 (2018), p. 2163 (cit. on pp. 43, 44).

[73] P. Warden and D. Situnayake. *TinyML: Machine Learning With Tensorflow Lite on Arduino and Ultra-low-power Microcontrollers*. 2019, 520 pages (cit. on p. 44).

[74] TensorFlow. *TensorFlow Lite guide*. URL: `https://www.tensorflow.org/lite/guide` (cit. on p. 44).

[75] TensorFlow. *TensorFlow Lite inference*. URL: `https://www.tensorflow.org/lite/guide/inference` (cit. on p. 45).

[76] *About Keras*. URL: `https://keras.io/about/` (cit. on p. 45).

[77] *Edge Impulse Documentation*. URL: `https://docs.edgeimpulse.com/docs/responding-to-your-voice` (cit. on pp. 46–48).

[78] Zerynth. *Zerynth OS*. URL: `https://www.zerynth.com/zos/` (cit. on p. 50).

[79] Zerynth. *ZM1 datasheet*. URL: `https://www.zerynth.com/download/20127/` (cit. on p. 53).

[80] *ESP - IDF Programming Guide*. URL: `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html` (cit. on pp. 53, 54).

[81] E.J. Aladesanmi and K.A. Folly. «Overview of non-intrusive load monitoring and identification techniques». In: *IFAC-PapersOnLine* 48.30 (2015), pp. 415–420 (cit. on p. 55).

[82] V. Hodge and J. Austin. «A Survey of Outlier Detection Methodologies». In: *Artificial Intelligence Review* 22.2 (2004), pp. 85–126 (cit. on p. 58).

[83] Wikipedia. *Percentile*. URL: `https://en.wikipedia.org/wiki/Percentile` (cit. on p. 59).

[84] Scikit Learn. *Preprocessing data*. URL: `https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-scaler` (cit. on p. 62).