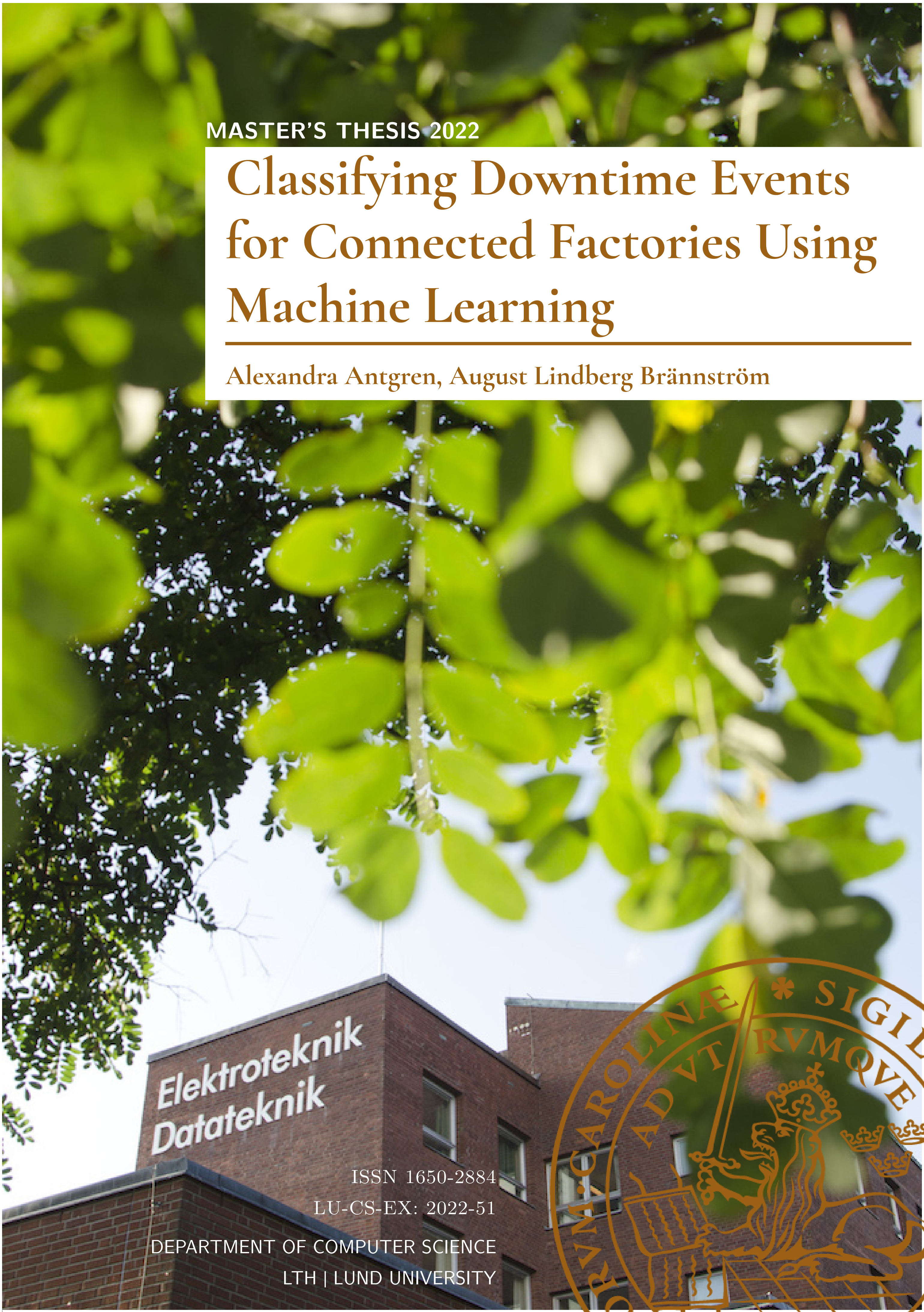


MASTER'S THESIS 2022

Classifying Downtime Events for Connected Factories Using Machine Learning

Alexandra Antgren, August Lindberg Brännström

ISSN 1650-2884
LU-CS-EX: 2022-51

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2022-51

Classifying Downtime Events for Connected Factories Using Machine Learning

Alexandra Antgren, August Lindberg Brännström

Classifying Downtime Events for Connected Factories Using Machine Learning

Alexandra Antgren
al5046an-s@student.lu.se

August Lindberg Brännström
august.lindberg.brannstrom@gmail.com

August 30, 2022

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Markus Borg, markus.borg@cs.lth.se
Sjoerd Dost, sjoerd.dost@northvolt.com

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

Machine downtime is an important subject in manufacturing because of its connection to production rate and business profit. The causes of machine downtimes are diverse and understanding the cause is critical to have actionable information, identify areas of improvement and set specific targets. This master's thesis explores the possibility of using machine learning to classify downtime events for machines in connected factories. In this study we use data from a Swedish Lithium-ion battery producer. We collected downtime data from one machine in one facility and combined it with data on active alarms from this same machine. The data was analysed, cleaned and features were selected for modeling. We implemented five baselines: three naive ones and two simple supervised learning models (Naive Bayes and Decision tree) and two ensemble models (Random forest and XGBoost). For correctly classifying a downtime event with one out of 17 categories, the Random forest model performed the best with an accuracy of 38.9%. When giving a Top-5 suggestion of the top five most probable categories for a downtime event, the Random forest model was the best with an accuracy of 82.3%. The results show that alarms being active during a downtime has a correlation with the reason of the machine being down. The findings indicate that machine learning can be used to determine the cause of downtime events but that more data is needed to get a higher accuracy.

Acknowledgements

We would like to thank...

Our LTH supervisor Markus Borg for providing significant support in every aspect from start to finish. Always with great optimism and insights.

Our Northvolt supervisor Sjoerd Dost for giving us the freedom to explore and tailor our master's thesis to a high degree. Asking critical questions making us dig deeper into the unknown and discover new insights.

Our colleague from the data team Lef Filippakis, who answered to our every needs in terms of data requests as well as elaborate discussions on how to restructure data and work with machine learning. Diogo Vasconcelos and Sebastian Callh from Northvolt, as well as Erik Hellsten from LTH, thank you all for providing feedback and support regarding exploratory data analysis, modeling strategies and hyperparameter tuning guidelines.

Contents

Chapter 1
Introduction

This chapter gives an introduction to the background and goals of this master's thesis project. The scope of the project is presented, along with assumptions and limitations. Previous research and it's relation to this master's thesis is established.

1.1 Background

This master's thesis project was performed in collaboration with the Swedish company Northvolt. Northvolt was founded in 2016 in Stockholm and aims to become Europe's leading supplier of sustainable, high-quality battery cells and systems. The company produces lithium-ion cells and has more than 3,000 employees and sites in Sweden, Poland, Germany, Norway and USA.

Northvolt wants to bring Industry 4.0 to batteries and set a new standard in the digitalization of battery assets. The company is harnessing data from every process in it's manufacturing and tagging battery materials and components with metadata, so that it later can be used for analysis. Northvolt has more than 150 employees working specifically with digitalization and expects that its adoption of it will bring about a significant competitive edge.

1.2 Problem description

Northvolt has machines in their production facilities that can either be in a producing state or in a non producing state, referred to as "down" in this thesis. A machine that is down can be so either due to a planned or unplanned reason. A planned reason could be "scheduled break" and an unplanned one "machine breakdown". Northvolt is recording and storing data on when their machines are down. It is vital for the company to understand the causes behind this to increase productivity.

For a machine called stacker situated in one of Northvolt's production facilities, operators (employees who operates equipment or a machine as used in production) are manually adding a category to why the stacker is down. Northvolt wants to examine the possibility of auto categorizing these so called 'downtime events' using machine learning. This master's thesis will therefore investigate the potential of using machine learning to automatically classify and identify the reason for the stacker not being in a producing state. Furthermore, there are business incentives behind successful autocategorization. It allows operators to save valuable time since they don't have to manually classify downtimes and it enables storing of valuable data for further analysis. Analysis using the stored data can in turn lead to less downtime and increased productivity. This could help Northvolt remain competitive.

1.3 Research question

The goal of this thesis is to contribute to Northvolt's digitalization journey. Specifically, we explore machine learning (ML) in the context of availability analysis and management. We aim at helping Northvolt utilize ML to understand why a machine is unavailable.

RQ1 How can Northvolt use machine learning to understand why a machine is unavailable?

RQ2 How can a classifier autocategorize downtime events from a machine given alarm and downtime data?

1.3.1 Scope

This thesis is a proof-of-concept and explores if autocategorization of downtime events can be done. The focus is exclusively on data from a machine called stacker in one of Northvolt's production facilities. The stacker as a machine is chosen as a good proof-of-concept since it is a very complex machine, meaning that if autocategorization works for the stacker, it could achieve better results for other less complex machines. The scope is to use historical downtime and alarm data from Northvolt. A fully working application to be used in production is not in the pipeline. Additionally it is not in the scope to predict the category for planned downtimes since these are planned gaps in the production schedule.

1.3.2 Assumption

The category which the operator has selected to classify each downtime event with is deemed to be accurate.

1.3.3 Limitations

To autocategorize downtime events we will use alarm data sent from a machine. More details on alarms is presented in 3.2. A machine is an individual unit that may consist of several modules. A limitation is that alarm data is sent for a unit whereas downtime events are sent per module, see more details of this subject in 2.2. This means that we have more granular information about each downtime event than of alarms. This adds a level of complexity since we won't know if a certain alarm received on unit level has anything to do with a particular module having a downtime. Another limitation is that the factory which we use data from is partially used as a testing and training facility and partly production facility. This may result in discrepancies in the data.

1.4 Related work

The use of Internet of things (IoT) techniques in industrial applications offers a new approach for factories built now and in the future. [27]. Embedding Industry 4.0 concepts into the manufacturing sector enables the interconnection of anything, anywhere and at any time to improve the the productivity and efficiency [27]. Northvolt aims to embed Industry 4.0 into their battery gigafactories and build up a digital infrastructure that can enhance production and battery performance.

In *Advances in Machine Learning Detecting Changeover Processes in Cyber Physical Production Systems* the authors used machine learning to determine automatically if a machine was either running or not running due to changeover being performed. The authors argued that with increasing data from machines due to the interconnection of them, traditional methods and analytical models have reached their limitations and they suggest that machine learning could be a good means of evaluation the data. Decision Trees and Ensemble Classifiers was implemented in the research and showed good results for the classification problem. Their type of Decision Tree algorithm achieved an overall accuracy of 92.8% [5]. In our thesis we use alarm data from a machine whereas in the paper sensor data was used from five sensors from their machine. Also, their classification problem only had the two target variables "changeover" or "production", whereas we have 17 different ones (53 in the initial dataset). This introduces a lot more complexity for the classifiers implemented in this thesis and we therefore expect a lower accuracy. In their work they also had 36,844 datapoints recorded and labelled whereas we in our final dataset have 1735, so approximately 95% less data.

In *Downtime Data Classification Using Naïve Bayes Algorithm on 2008 ESEC Engine* the authors show the potential of using a Naive Bayes algorithm to classify diagnostic history data for a certain type of machine [14]. The results shown are especially interesting for this thesis since it clearly shows the potential of using machine learning to understand why a machine is unavailable. However, the dataset as a whole is not the same as in this thesis and they have only two target variables to predict whereas we have 17. There are several other implementations in different areas using Naive Bayes classifier such as [3] and [17], [9] where it has shown good performance. In this thesis, Naive Bayes classifier will therefore be used as a baseline model to compare our implemented models against.

Another classification model widely used in many research settings is the Random forest model [24]. In *Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?* 179 different classifiers and 121 datasets were evaluated and the authors concluded that the Random forest versions were most likely to perform best for any dataset [6]. A major benefit of using Random forest for prediction modeling is the ability to handle datasets with a large number of predictor variables [24]. Since our dataset has many predictor variables Random forest seem like a good model to handle this with.

Combining Random forest with XGBoost is used in a paper for fault detection in wind turbines [28]. In the proposed design, Random forest is first used to rank the features by importance, and then the XGBoost model trains the ensemble classifier for each specific fault. In this thesis, both Random forest and XGBoost will be evaluated but not used in this combination. The authors showed that the proposed design was robust and showed strong anti-overfitting ability. [28]

1.5 Contribution

This thesis aims to give knowledge and methods into how historical production data can be utilized to categorize downtime events. It is a a proof-of-concept to see if machine learning can be used to determine the cause of downtime events for machines within the battery industry and indicate whether or not it can be implemented and rolled out for the remaining production equipment. It will also give information regarding what features make good predictors for downtime events.

Chapter 2
Theory

This chapter presents necessary theory in order to comprehend the project approach and interpret the results. Overall Equipment Efficiecy is introduced and downtime as a concept is defined. The process equipment is briefly presented and the current Downtime Management System is introduced. A selection of Machine learning models along with their corresponding evaluation metrics are presented.

2.1 Overall Equipment Efficiency

Productivity improvement is one of the biggest challenges for manufacturing companies in order to remain competitive in a global market [15]. Companies need to improve and optimize their productivity [10]. A well known way of measuring the effectiveness of a production facility is using the Overall Equipment Effiency (OEE) metric. OEE was first described by Nakajima in the 1980s and is used to measure the effectiveness and evaluate manufacturing operations across the industry [10]. OEE identifies the percentage of manufacturing time that is truly productive using three underlying factors: availability, performance and quality. OEE is calculated by multiplying three parameters:

$$\text{OEE} = \text{Availability} \times \text{Performance} \times \text{Quality}$$

Availability is calculated with:

$$\text{Availability} = \frac{\text{Total Hours Planned - Lost Time}}{\text{Total Hours Planned}}$$

Performance is calculated with:

$$\text{Performance} = \frac{\text{Actual Machine Speed}}{\text{Designed Machine Speed}}$$

Quality is calculated with:

$$\text{Quality} = \frac{\text{Number of Good Parts}}{\text{Total Parts Made}}$$

An OEE score of 100% means that the company is manufacturing only good parts, as fast as possible, with no downtime. The metric helps identify areas of improvement and set specific targets [21].

2.2 Stacker

A stacker is a machine used in the battery manufacturing process. It is one of the most complicated process steps in the battery manufacturing process at Northvolt, so high quality downtime data is of utmost importance. Northvolt has together with the supplier of the machine created a structured modular breakdown of the parts of the machine that performs a specific task. It is divided into six different modules where each module has it's own unique identifier.

Downtime data is gathered on module-level. If a module in the stacker is down that does not necessarily mean that the entire stacker is down. Alarm data for the stacker is however received on unit level, see 2.1 for an overview of how the alarms and downtime signaling is done. This causes a level of complexity when interpreting what alarm is related to which downtime event, more on this subject underneath and how the problem is mitigated under 3.2.4. The data used in this master's thesis comes from a stacker located in one of Northvolt's production facilities.

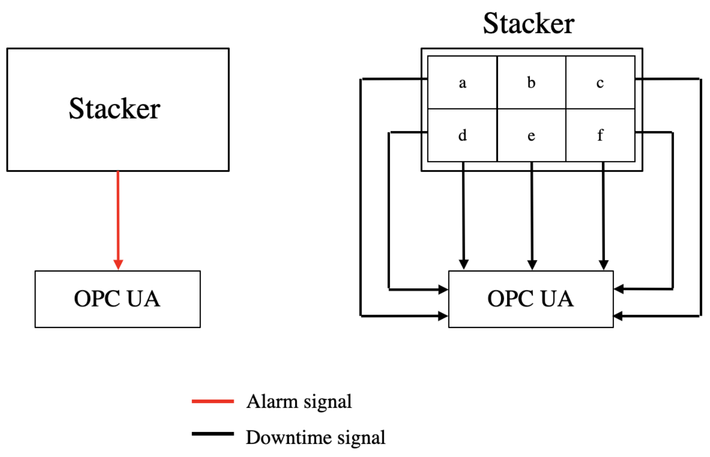


Figure 2.1: Unit and module levels for a stacker. To the left one may see how the alarm signal is handled on unit level, whereas the right side of the figure shows how the downtime signals are handled on module levels a-f. For details on what the functions of OPC UA is, see 2.3.1

2.3 Downtime

Machine downtime is an important subject in manufacturing because of its connection to production rate and business profit [16]. Reducing downtime in production processes is vital since it serves the purpose of maximising machine uptime, increasing the productivity of the machine. Every lost minute of operation can translate into significant cost to a firm due to increased lead times and a negative impact on customer satisfaction [1]. The causes of machine downtimes are diverse and differ from one machine to another. Causes can include problems with the actual machines such as machine breakdowns or be due to other factors such as a cleaning, a machine operator being unavailable or change of material. Understanding the cause behind a downtime is critical to have actionable information.

2.3.1 Downtime Management System at Northvolt

Norhtvolt has an internally built Downtime Management System. The system enables automatic recording of downtime events for a machine. It's purpose is to identify, categorize and manage downtime events. Tracking and storing of the data makes it possible to accumulate downtime on units, process areas, and factories and see statistics about the most contributing factors of downtime bottom-up. This gives Northvolt a standardized way of measuring downtime across all sites and units enabling comparison.

The structure of the Downtime Management System is presented in 2.2. The flow of information goes from labels 1-5 as seen in the figure.

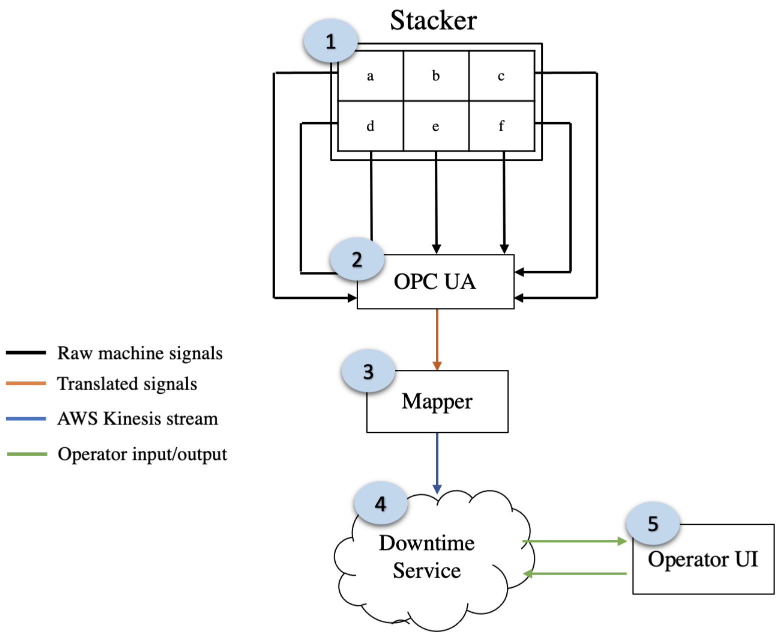
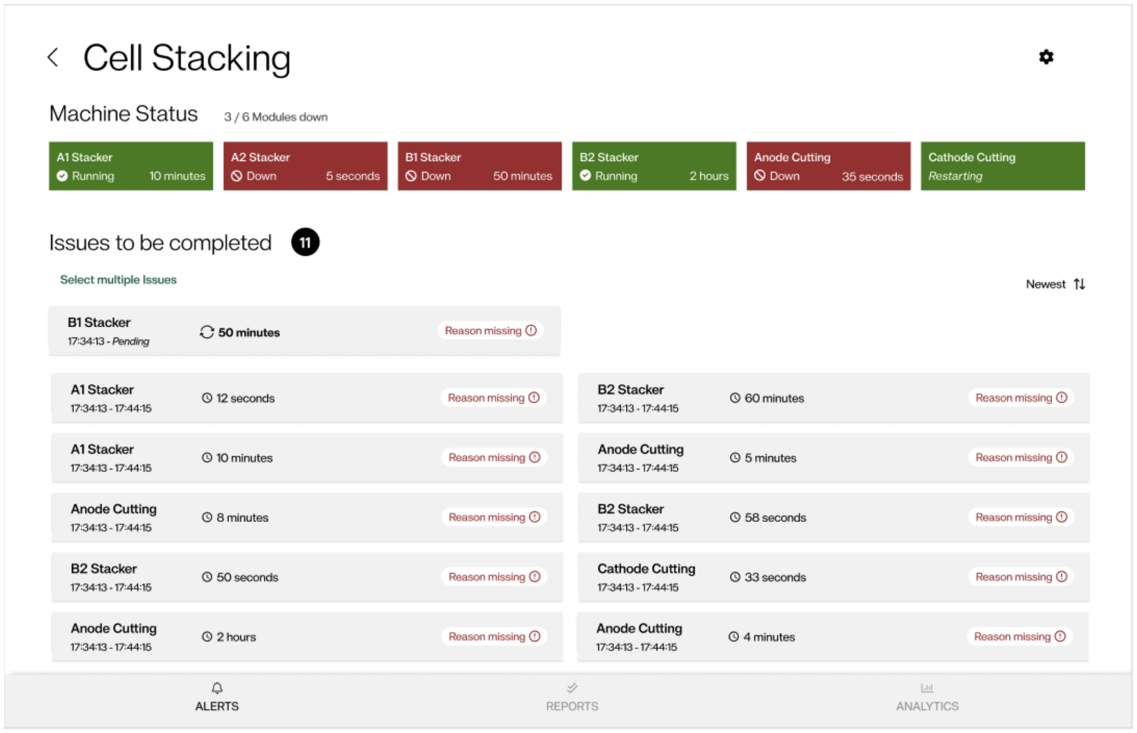


Figure 2.2: Overview of the Downtime Management System infrastructure

- **1 – Stacker**: One of the modules a-f registers that a stop of production has occurred and sends a signal to the OPC UA server.
- **2 – OPC Unified Architecture (OPC UA)**: OPC UA is an interoperability standard for secure and reliable exchange of data from sensors to cloud applications. [7] The OPC UA server calls data from the machine on a refresh cycle.
- **3 – Mapper**: When signal conditions from the OPC UA server indicate a stoppage, an event is created in the mapper that represents that stop. The mapper runs on factory machine gateways and is the link between the machine and the internal cloud services at Northvolt. The internal cloud services, such as the Downtime Service can then read and process data from the mapper.
- **4 – Downtime Service**: Events from the mapper are streamed via Amazon Kinesis Data Streams and are processed and captured by Northvolt's Downtime Service. Amazon Kinesis Data Stream is a serverless streaming data service. [2] When the Downtime Service receives an event from the mapper that indicates a stoppage for a machine, the service creates a downtime event with information from the event and stores it in a non-relational database called DynamoDB. When the Downtime Service receives a corresponding event indicating that the machine is running again, a timestamp is added to the downtime event and duration is calculated.
- **5 – Operator User Interface (UI)**: Once the Downtime Service has registered a downtime event a machine operator will be able to view this in an Operator User Interface (UI). The Operator UI gives operators, the users, an opportunity to view downtime events in real-time and manually categorize the reason to why a specific downtime occurred. The user categorizes by choosing from a list of predefined categories and can also add comments. See Operator UI in 2.3.


Cell Stacking

Machine Status 3 / 6 Modules down

A1 Stacker	A2 Stacker	B1 Stacker	B2 Stacker	Anode Cutting	Cathode Cutting
Running 10 minutes	Down 5 seconds	Down 50 minutes	Running 2 hours	Down 35 seconds	*Restarting*

Issues to be completed 11

Select multiple Issues

Newest

B1 Stacker 17:34:13 - *Pending* — 50 minutes — Reason missing

A1 Stacker 17:34:13 - 17:44:15	12 seconds	Reason missing	B2 Stacker 17:34:13 - 17:44:15	60 minutes	Reason missing
A1 Stacker 17:34:13 - 17:44:15	10 minutes	Reason missing	Anode Cutting 17:34:13 - 17:44:15	5 minutes	Reason missing
Anode Cutting 17:34:13 - 17:44:15	8 minutes	Reason missing	B2 Stacker 17:34:13 - 17:44:15	58 seconds	Reason missing
B2 Stacker 17:34:13 - 17:44:15	50 seconds	Reason missing	Cathode Cutting 17:34:13 - 17:44:15	33 seconds	Reason missing
Anode Cutting 17:34:13 - 17:44:15	2 hours	Reason missing	Anode Cutting 17:34:13 - 17:44:15	4 minutes	Reason missing

Figure 2.3: Northvolt's Downtime Operator User Interface, the view for when a user is opted to pick a downtime reason for a specific machine that most correctly represents the actual stop. (Fictive data due to confidentiality reasons)

Downtime categories

There are general and machine specific categories. The general ones are available for all machines whereas the machine specific ones can only be chosen as a category for that machine. An example of a machine specific category for the stacker is that a certain knife needs to be changed.

There are so called 'admin' users for the Downtime Management System that are the ones with knowledge of the machines and responsible of configuring and adding downtime categories that should be available for that machine.

2.4 Machine Learning

Machine Learning (ML) is a subsection within Artificial Intelligence (AI) which is used to develop algorithms that, based on a dataset, natively can learn and evolve it's ability to create predictions and decisions. The three key areas of ML can be divided into supervised learning, unsupervised learning and reinforcement learning. [11] In this master's thesis supervised learning models will be implemented. For supervised learning models data is provided in so called input-output pairs. It is up to the model to identify what input values that entails which output value. Data is divided into input values, referred to in this thesis as features and output values, referred to in this thesis as target variables. The data is then split into training and testing datasets, by which the training data is used to build the model and the testing data is used to verify how the model performs. The data used in this thesis may easily be divided into input-output-pairs and therefore supervised learning models are considered. During the training phase of a machine learning algorithm the quality of the so called fit is of great importance. Underfitting is when a model is unable to find the pattern of a training set's inputs and relation with its outputs. An underfitted model will perform poorly on new unseen data points as well. Overfitting is when the model is unable to find underlaying pattern between a training set's inputs and relation with its outputs, instead it takes the noise into account which makes the model to perform with a high score on the training data, but poorly on new unseen data.

2.4.1 Simple supervised learning models

Naive Bayes

Naive Bayes classifier generates predictions based on Bayes Theorem 2.1.

$$P(A|\mathrm{B}) = P(A)\frac{P(\mathrm{B}|A)}{P(\mathrm{B})} \tag{2.1}$$

It states that we can identify the probability of A happening given that B has occurred.

This can be rewritten with X for the n number of features, such as the alarms in our case, and y for the classes, which correspond to downtime categories in our case. The formula then looks like 2.2.

$$P(y|\mathrm{X}) = P(y)\frac{P(\mathrm{X}|y)}{P(\mathrm{X})} \qquad X = (x_1, x_2, x_3, ..., x_n) \tag{2.2}$$

As seen in 2.2 this is a rough predictor which does not have any means of giving different features different weights. This is a probabilistic model which takes the probability distribution of the inputs to see if there is a pattern towards the outputs. Any correlation of features will not be detected with this model.

Decision tree

A decision tree is a simple machine learning algorithm which utilise a tree-like structure for classification. The goal of the model is to make a decision and predict the value of a target variable by learning simple decision rules deduced from the data features. It is used in decision analysis and helps map out different courses of action, as well as their potential outcomes. The decision tree consists of a root node, decision nodes and terminal nodes (sometimes referred to as leaves) which are connected via branches. [12]

The root node is the beginning of the tree and represents the entire population being analysed. From that point, the population is divided into sub-groups. Each decision node has an attribute that is tested and depending on the decision being taken, a branch will be chosen from that to either a succeeding decision node, or a terminal node. This process is then repeated until a terminal node, representing a class label, is reached. The decision nodes are decision branches with each one representing a possible alternative from that point. The alternatives must be mutually exclusive meaning that if one branch is chosen, the others cannot be chosen. The alternatives must also be collectively exhaustive meaning that all possible alternatives must be included.

How to split at each node is made according to a metric called purity. If all of the data in a node belongs to a single class it is 100% pure and if it is split evenly into two subsets it is 100% impure. This metric is used by the model to optimize the splits. There are two main ways of determining the purity of a split, entropy and gini impurity. They are both used to calculate how the proportions of different classes stand after a split. For a decision tree used for binary classification entropy varies between 0 and 1. An entropy of 0 is equivalent to a node with only one class, whereas an entropy of 1 is equivalent to a node which contain equal part of each two classes. Entropy is mathematically defined as H in: 2.3

$$H = \sum_{n=i}^{c} P_i log_2(p_i) \tag{2.3}$$

Where c is the number of classes and P_i denotes the probability that one randomly chosen sample of label i is retrieved from the set.

Entropy may be used to calculate information gain. By using information gain a model can evaluate different potential splits of a node to minimize entropy for future nodes. Information gain can simply be expressed as how much entropy that is removed from each split. See information gain, denoted IG in 2.4.

$$IG(S, A) = H_S - \sum_{v \in values(A)} \frac{(S_v)}{(S)} H(S_v) \tag{2.4}$$

H_s denotes entropy of the subset S, before splitting on attribute A. S_v Refers to the subset of S after splitting. So by maximizing IG one lowers H between splits which in turn will yield a higher level of purity in the split. This results in the terminal node being reached with fewer steps and hence the predictive power will be greater and level of overfitting lower.

Decision trees provide a clear indication of which fields are most important for prediction or classification. Decision trees are prone to overfitting once the model becomes more complex and the tree is deepened. To combat this, there exist a technique called decision tree pruning which eliminates irrelevant nodes. Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using multiple decision trees within an ensemble, see 2.4.3.

2.4.2 Ensemble techniques

Ensemble techniques aim to improve the overall performance and achieve a higher level of prediction quality by combining multiple simple models and their outputs into a single prediction. This can assist in preventing overfitting and reducing the bias of a model. [8]. In this section three ensemble techniques are introduced which are used by the ensemble models presented in section 2.4.3.

Bagging

Bagging is a method with the goal to reduce the variance within the dataset. In bagging, the training data is randomly split into multiple sub-samples of the original training set. The split is done using sampling with replacement, meaning that each data point randomly selected from the original training set is returned to that set before the next data point is selected. Sampling with replacement ensures that the probability of choosing any specific data point remains constant. After the split, each of the sub-samples are provided to the models to be trained on. Once the models are trained they are tested with data from the testing set to create their individual predictions. The final prediction from each individual model is then aggregated. For classification models the majority result is the prediction that is outputted. For regression models the mean value is used instead. In bagging, the [13]

Boosting

Boosting is a method for improving the predictions of any given learning algorithm. In boosting, models are trained sequentially, meaning that the following model is trained on data which the previous model had troubles to predict. The idea is that each model tries to compensate for the weaknesses of the previous one in order to improve the overall accuracy of all models. The method iteratively have models complement each other to create good predictions.

Random subspace

Random subspace method, also called feature bagging, is a method which trains models in an ensemble on random samples of features instead of the entire feature set. The different models will have access to a different set of features and hence provide a level of variability which can provide a higher level of quality predictions once combined. The reason for this is that if one or a few features are deemed strong predictors then the individual models may be strongly correlated and hence they will overall predict the same outcome to a higher degree.

2.4.3 Ensemble models

Random forest

Random forest is a supervised machine learning model made up of a collection of decision trees to circumvent some of the downsides of decision trees. Once the Random forest model is trained and exposed to the testing set each decision tree will make its own prediction and a majority vote is made to determine the most probable prediction. The Random forest model combines the output of the decision trees to reach a single result. For classification models, the prediction outputted by most of the decision trees is the result. For regression models, the predictions from the individual decision trees are averaged and used as the result. The Random forest model uses both bagging, described in 2.4.2 and random subspace, described in 2.4.2 to create an uncorrelated forest of decision trees. Random subspace is used to create an element of variation for the decision trees and reduce the data for the individual models in terms of both samples and features. [4]

XGBoost

XGBoost is a decision-tree based machine learning model which uses gradient boosting techniques and can be used for both regression and classification purposes. Gradient boosting differs from ordinary boosting, as described in section 2.4.2 underneath "Boosting", by using the gradient of the loss function between iterations. The XGBoost classifier is deemed to be one of the most powerful machine learning algorithms when the dataset is sized small to medium.[25]

2.4.4 Evaluation metrics

Accuracy

Accuracy is the percentage of how many of the predictions made by the model that are correct. It is calculated as :

$$\text{accuracy} = \frac{\text{True}_{positive} + \text{True}_{negative}}{\text{True}_{positive} + \text{True}_{negative} + \text{False}_{positive} + \text{False}_{negative}}$$

Precision

Precision is the ratio between how many positive predictions a model classified correctly and the total number of positive predictions made by the model. It is calculated as:

$$\text{precision} = \frac{\text{True}_{positive}}{\text{True}_{positive} + \text{False}_{positive}}$$

Recall

The recall is the ability of the model to find all the positive samples, it returns a value of how many true positive a model recalls. It is calculated as:

$$\text{recall} = \frac{\text{True}_{positive}}{\text{True}_{positive} + \text{False}_{negative}}$$

F1-score

The F1-score can be interpreted as an average, or harmonic mean, of the precision and recall. An F1 score of 1 is max, and 0 is min. It is calculated as:

$$\text{F1-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

For precision, recall and F1-score we can also get a weighted and macro average. The weighted average considers how many of each class is in the dataset, whereas the macro average does not.

Top-N accuracy

Top-N accuracy is the standard accuracy of the true class matching to any of the N most probable classes predicted by the classification model. A classification is considered correct if any of the N predictions equals the correct class.

2.5 Hyperparameter tuning

Machine learning models has a set of parameters that can be tweaked to achieve better performance with regards to the different evaluation metrics, described in section 2.4.4, and overfitting.

2.5.1 Random forest parameters used for hyperparameter tuning

See table 2.1 for a list of parameters used in hyperparameter tuning for the random forest model. [23]

Table 2.1: Hyperparameter description - Random forest

Hyperparameter	Description
n_estimators	Determines the number of trees used in the random forest
criterion	The function in use to calculate the quality of a split. See more details regarding the criteria "Entropy" and "Gini" in section 2.4.1.
max_depth	The maximum depth of the trees, restricting the number of splits done.
max_features	The maximum number of features considered for splitting the trees.
min_samples_split	The minimum samples needed in to allow splitting a decision node. Restricting this will result in reaching the terminal node earlier and decide how well the model will fit.
min_samples_leaf	The minimum samples needed to define a cell as a terminal cell (also called leaf cell) and stop the splitting from proceeding further, no matter what the inpurity is.

2.5.2 XGBoost parameters used for hyperparameter tuning

See table 2.2 for a list of parameters used in hyperparameter tuning for the XGBoost model. [26]

Table 2.2: Hyperparameter description - XGBoost

Hyperparameter	Description
colsample_bytree	The subsample fraction of features to be used for training each tree. The selection is at random
gamma	Regularization parameter used to control the loss reduction.
learning_rate	Parameter used to control the speed of correction between each iteration of the boosting steps. May have implications for counteracting overfitting.
max_depth	Parameter used to control the maximum depth of the trees. To high depth of trees can cause the model to overfit.
n_estimators	Number of boosting rounds, and gradient boosted trees
subsample	The fraction of the training sample that will be used when training each tree. The selection is at random
tree_method	Defines the type of algorithms used to build the model as a whole

Chapter 3
Approach

This chapter presents the overall approach of this project. It is divided into a methodology part, a data section as well as baseline and modeling sections.

3.1 Methodology

In this master's thesis project we follow a framework called Cross-Industry Standard Process for Data Mining (CRISP-DM) to structure the workflow. See figure 3.1 for an overview of the process. The process is iterative and contains six parts: business understanding, data understanding, data preparation, modeling, evaluation and deployment. The following subsections cover the approach process by process, whereas specific details on data adjustments and modelling is presented under the respective sections 3.2 and 3.4 [19]

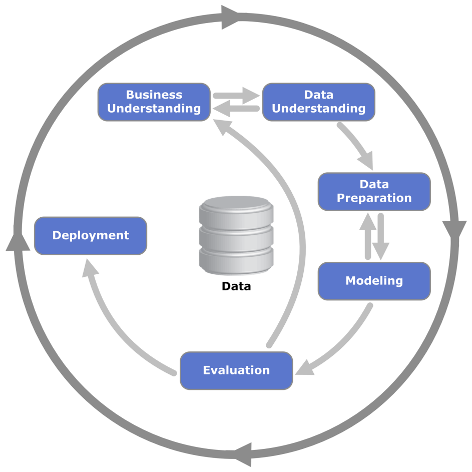


Figure 3.1: Overview of the CRISP-DM process. (CC-BY-SA-3.0 Kenneth Jensen)

Business Understanding

Business understanding is crucial to create a project roadmap and understand the problem at its core. In this thesis information was gathered through interviewing software engineers, process engineers, project managers and operators at Northvolt. The interviews were conducted in a semi-structured manner and the purpose was to understand the problem at hand and the data available as well as to gain domain-specific knowledge. The information gained is reflected and used throughout this thesis: from which machine we should collect data from, what data is included in the dataset, how it's gathered, what limitations there are and so forth.

We gathered an overall understanding of what the Downtime Management System at Northvolt is and how it works, described in 2.3.1. Additionally, information was used to improve the data quality by performing different data cleaning steps 3.2.4. Once researched we understood that the problem we needed to solve was a so called multiclass classification problem.

Data Understanding

There is stored downtime data for the stacker available from June 2021. The dataset used in this thesis therefore contains data from June 2021 to April 2022.

Data Understanding was an exploratory phase where we analysed the dataset, as described in 3.2.3. Data Understanding and Data Preparation was treated in an iterative fashion, adjustment on the data meant that new conclusions could be drawn and further adjustment followed. An overview of the data is described in 3.2.

Data Preparation

The data available needed cleaning and restructuring. The data cleaning included removing downtimes with more than one label, removing downtimes with no category and relabeling downtimes with an outdated category to a new one. Further details on this subject is presented in 3.2.4. In terms of restructuring the data we rearranged it in order to have each and every row represent a downtime event and alarms, along with the other features, represented as columns. see figure 3.3 for an illustration of the final dataset. Data preparation is important since all investments in data quality can be transferred between the models, which is not true for model code.

Modeling

Five baselines were established. The models were evaluated in accordance with the CRISP-DM model, and iterated with new models being added. Features were added and their relative importance was reviewed. In section 3.3 the process of introducing the baseline is presented and in section 3.4 details of modeling is described.

Evaluation

We evaluated the models using the following metrics: accuracy, precision, recall and F1-score. Additionally, the company wanted to know the Top-N accuracy for each model and it was therefore also used. For further detail in this subject, see section 2.4.4.

Deployment

The deployment step is not within the project scope. In section 5.2 we discuss how a future deployment could look like.

3.2 Data

3.2.1 Data Gathering

The data gathering step included merging of data from two data sources into one table. A SQL query was written to retrieve data from one table with downtime events and one with alarm events. The data was extracted from a database and converted into a Comma Separated Values (CSV) file format, which is suitable for processing [14]. To join the downtime and alarm data together we used an inner join with the condition of alarms occurring from and including the start and end time of a downtime event. See figure 3.2 for more details.

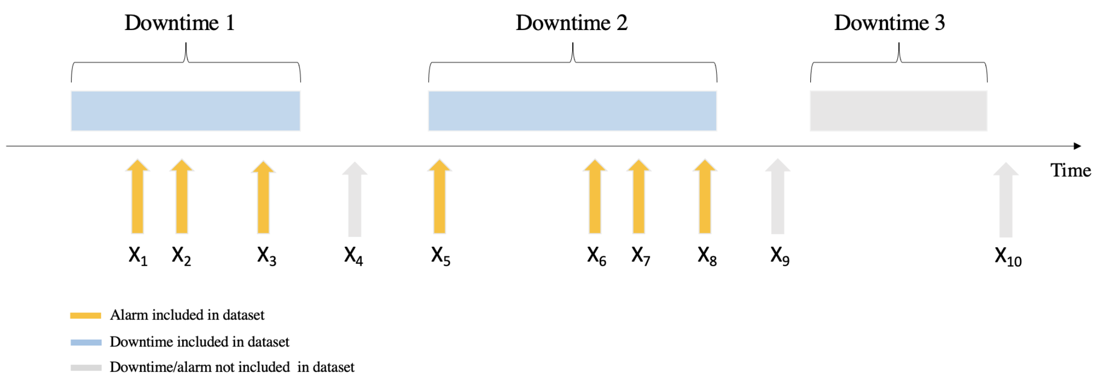
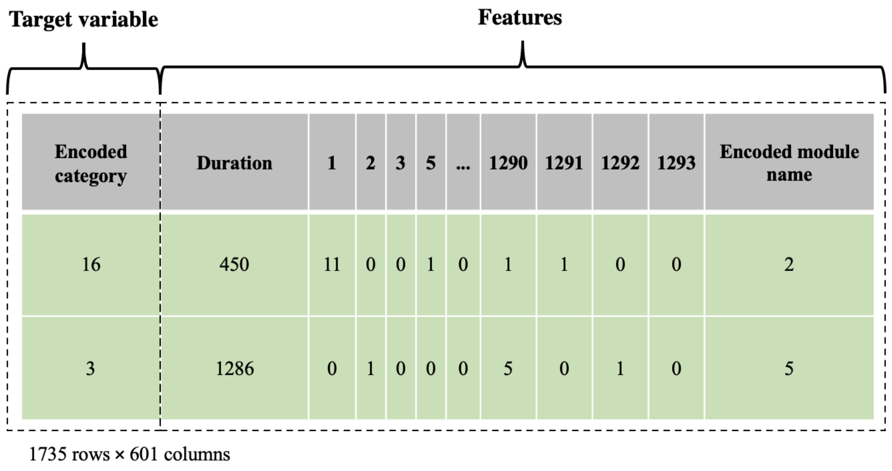

Downtime 1
Downtime 2
Downtime 3
Time
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
Alarm included in dataset
Downtime included in dataset
Downtime/alarm not included in dataset

Figure 3.2: Three downtime events are illustrated. Downtime 1 and Downtime 2 have alarms that are active (as seen in orange) during the respective downtime events. During the timeframe of Downtime 3, no alarms were active, and Downtime 3 is therefore excluded from the dataset. The alarms in grey are also not included in the dataset since they don't occur during a downtime event.

We excluded downtime events from our dataset that had no active alarms. This was done since the features of determining what type of downtime that has occurred is in the alarm data.

3.2.2 Dataset

The final dataset is a combination of downtime and alarm data. Every row represents a unique downtime event. As can be seen in figure 3.3 our final dataset contains 1735 unique downtime events, one target variable (encoded category) and 601 features.


Target variable
Features

Encoded category	Duration	1	2	3	5	...	1290	1291	1292	1293	Encoded module name
16	450	11	0	0	1	0	1	1	0	0	2
3	1286	0	1	0	0	0	5	0	1	0	5

1735 rows × 601 columns

Figure 3.3: Final dataset

- **Encoded category** is the numerical encoding of which category the downtime event belongs to. This is the target variable which our models will try to predict.
- **Duration** as described in section 3.2 underneath "Downtime". The reason to include duration as a feature is because there appeared to be a pattern between downtime categories and average downtime duration as covered in section 3.2.3 and showed in figure 3.7.
- **1-1293** are columns each representing one unique alarm. There were initially 1293 unique alarms for the stacker, but as described in section 3.1 underneath "Features" 694 of these were removed. One of the removed alarms was alarm with ID 4, as can be seen in figure 3.3 the column number 4, is not present in the final dataset. The number in the rows of these columns represent how many times the alarm was active during that downtime event. E.g. in the column named '1' it has a value of 11 in the first row and a value of 0 in the second row. This means that for the first downtime event the alarm with ID 1 was active 11 times and for the second downtime event it was never active. Instead of just showing a boolean of True or False to indicate that an alarm has been active during a downtime a decision was made to provide the number of times it had been activated during each downtime event since section 3.2.3 showed that the number of active alarms for each downtime vary between categories.
- **Encoded module name** is a numerical encoding of which module the downtime event belongs to.

Downtime

Each unique downtime between June 2021 to April 2022 is represented as a row in the dataset. Data attributes used in this thesis about each downtime event is as follows:

- **ID**: a unique identifier assigned to each individual downtime. The ID is generated upon the creation of a downtime event in Northvolt's Downtime Service.

- **Actor**: is a unique ID for a machine. In our dataset the actor field is the stacker's ID.
- **Start time**: is a timestamp of when the stacker went from a producing state to being down.
- **End time**: is a timestamp of when the stacker went from being down to a producing state.
- **Duration**: is the duration of the downtime in seconds, calculated as the difference between End time and Start time.
- **Category**: a label describing why the stacker was down. It is manually labeled by operators from a list of predefined categories or auto categorized by the Downtime Service. The category is what we aim to predict in this thesis.
- **Comments**: an optional free-text description written by operators, giving additional information about the downtime.

Alarm

Alarms are machine type specific, i.e., each process equipment has a unique set of alarms. The stacker in the chosen production facility has 1293 different alarms and each of these are represented as a column in the dataset. Out of the 1293 different alarms only 599 was active during a downtime event and the others were therefore removed. Data attributes used in this thesis about each alarm is as follows:

- **ID**: a unique identifier to each individual alarm event, generated upon alarm activation.
- **Actor**: is a unique ID for a machine. In our dataset the actor field is the stacker's ID.
- **Created**: is a timestamp of when the stacker received the specific alarm.
- **Alarm ID**: is an identifier that represents which type of alarm that has occurred. 1-1293 are possible alarm IDs.
- **Message**: is a description from the supplier of the machine of what type of alarm that has occurred in the form of a message.

3.2.3 Data Understanding

Data Understanding was an exploratory phase where we analysed the dataset. The dataset used in this thesis contains downtime and alarm data from June 2021 to April 2022 for the stacker. For the figures containing downtime categories, the categories are anonymized due to confidentiality reasons. The categories are also sorted in the same order so that one may compare them more easily.

Figure 3.4 shows the distribution of downtime events and their corresponding categories. As seen in 3.4 the dataset is rather imbalanced, meaning a majority of downtime events belong to a minority of the categories.

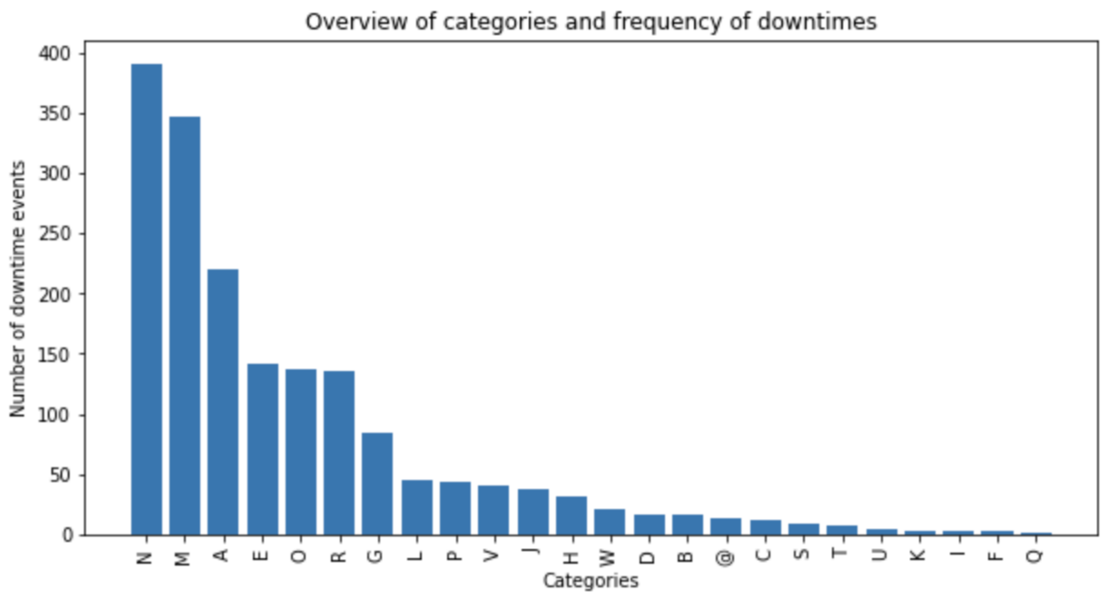


Figure 3.4: Distribution of downtime categories

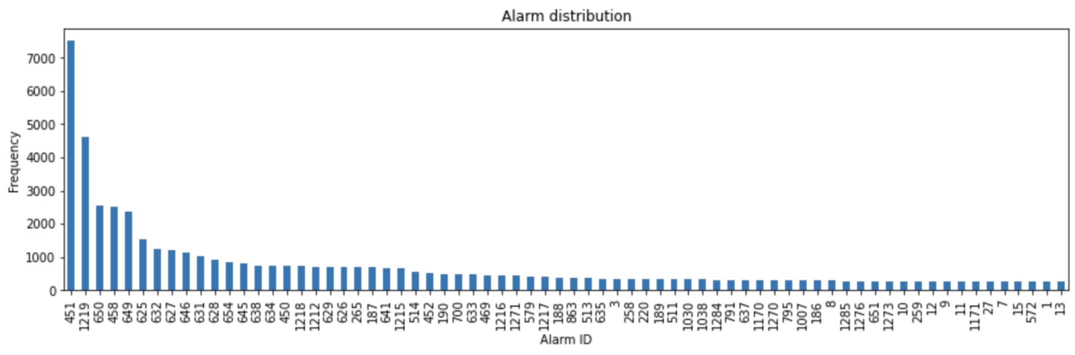


Figure 3.5: Overview of alarm distribution

In figure 3.5 the alarm distribution is presented for the top 70 most frequent alarms. As seen a few selection of alarms are overrepresented compared to the others. The total sum of alarms active during downtimes in the dataset is 76504, so the top three alarms, alarm ID 451, 1219 and 650 represent approximately 20% of the alarms. When viewing the right part of the graph one can tell that the alarms cause the dataset to be rather sparse since it consists of very many features (alarms), where most of them occur relatively seldom.

As presented in figure 3.6 the categories and frequency of alarms is presented. This in itself does not give very much information since it could easily be the case that downtime events which have longer duration accumulate more alarms, but once compared with 3.7 this effect can be mitigated since the two graphs don't fully align, only to some limited extent as seen specifically on category "T". If one compares 3.6 with 3.4 one may also see that the correlation of alarm frequency is not proportional to the frequency of downtime events with the specific category. The implication of this is that not only the individual specific alarms may impact the category of a downtime, but also that the number of alarms that occur may be relevant to determine what category it may be. This must be considered in the modeling.

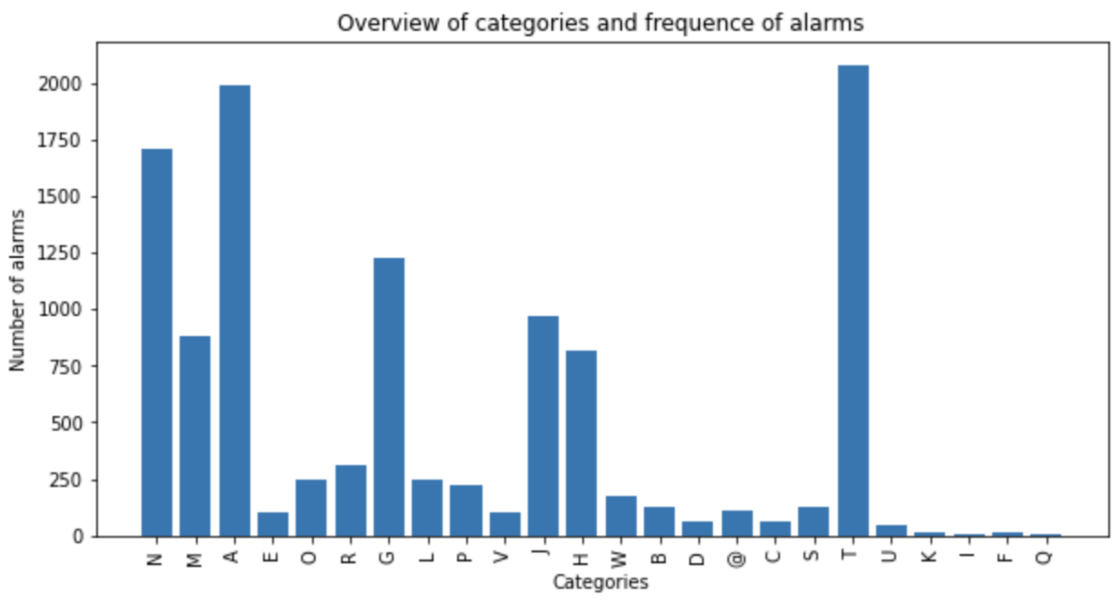


Figure 3.6: Overview of categories and frequency of alarms

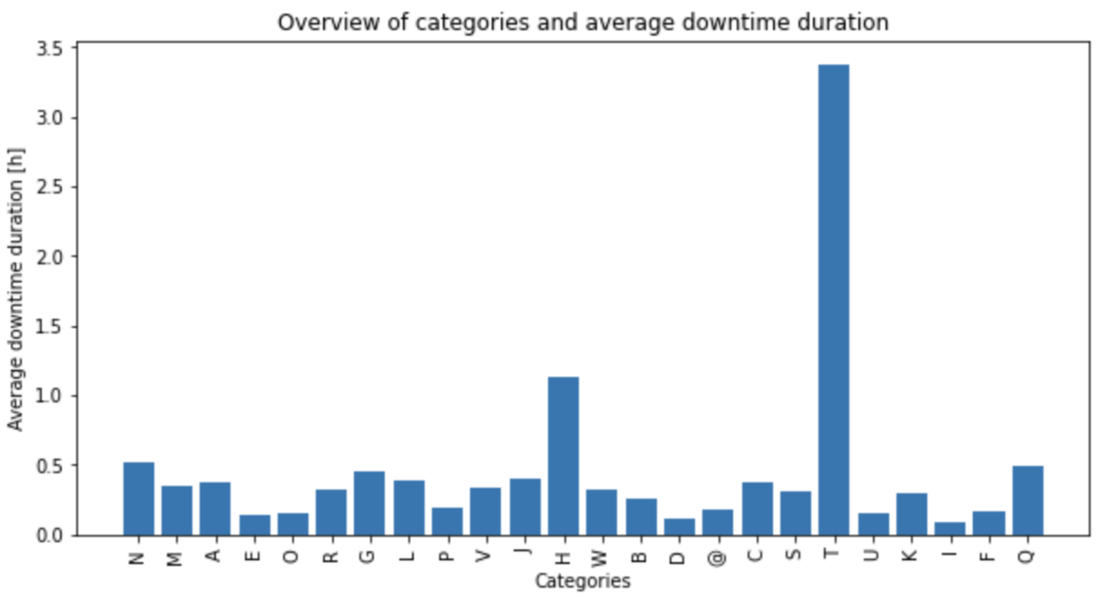


Figure 3.7: Overview of categories and average downtime duration

In 3.7 an overview of downtime categories and it's average downtime duration is presented. Besides for the implications of this figure, as covered above, one can note that the average downtime duration for individual categories differs largely, this is considered to be a potentially good feature for the models.

3.2.4 Data Preprocessing

To use the data as input to the models we needed to clean it and preprocess it.

Target variable

The target variable is the variable whose values are to be modeled and predicted by other variables. The target variable in our case is the downtime category that a downtime event has been labeled with. The category names are omitted for confidentiality reasons. We started with 53 different values on the target variable and ended up with 17, see figure 3.8. The number in the gray circle in the figure represents how many categories were left after performing the step described in the green box. We started out with 53 different values and after the first step 52 remained, and therefore it says 52 in the figure.

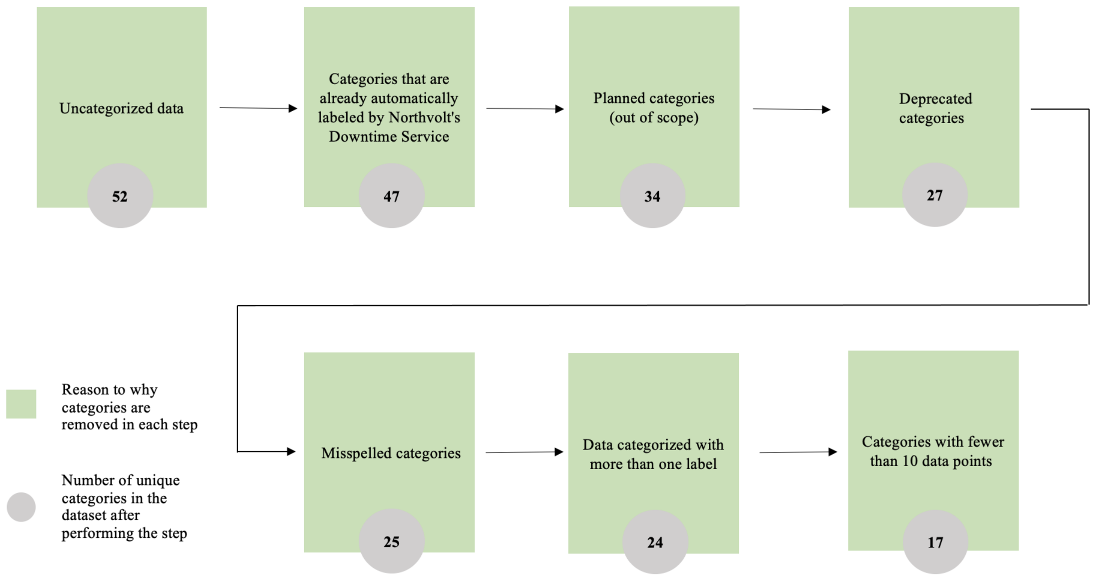

Uncategorized data
52
Categories that are already automatically labeled by Northvolt's Downtime Service
47
Planned categories (out of scope)
34
Deprecated categories
27
Reason to why categories are removed in each step
Number of unique categories in the dataset after performing the step
Misspelled categories
25
Data categorized with more than one label
24
Categories with fewer than 10 data points
17

Figure 3.8: Displaying how we went from 53 different values of the target variable to 17

- We removed all downtime events that were not labeled since we cannot use those for training classifiers.
- Downtime events with categories regarding planned downtimes were removed since these are out of scope.
- Categories that are already automatically labeled by Northvolt's Downtime Service were removed. The objective of the thesis is to classify downtime events with categories that are manually categorized.
- From our interviews in the Business Understanding phase 3.1 we realized that since the first labeled downtime event in our dataset some categories have changed, some are no longer used and have been replaced by others. We removed deprecated categories and some downtime events that could be relabeled with another valid category having the same meaning.
- We removed categories that were misspelled and relabeled the corresponding downtime events with the correctly spelled category. An example is removing a category "machine brekdow" and relabeling the downtime event to "machine breakdown".

- We removed all downtime events with more than one category, meaning that it had been labeled twice and we were unable to know which label was the correct one.
- Lastly we removed categories that had fewer than 10 data points. Less than 10 data points were deemed too few to predict the category accurately and train on.

As mentioned above, some categories were removed from the dataset since they had been deprecated. One of which had the label 'other'. In order to avoid removing all downtime events categorized as 'other' we relabeled some of them using comments written by operators in the Comments column indicating the label. This was only done when it was obvious that the comment had a correlation to the category and later confirmed with Northvolt.

When reviewing the historical downtime data we noticed that some categories which were not present in the beginning of the timeframe, were added later due to a need to categorize a certain machine fault. This may have problematic implications for a model running in production if new categories are added which the model hasn't previously seen. More on this subject in section 5.2.

Label encoding for the target variable

The machine learning models implemented in this thesis can't interpret the target variable in text format. We therefore converted each category to numerical format so that the models can interpret them. This was done using a label encoding method from SciKit-learn's library. The category "changeover" for example could after label encoding be represented by the number 16. We decided to use this method to convert our categories from text to numerical ones since our target variables does not have any ranking or ordering. This means that when we converted them from text format we didn't need to take into account any ordering. Another method that can be used is called One-Hot encoding. It adds a new column per unique category. In our dataset we have 17 unique categories and wanted to avoid increasing the dimensionality of the dataset, especially considering that the number of categories according to Northvolt will increase in the future.

Features

The stacker has 1293 alarms that can be triggered, see description of alarm data 3.2.2. Each specific alarm that can be triggered for the stacker machine was initially added as a feature to the dataset. These represented how many times that specific alarm was active during a specific downtime event. However, as mentioned in 3.2.2, out of 1293 unique alarms we found that 694 of these were never triggered at any given time for any downtime event in our dataset. These were therefore removed as features since they provided no information, see final dataset 3.3.

From the interviews we gained information that some modules might not have the same downtime categories as others. One module is for example first in line and might be down due to other reasons than the last module in the line. We therefore also included in the final dataset a feature based on which of the stacker's module the downtime event belonged to, see final dataset 3.3.

The duration of each downtime was added as a feature as well since figure 3.7 showed that the average duration varied largely between some of the categories, see final dataset 3.3.

Numerical encoding of the stacker's module names The stacker's module names is a feature in the final dataset. The module names are however in text format and the machine learning models implemented in this thesis need them to be in a numerical format. We therefore converted each module name to a numerical one so that the models can understand them. Since the modules have ordering we took this into account to not lose information. Therefore we decided to convert the six module names to a number between 1-6. A module that comes before another one in the line is represented by a lower number to keep the ordering.

Outliers Outliers exist in almost any dataset and can severely deteriorate the model accuracy. We examined and removed potential outliers in the "duration" feature. In our iterative work, this turned out to be especially important since it was shown to be the most important feature for predicting a downtime category.

Mitigate module/unit level problematic One limitation we identified from the start was that we only had alarm data on a unit level whereas we track downtime on the level of modules, see 2.2. In our initial dataset we therefore put all ongoing alarms during a downtime for a unit on all the modules belonging to that unit. However, during our interviews with the operators working with the stacker, gained in 3.1, we got information that they had learnt to recognize that some alarms belonged to a specific module. Using that information we were able to match specific alarms to a specific module and partly mitigate the problem. In the iterative process, this showed an improvement to the performance of the implemented models.

Training and test data

The dataset is divided into a training and test set with a ratio of 80/20 using SciKit-learn's package. Splitting is made with the stratify parameter set to the target variable of the dataset. This results in a split which takes proportions into account for each target variable, meaning that the division will strive towards an 80/20 split within each category. Besides for this condition, the splitting is made randomly. The test set is untouched for the entirety of the training process, and only used for the final evaluation of the models.

K-fold cross validation is used on the training data in order to most efficiently use the data for training the model. In K-fold cross validation the training data is divided into K subsets, whereas it is trained on K-1 subsets and validated on the remaining 1 subset. This is iterated K times, until each subset has been used as validation set. The scoring on the validation set is averaged out on all the iterations. Using cross fold validation also acts as a countermeasure to overfitting [20]. We acknowledge that our decision to use cross-validation disregards temporal relations in the data but it has still yielded positive results in previous work [20].

3.3 Establishing Baselines

To understand and evaluate our models' performance we implemented five different baseline models, divided into three naive baselines and two simple supervised learning models, see figure 3.9

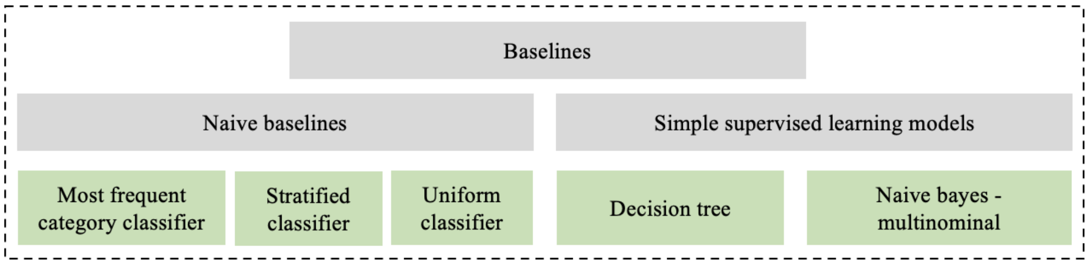

Baselines
Naive baselines
Simple supervised learning models
Most frequent category classifier
Stratified classifier
Uniform classifier
Decision tree
Naive bayes - multinominal

Figure 3.9: Baseline overview

- **Uniform classifier** generates predictions uniformly at random from the list of unique classes observed in the dataset, i.e. each class has equal probability.
- **Stratified classifier** generates random predictions by respecting the training set's class distribution. Categories which occur more often is hence predicted more frequently than those which occur less often.
- **Most frequent category classifier** always predicts the most frequent class label from the training dataset and makes predictions based on that label. This is a valid baseline in the presence of class imbalance as is the case in our dataset, i.e., the data contained some categories which were more frequently occurring than others, as covered in section 3.2.3 and presented in figure 3.4
- **Decision tree** is the most simple type of tree-based supervised learning models. The classifier creates predictions based on feature values within certain intervals, more on this described in 2.4.1. The quality of the split was evaluated with both gini and entropy, see section 2.4.1 for an overview of the two ways of measuring the quality of a split. In this thesis we did no tree pruning of the decision tree as we focused our efforts on implementing the ensemble models.
- **Naive Bayes** generates predictions based on Bayes Theorem as described in 2.4.1. The classifier is trained using the default parameters from the SciKit-learn package for multinominal classification.

3.4 Modeling

Two ensemble models, Random forest and XGBoost were implemented 3.10 following the modeling process as presented in figure 3.11.

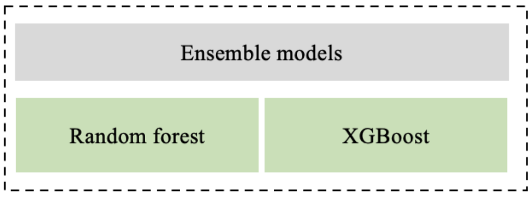


Figure 3.10: Ensemble models overview

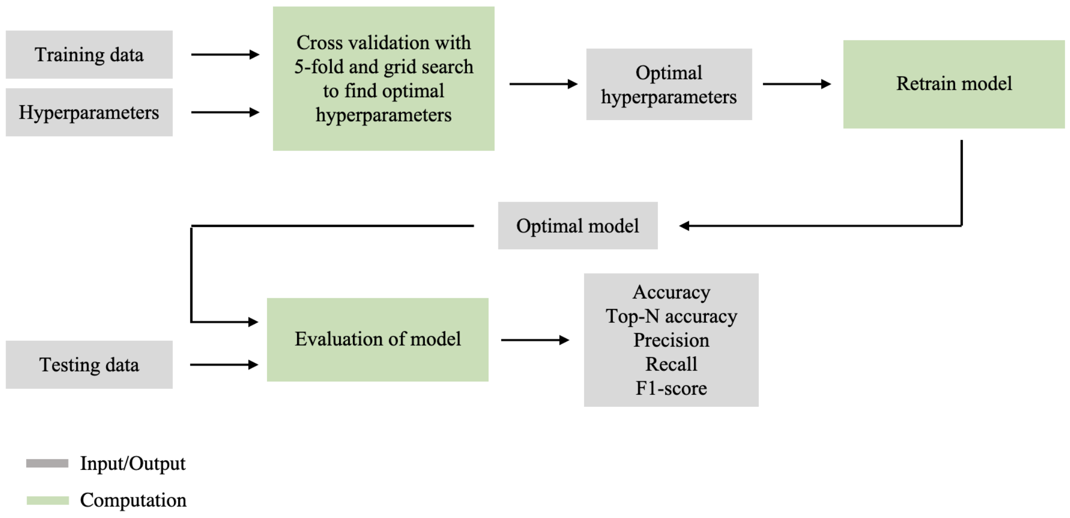


Figure 3.11: Modeling outline

- Training data from the test/train split (see 3.2.4) was used as input to a function along with a variety of hyperparameters. This was made using a grid-search which runs through all possible combinations of a set of parameters to see which parameters yields the highest score. SciKit-learn's package GridSearchCV [22] was used which combines hyperparameter optimization using grid-search and K-fold cross validation. In practice this meant that each hyperparameter combination is used to be trained and validated K times, see section "Training and test data" in 3.2.4 for further details. An evaluation is made once every combination of parameters has been used to train and validate model performance. A so called classification report was then generated with the accumulated performances and respective hyperparameters. Performance is measured as described in 2.4.4 (with the exception of Top-N accuracy), for each model. With this information one is able to draw conclusions on what hyperparameters which yielded the best score.
- Once optimal hyperparameters are identified they are used as an input to a model and training is done with the same training dataset.

- The fully trained model is tested with the unseen testing data. The result is evaluated and compared against the other models and baselines based on the metrics as described in section 2.4.4. As seen in section 3.2.3 the data is imbalanced and hence the weighed scores are being used instead of the macro ones.

Chapter 4
Result & Discussion

In the first section of this chapter we present the results of the five baseline and two ensemble models implemented in the modeling phase. In the next section a discussion follows to interpret the results. Then we discuss limitations and threats to the validity of our conclusions in the last section.

4.1 Results

4.1.1 Baselines

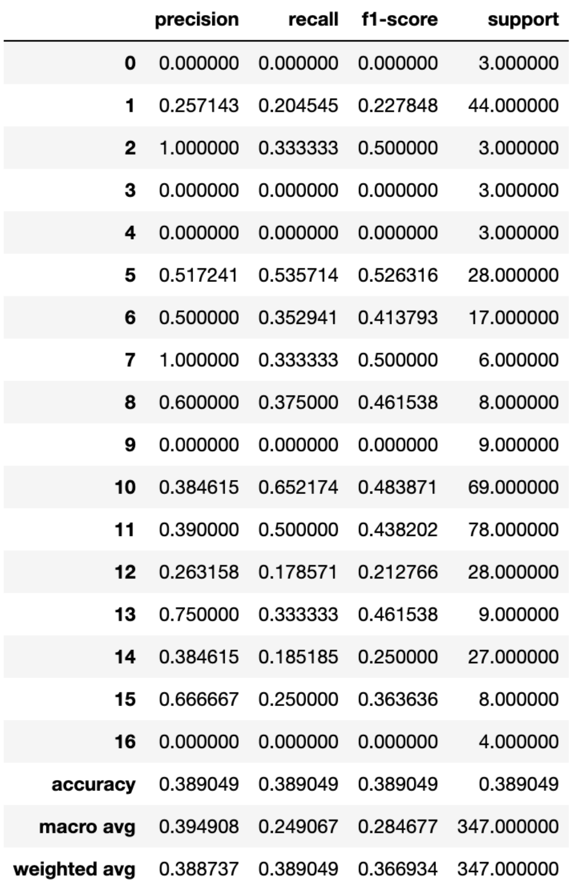
The performance of the five baseline models can be seen in table 4.1. Note that the values for F1-score, recall and precision are the weighted values, hence they take into account the number of actual occurrences of each class in the dataset. See section 2.4.4 for further details on the metric definitions. The best performing model with regards to all metrics is the decision tree.

Baseline classifiers				
Type	**Scoring**			
	Accuracy	**Precision**	**Recall**	**F1-score**
Random uniform	0.0446	0.1167	0.0446	0.0587
Random stratified	0.1247	0.1260	0.1247	0.1247
Most frequent	0.2232	0.0498	0.2232	0.0814
Naive Bayes (multinominal)	0.2345	0.1469	0.2345	0.1558
Decision tree (gini)	0.3087	0.3079	0.3087	0.3042

Table 4.1

4.1.2 Random forest

In figure 4.1 the precision, recall and F1-score is presented for each individual downtime category (anonymized) as well as the averaged scores and accuracy value for the testing set. The random forest model achieved an accuracy of 0.389. The weighted average values for precision, recall and F1-score are the following: 0.389, 0.389 and 0.367.



	precision	recall	f1-score	support
0	0.000000	0.000000	0.000000	3.000000
1	0.257143	0.204545	0.227848	44.000000
2	1.000000	0.333333	0.500000	3.000000
3	0.000000	0.000000	0.000000	3.000000
4	0.000000	0.000000	0.000000	3.000000
5	0.517241	0.535714	0.526316	28.000000
6	0.500000	0.352941	0.413793	17.000000
7	1.000000	0.333333	0.500000	6.000000
8	0.600000	0.375000	0.461538	8.000000
9	0.000000	0.000000	0.000000	9.000000
10	0.384615	0.652174	0.483871	69.000000
11	0.390000	0.500000	0.438202	78.000000
12	0.263158	0.178571	0.212766	28.000000
13	0.750000	0.333333	0.461538	9.000000
14	0.384615	0.185185	0.250000	27.000000
15	0.666667	0.250000	0.363636	8.000000
16	0.000000	0.000000	0.000000	4.000000
accuracy	0.389049	0.389049	0.389049	0.389049
macro avg	0.394908	0.249067	0.284677	347.000000
weighted avg	0.388737	0.389049	0.366934	347.000000

Figure 4.1: Classification report - Random forest

In table 4.2 the identified optimal hyperparameters, are presented. See more about hyperparameter tuning in section 2.5.

Table 4.2: Optimal hyperparameters

Random forest	
Hyperparameter	**Value**
criterion	entropy
max_depth	15
max_features	sqrt
min_samples_leaf	1
min_samples_split	2
n_estimators	750

The relative feature importance scores for the Random forest model's top 20 features are presented in figure 4.2. The duration feature proved to be the most important feature and the encoded module feature also proved to be a good predictor. The alarms seem to have quite similar feature importance, with the best alarm as a feature being alarm with ID 451.

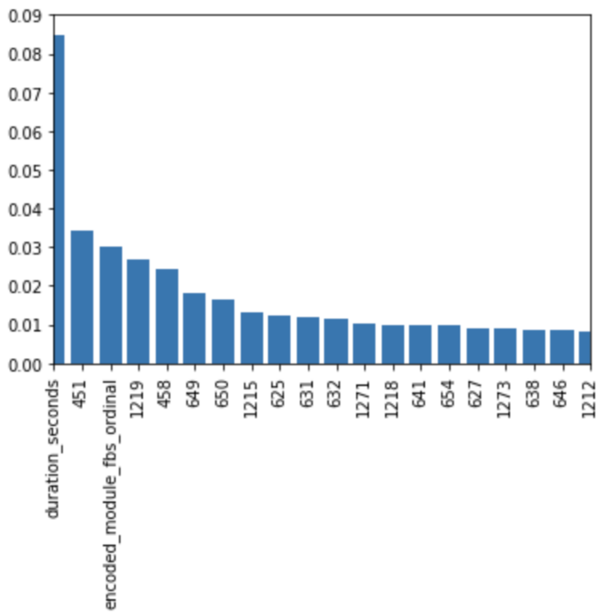
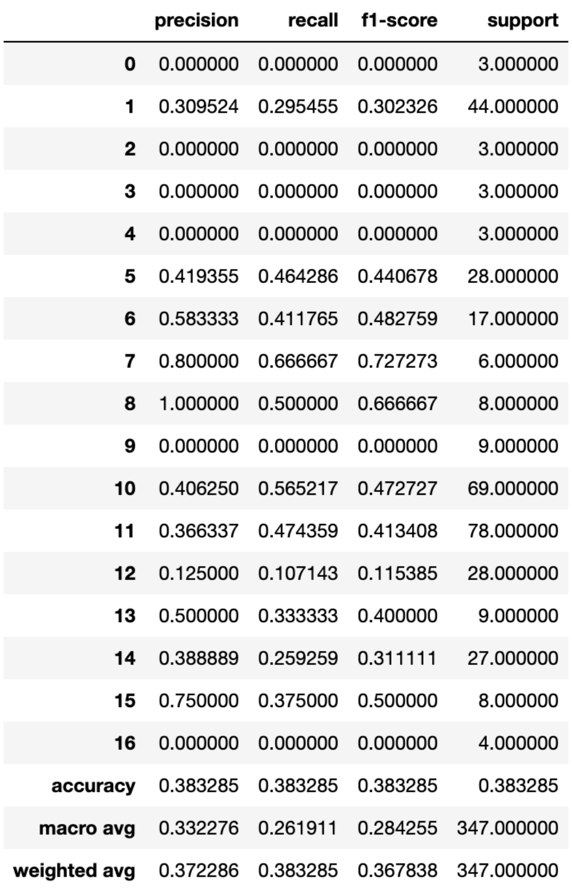


Figure 4.2: Feature importance Random forest (Gini importance)

4.1.3 XGBoost

In table 4.3 the precision, recall and F1-score is presented for each individual downtime category as well as the averaged scores and accuracy value for the testing set. The XGBoost model achieved an accuracy of 0.383. The weighted average values for precision, recall and F1-score are the following: 0.372, 0.383, 0.368.



	precision	recall	f1-score	support
0	0.000000	0.000000	0.000000	3.000000
1	0.309524	0.295455	0.302326	44.000000
2	0.000000	0.000000	0.000000	3.000000
3	0.000000	0.000000	0.000000	3.000000
4	0.000000	0.000000	0.000000	3.000000
5	0.419355	0.464286	0.440678	28.000000
6	0.583333	0.411765	0.482759	17.000000
7	0.800000	0.666667	0.727273	6.000000
8	1.000000	0.500000	0.666667	8.000000
9	0.000000	0.000000	0.000000	9.000000
10	0.406250	0.565217	0.472727	69.000000
11	0.366337	0.474359	0.413408	78.000000
12	0.125000	0.107143	0.115385	28.000000
13	0.500000	0.333333	0.400000	9.000000
14	0.388889	0.259259	0.311111	27.000000
15	0.750000	0.375000	0.500000	8.000000
16	0.000000	0.000000	0.000000	4.000000
accuracy	0.383285	0.383285	0.383285	0.383285
macro avg	0.332276	0.261911	0.284255	347.000000
weighted avg	0.372286	0.383285	0.367838	347.000000

Figure 4.3: Classification report – XGBoost

In table 4.3 the identified optimal hyperparameters, are presented.

Table 4.3: Optimal hyperparameters

XGBoost	
Hyperparameter	**Value**
colsample_bytree	0,5
gamma	0
learning_rate	0,2
max_depth	9
n_estimators	100
subsample	10
tree_method	exact

The relative feature importance scores for the XGBoost model's top 20 features are presented in figure 4.4. The most important feature is duration, with a score of almost 0.20. Alarm with ID 451 also proved to be a good predicator along with the encoded module feature. The rest of the alarms (except 1219), seem to have quite similar feature importance.

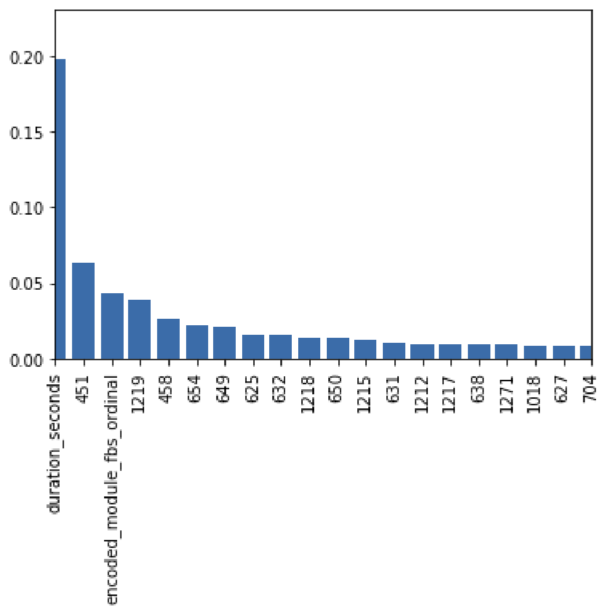


Figure 4.4: Feature importance XGBoost (using total gain measure)

4.1.4 Top-5 accuracy

In table 4.4 the Top-5 accuracy is presented for the three models that give a probability for each prediction, enabling us to get the Top-5 predictions. The Random forest model achieved the best score with a Top-5 accuracy of 0.823.

Type	Top-5 accuracy
Naive Bayes (multinominal)	0.672
Random forest	0.823
XGBoost	0.818

Table 4.4: Top-5 accuracy

4.2 Discussion

Ensemble models

The two ensemble models Random forest and XGBoost outperformed the five baselines with regard to accuracy, precision, recall, and F1-score. The Random forest model performed the best with an accuracy of 0.389 compared to the decision tree model which was the best performing baseline model with an accuracy of 0.309. Random forest utilises a large selection of ordinary decision trees to create its predictions as described in section 2.4.3 and this has been proven to be effective in classification in previous work, as described in the related work section 1.4. The XGBoost model however performed very similar to the Random forest one with an accuracy of 0.383.

The macro average values for precision, recall and F1-score are not of great interest compared with the weighed values since it is not taking into consideration that the distribution of downtime categories is uneven as seen in figure 3.4. In terms of the weighed values: the Random forest and XGBoost models have very similar values as can be seen in the classification report for random forest: 4.1 and for XGBoost: 4.3. Random forest have a slightly higher weighted average precision and recall value than the XGBoost model. The XGBoost model have a slightly higher weighted average F1-score (0.368 compared to 0.367). This minor difference is to be deemed negligible.

To get a wider image of how the models performed we measured Top-N accuracy. This was done for the two ensemble models and for Naive Bayes as a baseline to compare against. Top-5 accuracy was not calculated for the other baseline models since they do not provide probabilities for their predictions, not enabling us to get the next best prediction and so forth. The Random forest classifier performed the best in terms of Top-5 accuracy with a result of 0.823. The XGBoost model however achieved a result of 0.818 which is not a statistically significant difference compared to the Random forests result. The Naive Bayes model achieve a Top-5 accuracy of 0.672, which is lower than that for Random forest and XGBoost. We made the assumption that the hyperparameters identified with the highest accuracy would also yield the highest Top-5 accuracy, we deem this assumption reasonable, but if one would like to optimise for Top-5 accuracy one could rewrite the code in order to achieve this.

Top-5 suggestion is something Northvolt was very interested in. Giving a suggestion of top five categories to an operator will be an improvement of the current Downtime Management System and can help the operator in his or her selection of a category. According to Northvolt, it would make it easier for the operator, in line with "narrowing the selection" as argued by [18]. We hypothesize it would save time for the operator and make the categorization work task more accurate. Also it could make operators label more data which in turn would contribute to better annotated amount of training data for further supervised learning. Moreover, the feedback from this activity will benefit future work on autocategorization since the information of which category an operator ultimately selected could be used to further improve the classification model.

In comparison to the baselines there is a difference in terms of performance as measured by each evaluation metric, most notably comparing the Top-5 accuracies. This indicates that the correlation between the features and target variables is easier identified by the ensemble models and that the correlation is rather complex.

In comparison to related work such as [5], it is not surprising that our overall accuracy is substantially lower. We have more target variables (17 compared to 2) and 95% less data available. If we compare to another work [14], the authors received 100% accuracy. In that paper only two target variables (minor or major) were also used and we are therefore not surprised that their model performance was better than ours. Also, we believe our dataset has more complexity to it. In our dataset we have data from day one of the roll-out of the Downtime Management System at Northvolt. The dataset is therefore believed to contain some dataset inconsistency due to operators classifying data in different ways and not yet being experienced users. The fact that the machine which we collected data from is located in Northvolt's research and development facility, we believe also adds complexity to the data. However, these inconsistencies in the dataset might actually be representative of the true distribution of the data which the model will see if implemented. Another aspect which adds complexity to the dataset is that the Downtime Management System was continuously built on during the time span of the collected data. There can therefore be changes implemented affecting the tracking of the downtime events and alarms, contributing to further inconsistencies.

Baselines

- **Random uniform** This baseline performed the worst of the five baselines. This is deemed reasonable since the target space contains many classes and the occurrence of each class varies largely. It is the simplest way of modeling and hence it is not surprising that the model had the lowest scores in terms of accuracy, precision, recall and F1-score.

- **Random stratified** Predictions with this baseline is made randomly with consideration to the training set's class distribution. In the dataset some categories occur more often than others, as can be seen in figure 3.4. It was therefore expected that the Random stratified model, taking the distribution of the classes into account, would perform better than the Random uniform baseline.

- **Most frequent** This classifier predicts on the majority class every time which makes it a decent baseline classifier, seeing the distribution of downtime categories in figure 3.4. This classifier will be correct every time it sees the majority class and wrong all other times. This will result in a low quality prediction, but since the majority category is occurring very frequently the model's achieved accuracy of 0.223 is still decent.

- **Naive Bayes** Naive Bayes produce the second best predictions out of the baseline models with an accuracy of 0.235. It was expected to perform better than the three baselines mentioned above since it is able to create more refined predictions based on the features in the dataset, not just considering chance and the distribution of the target variables. One major flaw of the Naive Bayes model is that it does not take correlation of features into account, i.e. the model treats features as independent which is not very realistic for our case. For example, some alarms are module specific, and we have both these alarms as features and the module which the downtime event belongs to, meaning these features are very much dependent on each other. The Naive Bayes model's achieved accuracy is decent and quite similar to the accuracy for the 'most frequent' classifier (0.235 compared to 0.223).

- **Decision tree** The fifth and final baseline, Decision tree, achieved the best scores of all baseline models in terms of accuracy, precision, recall and F1-score. As mentioned in section 2.4.1 the decision tree is a simplistic model used for classification without ensemble techniques to counteract bias in the data. A lower accuracy score than the Random forest and XGBoost models is therefore to be expected. The model did however provide a good baseline to compare against.

Dataset

The dataset consists of relatively few downtime events, 1735, since the data is recorded from just one machine during an 11 month period. This is one factor which can negatively impact the models' ability to successfully train and identify true correlation between features and the target variables. The alarm data provides information on 599 alarms. Relatively few downtime events, combined with a large feature space and many categories to predict makes modeling a difficult task. Given the small size of the dataset, it is difficult to extrapolate how well our reported ensemble models' accuracies would hold up for a larger dataset collected over an extended time frame. Already by the end of this year there will be much more data collected since Northvolt's Downtime Management System is actively being used for the stacker and more data will be manually labeled.

Overfitting

In order to mitigate overfitting, actions were taken in terms of cross validation, hyperparameter tuning, and feature engineering. Cross validation allowed the model to train and validate the model progress without splitting up the data further than necessary. This mitigates the risk of overfitting. In the hyperparameter tuning phase, parameters which potentially have impact on the level of overfitting, were tweaked to improve results. For Random forest this included the maximum depth of the trees and the splits. For XGBoost the learning rate, subsample fraction and maximum depth of the trees was tweaked in particular.

Encoding

Another aspect that effects our models performance is encoding. Label encoding was used to convert the categorical text value of the target variable into a numerical one. However, since the label encoder used uses a number sequence it can be a problem that relations and comparisons between categories that do not exist are introduced. The models might assume that there is some hierarchy or order $0 < 1 < 2 \ldots < 16$ and might give X times more weight to a certain category in relation to another due to this. The models might also assume that categories with numbers close to each other are related even though it is not the case. This problem of creating relations that does not exist can be eliminated by using the One-Hot Encoding method instead. However, it was decided to not be used for the target variable since it adds a new column per unique category. This was considered challenging to manage since it would add many new columns to the data set, considering we have 17 categories. The decision was made especially considering that the number of categories can also increase in the future.

The feature regarding which stacker module the downtime event belonged to, was also encoded. The stacker modules have ordering and was therefore converted to numbers between 1-6, with the respective modules ordering taking into consideration. However, the encoding can contribute to the model creating some imposed correlations that does not exists, which affects the models' performances.

Training and test data

Since the total dataset is small a lot of thought went into how one would divide the data into training and testing sets. For hyperparameter tuning one could split the data into training, validation and testing data. Then one could train the model using training data, benchmark and find optimal parameters using the validation data and once optimal parameters were identified, test using the test data. The downside of this strategy is that this would result in less training data for the model, and since data was as scarce as it is, this was not deemed optimal. Instead K-fold cross validation was applied in order to utilise the data as much as possible in the training phase of the hyperparameter tuning. In the future work section, 5.2, some additional thoughts on how the testing and training can be divided is presented.

Feature importance

Feature importance for the two ensemble models is presented in figure 4.2 for Random forest and figure 4.4 for XGBoost. When comparing the two graphs one can see that both models have identified downtime duration as the most important feature. This means that duration has a large effect on both models and how they make their predictions. Downtime duration was added as a feature since we deducted that there was a correlation between downtime duration and specific downtime categories as seen in figure 3.7. Random forest and XGBoost also have the same four features as their top four most important ones. Worth noting is that the features with the highest feature importance are seen in figure 3.5 as the ones which occur most frequently, alarm ID 451, 1219 and 650 for example. The rest of the features, except the top five most important ones, for Random forest and XGBoost seem to have almost the same predictive power. Since feature importance measurements are strongly associated with a specific model, it is interesting that the models have almost the same features in their top 20. XGBoost has a larger difference between the importance score of its top two important features than the Random forest model (0.20 and 0.07 for XGBoost and 0.09 and 0.04 for Random forest). This can be explained by the XGBoost model picking one feature and using it to break down the tree further. The XGBoost model might then ignore some of the other correlated features. The feature picked is based on it having a high correlation with the other variables. In the Random forest model however the trees are not built from specific features but there is a random selection of features for each decision tree.

4.2.1 Limitations

This section reports a critical discussion of our work including the most important threats to the validity of our conclusions.

External validity

External validity is a way of reviewing how well the results of this thesis can be generalised into other contexts.

The method and result of this project apply directly to other setups with stacker-machines in Northvolt's future production facilities. For other production equipment with similar implementations that are to be added to the Northvolt fleet this can have implications as well, the alarms may come in a different format but the overall structure of the downtime management system is the same.

In the greater picture of things the results of this thesis are rather specific towards the use case at Northvolt, but if taken out of the Northvolt and battery manufacturing context the approach and findings may very well be adaptable to other industries for applications which depend on downtime management.

Internal validity

Internal validity is a way to review how internal bias and potential errors may impact the end result.

We assume that the operators' categorization is the truth. There can by all means be cases where the operators are not categorizing correctly due to mistakes, lack of time and so forth. We have for example seen in our data set a few selections of downtimes with two different categories. There might also be lacking categories to choose from for operators, so an operator might categorise with the most similar category instead.

The duration of a downtime event might not always be accurate due to inaccuracies with how the Downtime Management System functions. We consider both threats to internal validity acceptable for the proof-of-concept presented in this thesis.

Construct validity

Construct validity is a way to review the result of the thesis and if the metrics used to measure the result of the models are suitable.

The main means in this thesis of benchmarking the models is by comparing the measures of accuracy, precision, recall, F1-score and Top-5 accuracy. These performance metrics are all calculated in accordance with their definitions and they offer a good standardized overview of which model that provides the best results. See section 2.4.4 for more details on this subject.

Chapter 5
Conclusion & future work

5.1 Conclusion

This goal of this master's thesis was to identify if machine learning models could successfully classify the categories of downtime events based on downtime and alarm data. The approach of this project followed the CRISP-DM 3.1 framework, starting with business understanding in order to understand the problem, followed by an extensive iterative phase of data understanding, data preparation and modeling. Evaluation of model performance was done and the loop was iterated once more with further analysis and adjustments of the dataset, model tweaking and so forth.

The results show that alarms being active during a downtime event have a correlation with the reason of the machine being down. The ensemble models can therefore give information, at least to some extent, on why a machine is unavailable, especially the Random forest model. The findings indicate that machine learning can be used to determine the cause of downtime events. This master's thesis can be seen as a proof-of-concept whereas conclusive results would require larger amount of data to be analysed and supplied to the models for training. Machine learning may be used to either categorize a downtime event automatically or provide decision support to the operators to understand why a machine is unavailable. Due to the relatively low accuracy achieved by the best model we recommend, until more data is available for the model to be retrained on, to use the classification as an indication of what the category might be. Northvolt should determine a confidence threshold for how sure the model would have to be on a prediction to actually autocategorize that downtime event. If the probability for a prediction is below that threshold one could use the prediction instead as a suggestion for a category and leave it to the operator to confirm the option. By reviewing the results, presented in 4.1, one might be of the opinion that an accuracy of almost 40% for the two ensemble models implies that the predicting power of the models is rather weak, but first and foremost one need to compare with the baseline models. There is still a substantial difference between the ensemble models and the baselines in terms of the evaluation metrics. We suggest Northvolt to use the Top-5 predictions which was able to provide the Top-5 categories at an accuracy of 82%, using the Random forest model. This information can provide decision support to the operators and assist in their activities whilst increasing the downtime data quality with more categories being labeled. (RQ1)

This thesis suggests that one can train a machine learning model using active alarms, downtime duration and machine modules as features, to identify the Top-5 most probable downtime categories with an accuracy score of approximately 80%. (RQ2)

The research contribution goal of this thesis was to give knowledge and methods as a proof-of-concept into how historical production data can be used to categorize downtime events. The important features of determining the category of an event were also to be researched (see section 4.1 underneath the ensemble models) and the possibility to roll out a system for classification is discussed in the "discussion" section in 4.1. The approach used for analysing the data and modeling is presented in this report and there appear to be signals in the data to classify downtime categories, but in order to draw any further conclusions from this work more data is required.

5.2 Future work

First of all, as mentioned previously in the discussion section in 4.1, more data is needed. When this is in place, the model should be retrained and tested. After that we suggest that one should investigate using the alarm data available in more ways. Alarms are active during different times of a downtime event but this information was not taken into consideration in this thesis. We only have as feature if a certain alarm has been active during that downtime event or not but one could consider adding a feature with the delta between the time when the downtime event started and when the alarm was triggered. This would make this information available to the machine learning models. Also, we suggest to include alarms that were active before the actual downtime event. As described in the "Data gathering" section in 3.2 we only included alarms that were active during a downtime event. We think it would be interesting to include alarms that were active for example five minutes or one minute before a downtime started. In the future this might also help to actually predict a downtime event of a certain type even before it actually happens. This could help operators with troubleshooting and perhaps even decrease the duration of the downtime event.

Furthermore, it would be interesting to measure if a Top-5 suggestion actually helps the operator pick a category. One could use a product such as Mixpanel for measuring in real-time how the user interacts with the suggestions in the Downtime Operator UI.

The dataset initially had 53 categories and after the data was cleaned 17 remained as described in section 3.1. As described in section 3.2.4 there might be additional categories added in the future. The fact that categories can be removed/not used any longer affects the accuracy of the model when used with new data. It adds complexity and the model might then predict a category that does not exist any longer. There will therefore need to be some further data cleaning if applying the model to new data with new or removed categories. Also, the fact that new categories can be added means that if the model is used it would not have seen any downtime events with this category before and will therefore not make accurate predictions.

Using different splits of the training and testing data can be made in order to see if a better split can be achieved. Same goes for K-fold cross validation, one could try to use a different value on K than that of 5 which was used in this thesis. Once more data is available one could introduce a validation set as well to be used in the hyperparameter optimization step instead of cross validation to compare the results.

In this thesis autocategorization based on downtime event data and alarm data was researched for the stacker machine in one facility. This was chosen as a good proof-of-concept machine since it is a very complex machine. We suggest to follow the implemented steps in this thesis and test the possibility of autocategorization for other machines.

Bibliography

[1] Raid Al-Aomar, Saeed Aljeneibi, and Shereen Almazroui. Reducing operational downtime in service processes: A six sigma case study. In *2016 International Conference on Industrial Engineering, Management Science and Application (ICIMSA)*, pages 1–5, 2016.

[2] Amazon. *Amazon Kinesis*. `https//aws.amazon.com/kinesis/`. Accessed: 2022-05-20.

[3] Tessy Badriyah, Nadia Ayu Savitri, Umi Sa'adah, and Iwan Syarif. Application of naive bayes method for iugr (intra uterine growth restriction) diagnosis on the pregnancy. In *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pages 1–4, 2020.

[4] L Breiman. Random forests. *Machine Learning*, 45:29 – 30, 10 2001.

[5] Bastian Engelmann, Simon Schmitt, Eddi Miller, Volker Bräutigam, and Jan Schmitt. Advances in machine learning detecting changeover processes in cyber physical production systems. *Journal of Manufacturing and Materials Processing*, 4(4), 2020.

[6] Manuel Fernandez-Delgado, E. Cernadas, S. Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15:3133–3181, 10 2014.

[7] OPC Foundation. *What is OPC?* `https://opcfoundation.org/about/what-is-opc/`. Accessed: 2022-05-20.

[8] L.K. Hansen and Peter Salamon. Neural network ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12:993 – 1001, 11 1990.

[9] Fitriana Harahap, Ahir Yugo Nugroho Harahap, Evri Ekadiansyah, Rita Novita Sari, Robiatul Adawiyah, and Charles Bronson Harahap. Implementation of naïve bayes classification method for predicting purchase. In *2018 6th International Conference on Cyber and IT Service Management (CITSM)*, pages 1–5, 2018.

[10] Samuel H. Huang, John P. Dismukes, J. Shi, QI Su, Mousalam A. Razzak, Rohit Bodhale, and D. Eugene Robinson. Manufacturing productivity improvement using effectiveness metrics and simulation analysis. *International Journal of Production Research*, 41(3):513–527, 2003.

[11] A. Jung. Machine learning, the basics. *Machine Learning: Foundations, Methodologies, and Applications*, 1:12 – 15, 1 2022.

[12] A. Jung. Machine learning, the basics. *Machine Learning: Foundations, Methodologies, and Applications*, 1:70 – 72, 1 2022.

[13] A. Jung. Machine learning, the basics. *Machine Learning: Foundations, Methodologies, and Applications*, 1:140 – 142, 1 2022.

[14] Mira Chandra Kirana, Maidel Fani, Tri Shella Kartikasari, and Muhammad Nashrullah. Downtime data classification using naïve bayes algorithm on 2008 esec engine. In *2020 3rd International Conference on Applied Engineering (ICAE)*, pages 1–6, 2020.

[15] P. Muchiri and L. Pintelon. Performance measurement using overall equipment effectiveness (oee): literature review and practical application discussion. *International Journal of Production Research*, 46(13):3517–3535, 2008.

[16] S.C. Nwanya, J.I. Udofia, and O.O. Ajayi. Optimization of machine downtime in the plastic manufacturing. *Cogent Engineering*, 4(1), 2017.

[17] Jiaqi Pang and Jiali Bian. Android malware detection based on naive bayes. In *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, pages 483–486, 2019.

[18] Raja Parasuraman, Thomas B Sheridan, and Christopher D Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, 30(3):286–297, 2000.

[19] Randy Kerber Tom Khabaza Thomas P. Reinartz Colin Shearer Peter Chapman, Janet Clinton and Richard Wirth. Crisp-dm 1.0: Step-by-step data mining guide. 2000.

[20] R Bharat Rao, Glenn Fung, and Romer Rosales. On the dangers of cross-validation. an experimental evaluation. In *Proceedings of the 2008 SIAM international conference on data mining*, pages 588–596. SIAM, 2008.

[21] Anand Ravi. Cloud computing oee (overall equipment effectiveness) for reducing production downtime. *SAE International Journal of Materials and Manufacturing*, 6(3):481 – 486, 2013.

[22] SKLearn. *GridSearchCV documentation.* `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`. Accessed: 2022-05-20.

[23] SKLearn. *Random Forest Classifier documentation.* `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`. Accessed: 2022-05-20.

[24] Jaime Lynn Speiser, Michael E. Miller, Janet Tooze, and Edward Ip. A comparison of random forest variable selection methods for classification prediction modeling. *Expert Systems with Applications*, 134:93–101, 2019.

[25] XGBoost. *XGBoost documentation.* `https://xgboost.readthedocs.io/en/stable/`. Accessed: 2022-05-20.

[26] XGBoost. *XGBoost parameters.* `https://xgboost.readthedocs.io/en/stable/parameter.html`. Accessed: 2022-05-20.

[27] Hansong Xu, Wei Yu, David Griffith, and Nada Golmie. A survey on industrial internet of things: A cyber-physical systems perspective. *IEEE Access*, 6:78238–78259, 2018.

[28] Dahai Zhang, Liyang Qian, Baijin Mao, Can Huang, Bin Huang, and Yulin Si. A data-driven design for fault detection of wind turbines using random forests and xgboost. *IEEE Access*, 6:21020–21031, 2018.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2022-06-03

EXAMENSARBETE Classifying Downtime Events for Connected Factories Using Machine Learning

STUDENTER Alexandra Antgren, August Lindberg Brännström
HANDLEDARE Markus Borg (LTH), Sjoerd Dost (Northvolt)
EXAMINATOR Jacek Malec (LTH)

Kan maskininlärning användas för att kategorisera orsaken till varför en maskin är ur produktion?

POPULÄRVETENSKAPLIG SAMMANFATTNING **Alexandra Antgren, August Lindberg Brännström**

Att minska driftsstopp är ett viktigt ämne inom tillverkningsindustrin på grund av dess koppling till produktivitet och lönsamhet. I detta arbete presenteras hur maskininlärningsmodeller kan användas för att förstå orsaken till ett driftstopp och på så sätt kunna minska tiden som en maskin är ur produktion.

Varje timme, varje minut, varje sekund som en produktionslinje står stilla kostar det oerhörda summor för tillverkningsföretag. Att förstå orsaken bakom en maskins driftsstopp är avgörande för att kunna identifiera förbättringsområden och ha handlingsbar information.

I detta examensarbete implementerades maskininlärningsmodeller för att automatiskt kategorisera orsaken till en maskins driftstopp hos en svensk tillverkare av litiumjonbatterier. Detta gjordes med hjälp av historisk data på när maskinen var ur produktion och information om vilka alarm som skickats från samma maskin under samma tidsintervall. Datan analyserades, bearbetades och modeller valdes ut för modellering. Den modell som presterade bäst lyckades välja 1 av 17 kategorier med en träffsäkerhet på ca 40%. Detta anses vara en rimlig träffsäkerhet med tanke på att modellen hade många olika kategorier att välja mellan och en liten mängd tillgänglig data att tränas på. Resultatet kan jämföras med den bäst presterande baslinjemodellen som hade en träffsäkerhet på ca 30%.

Detta arbete har tillfört en större insikt i korrelationen mellan alarm som skickas från maskinen när den är ur produktion och orsaken till driftstopp. Möjligheten för företaget att använda vår modell för den undersökta maskinen är god. Vi rekommenderar att modellen till en början används för att ge en indikation på varför maskinen är ur produktion och inte ses som den slutgiltiga anledningen. Man kan också bestämma ett tröskelvärde för hur säker modeller måste vara på sin kategorisering för att man ska använda den. Vi rekommenderar att låta modellen föreslå fem kategorier istället för en, för att ge beslutsstöd åt de som sköter maskinen. När modellen ger fem förslag istället för en så presenteras i ca 80% av fallen den korrekta orsaken. Detta skulle spara värdefull tid för de som sköter maskinen. Tillvägagångssättet i rapporten kan följas för att bygga samma lösning för övriga maskiner i företagets fabrik.

Framtida arbeten med en större mängd insamlad data, hade sannolikt kunnat kategorisera orsaken till en maskins driftstopp med en högre träffsäkerhet.