People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronic

Project Report Presented in Partial Fulfillment of

the Requirements of the Degree of

'Master'

In Computer Engineering

Title:

Embedded AI-based induction motor diagnosis and fault classification

Presented By:

- **Mr. MAALLEM Nassim**

Supervisor:

- **Dr. TOUZOUT Walid**

Co-Supervisor:

- **Mr. OUAZENE Hamza**

Abstract

In any industry, regardless of how reliable the equipment is, it is prone to failures and degradation due to many factors ranging from environmental factors to simply reaching the end of the life cycle. This made fault diagnosis a necessary and reliable tool, to maintain equipment and extend their lifetime.

This report focuses on presenting the development of an AI-based system for fault diagnosis of an induction motor, using classification techniques and deploying it on an embedded system, with the aim of extending equipment lifetime and maintaining its efficiency. The project utilizes data collected from accelerometers and acoustic sensors to train multiple AI models mainly: Support-vector-machines, k-nearest-neighbor, Decision tree, Random forest, Feed forward neural network, and Long-short-term memory. As a result, the feed forward neural network model is found to perform the best in terms of accuracy, model size, and testing time among the evaluated models. Moreover, the system we deployed on an ESP32-S3 SoC, performed well and proved to be reliable for industrial application when tested with new data. Consequently, the findings highlight the reliability and precision of AI models in fault diagnosis tasks, and showed how to benefit from deploying such systems on embedded platforms. Overall, the presented report emphasizes the importance of fault diagnosis in industrial settings and showcases the practicality and effectiveness of AI on embedded systems in this domain.

Acknowledgments

First, I would like to thank Allah for the faith and strength he spreads in my heart, to not only achieve this humble body of work, but to continue my studies.
Al hamdoulillah.

I would also like to thank my family for their patience and support throughout the years.

I would like to express my heartfelt gratitude to my supervisor Dr. Touzout Walid, who accepted my thesis proposal and most importantly guided me in all the duration of this project.

I would also like to thank my co-supervisor Ouazene Hamza, whose help was monumental in and out of the workspace.

My thanks also go towards Bettahar Toufik and Hamed Salah, who supported me in the early times of this project.

Finally, my gratitude goes towards all the professors and colleagues who assisted me, and to those who contributed directly or indirectly to the completion of this work.

Dedications

I dedicate this modest work to:

My dear parents who supported me and provided me with all I needed to reach this stage in my studies, and their sacrifices throughout the years.

My siblings who are always a nice change of pace from the seriousness of work and life in general.

And finally to my close circle, who've been with me from the start.

Table of content:

List of figures

List of tables

List of abbreviations

Abbreviation	Definition
AI	artificial intelligence
ASM	attribute in the dataset using Attribute Selection Measure
BNN	binarized neural network structure
CAHAI	Committee on Artificial Intelligence
DFT	Discrete Fourier Transform
FFT	Fast fourier transform
ICP	in-Circuit Programming
IoT	Internet of Things
KNN	K-Nearest Neighbor
SVM	Support Vector Machine
DT	Decision Tree
RF	Random Forest
FFNN	Feed Forward Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
LCC	Life Cycle Cost
SoC	System on Chip
PCA	Principal Component Analysis

General introduction

Machines can vary from simple home appliances to massive multi million dollar industrial equipment. However, what they all have in common are failures. These faults can lead to costly repairs, downtime, and in several cases safety hazards. In rotor machines, faults can reduce efficiency, cause errors in the power output, cause physical damage, increased Co2 emissions and the possibility of dangers affecting the workforce.

In response to that, several measures and solutions have been proposed to deal with these issues. This includes approaches such as corrective, preventive and reactive maintenance, which as of late, have been leaning more and more into applying artificial intelligence techniques to automate the process of identifying and ultimately repairing the equipment.

Induction motors are a prime example of machines that require constant monitoring and maintenance due to the harsh operating conditions they experience. Faults in the induction motors can result for multiple reasons including mechanical wear and tear, erosion, corrosion, thermal stress, faults in sensors and actuators. These faults are considered widespread and a frequent recurrence thus requiring constant supervising. They can reduce the effectiveness of the motor, which can be crucial to its performance, the overall spending on the equipment and the environmental impact.

The emergence of artificial intelligence (AI) has revolutionized the field of fault diagnosis, offering powerful tools for analyzing complex data to make informed decisions. In this thesis, the focus is on developing an AI-based system for fault diagnosis of induction motors, to ultimately deploy it on an embedded system-on-chip (SoC) platform.

The primary objectives of this project are twofold. Firstly, to develop an AI-based system capable of monitoring the state of the induction motor and accurately diagnosing any potential faults. This involves collecting data from multiple sensors using a test bench, and preprocessing the data, to then train various AI models. Secondly, to evaluate the performance of these AI models and identify the most suitable one for deployment on an SoC platform. Factors such as accuracy, model size, and test time will be considered in the evaluation process. The aim is to select a model that offers the best combination of these factors, ensuring reliable and efficient fault diagnosis capabilities.

In this thesis we are going to first introduce maintenance in a general manner, and explain its types and advantages, in order to give an idea about the form of maintenance we will be addressing. We will then elaborate on the role of artificial intelligence in fault diagnosis and the common techniques used in industrial settings for classification problems. Furthermore, we will also be discussing deployment of AI models on embedded systems, its advantages, and the common techniques that facilitate the process. Finally, we will present the methodology we used and discuss the result we obtained, as well as evaluate the work on realistic data, representative of real induction motor failure.

CHAPTER I: Principles of maintenance

I.1 Introduction

Although the term maintenance was a recent entry to the modern language and workspace, it goes so far back to the time where humans made the first tools and sought to retain their functionality.

In this chapter, we will introduce notions of industrial maintenance and its types, as well as a general idea on supervising equipment and diagnosis.

I.2 Notations

I.2.1 Fault

When it comes to machines and equipment, it refers to any defect or deviation that affects the normal functioning of a system. It is represented by any deviation of the observed characteristics of the equipment from the ideal theoretical benchmark characteristics. Faults can be due to external as well as internal factors. It can appear on sensors, actuators or the system itself and can be a warning for future failures. It should be noted that a fault does not necessarily induce failure.

I.2.2 Failure

It is the inability of a system, component or equipment to perform its intended task and duty within specified performance requirements, meaning measured characteristics cease to correspond to the theoretical characteristics.

I.2.3 Degradation

It is the loss of performance of a piece of equipment or one of its functions over time without necessarily putting the system in critical condition due to wear & tear, aging, and the lifespan of the equipment decreasing naturally. This could be in one of two ways:

- One of the functions performed by the whole was lost.
- A subset of the equipment degraded or failed which does not affect the whole.

I.2.4 Breakdown

It refers to the sudden and unexpected failure of a piece of equipment. Breakdown describes the state the machine is in, while failure is the event that led to that state. We could say that time wise, failure is set in time as a date and the breakdown is the duration in time between the occurrence of failure and the repair end date.

I.2.5 Availability

It means the ability of an asset to be in a functioning state and available for utilization when called upon. The device could be in the degrading stage and still be available for use.

I.2.6 Reliability

The ability of an asset to perform its intended purpose without failure for a certain period of time.

I.2.7 Failure rate

This is considered to be an important term when it comes to predictive maintenance and it means the rate at which a machine is expected to fail. It could also be defined as the frequency of failure over time, which could be expressed as the number of failures per unit of time. Failure rate is a metric used in reliability engineering, used to predict the future reliability of a system.

I.2.8 Life cycle cost

It is the total cost of an asset over its entire life cycle, including acquisition, operation, maintenance, and disposable cost.

I.3 Maintenance

Maintenance is a crucial process aimed at preserving the performance, reliability, and safety of systems and equipment, minimizing defects and failures.[1] It involves some practices that aim to minimize or prevent the occurrence of defects and failures that might compromise the efficiency and safety of a system. The early detection of faults in pieces of machinery and the estimation of their lifespan is pivotal in ensuring their wellbeing and ideal workflow.

I.4 Maintenance and its types

Maintenance is defined as a set of activities, procedures, and precautions that are aimed to keep a machine or a system in working condition or restore its operation state in case of failure. However, while the general purpose is the same, the techniques used and the time of intervention could vary dividing maintenance into multiple types. There exists multiple representations of the groups and sub groups of maintenance, so for the purpose of this thesis we're adapting one with two main approaches:

- The reactive approach
- The proactive approach

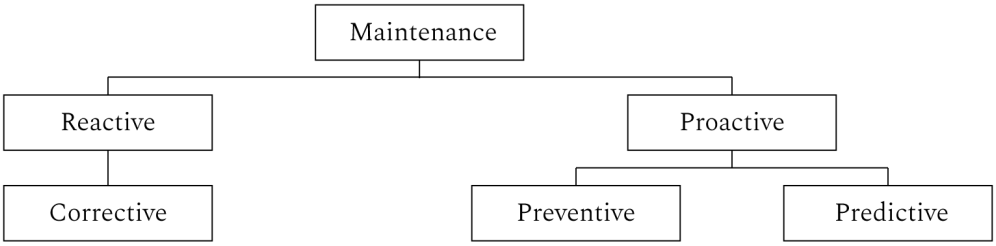


Figure 1: types of maintenance

I.4.1 Reactive maintenance

Reactive maintenance, also known as breakdown maintenance or run-to-failure maintenance, is an approach that responds to failure after their occurrence in a system or a piece of machinery. Here, repairs are made as needed to bring the machine back to its functioning state. If done using traditional techniques, it can be a less efficient and more costly type, as it often leads to unplanned downtime, increased repair cost, and reduction of the equipment lifespan.

I.4.1.1 Corrective maintenance

Corrective maintenance is a subset of reactive maintenance and shares a lot of similarity with the latter. For instance, it is based on the same principle of maintenance after a failure has occurred and it can be in some cases slightly costly and time consuming. However, even though corrective maintenance is a subcategory of reactive maintenance there are some distinctions to be made between the two:

- Corrective maintenance is generally considered as a planned or scheduled activity as opposed to reactive maintenance. It involves shutdown periods where the equipment is inspected and repaired. In contrast to that, reactive maintenance is most often that not, unplanned and only initiated once a failure of the system occurred.
- Corrective maintenance is generally associated with machines that have a long service life or remaining useful lifespan and are expected to undergo multiple periodic repairs, while reactive maintenance is often associated with equipment approaching their end.

I.4.2 Proactive maintenance

A strategy that focuses on identifying potential problems and taking actions early to prevent them from occurring. In this approach, regular equipment check ups and monitoring are indispensable to detect any signs of wear or failure. Ultimately, the purpose of proactive maintenance is to minimize downtime and extend the lifetime of the equipment, in addition to reducing the cost of maintenance.

I.4.2.1 Preventive maintenance

Preventive maintenance is a planned maintenance approach performed at regular intervals in order to prevent failures from taking place. It can either be time-based maintenance where a certain amount of time has elapsed, or usage-based maintenance where the machine has undergone a predefined amount of usage.

I.4.2.2 Predictive maintenance

Predictive maintenance is a more advanced type of maintenance that is primarily data driven and uses analytics and artificial intelligence to predict when maintenance should be performed. Its main point is to anticipate when machine failure might occur, with the key goal being to accurately predict the remaining useful lifespan (RUL).

predictive maintenance could also be used to identify the root cause of failure, which may ease resource reallocation to fix the issues more organically.

I.5 The levels of maintenance

The french standardization association (AFNOR) standard NF X 60-000 (2016) defines 5 levels of industrial maintenance ranked by increasing complexity of the corresponding maintenance intervention. This allows one to choose the suitable level of expertise required as well as the method necessary to carry out the operation.[2]

I.5.1 First level maintenance

It consists of simple interventions that are necessary and realized on easily available elements. These tasks do not require the equipment to be dismantled or opened and can be performed by the operator or a non-specialized operator using equipment support integrated into the property. This includes some surface level corrective actions such as changing consumables (light bulbs, fuses..etc), while it can also include some preventive actions such as condition monitoring rounds, daily lubrication of components and necessary checks and adjustments.

I.5.2 Second level maintenance

The second level corresponds to processes which are less complex and require straightforward simple procedures. Component replacement does not require full dismantling of the device.
These tasks are performed by qualified technicians with safety hazard training, therefore done by fairly qualified individuals. Overall this level of maintenance does not entail the full dismantling of equipment, thus a work related to isolated elements or results verification.

I.5.3 Third level maintenance

The third level's interventions are considered fairly complex, they are therefore preceded by a diagnosis and intervention stage. Maintenance at this level must take into consideration the equipment as a whole due to the fact that modifications here may affect the general operation.

I.5.4 Fourth level maintenance

At this level mastery of a technology since the maintenance operation is more critical and complex, requiring special technical expertise. They must therefore be carried out by a technician or a team of specialized technicians with a specific qualification, and supervised by a specialized manager.

I.5.5 Fifth level maintenance

At this stage maintenance involves renovation and restoration which is comparable to the manufacturing process. This is considered the most complex of them all.

I.6 Diagnosis

I.6.1 Definition

It is a corrective approach which refers to the process of analyzing a set of symptoms and variables, with the aim of establishing and identifying the state of a piece of equipment. In case of abnormal behavior, it allows us to find the root of the problem. It involves analyzing symptoms, conducting tests, and gathering relevant information to reach a conclusion about the underlying problem or condition. This is done in order to better choose the measures to be taken to repair the device, the operations to be done, and to get an approximation of the remaining useful lifespan.

I.6.2 Fault detection procedure

There exist two essential tasks:

- The observation of symptoms that led the system to failure.
- The identification of the source of failure

This is done through prior knowledge about the behavior of a system presented with different sets of variables and observing its behavior under different certain circumstances, accumulating information about the logical link between cause and

consequence. Our overall objective is to detect, locate and identify failures that could potentially have an impact on its operating safety.

- Detection: It consists of using a set of measures and fault indicators in order to create and generate what might be called symptoms.
- Localization: This consists of diving deeper to identify the failing part and perhaps the root cause for failure.
- Fault identification: it is the process of recognizing the defect and classifying it into its proper category. It is also possible to include decision making, in which case human intervention through taking corrective measures to avoid further degradation or to get the normal working flow back on track, thus returning to the norm

I.6.3 Fault diagnosis methods

The classification of diagnostic methods is not unique and is dependent on the specific field in which researchers are interested. We will be splitting diagnosis in our case into two distinct categories namely hardware redundancy and analytical redundancy. Briefly speaking, the use of hardware redundancy, is the use of multiple hardware such as sensor readings (two or three sensors), to analyze the same information and diagnose flaws. However, analytical redundancy necessitates knowledge of particular information provided by a mathematical model, a signal (frequency) information, or historical system data[3].

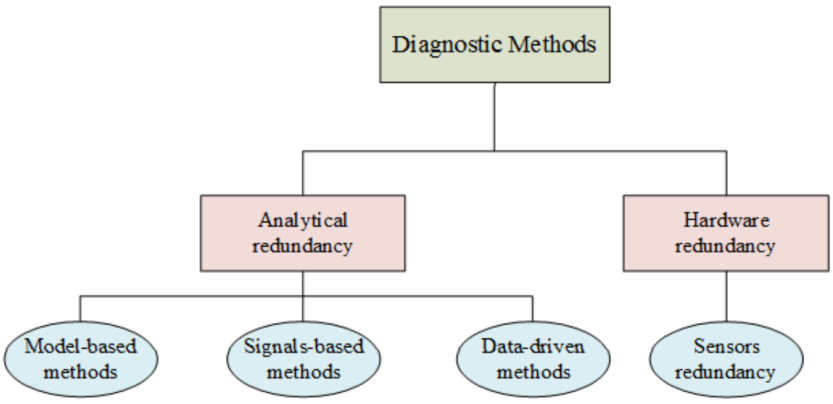


Figure 2: Diagnosis methods main types

I.6.3.1 Hardware redundancy

The key idea of the hardware redundancy is to use several sensors in order to measure the same outputs. The hardware redundancy deals with the comparison of duplicative signals generated by more than one sensor using algebraic relation between different system variables. The advantage of this method appears in its reliability and its simplicity. On the other hand, this diagnosis method presents a major setback which is the expensive cost of maintenance due to the extra equipment. An additional space is required to place the extra equipment that also limits the use of this method[4].

I.6.3.2 Analytical redundancy

The analytical redundancy is mainly based on specific information given by local or global modeling of the system. In this thesis, the analytical redundancy is divided into three categories: model-based methods, signals-based methods and data driven methods[4].

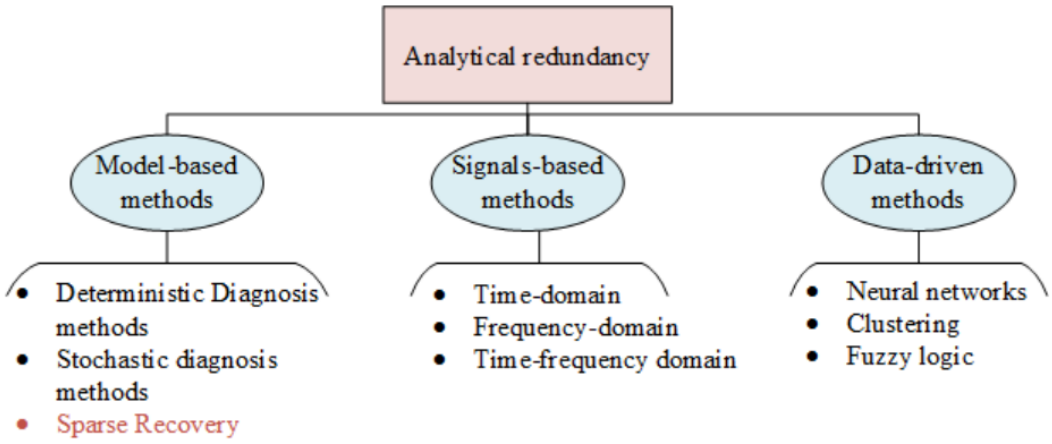


Figure 3: types of analytical redundancy

- **Model-based methods**

Because they do not call for extra hardware, the model-based techniques are far more effective than hardware redundancy. Model-based techniques are based on a mathematical model created by formulating certain fundamental physics concepts. In general, model-based approaches enable us to compare the measurements of actual systems with the outputs anticipated by the model. We can divide model-based methods with respect to the type of the model, into three categories: deterministic diagnostic methods, stochastic diagnostic methods, and sparse recovery methods.

- **Signals-based methods**

Signal-based methods depend on measured signals which contain the fault characteristics. On a general basis, signal based methods are divided into three areas: time-domain methods, frequency-domain methods and time-frequency methods. We will go into more details, as it is a critical aspect of our feature extraction process and the way we view the data collected for this problem as a whole[4].

- Time-domain methods are the basic diagnostic methods, based on extraction of time-domain characteristics of the base signals of the dataset. The common time-domain methods are: 6 root-mean-square, crest factor, absolute value, kurtosis.
- Frequency-domain methods are used to detect abnormality or defect by using the spectrum analysis tools. The Fast Fourier Transform FFT is the most applied method in frequency analysis technique. It aims to extract the fault indication from the data measured on the machine and to analyze the frequency characteristics and the amplitude of the defect.

- Time-frequency approaches are useful for analyzing non-stationary data. They try to determine the frequency characteristics of signals and extract their temporal variant properties, which can be useful diagnostic tools. The short-time Fourier transform, wavelet transform, and Hilbert transform are typical time-frequency approaches.

- **Data-driven methods**

Data-driven approaches are diagnosis techniques that do not need a physics mathematical model. A model of the process that links measured inputs to measured outputs is created using data-driven methodologies, and this model is then compared to the actual process to produce residuals. Among the data driven methodologies, fuzzy logic, neural networks, and clustering are often used diagnostic techniques.

I.7 Objectives of maintenance in the industry

The objective of maintenance is to ensure the optimal functioning and reliability of industrial equipment, systems, and processes. The primary goals of industrial maintenance include:

1. Equipment Reliability and Availability: The main objective of maintenance is to ensure that industrial equipment is reliable and available for operation when needed. This involves preventing unexpected failures, minimizing downtime, and maximizing the availability of critical assets.

2. Cost Optimization: Maintenance aims to optimize costs associated with equipment ownership and operation. This includes managing maintenance expenses, minimizing unplanned repairs, and optimizing maintenance strategies to achieve the desired balance between maintenance costs and equipment performance.

3. Risk Management: Maintenance helps mitigate risks associated with equipment failure. By implementing appropriate maintenance practices, businesses can reduce the likelihood of accidents, production disruptions, and safety hazards. Risk management in maintenance involves identifying potential risks, implementing preventive measures, and responding effectively to any emergent issues.

4. Asset Lifecycle Management: Industrial maintenance aims to maximize the lifespan and value of assets throughout their lifecycle. This involves proper maintenance planning, scheduling preventive and predictive maintenance activities, optimizing maintenance intervals, and considering factors such as obsolescence, upgrades, and replacements.

5. Continuous Improvement: Maintenance plays a role in continuous improvement efforts by identifying opportunities for equipment optimization, performance enhancement, and operational efficiency. It involves analyzing data,

identifying patterns, and applying corrective actions to enhance equipment reliability and performance over time.

6. Compliance and Regulatory Requirements: Industrial maintenance ensures compliance with relevant regulations, standards, and safety guidelines. It includes adhering to maintenance-related legal obligations, conducting inspections, and implementing maintenance practices that align with industry-specific requirements.

I.8 Conclusion

This chapter is a general introduction into maintenance, where we went over the principles of maintenance in industrial settings. We explained in this chapter the different types of maintenance, the levels at which maintenance could be conducted, some approaches to performing maintenance, as well as the notion of diagnosis.

Diagnosis could take many forms, and a failure can be detected and recognised in more than one way, using different inputs and different techniques. Regardless of the approach, the goal is to extend the equipment's useful lifetime.

CHAPTER II: An overview of artificial intelligence

II.1 Introduction

In the middle of the 20th century, machine learning was introduced to industry. Due to significant developments in the 1990s and early 2000s [22], which were made possible by the massive jump of processing power and data availability, neural networks were developed and found practical use in a variety of sectors. This was followed by the rise of big data and digital technologies which further accelerated the adoption of machine learning and deep learning in a variety of fields with a wide range of applications such as industrial diagnosis or diagnosis in the medical field.
In this chapter we will introduce artificial intelligence in a broad manner and describe its applications. We will also expand on machine learning and deep learning techniques and the algorithms used for diagnosis.

II.2 Overview of artificial intelligence

In order to explore the relation between industrial diagnosis and artificial intelligence, we have to define AI first. . This is, however, immediately challenging. In fact, the description and boundaries of AI are contested, without a universally accepted single definition, and are constantly shifting;
A lot of cutting-edge AI has filtered into general applications, often without being called AI because once something becomes useful enough and common enough it is not labeled AI anymore. [Nick Bostrom n.d.]
Leslie and al paper[5], draws on the Council of Europe's Ad hoc Committee on Artificial Intelligence (CAHAI) Feasibility Study, and defines AI systems as follows;

AI refers to machine-based systems that can, given a set of human-defined objectives, make predictions, recommendations, or decisions that influence real or virtual environments. AI systems interact with us and act on our environment, either directly or indirectly. Often, they appear to operate autonomously, and can adapt their behavior by learning about the context. (UNICEF 2021: 16)

All in all, artificial intelligence models need two things on a basic level: a large collection of data, and the algorithm that processes that data. Through what we have discussed above, it is safe to say that AI enhanced how industrial operations and maintenance was done and created a more reliable space.

II.3 Fundamentals of machine learning

II.3.1 Overview of machine learning

Machine learning is the design and study of software artifacts that use past experience to inform future decisions; machine learning is the study of programs that learn from data.

The fundamental goal of machine learning is to generalize, or to induce an unknown rule from examples of the rule's application.

While the principle of machine learning is the same which is to make a self sustainable system that imitates human logic, the way a machine learning model learn is different and can be classified into 4 main categories:

- Supervised learning
- Semi-supervised learning
- Unsupervised learning
- Reinforcement learning

These types of learning differ in the way they handle their input data and how they acknowledge it as well as how they improve themselves for optimal results. These differences are more apparent in real life scenarios, where each problem demands a different approach and has different prerequisites by nature.

According to the paper by Matt Harrison [6], the machine learning process usually consists of 6 main steps:

1. Problem understanding
2. Data understanding
3. Data preparation
4. Modeling
5. Evaluation
6. Deployment

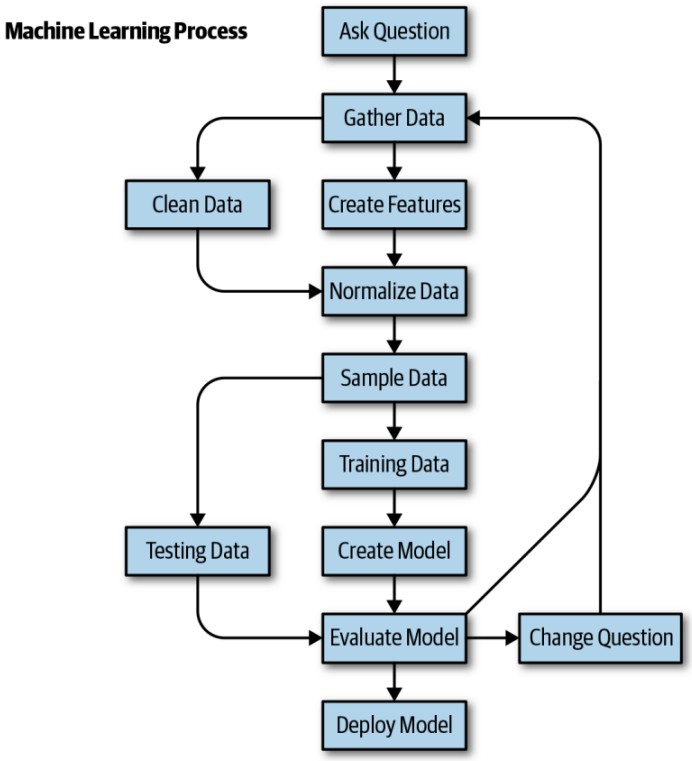


Figure 4: Machine learning problem solving process

II.3.2 Types of machine learning

As discussed, machine learning is an inclusive concept of a set of algorithms that opt to mimic human logic. However, the approach varies from one problem to another. We will explain each type briefly then expand on supervised learning for relevance to the problem at hand (diagnosis).

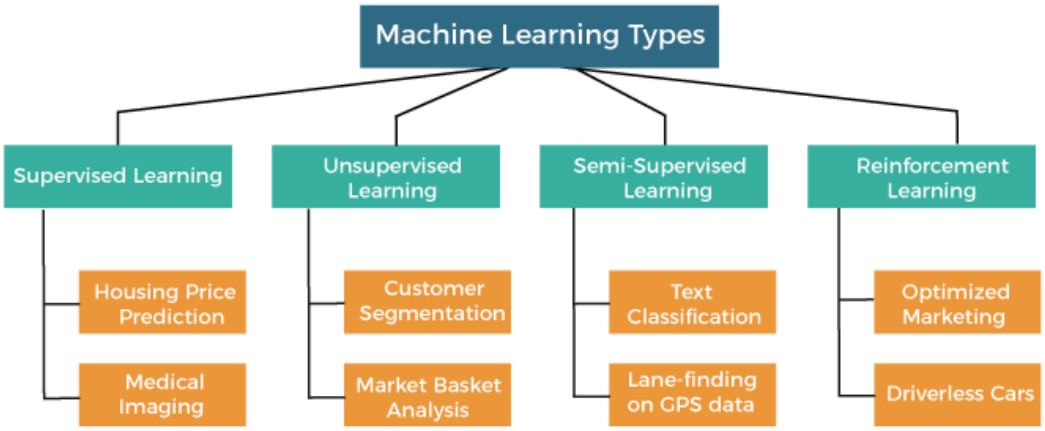


Figure 5: learning approaches in Machine learning

- Supervised Learning: is a machine learning approach where models are trained on labeled data, with inputs and corresponding desired outputs, to make predictions or classifications.[8]

- Unsupervised Learning: describes a machine learning approach where models analyze unlabeled data to discover patterns, structures, or relationships without explicit guidance[7].

- Semi-Supervised Learning: is a hybrid approach that utilizes both labeled and unlabeled data to train models, combining the advantages of supervised and unsupervised learning [36].

- Reinforcement Learning: is simply a learning paradigm where an agent interacts with an environment, learning from feedback in the form of rewards or punishments to optimize its actions and achieve a specific goal [37].

II.3.3 Supervised learning

Supervised machine learning, as the name implies, is based on supervision. It implies that in the supervised learning approach, we train the machines using a "labeled" dataset, and the machine predicts the output based on the correlation it extracts from input (explanatory variables) and output (response variables). The tagged data in this case indicates that some of the inputs have already been mapped to the output. Simply put, we first train the machine with the input and associated output, and then we ask the machine to replicate that process with only the input provided this time. This is similar to giving the machine questions and their respective answers which enables it to learn, then providing new questions and expecting the model to answer. We can distinguish two main tasks or categories: regression tasks and classification tasks.

II.3.3.1 Regression

Regression is a supervised learning technique used to predict continuous numeric values based on input variables. It aims to establish a mathematical relationship between the input variables (also known as predictors, independent variables, or features) and the continuous target variable (also known as the dependent variable or response variable). The goal is to create a model that can accurately estimate the relationship and make predictions on unseen data [7].

In regression, the model learns from a labeled training dataset, where each data point consists of a set of input variables along with the corresponding target values. The model then identifies patterns, trends, and dependencies in the data to create a mathematical function or equation. This function can be used to predict the target value for new data based on its input variables [8].

Regression techniques encompass a wide range of algorithms, including linear regression, polynomial regression, decision tree regression, random forest regression, and support vector regression, among others. Each algorithm has its own assumptions, advantages, and limitations, making them suitable for different types of data and problem domains.

II.3.3.2 Classification

Classification is a supervised learning mechanism for labeling a sample based on the features [6]. It is a technique used to assign data instances into predefined classes or categories based on their features. It is primarily employed in scenarios where the target variable represents discrete or categorical outcomes. The objective of classification is to build a model that can accurately classify new, unseen instances into the appropriate classes [7].

The model analyzes the patterns and relationships among the input variables to create decision boundaries or rules that differentiate between different classes. These boundaries or limits are based upon some classification algorithms which will be explored further.

Sentiment analysis, spam identification, picture recognition, disease diagnosis, and credit risk assessment are just a few of the uses for classification [9]. Depending on the data's features, the problem's complexity, and the need for interpretability, a classification algorithm is selected.

II.3.4 Support vector machines (SVM)

Support Vector Machines (SVM) have been developed in the framework of statistical learning theory (Vapnik, 1998) (Cortes and Vapnik, 1995), and have been successfully applied to a number of applications, ranging from time series prediction (Fernandez, 1999), to face recognition (Tefas et al., 1999), to biological data processing for medical diagnosis (Veropoulos et al., 1999). Their theoretical foundations and their experimental success encourage further research on their characteristics, as well as their further use [10].

A Support Vector Machine (SVM) is an algorithm that tries to fit a line (or plane or hyperplane) between the different classes that maximizes the distance from the line to the points of the classes. In this way it tries to find a robust separation between the classes. The support vectors are the points of the edge of the dividing hyperplane [6].

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine [11].
The SVM formulation involves solving an optimization problem to minimize the equation described in (eq.1) and to find the decision boundary. With a general formula being:

Minimize:

$$0.5\times \|w\|^2+ C\sum_{i=1}^{l} \xi \qquad \text{(eq.1)}$$

Subject to:

$$y_i\times \left(x_i x_T b\right) \geq 1-\xi_i, \; for\; all\; i \qquad \text{(eq.2)}$$

Where:

- w represents the weight vector that defines the decision boundary.
- b is the bias term.
- C is the regularization parameter that balances the trade-off between maximizing the margin and minimizing the training error.
- ξ represents the slack variables that allow for some misclassification of training examples.
- x_i is a training example (input vector) with its corresponding class label y_i .
- T is the full size of the dataset, l is an instance index.

Consider Figure 6, in which there are two different categories that are classified using a decision boundary or hyperplane:

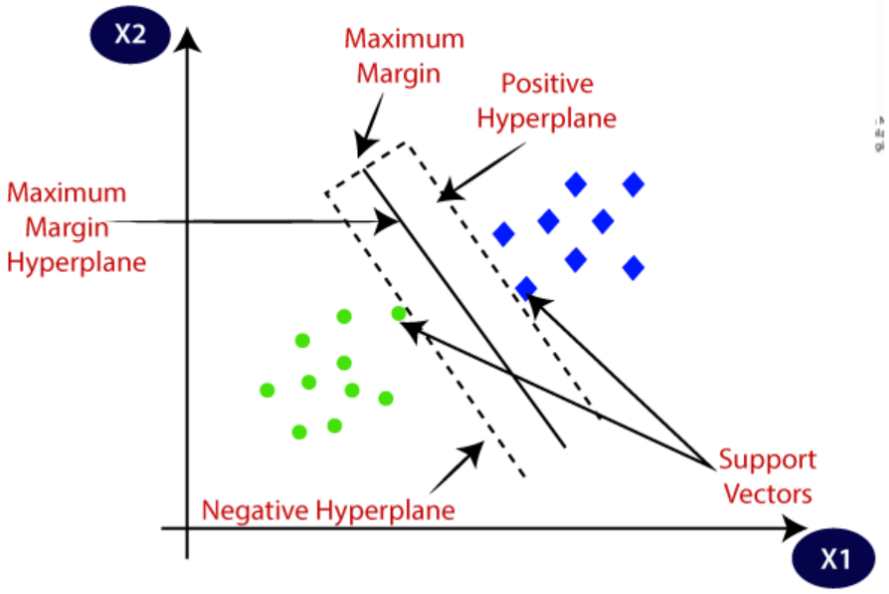


Figure 6: Binary classification diagram using SVM

Support vector machines can be of two types depending on the way they fit the data:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

The advantage of SVM is their excellent capability to deal with high dimensional and non linearly separable classification problems, with one of the only drawbacks being the sensitive parameters that we need to set just right (although that holds as well for most of the classification models we are going to use).

II.3.5 K-nearest neighbor (KNN)

K-Nearest Neighbors (KNN) is a simple yet effective algorithm used for both classification and regression tasks. In KNN, the class or value of an unseen data point is determined by considering its k nearest neighbors in the training set [7]. The algorithm

operates by calculating the distance between the new data point and all other data points in the training set using a distance metric such as Euclidean distance. The k nearest neighbors are then selected based on the shortest distance [12]. For classification tasks, the class label of the new data point is determined through majority voting among its k nearest neighbors. In regression tasks, the predicted value is calculated as the average of the target values of the k nearest neighbors [13].

K-Nearest Neighbors is a non parametric algorithm and a lazy learner which means it does not make any assumptions on the underlying data, nor does it learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. It just saves the dataset during the training phase and then classifies fresh data into a category that is quite similar to the new data as it comes in.

We can represent the working of a K-nn model in 5 steps:

1. chose the number of neighbors or the K.
2. Compute the euclidean distance from the data point to the set

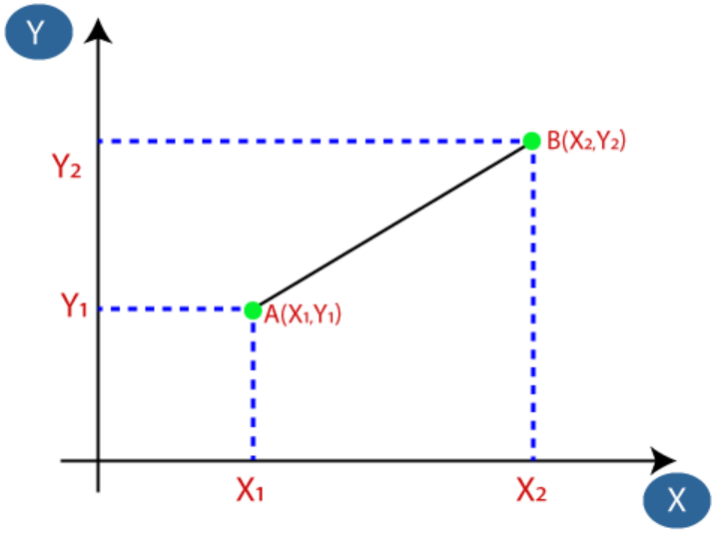


Figure 7: plane representation of distance calculation between two data points

The euclidean distance between point A and point B is calculated by the equation (eq.3) as follows:

$$dis = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2} \qquad \text{(eq.3)}$$

3. Select the K nearest neighbors as per the distance calculated
4. Counting the number of neighbors from each category
5. Assign the new data point to the majority class of it K nearest neighbors

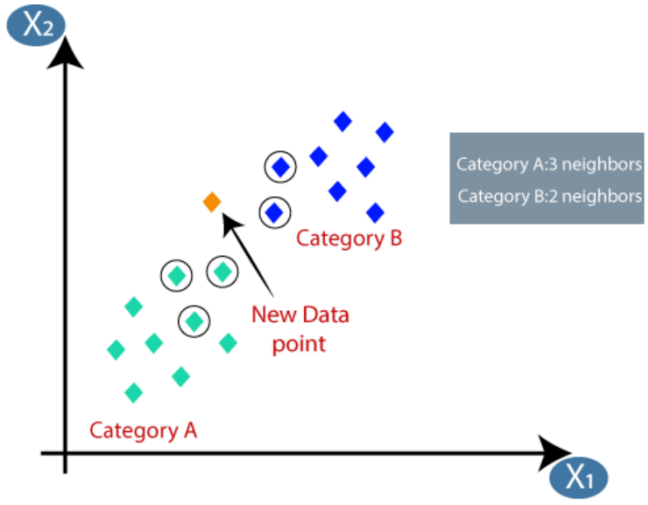


Figure 8: Binary classification diagram using KNN

II.3.6 Decision trees

A well-liked machine learning approach called decision trees is utilized for both classification and regression problems. The technique creates a model that resembles a tree, with core nodes standing in for feature tests, the branches representing a decision rule, and leaf nodes for the projected class name or value.

The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in a recursive manner called recursive partitioning. This flowchart-like structure helps in decision-making. It is visualized like a flowchart diagram which easily mimics human level thinking. That is why decision trees are easy to understand and interpret [14].

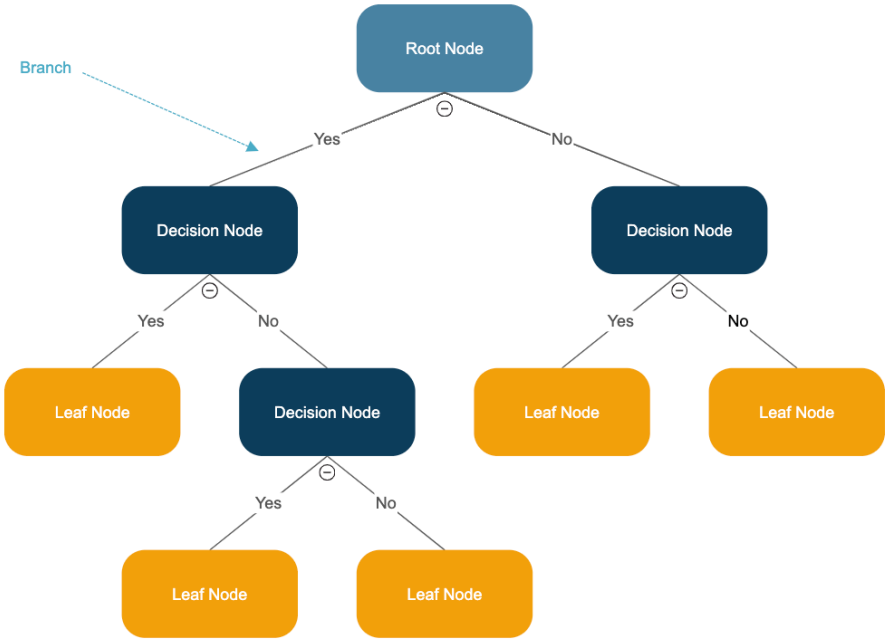


Figure 9: decision tree example classification example of heart failure

A decision tree is a white box type of Machine Learning algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as with neural networks. Its training time is faster compared to a neural network [15].

The time complexity of decision trees is a function of the number of records and attributes in the given data. The decision tree is a distribution-free or non-parametric method which does not depend upon probability distribution assumptions. Decision trees can handle high-dimensional data with good accuracy.[14]

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.[19]

For the next node, the algorithm again compares the attribute value with the other sub-nodes and moves further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

1. Find the best attribute in the dataset using Attribute Selection Measure (ASM).
2. Make that attribute a decision node and break the dataset into smaller subsets.
3. Generate the decision tree node, which contains the best attribute.
4. Recursively make new decision trees using the subsets of the dataset created in step -2. Continue this process until a stage is reached where you cannot further classify the nodes and call the final node as a leaf node.
5. To stop the process one of these three conditions must be met:

- All the tuples belong to the same attribute value.
- There are no more remaining attributes.
- There are no more instances.

Figure 10 explains the process of making a decision tree algorithm:

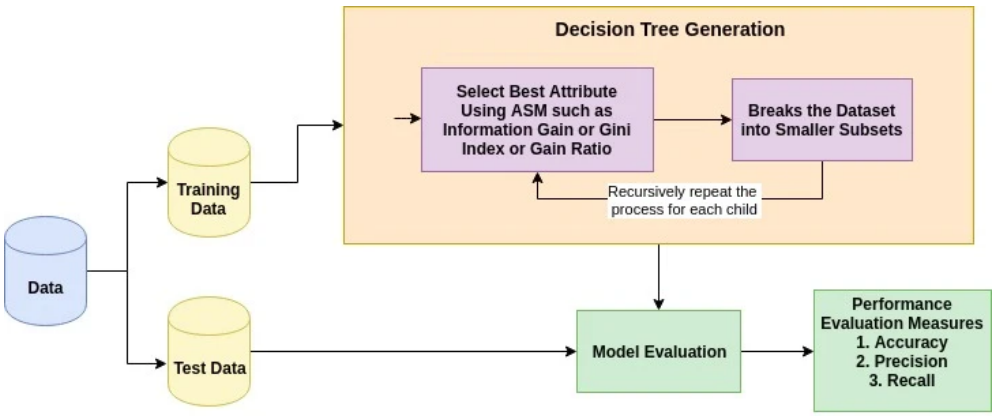


Figure 10: Broad view of training decision tree models

The decision tree relies on multiples measures to do its attribute selection measure in short ASM, the most critical measures are:

- **Information gain:** it is an impurity-based criterion that uses the entropy measure that originates from information theory as the impurity measure (s.t: En is the entropy).

$$infoGain(a_i, S) = En(y, S) - \sum_{v_i, j \in dom(a_i)} \frac{|\sigma a_i = v_{i,j} S|}{|S|} . En(y, \sigma a_i = v_{i,j} S) \quad \text{(eq.4)}$$

Or simply:

$$infoGain(a_i, S) = En(S) - [(Wa) \times En(each\ feature)] \quad \text{(eq.5)}$$

S.t:

$$Wa = WeightedAverage = \left(\frac{\left| \sigma a_i = v_{i,j} S \right|}{|S|} \right) \quad \text{(eq.6)}$$

The weighted average is the probability that an arbitrary tuple in the training set S belongs to a class C.

Where:

$$En(y,S) = \sum_{c_j \in dom(y)} - \frac{\left| \sigma y = c_j S \right|}{|S|} . \log_2 \left(\frac{\left| \sigma y = c_j S \right|}{|S|} \right) \quad \text{(eq.7)}$$

- **Gini index:** Gini index is an impurity-based criterion that measures the divergences between the probability distributions of the target attribute's values. The Gini index has been used in various works such as Breiman's paper [19], and it is defined as:

$$Gini(y,S) = 1 - \sum_{c_j \in dom(y)} \left(\frac{\left| \sigma y = c_j S \right|}{|S|} \right)^2 \quad \text{(eq.8)}$$

In the case of a discrete-valued attribute, the subset that gives the minimum gini index for that chosen is selected as a splitting attribute. In the case of continuous-valued attributes, the strategy is to select each pair of adjacent values as a possible split point, and a point with a smaller gini index is chosen as the splitting point [14].

Consequently the evaluation criterion for selecting the attribute a_i is defined as:

$$GG\left(a_i, S\right) = Gini(y,S) - \sum_{v_i, j \in dom\left(a_i\right)} \frac{\left| \sigma a_i = v_{i,j} S \right|}{|S|} . Gini\left(y, \sigma a_i = v_{i,j} S\right) \quad \text{(eq.9)}$$

(s.t: GG is gini gain)

And thus, the attribute with the minimum Gini Index is chosen as the splitting attribute.

II.3.7 Random forests

Random Forests were introduced by Leo Breiman's paper [17], which was inspired by earlier work by Amit and Geman in the paper referenced in [18]. Random Forests can be used for either a categorical response variable, or a continuous response. Similarly, the predictor variables can be either categorical or continuous [16]. In short, it is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting [20].

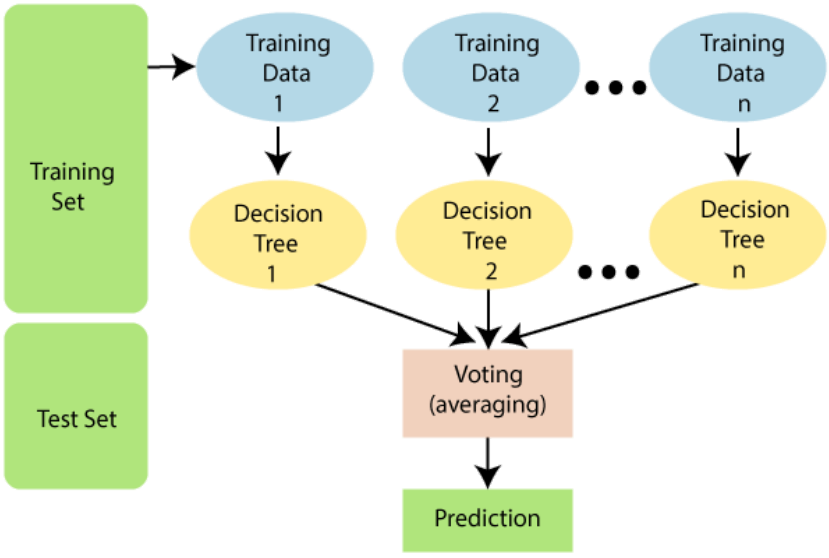


Figure 11: Ensemble learning for random forests

Random forests work in few steps:

1. Data sampling: Random Forests randomly select subsets of the training data with replacement (known as bootstrap samples) to create multiple training sets.

2. Building decision trees: For each bootstrap sample, a decision tree is constructed by recursively partitioning the data based on different features and split points. The splitting is done by maximizing information gain or minimizing impurity measures like Gini index or entropy.

3. Random feature selection: At each node of the decision tree, a random subset of features is considered for determining the best split. This random feature selection helps to decorrelate the trees and reduce overfitting.

4. Tree ensemble: After building a set of decision trees, predictions from each tree are combined using majority voting for classification or averaging for regression. This ensemble approach improves the accuracy and generalization of the model.

5. Prediction: To make predictions on unseen data, the input data is passed through each decision tree in the forest, and the final prediction is obtained by aggregating the individual tree predictions.

Random forests are capable of performing both Classification and Regression tasks and are capable of handling large datasets with high dimensionality, while its ensemble learning nature helps enhance its accuracy and combat overfitting.

II.3.8 Fundamentals of deep learning

II.3.8.1 Overview of deep learning

Deep learning is a sub-field of machine learning and artificial intelligence as a whole, it is based on the concept of artificial neural networks which is a mere imitation of the human brain and the neurons within. Artificial neural networks with numerous layers are trained and used in the deep learning discipline, which enables them to learn complicated representations and patterns from vast quantities of data.

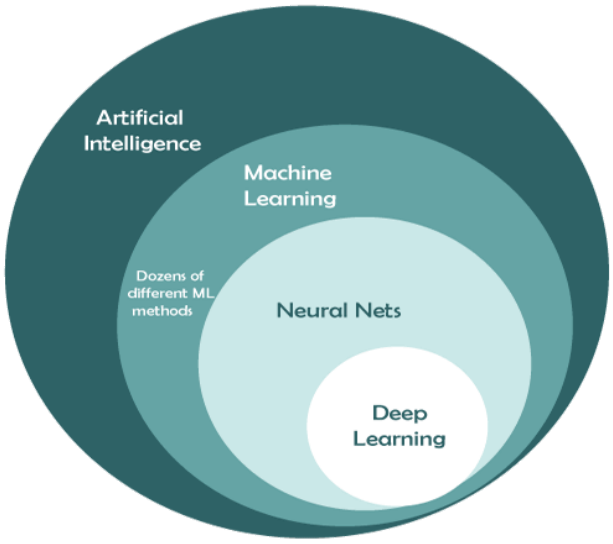


Figure 12: Artificial intelligence and its subsets

In deep learning, neural networks are built with a number of hidden layers, which allows them to pick up on hierarchical data representations. The network may learn increasingly abstract and meaningful representations of the input data since each layer takes and improves characteristics from the layer before it. Deep learning is very useful for applications like speech and images recognition and natural language processing, because it can automatically learn and extract features.

A highly potent framework for supervised learning is offered by contemporary deep learning. A deep network can depict functions that are increasingly sophisticated by adding layers and units inside levels. Given sufficiently big models and sufficiently large datasets of labeled training instances, deep learning may be used to quickly convert an input vector to an output vector [21].

II.3.8.2 Deep feedforward neural networks

Deep feedforward networks, also often called feedforward neural networks, or multilayer perceptrons (MLPs), are the quintessential deep learning models. The goal of a feedforward network is to approximate some function f^*, that describes the relation between inputs and response variables.

For example, for a classifier, $y = f^{*}(x)$maps an input x to a category y. A feedforward network defines a mapping y = f(x, θ) and learns the value of the parameters θ that result in the best function approximation.

These models are called feedforward because information flows through the function being evaluated from x towards the output y(through the intermediate computations used to define f). There are no feedback connections in which outputs of the model are fed back into itself.

The basic structure of an ANN is the artificial neuron, which resembles the biological neuron in its shape and function [23].

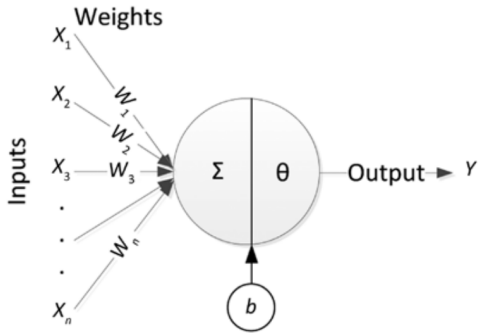


Figure 13: graphical representation of a neuron and its parameters

Mathematically speaking, a neuron can be represented by the following function:

$$y = \theta\left(\sum_{i=1}^{n} W_i X_i + b\right) \qquad \Rightarrow \qquad y = \theta(W.X + b) \tag{eq.10}$$

Where $\boldsymbol{X}$ is the input vector, $\boldsymbol{W}$ is the weight matrix connecting the input layer to hidden layer, $\boldsymbol{b}$ is the bias vector, and θ is the activation function.

This yields in turn an interconnected structure called a neural network with weighted connections from layer to layer and with each layer having its proper activation function. This process which we described so far is called forward propagation.

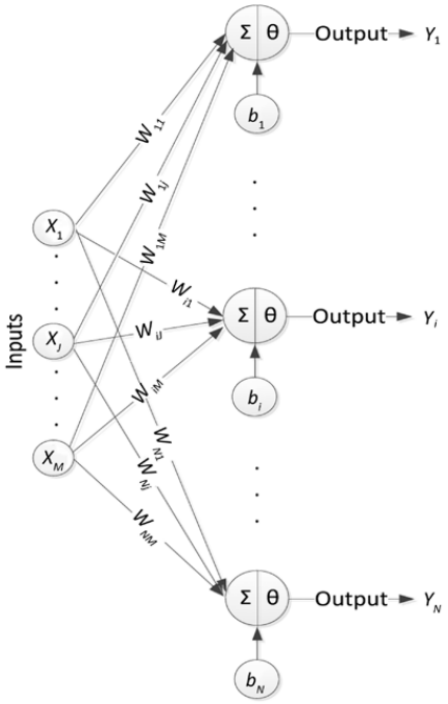


Figure 14: single layered neural network without output

Mathematically, the following equation represents the output y:

$$Y=\begin{bmatrix} Y_1 \\ \vdots \\ Y_i \\ \vdots \\ Y_N \end{bmatrix}=\begin{bmatrix} \theta\left(\sum_{j=1}^{M} W_{1j}X_j+b_1\right) \\ \vdots \\ \theta\left(\sum_{j=1}^{M} W_{ij}X_j+b_i\right) \\ \vdots \\ \theta\left(\sum_{j=1}^{M} W_{Mj}X_j+bM\right) \end{bmatrix}=\theta(W.X+b) \qquad \text{(eq.11)}$$

In artificial neural networks and in feed forward neural networks, the aim is to compare the obtained output to the desired result and adjust the parameters (weights and biases) accordingly. This procedure of constantly predicting and re-adjusting is called backpropagation.

through equation (eq.12), we may update the weights and the same could be done for the biases:

$$w_{new} = w_{old} - \alpha\left(\frac{\partial J}{\partial w}\right) \tag{eq.12}$$

$$J(w^t, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \tag{eq.13}$$

Such that α is the learning rate or the step size at which the weights of the neural networks are updated during the training process. L is the loss function used and J is the average loss.

The goal is to adjust the hyperparameters to minimize the average loss, because the larger J is, the worse the performance gets.

The number of hidden layers and the number of neurons in each layer are two hyperparameters that need to be paid attention to. A visualization of a full deep feedforward neural network is represented in Figure 15 [23]:

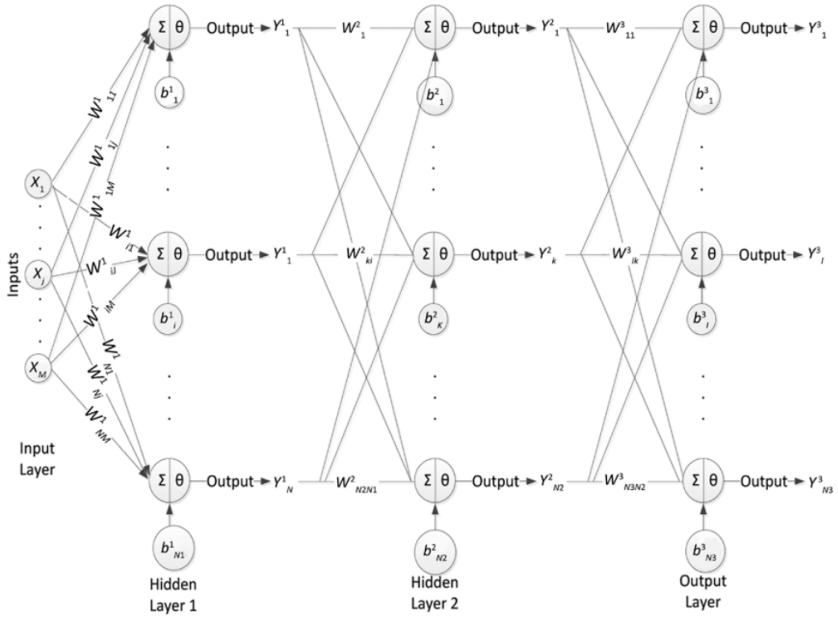


Figure 15: multilayered neural network without output

II.3.8.3 Recurrent neural networks (RNN)

Recurrent neural networks or RNNs are a family of neural networks for processing sequential data. A recurrent neural network is a neural network that is specialized for processing a sequence of values x (1) , . . . , x (τ) . RNN's can scale to much longer sequences than would be practical for networks without sequence-based specialization. Most recurrent networks can also process sequences of variable length [22].

The key characteristic of RNNs is their ability to maintain a hidden state, or memory, which is updated at each time step and serves as a summary of the past inputs. This hidden state allows the network to capture and remember information from previous steps and use it to influence the current prediction or output.

Recurrent neural networks can be built in many different ways. Much as almost any function can be considered a feedforward neural network, essentially any function involving recurrence can be considered a recurrent neural network [22].
Generally, we would represent an RNN hidden state by the equation:

$$h^{(t)} = f\left(h^{(t-1)}, x^{(t-1)}, \theta\right) \qquad \text{(eq.14)}$$

S.t: h(t) represents the state of a hidden layer h at instant t.

When the recurrent network is trained to perform a task that requires predicting the future, the network typically learns to use h(t) as a kind of lossy summary of the past sequence of inputs up to t. This summary is in general necessarily lossy, since it maps an arbitrary length sequence (x (t) , x (t–1) , x (t–2) , . . , x (2) , x (1)) to a fixed length vector h(t) [22]. Figure 16 shows how a recurrent neural network is structured.

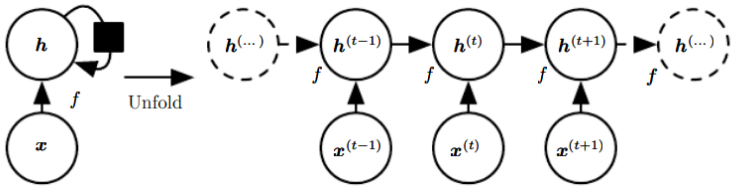


Figure 16: unfolded graph of a recurrent neural network (RNN)

In this network the outputs are not included, it just processes information from the input $\boldsymbol{x}$ by incorporating it in the state $\boldsymbol{h}$ and passing it forward through time. In the left diagram, the black box indicates a delay of a single time step. While the right diagram is simply the unfolded computational graph of the left one.

- **Advantages of RNNs:**

- RNNs are well-suited for tasks involving sequential data, such as natural language processing, speech recognition, and time-series analysis. They can effectively capture temporal dependencies and patterns in the data.

- RNNs can handle inputs and outputs of variable lengths, making them flexible for tasks where the length of the sequences varies.

- RNNs have a hidden state that allows them to retain information from previous inputs, enabling them to maintain a memory of the context and make informed predictions.

- RNNs can process data in an online or streaming fashion, making them suitable for real-time prediction tasks where predictions are required immediately.

- RNNs are translationally invariant, meaning they can recognize patterns regardless of their position in the sequence. This property makes them robust to slight variations in input data.

- **Disadvantages of RNNs**

- RNNs can suffer from the vanishing gradient problem, where the gradients become extremely small or vanish as they propagate through many time steps, making it difficult to capture long-term dependencies. Similarly, gradients can explode and cause unstable training.

- RNNs can be computationally expensive to train and evaluate, especially for long sequences or complex tasks, due to the need to process data sequentially and maintain hidden states.

- RNNs inherently process data sequentially, which limits parallelism and can result in slower training and inference compared to other architectures.

- Although RNNs can capture short-term dependencies well, they may struggle to capture long-term dependencies that span a large number of time steps, even with techniques like LSTM and GRU.

- RNNs are sensitive to the ordering of input data, and variations in the input order can affect their performance. This sensitivity can make it challenging to model tasks where the order of the input is not meaningful.

II.3.8.4 Long short term memory (LSTM) networks

Numerous attempts were made in the 1990s to address the problem of vanishing gradients for RNNs. These included non-gradient based training algorithms, such as simulated annealing and discrete error propagation, and the focus of this section, “LSTM”.

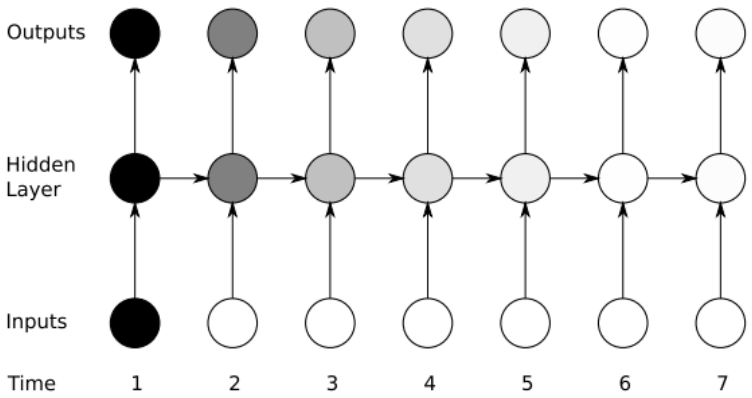


Figure 17: The vanishing gradient problem for RNNs

The LSTM architecture consists of a set of recurrently connected subnets, known as memory blocks. These blocks can be thought of as a differentiable version of the memory chips in a digital computer. Each block contains one or more self-connected memory cells and three multiplicative units: the input, output and forget gates, that provide continuous analogues of write, read and reset operations for the cells [24].

An LSTM network is the same as a standard RNN, except that the summation units in the hidden layer are replaced by memory blocks, as illustrated below.

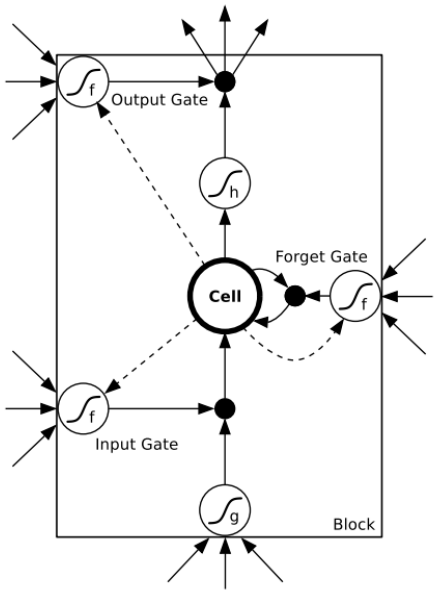


Figure 18: LSTM memory block with one cell

The multiplicative gates allow LSTM memory cells to store and access information over long periods of time, thereby mitigating the vanishing gradient problem. For example, as long as the input gate remains closed (i.e. has an activation near 0), the activation of the cell will not be overwritten by the new inputs arriving in the network, and can therefore be made available to the net much later in the sequence, by opening the output gate.[23]

As opposed to RNNs, the preservation over time for LSTMs would look similar to the figure Figure 19.

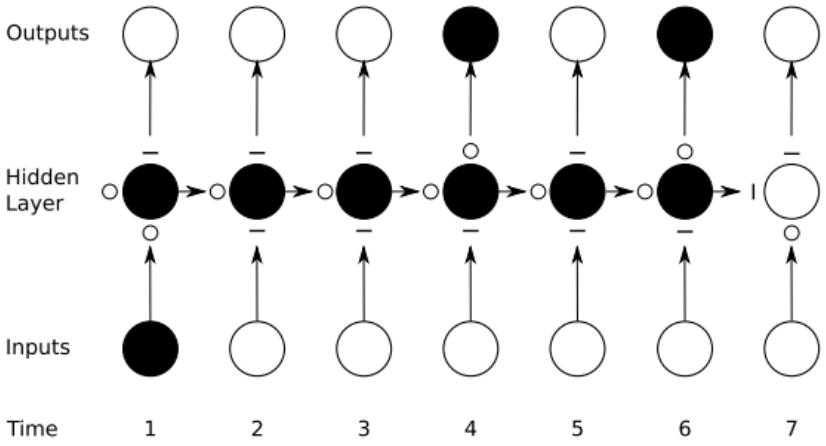


Figure 19: Preservation of gradient information by LSTM

II.4 Performance evaluation of learning models

II.4.1 confusion matrix

A confusion matrix also known as error matrix is a tabular representation of prediction outcomes of any classifier, which is used to describe the performance of the classification model on a set of test data when true values are known. In the matrix, columns are for the prediction values, and rows specify the actual values.

The confusion matrix helps in understanding the model's performance by showing the number of correct and incorrect predictions for each class. It provides valuable information for computing various evaluation metrics such as accuracy, precision, recall, and F1 score [25].

As an example of a binary classification confusion matrix:

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

Figure 20: confusion matrix of a binary classifier

II.4.2 Accuracy and error accuracy

Accuracy measures the proportion of correctly classified instances out of the total number of instances in a dataset. It is a simple and intuitive metric that indicates the overall correctness of the model's predictions. While the error accuracy is the exact opposite, by finding the rate of incorrectly classified instances over the whole data.

$$Accuracy = \frac{\#\ correct\ predictions}{\#\ predictions} \quad \text{(eq.15)}$$

$$Error\ accuracy = \frac{\#\ incorrect\ predictions}{\#\ predictions} = 1 - Accuracy \quad \text{(eq.16)}$$

II.4.3 Precision

The precision metric is used to overcome the limitation of Accuracy. The precision determines the proportion of positive predictions that was actually correct. It can be calculated as the True Positive or predictions that are actually true to the total positive predictions (True Positive and False Positive). In other words, it calculates the ratio of the number of correct predictions of classA over all the predictions of classA including both right and wrong predictions.

$$Precision = \frac{TP}{TP + FP} \quad \text{(eq.17)}$$

II.4.4 Recall (sensitivity)

It is also similar to the Precision metric; however, it aims to calculate the proportion of actual positives that was identified incorrectly. It can be calculated as True Positive or predictions that are actually true to the total number of positives, either correctly predicted as positive or incorrectly predicted as negative (true Positive and false negative).

$$Recall = \frac{TP}{TP + FN} \quad \text{(eq.18)}$$

II.4.5 F1-score

F-score or F1 Score is a metric to evaluate a classification model on the basis of predictions that are made for the positive class. It is calculated with the help of Precision and Recall. It is a type of single score that represents both Precision and Recall. So, the F1 Score can be calculated as the harmonic mean of both precision and Recall, assigning equal weight to each of them.

$$F1\_score = 2 \times \left(\frac{precision \times recall}{precision + recall} \right) \quad \text{(eq.19)}$$

II.5 Performance optimization techniques

II.5.1 K-fold cross validation

Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. It could also be said that it is a technique to check how a statistical model generalizes to an independent dataset and to the performance on the actual test set.

K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called folds. For each learning set, the prediction function uses k-1 folds, and the rest of the folds are used for the test set.

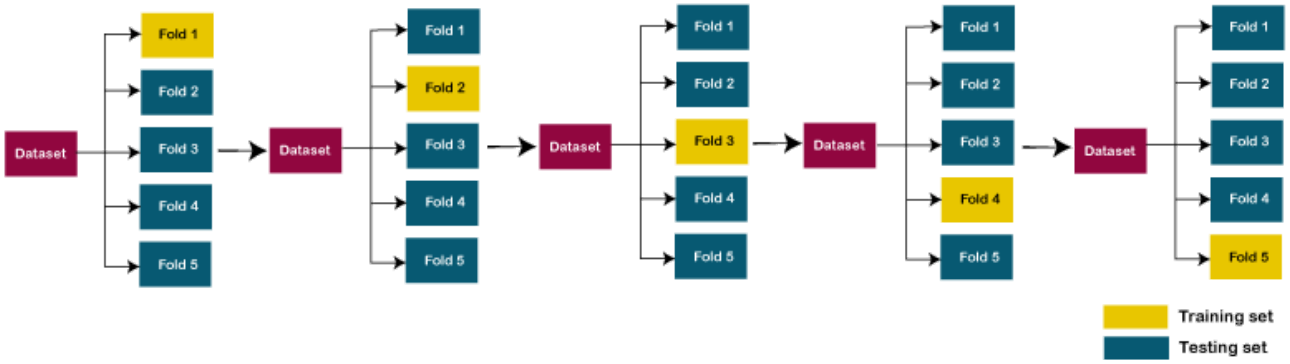


Figure 21: A five folds cross validation process

II.5.2 Regularization

Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it. This technique can be used in such a way that it will allow to maintain all variables or features in the model by reducing the magnitude of the variables. Hence, it maintains accuracy and a generalization of the model.

This could be done using multiple techniques such as:

- Lasso Regularization (L1 regularization)
- Ridge Regularization (L2 regularization)
- Elastic Net Regularization (L1 and L2 regularization)

Regularization techniques help prevent overfitting by controlling the model's complexity. By adding a regularization term to the loss function, the model is discouraged from relying too heavily on specific features or fitting noise in the training data.

II.5.3 Principle component analysis

Principal component analysis is an unsupervised learning approach that is used in machine learning to reduce dimensionality. With the use of orthogonal transformation, it is a statistical procedure that transforms the observations of correlated characteristics into a collection of linearly uncorrelated data. The Principal Components are these newly altered features. It is considered one of the widely used tools for exploratory data analysis

and predictive modeling. In short, it is a method for identifying significant patterns in the provided dataset by lowering the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data. It works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality.

II.6 conclusion

In this chapter, Theoretical aspects of machine learning and deep learning and a variety of their classification algorithms were discussed.

The strong points of each of those algorithms and where they may fall short were also highlighted. This chapter also explained how to evaluate our models and then how their performance could be optimized.

CHAPTER III: AI on Embedded systems

III.1 Introduction

Embedded systems are one of, if not the most vital part of any industrial chain in the digitalized era. They provide a platform to program complex operations on microchips, in order to run and manage entire productions. For decades, hardware simply could not keep up with the rapid advancement of mathematical and statistical theory, as well as the early development of artificial intelligence in the late 1950's. It was the case up until the late 2000's where we've seen the surge of more powerful processors and microcontrollers, which enabled developers to implement those early concepts in practice.

III.2 Advantages of AI on embedded systems

The idea of implementing artificial intelligence models on low cost embedded systems seemed good especially because it introduces intelligent capabilities to edge devices. This brings multiple crucial points such as:

- Achieving real time interference and reducing latency.

- Locally processing data enhances privacy and security which minimizes reliance on cloud-based services.

- Allowing for independence from the internet (offline) .

- eliminates the need for continuous data transmission to the cloud, reducing bandwidth costs and reliance on cloud infrastructure which achieves better cost effectiveness.

III.3 Key techniques of Embedded AI

Embedded systems are resource-constrained by nature raising two main questions:

- How to handle complex artificial intelligence models with their hardware limitations?
- How to support those models.

In this section we will explain two crucial techniques to settle these challenges, with their main focus being optimizing the algorithms and maintaining good performance.

III.3.1 Model compression

III.3.1.1 Network structure redesign

It is the practice of creating new network structures in order to improve on the already existing designs. SqueezeNet, MobileNet and Once-For-All network (OFA)..etc were some redesign proposals which we are going to explore briefly to give an idea about how the redesign works.

- **SqueezeNet:** Landola et al [30], proposed this lightweight network that maintains accuracy using fewer parameters. SqueezeNet consists of two parts: a convolutional neural network architecture designed by the authors, and the Fire module. Three main techniques were used to maintain accuracy: using 1 × 1 filters instead of partial 3 × 3 filters, using a squeeze layer to reduce the input channels of 3 × 3 filters, delaying downsampling (postponing the downsampling process to the end of the network).

Figure 22: SqueezeNet network structure

- **MobileNet:** Howard et al [31], proposed MobileNet, a lightweight network for mobile and embedded vision applications. Two global hyperparameters were introduced in this design, α (Width Multiplier) and ρ (Resolution Multiplier), that can be balanced in terms of latency and accuracy. The core components of MobileNet include depthwise separable convolution. We decompose the standard convolution as follows:

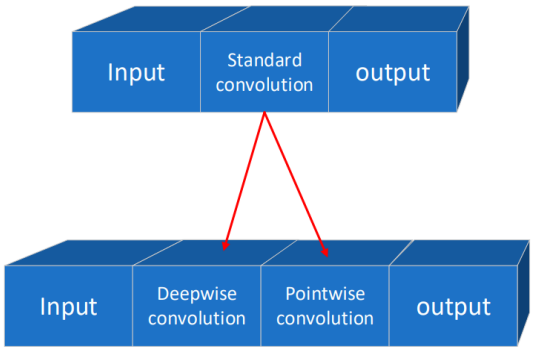


Figure 23: Depth separable convolution

III.3.1.2 Quantization

In embedded AI, compressing floating point in neural networks parameters to reduce complexity and size is called quantization. It achieves those goals by reducing the number of bits used by floating point numbers, while also respecting the accuracy. The quantization step is an iterative process to achieve acceptable accuracy of the network.

It is customary to express the weights and biases of a neural network as 32-bit floating-point values, which gives the neural network a high degree of precision and eventually, accuracy.

Quantization thus, converts a neural network from using the standard precision to a reduced precision such as 8-bit integers. This can also lead to obtaining less latency and better power efficiency.

III.3.1.3 Pruning

Pruning is a technique used in neural networks to eliminate duplicate input by assessing the relevance of each unit and deleting unnecessary bits. One technique discussed in [32], consists of training the neural network to learn the important connections then pruning unimportant connections, it is then followed by retraining the network to adjust the weights of the remaining connections.

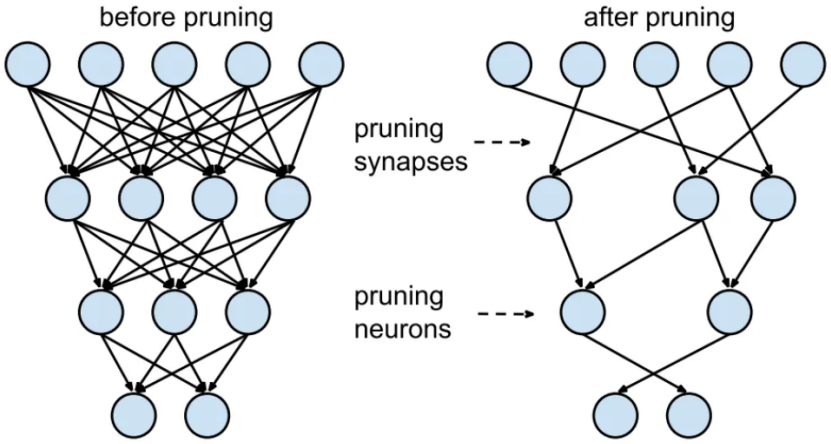


Figure 24: A neural network before and after pruning

III.3.2 neural networks binarization

Neural networks, with their large size and numerous parameters, pose challenges for resource-constrained devices. To address this, binarization methods have been proposed. These methods combine weights and activations into fixed-point parameters, leading to memory and inference time savings [29]. Binarized neural networks are similar to feedforward networks, but their weights and activations are constrained to 1 or -1. As a result, the sign function is often used as the activation function, as it provides binary outputs instead of other activation functions (e.g., relu or sigmoid) [29].

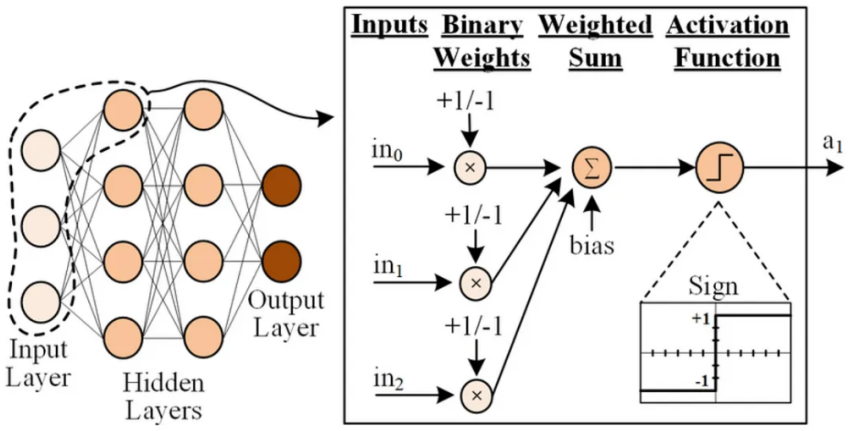


Figure 25: a binarized neural network structure (BNN)

III.4 ESP-32 System-on-Chip

III.4.1 Overview

In this project we made use of the ESP32-S3 as our embedded system or SoC of choice.
The ESP32-S3 is a low cost, low-power MCU-based system-on-chip (SoC). It was designed by Espressif Systems, built for mobile, wearable electronics and internet of things (IoT) applications.
It features outstanding characteristics for low-power chips with integrated 2.4 GHz Wi-Fi and Bluetooth Low Energy (Bluetooth LE). It consists of a high-performance dual-core microprocessor (Xtensa 32-bit LX7), a low power coprocessor, a Wi-Fi baseband, a Bluetooth LE baseband, RF module, and numerous peripherals.

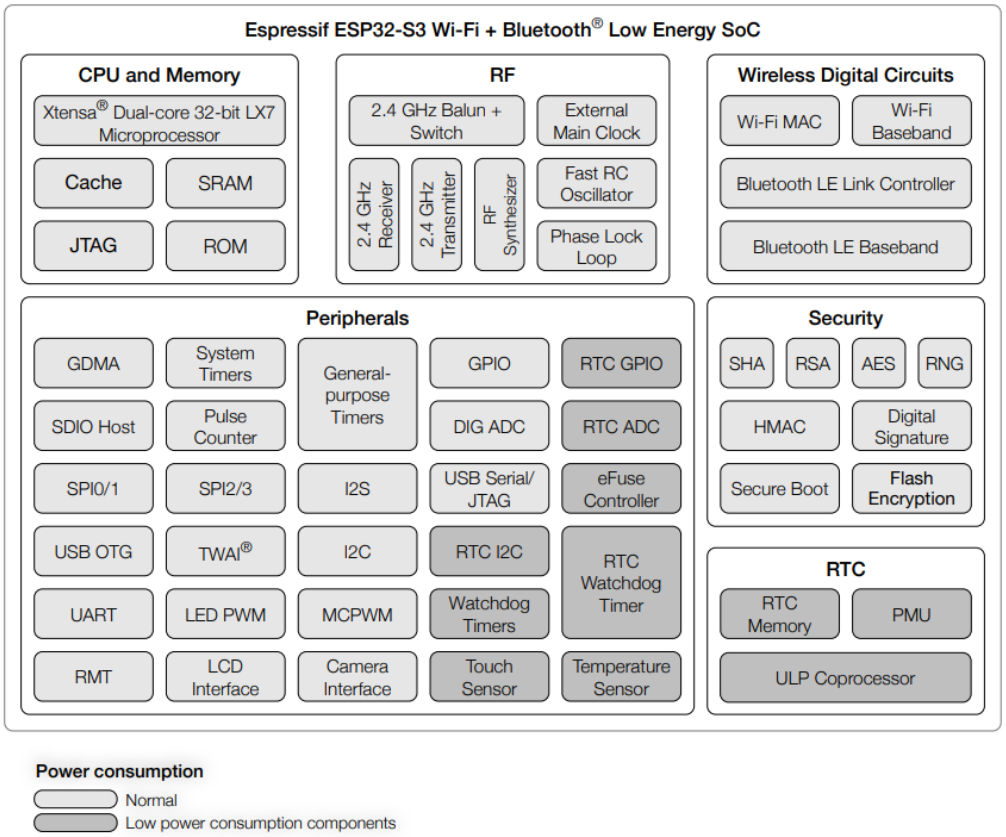


Figure 26: ESP32-S3 Functional block diagram representing a variety of its peripherals

We will focus in this report more on the characteristics that are relevant to our work.

Table 1: ESP32-S3 main features

CPU: ● Xtensa® dual-core 32-bit LX7 microprocessor, up to 240 MHz ● CoreMark® score 2 cores at 240 MHz: 1181.60 CoreMark; 4.92 CoreMark/MHz
Memory: ● 128-bit data bus and SIMD commands ● 384 KB ROM • 512 KB SRAM ● 16 KB SRAM in RTC ● SPI, Dual SPI, Quad SPI, Octal SPI, QPI and OPI interfaces that allow connection to multiple flash and external RAM ● Flash controller with cache is supported ● Flash in-Circuit Programming (ICP) is supported
Power management: ● Power Management Unit with five power modes ● Ultra-Low-Power (ULP) coprocessors: - ULP-RISC-V coprocessor - ULP-FSM coprocessor
Peripherals interfaces: ● 45 × programmable GPIOs ● Digital interfaces: - 4 × SPI - 1 × LCD interface (8-bit ~16-bit parallel RGB, I8080 and MOTO6800), supporting conversion between RGB565, YUV422, YUV420 and YUV411 - 1 × DVP 8-bit ~16-bit camera interface - 3 × UART - 2 × I2C - 2 × I2S - 1 × RMT (TX/RX) - 1 × pulse counter - LED PWM controller, up to 8 channels - 1 × full-speed USB OTG - 1 × USB Serial/JTAG controller

III.4.2 Application of the ESP32

With low power consumption and decent characteristics compared to similar SoC's, ESP32-S3 is an ideal choice for IoT application in the following areas:

- Smart Home
- Industrial Automation
- Health Care
- Consumer Electronics
- Smart Agriculture
- POS machines
- Service robot
- Audio Devices
- Generic Low-power IoT Sensor Hubs
- Generic Low-power IoT Data Loggers
- Cameras for Video Streaming
- USB Devices
- Speech Recognition
- Image Recognition
- Wi-Fi + Bluetooth Networking Card
- optimized AI model training & testing

III.5 Conclusion

In this chapter we discussed the advantages of deploying AI models on embedded systems and the challenges we face in doing so. We also present common techniques needed to transition to resource limited devices. These techniques are mostly to optimize the models and limit memory and power usage without trading off the model's performance.

We also discussed the popular system-on-chip ESP32-S3, its characteristics and its application, which makes it a suitable device to test our models on.

CHAPTER IV: Fault diagnosis on ESP32

IV.1 Introduction

Automated condition monitoring is a procedure that counts on surveilling and observing the changes that occur in the system's vital parameters [26], such as electrical current, acoustic emissions, vibration signals, and thermal images [27]. This approach generally considers generated signals, such as vibration, acoustic, and temperature as health indicators of the monitored machine or equipment state [28].

In this chapter we will apply all what we have previously discussed. We will attempt to design, train and test machine learning and deep learning models. We will then compare them in terms of performance, in order to pick the most suitable for deployment on an embedded system.

IV.2 Experimental framework

The adopted dataset was chosen from MAFAULDA machinery fault database [33]. It contains different types of signals consisting of information about several types of simulated faults encountered in real industrial situations. Experiments were performed on a SpectraQuest's Machinery Fault Simulator (MFS) Alignment–Balance–Vibration (ABVT), and the diagnosis of degree of imbalance of motor rotation stability, was chosen for this thesis as the target failure.

Technical specification of this test bench could be found in Table 2 and the data acquisition system elements are explained in Table 3 . It should also be mentioned that only early stages of failure with 49 increasing rotation frequencies have been taken in the aim to test the sensitivity of the used methods [28].

The sequences were generated at a sampling rate of 50 kHz for 5 seconds, thus generating 250.000 data points per sequence.

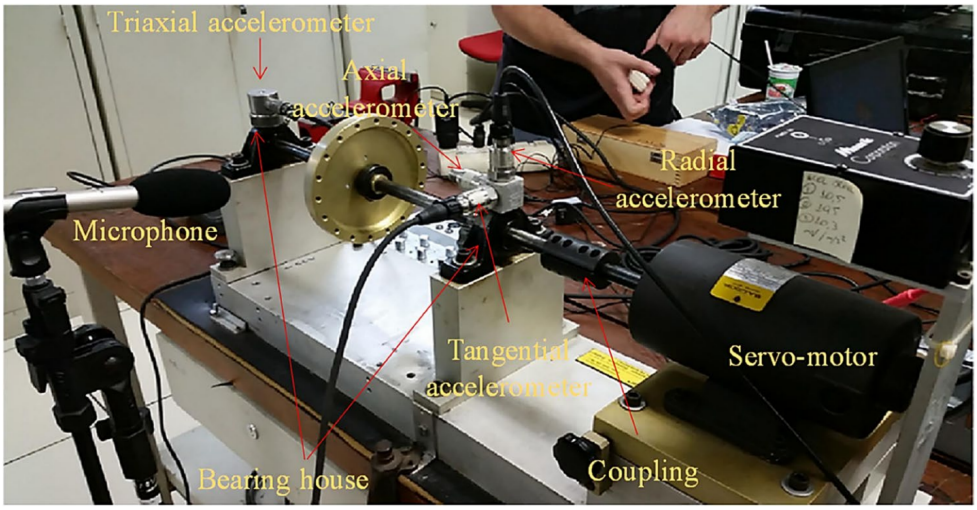


Figure 27: SpectraQuest experimental simulations test bench

In the full dataset we have 7 machine states: normal state, 6g unbalance, 10g unbalance, 15g unbalance, 20g unbalance, 25g unbalance, 30g unbalance.

Each state contains 8 features for the 8 available sensors, and has sequences sampled with a rate of 50 kHz for 5 seconds. Data is obtained for 49 increasing rotation frequencies, which yields 12.250.000 data points for each state.

Since we have 7 classes, a full concatenated dataset containing all the sequences from all the classes is approximately of the size 84.250.000 data points × 8 features.

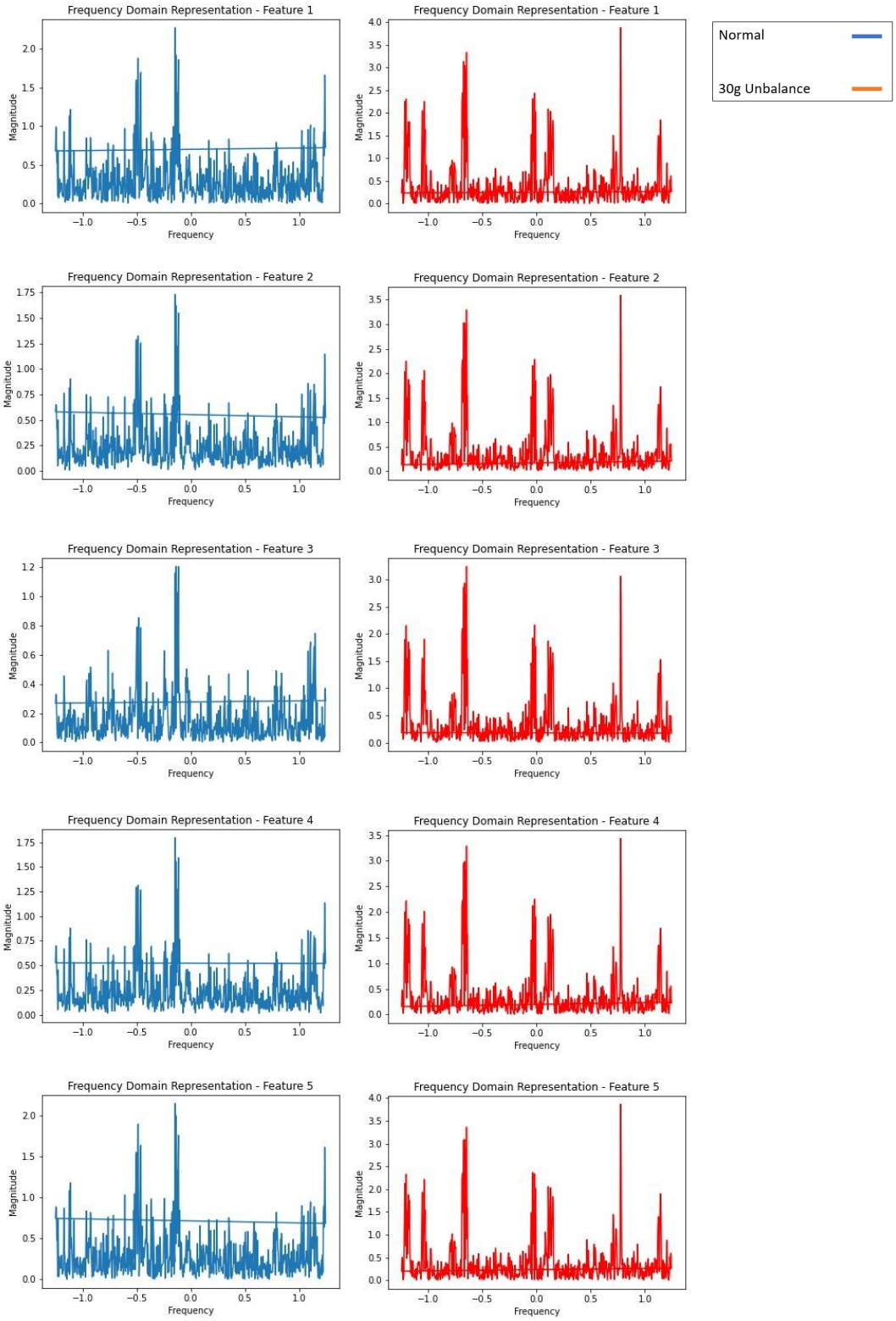
Table 2: Test bench specifications

Specifications	Values	Unit
Motor	1/4	CV DC
Frequency range	700 - 3600	rpm
System weight	22	kg
Axis diameter	16	mm
Axis length	520	mm
Rotor	152.4	mm
Bearing distance	390	mm

Table 3: Data acquisition system specifications

Secnos's type	Sensibility	Frequency range	Measurement range
Three (03) Industrial IMI Sensors, Model 601A01 accelerometers on the radial, axial, and tangential direction	(±20%) 100 mV per g (10.2 mV per m/s2)	(±3 dB) 16–600,000 CPM (0.27–10.000 Hz)	±50 g (±490 m/s2)
One IMI Sensors triaxial accelerometer, Model 604B31, returning data over the radial, axial, and tangential directions	(±20%) 100 mV per g (10.2 mV per m/s2)	(±3 dB) 30–300,000 CPM (0.5–5.000 Hz)	±50 g (±490 m/s2)
Monarch Instrument MT-190 analog tachometer	-	-	-
Shure SM81 microphone	-	20–20.000 Hz	-
Two (02) National Instruments NI 9234; 4 Channel analog acquisition modules	-	Sampling frequency of 51.2 kHz	-

We will display and compare the graphical representation of all features of normal data and 30g unbalanced data, in the frequency domain.



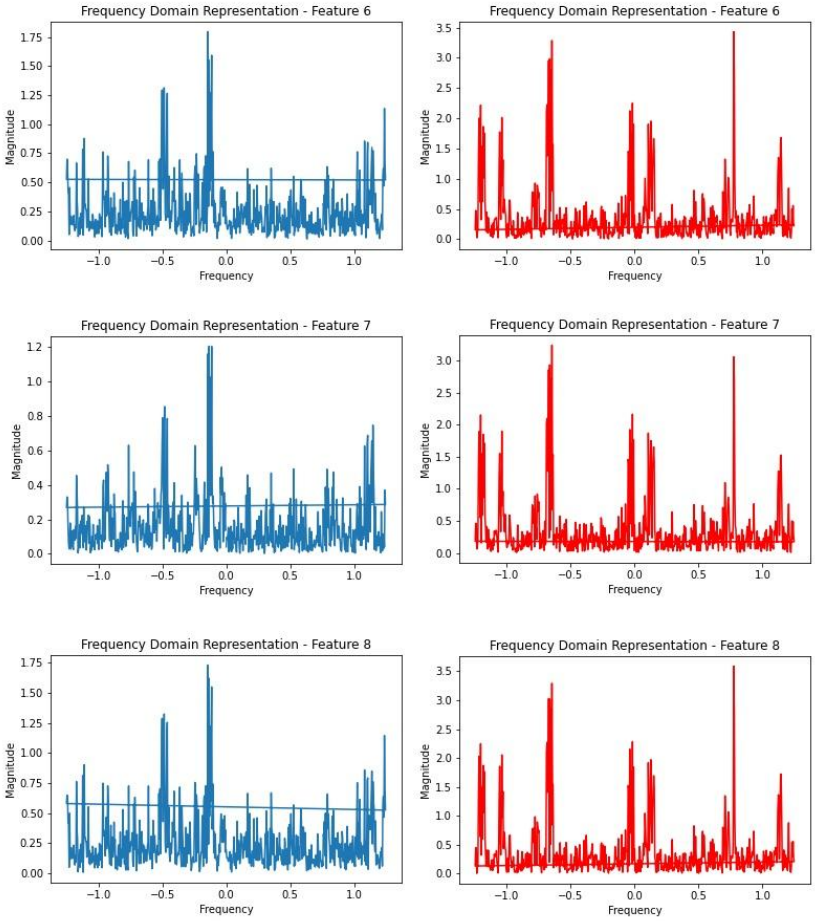


Figure 28: frequency-domain plots of features from normal and 30g unbalance data

We notice from these graphs of the 30g unbalance the presence of abnormal magnitudes in certain frequencies, which are not present in the normal data. We can also notice a periodic pattern, where the magnitude spikes frequently and regularly in a periodic way.

We will then perform feature extraction to transform our dataset into a more useful format, which will then be used to find the dataset properties and pattern through AI, to then exploit them for classification.

IV.3 Feature extraction and data transformation

Feature extraction is the process of selecting and transforming raw data into a reduced and meaningful representation of information that captures relevant information used to solve a problem.

In this project we down sampled our entire dataset by 1/20.000 in order to overcome overfitting and to simulate a more realistic dataset. Since the dataset is artificially induced, by reducing the size of the dataset we will attempt to emulate a dataset that is obtained in real scenarios. We obtain a dataset that is 4210 rows × 8 columns.

IV.3.1 Fast Fourier Transform

Fast fourier transform (FFT) converts data from time to frequency domain, taking a signal over a period of time into its frequency components. These frequency components are the signal's sinusoidal constituents at their respective frequency and amplitudes.

An FFT in short is a fast and efficient method to calculate the Discrete Fourier Transform (DFT), and it is represented mathematically by the equation:

$$X_k = \sum_{n=0}^{N-1} x_n e^{\left(\frac{-i2\pi kn}{N}\right)} \quad , \quad k = 0, \ldots, N-1 \qquad \text{(eq.20)}$$

Where x_n is the input sequence in time domain, X_k is the output sequence in the frequency domain, and N is the number of samples.[34]

IV.3.2 FFT_convolve Function

To get new and more suitable features, we apply a function called FFT_convolve( ), which as the name suggests, performs convolution between a signal and another. Here's a brief on its steps:

1. The function takes two signals, one considered as the input and the second as the kernel (filter). In our case we take the signal as input and its inverse as the kernel.
2. Zero padding to ensure that both signals used have the same length.
3. We apply FFT on both signals, representing them as complex-valued spectra.
4. Perform convolution between the two signals (elements wise multiplication).

5. Applying inverse FFT to the result yields the convolution of the original signal and the original kernel.
   We performed convolution in the frequency domain for computational efficiency, as it would be increasingly complicated to perform it directly in the time domain.
6. It is optional to truncate the size of the obtained dataset but it might be useful in. The length (number of features) of the output signal in our case is 2N-1 features.

Finally, the obtained dataset is 8419 rows × 15 columns. This is due to element multiplication and an overlap between data caused by reaching the boundaries of the dataset while performing convolution. This can be eliminated using truncation.

IV.3.3 Why use autocorrelation?

Autocorrelation is a statistical technique that measures the similarity between a time series sequence and a lagged version of itself. It can be crucial in identifying trends and patterns in the signal. If we obtain a high correlation, we might say that there is a high similarity between the current state of the signal and a past state. This can be useful for forecasting future values of the series, as well as for identifying the underlying causes of the patterns and trends. Autocorrelation might also help in identifying randomness and noise in a sequence, which would give information on how to deal with the dataset.[35]

IV.4 Models Evaluation

In this section we are going to evaluate the 6 trained models on a variety of metrics, notably: confusion matrix, classification report metrics, test time and size.
For good performance on embedded systems, we need our model to be as light as possible and to have a short test time, on top of good accuracy.

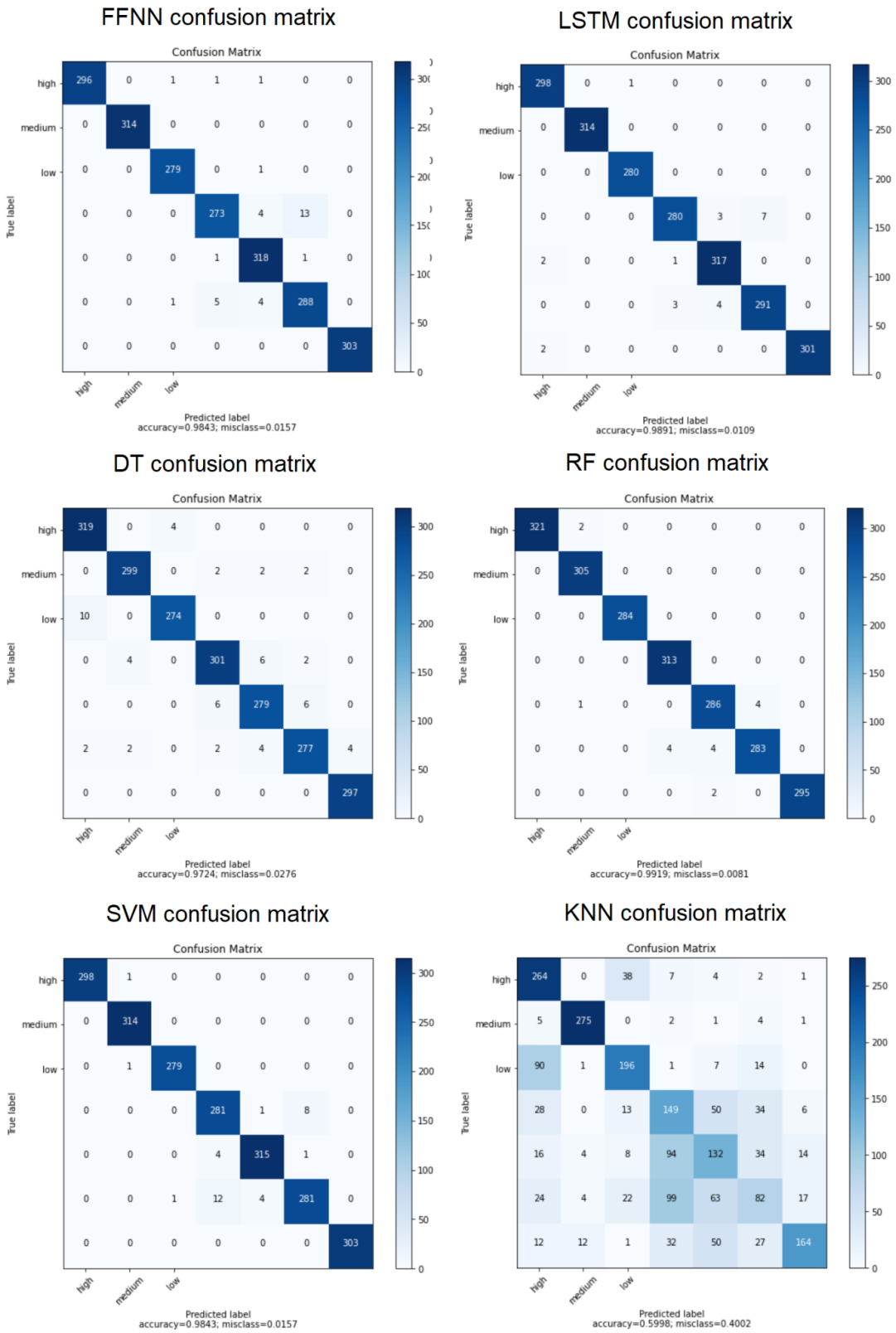


Figure 29: confusion matrices of 6 employed AI techniques for fault diagnosis

Table 4: classification report of FFNN model

class	precision	recall	f1-score
normal	1	0.99	1
6g unbalance	1	1	1
10g unbalance	1	1	1
15g unbalance	0.98	0.89	0.93
20g unbalance	0.99	0.97	0.98
25g unbalance	0.86	0.99	0.92
30g unbalance	1	1	1

Table 4: classification report of LSTM model

class	precision	recall	f1-score
normal	1	1	1
6g unbalance	1	1	1
10g unbalance	1	1	1
15g unbalance	0.95	0.98	0.97
20g unbalance	0.98	1	0.99
25g unbalance	0.97	0.95	0.96
30g unbalance	1	0.99	0.99

Table 5: classification report of decision tree model

class	precision	recall	f1-score
normal	0.96	0.99	0.98
6g unbalance	0.98	0.98	0.98
10g unbalance	0.99	0.96	0.98
15g unbalance	0.97	0.96	0.96
20g unbalance	0.96	0.96	0.96
25g unbalance	0.97	0.95	0.96
30g unbalance	0.99	1	0.99

Table 6: classification report of random forest model

class	precision	recall	f1-score
normal	1	0.99	1
6g unbalance	1	1	1
10g unbalance	1	1	1
15g unbalance	0.98	0.89	0.93
20g unbalance	0.99	0.97	0.98
25g unbalance	0.86	0.99	0.92
30g unbalance	1	1	1

Table 7: classification report of SVM model

class	precision	recall	f1-score
normal	1	0.99	0.99
6g unbalance	1	1	1
10g unbalance	0.98	0.99	0.99
15g unbalance	0.95	0.98	0.97
20g unbalance	1	0.98	0.99
25g unbalance	0.97	0.95	0.96
30g unbalance	1	1	1

Table 8: classification report of KNN model

class	precision	recall	f1-score
normal	0.6	0.99	1
6g unbalance	0.93	0.95	0.94
10g unbalance	0.71	0.63	0.67
15g unbalance	0.39	0.53	0.45
20g unbalance	0.43	0.44	0.43
25g unbalance	0.42	0.26	0.32
30g unbalance	0.81	0.55	0.65

Finally, to decide which model to deploy on the ESP32 we've chosen for this project, we will do a direct comparison between: the weight of the model, test time and accuracy.

Table 9: Comparison of AI models in term of accuracy, test time, weight

	FFNN	LSTM	DT	RF	SVM	KNN
accuracy	0.984	0.989	0.972	0.991	0.984	0.599
test time(ms)	19	27	89	117	21	58
size	86.9 KB	2.45 MB	186 KB	228 KB	304 KB	67 kb

From these results, we can say that the feed forward neural network performs best, and would be the most suitable for a light SoC (the ESP32-S3 in our case).

IV.5 Deployment of the model on the ESP32-S3

There exist many AI models optimization and compression frameworks, each with its own deployment and implementation approach. For this thesis, we used the edge impulse framework.

Edge impulse is a compression framework that reduces the size of the model by using first-order optimization and sparsity-inducing regularization, without compromising prediction accuracy. The framework also provides a user-friendly web interface for uploading and compressing models, along with example codes.

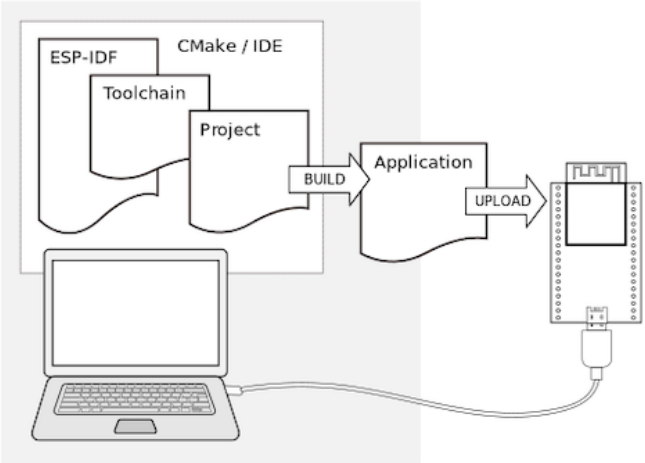


Figure 30: ESP32 application deployment steps and workflow

Using the edge impulse framework, we can obtain our chosen model as an arduino library, which we can take advantage of to test our model.
By uploading our test set we obtain the following confusion matrix

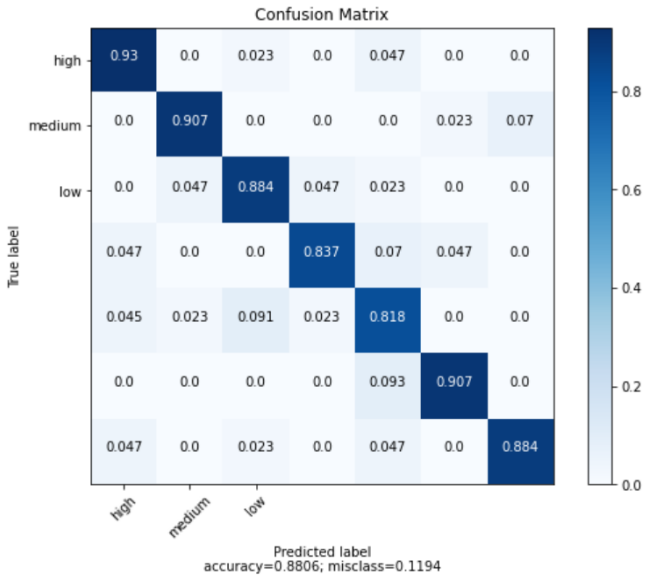


Figure 31: confusion matrix of FFNN for classification deployed on ESP32

From that confusion matrix we can obtain the following classification report:

Table 10: classification report of FFNN for classification deployed on ESP32

class	precision	recall	f1-score
normal	0.83	0.91	0.87
6g unbalance	0.93	0.90	0.91
10g unbalance	0.86	0.82	0.84
15g unbalance	0.92	0.80	0.86
20g unbalance	0.78	0.82	0.80
25g unbalance	0.93	0.91	0.92
30g unbalance	0.90	0.84	0.87

This shows that on a practical level, performing fault diagnosis using a feed forward neural network classifier deployed on an ESP32, is reliable in real world scenarios with an accuracy of 88.06 %.

Although we notice a drop in accuracy, from testing using the full model to testing using the compressed model, it is still considered a reliable model according to industry standards.

It also shows the effectiveness of the compression framework used, by optimizing a fairly large and complex neural network into a model that uses only 4.3 KB of RAM and 116.3 KB of FLASH memory, with a latency of only 25ms.

IV. Conclusion

In this chapter we explored all the steps that led to achieving a fault diagnosis system on a system-on-chip. From presenting the experimental benchmark which allowed us to generate the required data to performing feature extraction and data transformation. We then utilized multiple machine learning and deep learning models to perform fault classification.
Through monitoring accuracy, test time and size, we concluded that the feedforward neural network was performing best. Therefore, we used the compression framework edge impulse to deploy it on an ESP32-S3. Despite the slight drop in accuracy, it is still performing up to the industrial standards.

General conclusion

Equipment in industrial workspaces, regardless of its durability, is prone to failures and malfunctions, which may inevitably lead to financial and possibly human loss. This created the necessity of adapting new techniques, based on data and statistics instead of relying on human intuition or just waiting for the equipment to break in order to repair it. A prime example of heavily used equipment in all industrial fields is the induction motor, which could fail due to a multitude of reasons and should be maintained regularly.

For these reasons, more sophisticated techniques were beginning to be adapted in industrial maintenance such as predictive maintenance and fault diagnosis. These methods are based on artificial intelligence in order to recognize or predict future faults. The use of such techniques in combination with the practical aspect of embedded systems and systems on chips, reduce equipment failure risks and safety hazards significantly.

We started by generating synthetic data by simulating faults on a real and intact induction motor. We then performed basic preprocessing for the data. This led us to using six machine learning and deep learning techniques, namely: SVM, k-nn, decision trees, random forests, feed forward neural networks, and long-short-term memory models.
The models performed mostly well, with the exception of the SVM model and k-nn model which prompted a more thorough optimization of the algorithms. We applied techniques such as data regularization,principal components analysis (PCA), dropout, and grid search before settling for the best results.

The feed forward neural network was chosen for deployment on our embedded system of choice, the ESP32-S3. The tests on our SoC were relatively successful as our final system performed well on new data.

Therefore, we can conclude that the use of deep neural network classification on embedded systems is a useful tool for fault diagnosis on induction motors. Maintenance of such equipment can be improved upon by the use of predictive maintenance, which we could achieve in the future with the appropriate data and artificial intelligence techniques.

References

[1].Moubray, J. (1997). Reliability-centered Maintenance. Industrial Press Inc.

[2].Marc-Aantoine Talva.(2021).The five levels of maintenance. Mobility work.From https://mobility-work.com/blog/maintenance-levels-afnor.

[3].Chekhchoukh Djaafar.(2022).Improving condition-based maintenance of naval propulsion plants using ensemble learning.M'hamed Bougara University, Department of Mechanical engineering.

[4].Derbel, S., Feki, N., Nicolau, F., Barbot, J. P., Abbes, M. S., et al. (2020). Diagnosis methods for mechatronic systems. In Mechatronics 4.0 (pp. 43-55). Hammamet, Tunisia: ffhal-03185167f.

[5].Leslie et al.(2021).Artificial intelligence, human rights, and the rule of law.Alan Turing Institute.

[6].Matt Harrison.(n.d.).Machine Learning Pocket Reference Working with Structured Data in Python.O'Reilly Media.

[7].Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

[8].Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

[9].Marsland, S. (2015). Machine Learning: An Algorithmic Perspective. CRC Press.

[10].Evgeniou, T., & Pontil, M. (2001). Support Vector Machines: Theory and Applications. In Springer eBooks.

[11].JavaTpoint. (n.d.). Machine Learning - Support Vector Machine Algorithm. javatpoint.com. From https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

[12].Cover, T., & Hart, P. (1967). Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory, 13(1), 21-27

[13].Zhang, T. (2010). Pattern Recognition and Machine Learning. Springer.

[14].Avinash Navlani.(2023).Decision Tree Classification in Python Tutorial.datacamp.com.from https://www.datacamp.com/tutorial/decision-tree-classification-python

[15].Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

[16].Adele Cutler, D. Richard Cutler and John R. Stevens.(2011).Random Forests.Ensemble Machine Learning: Methods and Applications (pp.157-176).

[17].Breiman, L. (2001). Random Forests. Machine Learning 45 (1) pp. 5–32.

[18]. Amit, Y., Geman, D.(1997). Shape quantization and recognition with randomized trees. Neural Computation 9(7) pp. 1545-1588 .

[19].Breiman, L.(2001). Bagging Predictors. Machine Learning 24 (2) pp. 123–140 .

[20].JavaTpoint. (n.d.). Machine Learning - random forests Algorithm. javatpoint.com. From https://www.javatpoint.com/machine-learning-random-forest-algorithm

[21].Rene Y. Choi; Aaron S. Coyner; Jayashree Kalpathy-Cramer; Michael F. Chiang; J. Peter Campbell.(2020).Introduction to Machine Learning, Neural Networks, and Deep Learning.tvst.

[22].Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

[23].Mariette Awad;rahul khanna.(2015).Deep Neural Networks.Efficient Learning Machines (pp.127-147)

[24].Alex Graves.(2012).Supervised Sequence Labeling with Recurrent Neural Networks.Springer

[25].Powers, D. M. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. Journal of Machine Learning Technologies, 2(1), 37-63.

[26].Swamidass.PM.(2000).MACHINE RELIABILITY Encyclopedia of Production and Manufacturing Management. Springer, Boston MA.

[27].Li X, Shao H, Lu S, Xiang J, Cai B.(2022).Highly efficient fault diagnosis of rotating machinery under time-varying speeds using LSISMM and small infrared thermal images. IEEE Trans Syst, Man, Cybern: Syst.

[28].Toufik Bettahar;Rahmoune Chemseddine;Djamel Benazzouz. Faults' Diagnosis of Time-Varying Rotational Speed Machinery Based on Vibration and Acoustic Signals Features Extraction, and Machine Learning Methods.Springer.

[29].Zhaoyoun Zhang, Jingpeng Li.(2023).A Review of Artificial Intelligence in Embedded systems.Micromachines.

[30].Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer.(2016).K.J. SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and<0.5 MB model size.arXiv:1602.07360.

[31].Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H.J. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[32].Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural networks. In Advances in Neural Information Processing Systems (Vol. 28).
[33].MaFaulDa - Machinery Fault Database, http://www02.smt.ufrj.br/682~ofshore/mfs/. Accessed 04 Jan 2022.
[34].Frigo, M., & Johnson, S. G. (2005). FFTW: An adaptive software architecture for the FFT. In Proceedings of the IEEE (Vol. 93, No. 2, pp. 216-231). IEEE.
[35].Egor Howell.(2022).Autocorrelation for time series analysis.Towards data science. From https://towardsdatascience.com/autocorrelation-for-time-series-analysis-86e68e631f77
[36].Chapelle, O., Scholkopf, B., & Zien, A. (2009). Semi-supervised learning. MIT Press.
[37].Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT Press.