Leveraging the Enterprise Knowledge Graph for Predictive Maintenance

Ivan Salomone

Supervised by Dr Charlie Abela

Department of Artificial Intelligence

Faculty of ICT

University of Malta

August, 2021

A dissertation submitted in partial fulfilment of the requirements for the degree of M.Sc. in Artificial Intelligence.

First edition, Wednesday 18th August, 2021

Acknowledgements

I would like to express my gratitude to my supervisor, Dr Charlie Abela, for his dedication and continuous guidance. His valuable advice was fundamental to the successful completion of this dissertation.

This work would not have been possible without the support of my family and friends, whom I thank for always being there in times of need. Special thanks goes to my wife, Jeanelle, and my daughters, Kirsten and Julia, for their love, patience and encouragement, and for the sacrifices they had to make to allow me to focus on my studies.

Abstract

Knowledge is an important resource for manufacturing enterprises, hence the collection, storage and analysis of data is a key function within such firms. An important source of data is the Industrial Internet of Things (IIoT), where sensors embedded on industrial machines produce data that can provide insights about the operations and condition of the machines. This data brings about new challenges but also new opportunities. One such opportunity is Predictive Maintenance (PdM) that aims to reduce maintenance costs by maximising the use of the various machine parts and inventory whilst minimising unplanned machine outages. In PdM, machine failures are predicted by monitoring some machine health or performance indicators that come in the form of IIoT data. The challenge, however, is to collect and store the various IIoT data, that are heterogeneous in nature, and to exploit them for PdM. Research shows that Enterprise Knowledge Graphs (EKGs) are flexible data structures that are capable of integrating heterogeneous data, and can thus provide a solution to this challenge.

In this dissertation we investigated the use of the EKG as an integration paradigm for the IIoT data generated by wire bonding machines, and as the foundations of a PdM framework for these machines. The use of the EKG for PdM is scarce in literature, which gave us motive to contribute another case study based on this coupling. We designed and built an ontology modelling the wire bonders, their states, sensors and the observations of these sensors. The ontology was then used to transform the IIoT data into an EKG. Machine Learning (ML) models were also trained to predict wire bonder faults upon the IIoT data. These were then integrated into a PdM framework that extracts IIoT data from the EKG to forecast possible faults of the wire bonders.

The results obtained are promising and show that the ontology and the EKG were adequate for storing IIoT data, achieving an accuracy of *1.0*. They also show that the PdM framework was able to predict wire bonder faults with an F1-score of *0.75* up to two hours in advance. The EKG served to integrate the disparate data sources that were needed for PdM, and to standardise the vocabulary of such data. This simplified the PdM framework that would have otherwise needed to cater for the different data structures and vocabularies when retrieving the IIoT data. Furthermore, the EKG was able to infer new knowledge through the ontological reasoning, thus completing the knowledge extracted from the wire bonders. These results also demonstrate that the EKG can be leveraged for PdM.

Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface	16
CBM	Condition Based Maintenance	12
CSV	Comma Separated Values	43
DUL	DOLCE+DnS Ultralite	48
EKG	Enterprise Knowledge Graph	1
ETL	Extract Transform and Load	6
ETS	Equipment Tracking System	5
FFT	Fast Fourier Transform	36
GBC	Gradient Boosting Classifier	7
IC	Integrated Circuit	3
IIoT	Industrial Internet of Things	1
JSON	JavaScript Object Notation	43
KG	Knowledge Graph	7
kNN	k-Nearest Neighbours	56
LPG	Labelled Property Graph	19
ML	Machine Learning	6
MTBF	Mean Time Between Failures	12
NBC	Naïve Bayes Classifier	56
NER	Named Entity Recognition	18
NLP	Natural Language Processing	20
OBDA	Ontology Based Data Access	18
OWL	Web Ontology Language	14
PCA	Principal Component Analysis	35

PdM	Predictive Maintenance	2
PM	Preventive Maintenance	12
RF	Random Forest	7
RM	Reactive Maintenance	12
RDF	Resource Description Framework	13
RDFS	Resource Description Framework Schema	14
RMS	Root Mean Square	36
RMSE	Root Mean Square Error	37
RUL	Remaining Useful Life	28
SOSA	Sensor, Observation, Sample, and Actuator	25
SSN	Semantic Sensor Network	24
SVM	Support Vector Machine	35
TSDB	Time Series Database	33
URI	Universal Resource Identifier	13
W3C	World Wide Web Consortium	24
WWW	World Wide Web	13

1 Introduction

Industry 4.0 has been the topic of many studies conducted by research centres and universities (Hermann et al., 2016). It encapsulates a set of concepts that define the fourth industrial revolution, where *Smart Factories* connect people, products, machines and data to innovatively redesign their business processes. It implies that unless manufacturing enterprises exploit knowledge about their internal and external environments, they will lose competitiveness in the market.

1.1 | Motivation

A modern factory generates large volumes of data on a daily basis (Ringsquandl et al., 2017a,b). Such data comes in many forms and from various sources. Of particular interest for this dissertation is the data generated by industrial machines and their embedded sensors. This is referred to as the Industrial Internet of Things (IIoT), where a network of sensors, middleware and software produce data that, through advanced analytics, can provide a deeper insight of the company's operations and assets (Gilchrist, 2016). The diversity of machines and software that produce this data makes their integration quite challenging (Schabus and Scholz, 2017). Such data, however, is paramount for analytics and decision making. The challenge is to collect and store this data in a way that it can be used to improve the decision making function and operations in general.

Several studies look at the Enterprise Knowledge Graph (EKG) as a solution to this problem (Medina-Oliva et al., 2014; Ringsquandl et al., 2017a,b; Schabus and Scholz, 2017; Song et al., 2017). The properties of an EKG make it a flexible data structure that can represent heterogeneous data sources as a network of related entities (Ringsquandl et al., 2017a). Moreover, the semantic enrichment detaches the data from its original structure and vocabulary, which are normally bound to a particular sub-system, and transforms it into a generic reusable form that can be applied to other similar sub-systems (Medina-Oliva et al., 2014). For example, a machine (a system of several parts and components) hosts a number of sensors each observing some property of interest. Such sensors are in turn sub-systems of the machine and produce data that may or may not have a homogeneous form. In the wider scope, when considering the numerous machines contributing within a production line, these varying in function, brand and model, the diversity of data formats increases. The flexibility of the EKG allows for the adaptation of this heterogeneous data into a generic data structure that facilitates data analytics on a broader perspective (Medina-Oliva et al., 2014; Ringsquandl et al., 2017a,b; Schabus and Scholz, 2017; Song et al., 2017).

One way to benefit from data analytics on IIoT data is Predictive Maintenance (PdM) (Ran et al., 2019). Machine maintenance is an important activity within a manufacturing environment, but also a costly one. Apart from the actual cost of maintenance with respect to resource allocation, machine parts and other inventory, companies also consider the reduced capacity utilisation caused by idle machinery, which results in less production. Even costlier are unplanned outages that usually constitute a longer machine downtime due to the need to mobilise resources at short notice. A PdM strategy aims to forecast outages at an early stage to allow for the planning and execution of maintenance activities thus reducing machine downtime, however, the amount of time a fault can be predicted in advance depends on the fault being investigated (Susto et al., 2015). Another advantage of PdM is that machine maintenance would only be applied when required, thus reducing maintenance activities on the machine (Ran et al., 2019).

There are many architectures that an enterprise can consider for its PdM framework. The predictive model can obtain the machine health indicators directly from flat files (Calabrese et al., 2020; Susto et al., 2015) or from a data lake (Spendla et al., 2017). Alternatively, NoSQL databases can be used (Kovalev et al., 2018). Kharlamov et al. (2017), Schmidt et al. (2017), Medina-Oliva et al. (2014) and Voisin et al. (2013) opted for an EKG as the knowledge base that supports their PdM framework. When considering the advantages that the EKG offers on other data models, the number of research projects that use this approach is relatively small. The EKG is regarded as an Enterprise Information System, that integrates the various data sources within an enterprise to facilitate decision making and reporting (Galkin et al., 2017), and this could possibly be one of the reasons for the limited research on the use of the EKG for PdM. This gives us motive to contribute another case study based on a real-life scenario, which demonstrates how the EKG can be leveraged for PdM.

1.2 | Problem Definition

In this dissertation we consider an actual use case from a local semiconductor chip manufacturer. One of the initiatives taken by the company to improve operations is the adoption of PdM in order to reduce unplanned stoppages in production. One of the types of machines used along the production line of semiconductor chips are the wire bonders, that are the scope of this research.

Wire bonders are more prone to failures in comparison with other machines used in semiconductor production, mainly due to the complexity of the function they automate (Klingert et al., 2017). Wire bonders connect the die, a silicon block featuring an Integrated Circuit (IC), to the underlying substrate using fine threads of wire (Tsai et al., 2016). As shown in Figure 1.1, the wire bonds are formed between the pads on the die and the lead frame on the substrate and this requires extreme precision due to the small size of the components.

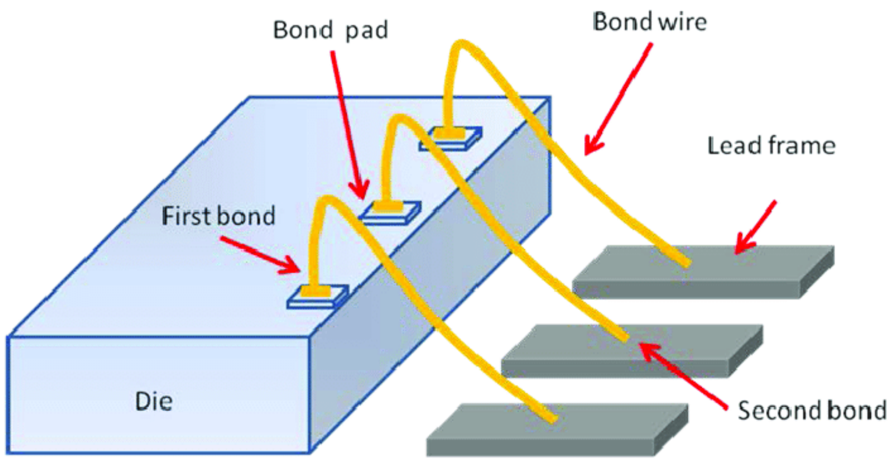


Figure 1.1: A representation of wire bonds connecting the die to the substrate, reproduced from Tsai et al. (2016).

Whilst in operations, wire bonders generate a lot of data in the form of event logs and sensor observations. Such data differs in terms of vocabulary, format and the frequency of the recordings. The harmonisation and integration of the data is essential for the data to be used for PdM, hence the choice of a data integration paradigm and a flexible data structure that can store all this data is imperative. Moreover, we need to identify which features from this data are indicative that the machine is progressing towards a failure and find means how such predictions can be made as early as possible.

1.3 | Research Question

The main research question of this dissertation is the following:

Can an Enterprise Knowledge Graph (EKG) be leveraged to perform Predictive Maintenance (PdM)?

The below, more granular questions will be tackled to help us provide an answer to the main research question.

(RQ1) Can an EKG effectively store IIoT data?
For an EKG to support PdM it needs to store the IIoT data generated by wire bonders upon which faults can be predicted. Therefore, the main research question cannot be answered unless we determine that an EKG can fit IIoT data.

(RQ2) Can wire bonder failures be predicted from IIoT data and machine logs?
We need to determine whether the IIoT data and logs that are generated by the wire bonders can be used to predict their faults. This question investigates whether such data have the prerequisites to perform PdM.

(RQ3) Can an EKG structure support a PdM framework?
This question investigates whether the EKG can supply a PdM framework for wire bonders with the necessary data in a timely manner and whether the EKG provides any advantages over other data structures.

1.4 | Aims and Objectives

The aims of this research are (i) to build an EKG by combining the various data collections within a manufacturing production line, and (ii) to demonstrate that the EKG is a valid data structure to support a PdM framework and offers a number of advantages over other data models.

These aims can be achieved by fulfilling the following objectives:

1. ***Design an ontology to model enterprise-level data consisting of data on wire bonding machines, their sensor network and their measured observations.***

   This objective aims to partly answer RQ1. The challenge here is to design a structure that is flexible enough to fit the heterogeneous data sources, yet sufficiently descriptive to preserve as much as possible of the original data.

2. ***Transform the data generated by the wire bonders into an EKG in real-time using automated or semi-automated approaches.***

   The volumes of data generated by the wire bonders do not permit the updating of the EKG to be done manually, therefore, we must find a way for this to be automated. Moreover, the IIoT data needs to be transferred into the EKG in real-time so that machine faults are predicted as early as possible. This objective also aims to answer RQ1.

3. ***Train ML models to predict faults of wire bonders with enough lead time to allow for corrective actions.***

   This objective aims to answer RQ2. It includes finding which ML models are suitable to predict such faults, which features from the data are useful for the predictions, which data preprocessing methods facilitate the models' performance, and how much in advance can the faults be predicted.

4. ***Build a PdM framework that consumes the EKG for sensor observations and takes advantage of its characteristics.***

   This objective aims to demonstrate that the EKG can supply the PdM framework with the required sensor observations to make the predictions. Moreover, to predict wire bonder faults as early as possible, the time lag between the sensor observations and the data reaching the PdM framework should be kept to a minimum. This objective aims to answer RQ3.

1.5 | Contributions

In this research we demonstrate the effectiveness of the EKG in storing the observations produced by the various sensors deployed on industrial machines, and in supporting a PdM framework. The set up is based on that of Schmidt et al. (2017) and we use data produced by wire bonding machines from an actual production line of a local semiconductor manufacturing firm. We use three datasets - the Equipment Tracking System (ETS) event logs, the machine logs and the IIoT data. The ETS logs consist of a series of messages and events generated by the machines throughout their operations while the machine logs consist of a series of sensor observations together with the records of the settings of the machine. The IIoT dataset, on the other hand, consist of a series of sensor observations. Whilst the ETS logs are generated whenever an event occurs, the machine logs and IIoT records are generated every five minutes and three minutes respectively. The structure, format and vocabulary of all three datasets is different. The datasets cover a stretch of 23 days of operations for four wire bonders. More details about the datasets are provided in Appendix A.

Using the method outlined by Noy et al. (2001), we design and build an ontology that models the machines, their states, their sensors and sensor observations, then we build and validate an EKG based upon this schema. We also follow the approach of Ringsquandl et al. (2017a) and use Extract Transform and Load (ETL) scripts that extract data from the machine datasets, transform it into the required structure and load the data into the EKG, thus automating the ingestion of data into the knowledge store.

We use the datasets generated by the wire bonders to train different Machine Learning (ML) models to predict faults. Based on the approaches of Soares (2015), Calabrese et al. (2020) and Gandhi et al. (2018), we experiment with different preprocessing methods in order to find the optimal set-up for the predictive models. The best performers are then integrated into the PdM framework, which is shown in Figure 1.2, and is built using a producer/consumer design pattern. The producer consists of the wire bonders that generate the data and the ETL scripts that extract this data and load it into the EKG. On the other hand, the consumer retrieves real-time sensor observations that are used by the predictive models to forecast possible faults. The consumer also features a dashboard that serves to project the necessary information to the plant personnel.

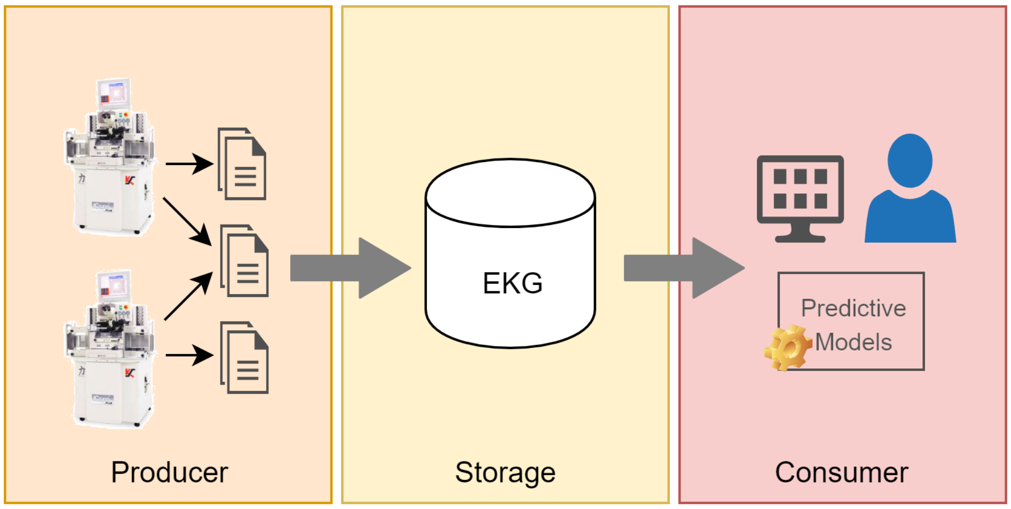


Figure 1.2: A high-level representation of the PdM framework produced.

The results are encouraging and demonstrate that the EKG was well suited to store IIoT data. A retrospective evaluation (Paulheim, 2017) of the EKG results in an accuracy score of *1.0*. Moreover, on experimenting with the different models and preprocessing set-ups, we achieve a fault prediction with an accuracy of *0.664* and an F1-score of *0.75*, 120 minutes ahead of the actual fault. These results are achieved by training a Random Forest (RF) model on a dataset that is scaled using L2 normalisation and smoothened using a simple moving average with a 30-minute rolling time window. The dataset is reduced using a filter feature selection method and class balanced using SMOTE. Similarly, we obtain a fault prediction with an accuracy of *0.672* and an F1-score of *0.707*, 90 minutes ahead of the fault occurrence, by training a Gradient Boosting Classifier (GBC) on a similar dataset.

We finally integrate the EKG and predictive models into the system featured in Figure 1.2. The end result is a PdM framework operating on top of an EKG and forecasting wire bonder faults 120 minutes and 90 minutes in advance.

1.6 | Document Structure

This chapter provided an introduction to the domain of this dissertation. It also explained our motivation and the research questions that this dissertation was aimed to answer. An overview of the main contributions of this research project was also provided. The remaining part of the document is organised as follows.

More context to this dissertation is given in Chapter 2 by explaining the semiconductor manufacturing process, with particular emphasis on the wire bonding procedure and the various maintenance strategies that a manufacturing firm can adopt. It also introduces the Knowledge Graph (KG) and its specialised version, the EKG. Consequently, the chapter discusses PdM as well as its advantages and disadvantages. It also explains how Machine Learning is used for Predictive Maintenance.

Chapter 3 presents the methodology we adopted throughout this research. First, the datasets used in this dissertation are explained. Then the approach used to design, create and populate the EKG is described. The chapter proceeds to explain how a number of ML models were trained and validated and how the optimal models were chosen via experimentation. Finally, the resulting PdM framework is presented as the integration of the various components created throughout this research project.

Chapter 4 features an evaluation of the deliverables produced in this dissertation. The results of the evaluation are presented and discussed. The chapter closes with an interpretation of the results obtained.

Finally, Chapter 5 concludes this dissertation by discussing the achievements of this research against its objectives and how the research questions were answered. It also proposes further work that can be explored in the future to extend this research.

2 Background & Literature Overview

In this chapter we provide a backdrop to this dissertation by describing the semiconductor manufacturing process, with particular emphasis on wire bonding and machine maintenance. We proceed by reviewing the state of the art of Knowledge Graphs, focusing on Enterprise Knowledge Graphs. We also describe how these are used to model sensor networks and the various ontologies that are available in this regard. Consequently, we delve into Predictive Maintenance and explain the role that machine health indicators play in the forecasting of machine failures. Finally, we go through the steps involved in producing a Machine Learning model for Predictive Maintenance.

2.1 | The Semiconductor Manufacturing Process

The production floor of a semiconductor manufacturing plant is divided into two parts: the front-end and the back-end (Ovacik and Uzsoy, 2012). The front-end is where the semiconductor components are worked upon whilst in their raw form, while in the back-end the components are sealed into a covering shell and therefore less sensitive to contamination. The front-end requires a cleanroom[1] controlled environment to reduce the risk of contamination.

2.1.1 | Die preparation

The manufacturing process starts with raw silicon wafers upon which ICs are fabricated (Ovacik and Uzsoy, 2012). Each wafer can contain thousands of such circuits. A photoresistant substance is applied to the wafer, which is then exposed to ultra violet light passing through a mask with the pattern of the circuit. The wafer is then baked and

[1]ISO 14644-1:2015 Cleanrooms and associated controlled environments

cleaned from impurities. Each circuit on the wafer is then electrically tested by means of thin probes. Defective circuits are marked to be later discarded.

The wafers are mounted onto a machine that cuts the circuits loose thus forming dies, small semiconductor blocks each featuring an IC (Ovacik and Uzsoy, 2012). The dies are then attached to strips of substrate components, a process called *die attach*.

2.1.2 | Wire bonding

The wire bonding process is a delicate step in the semiconductor production process. It physically wires the die/s to the underlying substrate (Schuettler and Stieglitz, 2013). Extreme precision is required due to the small size of the components. The wire used consists of a thin thread of bare metal, usually copper, silver or gold, depending on the needs of the component being produced. The wire diameter normally ranges between 17.8 µm and 25.4 µm.

The wire bonding method used is known as the *ball-wedge* method. The steps taken by the machine to produce a wire bond are shown in Figure 2.1, and are described by Schuettler and Stieglitz (2013) as follows:

1. The machine positions the capillary with the wire thread passing through it over the pad and applies a high voltage spark to the tip of the wire.
2. The wire melts forming a *ball* at the tip.
3. The capillary presses the *ball* against the pad whilst applying heat or ultrasound energy to produce the first bond.
4. The capillary lifts off.
5. The capillary positions itself above the lead frame.
6. Once again the capillary presses against the lead frame whilst applying heat or ultrasound energy to produce the second bond.
7. A clamp within the capillary closes to trap the wire so that when the capillary lifts off the wire breaks itself from the wire bond.

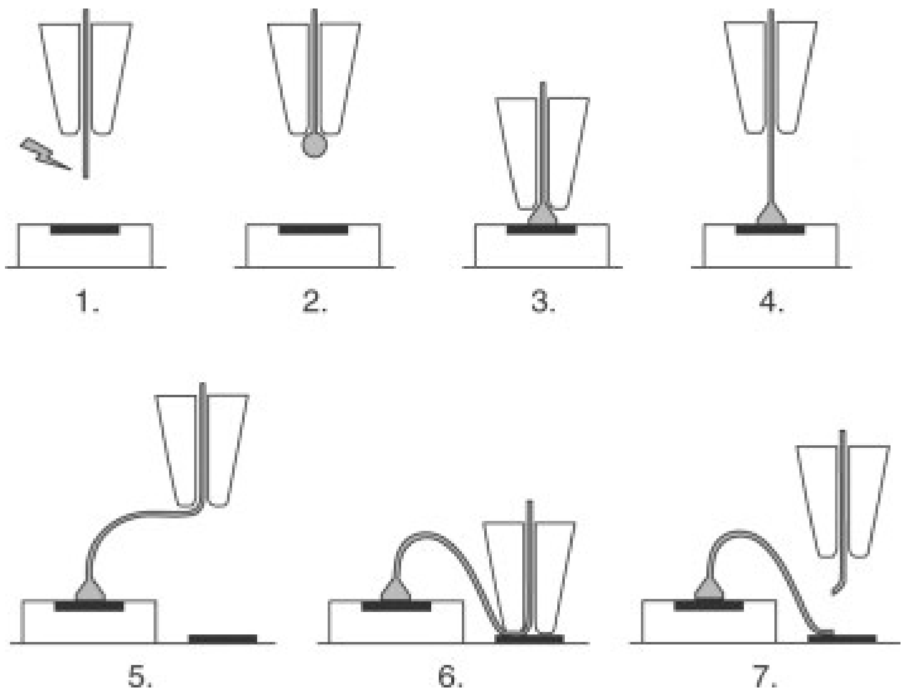


Figure 2.1: The wire bonding process, reproduced from Schuettler and Stieglitz (2013).

2.1.3 | Microchip Closure and Testing

The bonded components are then transferred to the back-end of the production floor where the devices are sealed into a resin casing (Ovacik and Uzsoy, 2012). The microchips are then marked with the name of the manufacturer and a serial number.

The devices are then put to a series of tests to ensure an adequate quality of the product (Ovacik and Uzsoy, 2012). Such tests are carried out under different environmental conditions such as varying temperature and humidity. Moreover, the microchips are subjected to thermal stress tests by being subjected to high currents at a temperature of approximately 125 °C for not less than 24 hours. The microchips are also inspected visually for defect such as bent leads or chipped casings.

2.1.4 | Maintenance Strategies

Machine maintenance is an essential activity within manufacturing companies. Mobley (2002) states that maintenance activities account for between 15% and 60% of the total cost of goods produced. Therefore the maintenance strategy adopted by an enterprise can impact its competitiveness with regards to prices, quality and performance (Ran et al., 2019). On one hand, frequent maintenance is costly in terms of parts and inventory, but also due to the slow-down in productivity as a result of idle machinery. On the other hand, scarce maintenance leads to unexpected machine outages that result in longer equipment downtime and even costlier maintenance (Mobley, 2002). So an effective maintenance strategy should strike a balance between machine health and maintenance costs.

The simplest strategy an enterprise can adopt is Reactive Maintenance (RM), where maintenance is only applied when a machine breaks down (Mobley, 2002). This run-to-failure approach reduces the burden of maintenance planning and scheduling, however, Mobley (2002) found that companies using this method on average spend three times as much on maintenance activities and need to have extensive inventories when compared with companies employing other strategies. Mobley (2002) continues that although RM was a popular method in the past it is rarely used nowadays.

In Preventive Maintenance (PM), it is assumed that a machine degrades in a similar way to its counterparts (Mobley, 2002). Maintenance is applied on a timely basis so as to prevent potential breakdowns. The frequency of maintenance depends on the Mean Time Between Failures (MTBF) of the type of machine. A typical degrading pattern can be seen in Figure 2.2, which shows that the probability of a machine failure is higher in the beginning due to installation problems and after exhausting its expected lifetime. The problem with PM is that parts are replaced based on the MTBF statistic and not their condition, which results in parts being replaced before their end-of-life.

Preventive Maintenance (PM) remains the most common maintenance strategy but industry leaders are moving towards Predictive Maintenance (PdM) (Shin and Jun, 2015). PdM, which is also known as Condition Based Maintenance (CBM), involves the monitoring of some machine health or performance indicator/s, based on which machine failures can be forecasted (Mobley, 2002). The goal is to maximise the interval between maintenance activities whilst minimising unscheduled outages. An advantage of PdM over PM is that in the latter, parts are replaced on a schedule basis, irrespective of their wear or condition (Ran et al., 2019). This results in parts being discarded before the end of their useful life. In PdM, it is the health condition of the machine that calls for maintenance and therefore, parts are used until the end of their useful life. So by avoiding running to failure and experiencing unforeseen stoppages, and by postponing maintenance activities until really needed and maximising the utilisation of machine parts, PdM is more cost effective than the other maintenance strategies (Ran et al., 2019).

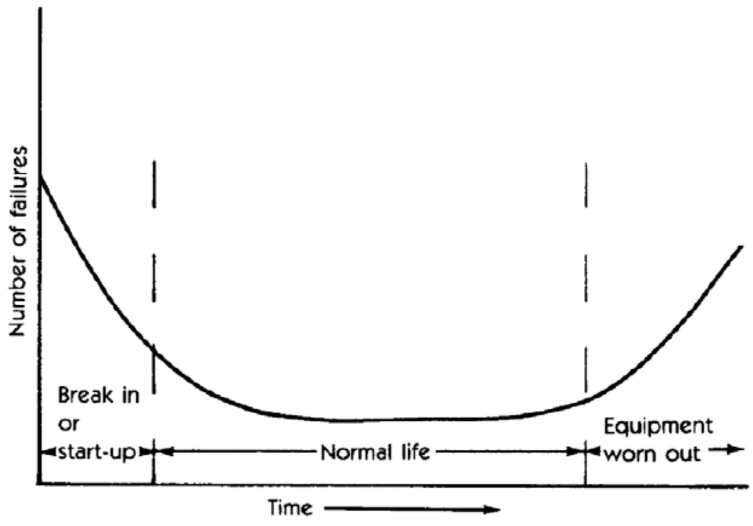


Figure 2.2: A plot showing the number of failures throughout a typical lifetime of a machine, reproduced from Mobley (2002).

2.2 | Knowledge Graphs

Knowledge Graphs (KGs) are founded on the principles of the *Semantic Web*. In Berners-Lee et al. (2001), the authors describe a World Wide Web (WWW) where agents, pieces of software that can understand the meaning of data through the use of semantics, can communicate and exchange knowledge without the need for human interaction. Such agents and data are connected yet decentralised. The authors also stress the importance of the Resource Description Framework (RDF) in fulfilling the idea of a Semantic Web. RDF is based on the Extensible Markup Language (XML) and allows data to be structured as *triples*, each made up of a *subject*, a *predicate* and an *object*. Each triple represents a fact and its structure is very similar to the way humans express knowledge (Berners-Lee et al., 2001). For example, the fact "Moby-Dick was written by Herman Melville" would be loosely expressed as `<ex:Moby-Dick> <ex:hasAuthor> <ex:HermanMelville>`, where `ex` represents an example *namespace*[2]. The full RDF syntax to represent this statement is presented in Listing 2.1. The subject and predicate consist of a Universal Resource Identifier (URI). The object can either be a URI or a value. The URI is a global identifier that distinguishes the resource from other such resources on the WWW.

[2]The namespace is a prefix that together with the suffix form the URI. For example <ex:Moby-Dick> represents the URI http://www.example.com/Moby-Dick.

```
@prefix ex: <http://www.example.com/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://www.example.com/Moby-Dick> a ex:Book ;
                                ex:hasAuthor ex:HermanMelville .
<http://www.example.com/HermanMelville> a ex:Person .
```

Listing 2.1: An example RDF syntax.

Knowledge Graphs (KGs) grew in popularity in 2012, when Google announced their KG as the backbone of the search engine, allowing users to search for *things* rather than *strings* (Ehrlinger and Wöß, 2016; Gomez-Perez et al., 2017). The success story of the Google KG also inspired others to follow suit. In Paulheim (2017), the author defines the basic characteristics of a KG. He states that a KG should be an entity-centric graph describing real-world entities and their interrelations. It should also have a formally defined schema, also referred to as the *Tbox* (terminology box) or the ontology. Whereas the actual entity instances form the assertion part or the *Abox*. A KG also permits having interrelations between arbitrary entities. Moreover, a KG should cover various topical domains.

2.2.1 | Ontologies

Ontologies can be seen as the conceptualisation of a real-world domain (Uschold et al., 1996). They express some shared understanding of the domain by defining the concepts of the domain and their inter-relationships. They are an integral part of a KG as they define its formal specification (Schmidt et al., 2017). Not only do they stipulate the vocabulary that is used within the KG, but they also specify the relationships that can exist between entities and formulate a set of rules that model the real-life domain. They also formalise any assumptions about the domain (Noy et al., 2001). Ontologies are meant to be reused or extended so that the domain specifications are applied to other KGs modelling the same domain, thus minimising the implementation effort.

In the Semantic Web sphere, ontologies are specified using the Resource Description Framework Schema (RDFS) or Web Ontology Language (OWL) languages (Powers, 2003). OWL is more expressive and allows for explicit constraints on the statements within the KG. In RDFS constraints are loosely imposed.

2.2.1.1 | Ontology Construction

Most commonly, ontologies are created manually (Pan et al., 2017). This is not a straightforward task and requires an in-depth knowledge of the domain being modelled. Moreover, the authors of the ontology must be aware of the planned use of the KG.

Uschold et al. (1996) presents a procedural approach to build an ontology. The process indicated by the authors starts by determining the ultimate use of the ontology and establishing its scope. The ontology is then built by producing a list of the domain concepts lying within the established scope and their inter-relationships. The concepts and relationships should be defined using an unambiguous vocabulary. One should also consider other existing ontologies covering the same domain to potentially integrate them into the ontology being built either wholly or partially. The ontology is then coded using a formal representation language. Uschold et al. (1996) proceeds by referring to the evaluation methods suggested by Gómez-Pérez et al. (1995), that says the ontology should be evaluated against the initial requirements, using competency questions or tested in the real-world. Lastly the ontology should be documented.

The method to build an ontology described in Noy et al. (2001) is slightly different than that in Uschold et al. (1996). Once again the process starts by establishing the domain and scope of the ontology. The authors suggest that a number of competency questions are set at the beginning. These are questions that the final ontology should be able to answer and help to establish the scope. Contrary to Uschold et al. (1996), Noy et al. (2001) suggests that existing ontologies for the domain are considered before any attempt to build a new ontology. Ontologies can be reused wholly or partially, but the reuse can significantly reduce the effort involved in building an ontology. The actual building of the ontology starts by drawing up a list of concepts that acts as a sketch of the ontology. Consequently, the actual classes are defined either following a top-down approach, that starts with the general concepts to then specialise into more specific concepts, or by using a bottom-up approach. The authors indicate that a mixture of both approaches can also be used. For each class that is identified, its properties (also referred to as *slots*) are specified followed by the inter-relationships between classes. The rules and constraints (also called facets) for the properties and relationships are also specified. Finally, the actual instances of the classes are created.

2.2.2 | Enterprise Knowledge Graphs

Enterprise Knowledge Graphs (EKGs) are specialised KGs. It is the enterprise characteristics of the EKG that distinguishes it from a KG (Galkin et al., 2017). Unlike the characteristics defined by Paulheim (2017) requiring a KG to incorporate various domains, Galkin et al. (2017) and Jetschni and Meister (2017) state that the EKG should consist of a collection of knowledge within the organisational domain. In Galkin et al. (2017), the authors compare the EKG to other data integration paradigms, such as data warehouses, data lakes and enterprise search, to determine the best fit for an Enterprise Information System. The qualities considered in this comparison are whether they integrate conceptual or operational data, their ability to integrate heterogeneous data, their ability to integrate internal (i.e. from within the enterprise) and external data, the number of data sources the paradigm can integrate, whether the integration is physical or virtual, the domain coverage, and their ability to incorporate semantics for machine data consumption. The findings of Galkin et al. (2017) show that the EKG is the only paradigm that incorporates both conceptual and operational data, through the Tbox and Abox parts of the KG respectively. EKGs are also suitable to integrate diverse data models or structures, and can also integrate both internal and external data thus offering high domain coverage. Thanks to the use of URIs, EKGs also score the highest in semantic integration. Galkin et al. (2017) concludes that the EKG is a very strong candidate among the data integration paradigms an enterprise may consider when choosing its Enterprise Information System.

Galkin et al. (2017) also positions the EKG within the ecosystem of enterprise applications. It consumes the various heterogeneous data sources as shown in Figure 2.3. The Ontological Coherence Layer consists of the various ontologies. Such ontologies can either be domain specific, and therefore define the various content of the EKG together with their interrelations and rules, or they can define higher level concepts, such as for instance a security ontology that restricts access to the data according to specific business rules. Both the EKG and the ontologies are not static, but rather follow the knowledge evolution process within the organisation (Jetschni and Meister, 2017). An Application Programming Interface (API) layer is also featured in Figure 2.3. This API layer exposes the EKG for the consumption of the various enterprise applications.

2.2.2.1 | EKG Implementation

In an abstract form, an EKG can be seen as having three main components: the *Knowledge Acquisition and Integration* layer, the *Knowledge Storage* layer and the *Knowledge Consumption* layer (Pan et al., 2017). The subcomponents of each layer are shown in Figure 2.4, where the Knowledge Acquisition and Integration consists of a cycle of knowledge management stages that ingest knowledge into the Knowledge Storage and the

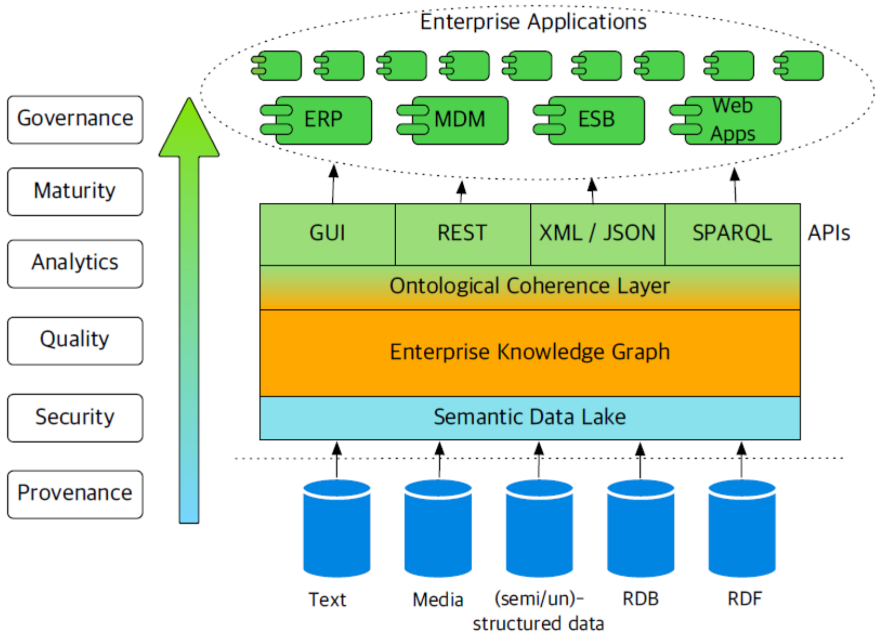


Figure 2.3: The position of an EKG within the ecosystem of enterprise applications, reproduced from Galkin et al. (2017).

Knowledge Consumption incorporates a number of services facilitating the use of the acquired knowledge.

a) Knowledge Acquisition and Integration: The first cycle within the knowledge life-cycle is the construction of the EKG (Pan et al., 2017). The EKG can be built in a bottom-up manner, that is by examining the data and identifying entities and relations, or in a top-down manner by defining a number of use cases for the EKG to then determine which data is needed to satisfy the use cases. Ensuing cycles of the knowledge lifecycle lead to the ingestion of new knowledge into the EKG.

The first step in the Knowledge Acquisition and Integration cycle involves the development of the ontology, which we discussed in Section 2.2.1.1. In the *Data Lifting* and *Data Annotation* steps, the data that is earmarked to be transferred into the EKG is mapped onto the developed ontology and transformed into the required structure (Pan et al., 2017). There are different ways how to go about this. The mappings between existing data and the ontology are at times created manually (Petersen et al., 2017; Schmidt et al., 2017). The data is then transformed and migrated into the EKG using Extract Transform and Load (ETL) scripts. Sometimes the large amounts of mappings required make it

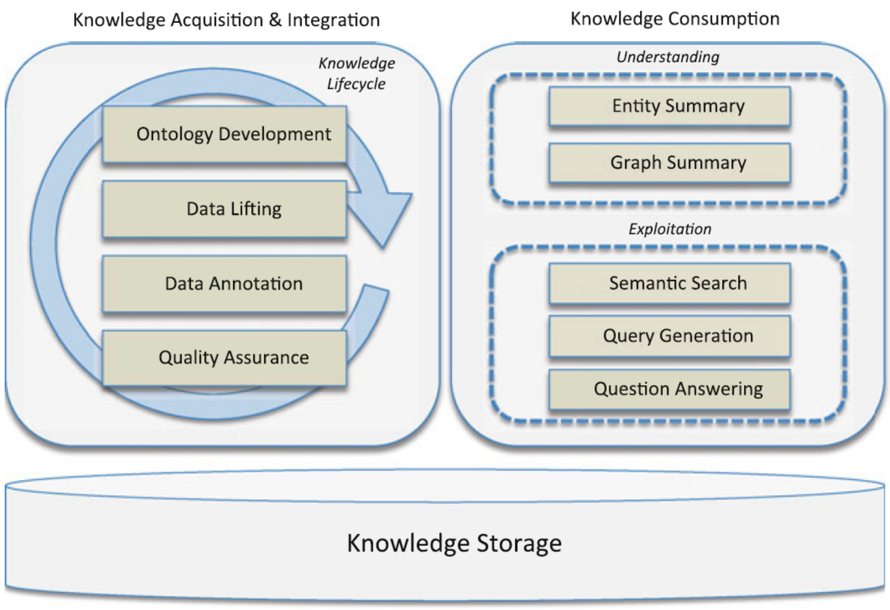


Figure 2.4: A high-level view of an EKG, reproduced from Pan et al. (2017).

very time consuming for these to be done manually, so a semi-automatic approach is taken (Song et al., 2017). One such approach is Named Entity Recognition (NER) that can identify entities and their classes from pieces of text. NER methods can be either rule-based or use a supervised learning method. Finally, the *Quality Assurance* step ensures the correctness of the resultant EKG.

b) Knowledge Storage: This layer is the actual container of the data. To materialise an EKG an enterprise can resort to an RDF store (Pan et al., 2017). Popular RDF stores include Apache Jena[3], GraphDB[4], Virtuoso[5], Stardog[6] and AllegroGraph[7]. Medina-Oliva et al. (2014) and Song et al. (2017) use RDF stores for implementing their EKGs.

Since most data in enterprises is already stored in structured repositories, such as relational databases, one can opt for an Ontology Based Data Access (OBDA) implementation (Pan et al., 2017). Using this paradigm an ontological layer (Tbox) is created on top of the existing data sources, together with mappings between the ontology and the data sources. Such an architecture is depicted in Figure 2.5, where the EKG consumers can

[3]`https://jena.apache.org/` (last accessed 21/09/2020)
[4]`http://graphdb.ontotext.com/` (last accessed 10/10/2020)
[5]`https://virtuoso.openlinksw.com/` (last accessed 21/09/2020)
[6]`https://www.stardog.com/categories/rdf/` (last accessed 21/09/2020)
[7]`https://allegrograph.com/` (last accessed 21/09/2020)

extract knowledge found in the original data sources by querying the ontological layer. The Abox in this case can either be virtual, that is the data is kept solely in their original source and retrieved whenever needed, or materialised, that is the knowledge is extracted into the EKG via the mappings and refreshed periodically. Petersen et al. (2017) and Schmidt et al. (2017) use this paradigm for their EKG implementation by using the R2RML[8] mapping language. Kharlamov et al. (2017) use a tailor-made OBDA-based method to map onto streaming data.

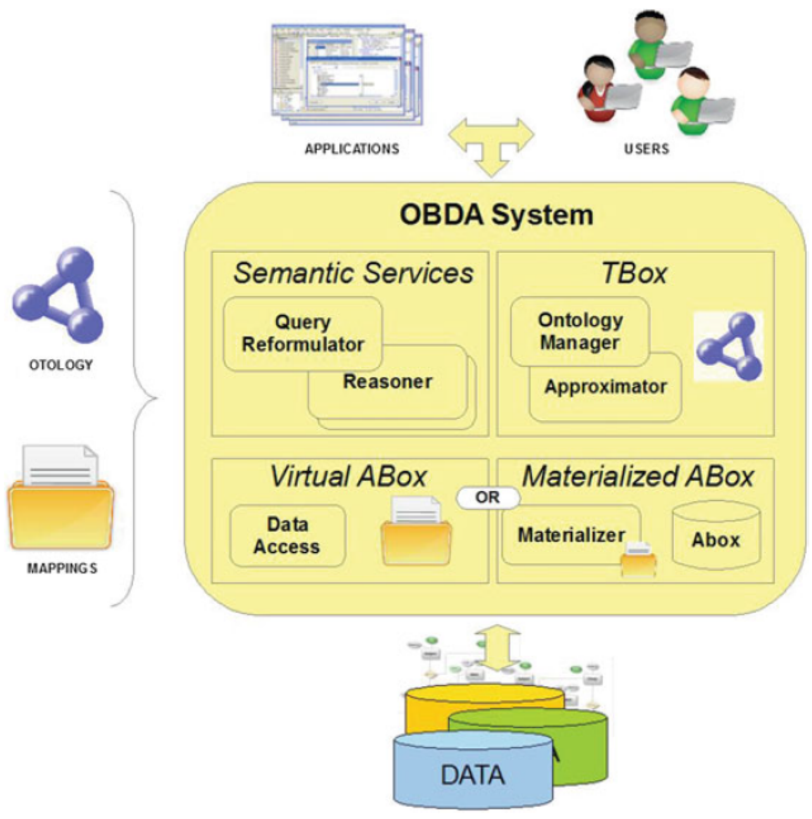


Figure 2.5: An EKG architecture using OBDA, reproduced from Pan et al. (2017).

Another alternative is the use of Labelled Property Graphs (LPGs) (Pan et al., 2017). Although KGs are normally attributed with RDF, they can also be implemented using LPGs. LPGs consist of graph structures having entities as nodes and their interrelationships as edges. In addition nodes and edges may have additional properties in the form of key-value pairs. Schabus and Scholz (2017) implement their EKG using

[8]`https://www.w3.org/TR/r2rml/` (last accessed 21/09/2020)

Neo4j[9], one of the leading LPG platforms.

c) Knowledge Consumption: This layer acts as an interface to the EKG for consumption by its users, whether human or other software agents (Pan et al., 2017). It comprises a number of services that allow the users to query the EKG and get a result in return. Queries over the EKG are normally expressed using SPARQL[10], however, some KG implementations incorporate Natural Language Processing (NLP) techniques that allow its human users to ask questions using natural language.

An example of the Knowledge Consumption layer can be seen in Figure 2.6, and shows the EKG that was implementated in Petersen et al. (2017). The software applications, being either machine agents or end-user programs, interact with the EKG via SPARQL endpoint APIs. In this case the EKG follows the OBDA paradigm. A SPARQL endpoint is also used for information retrieval in Schmidt et al. (2017), while in Schabus and Scholz (2017), the authors use the Cypher[11] API of Neo4j.

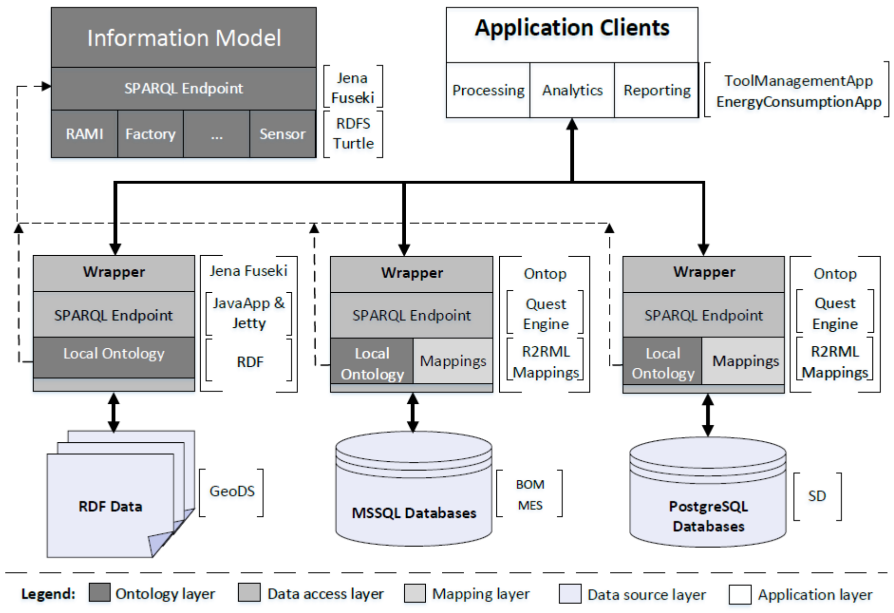


Figure 2.6: A blueprint of the EKG implementation used in Petersen et al. (2017).

In Song et al. (2017), the authors explain how in their implementation of the EKG they integrated an NLP tool that translates questions posed using natural language into

[9]`https://neo4j.com/` (last accessed 10/10/2020)
[10]`https://www.w3.org/TR/rdf-sparql-query/` (last accessed 22/09/2020)
[11]`https://neo4j.com/docs/http-api/current/introduction/` (last accessed 10/10/2020)

SPARQL queries. In this way, non-technical users of the EKG can retrieve knowledge using plain English.

2.2.2.2 | Applications of the EKG

Enterprises resort to EKGs for different reasons but some common themes have been observed in literature. These are discussed hereunder while Table 2.1 (Pg 23) provides a summary of our findings by listing the various literature and indicating the reason why the authors opt for a KG structure.

a) As a tool to aid decision making: Decision makers need information at their fingertips and the data integration capabilities of the EKG makes it a strong tool to assist decision making (Galkin et al., 2017). In Schabus and Scholz (2017), the authors describe how an EKG facilitated decision making within a semiconductor manufacturing firm by integrating data from several data repositories using ETL tools. By querying the EKG, decision makers were able to identify and address bottlenecks in the production process. The authors also describe a scenario where a contamination in the production floor occurred and the floor managers could trace potentially contaminated products through the EKG. These could then be examined to determine their validity for consumption.

Petersen et al. (2016) and Petersen et al. (2017) tackle the lack of a single data container and of a single interface to all enterprise data of a multinational manufacturing company by implementing an EKG. The authors describe how, before the EKG implementation, the information model within the company was complex and scattered, and this was a barrier for the management to compile the information they required to base their decisions upon. The company kept track of its assets through an application that sits on top of the EKG and when inventory was needed at a plant the management could locate the nearest facility that had the inventory in store by querying the EKG. The EKG was also used to forecast expenses based on past operations data.

In Golebiowska et al. (2001), the authors explain how an EKG was developed for an automobile manufacturer to preserve knowledge acquired in past projects. This knowledge was then exploited when taking decisions in new projects. Moreover, Kharlamov et al. (2017), Schmidt et al. (2017), Medina-Oliva et al. (2014), Voisin et al. (2013) and Efthymiou et al. (2012) use an EKG to support PdM. These are discussed in more detail in Section 2.3.3.

b) To merge heterogeneous data: The diversity of data structures hinders interoperability and data integration, but EKGs can facilitate this (Galkin et al., 2017). In Schabus and Scholz (2017), the EKG incorporates data about production assets, equipment, spatial information of the production floor, historical production data and planned production operations. Such data comes from a variety of proprietary IT systems, each having their own format. The authors consider the use of an EKG as a solution to this problem. Petersen et al. (2017) describes a similar situation where the authors implement an EKG to integrate large volumes of data in different formats and from various sources to establish a single container for all enterprise data.

Medina-Oliva et al. (2014), Schmidt et al. (2017) and Kharlamov et al. (2017) integrate sensor readings data that originates from various machines. This data is heterogeneous in terms of structure and format due to the diversity of machines being used within the respective scenario. Once again the authors considered the EKG a suitable data integration paradigm to integrate not only the heterogeneous IIoT data, but also the data about the machines, parts and operations that give context to these data.

Another use of an EKG for integrating heterogeneous data was found in Golebiowska et al. (2001) whereby an RDF database that was developed as part of this research study merges structured data coming from different databases to unstructured data emerging from project documents. Once again the EKG structure proved flexible enough to incorporate such data.

c) To standardise the vocabulary: Data coming from different data sources, whether heterogeneous or not, can make use of different semantics (Medina-Oliva et al., 2014). A semantic model at a higher level is necessary to find the commonalities between the data so that these can be integrated. The ontology layer in EKGs provides this semantic integration. In doing so, it also provides a common vocabulary and therefore a common understanding of the concepts involved. Such an approach is also seen in Schmidt et al. (2017), Petersen et al. (2016) and Petersen et al. (2017) where the schemas of the low-level data is mapped to the ontologies so that the software applications that sit on top of the EKG can use a common vocabulary that is defined in the ontologies.

d) For context awareness: Another reason to use EKGs is to achieve context-aware software and/or machines. Laroche et al. (2016) discusses the factories of the future where workers have software assistants that understand the current situation and provide the help and knowledge needed by the person at the right place and time. The authors state that although factories have not reached such a stage, the technologies are already available and these are EKGs. The capturing and storing of the knowledge is crucial, as is the ontology that formalises the knowledge. The authors also present a prototype of this software assistant, that incorporates a web interface where the human worker can ask questions and the assistant, by implicitly understanding the situation provides the right information. Moreover, through Augmented Reality, the assistant can determine the state and place of the product being manufactured and advise the worker without being prompted. Garofalo et al. (2018) also speaks about context-aware robots that interact with EKGs. When handling materials, such robots can determine the positions of other robots and stations and determine the shortest travel path to save time and energy.

Petersen et al. (2016) also tackles context awareness by introducing an EKG to act as a semantic layer on top of the data sources of the factory. Consequently, the applications consuming the data can make use of their semantic properties to provide a better output to its users. For example, when the user is looking for inventory, the application can determine the user's location and locate the nearest store having the required inventory.

Reference	**A**	**B**	**C**	**D**
Efthymiou et al. (2012)	✓	✓		
Garofalo et al. (2018)		✓		✓
Golebiowska et al. (2001)	✓	✓		
Kharlamov et al. (2017)	✓	✓		
Laroche et al. (2016)				✓
Medina-Oliva et al. (2014)	✓	✓	✓	
Petersen et al. (2016)	✓	✓	✓	✓
Petersen et al. (2017)	✓	✓	✓	
Schabus and Scholz (2017)	✓	✓		
Schmidt et al. (2017)	✓	✓	✓	
Voisin et al. (2013)	✓	✓		

Table 2.1: The reasons why a KG paradigm was chosen (A - Decision making; B - Heterogeneous data; C - Standard vocabulary; D - Context awareness).

2.2.3 | Use of Knowledge Graphs for Sensor Networks

Knowledge Graphs (KGs) are also used to model sensor networks and their readings. In the manufacturing domain, Petersen et al. (2016), Schmidt et al. (2017) and Kharlamov et al. (2017) model sensor networks in a KG, which also incorporates sensor observations. These observations are then used for decision making (Petersen et al., 2016) and for machine fault prediction and fault diagnosis (Kharlamov et al., 2017; Schmidt et al., 2017). Medina-Oliva et al. (2014) also capture sensor observations in a KG that is then used for PdM of ship engines. In all four research projects, it was primarily the KG's ability to integrate heterogeneous data that drove the authors towards using this paradigm.

Umiliacchi et al. (2011) also implements a KG that incorporates the readings of the various sensors deployed on trains. The sensors transmit their observations to a central KG and are then used for PdM. The authors explain how the ontology provides a generalised view of the subsystems of a train, for example a propulsion engine, whether it is diesel or electric powered, is identified by the software as an engine without the need to be specifically coded to cater for the differences between the two variants. Hence the software is able to monitor the condition of both types of propulsion engines as if they were the same. In Klotz et al. (2018), a KG integrates the readings of sensors deployed on cars. The authors suggest that a KG structure would be suitable to integrate sensor data irrespective of the properties (brand, model, etc.) of the source sensor. It also helps to achieve a common vocabulary for all sensor readings. The authors indicate that such a KG can be exploited for fleet monitoring and trajectory mining.

Sensor networks are also widely used to measure meteorological conditions. In Catherine et al. (2019), the authors capture sensor readings in a KG that can then be used to analyse weather conditions. They argue that the flexibility of RDF and the fact that it is an open standard make the KG usable to a number of stakeholders that are interested in monitoring weather conditions. Gray et al. (2011) presents another use case where a KG is implemented as the backbone of a flood emergency response system. The authors exploit the KG's properties to integrate sensor readings with information originating from relational databases and maps.

2.2.3.1 | Ontologies Modelling Sensor Networks

In Compton et al. (2012), the World Wide Web Consortium (W3C) introduced the Semantic Sensor Network (SSN) ontology[12]. Although similar ontologies were available before the SSN ontology, most were problem specific and had a narrow scope of application (Wang et al., 2015). Others, like CSIRO (Compton et al., 2009) and OntoSensor (Russomanno and Goodwin, 2008), lacked expressiveness. The SSN ontology provides a generic, field-independent model with full expressive power (Wang et al., 2015).

[12]`https://www.w3.org/2005/Incubator/ssn/ssnx/ssn` (last accessed 10/11/2020)

Successors of the SSN build upon it to address its limitations, either to focus on a particular application or to extend the ontology by filling particular gaps (Wang et al., 2015). WSSN (Bendadouche et al., 2012) and SCO (Müller et al., 2013) extend the SSN by adapting the ontology to be used for wireless sensors and cloud connected sensors respectively. Schlenoff et al. (2013) propose a more specialised ontology for the manufacturing industry by adding concepts such as *action* (of a robot), *state* and *physical location*. Moreover, the W3C produced the Sensor, Observation, Sample, and Actuator (SOSA) ontology[13] as an extension of the SSN to incorporate lessons learned since its release (Janowicz et al., 2019). Part of the SOSA ontology that can be used to model machines (platforms in SOSA vocabulary), their sensors and sensor readings (observations) can be seen in Figure 2.7. A summary of the contributions brought by the various sensor ontologies reviewed in this research project is provided in Table 2.2.

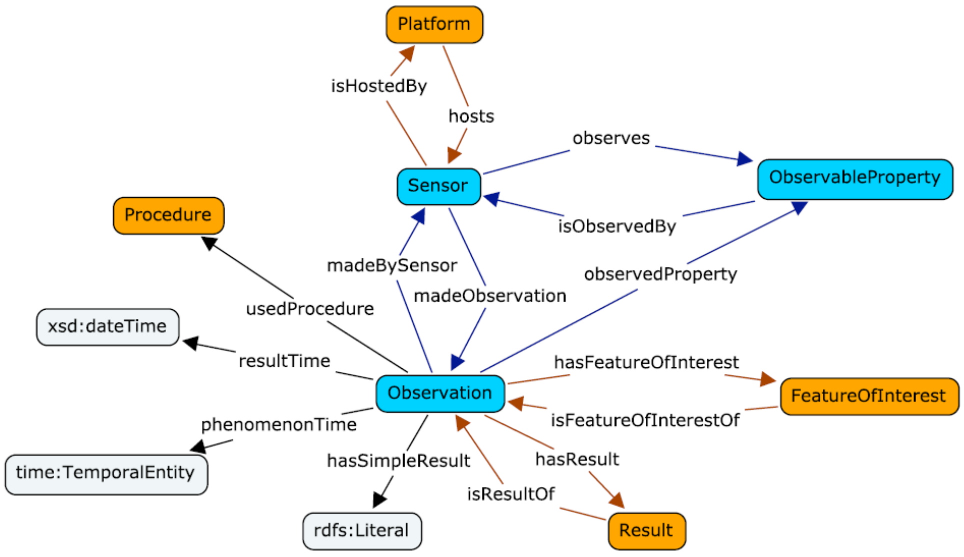


Figure 2.7: A partial representation of the SOSA ontology, reproduced from Janowicz et al. (2019).

Notwithstanding the advantages of reusing existing ontologies and the availability of ontologies for modelling sensor networks and their observations, some implementations we found in literature opt to create their own ontologies (Kharlamov et al., 2017; Medina-Oliva et al., 2014; Schmidt et al., 2017; Umiliacchi et al., 2011). This goes against the practices described in Uschold et al. (1996) and Noy et al. (2001), but results in an ontology that is specialised to the problem under review. On the other hand, Gray et al.

[13]`https://www.w3.org/TR/vocab-ssn/` (last accessed 10/11/2020)

Ontology	**Reference**	**Contributions**
OntoSensor	Russomanno and Goodwin (2008)	Addresses the lack of a formal conceptualisation of sensor networks in a computer-readable form.
CSIRO	Compton et al. (2009)	Introduced the ability to describe sensor compositions.
SSN	Compton et al. (2012)	Introduced a generic, field-independent ontology that addresses the lack of expressivity of previous ontologies.
WSSN	Bendadouche et al. (2012)	Extends the SSN by introducing a new communication pattern for wireless sensor networks.
SCO	Müller et al. (2013)	Extends the SSN by introducing concepts for sensors deployed in a cloud environment.
NIST	Schlenoff et al. (2013)	Specialises the SSN to the manufacturing industry by introducing new concepts such as robots, their actions and states.
SOSA	Janowicz et al. (2019)	Redefines the SSN ontology by addressing gaps exposed through its usage and updates the ontology to modern trends in sensor networks.

Table 2.2: A summary of the contributions made by each sensor ontology.

(2011) and Petersen et al. (2016) make use of the SSN ontology, while Klotz et al. (2018) and Catherine et al. (2019) use both the SSN and the SOSA ontologies.

2.2.4 | Evaluating Knowledge Graphs

Knowledge Graph (KG) evaluation can be split into two stages - the evaluation of the ontology, and the evaluation of the KG as a whole. Raad and Cruz (2015) lists a number of quality criteria upon which an ontology can be validated. Not all of these criteria are equally important for every ontology, as this depends on the ultimate use of the ontology. The quality criteria for ontologies as identified in Raad and Cruz (2015) are:

- Accuracy
- Completeness
- Conciseness
- Adaptability
- Clarity

- Computational efficiency
- Consistency

According to Raad and Cruz (2015), when there is already an ontology that is considered to be a gold standard for the domain of interest, this should be taken into consideration when evaluating an ontology. By mapping the concepts (classes), properties and relationships of the ontology to the gold standard, one can assess the accuracy, completeness and conciseness of the developed ontology. The drawback of this approach is that a gold standard is hard to find. The authors refer to another evaluation method whereby an ontology is compared to a text corpus that describes the domain. Term extraction methods are used to compile a list of important concepts that are then mapped onto the ontology. This also measures the accuracy, completeness and conciseness of the ontology.

Task-based evaluation methods measure the extent to which an ontology improves a task (Obrst et al., 2007; Raad and Cruz, 2015). This method is useful when the ontology is meant to be used for a particular task. The evaluation methodology measures the accuracy of knowledge base on responses provided by its inferential component (Obrst et al., 2007). On the other hand, criteria-based evaluation methods assess the ontology against a set of predefined criteria, be it structural or in terms of quality metrics. These two methods are more focused on measuring the adaptability, clarity, computational efficiency and consistency of an ontology (Raad and Cruz, 2015).

Gold standards are also used to evaluate KGs (Paulheim, 2017). In such cases, parts of the KG are constructed manually by domain experts and the automatically or semi-automatically generated KG is evaluated against the gold standard. Another evaluation approach is retrospective evaluation, where the KG (or parts of it) is inspected by human experts (Paulheim, 2017). The quality metric used in such an evaluation is normally the *accuracy*, that is a ratio of correct statements with respect to the total statements inspected (Paulheim, 2017). This approach is used by Rychtyckyj et al. (2017) who state that the EKG was tested via a number of queries (competency questions) whose results were manually scrutinised. When inconsistencies were found in the results these were investigated and the ontology corrected. A similar approach was used in Golebiowska et al. (2001) and Schabus and Scholz (2017) where the EKG was tested through its ability to retrieve the right information.

2.3 | Predictive Maintenance

Predictive Maintenance (PdM) received increasing attention with the rise of Industry 4.0, both in research and in industrial application (Ran et al., 2019). As discussed in Section 2.1.4, PdM has a number of advantages on the other maintenance strategies, however, it also brings about a number of challenges. PdM relies on the availability of machine health indicators, upon which machine faults are predicted (Ran et al., 2019). To produce these indicators, the enterprise must have a sensory infrastructure, which is not always readily available and involves an additional investment to implement. The industrial machines must be equipped with several sensors (or other capturing devices such as cameras) that are networked with the supporting software and hardware components that make up the PdM framework. Moreover, the business processes involved are more complex than their counterparts in other maintenance strategies.

The traditional approach to PdM involves the use of knowledge-based systems such as Expert Systems, where the domain knowledge is expressed as a number of *if-then* rules that specify the consequences of the various factors affecting a machine (Ran et al., 2019). Mathematical models are also used to model the physical processes of machines and to predict machine failure. In Rao et al. (2009), the authors use Symbolic Dynamic Filters to detect anomalies in time-series data while in Efthymiou et al. (2012) they are used to estimate the Remaining Useful Life (RUL) based on IIoT data. Kinghorst et al. (2017) also uses a mathematical model to predict machine failure. The authors use the Hidden Markov Model applied on sensor readings to predict the possibility that machine tools which are used in the production of semiconductor wafers become contaminated and therefore halt the production cycle.

Knowledge-based systems are expensive and time-consuming to implement as they require a deep knowledge of the machine (Ran et al., 2019). Moreover, industrial machines are complex systems that are hard to model. Knowledge-based systems are also unable to adapt to changes in the environment, such as the deterioration of the machine with long-term use or the introduction of new faults.

Another approach to PdM involves the use of ML, where one or more ML models are used to predict machine faults. Two types of predictions can be made: whether or not a machine will fail within a specified time period (classification problem) or predicting the RUL, that is the remaining time to the next fault (regression problem) (Yang et al., 2016). In some cases, as described by Yang et al. (2016), both types of models are used together so that when the classifier predicts a failure, the regression model can estimate the RUL. ML models are powerful predictive tools and their ability to identify hidden relationships in multivariate data makes them suitable for PdM (Carvalho et al., 2019). These methods require historical data to train the model, so availability of such records can be a determining factor of whether or not to opt for this method. In this dissertation we focus primarily on the ML approach to PdM.

2.3.1 | Machine Health Indicators

Predictive Maintenance (PdM) is based on the assumption that a machine gradually progresses towards a failure (Deloux et al., 2009). By monitoring the health condition of the machine, it is possible to determine when it is progressing towards a failure before the failure actually happens. The process of machine health degradation with time is shown in Figure 2.8. Once the machine health starts to deteriorate, a potential failure (P) can be predicted. If no remedial action is taken, the machine progresses to the point of functional failure (F) (Lorenzoni and Kempf, 2015). The period between P and F is referred to as the *P-F interval*. The P-F interval can be used to decide on the frequency of the machine health condition inspections. If the inspection intervals are longer than the P-F interval, there is the risk that the fault goes unnoticed and the machine runs to failure. On the other hand, frequent inspections may incur additional burden, both operational and financial.

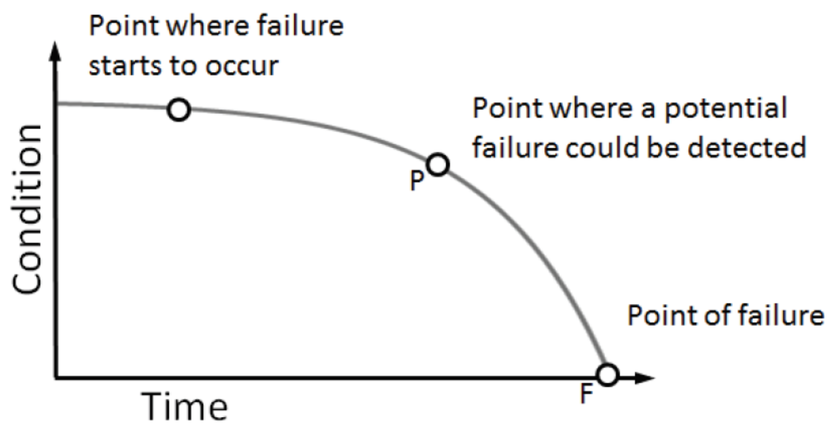

Point where failure starts to occur
Point where a potential failure could be detected
P
Condition
Point of failure
F
Time

Figure 2.8: The P-F curve - a plot showing the machine health deterioration with time until the point of functional failure (F). The period between a potential failure (P) and the actual failure (F) is referred to as the P-F interval. The figure was reproduced from Lorenzoni and Kempf (2015).

The health indicators that measure the machine condition vary according to the type of machine (Hashemian, 2010). For instance, pressure measurements are important health indicators for pumps and valves, but give little information about the condition of electric motors, where sound is much more indicative. Other commonly used health indicators include vibrations, temperature, surface friction and ultrasonic readings (Mobley, 2002).

All machines that involve motion vibrate. Changes in the vibration profile of a machine can be indicative of a deteriorating condition (Mobley, 2002). In the manufacturing industry, vibrations are used as health indicators for various types of machines. Schmidt and Wang (2018) present a case within the automobile manufacturing industry where vibrations of ball-bar components are used to measure for PdM. Similarly, Kharlamov et al. (2017) capture vibrations measurements for diagnostics of faults of power generating equipment. On the other hand, Seryasat et al. (2010) predict ball bearing failures based on machine vibrations, while Paolanti et al. (2018) measure the vibrations for PdM of cutting tools.

Temperature is also a common health indicator for industrial machines (Amihai et al., 2018; Kharlamov et al., 2017; Schmidt and Wang, 2018). Moreover, temperature together with pressure measurements have been used by Bruneo and De Vita (2019) for PdM of jet engines. In Umiliacchi et al. (2011), the authors describe how the various sensors deployed on trains contribute to the PdM of their mechanical equipment. Among the machine health indicators measured by these sensors are air pressure, current, voltage and the speed at which the motorised doors close.

Kovalev et al. (2018) describe the PdM and fault diagnosis of various devices found within households. Sensors were installed on power transformers, pumps, air heating, cooling and ventilation systems, gas boilers and lighting fittings. Such sensors measure sound, temperature, pressure, current and voltage, water and gas consumption, air velocity and water and gas leaks. Based on these health indicators, the authors use ML methods to predict possible faults of the devices and use data mining methods to diagnose the cause of the fault.

Another data source that is used in literature to determine the machines' condition are the logs generated by the same machines (Calabrese et al., 2020; Sipos et al., 2014). NLP techniques are used to extract features out of the unstructured text within the files. In Calabrese et al. (2020), the features are then used for PdM of woodworking industrial machines, while in Sipos et al. (2014), they are used for PdM of medical scanners.

To summarise this section, Table 2.3 lists the relevant literature together with the machine health indicators that were used to assess the condition of the machine. It shows how the health indicators measured vary according to the type of equipment.

Reference	Equipment	Machine Health Indicator						
		VBR	TMP	PRS	SND	CUR	LOG	OTH
Amihai et al. (2018)	Pumps and motors	✓	✓					
Bruneo and De Vita (2019)	Jet engines		✓	✓				
Calabrese et al. (2020)	Woodworking industrial machines						✓	
Efthymiou et al. (2012)	*	✓	✓	✓				✓
Farokhzad et al. (2012)	Centrifugal water pumps	✓						
Gandhi et al. (2018)	Ball-bar components							✓
Kanawaday and Sane (2017)	Slitting machines			✓				✓
Kharlamov et al. (2017)	Power generating machines	✓	✓	✓				
Kovalev et al. (2018)	Household devices		✓	✓	✓	✓		✓
Paolanti et al. (2018)	Wood cutting machines	✓				✓		✓
Schmidt and Wang (2018)	Ball-bar components	✓	✓					✓
Seryasat et al. (2010)	Ball bearings	✓						
Sipos et al. (2014)	Medical scanners						✓	
Susto et al. (2015)	Ion implantation tools			✓		✓		✓
Umiliacchi et al. (2011)	Mechanical equipment on trains	✓	✓	✓				

Table 2.3: The machine health indicators used for the various types of equipment. Machine health indicators are denoted as follows: VBR - Vibrations; TMP - Temperature; PRS - Pressure; CUR - Current/Voltage; LOG - Log files; OTH - Other indicators. (* Not specified)

2.3.2 | The Machine Learning Approach to Predictive Maintenance

The major steps of the ML approach to PdM are outlined in Figure 2.9. First, the data is identified, collected and stored, proceeding with the modification of such data to make it more suitable for the learning activity (Soares, 2015). Consequently, a suitable model is trained and validated. Finally, the PdM system is deployed and maintained. While the steps are sequential in nature, the process is reiterated periodically to allow the model to be retrained with newly acquired data, thus making the system adaptable to changes in the environment (Sipos et al., 2014; Soares, 2015).

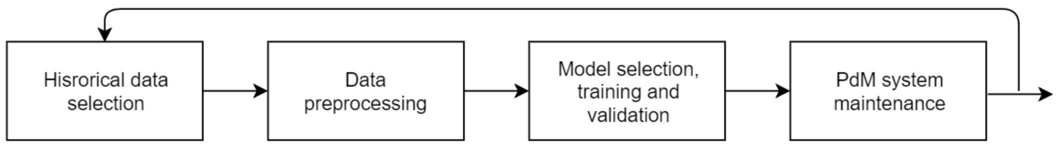

Hisrorical data selection
Data preprocessing
Model selection, training and validation
PdM system maintenance

Figure 2.9: The life cycle of the Machine Learning approach to Predictive Maintenance (adapted from Soares (2015)).

2.3.2.1 | Historical Data Selection

The Historical Data Selection step deals with capturing the data and storing it (Soares, 2015). The data consists of event data keeping trace of what happened and the machine health indicators that were discussed in Section 2.3.1, which are mostly IIoT data (Efthymiou et al., 2012). The type of health indicators needed depend on the type of equipment under surveillance, so these must be chosen accordingly (Hashemian, 2010). The integration of IIoT data provides a number of challenges. Such data is heterogeneous since it originates from different sensors that measure different phenomena (Khaleghi et al., 2013; Schmidt et al., 2017). Moreover, Khaleghi et al. (2013) state that sensors tend to be inaccurate and multiple sensors measuring the same phenomenon will produce conflicting observations. The authors suggest resolving these conflicts by taking a consensus average in order to have an approximation of the truth. Another difficulty in integrating IIoT data is that the sampling frequencies of the various sensors may vary (Soares, 2015). Some measurements, such as the chemical concentration of a substance, are more expensive to obtain than others, so these would have a slower sampling frequency. Nonetheless, the integration of the data sources is fundamental for the success of the PdM model.

The data storage component can take many forms. Data can be kept in flat files (Calabrese et al., 2020; Susto et al., 2015) or stored in a data lake (Spendla et al., 2017). NoSQL databases such as Apache Hive[14] are also used (Kovalev et al., 2018). Schmidt et al. (2017), Medina-Oliva et al. (2014) and Kharlamov et al. (2017) store the IIoT data within an EKG. In Schmidt et al. (2017), the authors opt for an OBDA architecture while Kharlamov et al. (2017) use an adaptation of the OBDA architecture that catered for streaming data. On the other hand, Medina-Oliva et al. (2014) use an RDF store.

The data storage component may also take the form of a Time Series Databases (TSDBs) (Zhang et al., 2019). These databases are designed to handle the storage of continuous data streams, such as IIoT, where all the data is time bound. A drawback of TSDBs is that they lack semantic support, hence when storing IIoT data the relationship between the sensor observations and the sensor generating the data is not captured. In Zhang et al. (2019), the authors address this limitation by proposing a semantically enriched TSDB that is achieved by having an ontological layer on top of the TSDB.

2.3.2.2 | Data Preprocessing

The goal of the Data Preprocessing step is to produce a dataset that facilitates the learning process of the ML model (Soares, 2015). It consists of four activities: data cleaning, data transformation, data reduction and data labelling.

a) Data cleaning: IIoT data is often imprecise, noisy and incomplete (Soares, 2015). So the data is cleaned by handling any missing information and by correcting erroneous measurements.

Records with missing data are either imputed or discarded (Soares, 2015). Imputations replace a missing value with an estimate, which can be calculated as the mean for the respective variable (Soares, 2015) or a moving average (Kanawaday and Sane, 2017). Another method is to carry forward the last known value (Soares, 2015). Regression models can also be used to estimate the missing values. In some cases, incomplete data is discarded. For example, in Amihai et al. (2018), if the recordings from a sensor are available for less than a day then the records for the respective sensor are discarded. Any dimensions that are not useful, such as constant dimensions, are also removed (Borgi et al., 2017).

Incorrect data samples are hard to identify unless they are outliers (Soares, 2015). Outliers are data samples that are inconsistent with the remaining data. Outlier detection techniques can be categorised into two - *univariate* that take into consideration only the

[14] `https://hive.apache.org/` (last accessed 01/10/2020)

variable under review, and *multivariate* that consider multiple variables. Multivariate methods are mostly effective when the variables are highly correlated since the method exploits the correlation to detect abnormalities. The identified outliers are either removed or replaced by an imputation. Another method is to average the data to reduce the impact of outliers (Amihai et al., 2018). Rolling averages maintain the trend of the data recorded but dilute the effect of noise (Kinghorst et al., 2017). The window length of such rolling averages must be chosen carefully as a time window that is too long would dilute the trend of the variable whilst one that is too short will not reduce the effect of noise.

b) Data transformation: In this activity the data is transformed into a form that favours the learning process of the ML model. The variables within the data normally have different magnitudes and this can hinder the learning process (Soares, 2015). *Normalisation* and *scaling* techniques are used to transform the variables to a common magnitude. Some examples of these techniques are:

Min-max normalisation transforms the features to a range between a minimum and a maximum number (Soares, 2015). The normalised value v' is obtained through Equation 2.1 where v is the original unnormalised value, and A_{min} and A_{max} are respectively the minimum and maximum values of feature A. For example, Bruneo and De Vita (2019) use this technique to rescale all variables to values between -1 and 1.

$$v' = \frac{v - A_{\min}}{A_{\max} - A_{\min}} \tag{2.1}$$

Z-score normalisation (sometimes referred to as *standardisation*) transforms the data such that they have a mean of *0* and a standard deviation of *1* (Soares, 2015). This is obtained through Equation 2.2 where value v is transformed into a normalised value v'. $\bar{A}$ is the mean of feature A and σ_{A} is the standard deviation of A. This technique is particularly useful when the minimum and maximum of a feature are not known or when the data contain outliers (Soares, 2015).

$$v' = \frac{v - \bar{A}}{\sigma_{\mathrm{A}}} \tag{2.2}$$

Decimal scaling divides all values of a feature by a common denominator so that all new normalised values v' fall in the range $0 < v' < 1$ (Han et al., 2011). This is achieved through Equation 2.3 where v is the unnormalised value and j is the smallest integer such that $max(v') < 1$.

$$v' = \frac{v}{10^j} \tag{2.3}$$

In production, machine failures are rare when compared normal operations and this is reflected in the collected dataset (Ran et al., 2019). This causes a *class imbalance*, where the samples representing a machine failure are considerably less than those describing normal operations. Such a bias diminishes the learning potential of the ML model and should be addressed at preprocessing stage. This is done through *resampling*, where either artificial samples are generated for the lesser class (known as *oversampling*), or by removing samples from the dominant class (known as *undersampling*) (Kovalev et al., 2018).

c) Data reduction: IIoT data is typically high dimensional and contains a large number of samples (Fernandes et al., 2019). Such volumes of data increase the computational requirements of the learning process and decrease the learning capabilities of most ML algorithms. The latter is due to the increased risk of overfitting when certain models are trained on very large datasets (Fernandes et al., 2019). Moreover, IIoT data tends to be highly correlated, meaning that some dimensions do not add new information (Khaleghi et al., 2013). There are two categories of data reduction methods, one aimed at reducing the feature space and another aimed at reducing the number of samples (Soares, 2015). The former is further sub-categorised into *dimensionality reduction* techniques and *feature selection* techniques.

Through dimensionality reduction, the data is transformed into a smaller space by projecting it into dimensions that are made up of a combination of the original dimensions (Soares, 2015). A popular dimensionality reduction technique is Principal Component Analysis (PCA), that was also used in Kovalev et al. (2018) and Baptista et al. (2018).

In feature selection, a subset of features is chosen from the original feature space and these include three classes: *filter*, *wrapper* and *embedded* (Soares, 2015). Filter models rank the features by some evaluation metric (such as Pearson's correlation) to then take only the best performing features. Wrapper methods, on the other hand, use a learning algorithm to determine the relevance of a feature subset to the target concept. Some ML models have feature selection embedded within the learning algorithm, hence the embedded method (Fernandes et al., 2019). All three feature selection methods are used for PdM, for example Gandhi et al. (2018) uses a filter method while Sipos et al. (2014) and Kinghorst et al. (2017) use a wrapper method. The embedded method comes with the use of ML algorithms with in-built feature selection such as *Random Forest* and *Support*

Vector Machines (SVM), such as those used in Baptista et al. (2018), Calabrese et al. (2020) and Sipos et al. (2014).

Some machine health indicators, such as vibrations, are measured as waveforms (Seryasat et al., 2010). In such cases, Fast Fourier Transform (FFT) is used to transform waveforms to the frequency domain that can then be binned into frequency bins. The value of each frequency bin can then represent the waveform as data. Given the vast range of frequencies, such data results in a lot of dimensions. Seryasat et al. (2010), Farokhzad et al. (2012) and Amihai et al. (2018) use mathematical functions to compute single-dimensional properties of the FFT data, thus reducing the number of dimensions. Seryasat et al. (2010) and Amihai et al. (2018) calculate the Root Mean Square (RMS) of the wave and use it as a feature, while Farokhzad et al. (2012), in addition to the RMS, also computes the mean, standard deviation, variance, kurtosis, skewness and slippage.

In the case of text-based features, such as log files, the *bag of words* method can be used to extract the most important features from the text (Calabrese et al., 2020). This reduces the long strings into word vectors. Moreover, when the training data contains too many instances, numerosity reduction is applied (Soares, 2015). A sample of the dataset is taken in order to reduce the volume.

d) Data labelling: The final activity of the Data Preprocessing step is the labelling of the various samples within the training dataset. For classification learning, the samples are labelled to the corresponding class (Calabrese et al., 2020; Susto et al., 2015), while for RUL learning the label would consist of a continuous variable (Ran et al., 2019). Moreover, for classification-based PdM, Susto et al. (2015) state that since the ML model needs to predict the machine fault a time t in advance, the samples corresponding to t before a fault are attributed to the positive class. So for a machine failure at time f, all samples observed between time $f - t$ and f are labelled as positive. The authors also state that the time a fault can be predicted in advance depends on the fault being observed and that this should be determined through experimentation with different fractions of the maintenance cycle. Such experimentation is used by Amihai et al. (2018), Calabrese et al. (2020) and Sipos et al. (2014). While Amihai et al. (2018) and Sipos et al. (2014) consider time windows of between one and seven days ahead in their experiments, Calabrese et al. (2020) consider time windows of 10 to 30 days ahead. Paolanti et al. (2018) state that this time window can be determined by examining the data.

2.3.2.3 | Model Selection, Training and Validation

The purpose of this step is to train and validate an ML model for a PdM problem. Soares (2015) describes two methods for choosing a model - a *white-box* and a *black-box* method. The former is model-driven and the physical mechanics of the machine or device are modelled mathematically. Such method is very complex and requires input from the domain experts. In the ML approach, the black-box method is often used, where the method is data-driven. In this case, little domain knowledge is required since the model will learn from the data provided (Soares, 2015). Finding the right model and fine-tuning it requires some experimentation.

Before training the model, a portion of the training data is set aside for testing the model (Calabrese et al., 2020; Kanawaday and Sane, 2017). This testing set is typically between 20% to 30% of the training data. The remaining part of the training data is split into the training set and the validation set. Then, a number of ML models are trained and validated using varying hyperparameters (Calabrese et al., 2020). *Overfitting* may occur when training a model and this happens when the ML algorithm performs well on the training data but poorly on unseen samples (Soares, 2015). To prevent overfitting, *cross-validation* is used where the training and validation steps are iterated, changing the training and validation sets in each iteration. In many cases, *k-fold cross-validation* is used (Calabrese et al., 2020; Gandhi et al., 2018; Sipos et al., 2014; Soares, 2015). This consists of the training set being split into k parts (folds) and the training-validation cycle is iterated k times, each time a different fold is chosen as the validation set. However, since in PdM the dataset is temporal, this fold iteration causes a situation where the past is predicted based upon the future (Bergmeir and Benítez, 2012). In Bergmeir and Benítez (2012), although the authors found k-fold cross-validation to perform equally well on time-bound data, they recommend the use of *block cross-validation* when dealing with such datasets, where the order of the folds is preserved during the iterations. Following cross-validation, the combination of hyperparameters that perform best are then evaluated using the testing data to decide upon the ultimate model to be used in production (Calabrese et al., 2020). Table 2.4 (Pg 40) shows the various ML models used in literature and whether these address a classification problem or a regression problem (RUL). It also shows the evaluation metrics used in each research project.

As shown in Table 2.4 (Pg 40), the Root Mean Square Error (RMSE) is often used to evaluate regression models. This is calculated using Equation 2.4, where n is the number of samples in the test set, $\hat{y}_i$ is the predicted value and y_i is the actual value.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{2.4}$$

On the other hand, for classification problems, *Accuracy*, *Precision* and *Recall* are most commonly used. These are calculated by Equation 2.5 to 2.7 respectively, where *TP* is the number of True Positives, *TN* is True Negatives, *FP* is False Positives, and *FN* is False Negatives. Precision and recall are important metrics in PdM since the former measures the fraction of positive predictions that are correctly classified, whilst the latter measures the fraction of positive predictions that are correct (Susto et al., 2015). In PdM, a low precision implies more false positives, hence unnecessary stoppages and unused machine/parts lifetime, while a low recall signifies more false negatives and therefore unexpected stoppages. Another way to use *Precision* and *Recall* is to use the *F1-score*, that calculates the harmonic mean of both measures (Han et al., 2011). The F1-score is calculated using Equation 2.8.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{2.5}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{2.6}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{2.7}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{2.8}$$

The results achieved by other studies tackling PdM vary. Calabrese et al. (2020) report a precision of *0.991* and a recall of *0.996* when using a GBC to predict faults of woodworking industrial machines 30 days in advance. Susto et al. (2015), on the other hand, report a precision of *0.613* and a recall of *0.63* for PdM of ion implantation tools using an SVM. Paolanti et al. (2018) obtained an accuracy of *0.95*, a precision of *0.94* and a recall of *0.95* when using an RF to predict failures of cutting machines, while Kanawaday and Sane (2017) measured an accuracy of *0.987* in predicting failures of slitting machines using a deep neural network.

Reference	Prediction Type	ML Model/s	Evaluation Metrics
Amihai et al. (2018)	RUL	Random Forest	RMSE
Baptista et al. (2018)	RUL	k-Nearest Neighbors Random Forest Neural Networks Support Vector Regression Linear Regression	RMSE
Bruneo and De Vita (2019)	RUL	Long Short-Term Memory	RMSE
Calabrese et al. (2020)	Classification	Gradient Boosting Random Forest Extreme Gradient Boosting	Accuracy Precision Recall
Farokhzad et al. (2012)	Classification	Neural Networks	MSE
Gandhi et al. (2018)	RUL	Decision Tree Random Forest	*
Kanawaday and Sane (2017)	Classification	Support Vector Machines Naive Bayes Deep Neural Networks Decision Trees	Accuracy
Kovalev et al. (2018)	Both	Long Short-Term Memory Recurrent Neural Networks Hidden Markov Model	No. of FP No. of FN
Lorenzoni and Kempf (2015)	Classification	Bayesian Networks	*
Paolanti et al. (2018)	Classification	Random Forest	Accuracy Precision Recall
Sipos et al. (2014)	Classification	Support Vector Machines	AUC Precision Recall
Susto et al. (2015)	Classification	Support Vector Machines k-Nearest Neighbors	Accuracy Precision Recall
Umiliacchi et al. (2011)	RUL	Bayesian Networks	*

Table 2.4: A summary of the ML models used in the various research projects and the metrics used to evaluate the models. (* Not specified)

2.3.2.4 | System Maintenance

The System Maintenance step consists of activities related to the upkeep of the PdM system in order to maintain the required performance (Soares, 2015). Changes in the production environment, such as the introduction of new machines or the replacement of sensing devices, may lead to a degradation of the system performance. These changes must be reflected in the system:

1. by ensuring that any new data sources are captured by the system;
2. by ensuring that the required data transformations are carried out on the newly acquired data; and
3. by ensuring that the predictive model is representative of the new reality.

2.3.3 | Predictive Maintenance Supported by Knowledge Graphs

In Section 2.2.2.2 we discussed the uses of an EKG, amongst which is PdM. The implementation described by Schmidt (2018) uses an EKG as the basis for PdM of industrial machines within an automobile manufacturer. The EKG combines three heterogeneous data sources to store information about machines, parts and sensor observations. The author tests different ML models, amongst which kNN, SVM, RF and GBC, on four years of data for 29 similar machines to predict failures of a cutting tool. In this study, the EKG is leveraged as a data integration paradigm for the data, that would have otherwise been challenging due to the diversity of the data sources. The data integration capabilities of the EKG are also exploited in Kharlamov et al. (2017), where the authors refer to a manufacturer of energy generating machines that offers a PdM service to its clients whereby it monitors the performance of the said machines in production to detect potential issues. When such issues are detected, the manufacturer's experts extract the required data about the machine and analyse it to eventually decide on a way forward. The data analytics are carried out over an EKG that combines data from various sources and takes various forms. The authors state that without a strong data integration paradigm such as the EKG, the company would need specialised personnel to query the complex databases for the intended data.

Medina-Oliva et al. (2014) and Voisin et al. (2013) tackle PdM of ship engines, where the data generated by the different components is integrated in a KG. Apart from using the KG to integrate heterogeneous data, it serves to standardise the vocabulary so that different databases can be queried using the same semantics. The authors also exploit the ontology to define a rules about the domain, for example, a fuel engine cannot develop an electrical fault, thus the consistency of the data is ensured. Furthermore, in Efthymiou et al. (2012) the authors make use of a KG to store knowledge about the characteristics of past machine failures and the maintenance steps taken. This is put to use when a fault is predicted to provide a diagnosis and suggest a solution. Once again, the authors take advantage of the KG's ability to combine data having different structures to preserve the knowledge that comes from various sources.

2.4 | Summary

In this chapter, we briefly described the semiconductor manufacturing process and how wire bonding machines contribute towards the end products. We also introduced Knowledge Graphs and how ontologies are used to define them. We proceeded by discussing Enterprise Knowledge Graphs and the benefits they can contribute to an enterprise, especially where it comes to decision making and the integration of heterogeneous data sources. We also saw how the properties of the EKG makes it a powerful paradigm for modelling sensor networks.

We then switched focus to Predictive Maintenance, explaining the benefits it offers and the challenges involved in its implementation. We also discussed machine health indicators and how these were used in previous research for PdM. Moreover, we described to Machine Learning approach to PdM and the state of the art of their implementations. Finally, we saw how KGs can be used to support PdM

3

Methodology

In this chapter we explain how this research was conducted. First, we introduce the datasets that were used in this dissertation and describe how this data was collected. We proceed to describe the steps taken to design an ontology and implement an EKG, and how the latter was populated with the industrial data. Next, we move on to explain how we trained and evaluated various ML models, and the experiments we carried out in order to choose the ones that gave the best performance. Finally, we describe how we produced a PdM framework consisting of a graphical interface that used the predictive models and was supported by the EKG. The system projected key machine health indicators to its users and also alerted them of possible upcoming faults.

3.1 | The Data

The data used in this dissertation is actual industrial machine data that was generated in a semiconductor manufacturing environment. These datasets were made available to us by the manufacturing firm for the purpose of this research. We held several meetings with the domain experts to gain insights about the processes, machines, data and malfunctions of the said machines. Furthermore, we complemented this knowledge with relevant information obtained from the machine manuals and documentation.

The machines in question were wire bonders, that as explained in Section 2.1.2, play an important role in the manufacturing of ICs. We scoped the study to four wire bonding machines, all of the same brand and model, namely the IConn PLUS[1] ball bonder by Kulicke & Soffa.

[1]`https://www.kns.com/Products/Equipment/Ball-Bonder/IConn-PLUS` (last accessed 05/10/2020)

Three datasets were used for this dissertation, that were produced by different components of the wire bonders. These were the *Equipment Tracking System (ETS) event logs*, the *machine logs* and the *IIoT data*. The machine logs were in JavaScript Object Notation (JSON) format while the other datasets consisted of Comma Separated Values (CSV) files. Some additional details about the datasets are provided in Table 3.1.

Dataset	**Number of machines**	**Number of days in dataset**	**Number of files**	**Number of variables**	**Number of samples**	**Total size**
ETS event logs	489	23	47	27	2,082,554	563 MB
Machine logs	435	40	167	91	4,938,937	10.7 GB
IIoT data	4	113	236,845	1,093	236,845	2.7 GB

Table 3.1: Information about the datasets used in this dissertation.

We explored the datasets to get a better understanding of the data. We first merged all the data scattered across numerous files into one file per dataset. To do this, we wrote Python[2] (version 3.7.2) scripts using the Pandas[3] library (version 0.24.2). For the machine logs, the Json[4] library (version 2.0.9) was also used. Apart from merging them into one file, the machine logs were converted from nested JSON format to CSV. The Matplotlib[5] (version 3.0.3) library was used to visualise the data in Python.

a) Equipment Tracking System (ETS) event logs: The ETS keeps track of all machines in operations. It captures machine events and operations, which are then stored in CSV files. The system generates two such files per day, one covering the day shift (between 06:00 and 18:00) and another for the night shift (between 18:00 and 06:00). These files are stored on a file server.

Each file contains the events for all machines in operations. Each row within the file describes an event that happened, whether it is a production step, an error or a change in the state of the machine. Events are recorded once ended. So if a machine prompted an error at time t and the operator took n minutes to act upon it, the event is recorded with timestamp t+n. The duration of the event (n) is still recorded, so to determine when an error occurred both the timestamp and the duration must be taken into account.

[2]`https://www.python.org/downloads/release/python-372/` (last accessed 17/10/2020)

[3]`https://pandas.pydata.org/pandas-docs/stable/whatsnew/v0.24.2.html` (last accessed 17/10/2020)

[4]`https://docs.python.org/3.7/library/json.html` (last accessed 17/10/2020)

[5]`https://matplotlib.org/3.0.3/contents.html` (last accessed 17/10/2020)

The variables that were useful for this dissertation are described in Table 3.2. More variables are described in Appendix A. The dataset contained events for all wire bonders for the period between 20/03/2019 and 12/04/2019. The timestamp consisted of a concatenation of the date and time as a string in the format `YYYYmmdd HHMMSSfff`[6]. It was also observed that, while the description of the event is normally chosen from a list of predefined messages, in some cases the description was manually inputted by the operators. This created multiple strings referring to the same event.

Variable	Datatype	Description
MANUFACTURINGMODULE	String	The physical location of the machine (known as tunnels).
EQUIPMENTNAME	String	The name of the machine, also used as its unique identifier.
TXNTIMESTAMP	Datetime	The timestamp of the record. This is taken to be the end time of the event.
DURATION	Float	The duration of the event.
EQPSTATE	String	The state of the machine.
REASONCODEID	String	A description of the event.

Table 3.2: The main variables of the Equipment Tracking System event logs.

b) Machine Logs: The wire bonders generate their own logs. Every five minutes, all the machines dump their log entries into the same JSON file that is stored on a file server. A new log file is started every six hours. The dataset contained logs for the period between 03/03/2019 and 12/04/2019.

The log entries consisted of a timestamp and the wire bonder name, followed by a set of readings from various sensors deployed on the machine. The log also contained the settings of the machine as at the time of recording. The sensor observations include the bond forces, the current applied to cause the spark, ball placement errors and the temperatures at various zones of the machine. The timestamp is presented as a Posix epoch. It was observed that whenever the sensor reading is undetermined, the value was set to 999 or -999. The dataset also contained several dimensions that were constant. More details on the variables that were used for this dissertation are presented in Appendix A.

c) IIoT data: The manufacturing firm installed more sensors on the wire bonders to gain additional insights about the machines. These sensors measure the acceleration, vibrations, sound, magnetism, temperature, pressure and humidity of the machine. The

[6] Y - year; m - month; d - day; H - hour; M - minute; S - second; f - milliseconds.

acceleration, vibrations and magnetism are captured in three dimensions (x, y and z axis). Apart from the temperature, pressure and humidity, all observations are measured in waveforms. These waveforms are transformed to the frequency domain using Fast Fourier Transform (FFT) and binned in frequency bins. Acceleration, vibrations and magnetism waveforms are binned into 64 frequency bins for each axis (x, y and z axis), while sound is binned into 512 frequency bins. The mean amplitude for each frequency bin is then recorded.

Each machine produces a set of observations every three minutes, which are stored as a CSV file. Each file contains only one sample, which is made up of a timestamp and the set of sensor observations. The timestamp for the IIoT is also represented as a Posix epoch. The machine identifier (its name) does not feature within the data but is part of the file name. Moreover, the CSV file does not have any headers. As with the other data files, the IIoT data files are stored on a file server.

The dataset contained sensor observations from four machines for the period between 30/11/2018 and 12/04/2019. In analysing the data we found that there are a lot of outliers in the data. Outliers were measured using the *z-score* statistic and visualised through box plots. Two such plots are featured in Figure 3.1, that show the outliers in the *Pressure* and *Humidity* variables. We also noticed that the FFT variables are highly correlated. We used the Pearson correlation coefficient to measure the pairwise linear correlation. A heatmap of the pairwise Pearson correlation coefficient for the FFT dimensions of vibrations along the x-axis is portrayed in Figure 3.2. As can be seen in the chart, many pairs have a coefficient towards the upper end of the spectrum. The other FFT variables show a similar scenario. More details on the IIoT dataset are presented in Appendix A.

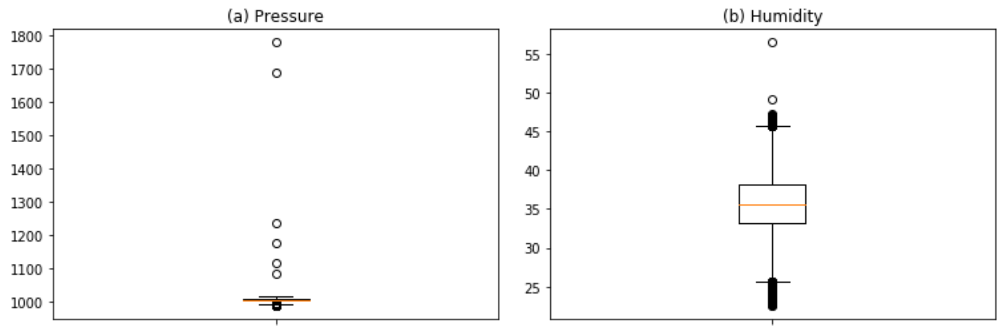


Figure 3.1: Box plots showing outliers in the (a) Pressure and (b) Humidity variables.

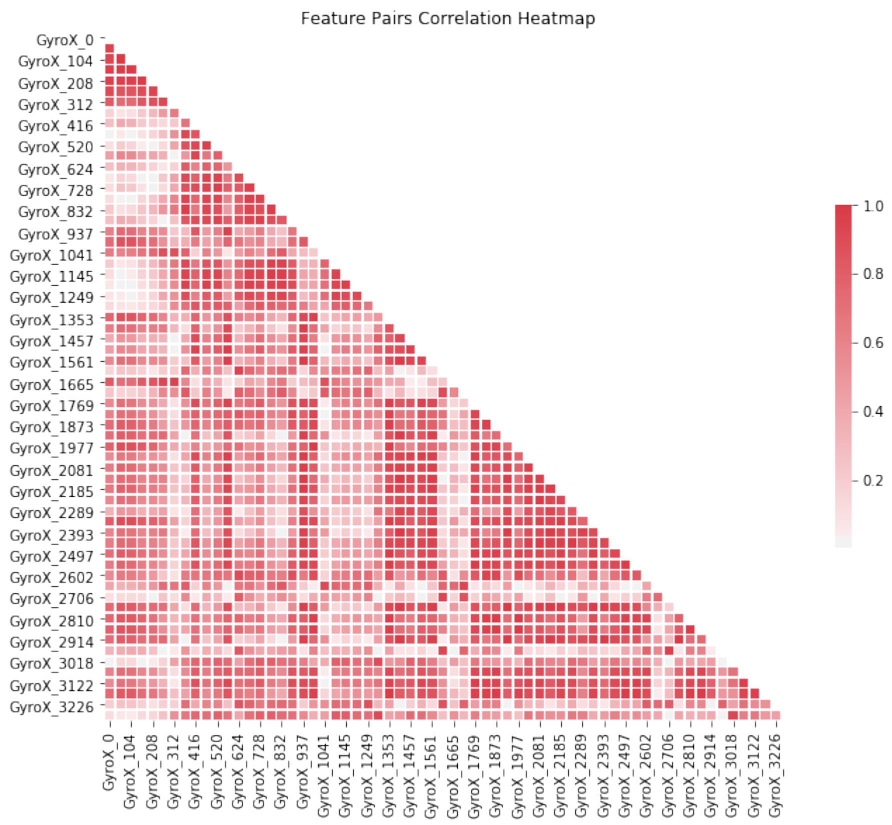


Figure 3.2: A heatmap showing the pairwise correlation between the FFT dimensions for vibrations along the x-axis.

3.2 | Creation of the Enterprise Knowledge Graph

To design and build the EKG, we followed the approach of Pan et al. (2017), that we described in Section 2.2.2.1. We first developed the ontology of the EKG. Next, we developed ETL-like scripts that transform the data into the required structure and push them into the EKG. Lastly, we tested the EKG for correctness and completeness.

3.2.1 | Ontology Development

The approach we took to develop the ontology is based on the one described in Noy et al. (2001), discussed in Section 2.2.1.1. This approach constitutes a simple yet effective procedural method, that can be considered as a beginner's guide to ontology development. Moreover, in contrast with the method proposed by Uschold et al. (1996), this approach considers reusing existing ontologies at the very beginning of the process, which, in view of the number of existing ontologies in the sphere of sensor networks, we considered that this method would speed up our work.

Step 1 - Determining the domain and scope of the ontology: The domain was determined by the ultimate use of the EKG, that is PdM. Therefore, our domain is the industrial domain and more specifically industrial machines. The scope was in part dictated by the data and variables we had at hand, which we described in Section 3.1. We also considered the questions that the EKG would need to answer in order to predict machine faults. These questions are listed below:

In which state was wire bonder x at time t?
What is the latest temperature observation of wire bonder y?
Which sensors measure vibrations?
Which wire bonder have a sensor of type a?
What are the latest sensor observations of wire bonder z?
What was the change in temperature for wire bonder x between time t and time t+n?

Based on such questions, we determined that the scope of the ontology should be the wire bonders, their states and events, their sensors and the observations made by these sensors.

Step 2 - Reusing existing ontologies: As we discussed in Section 2.2.3.1, there already exist several ontologies that model sensor networks and their observations. These were examined and considered. The one presented by Schlenoff et al. (2013) is an adaptation of the SSN ontology (Compton et al., 2012) specialised for the manufacturing domain and was fit for our purpose. However, the actual OWL ontology was not publicly available.

The SSN ontology (Compton et al., 2012) and its latest improvement, the SOSA ontology (Janowicz et al., 2019) were also adequate. Although these ontologies did not cover the full scope of our ontology, they catered for most concepts. Relevant concepts from the SSN and SOSA ontologies are shown in Figure 3.3, where concepts in blue derive from the SSN ontology and concepts in green from the SOSA ontology. These concepts were taken into account in designing the ontology.

Step 3 - Listing important terms: We devised a list of prominent terms in a brainstorming manner. The list was then short-listed by grouping similar terms or removing

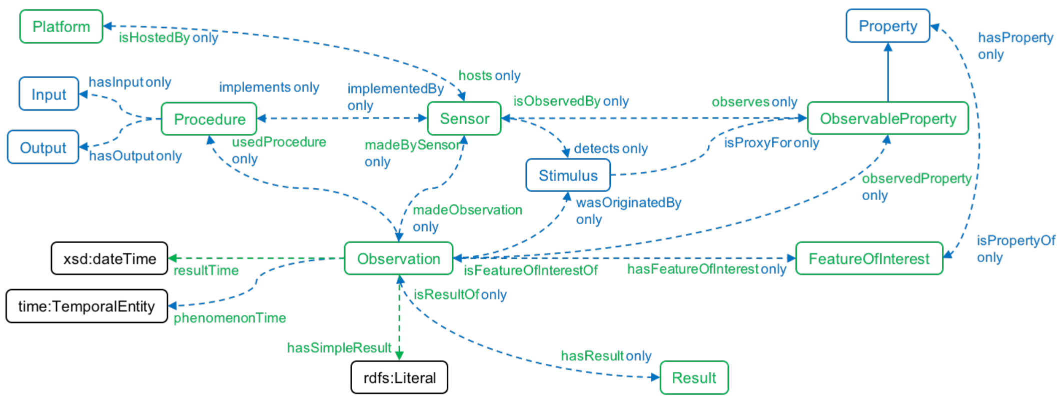


Figure 3.3: A partial representation of the SSN and SOSA ontologies. (Reproduced from `https://www.w3.org/TR/vocab-ssn/`)

those less relevant. The identified terms are shown in Table 3.3. The terms *operator*, *product* and *work order* were deemed less relevant for PdM and therefore discarded.

Term	Class
Wire bonder	Platform
Sensor	Sensor
Sensor observation	Observation
Pressure; Vibrations; Temperature; etc.	ObservableProperty
Error; Machine state	State
Tunnel	Place
Operator; Product; Work order	-

Table 3.3: The relevant terms identified and their mapped classes.

Step 4 - Defining the classes and their hierarchy: We mapped the terms listed in the previous step onto the SSN and SOSA classes. This left only three unmatched terms. These are: *error*, *machine state* and *tunnel*. Then we considered generalising the terms to form the hierarchy. Thus *error* and *machine state* were generalised into *state* and *tunnel* into *place*. For the latter, we considered the existing class within the DOLCE+DnS Ultralite (DUL) ontology, that is compatible with the SSN ontology. The final mappings are shown in Table 3.3.

Step 5 - Defining slots and facets: The slots (properties) for each class were defined subject to the data we had available. For example, a wire bonder has a name and an alias while a sensor has a name, a description and a type. The facets (rules such as slot cardinality, and permitted types and values) present in the SSN, SOSA and DUL ontologies were reviewed to ensure they applied to our case. These had an impact on the design of our ontology, for example, for the Wire Bonder class to host a *sosa:Sensor*, it must be a sub-class of *sosa:Platform*. Using the OWL language, this slot is defined as shown in Listing 3.1, where the *sosa:isHostedBy* property can belong only to the domain *sosa:sensor* in relation to objects of the class *sosa:Platform*. Similarly, for the Wire Bonder to be located in a Tunnel, the latter being a sub-class of *dul:Place*, it must also be a sub-class of *dul:Entity*. We were also forced to add another class *sosa:Procedure* due to a facet in the *sosa:Sensor* class, which mandates that a sensor has at least one procedure describing how an observation is made. We also considered using the class *dul:State* to record the different states of the wire bonders, however, the facets of this class mandate the inclusion of other classes, that would have over-engineered the ontology. We therefore discarded this option and created a new class.

```
sosa:isHostedBy rdf:type owl:ObjectProperty ;
        schema:domainIncludes sosa:Sensor ;
        schema:rangeIncludes sosa:Platform ;
        rdfs:isDefinedBy <http://www.w3.org/ns/sosa/> ;
        rdfs:label "is␣hosted␣by"@en .
```

Listing 3.1: The definition of the isHostedBy relationship between a Sensor and a Platform.

Although the method described by Noy et al. (2001) is sequential, from Step 4 we iterated back to Step 2 to ensure that we reuse as much as possible from existing ontologies, without adding unnecessary effort. Finally, the OWL 2 ontology was developed using Protégé[7] (version 5.5.0). A partial representation of the resulting ontology is shown in Figure 3.4, where classes shown in blue belong to the SOSA ontology, classes in green belong to the DUL ontology and classes in yellow were created specifically for this research project. The white boxes represent literal values. The full ontology model can be seen in Appendix B.

The *Wire Bonder* class hosts *sosa:Sensor* instances and is located in a *Tunnel*, that is a sub-class of *dul:Place*. Wire bonders can also have a *State* that is defined by a name and a timestamp. Sensors, on the other hand, observe some *sosa:ObservableProperty* and can make observations (*sosa:Observation*) that consist of a timestamp and a value. An example of an observation as modelled by the ontology is illustrated in Figure 3.5.

[7]`https://protege.stanford.edu/` (last accessed 08/10/2020)

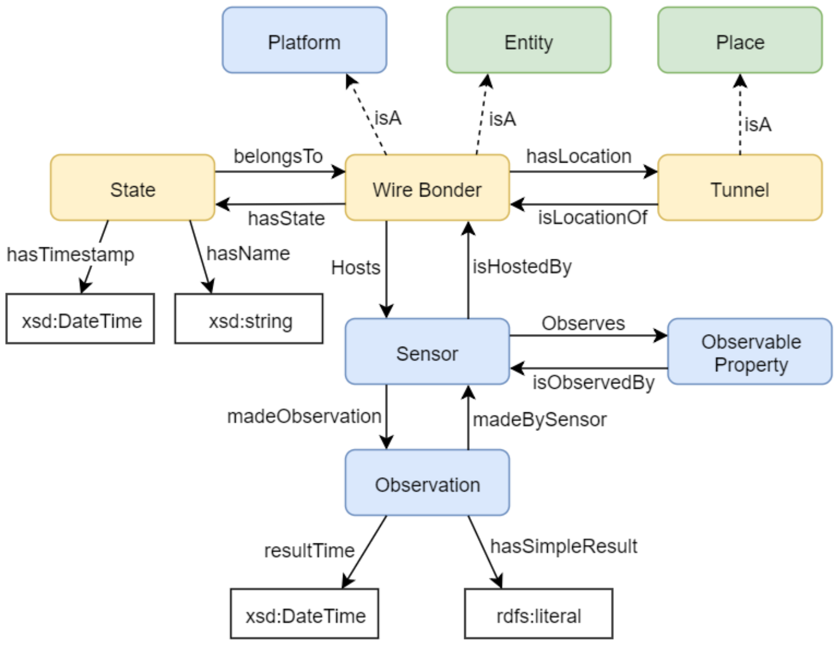


Figure 3.4: A partial representation of the ontology produced.

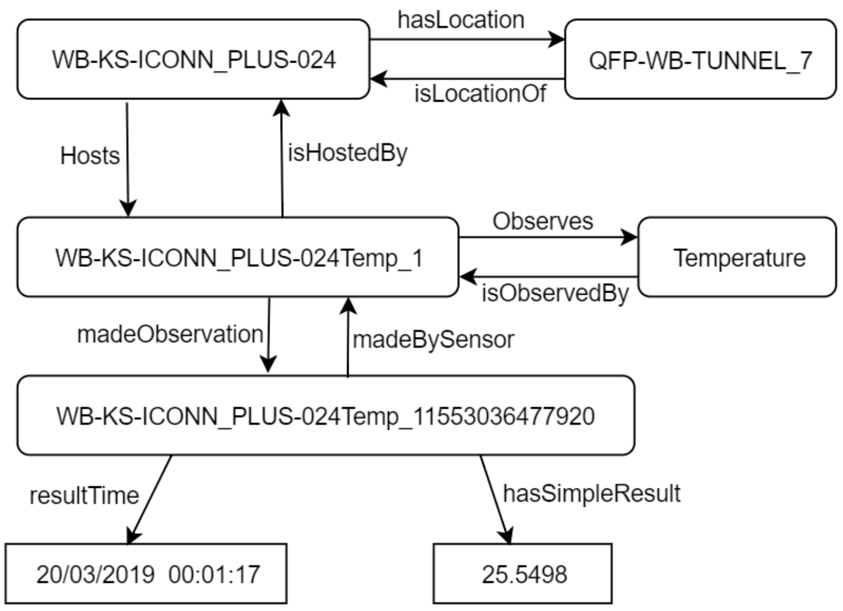


Figure 3.5: A representation of a sensor observation mapped onto the ontology.

3.2.2 | Knowledge Store

The knowledge storage platforms taken into consideration were an OBDA architecture, an LPG and an RDF store. The different architectures are discussed in Section 2.2.2.1. Given that the source data resides in several files and that new files are constantly being generated as explained in Section 3.1, an OBDA architecture was not deemed suitable as this would result too expensive to maintain with new mappings. On the other hand, for both an LPG and an RDF store, ETL-like scripts can be developed to extract and transform new source data to be pushed into the knowledge store via their APIs. The advantage of using an RDF store over an LPG is that in the former a standard language (SPARQL) is used for interactions, whereas LPGs make use of proprietary languages, such as Cypher in the case of Neo4j. The use of open standards would avoid vendor lock-in should there be the need to replace the platform. Moreover, RDF stores are able to infer new knowledge by exploiting the ontological logic, a feature that LPGs lack. After these considerations, we opted to use an RDF store.

We chose Ontotext GraphDB[8] for our knowledge store implementation. We considered various factors when choosing the right RDF store. The maturity of the product was taken into account as this influences its stability and robustness. We also wanted a platform that is scalable, as with the amount of data generated daily the system was bound to grow. The software portability was also considered. Ontotext GraphDB satisfied all these criteria. It has been around for a number of years and also offers a detailed documentation. It can operate on Linux, MS Windows and iOS and also includes a number of APIs to interface with. Moreover, enterprise packages are available that are scalable and also include vendor support.

Ontotext GraphDB Free version 9.1.0 was used for this dissertation. It was implemented on a computer running on Microsoft Windows 10 Pro.

3.2.3 | Data Lifting and Annotation

The Tbox of the EKG was uploaded directly into the knowledge store via the GraphDB user interface. In a similar approach to those we discussed in Section 2.2.2.1, the transfer of the data into the EKG was done via ETL scripts. We wrote scripts using Python (version 3.7.2) that read the data from its original sources, transformed it into triples and stored it into the RDF store. Different scripts were written for each of the three data sources. The scripts also catered for the mappings onto the ontology and to generate the

[8]`https://graphdb.ontotext.com/` (last accessed 10/10/2020)

appropriate URIs. The triples were then stored into GraphDB via the RDF4J[9] API using the RDFLib[10] (version 4.2.2) library in Python. The SPARQLwrapper[11] (version 1.8.5) library was also used to query the RDF store via its SPARQL endpoint.

3.2.4 | Knowledge Graph Validation

Since our ontology was developed to cater for PdM, and there was no ontology that is recognised as a gold standard for PdM, we considered the task-based method (Obrst et al., 2007; Raad and Cruz, 2015) for the evaluation of the ontology. We posed a number of competency questions in the form of SPARQL queries to determine whether the EKG was able to provide an answer. Some of the facts we queried were not explicitly written into the EKG by the ETL scripts and therefore required the knowledge store to infer such knowledge based on the rules of the ontology. For example, the ETL scripts specified that a wire bonder m "hosts" a sensor s and that sensor s "madeObservation" o at time t. Then we queried what observations were made at time t by a sensor that "isHostedBy" m and analysed the result.

Once again, to evaluate the EKG we did not have a gold standard to compare to. The EKG was validated using a retrospective evaluation approach described by Paulheim (2017). The results of the competency questions were manually inspected for correctness. A similar evaluation approach was also used by Golebiowska et al. (2001), Rychtyckyj et al. (2017) and Schabus and Scholz (2017). The accuracy of the results was measured as the ratio of correct triples retrieved (Paulheim, 2017).

3.3 | Producing the Predictive Models

Our goal was to produce an ML model that effectively predicts wire bonder faults. We opted for a classification approach to PdM, where the model output determines whether the wire bonder is running under normal conditions, or whether it is approaching a possible failure (within a predefined time window). We used a similar approach to the ones used in Soares (2015), Calabrese et al. (2020) and Gandhi et al. (2018), which we described in Section 2.3.2. In our approach, we conducted experiments by using different preprocessing methods to prepare the datasets and training different ML models on

[9]`https://graphdb.ontotext.com/documentation/free/using-graphdb-with-the-rdf4j-api.html#rdf4j-api` (last accessed 10/10/2020)
[10]`https://rdflib.readthedocs.io/en/4.2.2/` (last accessed 10/10/2020)
[11]`https://pypi.org/project/SPARQLWrapper/` (last accessed 10/10/2020)

these datasets in order to determine what worked best for our problem. These experiments are discussed in more detail in Section 3.3.3.

The *Historical Data Selection* part of the method, that consists of the acquisition and storage of the machine health indicators was explained in detail in Section 3.1 and Section 3.2.2 respectively. However, we must stress that for training the predictive models we did not make use of the EKG mentioned in Section 3.2 but rather we trained and validated the ML models using datasets in CSV format. The EKG was used at a later stage as described in Section 3.4. The preparation of the training datasets is explained hereunder.

In producing the predictive models, the following software and libraries were used:

- Python[12] version 3.7.6
- Jupyter Notebook[13] version 6.0.3
- Numpy[14] version 1.18.1
- Pandas[15] version 1.0.1
- Scikit-learn[16] version 0.22.1
- Imbalance-learn[17] version 0.7.0
- Matplotlib[18] version 3.1.3

3.3.1 | Data Preprocessing

The goal of this step was to produce several datasets that can potentially facilitate the learning process of the ML model (Soares, 2015). As discussed in Section 2.3.2.2, data preprocessing consists of four activities: data cleaning, data transformation, data reduction and data labelling.

[12]`https://www.python.org/downloads/release/python-376/` (last accessed 11/10/2020)
[13]`https://jupyter-notebook.readthedocs.io/en/stable/` (last accessed 11/10/2020)
[14]`https://pypi.org/project/numpy/1.18.1/` (last accessed 11/10/2020)
[15]`https://pandas.pydata.org/pandas-docs/version/1.0.1/index.html` (last accessed 11/10/2020)
[16]`https://scikit-learn.org/0.22/` (last accessed 11/10/2020)
[17]`https://pypi.org/project/imbalanced-learn/` (last accessed 11/10/2020)
[18]`https://matplotlib.org/3.1.3/index.html` (last accessed 11/10/2020)

a) Data cleaning: The timespan covered by the three datasets that were at our disposal varied. Thus to integrate the data into a dataset that can be used to train and validate the models, only the timespan common throughout all datasets was taken. This covered twenty-seven days of operations. The machines for which data was available also differed across the datasets. Likewise, we kept the data coming from wire bonders that were common across all datasets. The remaining data was considered out of scope and hence discarded. Constant dimensions were also removed since these gave no added information about the condition of the machine. Furthermore, undetermined sensor readings that were set by the wire bonders a default value were reset to a *null* value.

Next, we harmonised the format of the timestamp across all datasets so that these were comparable. We integrated the machine logs and IIoT datasets to have a complete time series of machine health indicators. As mentioned in Section 3.1, the sampling frequency for both datasets differed. Since the IIoT had a shorter sampling frequency, the records in the machine logs were merged with their closest reading in the IIoT dataset. This, however, resulted in several missing values for the machine logs. The gaps were imputed by the carry-forward method described in Soares (2015), where missing machine log records were filled-in with their last known values. This step also imputed *null* values set for undetermined sensor readings.

b) Data transformation: As discussed in Section 2.3.2.2, the aim of this activity was to transform the data into forms that facilitated the learning process. We experimented with different normalisation methods in order to find the one that best fits our problem. The methods we used were min-max normalisation, z-score normalisation and L2 normalisation. Moreover, we also used two methods that handle class imbalance, an over-sampling technique (SMOTE) and an under-sampling method (random under-sampling).

As stated in Section 3.1, the data contained outliers, which may or may not be erroneous values. To reduce the effect of these outliers, the data was averaged using a sliding window, a method similar to that used by Amihai et al. (2018). Different datasets were prepared with different window lengths.

c) Data reduction: In order to reduce the risk of overfitting and to minimise the model complexity, we resorted to data reduction. Once again, more than one technique was tried so that we could choose according to the results obtained. We used Pearson's correlation coefficient to identify highly correlated variables as discussed in Section 3.1. For variable pairs with Pearson's coefficient greater or equal to *0.9* ($\geqslant 0.9$), we discarded one of the variables since these added little to no new information to the learning algorithm. We also experimented with PCA, a method used in several studies we found in literature (see Section 2.3.2.2). We also used a univariate filter method that uses the ANOVA F-value between the variables and the class to choose the best k variables, where k is a parameter to the method. Additionally, some of the models we used, such as random forest, also have feature selection strategies embedded within their algorithm.

Since most of the training data consisted of raw FFT variables, one way to reduce the numerous dimensions is the method used by Seryasat et al. (2010), Farokhzad et al. (2012) and Amihai et al. (2018). This method uses mathematical functions to derive new features out of the FFT variables, which can then replace the FFT variables in the training data but still describe the waveform. The functions used include the RMS, mean, standard deviation, variance, kurtosis, skewness and crest factor. Using this method, we reduced the number of dimensions from 1,178 to 161.

d) Data labelling: We used the ETS event logs to identify machine faults. Some faults were highlighted by the domain experts in our discussions, but the logs contained several other messages. We tried to distinguish between errors and other messages, however, some logs were hard to interpret due to the technical nature of the descriptions. Whenever we were unable to determine whether a log entry was a fault, we assumed it to be so. Adhering to the method outlined by Susto et al. (2015) and Amihai et al. (2018), for each fault, we labelled the datasets based on the time the model should predict a fault in advance. For example, to train a model to predict machine faults 60 minutes in advance, for each error identified we labelled the samples for the respective wire bonder between 60 minutes ahead of the error and the time of the error as positive instances. For simplicity we refer to this "time ahead" as the *prediction window*. Similar to the method described in Amihai et al. (2018), we prepared several training datasets with varying prediction windows so that different classifiers can be trained to predict machine faults at different instances ahead of the fault. For example, two models can be trained, one warning of a possible fault two hours in advance and another model raising the alarm one hour in advance. In PdM, the aim is to detect faults as early as possible, possibly days (Amihai et al., 2018; Sipos et al., 2014) or weeks ahead (Calabrese et al., 2020), however, since our data showed that the wire bonders typically fail four to eight times a day, we resorted to prediction windows shorter than four hours. An empirical analysis of the data did not yield any noticeable different between the behaviours of the variables in times of normal operations and when approaching faults, so we experimented with prediction windows ranging between 60 and 180 minutes to determine the P-F interval of the wire bonders.

The domain experts identified some errors that are more expensive than others when they occur. We created training datasets such that only these errors are labelled as positives, with the aim that the learning algorithm is trained to predict these errors in particular. These errors were *non-stick on pad*, where the wire bonder fails to bond the wire thread on the pad of the die; *non-stick on lead*, where the wire bonder fails to bond with the lead frame; and *short tail*, where the second bond (or tail bond) is too weak and breaks when the capillary lifts off.

3.3.2 | Model Selection, Training and Validation

As discussed in Section 2.3.2, following the preparation of the training datasets during the data preprocessing stage, the ML models were trained and validated. Since we had data for four wire bonding machines, we used the data for three wire bonders for training the models and the data for the fourth machine to evaluate the models' performance. In this way, we could simulate the models' performance in production, where the predictions will be based on data that was unseen by the model during the learning process. The test set consisted of five days of data for a machine that was excluded from the training data. We also considered training ML models for each wire bonder, however, given the numerous wire bonders used within the manufacturing firm, this would result in a complex and expensive system to maintain. Hence the idea was shelved.

We chose different ML algorithms based on the literature we discussed in Section 2.3.2. Since we were approaching PdM as a classification problem, we opted for the following models:

- k-Nearest Neighbours (kNN) (Susto et al., 2015)
- Random Forest (RF) (Calabrese et al., 2020; Kanawaday and Sane, 2017; Paolanti et al., 2018)
- Naïve Bayes Classifier (NBC) (Kanawaday and Sane, 2017)
- Gradient Boosting Classifier (GBC) (Calabrese et al., 2020)
- Support Vector Machine (SVM) (Kanawaday and Sane, 2017; Sipos et al., 2014; Susto et al., 2015)

We followed the method used in Calabrese et al. (2020) and trained each of the chosen ML models using different datasets transformed using the various preprocessing methods discussed in Section 3.3.1. This method allowed for finding the preprocessing and model combination that gave the best results through a series of experiments. We also trained the models using datasets with different prediction windows. In our training, we used *cross-validation* to validate the models. We experimented with both *k-fold cross-validation* and *block cross-validation* as suggested by Bergmeir and Benítez (2012). When using cross-validation, the samples were not shuffled within the folds or blocks so as to preserve the temporal aspect of the data. We also employed a randomised hyperparameter search to determine the hyperparameters that gave the best results.

Finally, the models were evaluated on the test set that was set aside before the training process. The testing of the models on unseen data ensured that the models were not overfit. Following practices we found in literature (see Table 2.4) we measured the *accuracy* (Equation 2.5), *precision* (Equation 2.6), *recall* (Equation 2.7) and *f1-score* (Equation 2.8) to measure the performance of the models.

3.3.3 | Experimental Methodology

We conducted a number of experiments to determine the best arrangement for the predictive model to be used for PdM of wire bonders. These were planned in a way to test all the major activities involved in the ML approach to PdM. In each experiment, only one element was varied whilst keeping the other elements constant, so that the impact of the element under test can be observed. For example, when testing the data reduction methods, the data transformations and labelling used within the experiment were kept constant to reveal the impact of the reduction methods on the results. A matrix showing which experiments were aimed to investigate which activity of the ML process is presented in Table 3.4. The experiments were carried out using Jupyter Notebook on a Python environment. Notebooks were created for the various experiments, which used a randomized search cross-validation to look for the optimal hyperparameters in training and validating the models. The experiments were run using the datasets prepared through different preprocessing methods, as described in Section 3.3.1. After the cross-validation, the trained ML model was evaluated upon its performance on the test set. In each experiment, the *accuracy*, *precision*, *recall* and *f1-score* were measured. The results of both the cross-validation and the evaluation on the test set were recorded. Finally, the models scoring the best results on unseen data throughout all the experiments were considered for the PdM framework.

Activity	Experiments							
	1	2	3	4	5	6	7	8
Data Cleaning						✓		
Data Transformation			✓	✓				
Data Reduction					✓			
Data Labelling							✓	✓
Model Selection	✓							
Training & Validation		✓						

Table 3.4: The experiments aimed to investigate each activity within the ML process.

3.3.3.1 | Experiment 1 - Testing the different models

Aim - The purpose of this experiment was to determine which ML models were able to classify possible failures from normal machine operations.

Method - All the models were trained on the same dataset. Their performance was validated using a 5-fold cross-validation. Consequently, the models were evaluated on unseen data.

Models under test - kNN, RF, NBC, GBC and SVM.

Dataset/s - A dataset that was normalised using L2 normalisation and labelled with a 90-minute prediction window was used.

Following the poor performance obtained by the SVM in this experiment, the model was not included in other experiments. More details on this are provided in Section 4.2.1.

3.3.3.2 | Experiment 2 - K-fold cross-validation vs. Blocked cross-validation

Aim - This experiment served to determine which cross-validation method was best suited to facilitate the learning process in line with Bergmeir and Benítez (2012).

Method - Since this experiment aimed at testing the cross-validation methods and not the models themselves, not all models were put to test. Two ML models were trained and validated on the same dataset using two distinct cross-validation techniques: the k-fold cross-validation and the blocked cross-validation. Five folds where used in the k-fold method and five blocks were used in the blocked cross-validation. The trained models were also evaluated on unseen data. Since the experiment was aimed at testing the cross-validation method and not the predictive performance, some ML models were excluded from the experiment.

Models under test - RF and GBC.

Dataset/s - A dataset that was normalised using L2 normalisation and class-balanced by using SMOTE was used. The dataset was labelled with a 90-minute prediction window.

3.3.3.3 | Experiment 3 - Testing the different scaling methods

Aim - This experiment was scoped to demonstrate how the performance of the models was affected by the different scaling methods.

Method - The ML models were trained and validated on three datasets that were transformed using different scaling methods. The training and validation was done using a 5-fold cross-validation method. Each trained model was eventually evaluated on unseen data.
Models under test - kNN, RF, NBC and GBC.

Dataset/s - Three identical datasets that were scaled using different methods. The first using L2 normalisation, the second using min-max normalisation and the third using z-score normalisation, and all of which were labelled with a 90-minute prediction window were used in this experiment.

3.3.3.4 | Experiment 4 - Testing methods handling class imbalance

Aim - The purpose of this experiment was to find out whether techniques handling class imbalance improved the performance of the learning algorithms.

Method - The models were trained and validated using three sets of datasets, each set having a different prediction window. This was necessary because the sample distribution among the classes differed across the different prediction windows. A 5-fold cross-validation was used for training and validation, with the trained models later being evaluated on unseen data.

Models under test - kNN, RF, NBC and GBC.

Dataset/s - Three prediction windows were used, these being 60, 90 and 120 minutes. For each of the prediction windows, two datasets normalised using L2 normalisation were created and were treated for class imbalance, one using SMOTE and another using random under-sampling. Another dataset was created for each prediction window that was also normalised using L2 normalisation but were not treated for class imbalance. This resulted in a total of nine datasets.

3.3.3.5 | Experiment 5 - Testing data reduction methods

Aim - This experiment aimed to show the impact that different data reduction methods had on the predictive performance of the various models.

Method - Three data reduction methods were tested: PCA, a filter method based on the ANOVA F-value and the feature engineering method described in Farokhzad et al. (2012). Datasets were preprocessed using these reduction methods and then used for training and validation. Once again, a 5-fold cross-validation was used to train and validate the models, which were later evaluated on unseen data.

Models under test - kNN, RF, NBC and GBC.

Dataset/s - Three identical datasets scaled using L2 normalisation, treated for class imbalance using SMOTE and labelled using a 90-minute prediction window were used. Each of these datasets was reduced using one of the three data reduction methods being tested.

3.3.3.6 | Experiment 6 - Testing the performance on noise-smoothened data

Aim - The aim was to determine whether the performance improved by reducing the effect of outliers and which time window provides the best results as stated in Kinghorst et al. (2017).

Method - All the models were trained and validated on three datasets, two were treated for outliers using a rolling average while the other was not. A 5-fold cross-validation was used for training and validation, with the trained models later evaluated on unseen data.

Models under test - kNN, RF, NBC and GBC.

Dataset/s - Two datasets scaled using L2 normalisation were resampled using a simple moving average with a rolling time window of 15 and 30 minutes respectively. The datasets were labelled using a 90 minute prediction window and treated for class imbalance using SMOTE. The features were reduced using a filter method based on the ANOVA F-value.

3.3.3.7 | Experiment 7 - Testing different prediction windows

Aim - This experiment was scoped to determine the relationship between the prediction window and the performance of the models.

Method - All the models were trained on datasets labelled using different prediction windows. Their performance was validated using a 5-fold cross-validation. Consequently, the models were evaluated on unseen data.

Models under test - kNN, RF, NBC and GBC.

Dataset/s - Datasets normalised using L2 normalisation but labelled using different prediction windows were used. The prediction windows were 60, 90, 120, 150 and 180 minutes. All the datasets were balanced using SMOTE and smoothened using a simple moving average with a rolling time window of 30 minutes. The features were reduced using a filter method based on the ANOVA F-value.

3.3.3.8 | Experiment 8 - Forecasting selected faults

Aim - In this experiment we tested the models' ability to learn to identify that an upcoming fault falls within a set of preselected expensive faults.

Method - The models were trained on a dataset that was labelled on selected faults only. A 5-fold cross-validation was used to train and validate the models on each dataset. The trained models were then evaluated on unseen data.

Models under test - kNN, RF, NBC and GBC. The SVM was excluded from the experiment due to its poor performance in previous tests.

Dataset/s - A dataset was prepared that was labelled only on expensive faults with a 90-minute prediction window.It was scaled using L2 normalisation and smoothened using a simple moving average with a rolling time window of 30 minutes. SMOTE was used to deal with class imbalance and the features were reduced using a filter method based on the ANOVA F-value.

3.4 | The Predictive Maintenance Framework

The PdM framework brought together the various components that collectively provide the necessary information to the user to take an informed decision. As discussed in Section 2.1.4, such a decision is not taken lightly as stopping the machine for maintenance is costly. Thus, providing relevant and timely information is important. The various components that made up our PdM framework are shown in Figure 3.6. For simplicity, the diagram features a single wire bonder, but the system was meant to gather data from multiple machines concurrently and harvest the data into the EKG for its consumption by the dashboard.

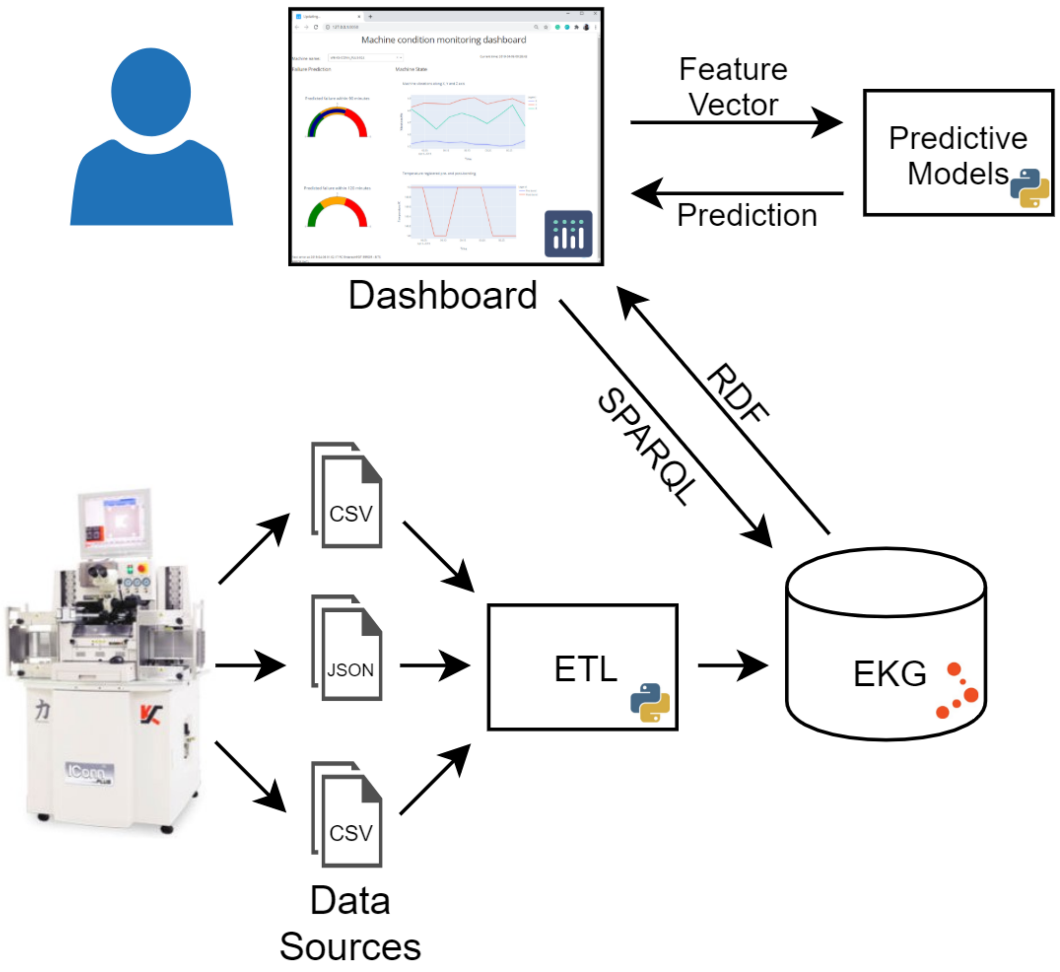


Figure 3.6: A schematic view of the PdM framework produced through this research project.

3.4.1 | Extract Transform and Load Scripts

As discussed in Section 3.1, the wire bonding machines generate three sets of data: the *ETS event logs* in CSV format, the *machine logs* in JSON format and the *IIoT data*, which are also in CSV format. We developed ETL scripts that read the data from these files, transform the data by mapping it onto the ontology explained in Section 3.2.1 and storing it in the EKG. The scripts were written using Python (version 3.7.6). The main libraries that were used are Pandas (version 1.0.1), RDFLib (version 4.2.2) and SPARQLwrapper (version 1.8.5).

3.4.2 | The Enterprise Knowledge Graph

The EKG was implemented on Ontotext GraphDB Free (version 9.1.0) and the process to design, build and populate it is explained in Section 3.2. It consisted of two parts, the *Tbox* that formally defined its schema and the *Abox*, that represents the actual entity instances. Our EKG stored information about the wire bonders, their sensors and sensor observations, together with the timeline of states that the machines went through during their operations. The IIoT data stored within the EKG was then used by the PdM framework to predict faults of the wire bonders. The properties of the EKG were exploited to harmonise the heterogeneous data into a homogeneous structure using a common vocabulary, thus simplifying the retrieval of the indicators that were required as features for the predictive models. Without the EKG, the PdM framework would have to cater for the complexities brought by the diversity of the data structures and vocabularies found within the different data files generated by the wire bonders. The EKG was evaluated using the KG validation methods described in Section 3.2.4, which gave assurance that the data contained within it was correct and complete, and therefore that the fault predictions were based on the intended data.

3.4.3 | The Predictive Models

The predictive models forecasted wire bonder faults based upon a vector of IIoT features. The models were binary classifiers, therefore the prediction indicated whether or not a wire bonder would give an error within a specific time window. Two predictive models were used, one forecasting a fault within the next 120 minutes and another with a prediction window of 90 minutes. The approach of having multiple prediction windows was derived from Amihai et al. (2018).

The models were chosen based on the results of the experiments described in Section 3.3.3. They were trained using Python (version 3.7.6), and the chosen models were then packaged for reuse using the Pickle[19] (version 4.0) library.

3.4.4 | The Dashboard

The ultimate aim of the PdM framework was to support the decision making process with respect to maintenance of the wire bonders. The dashboard was the component closest to the user and was intended to provide the necessary information for the user to take an informed decision. We implemented our dashboard using the Plotly Dash[20]

[19]`https://docs.python.org/3.7/library/pickle.html` (last accessed 15/10/2020)
[20]`https://plotly.com/dash/` (last accessed 15/10/2020)

(version 1.8.0) framework on top of Python (version 3.7.6) to project a number of key machine health indicators in the form of charts. As shown in Figure 3.6, the dashboard requests the latest sensor observations from the EKG via a SPARQL query. The EKG in turn responds with the result of the query as an RDF message. Consequently, the dashboard interprets the RDF message and updates the charts with the latest sensor observations. Moreover, these observations are transformed as a feature vector, which is passed on to the predictive models. The dashboard then receives the output of the predictive models and plots them in the form of a gauge chart.

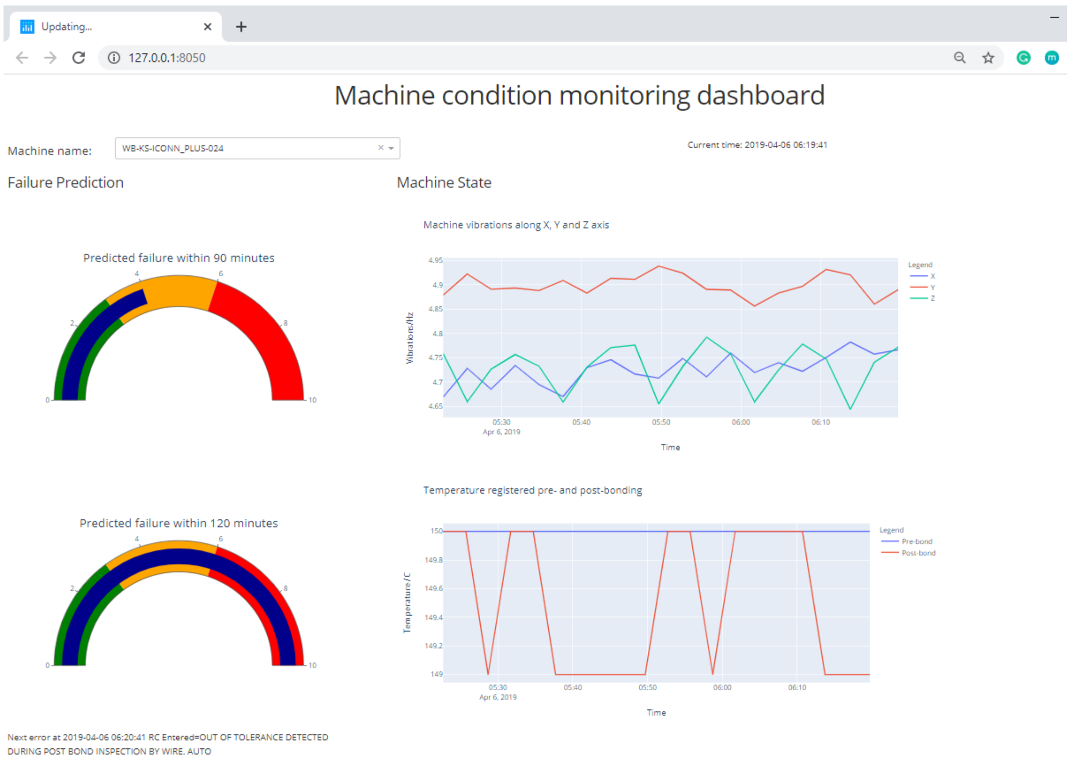
A snapshot of the resultant dashboard is shown in Figure 3.7. For a given wire bonder, the dashboard projected in the form of charts a series of vibrations observations, and the pre-bonding and post-bonding temperatures, both for the past hour. These indicators originate from the IIoT data and machine logs respectively and were selected based on the literature discussed in Section 2.3.1, where vibrations and temperature are most frequently considered as an indication of the machine health. Moreover, the dashboard featured two gauge charts that measured the number of positive predictions returned by the models in the last 30 minutes. The top gauge had a prediction window of 90 minutes while the bottom one had a prediction window of 120 minutes.

The dashboard was designed to focus on one wire bonder at a time. The machine under review is chosen from a menu featured on the dashboard.

As the PdM framework could not be tested on the actual factory floor obtaining readings directly from the machines, a simulation was used for testing. We took the data for twelve hours out of the datasets and created a dummy process that updated the EKG every three minutes. Simultaneously, the dashboard requested the latest data available from the EKG every three minutes and updated the charts with the latest information.

3.5 | Summary

In this chapter we presented our approach to produce a PdM framework, which was based on the literature discussed in Chapter 2. Our goal was to determine whether an EKG was a suitable paradigm to store IIoT data and if it could be exploited for PdM. To do so, we obtained actual industrial machine data, which we described in the first part of this chapter. Consequently, in Section 3.2, we discussed how we designed and implemented an EKG, that we later populated with data for wire bonders through the use of ETL scripts.


Updating...
127.0.0.1:8050
Machine condition monitoring dashboard
Machine name:
WB-KS-ICONN_PLUS-024
Current time: 2019-04-06 06:19:41
Failure Prediction
Machine State
Predicted failure within 90 minutes
Predicted failure within 120 minutes
0
2
4
6
8
10
Machine vibrations along X, Y and Z axis
Vibrations/Hz
Time
Legend
X
Y
Z
05:30
Apr 6, 2019
05:40
05:50
06:00
06:10
Temperature registered pre- and post-bonding
Temperature/C
Legend
Pre-bond
Post-bond

Next error at 2019-04-06 06:20:41 RC Entered=OUT OF TOLERANCE DETECTED DURING POST BOND INSPECTION BY WIRE. AUTO

Figure 3.7: The dashboard interface that projects the latest vibrations and temperature observations together with the number of positives returned by the predictive models.

In Section 3.3, we described how we trained and evaluated different ML models and the experiments we carried out to determine the configuration that gave us the best results. Finally, in Section 3.4, we explained how all the components were integrated together in the form of a PdM framework. The framework also features a dashboard that, in addition to the prediction results, projected real-time IIoT data to its users.

4 Evaluation

Following the methodology discussed in Chapter 3, in this chapter we discuss at length the results obtained and how these contribute to the achievement of our objectives. The objectives of this dissertation included the designing of an ontology to model wire bonding machines, their sensors and the observations of these sensors. This ontology could then act as the backbone of the EKG, which stored the data produced by the wire bonders. We review our EKG implementation in line with these objectives and discuss the results obtained. We also present and discuss the results of the experiments carried out on the predictive models in line with our objective to forecast faults in a particular industrial machine. Lastly, we discuss the outcome of the implementation of the PdM framework, where the EKG and the predictive models were integrated with other components to form a software ecosystem that furnishes its users with the necessary information to take informed decisions with regards to machine maintenance. This was also an objective of this dissertation.

4.1 | Evaluation of the Enterprise Knowledge Graph

The method we adopted to implement the EKG is described in Section 3.2. We first developed an ontology, then formed the knowledge store and created ETL scripts that transfer the machine data into the EKG. Finally, we validated both the ontology and the EKG.

4.1.1 | The Ontology

We designed and developed our ontology using a method based on Noy et al. (2001), as described in Section 3.2.1. The availability of existing ontologies in the sensor network domain facilitated our work. We considered the SOSA ontology (Janowicz et al., 2019) as the most suitable candidate for our needs and took on a number of classes from this ontology. We extended it by adding classes from the DUL ontology as well as classes of our own.

Since there was no ontology that could be considered as a gold standard in the PdM domain, we considered the task-based method (Obrst et al., 2007; Raad and Cruz, 2015) to evaluate our ontology. This method aims to quantify the extent to which an ontology improves a given task. To measure the contribution of our ontology, we loaded data into the knowledge store without uploading the ontology and then posed a number of competency questions in the form of SPARQL queries. The competency questions we used were the ones identified while designing the ontology (see Section 3.2.1). We took note of the results and repeated the process after uploading the ontology. Once again we took note of the results. The competency questions were designed in a way to cover all major classes of the ontology and by keeping in mind the context of PdM. There were other valid questions that we intended to include but could not due to the lack of data. For example, questions like *"When was maintenance last applied to wire bonder x?"* and *"How long after the last maintenance did wire bonder y fail?"* would have potentially contributed new insights that could be used for PdM, however, these were not included as we did not have access to maintenance records.

The results of this exercise are shown in Table 4.1, that lists the competency questions posed to the knowledge store and an indication of whether any results were returned. For three of the questions, namely 1, 4 and 6, the EKG could only provide a reply once we applied the ontology. Therefore, we deduced that the ontology was playing a role by defining logic upon which the EKG could infer new knowledge to answer these questions. The replies of these three questions were taken as the sample upon which we evaluated the ontology. To determine the accuracy of the ontology, we manually analysed the resulting triples for correctness by comparing them against the original datasets. It resulted that all three questions were correctly answered, therefore the accuracy of the ontology was calculated to be *1.0*.

Based on this result, we concluded that the ontology we developed was suited for its intended purpose. The ontology allowed the inference engine of the knowledge store to add new triples through the rules and logic defined in the said ontology. For example, when recording an observation o, the ETL script generated triples defining the observation as well as a statement that o was "madeBySensor" s. By applying the logic within the ontology, the knowledge store inferred that sensor s "madeObservation" o. Such inferencing simplified the ETL scripts and made them more efficient as not all

Competency Question	**Without Ontology**	**With Ontology**
1. In which state was wire bonder x at time t?	✗	✓
2. What is the latest temperature observation of wire bonder y?	✓	✓
3. Which sensors measure vibrations?	✓	✓
4. Which wire bonders have a sensor of type a?	✗	✓
5. What are the latest sensor observations of wire bonder z?	✓	✓
6. What was the change in temperature for wire bonder x between time t and time t+n?	✗	✓

Table 4.1: The results of the competency questions (✓ - question answered; ✗ - no reply).

facts needed to be explicitly specified. Moreover, the completeness of the EKG was maintained as shown in Figure 4.1 that depicts the knowledge explicitly specified by the scripts (solid arrows) and that inferred through the ontology (dotted arrows).

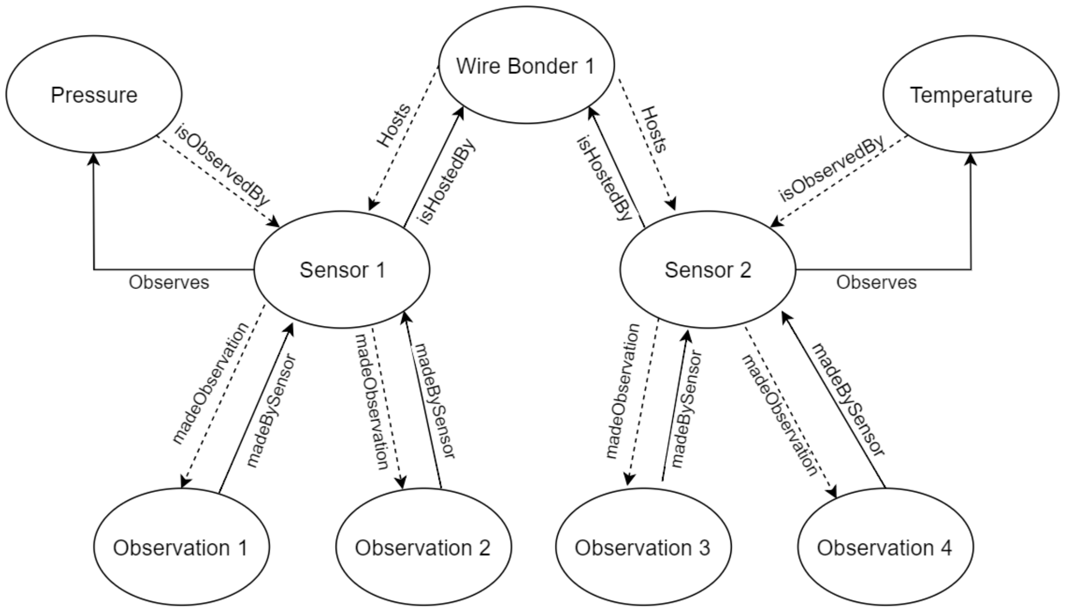


Figure 4.1: A partial representation of the knowledge that was explicitly stated by the ETL scripts (solid arrows) and that inferred by the knowledge store (dotted arrows).

4.1.2 | The Enterprise Knowledge Graph

As discussed in Section 2.2.4, KGs can be evaluated against a gold standard, where the generated KG (or parts of it) is compared to a manually created equivalent for correctness and completeness (Paulheim, 2017). Paulheim (2017) states that when a gold standard is not available, retrospective evaluation may be used. In this method, samples of the KG are taken and manually checked for correctness. The evaluation metric used is normally the accuracy and is calculated as the ratio of correct triples retrieved to the total triples. Since no gold standard was at our disposal, we evaluated our EKG using to the retrospective approach described by Paulheim (2017).

We used the results of the six SPARQL queries for the competency questions listed in Table 4.1 as the samples upon which we evaluated our EKG. These results were taken after the ontology was applied and therefore the knowledge could be completed by the inferencing engine. We considered these samples as representative of the EKG since they covered all the major classes involved. We manually analysed the results for correctness and completeness by comparing them against the original data. The results for all six queries were found to be correct and complete with an accuracy of *1.0*.

The results of the evaluation were positive and showed that the EKG was suitable to integrate the various heterogeneous data sources into a homogeneous data structure. This is consistent with the findings discussed in Section 2.2.2.2. Furthermore, in line with other research studies that were discussed in Section 2.2.3, the results also indicate that the EKG was able to store IIoT data.

4.2 | The Predictive Models

In Section 3.3, we explained the method we used to produce the predictive models. This method was based on the approaches of Soares (2015), Calabrese et al. (2020) and Gandhi et al. (2018), and involved experimenting with different ML models to find the one that gave the best results. We split the data into a training dataset and a test set. Since we had data for four wire bonders, we took the data for a single machine as the test set and the remaining data for training and validation. We used different preprocessing techniques to transform the training data into various datasets that were then used in our experiments. Cross-validation was used to train and validate different ML models using the various preprocessed datasets. The resulting accuracy, precision, recall and F1-score were measured and recorded. The trained models were then evaluated on the test set, where once again the evaluation metrics were measured and recorded. Our evaluation method was based on the methods used in Calabrese et al. (2020), Paolanti et al. (2018) and Susto et al. (2015).

In this part of the document we present the results of the experiments that we conducted to find the optimum set up for our predictive models. An explanation of the set up of each experiment can be found in Section 3.3.3.

4.2.1 | Experiment 1 - Testing the Different Models

The aim of this test was to determine which ML models were able to classify a set of IIoT observations as an indication of normal machine operations or an indication of a possible upcoming fault. Several ML models were trained, validated and tested. The outcome of the experiment is shown in Table 4.2, which features the results of a 5-fold cross-validation and the testing on unseen data.

	Model	Accuracy	Precision	Recall	F1-score
Validation	KNN	**0.583**	**0.571**	0.707	0.632
	RF	0.521	0.519	0.725	0.605
	NBC	0.506	0.509	0.685	0.584
	GBC	0.501	0.505	0.724	0.595
	SVM	0.507	0.507	**1.000**	**0.672**
Testing	KNN	**0.613**	0.615	0.678	0.645
	RF	0.488	0.505	0.611	0.553
	NBC	0.526	0.527	0.839	0.647
	GBC	0.573	**0.672**	0.345	0.456
	SVM	0.519	0.519	**1.000**	**0.683**

Table 4.2: The results measured in Experiment 1.

The results show that all the models could classify the samples to some extent, except for SVM. Although SVM had a recall score of *1.0* and an F1-score of *0.683*, in reality all instances were classified as positives, thus the model could not distinguish between the two classes. Although the SVM model obtained positive results in Kanawaday and Sane (2017), Sipos et al. (2014) and Susto et al. (2015), it did not fit our problem and following this result the model was omitted from further tests. The NBC and kNN performed best with an F1-score of *0.647* and *0.645* respectively. The kNN, however, had a better precision and a better accuracy. A plot of the F1-score measured for each model is featured in Figure 4.2.

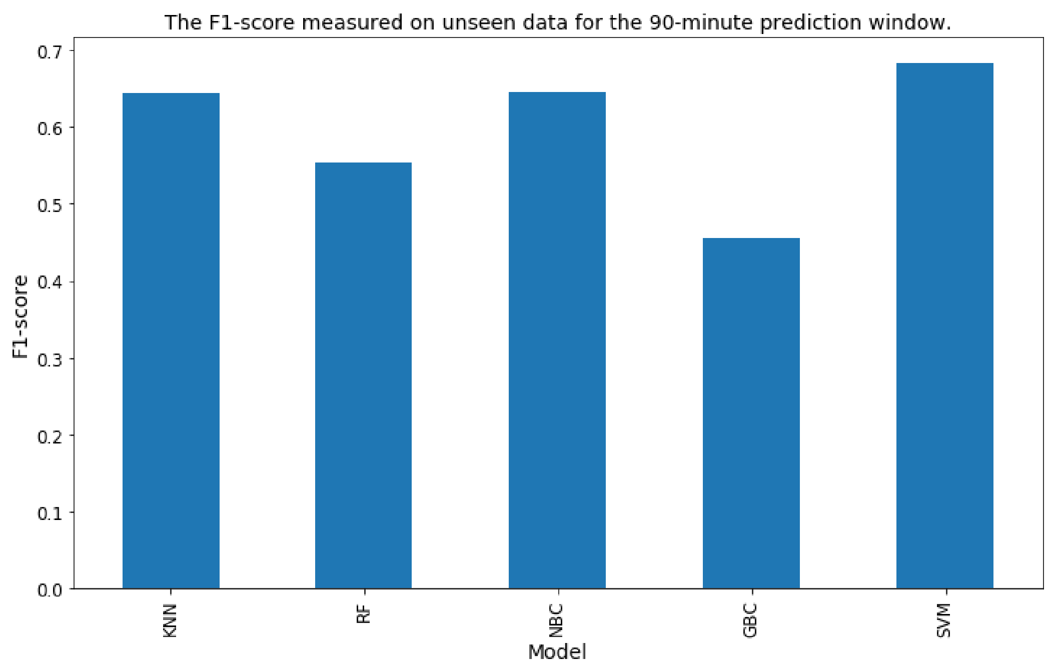


Figure 4.2: The F1-score measured on unseen data for the 90-minute prediction window in Experiment 1.

We observed that the accuracy achieved by the models ranges between *0.5* and *0.6*, which is close to an accuracy obtained when predicting by pure chance. Moreover, in the majority of cases, the models exhibited a recall in the region of *0.7* but a precision towards *0.5*, which indicates a high rate of false positives. We also observed that in some cases the models performed better on unseen data than on the validation. This can be attributed to the different class imbalance found in the validation and testing sets, which favoured certain models. Such bias could have been reduced by using preprocessing methods to handle class imbalance.

4.2.2 | Experiment 2 - K-Fold Cross-Validation vs. Blocked Cross-Validation

In k-fold cross-validation, the training set is split into *k* parts and each training-validation iteration uses a different part of the data as the validation set. When the training dataset consists of a time series, this shuffling of segments of the dataset creates a situation where the learning algorithm is trained on data that is in the future with respect the the validation set. In Section 2.3.2.3, we refer to the statement made by Bergmeir and

Benítez (2012) about this issue. The authors state that blocked cross-validation can be used when dealing with time series data, since the this method preserves the temporal aspect of the data. They also state that when they contrasted the results of k-fold and blocked cross-validation they noticed no significant change in performance between the two methods.

The goal of this experiment was to test which cross-validation method gives the best model performance. Since the experiment examined the cross-validation method and not the predictive ability of the models, we only trained two ML models and validated them on the same training data using both k-fold cross-validation and blocked cross-validation. The experiment results for the 5-fold cross-validation are presented in Table 4.3, while the corresponding results for blocked cross-validation (using five blocks) are shown in Table 4.4.

	Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Validation	RF	**0.546**	**0.541**	0.686	**0.605**
	GBC	0.505	0.508	**0.724**	0.597
Testing	RF	0.477	0.496	**0.529**	0.512
	GBC	**0.591**	**0.628**	0.518	**0.568**

Table 4.3: The results of Experiment 2 for K-fold cross-validation.

	Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Validation	RF	0.495	0.501	0.639	0.561
	GBC	**0.505**	**0.508**	**0.724**	**0.597**
Testing	RF	0.470	0.490	0.511	0.500
	GBC	**0.591**	**0.628**	**0.518**	**0.568**

Table 4.4: The results of Experiment 2 for Block cross-validation.

The results show no difference in the performance of the GBC, while RF performed marginally better when trained using k-fold cross-validation but the difference is negligible. This concurs with the findings of Bergmeir and Benítez (2012). The results potentially indicate that although in k-fold cross-validation the folds were shifted hence breaking the temporal order of the data, since the order within the folds was preserved (by not shuffling the data), each fold can be seen as a shorter time series in itself. Thus the models were still able to learn when using this method.

4.2.3 | Experiment 3 - Testing the Different Scaling Methods

The aim of this experiment was to demonstrate the impact of the various scaling methods on the ML models' performance. We trained the ML models on datasets transformed using L2 normalisation, min-max normalisation and z-score normalisation and observe the model performance. The outcome from the experiment are presented in Table 4.5 that shows the results of a 5-fold cross-validation and those measured when testing the models on unseen data.

	Model	Scaling	Accuracy	Precision	Recall	F1-score
Validation	KNN	L2	0.583	0.571	0.707	0.632
		MinMax	**0.586**	**0.604**	0.528	0.564
		Z-score	0.518	0.526	0.483	0.503
	RF	L2	0.521	0.519	0.725	0.605
		MinMax	0.485	0.494	0.705	0.581
		Z-score	0.478	0.489	0.673	0.567
	NBC	L2	0.506	0.509	0.685	0.584
		MinMax	0.528	0.521	**0.855**	**0.648**
		Z-score	0.481	0.486	0.417	0.449
	GBC	L2	0.501	0.505	0.724	0.595
		MinMax	0.525	0.525	0.660	0.585
		Z-score	0.525	0.525	0.660	0.585
Testing	KNN	L2	**0.613**	0.615	0.678	0.645
		MinMax	0.455	0.481	0.633	0.546
		Z-score	0.501	0.510	0.937	0.661
	RF	L2	0.488	0.505	0.611	0.553
		MinMax	0.514	0.517	0.983	0.677
		Z-score	0.502	0.511	0.915	0.656
	NBC	L2	0.526	0.527	0.839	0.647
		MinMax	0.519	0.519	**1.000**	**0.683**
		Z-score	0.472	0.491	0.480	0.485
	GBC	L2	0.573	**0.672**	0.345	0.456
		MinMax	0.518	0.518	0.995	0.682
		Z-score	0.518	0.518	0.995	0.682

Table 4.5: The results obtained from Experiment 3.

We observed that the L2 normalisation method resulted in a better precision for all models. Moreover, MinMax and z-score normalisation resulted in RF and GBC classifying most examples towards the positive class. This is shown by the high recall score and the relatively low precision. The same applies for kNN using z-score and NBC using MinMax. In these cases the data scaling hindered learning.

4.2.4 | Experiment 4 - Testing Methods Handling Class Imbalance

This test aimed to measure the effect methods handling class imbalance have on the performance of the learning algorithms. We used three datasets, labelled using a different prediction windows (60, 90 and 120 minutes) since the class imbalance varied across the different datasets. For example, the positively labelled samples for the 120-minute prediction window were twice as much those for a 60-minute prediction window. Some of the results of the experiment are displayed in Table 4.6 and Table 4.7, that show the precision and F1-score as measured on unseen data respectively.

The results show that the impact of the two methods handling class imbalance on the model performance vary according to the model itself. This is also evident from Figure 4.3 that shows a plot of the F1-score recorded for the various models for a 90-minute prediction window. The largest impact was observed on the GBC, where SMOTE improved its performance for both the 90-minute and the 120-minute prediction windows, while under sampling decreases the model performance. The other models were either unaffected of their performance was deteriorated when using the methods tested. However, it should be noted that since the datasets were imbalanced towards the negative class, when no class imbalance handling methods were applied there was a bias towards models predicting a majority of negatives. Potentially, this may have distorted the results.

Model	60-minute			90-minute			120-minute		
	No Imbalance Handling	**SMOTE**	**Under Sampling**	**No Imbalance Handling**	**SMOTE**	**Under Sampling**	**No Imbalance Handling**	**SMOTE**	**Under Sampling**
KNN	0.400	0.476	0.473	0.615	0.614	0.606	0.545	0.705	0.697
RF	0.421	0.412	0.457	0.505	0.489	0.512	0.498	0.646	0.559
NBC	0.432	0.495	0.534	0.527	0.532	0.532	0.620	0.623	0.624
GBC	0.373	0.529	**0.765**	0.672	0.628	**0.693**	0.460	0.726	**0.765**

Table 4.6: The Precision measured for the various models as measured on unseen data in Experiment 4.

Model	60-minute			90-minute			120-minute		
	No Imbalance Handling	**SMOTE**	**Under Sampling**	**No Imbalance Handling**	**SMOTE**	**Under Sampling**	**No Imbalance Handling**	**SMOTE**	**Under Sampling**
KNN	0.543	0.529	0.546	0.645	0.644	0.627	0.692	0.683	0.667
RF	0.528	0.553	0.499	0.553	0.498	0.513	0.606	0.503	0.528
NBC	0.593	0.589	**0.613**	**0.647**	0.619	0.620	**0.748**	0.678	0.678
GBC	0.469	0.186	0.117	0.456	0.568	0.301	0.585	0.600	0.293

Table 4.7: The resulting F1-score for the various models as measured on unseen data in Experiment 4.

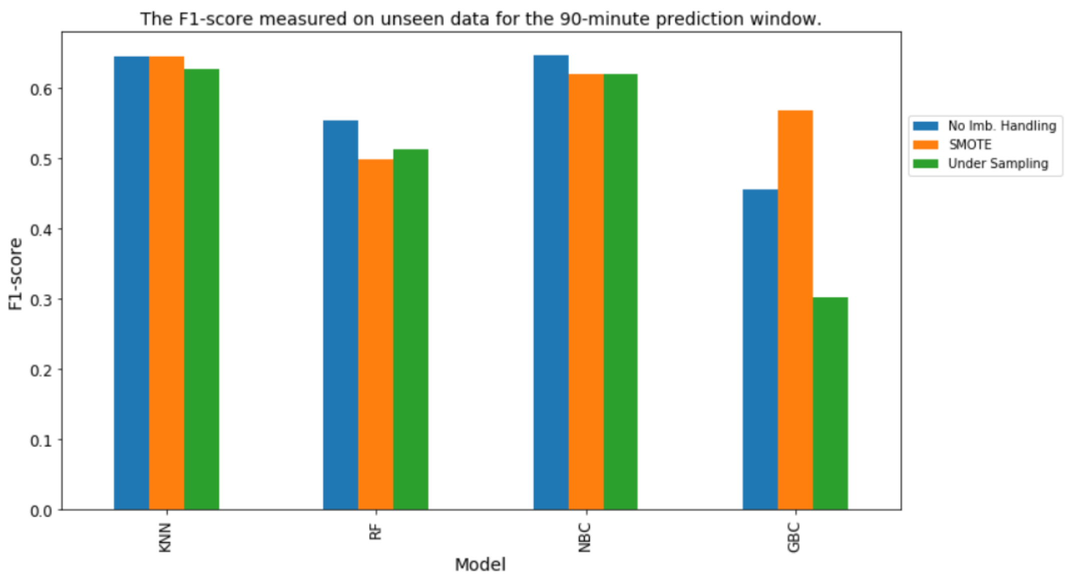


Figure 4.3: The F1-score measured on unseen data for the 90-minute prediction window in Experiment 4.

4.2.5 | Experiment 5 - Testing Data Reduction Methods

We conducted this experiment to determine whether data reduction methods improved the learning ability of the ML models and which method gave the best outcome. Three data reduction methods were put to test - PCA, a filter method and the feature engineering method used by Farokhzad et al. (2012), where the numerous FFT dimensions were replaced by other features that described the properties of the waveform. The ML models were trained and validated on datasets that are reduced using the three reduction methods and their performance was measured. The trained models were then tested on unseen data. The experiment results are shown in Table 4.8.

When we compared the obtained results against those obtained in previous experiments, we observed that the reduction methods gave a similar performance to that seen in Experiment 1 for kNN and NBC. The filter method, however, improved the F1-score for GBC from *0.456* as measured in Experiment 1 to *0.597*. RF suffered a drop in performance that may be attributed to the embedded feature selection technique within the model's algorithm. Moreover, the method used in Farokhzad et al. (2012) gave results for kNN and NBC that are comparable to those measured in Experiment 1, but with the advantage of having less dimensions.

	Model	Reduction	Accuracy	Precision	Recall	F1-score
Validation	KNN	No reduction	0.584	0.572	0.705	0.632
		PCA	0.574	0.559	0.752	0.642
		Filter Method	0.587	0.578	0.683	0.626
		Farokhzad et al. (2012)	0.590	0.573	0.742	**0.647**
	RF	No reduction	0.540	0.536	0.690	0.603
		PCA	0.490	0.498	**0.920**	**0.647**
		Filter Method	0.531	0.521	0.917	0.664
		Farokhzad et al. (2012)	0.532	0.526	0.765	0.624
	NBC	No reduction	0.500	0.505	0.631	0.561
		PCA	0.454	0.437	0.271	0.334
		Filter Method	**0.598**	**0.616**	0.544	0.578
		Farokhzad et al. (2012)	0.506	0.509	0.685	0.584
	GBC	No reduction	0.505	0.508	0.724	0.597
		PCA	0.570	0.557	0.736	0.634
		Filter Method	0.571	0.579	0.561	0.570
		Farokhzad et al. (2012)	0.426	0.429	0.401	0.415
Testing	KNN	No reduction	0.612	0.614	0.676	0.644
		PCA	0.611	0.611	0.685	0.646
		Filter Method	0.483	0.502	0.591	0.543
		Farokhzad et al. (2012)	0.608	0.612	0.671	0.640
	RF	No reduction	0.469	0.489	0.508	0.498
		PCA	0.510	0.515	0.970	0.673
		Filter Method	0.519	0.519	**1.000**	**0.683**
		Farokhzad et al. (2012)	0.613	**0.709**	0.430	0.535
	NBC	No reduction	0.527	0.532	0.741	0.619
		PCA	0.544	0.662	0.248	0.361
		Filter Method	**0.652**	0.691	0.595	0.639
		Farokhzad et al. (2012)	0.526	0.527	0.839	0.647
	GBC	No reduction	0.591	0.628	0.518	0.568
		PCA	0.514	0.517	0.990	0.679
		Filter Method	0.621	0.665	0.542	0.597
		Farokhzad et al. (2012)	0.519	0.556	0.362	0.439

Table 4.8: The results for Experiment 5.

4.2.6 | Experiment 6 - Testing the Performance on Noise-Smoothened Data

The IIoT data was noisy and contained outliers. In this experiment we smoothened the data using a simple moving average to lessen the effect of outliers. The aim of the experiment was to measure the impact of such smoothening on the performance of the ML models and to determine which rolling time window length produced the best results. We prepared two datasets using a simple moving average with a rolling time window of 15 and 30 minutes respectively. The results measured through a 5-fold cross-validation and eventually through the testing of the models on unseen data are shown in Table 4.9.

	Model	Time Window	Accuracy	Precision	Recall	F1-score
Validation	KNN	15 minutes	0.565	0.560	0.667	0.609
		30 minutes	0.533	0.528	**0.740**	0.616
	RF	15 minutes	0.560	0.552	0.694	0.615
		30 minutes	0.555	0.548	0.692	0.612
	NBC	15 minutes	0.491	0.498	0.599	0.544
		30 minutes	0.604	0.621	0.558	0.588
	GBC	15 minutes	**0.618**	**0.623**	0.622	0.623
		30 minutes	0.604	0.601	0.649	**0.624**
Testing	KNN	15 minutes	0.598	0.617	0.593	0.605
		30 minutes	0.593	0.607	0.611	0.609
	RF	15 minutes	0.641	0.623	**0.782**	0.693
		30 minutes	0.483	0.501	0.622	0.555
	NBC	15 minutes	0.533	0.546	0.582	0.563
		30 minutes	0.649	0.703	0.558	0.622
	GBC	15 minutes	0.634	**0.760**	0.431	0.550
		30 minutes	**0.672**	0.658	0.765	**0.707**

Table 4.9: The results for Experiment 6.

The results showed that while kNN and NBC did not benefit from data smoothening, RF and GBC displayed an improved performance. In the case of RF, the F1-score increased from *0.553* to *0.693* on data treated with a simple moving average using a rolling time window of 15 minutes. On the other hand, GBC displayed an increase in F1-score from *0.456* to *0.55* using 15-minute time window, and to *0.707* using a 30-minute time window. A comparison between the F1-score obtained without the data smoothening and those obtained after transforming the data using the simple moving average are projected in Figure 4.4, where the improvements for RF and GBC can be observed.

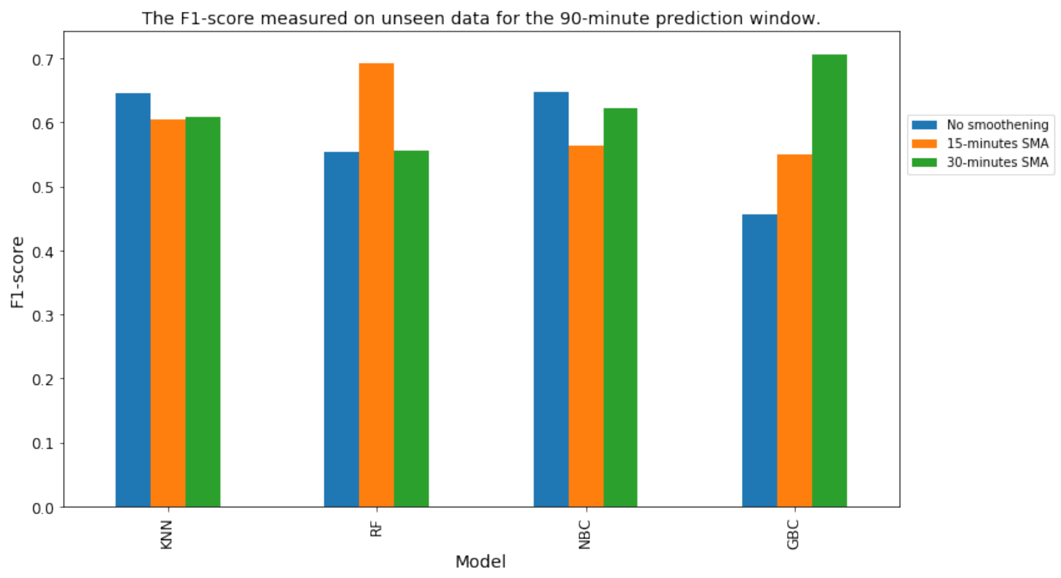


Figure 4.4: The F1-score measured on unseen data for the 90-minute prediction window in Experiment 6.

4.2.7 | Experiment 7 - Testing Different Prediction Windows

We also studied how the models performed across different prediction windows. In this experiment we prepared five datasets labelled using different prediction windows. All the datasets were scaled using L2 normalisation, class-balanced using SMOTE and smoothened using a simple moving average with a rolling time window of 30 minutes. Feature selection was also applied.

The F1-scores measured for each ML model in the experiment are shown in Table 4.10, while Figure 4.5 shows a plot of the F1-score obtained by the models on unseen data over the different prediction windows. We observed that longer prediction windows resulted in a better model performance. The results of kNN increased gradually and stabilised at *0.72* for the 120 and 150 prediction windows. RF performed better for prediction windows of 120 minutes and above while NBC produced an F1-score of *0.782* for the 150-minute prediction window, which was remarkably higher than the scores for the other prediction windows. The F1-score measurements for the GBC were stable in the region of *0.7* for prediction windows of 90 minutes and above.

	Model	**Prediction Window (minutes)**				
		60	**90**	**120**	**150**	**180**
Validation	KNN	0.637	0.616	0.637	0.685	0.637
	RF	**0.710**	0.612	**0.716**	**0.748**	**0.709**
	NBC	0.598	0.588	0.598	0.734	0.598
	GBC	0.698	**0.624**	0.698	0.676	0.698
Testing	KNN	0.503	0.609	0.650	0.721	0.719
	RF	**0.594**	0.555	**0.750**	0.743	**0.804**
	NBC	0.508	0.622	0.606	**0.782**	0.632
	GBC	0.574	**0.707**	0.682	0.755	0.695

Table 4.10: The F1-score measured for the various prediction windows in Experiment 7.

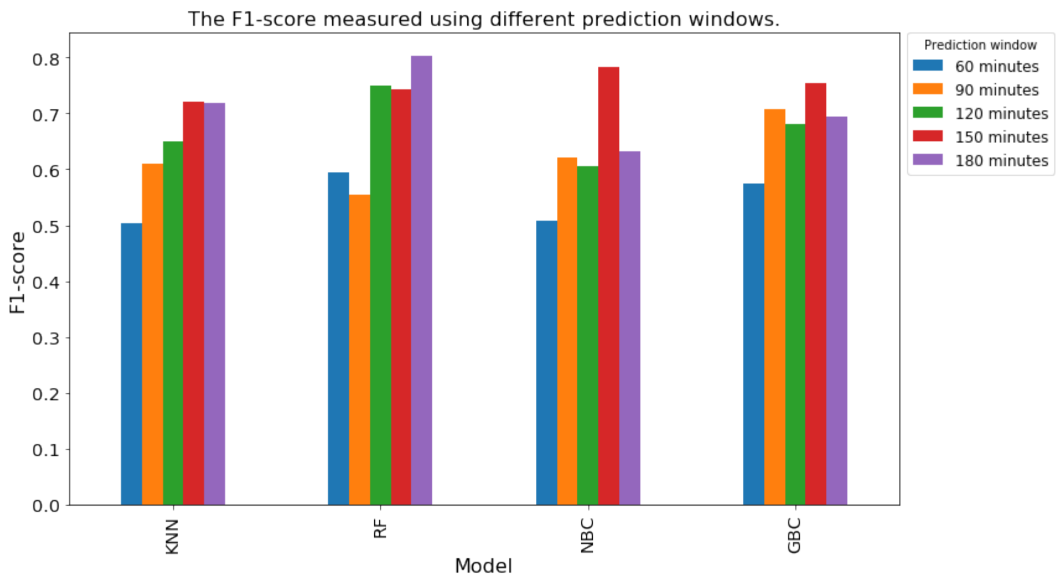


Figure 4.5: The F1-score measured on unseen data using different prediction windows as recorded in Experiment 7.

4.2.8 | Experiment 8 - Forecasting Selected Faults

Not all machine faults are equal in terms of losses for the manufacturing firm. In this experiment we tested the ML models' ability to predict three faults that are more expensive than others. We labelled a dataset using a 90-minute prediction window for these three faults. The dataset was scaled using L2 normalisation, smoothened using a simple moving average with a rolling time window of 30 minutes and class balanced through SMOTE. Feature selection was also applied to reduce the data.

	Model	Accuracy	Precision	Recall	F1-score
Validation	KNN	0.608	0.015	**0.536**	0.029
	RF	**0.796**	0.016	0.300	0.031
	NBC	0.368	0.008	0.479	0.016
	GBC	0.688	**0.017**	0.479	**0.032**
Testing	KNN	0.608	0.135	0.426	0.205
	RF	**0.657**	**0.152**	0.413	**0.222**
	NBC	0.430	0.118	0.585	0.196
	GBC	0.493	0.134	**0.599**	0.219

Table 4.11: The results measured in Experiment 8 when training ML models to forecast selected faults.

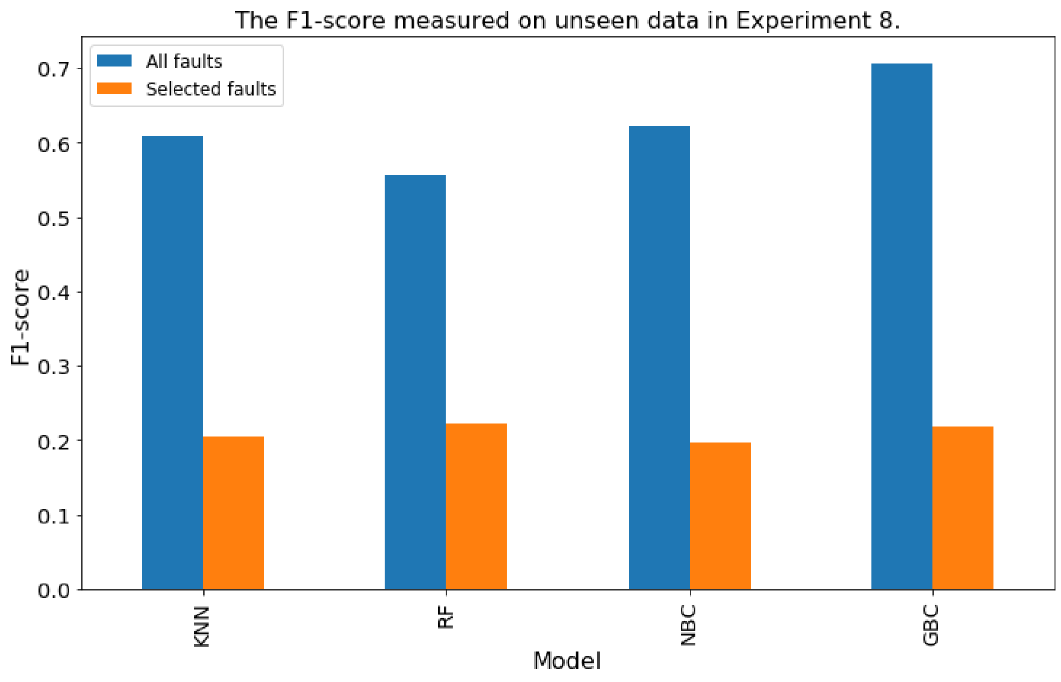


Figure 4.6: A comparison of the F1-score measured when predicting all machine faults with that measured when predicting selected faults.

The results obtained when validating and testing the learning algorithms are displayed in Table 4.11 while Figure 4.6 shows a comparison of the models' performance as measured in this experiment against their equivalent when classifying all machine faults. We observed a drop in performance for all models, with lower precision and F1-score when compared to previous experiments. The results potentially indicate that the wire bonders' behaviour in terms of machine health indicators (vibrations, temperature, etc.)

does not vary between expensive and non-expensive faults, therefore the models were unable to distinguish the expensive faults from the rest.

4.3 | The Predictive Maintenance Framework

We integrated the various components produced in this research project into a PdM framework. An architectural view of our PdM framework is presented in Figure 3.6. This consisted of:

1. The ETL scripts that extracted the data generated by the wire bonders, mapped it onto the ontology and loaded it into the knowledge store;
2. The EKG that acted as a persistent repository of the IIoT data, but also completed the knowledge through inferencing;
3. The predictive models that forecasted machine faults within specific time windows based on vectors of machine health indicators; and
4. A dashboard that retrieved the latest IIoT observations from the EKG, passed on this data to the predictive models to received back the predictions and projected selected indicators as well as the predictions to its users.

We found no formal evaluation method for PdM frameworks in literature, so we evaluated its core components - the ontology (see Section 4.1.1), the EKG (see Section 4.1.2) and the predictive models (see Section 4.2). Both the ontology and the EKG resulted in an accuracy score of *1.0*. The predictive models used in our PdM framework were chosen based on the results of the experiments described in Section 4.2. Two predictive windows are considered, the 90-minute and 120-minute time windows, and the best performing ML models for these time windows were integrated into the PdM framework. The evaluation results of the models used within the system are featured in Table 4.12.

Predictive window	Model	Accuracy	Precision	Recall	F1-score
90-minute	GBC	**0.672**	0.658	0.765	0.707
120-minute	RF	0.664	**0.685**	**0.829**	**0.750**

Table 4.12: The evaluation results of the ML models used within the PdM framework.

We tested the PdM framework functionality through a simulation, where we employed a dummy process that updated the EKG with new IIoT data every three minutes. At the same time, every three minutes the dashboard requested the latest data available from the EKG through a SPARQL query, obtaining an RDF message in return. The IIoT observations were extracted from the RDF message and passed on to the predictive models for their predictions. Selected machine health indicators and the predictions were projected on the dashboard in the form of charts. During this testing we noted that the system met its intended functionality. We also noted that the results projected on the dashboard were correct.

4.4 | Discussion

As stated in Section 4.1.1, the ontology evaluation using the task-based approach (Obrst et al., 2007; Raad and Cruz, 2015) resulted in an accuracy of *1.0*. This shows that the ontology, and therefore the EKG schema, was fit for its intended purpose. In fact, the knowledge store was capable of inferring new knowledge based upon the ontology. A similar behaviour was observed by Umiliacchi et al. (2011), where the ontology developed was able to infer knowledge that was not specifically coded. The retrospective evaluation (Paulheim, 2017) on the EKG also resulted in an accuracy score of *1.0*. This gave the assurance of the correctness of the IIoT measurements that were being used for PdM. It also signified that the EKG was a suitable paradigm for storing IIoT data, hence answering research question *RQ1*. This is also backed up by other research studies that used the EKG to store IIoT data, which were discussed in Section 2.2.3. It should be noted that the EKG was evaluated by a single judge, which was a shortcoming since Paulheim (2017) recommends to have multiple judges. According to Petersen et al. (2017), the ideal profile for such judges are the domain experts, who have in-depth knowledge of the field and can provide insight about the validity of the EKG in addition to its accuracy. However, due to the unavailability of the domain experts in the evaluation period, we evaluated the correctness of the results obtained through the SPARQL queries by manually comparing them against the original datasets.

The experiments we conducted aimed at finding the ML models that provided the best performance. These resulted to be the GBC and RF that produced an F1-score of *0.707* and *0.75* respectively. We also measured the precision and recall since these are important metrics in PdM. As discussed in Section 2.3.2.3, the precision measures the fraction of positive predictions that were correctly classified by the models, while the recall measures the fraction of positively classified instances that were correct. In terms of PdM, a low precision is an indication of a high rate of false positives, which results in unnecessary maintenance activities and therefore unnecessary costs. A low recall is an indication of a high fraction of false negatives, which results in faults going undetected and leading to unplanned machine stoppages due to breakdowns. As shown in Table 4.12, the models we used for our PdM framework have a precision of *0.658* and *0.685*. This means that 34.2% of the fault alerts given by the framework 90 minutes in advance of the possible fault and 31.5% of those given 120 minutes in advance, were false alarms. On the other hand, our models register a recall of *0.765* and *0.829*. This means that the GBC did not detect 23.5% of the faults while the RF did not detect 17.1% of the faults.

In Calabrese et al. (2020), the authors trained their models using the log files for five months of operations of 14 woodworking industrial machines. The authors report a precision of *0.991* and a recall of *0.996*. This results in an F1-score of *0.993* when predicting faults 30 days in advance. In Paolanti et al. (2018), the authors obtained a precision of *0.94* and a recall of *0.95*, resulting in an F1-score of *0.945*. In this case the data consisted of more than 530,000 samples of rotation speed, power, current and vibrations taken from a single wood cutting machine. The model was then evaluated on fault predictions for the same machine from which the training samples were collected. On the other hand, Susto et al. (2015) report a precision of *0.693* and a recall of *0.63*, which result in an F1-score of *0.66*. The authors trained their models on a dataset consisting of electrical and pressure records for 3,671 runs (from part installation to failure) of several ion implantation tools. In Kanawaday and Sane (2017), the authors use a dataset consisting of one month of pressure and tension records for a single slitting machine to train their model. They report an accuracy of *0.987* when evaluating the model on the same machine. In contrast with the results obtained by previous research studies, our results were inferior and only comparable to those obtained by Susto et al. (2015). However, it must be noted that different machines exhibit different behaviour and the results were not comparing like with like as the problems being tackled varied, as did the data that was used. They do however give an indication of the level of accuracy that is expected in industry.

Better results could have potentially been achieved by improving the quality of the training data. When compiling the training data, we had no knowledge of maintenance activities that were carried out on the wire bonders during the 23 days covered by the datasets. Calabrese et al. (2020) and Paolanti et al. (2018) took into account the maintenance records in preparing their training data since maintenance activities alter the condition of the machines. If we had maintenance records at our disposal, these would have been factored in when preparing the datasets, which would have potentially improved the results. Moreover, Sipos et al. (2014) emphasise on the importance of the involvement of the domain experts in each step of the model development process. In our work the lack of domain expertise was substantially felt in the data labelling step. The technical nature of the messages produced by the wire bonders made them hard to interpret and therefore to distinguish between faults and non-faults. When in doubt we assumed the messages to signify faults, but in doing so we potentially included label noise in the training data. The involvement of the domain experts in the data labelling exercise would have resulted in better quality datasets that would possibly have provided better results. Moreover, other research studies used more training data samples than that made available for our dissertation. The availability of more data could potentially improve the results. Notwithstanding the room for improvement, the results obtained show that the ML models were able to predict faults of the wire bonders based on IIoT data with an accuracy of *0.66* to *0.67* and an F1-score of *0.7* to *0.75* thus answering *RQ2*. This is in line with previous research found in literature as discussed in Section 2.3.2.3.

The prototype of the PdM framework that was developed as part of this dissertation integrated the ETL scripts, the EKG and the predictive models. Moreover, a dashboard was added to convey the predictions to the user as described in Section 3.4.4. The PdM framework aimed to predict wire bonder faults up to 120 minutes in advance. As discussed in Section 2.3, a PdM strategy should predict faults as early as possible. Furthermore, the time period a fault is predicted in advance depends on the problem being investigated (Susto et al., 2015). In the case of the wire bonders used in this study it was observed that these generate between four to eight errors a day. Therefore, we had to consider shorter prediction windows than those we found in literature that normally range from days (Amihai et al., 2018; Sipos et al., 2014) to weeks (Calabrese et al., 2020).

As discussed in Section 3.4, our PdM framework was supported by an EKG. This coupling did not improve the predictive performance of the framework, however, it resulted in a number of benefits.

1. The EKG was able to integrate disparate data sources. Not only did such data vary in format, structure and vocabulary but even the sampling frequency differed across the sources. In our implementation, this problem was handled by mapping the data sources onto the ontology, that provided a homogeneous data structure that was flexible enough to accommodate all the data sources. In this way, the consumer side of the PdM framework did not need to cater for the complexity caused by the heterogeneous data sources. This approach also features in

the works of Medina-Oliva et al. (2014) and Schmidt et al. (2017), who use an EKG as a data integration paradigm to support their PdM framework.

2. The complexity of the data retrieval of the PdM framework was also reduced by the common vocabulary brought about by the EKG. The three datasets used in this dissertation had different vocabularies. Furthermore, the wire bonders used two different vocabularies when generating machine logs even though the format was the same. By introducing the EKG, the different vocabularies were transformed into more generic terms such as *sensor* and *observation*, but preserving the information about which sensor was generating which observation. Therefore, the consumer part of the PdM framework could request the sensor observations without the need to know which sensor is generating such observations and which vocabulary is being used. This property of a KG was also exploited by Schmidt et al. (2017) where the PdM framework uses the semantics brought by the ontology to retrieve data from disparate data sources using different vocabularies.

3. As discussed in Section 4.1.1, the EKG was able to infer new knowledge by using the ontological logic. This reduced the complexity of the ETL scripts and improved their efficiency since the number of triples that needed to be transferred to the EKG was reduced. At the same time, the completeness of the knowledge stored within the EKG was maintained since the missing triples were inferred. This occurrence was also observed in Umiliacchi et al. (2011), where the PdM software could make use of knowledge that was not explicitly written into the RDF store. This was also an advantage of the use of the EKG over other data storage paradigms that do not inference new knowledge.

The outcome obtained from the PdM framework shows that the EKG was a suitable data structure to support PdM. Moreover, the PdM framework benefitted from the characteristics of the EKG, thus providing an answer to *RQ3*.

4.5 | Summary

We introduced this chapter by revisiting the objectives of this research project. We then explained how the ontology and the EKG were evaluated and presented the results obtained from such evaluations. We also saw the results obtained in the various experiments we conducted on the predictive models. Moreover, we described how the PdM framework was implemented through the integration of the components developed throughout this research project and how it was tested for correctness. Finally, we discussed and interpreted the results obtained. We also saw how the findings provide answers to our research questions.

5 Conclusions

This chapter summarises our dissertation and explains how we achieved our objectives and what answers we can provide to our research questions. We also discuss the limitations of this dissertation and then proceed with a discussion on how this research can mature further by proposing other areas of research related to the use of the EKG in supporting PdM. These may be taken up as future research projects.

In Chapter 1 of this dissertation, we explained the motivation behind this research. We stated that a major characteristic of the EKG is its data integration capabilities, including its ability to integrate heterogeneous data. We also explain that the semantic enrichment of the data within the EKG detaches the various data sources from their original vocabulary and transforms them into a structure that is described by a harmonised vocabulary. We hypothesised that these characteristic can be exploited to store IIoT data, that was heterogeneous in its form and vocabulary, into a homogeneous data structure. The EKG can then be leveraged for PdM. This is not a novel concept, in fact Kharlamov et al. (2017), Medina-Oliva et al. (2014), Schmidt et al. (2017) and Voisin et al. (2013) take on this approach. However, considering the advantages of PdM and the flexibility the EKG can provide, the number of research projects using this approach is relatively small, possibly because the EKG is regarded as an Enterprise Information System rather than a data model specialised for a specific task. This motivated us to provide another case study demonstrating how the characteristics of the EKG can be exploited for PdM of wire bonding machines. Pursuing this motivation, we defined our research questions, and the aims and objectives of this dissertation.

Chapter 2 started by describing the semiconductor manufacturing process, with particular attention to the wire bonding process and the various maintenance strategies that manufacturing firms adopt. It proceeds by explaining EKGs and ontologies, how they are created, used and evaluated. Lastly, the chapter discusses PdM, focusing on the ML based approach to PdM.

Our methodological approach is described in Chapter 3. First, we introduced the datasets that were used in this dissertation. Second, we described how we created the ontology to then proceed with creating and populating the EKG. Third, we explained how we preprocessed the data to then train and validate a number of ML models through a series of experiments. Lastly, we discussed how we implemented a PdM framework using the EKG as its source of knowledge.

In Chapter 4 we evaluated the ontology and the EKG. We also presented the results of the experiments on the ML models, and described how we chose the predictive models to be used within the PdM framework. An interpretation of the obtained results concludes the chapter.

5.1 | Achieved Aims and Objectives

This dissertation investigated whether the EKG can be leveraged for PdM. This was split into three research questions as follows:

(RQ1) Can an EKG effectively store IIoT data?
(RQ2) Can wire bonder failures be predicted from IIoT data and machine logs?
(RQ3) Can an EKG structure support a PdM framework?

In line with these research questions, a number of objectives were established. The first objective was to design an ontology for wire bonders, their sensor network and their measured observations. This objective was achieved through the procedure described in Section 3.2.1. The evaluation of the ontology is described in Section 4.1.1 and resulted in an accuracy of *1.0*. The designed ontology models wire bonding machines, their states, their sensors and sensor observations as shown in Figure 3.4.

The second objective involved the transformation of IIoT data into an EKG using automated or semi-automated approaches. This objective was also achieved. We followed the method explained in Section 3.2 to build the EKG in the form of an RDF store using the developed ontology as its schema. The IIoT data was extracted, mapped onto the ontology and pushed into the RDF store using ETL scripts that were specifically written for this task. The EKG was evaluated as explained in Section 4.1.2, resulting in an accuracy score of *1.0*.

Through the achievement of the first two objectives, we answered *RQ1*. The EKG was able to effectively store the IIoT data generated by wire bonders in addition to knowledge about the wire bonders themselves, their states and their sensors. Moreover, the EKG was able to infer missing links in the knowledge through the logic defined in the ontology.

The third objective consisted of training an ML model to predict faults of wire bonding machines. Section 3.3 describes how we trained different ML models on a dataset composed of IIoT data and machine logs, and experimented with various preprocessing techniques to predict such faults within specific time windows. We then retained the best two performers to incorporate them within the PdM framework. These were the RF and GBC, both trained on a dataset that was scaled using L2 normalisation, class-balanced using SMOTE, smoothened using a simple moving average with a 30-minute rolling time window and reduced using a filter method. The RF predicted machine faults up to 120 minutes in advance with an accuracy of *0.664* and an F1-score of *0.75*. The GBC, on the other hand, predicted faults up to 90 minutes ahead with an accuracy of *0.672* and an F1-score of *0.707*. We can therefore state that the third objective was also achieved and that this answers *RQ2* - that the ML models were able to predict wire bonder failures from IIoT data and machine logs.

The last objective of this dissertation was to build a PdM framework that is supported by an EKG. To achieve this objective we built a PdM framework as described in Section 3.4. The framework consisted of a dashboard that retrieved the latest IIoT data from the EKG and passed on this data to the predictive models to obtain a set of predictions in return. The IIoT data and the predictions were then projected to the user through the graphical user interface. A pictorial representation of the final product is featured in Figure 3.6.

The achievement of this objective allowed us to answer *RQ3*, whereby we investigated whether the EKG can support a PdM framework. Moreover, we concluded that the EKG was leveraged by the PdM framework in three ways:

1. The EKG served as a data integration paradigm, merging IIoT data coming from different sources and having a different format, structure, vocabulary and sampling frequency into a homogeneous data structure that the PdM framework could then query for the latest sensor observations. Without the EKG, the data extraction procedures of the PdM framework would have been very complex to cater for this diversity. Moreover, multiple such procedures would be needed as these must be specific for each data source.

2. The EKG also served to standardise the vocabulary across all data sources. Each data source used in this dissertation had its own vocabulary. Additionally, different models of wire bonders used different vocabulary to record the machine logs. With the introduction of the EKG, the PdM framework could consider every wire bonder as having the same characteristics and using a common vocabulary to describe these characteristics. Once again, without the EKG, the data extraction methods of the PdM framework would need to be specific to the vocabulary of the respective data source, thus multiple such methods are required and these methods may not have been reusable for other wire bonders.

3. The EKG was able to infer new knowledge through ontological reasoning. This completed any gaps in the knowledge that was obtained through the ETL scripts. This feature of the EKG reduced the complexity and increased the efficiency of the ETL scripts as these did not need to be complete, but the completeness of the knowledge was still maintained by the EKG. Without the EKG the data extraction methods would need to ensure that the data being extracted is complete.

5.2 | Limitations

Although the results of the PdM framework prototype we presented in Chapter 4 were promising, one must also consider its limitations. The prototype was designed and tested for a limited number of wire bonders and its behaviour when dealing with a large number of wire bonders was not tested. In such scenario, the velocity of the data increases and the prototype was not equipped to handle data velocity.

Another limitation of the system is that it assumed that all wire bonders had the same sensors, when this may not be the case. This could be done because we used one particular model of wire bonder, but different models may have a different sensory network and the system was not designed to cater for such an occurrence. Moreover, the EKG did not discover new machines and sensors automatically from the data being generated, as the ETL scripts extracted only the sensor observations for known devices. Therefore, wire bonders and sensors had to be manually defined in the EKG. Additionally, the system did not validate the IIoT data, so it had no inbuilt mechanism to stop noise. These limitations can be addressed by enhancing the ETL processes to be more flexible and cater for these issues. For the auto-discovery of wire bonders and their sensors to be implemented, the ontology needs to be further specialised for wire bonders. It must also define each sensor characteristics so that when data from unregistered equipment is detected, the EKG can infer the properties of the equipment from the characteristics of the data.

The predictive models generated many false positives, which if taken into account would result in unnecessary stoppages. This is very expensive for the manufacturing firm as it constitutes a waste of time and resources. Therefore, the ML models' performance must improve for the PdM framework to be feasible. The predictions may improve by taking maintenance records into account during the preparation of the training data in a similar approach to those used in Calabrese et al. (2020) and Paolanti et al. (2018). Moreover, as explained in Section 4.4, the data labelling process can be improved by involving the domain experts to ensure that the training data is correctly labelled. Another limitation, as determined through Experiment 8 (Section 4.2.8), was that the predictive models were unable to predict selected faults that are more critical than others. The reason behind this could potentially be that the "symptoms" of these faults on the wire bonders in terms of machine health indicators were not different than those for other faults, therefore the models could not distinguish expensive from non-expensive faults.

The PdM framework as designed for this research incorporated two binary classifiers that predicted a fault within a specific time window. When a possible fault was predicted, the system could not distinguish the type of the fault being detected, hence it could not recommend the action to be taken. Some faults require the wire bonder to be stopped for maintenance, whilst for other faults a realignment of certain machine parts would solve the problem. Without knowledge of the type of upcoming fault it is difficult for the machine operator to decide upon the best remedial action to take.

5.3 | Future Work

The objectives set for this dissertation were met, nonetheless, this research can be extended to further explore the potential of the EKG within the sphere of PdM. We propose four related works that can be considered in the future.

a) Predict the upcoming fault: As discussed in Section 5.2, one of the limitations of our PdM framework was that, although it predicted a possible upcoming fault, there was no indication of what this fault might have been. Further research may explore the identification of the fault to aid the machine operator to determine the best action to take. This would improve the effectiveness of the PdM framework. A possible approach to do this is that used by Farokhzad et al. (2012), where the features are passed through an artificial neural network that was trained to output different codes that signify the predicted fault.

b) Include maintenance data into the EKG: As explained in Section 4.4, the maintenance records of the wire bonders were not taken into account in this dissertation. Maintenance, however, plays an important role in the health level of the machines. Works such as Calabrese et al. (2020) and Paolanti et al. (2018) take maintenance records into account when training their predictive models. It would be interesting to investigate whether maintenance records of wire bonders can ameliorate the performance of the predictive models. Moreover, a future study should investigate how the ontology must be adapted to fit the maintenance records and how such information can be collected into the EKG.

c) Estimate the RUL of a machine: As discussed in Section 2.3, another use of ML in PdM is to estimate the remaining time to the next failure, or the RUL. In Amihai et al. (2018), the authors use vibrations FFT data to estimate the RUL of industrial pumps by using an RF model. They used a proprietary software that acts as middleware by reading the sensor observations and sending them to the ML model. Future research can investigate whether an EKG can support a PdM framework that estimates RUL and whether its properties offer any value added over other data models.

d) Extend the PdM framework to other types of machines: This dissertation dealt with wire bonding machines, creating a PdM framework that monitors their machine health indicators and alerting of any predicted upcoming faults. An extension of this research can focus on widening the scope of the PdM framework to other types of machines. As we saw in Section 2.3.1, the machine health indicators that are indicative of the condition of a machine vary according to the type of machine. Hence the future work should investigate how to use the EKG for context awareness as discussed in Section 2.2.2.2, to allow the PdM framework to determine the type of the machine that is generating the IIoT data. This may then lead the PdM framework to extract the required machine health indicators to be passed on to the corresponding ML models for this machine type. A similar concept is discussed in Medina-Oliva et al. (2014), where the authors use a context aware system that identifies the equipment generating the data and searches the knowledge base for diagnostic tasks related to similar units.

5.4 | Final Remarks

This dissertation studied whether the EKG is a suitable data structure to store IIoT data generated by wire bonding machines and whether it can be leveraged for PdM. We used actual data generated in a semiconductor manufacturing plant to design an ontology and construct an EKG. ETL scripts were written to transfer the IIoT data into the EKG. The IIoT data was also used to train and validate various ML models to predict machine faults within a specific time window. All the developed components were then integrated into a PdM framework that extracted the machine health indicators from the EKG, and were then projected on a dashboard. Additionally, the dashboard alerted of any predicted upcoming wire bonder faults.

The results generated through this research are promising and provided an answer to our main research question, showing that the EKG can be leveraged for PdM as a data integration paradigm for heterogeneous IIoT data, by standardising the vocabulary of disparate data sources and by inferring new knowledge through ontological reasoning.

Dataset structures

1a) ETS data structure (main variables)

Variable	Datatype	Description
MANUFACTURINGMODULE	String	The physical location of the machine (known as tunnels).
EQUIPMENTNAME	String	The name of the machine, also used as its unique identifier.
STEP	String	The current step within the production process.
PRODUCT	String	The name of the product being produced.
QTY	Integer	The quantity of products being produced.
TXNTIMESTAMP	Datetime	The timestamp of the record. This is taken to be the end time of the event.
DURATION	Float	The duration of the event.
EQPSTATE	String	The state of the machine.
LOGINTIME	Datetime	The time when the user logged into the machine.
LOGINUSER	Integer	The ID of the user operating the machine.
LOGINNAME	String	The name of the user operating the machine.
REASONCODEID	String	A description of the event.
EQPTYPE	String	The equipment type.
EQPBRAND	String	The brand of the machine.
EQPMODEL	String	The model of the machine.

1b) Sample data from the ETS logs

```
20190320 060000,QFP-WB-TUNNEL_5,MAINT-AREA-QFP_WB_KS_ICONN_PLUS,WB-KS-ICONN_PLUS-045,WB KS687,22912XDMRF,
A20-WIRE-BONDING-COPPER-B,C9X6*UM14BGS,0,8.232,T,T,PE-PROD,TD,TUD,TFUD,TFUD,="ProcessStop",
20190319 183931,255752,OPERATOR1152,20190320 020812619,1,WB-KS-ICONN_PLUS,WB,KS,ICONN_PLUS
20190320 060000,QFP-WB-TUNNEL_5,MAINT-AREA-QFP_WB_KS_ICONN_PLUS,WB-KS-ICONN_PLUS-045,WB KS687,22912XDMRF,
A20-WIRE-BONDING-COPPER-B,C9X6*UM14BGS,0,.494,T,T,PE-PROD,TU,TUP,TPT,TPR,="ProcessStart",
20190319 183931,255752,OPERATOR1152,20190320 020804387,1,WB-KS-ICONN_PLUS,WB,KS,ICONN_PLUS
20190320 060000,QFP-WB-TUNNEL_5,MAINT-AREA-QFP_WB_KS_ICONN_PLUS,WB-KS-ICONN_PLUS-045,WB KS687,22912XDMRF,
A20-WIRE-BONDING-COPPER-B,C9X6*UM14BGS,0,28.517,T,T,PE-PROD,TD,TUD,TFUD,TFUD,="ProcessStop",
20190319 183931,255752,OPERATOR1152,20190320 020803893,1,WB-KS-ICONN_PLUS,WB,KS,ICONN_PLUS
20190320 060000,QFP-WB-TUNNEL_5,MAINT-AREA-QFP_WB_KS_ICONN_PLUS,WB-KS-ICONN_PLUS-045,WB KS687,22912XDMRF,
A20-WIRE-BONDING-COPPER-B,C9X6*UM14BGS,0,.4,T,T,PE-PROD,TU,TUP,TSBP,TSBPM,
="RC␣Entered=ALIGN␣STATUS␣-␣INADEQUATE␣EYE␣POINTS␣FOUND␣VISION",20190319 183931,255752,OPERATOR1152,
20190320 020735059,1,WB-KS-ICONN_PLUS,WB,KS,ICONN_PLUS
20190320 060000,QFP-WB-TUNNEL_5,MAINT-AREA-QFP_WB_KS_ICONN_PLUS,WB-KS-ICONN_PLUS-045,WB KS687,22912XDMRF,
A20-WIRE-BONDING-COPPER-B,C9X6*UM14BGS,0,270.807,T,T,PE-PROD,TD,TUD,TFUD,TFUD,="ProcessStop",
20190319 183931,255752,OPERATOR1152,20190320 020734659,1,WB-KS-ICONN_PLUS,WB,KS,ICONN_PLUS
20190320 060000,QFP-WB-TUNNEL_5,MAINT-AREA-QFP_WB_KS_ICONN_PLUS,WB-KS-ICONN_PLUS-045,WB KS687,22912XDMRF,
A20-WIRE-BONDING-COPPER-B,C9X6*UM14BGS,2,86.705,T,T,PE-PROD,TU,TUP,TPT,TPR,="UnitProcessed",
20190319 183931,255752,OPERATOR1152,20190320 020303852,1,WB-KS-ICONN_PLUS,WB,KS,ICONN_PLUS
```

2a) Machine Logs structure (main variables)

Variable (Format A)	Variable (Format B)	Description
Equipment	Equipment	The name of the machine, also used as its unique identifier.
DateTimeUtc	DateTimeUtc	The timestamp of the record.
SVID_3007	SV_ProcState	The current state of the wire bonder.
SVID_3008	SV_PrevProcState	The previous state of the wire bonder.
SVID_3030	SV_EquipmentMode	The current mode of operation.
SVID_3031	SV_LightTower_State	The state of the light tower.
SVID_3045	SV_OpeModeCode	The current open mode.
SVID_3051	SV_USGImpedance	USG low impedance tuning of the machine.
SVID_3052	SV_USGFrequency	Machine free air tuned low USG frequency.
SVID_3062	SV_HFUSGFrequency	Machine free air tuned high USG frequency.
SVID_3064	SV_HFUSGImpedance	USG high impedance tuning of the machine.
SVID_3096	SV_InhibitAuto	Inhibit auto mode indicator.
SVID_3116	SV_CompletedPPEdit	Last edit performed.
SVID_3200	SV_Zone1_Temperature	Temperature at pre-bond state (Celsius).
SVID_3202	SV_Zone3_Temperature	Temperature at post-bond state (Celsius).
SVID_3204	SV_Zone2_Temperature_offset	Bond temperature offset.
SVID_3205	SV_Zone3_Temperature_offset	Post-bond temperature offset.
SVID_3300	SV_Input_LF_Count	Number of lead frames removed.
SVID_3301	SV_Output_LF_Count	Number of lead frames inserted.
SVID_3302	SV_SECS_Devices_Processed	Number of successfully processed devices.
SVID_3401	SV_Osc_Current_Factor	USG current factor.
SVID_3406	SV_HF_Person_Current_Factor	High frequency USC current factor.
SVID_3543	SV_PbiUnitNumber	Post bond inspection number
SVID_3549	SV_PbiUnitAvgXBallPlcErr	Average ball placement error (X)
SVID_3550	SV_PbiUnitXBallPlcErrMin	Minimum ball placement error (X)
SVID_3551	SV_PbiUnitXBallPlcErrMax	Maximum ball placement error (X)
SVID_3553	SV_PbiUnitAvgYBallPlcErr	Average ball placement error (Y)
SVID_3614	SV_LatestErrorContext	Latest error context.
SVID_3616	SV_LatestErrorWire	Latest error wire.
SVID_3617	SV_LatestErrorBond	Latest error bond.

2b) A single sample from the Machine Logs dataset

```
{"Equipment":"WB-KS-ICONN-207",
 "DateTimeUtc":"\/Date(1547337618606)\/",
 "SensorData":{"SVID_3105":"SeqStopXhairCorrection","SVID_3803":[18,18],"SVID_3801":[12,12],"SVID_3805":[1,1],
                  "SVID_4016":[0,0],"SVID_4019":[0,0],"SVID_3804":[9,9],"SVID_3802":[115,115],"SVID_4017":[0,0],
                  "SVID_4018":[0,0],"SVID_3808":[18,18],"SVID_3806":[5,5],"SVID_3810":[0.4,0.3],"SVID_4022":[1,1],
                  "SVID_4020":[1,1],"SVID_4021":[0,0],"SVID_3809":[4,6],"SVID_3807":[100,100],"SVID_4029":[1,1],
                  "SVID_3800":[1,2],"SVID_4024":[15,15],"SVID_4027":[0,0],"SVID_4023":[400,400],"SVID_4025":[0,0],
                  "SVID_4026":[0,0],"SVID_3011":"2019011300584000","SVID_3116":2,"SVID_3812":[30,30],"SVID_3030":1,
                  "SVID_3811":[1.35,1.35],"SVID_3405":"7.475204","SVID_3406":"1.068798","SVID_3062":121121,
                  "SVID_3064":47.9537468,"SVID_3096":0,"SVID_3300":13,"SVID_3813":[15,5],
                  "SVID_3830":"Bond␣Force␣Calibration","SVID_3617":2,"SVID_3614":16386,"SVID_3421":"U2[6,7]",
                  "SVID_3420":"49-6","SVID_3616":10,"SVID_3031":16384,"SVID_4028":[1.45,1.3],"SVID_3815":[-4,-2],
                  "SVID_3045":4000,"SVID_3401":7.699526,"SVID_3301":12,"SVID_3576":0.01693913,"SVID_3581":0.05508021,
                  "SVID_3568":211720,"SVID_3567":106014,"SVID_3566":"12/27/18␣06:52:03","SVID_3569":635171,
                  "SVID_3579":0.01693913,"SVID_3578":0.01693913,"SVID_3584":0.05508021,"SVID_3583":0.05508021,
                  "SVID_3549":0.01693913,"SVID_3553":0.05508021,"SVID_3543":211720,"SVID_3542":"01/13/19␣00:54:36",
                  "SVID_3551":0.01693913,"SVID_3550":0.01693913,"SVID_3555":0.05508021,"SVID_3554":0.05508021,
                  "SVID_3520":569113,"SVID_3004":"DM00159639H_MV1E_0.6G_15L","SVID_3008":2,"SVID_3007":3,
                  "SVID_3814":[20,0],"SVID_3302":211724,"SVID_3052":50519,"SVID_3051":149.008972,"SVID_3200":130,
                  "SVID_3203":-12,"SVID_3201":175,"SVID_3204":-22,"SVID_3202":100,"SVID_3205":-10},
 "SetPointData":{"ECID_1119":true,"ECID_1036":4000,"ECID_1040":1000,"ECID_1118":true,"ECID_1508":"Unknown",
                   "ECID_1093":1}}
```

3a) IIoT data structure

Variable	Datatype	Description
Timestamp	Datetime	Timestamp in Posix time
AccX_0	Float	
..	Float	64 FFT bins from 0 Hz to 3.3 kHz for acceleration along the X axis
AccX_3278	Float	
AccY_0	Float	
..	Float	64 FFT bins from 0 Hz to 3.3 kHz for acceleration along the Y axis
AccY_3278	Float	
AccZ_0	Float	
..	Float	64 FFT bins from 0 Hz to 3.3 kHz for acceleration along the Z axis
AccZ_3278	Float	
GyroX_0	Float	
..	Float	64 FFT bins from 0 Hz to 3.3 kHz for motion along the X axis
GyroX_3278	Float	
GyroY_0	Float	
..	Float	64 FFT bins from 0 Hz to 3.3 kHz for motion along the Y axis
GyroY_3278	Float	
GyroZ_0	Float	
..	Float	64 FFT bins from 0 Hz to 3.3 kHz for motion along the Z axis
GyroZ_3278	Float	
Sound_0	Float	
..	Float	512 FFT bins from 0 Hz to 24 kHz for sound
Sound_23956	Float	
MagX_0	Float	
..	Float	64 FFT bins from 0 Hz to 49 Hz for magnetism along the X axis
MagX_4914	Float	
MagY_0	Float	
..	Float	64 FFT bins from 0 Hz to 49 Hz for magnetism along the Y axis
MagY_4914	Float	
MagZ_0	Float	
..	Float	64 FFT bins from 0 Hz to 49 Hz for magnetism along the Z axis
MagZ_4914	Float	
Press_0	Float	Pressure (hPa)
Temp_1	Float	Tempure 1 (Celsius)
Temp_2	Float	Tempure 2 (Celsius)
Hum_0	Float	Humidity (rH)

3b) A single sample from the IIoT dataset

```
1544329551845, 0.140351, 0.162535, 0.377842, 0.655402, 0.607056, 0.23281, 0.175543, 0.117039, 0.0586403, 0.038782,
0.0250604, 0.0174227, 0.0153434, 0.016584, 0.0213786, 0.0395947, 0.0648703, 0.146411, 0.221832, 0.100012, 0.0548418,
0.0342347, 0.0235935, 0.0178965, 0.0145263, 0.0127666, 0.0110924, 0.0100339, 0.00938973, 0.00899834, 0.00851992,
0.00841965, 0.00843489, 0.00838367, 0.00850296, 0.00864012, 0.00874337, 0.0089123, 0.00916262, 0.0093141, 0.00953514,
0.00963165, 0.00995479, 0.010211, 0.0104953, 0.0106819, 0.0109784, 0.0112502, 0.0114399, 0.0116384, 0.0118419,
0.0121426, 0.0121632, 0.0124657, 0.0125258, 0.0126979, 0.0127139, 0.0130013, 0.0131214, 0.013178, 0.0132269, 0.0132178,
0.0132495, 0.013218, 0.0435372, 0.0844868, 0.192331, 0.282712, 0.353034, 0.153065, 0.088893, 0.0465027, 0.0322059,
0.0205334, 0.0133051, 0.0111388, 0.0143933, 0.0247839, 0.0670873, 0.197605, 0.268838, 0.0568928, 0.0397312, 0.0219821,
0.0104986, 0.00723207, 0.00594376, 0.00532793, 0.00477208, 0.00458974, 0.00449373, 0.00454664, 0.00490105, 0.00515922,
0.00526163, 0.00546561, 0.00563263, 0.00580485, 0.00601879, 0.00621609, 0.00642487, 0.00670677, 0.00697593, 0.00717774,
0.00744699, 0.00765592, 0.00793135, 0.0081125, 0.0083921, 0.00865858, 0.00884595, 0.00901227, 0.009229, 0.00940926,
0.00965742, 0.0097528, 0.00990359, 0.0101897, 0.0102687, 0.0104404, 0.0104889, 0.0106294, 0.0106736, 0.0107834,
0.0108647, 0.010966, 0.0109588, 0.0109602, 3.88229, 5.90697, 8.16279, 9.43413, 8.84884, 2.68597, 1.65497, 1.04773,
0.565569, 0.403172, 0.296933, 0.230693, 0.167799, 0.121622, 0.121645, 0.138163, 0.203214, 0.259114, 0.295282, 0.379804,
0.406257, 0.455711, 0.430239, 0.371668, 0.288993, 0.273787, 0.279841, 0.26986, 0.377047, 0.520736, 0.626613, 0.668789,
0.681416, 0.607859, 0.609606, 0.586527, 0.449187, 0.401362, 0.394167, 0.38097, 0.342912, 0.233715, 0.150407, 0.0991088,
0.0825948, 0.0793234, 0.0842202, 0.0772517, 0.0609333, 0.0511382, 0.0470647, 0.0490257, 0.0586629, 0.0696812, 0.0738302,
0.0631287, 0.0564606, 0.0510075, 0.0482013, 0.0432533, 0.0347232, 0.0272622, 0.0212513, 0.0147019, 7.68974, 9.2023,
8.72909, 7.88263, 7.13883, 6.23779, 5.49876, 4.7659, 4.16901, 3.81796, 3.5646, 3.33727, 3.09394, 2.96472, 2.85873,
2.7249, 2.6085, 2.56278, 2.45152, 2.95011, 2.48608, 2.18515, 2.12112, 2.03272, 2.03705, 2.06053, 2.0129, 1.96645,
1.95072, 1.9329, 1.96249, 1.97906, 1.98996, 1.9986, 1.90319, 1.88937, 1.92425, 1.9699, 2.23731, 2.00816, 1.94094,
1.93065, 1.96357, 2.01877, 2.1018, 2.0659, 1.9813, 1.89227, 1.84033, 1.83001, 1.78959, 1.82651, 1.83086, 1.70179,
1.67274, 1.66285, 1.72563, 1.92792, 1.73359, 1.67883, 1.70246, 1.69035, 1.72327, 1.76323, 11.4177, 9.70081, 7.82455,
7.7889, 7.51939, 6.72093, 5.76974, 4.85663, 4.23674, 3.90842, 3.54094, 3.20539, 2.79466, 2.79684, 2.66076, 2.47822,
2.35683, 2.48527, 2.44444, 3.16771, 2.37654, 2.19014, 2.01182, 1.85467, 1.88379, 1.95936, 1.97319, 1.77999, 1.74412,
1.72384, 1.90302, 1.95364, 2.01391, 2.01329, 1.76021, 1.68635, 1.73726, 1.65028, 2.25777, 1.86664, 1.69801, 1.64513,
1.6278, 1.72938, 1.88895, 1.96787, 1.71279, 1.63734, 1.56269, 1.55708, 1.60511, 1.70158, 1.74543, 1.55449, 1.5213,
1.54819, 1.56011, 2.11063, 1.80333, 1.5716, 1.43611, 1.39915, 1.51095, 1.64856, 8.77489, 8.65304, 8.14047, 7.52437,
7.10499, 6.32862, 5.51723, 4.75273, 4.40132, 3.96368, 3.61158, 3.33524, 2.93234, 2.77585, 2.65346, 2.51072, 2.47489,
2.46898, 2.47964, 2.49662, 2.51694, 2.51761, 2.38724, 2.24136, 2.10326, 2.0039, 1.95427, 1.9826, 1.93122, 1.9085,
1.9606, 2.0432, 2.15359, 2.10561, 2.04729, 1.94498, 1.85208, 1.85789, 1.85582, 1.87982, 2.02414, 2.18929, 2.33886,
2.41477, 2.47148, 2.42262, 2.33015, 2.20747, 2.14288, 2.04043, 1.87982, 1.7971, 1.76775, 1.74094, 1.75342, 1.79944,
1.81745, 1.8655, 1.84908, 1.81891, 1.8145, 1.80378, 1.82057, 1.79603, 3.85364, 10.953, 17.7583, 23.386, 26.4017,
16.2258, 10.1737, 8.43046, 6.86907, 6.82674, 4.56397, 4.09136, 3.52916, 2.77097, 2.64204, 3.24958, 2.50966, 2.45599,
3.02728, 2.72548, 3.10592, 2.95672, 3.17426, 2.16025, 1.80885, 2.10076, 2.70484, 2.02878, 1.63792, 1.46044, 1.54909,
1.90741, 1.70944, 1.93498, 2.91317, 2.88698, 2.48124, 2.26319, 1.68097, 1.69012, 1.79377, 1.80086, 1.63739, 2.1628,
2.51308, 2.22975, 1.8419, 2.15334, 2.83709, 4.54923, 4.86189, 3.37529, 2.91856, 2.95073, 2.20569, 1.96927, 1.77765,
1.88203, 2.02691, 2.31696, 2.52479, 2.57035, 2.22103, 1.93484, 1.78117, 1.62204, 1.52972, 1.54814, 1.60725, 1.59395,
1.49786, 1.31066, 1.18494, 1.2214, 1.24022, 1.24492, 1.42192, 1.54414, 1.38226, 1.46491, 1.67479, 1.68353, 1.48276,
1.14063, 1.02227, 1.15117, 1.31375, 1.35362, 1.32485, 1.22891, 1.31824, 1.41163, 1.33572, 1.33922, 1.39128, 1.23255,
1.33833, 1.42232, 1.49477, 1.54975, 1.49749, 1.49869, 1.45106, 1.25769, 1.18971, 1.1749, 1.22953, 1.31238, 1.31108,
1.26586, 1.27362, 1.34704, 1.48846, 1.69402, 1.68808, 1.64695, 1.56171, 1.51647, 1.41609, 1.26974, 1.13901, 1.10447,
1.21659, 1.41155, 1.52095, 1.51514, 1.50332, 1.58335, 1.6666, 1.80538, 2.06082, 2.02874, 1.94834, 1.90967, 1.87162,
1.67258, 1.5423, 1.54685, 1.49569, 1.51923, 1.4421, 1.338, 1.36576, 1.42416, 1.52168, 1.64114, 1.73544, 2.00433,
2.13352, 1.91701, 1.64447, 1.56694, 1.48608, 1.48118, 1.60337, 1.55064, 1.45427, 1.46644, 1.41753, 1.37876, 1.55493,
1.65946, 1.63933, 1.53409, 1.39132, 1.35368, 1.39603, 1.39497, 1.32417, 1.30837, 1.31173, 1.27345, 1.25992, 1.27322,
1.2733, 1.33313, 1.3247, 1.35917, 1.33136, 1.30024, 1.29421, 1.29838, 1.25952, 1.17509, 1.17539, 1.24694, 1.37636,
1.39729, 1.32282, 1.28364, 1.24639, 1.31549, 1.41126, 1.50832, 1.6354, 1.65465, 1.74471, 1.86028, 1.80162, 1.68831,
1.48904, 1.3352, 1.25494, 1.22924, 1.31387, 1.40884, 1.3995, 1.28812, 1.28053, 1.26024, 1.30362, 1.40784, 1.46287,
1.50658, 1.53946, 1.53208, 1.58467, 1.49811, 1.38026, 1.33408, 1.3605, 1.42994, 1.54522, 1.55935, 1.63117, 1.58549,
1.54428, 1.52166, 1.52181, 1.51622, 1.4819, 1.53082, 1.61284, 1.64132, 1.78349, 1.77703, 1.68456, 1.68798, 1.82623,
2.0626, 2.26931, 2.24531, 2.27727, 2.37767, 2.36873, 2.18574, 2.08046, 1.79521, 1.65484, 1.58674, 1.62524, 1.69284,
1.63869, 1.60569, 1.49603, 1.36931, 1.34845, 1.25784, 1.16605, 1.23325, 1.3113, 1.32004, 1.24599, 1.22083, 1.24542,
1.25899, 1.25126, 1.22456, 1.20152, 1.27396, 1.36379, 1.29344, 1.1928, 1.17528, 1.14174, 1.05306, 1.05263, 1.05111,
1.01186, 1.00071, 1.01262, 1.06222, 1.04614, 1.02737, 1.03292, 1.02645, 1.05576, 1.02089, 0.966899, 1.00551, 1.06144,
1.44017, 1.02698, 0.945458, 0.90754, 0.914496, 0.942274, 0.953636, 0.956054, 0.962801, 0.963022, 0.973426, 1.02309,
1.10553, 1.06722, 1.00322, 1.00684, 1.02749, 1.04906, 1.09356, 1.12591, 1.11466, 1.00689, 0.973923, 0.966092, 0.940978,
0.8995, 0.901481, 0.924687, 0.925126, 0.932677, 0.948044, 0.970194, 0.949396, 0.889974, 0.880771, 0.887597, 0.87533,
0.888609, 0.917996, 0.956911, 0.962137, 0.91165, 0.898914, 0.905613, 0.882678, 0.864319, 0.867324, 0.849205, 0.859825,
0.8778, 0.90602, 0.985129, 1.02299, 1.18648, 4.56092, 4.57906, 1.23718, 1.15926, 1.14874, 1.09774, 1.08662, 1.10963,
1.12916, 1.13967, 1.12368, 1.10736, 1.12274, 1.10978, 1.06451, 1.01509, 0.985817, 0.95682, 0.943779, 0.888244, 0.850105,
0.860354, 0.845326, 0.832066, 0.869336, 0.880069, 0.858414, 0.833744, 0.823772, 0.812454, 0.825661, 0.848911, 0.861886,
0.862353, 0.850249, 0.856772, 0.862311, 0.881014, 0.900877, 0.922228, 0.90467, 0.886851, 0.868363, 0.870205, 0.901312,
0.843471, 0.849848, 0.995506, 1.06638, 0.850935, 0.866643, 0.915399, 0.961889, 0.968181, 0.97314, 0.974264, 0.957718,
0.915557, 0.859411, 0.852366, 0.849763, 0.824769, 0.771205, 0.734416, 0.723701, 0.713714, 0.680252, 0.672208, 0.693906,
0.687905, 0.678037, 0.680182, 0.682276, 0.673091, 0.657992, 0.627294, 0.625639, 0.63989, 0.644429, 0.643007, 0.648277,
0.639816, 0.618529, 0.592196, 0.592046, 0.611992, 0.620911, 0.617387, 0.605452, 0.599818, 0.590026, 0.58225, 0.585784,
0.593931, 0.596038, 0.597463, 0.588902, 0.598713, 0.601191, 0.600151, 0.599974, 0.575551, 0.562048, 0.551509, 0.554265,
0.559461, 0.567157, 0.570807, 0.564079, 0.556273, 0.555393, 0.563983, 0.551024, 0.543964, 0.534544, 0.522196, 0.517061,
0.513052, 0.506628, 0.511502, 0.518674, 0.512916, 0.508515, 0.507791, 0.506718, 0.505545, 0.498124, 0.498037, 0.510568,
0.518801, 0.525085, 0.513109, 0.498523, 0.481196, 0.460838, 0.45257, 0.44842, 0.442762, 0.443711, 0.446817, 0.450713,
0.465906, 0.473673, 0.474267, 0.480519, 0.483923, 0.478726, 0.477335, 0.463236, 0.450535, 0.445008, 0.436271, 0.445918,
0.448159, 0.446182, 0.446763, 0.449198, 0.44759, 0.440894, 0.436385, 0.435428, 0.435819, 0.447877, 0.449096, 0.445102,
0.436253, 0.439804, 1.07801, 0.355309, 0.206391, 0.096041, 0.0619158, 0.0505433, 0.196857, 0.128697, 0.044281, 0.0349287,
0.0370717, 0.0446592, 0.0522259, 0.0582757, 0.0367384, 0.0365492, 0.0302027, 0.0427508, 0.0645847, 0.134875, 0.0427978,
0.0386466, 0.0252631, 0.0244547, 0.028602, 0.0674254, 0.0572727, 0.0602625, 0.0470131, 0.0287518, 0.0205014, 0.0570889,
0.0578055, 0.0487118, 0.0483724, 0.0307764, 0.0194737, 0.0358238, 0.061233, 0.0484821, 0.0488377, 0.0308281, 0.0202133,
0.0284175, 0.0563274, 0.0396292, 0.0565308, 0.0428997, 0.0245944, 0.0255443, 0.0442504, 0.0417156, 0.0282677, 0.0207047,
0.0175982, 0.0203497, 0.0386213, 0.0444794, 0.035359, 0.0424578, 0.0246229, 0.0187892, 0.0280509, 0.0420679, 1.3713,
0.404562, 0.236252, 0.0874457, 0.0292688, 0.0299227, 0.0554707, 0.0405647, 0.0204821, 0.0237705, 0.0273106, 0.0269662,
0.032343, 0.0338886, 0.0218375, 0.0235182, 0.0210697, 0.0222064, 0.031335, 0.0531387, 0.0190261, 0.0208995, 0.0175207,
0.0148224, 0.0193158, 0.0497636, 0.036543, 0.0390687, 0.0333116, 0.0191607, 0.014331, 0.0346526, 0.0334894, 0.0303835,
0.0337696, 0.0226289, 0.0143451, 0.0286519, 0.0442745, 0.0329842, 0.0336011, 0.0226673, 0.0142058, 0.0204054, 0.0421557,
0.0318783, 0.0395467, 0.0307844, 0.0172333, 0.0182462, 0.0347106, 0.0332734, 0.0249783, 0.0198114, 0.0156317, 0.0154504,
0.0326826, 0.0365872, 0.0302313, 0.0337355, 0.0228829, 0.0156178, 0.0245702, 0.0349375, 1.44294, 0.487582, 0.281217,
0.129106, 0.0767487, 0.0655245, 0.237284, 0.158743, 0.050869, 0.0355709, 0.043961, 0.0492114, 0.0691462, 0.0738526,
0.0428588, 0.0384216, 0.0340364, 0.0475443, 0.0780251, 0.163954, 0.0546059, 0.0456951, 0.0303563, 0.0273428, 0.0329038,
0.0820468, 0.065237, 0.0680897, 0.0499921, 0.027629, 0.026662, 0.0671015, 0.0736497, 0.056863, 0.053277, 0.0294858,
0.0239902, 0.0405918, 0.0755732, 0.056632, 0.0567864, 0.0303834, 0.0232254, 0.029786, 0.0651051, 0.0463318, 0.068968,
0.0457036, 0.0259729, 0.0272711, 0.0529793, 0.0524145, 0.0311142, 0.0260773, 0.0225761, 0.0234896, 0.0441, 0.0550994,
0.0434422, 0.0506772, 0.0266426, 0.0201678, 0.0283458, 0.0513797, 1012, 27.3891, 26.1591, 33.9021;
```

B

The Ontology

In this dissertation, an ontology was developed to structure and govern the knowledge contained in the EKG. The approach taken to develop this ontology is described in Section 3.2.1, while the method used to evaluate it is described in Section 4.1.1. The end product is an OWL 2 ontology modelling wire bonders, their states, their sensors and the various observations made by these sensors. Figure B.1 depicts the ontology produced.

As we mention in Section 3.2.1, in developing our ontology we reuse several classes from existing ontologies. Only the classes colour coded in yellow in Figure B.1 were created specifically for this work. The classes colour coded in blue origin from the SOSA ontology while those in green from the DUL ontology. The *System* class is taken from the SSN ontology. The white boxes represent literal values.

It should be noted that some classes, such as the *SOSA:Procedure*, were not required but had to be included to preserve the rules of the ontologies we reused.

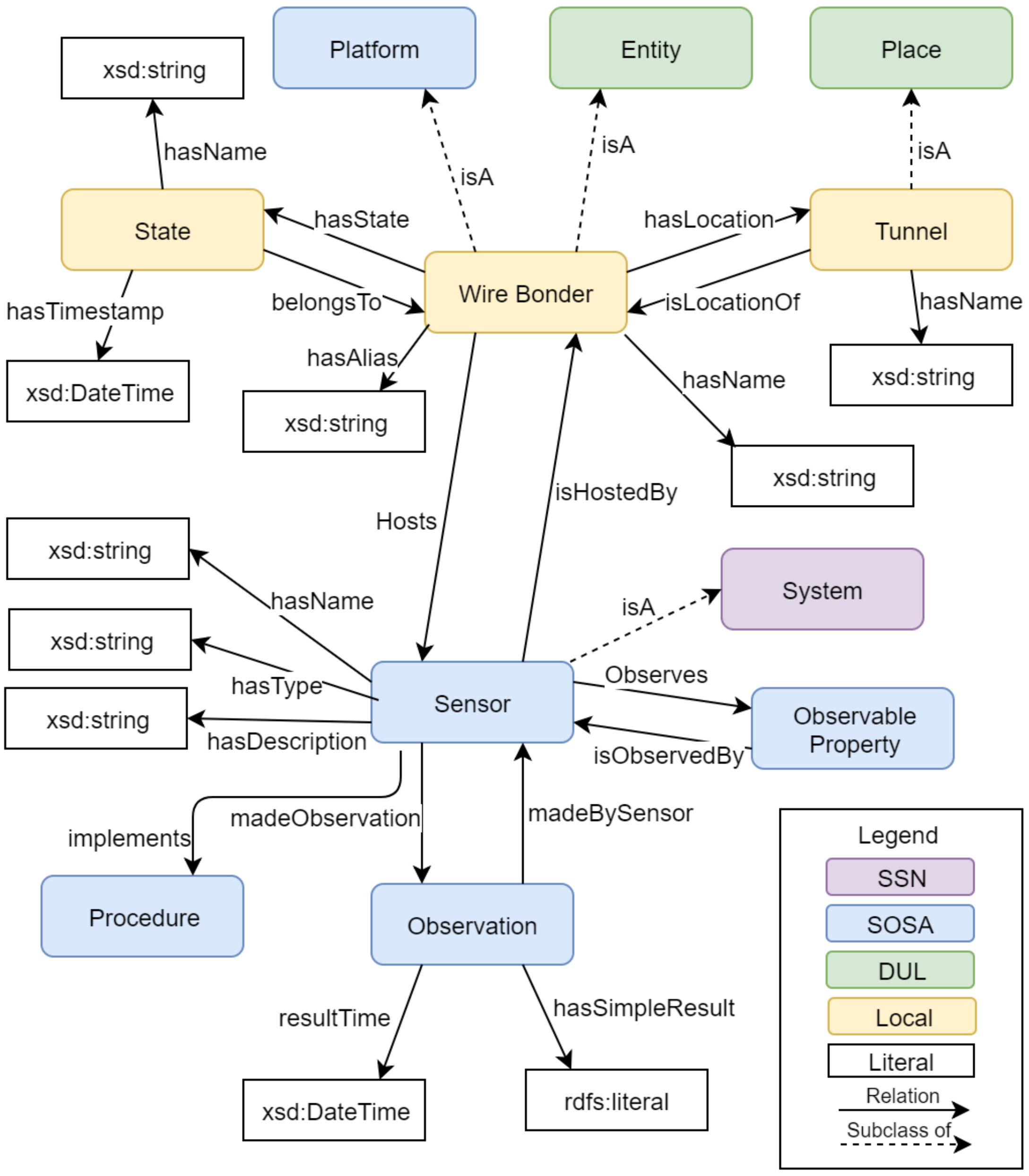


Figure B.1: A representation of the ontology produced in this dissertation.

C

Media Content

The enclosed pen drive contains the following folders:

1. **Data** - This folder contains the raw data as generated by the wire bonders. Inside are three sub-folders *ETS*, *Machine Logs* and *IIoT* that contain the three datasets described in Section 3.1.

2. **Data Preparation** - This folder contains various Jupyter Notebooks that were used to explore the data and to prepare the datasets to be used for the experiments.

3. **EKG** - This folder contains the RDF store built in this dissertation as well as the ontology that was developed (see Section 3.2).

4. **ETL** - The ETL scripts described in Section 3.2 can be found in this folder.

5. **Experiments** - This folder contains the source code and datasets used to conduct the experiments described in Section 3.3.3.

6. **PdM Dashboard** - This folder contains the source code of the dashboard developed as the GUI of the PdM framework, as described in Section 3.4.4.

References

Ido Amihai, Ralf Gitzel, Arzam Muzaffar Kotriwala, Diego Pareschi, Subanataranjan Subbiah, and Guruprasad Sosale. An industrial case study using vibration data and machine learning to predict asset health. In *2018 IEEE 20th Conference on Business Informatics (CBI)*, volume 1, pages 178–185. IEEE, 2018.

Marcia Baptista, Shankar Sankararaman, Ivo P de Medeiros, Cairo Nascimento Jr, Helmut Prendinger, and Elsa MP Henriques. Forecasting fault events for predictive maintenance using data-driven techniques and arma modeling. *Computers & Industrial Engineering*, 115:41–53, 2018.

Rimel Bendadouche, Catherine Roussey, Gil De Sousa, Jean-Pierre Chanet, and Kun Mean Hou. Extension of the semantic sensor network ontology for wireless sensor networks: The stimulus-wsnnode-communication pattern. 2012.

Christoph Bergmeir and José M Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, 2012.

Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

Tawfik Borgi, Adel Hidri, Benjamin Neef, and Mohamed Saber Naceur. Data analytics for predictive maintenance of industrial robots. In *2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, pages 412–417. IEEE, 2017.

Dario Bruneo and Fabrizio De Vita. On the use of lstm networks for predictive maintenance in smart industries. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 241–248. IEEE, 2019.

Matteo Calabrese, Martin Cimmino, Francesca Fiume, Martina Manfrin, Luca Romeo, Silvia Ceccacci, Marina Paolanti, Giuseppe Toscano, Giovanni Ciandrini, Alberto Carrotta, et al. Sophia: An event-based iot and machine learning architecture for predictive maintenance in industry 4.0. *Information*, 11(4):202, 2020.

Thyago P Carvalho, Fabrízzio AAMN Soares, Roberto Vita, Roberto da P Francisco, João P Basto, and Symone GS Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137:106024, 2019.

Roussey Catherine, Bernard Stephan, Andre Geraldine, and Boffety Daniel. Weather data publication on the lod using sosa/ssn ontology. *Semantic Web*, 2019.

Michael Compton, Holger Neuhaus, Kerry Taylor, and Khoi-Nguyen Tran. Reasoning about sensors and compositions. In *Proceedings of the 2nd International Conference on Semantic Sensor Networks-Volume 522*, pages 33–48. Citeseer, 2009.

Michael Compton, Payam Barnaghi, Luis Bermudez, RaúL GarcíA-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. The ssn ontology of the w3c semantic sensor network incubator group. *Web semantics: science, services and agents on the World Wide Web*, 17:25–32, 2012.

Estelle Deloux, Bruno Castanier, and Christophe Bérenguer. Predictive maintenance policy for a gradually deteriorating system subject to stress. *Reliability Engineering & System Safety*, 94(2):418–431, 2009.

K Efthymiou, N Papakostas, D Mourtzis, and G Chryssolouris. On a predictive maintenance platform for production systems. *Procedia CIRP*, 3:221–226, 2012.

Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48:1–4, 2016.

Saeid Farokhzad, Hojjat Ahmadi, Ali Jaefari, Mohammad Reza Asadi Asad Abad, and Mohammad Ranjbar Kohan. 897. artificial neural network based classification of faults in centrifugal water pump. *Journal of Vibroengineering*, 14(4), 2012.

Marta Fernandes, Alda Canito, Verónica Bolón-Canedo, Luís Conceição, Isabel Praça, and Goreti Marreiros. Data analysis and feature selection for predictive maintenance: A case-study in the metallurgic industry. *International journal of information management*, 46:252–262, 2019.

Mikhail Galkin, Sören Auer, María-Esther Vidal, and Simon Scerri. Enterprise knowledge graphs: A semantic approach for knowledge management in the next generation of enterprise information systems. 2017.

Kanika Gandhi, Bernard Schmidt, and Amos HC Ng. Towards data mining based decision support in manufacturing maintenance. *Procedia Cirp*, 72:261–265, 2018.

Martina Garofalo, Maria Angela Pellegrino, Abdulrahman Altabba, and Michael Cochez. Leveraging knowledge graph embedding techniques for industry 4.0 use cases. *arXiv preprint arXiv:1808.00434*, 2018.

Alasdair Gilchrist. *Industry 4.0: the industrial internet of things*. Springer, 2016.

Joanna Golebiowska, Rose Dieng-Kuntz, Olivier Corby, and Didier Mousseau. Building and exploiting ontologies for an automobile project memory. In *Proceedings of the 1st international conference on Knowledge capture*, pages 52–59, 2001.

Asunción Gómez-Pérez, Natalia Juristo, and Juan Pazos. Evaluation and assessment of knowledge sharing technology. *Towards very large knowledge bases*, pages 289–296, 1995.

Jose Manuel Gomez-Perez, Jeff Z Pan, Guido Vetere, and Honghan Wu. Enterprise knowledge graph: An introduction. In *Exploiting linked data and knowledge graphs in large organisations*, pages 1–14. Springer, 2017.

Alasdair JG Gray, Raúl García-Castro, Kostis Kyzirakos, Manos Karpathiotakis, Jean-Paul Calbimonte, Kevin Page, Jason Sadler, Alex Frazer, Ixent Galpin, Alvaro AA Fernandes, et al. A semantically enabled service architecture for mashups over streaming and stored data. In *Extended Semantic Web Conference*, pages 300–314. Springer, 2011.

Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

Hashem M Hashemian. State-of-the-art predictive maintenance techniques. *IEEE Transactions on Instrumentation and measurement*, 60(1):226–236, 2010.

Mario Hermann, Tobias Pentek, and Boris Otto. Design principles for industrie 4.0 scenarios. In *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, pages 3928–3937. IEEE, 2016.

Krzysztof Janowicz, Armin Haller, Simon JD Cox, Danh Le Phuoc, and Maxime Lefrançois. Sosa: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 56: 1–10, 2019.

Jonas Jetschni and Vera G Meister. Schema engineering for enterprise knowledge graphs: A reflecting survey and case study. In *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 271–277. IEEE, 2017.

Ameeth Kanawaday and Aditya Sane. Machine learning for predictive maintenance of industrial machines using iot sensor data. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 87–90. IEEE, 2017.

Bahador Khaleghi, Alaa Khamis, Fakhreddine O Karray, and Saiedeh N Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information fusion*, 14(1):28–44, 2013.

Evgeny Kharlamov, Theofilos Mailis, Gulnar Mehdi, Christian Neuenstadt, Özgür Özçep, Mikhail Roshchin, Nina Solomakhina, Ahmet Soylu, Christoforos Svingos, Sebastian Brandt, et al. Semantic access to streaming and static data at siemens. *Journal of Web Semantics*, 44:54–74, 2017.

Jakob Kinghorst, Omid Geramifard, Ming Luo, Hian-Leng Chan, Khoo Yong, Jens Folmer, Minjie Zou, and Birgit Vogel-Heuser. Hidden markov model-based predictive maintenance in semiconductor manufacturing: A genetic algorithm approach. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1260–1267. IEEE, 2017.

F Klingert, G Roeder, M Schellenberger, A Bauer, L Frey, M Brueggemann, and K Pressel. Condition-based maintenance of mechanical setup in aluminum wire bonding equipment by data mining. In *2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, pages 72–77. IEEE, 2017.

Benjamin Klotz, Raphaël Troncy, Daniel Wilms, and Christian Bonnet. Vsso: The vehicle signal and attribute ontology. In *SSN@ ISWC*, pages 56–63, 2018.

Dmitry Kovalev, Ivan Shanin, Sergey Stupnikov, and Victor Zakharov. Data mining methods and techniques for fault detection and predictive maintenance in housing and utility infrastructure. In *2018 International Conference on Engineering Technologies and Computer Science (EnT)*, pages 47–52. IEEE, 2018.

Florent Laroche, Mohamed Anis Dhuieb, Farouk Belkadi, and Alain Bernard. Accessing enterprise knowledge: a context-based approach. *CIRP Annals*, 65(1):189–192, 2016.

Anselm Lorenzoni and Michael Kempf. Degradation processes modelled with dynamic bayesian networks. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 1694–1699. IEEE, 2015.

Gabriela Medina-Oliva, Alexandre Voisin, Maxime Monnin, and Jean-Baptiste Leger. Predictive diagnosis based on a fleet-wide ontology approach. *Knowledge-Based Systems*, 68:40–57, 2014.

R Keith Mobley. *An introduction to predictive maintenance*. Elsevier, 2002.

Heiko Müller, Liliana Cabral, Ahsan Morshed, and Yanfeng Shu. From restful to sparql: A case study on generating semantic sensor data. In *SSN@ ISWC*, pages 51–66, 2013.

Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

Leo Obrst, Werner Ceusters, Inderjeet Mani, Steve Ray, and Barry Smith. The evaluation of ontologies. In *Semantic web*, pages 139–158. Springer, 2007.

Irfan M Ovacik and Reha Uzsoy. *Decomposition methods for complex factory scheduling problems*. Springer Science & Business Media, 2012.

Jeff Z Pan, Guido Vetere, Jose Manuel Gomez-Perez, and Honghan Wu. *Exploiting Linked Data and Knowledge Graphs in Large Organisations*. Springer, 2017.

Marina Paolanti, Luca Romeo, Andrea Felicetti, Adriano Mancini, Emanuele Frontoni, and Jelena Loncarski. Machine learning approach for predictive maintenance in industry 4.0. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6. IEEE, 2018.

Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

Niklas Petersen, Michael Galkin, Christoph Lange, Steffen Lohmann, and Sören Auer. Monitoring and automating factories using semantic models. In *Joint International Semantic Technology Conference*, pages 315–330. Springer, 2016.

Niklas Petersen, Lavdim Halilaj, Irlán Grangel-González, Steffen Lohmann, Christoph Lange, and Sören Auer. Realizing an rdf-based information model for a manufacturing company–a case study. In *International Semantic Web Conference*, pages 350–366. Springer, 2017.

Shelley Powers. *Practical RDF: solving problems with the resource description framework*. " O'Reilly Media, Inc.", 2003.

Joe Raad and Christophe Cruz. A survey on ontology evaluation methods. In *7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015)*, 2015.

Yongyi Ran, Xin Zhou, Pengfeng Lin, Yonggang Wen, and Ruilong Deng. A survey of predictive maintenance: Systems, purposes and approaches. *arXiv preprint arXiv:1912.07383*, 2019.

Chinmay Rao, Asok Ray, Soumik Sarkar, and Murat Yasar. Review and comparative evaluation of symbolic dynamic filtering for detection of anomaly patterns. *Signal, Image and Video Processing*, 3(2):101–114, 2009.

Martin Ringsquandl, Evgeny Kharlamov, Daria Stepanova, Steffen Lamparter, Raffaello Lepratti, Ian Horrocks, and Peer Kröger. On event-driven knowledge graph completion in digital factories. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 1676–1681. IEEE, 2017a.

Martin Ringsquandl, Steffen Lamparter, Raffaello Lepratti, and Peer Kröger. Knowledge fusion of manufacturing operations data using representation learning. In *IFIP International Conference on Advances in Production Management Systems*, pages 302–310. Springer, 2017b.

DJ Russomanno and JC Goodwin. Ontosensor: An ontology for sensor network application development, deployment, and management. *Handbook of Wireless Mesh and Sensor Networking*, 2008.

Nestor Rychtyckyj, Venkatesh Raman, Baskaran Sankaranarayanan, P Sreenivasa Kumar, and Deepak Khemani. Ontology reengineering: A case study from the automotive industry. *AI Magazine*, 38(1), 2017.

Stefan Schabus and Johannes Scholz. Semantically annotated manufacturing data to support decision making in industry 4.0: A use-case driven approach. In *Data Science–Analytics and Applications*, pages 97–102. Springer, 2017.

Craig Schlenoff, Tsai Hong, Connie Liu, Roger Eastman, and Sebti Foufou. A literature review of sensor ontologies for manufacturing applications. In *2013 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pages 96–101. IEEE, 2013.

Bernard Schmidt. *Toward Predictive Maintenance in a Cloud Manufacturing Environment - A population-wide approach*. PhD thesis, 2018.

Bernard Schmidt and Lihui Wang. Cloud-enhanced predictive maintenance. *The International Journal of Advanced Manufacturing Technology*, 99(1-4):5–13, 2018.

Bernard Schmidt, Lihui Wang, and Diego Galar. Semantic framework for predictive maintenance in a cloud environment. In *10th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME'16, Ischia, Italy, 20-22 July 2016*, volume 62, pages 583–588. Elsevier, 2017.

M Schuettler and T Stieglitz. Microassembly and micropackaging of implantable systems. In *Implantable Sensor Systems for Medical Applications*, pages 108–149. Elsevier, 2013.

OR Seryasat, F Honarvar, Abolfazl Rahmani, et al. Multi-fault diagnosis of ball bearing using fft, wavelet energy entropy mean and root mean square (rms). In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 4295–4299. IEEE, 2010.

Jong-Ho Shin and Hong-Bae Jun. On condition based maintenance policy. *Journal of Computational Design and Engineering*, 2(2):119–127, 2015.

Ruben Sipos, Dmitriy Fradkin, Fabian Moerchen, and Zhuang Wang. Log-based predictive maintenance. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1867–1876, 2014.

Symone Gomes Soares. *Ensemble learning methodologies for soft sensor development in industrial processes*. PhD thesis, 2015.

Dezhao Song, Frank Schilder, Shai Hertz, Giuseppe Saltini, Charese Smiley, Phani Nivarthi, Oren Hazai, Dudi Landau, Mike Zaharkin, Tom Zielund, et al. Building and querying an enterprise knowledge graph. *IEEE Transactions on Services Computing*, 2017.

Lukas Spendla, Michal Kebisek, Pavol Tanuska, and Lukas Hrcka. Concept of predictive maintenance of production systems in accordance with industry 4.0. In *2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, 2017.

G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi. Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3):812–820, June 2015.

Jinn-Tsong Tsai, Cheng-Chung Chang, Wen-Ping Chen, and Jyh-Horng Chou. Optimal parameter design for ic wire bonding process by using fuzzy logic and taguchi method. *IEEE Access*, 4:3034–3045, 2016.

Paolo Umiliacchi, David Lane, Felice Romano, and A SpA. Predictive maintenance of railway subsystems using an ontology based modelling approach. In *Proceedings of 9th world conference on railway research, May*, pages 22–26, 2011.

Michael Uschold, Michael Gruninger, et al. Ontologies: Principles, methods and applications. *TECHNICAL REPORT-UNIVERSITY OF EDINBURGH ARTIFICIAL INTELLIGENCE APPLICATIONS INSTITUTE AIAI TR*, 1996.

Alexandre Voisin, Gabriela Medina-Oliva, Maxime Monnin, Jean-Baptiste Leger, and Benoit Iung. Fleet-wide diagnostic and prognostic assessment. 2013.

Xiang Wang, Xiaoming Zhang, and Mei Li. A survey on semantic sensor web: sensor ontology, mapping and query. *International Journal of u-and e-Service, Science and Technology*, 8(10):325–342, 2015.

Chunsheng Yang, Qiangqiang Chen, Yubin Yang, and Nan Jiang. Developing predictive models for time to failure estimation. In *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 133–138. IEEE, 2016.

Shuai Zhang, Wenxi Zeng, I-Ling Yen, and Farokh B Bastani. Semantically enhanced time series databases in iot-edge-cloud infrastructure. In *2019 IEEE 19th international symposium on high assurance systems engineering (HASE)*, pages 25–32. IEEE, 2019.