NTNU
Norwegian University of
Science and Technology

Using Domain Knowledge in Classifying Industrial Data from the Oil and Gas Sector

Eirik Nilsen Fosse
Sondre Sæterli Hjetland

Master of Science in Informatics
Submission date: June 2018
Supervisor: Jon Atle Gulla, IDI

Norwegian University of Science and Technology
Department of Computer Science

This thesis is dedicated to the city of Trondheim for five wonderful years

Summary

Finding good features for performing supervised learning on high dimensional industrial datasets can be challenging, as the feature set typically consists of hundreds to thousands of features. Specific features might follow protocols or custom coding standards that unless decoded, are unusable by machine learning algorithms. This is often the case in industrial environments, where you need domain knowledge to interpret the semantics of the data. The objective of this research is to enable classification of industrial work orders into a predefined set of failure mode codes. Analyzing the effect of incorporating domain knowledge in the preprocessing phase of the supervised learning process is the main focus of the study. A thorough analysis is conducted to assess multiple supervised learning algorithms, to find fitting evaluation metrics, as well as to appraise the effect of extracting features from both structured and unstructured fields. Our experiments show that incorporating domain knowledge in the preprocessing phase improves the performance of the classifiers substantially. By utilizing domain knowledge we were able to increase the performance of the classifiers with ≈ 0.07 measured by Cohen's Kappa, an average relative improvement of 25.2%. An assessment of the feature importance in one of the final classifiers, showed that the sum of the importance of features extracted using domain knowledge was 38.97%. This implies that applying domain knowledge during feature extraction is crucial in order to avoid erroneous pruning of important encoded features, and to be able to extract more information from the dataset. The best classifier is currently not accurate enough to automatically label work orders with a failure mode code, but it is accurate enough to suggest failure mode codes when an operator submits new work orders.

Sammendrag

Det kan være utfordrende å finne gode attributter for å utføre veiledet maskinlæring på høydimesjonelle industrielle datasett, siden attributtsettet ofte består av hundrevis av attributter. Noen attributter kan være kodet etter spesifikke standarder eller protokoller, som gjør at attributtene ikke kan utnyttes av maskinlæringsalgoritmer med mindre de blir dekodet. Slike attributter finnes ofte i industrielle sammenhenger, og man er da avhengig av domenekunnskap for å forstå semantikken i dataen. Målet med dette arbeidet er å kunne klassifisere industrielle arbeidsordre inn i et predefinert sett med feilkoder. Det vil bli fokusert på å analysere effekten av å inkorporere domenekunnskap i preprosesseringsfasen av den veiledede læringen. Det er også gjennomført grundige analyser for å vurdere flere ulike maskinlæringsalgoritmer, evaleringsmetrikker, og effekten av å ekstrahere attributter fra både strukturerte og ustrukturerte felter. Eksperimentene våre viser en vesentlig forbedring i klassifiseringsalgoritmene vi testet, når domenekunnskap ble inkorporert i preprosesseringsfasen. Ved å utnytte domenekunnskap økte ytelsen av den beste klassifiseringsmodellen med ≈ 0.07 målt i Cohen's Kappa, en gjennomsnittlig relativ forbedring på 25.2%. Det ble også gjort en undersøkelse av hvor mye hvert enkelt attributt bidro i en av de endelige klassifiseringsmodellene. Denne undersøkelsen viste at viktigheten av attributter som var ekstrahert med domenekunnskap til sammen utgjorde 38.97%. Dette impliserer at det er helt avgjørende med domenekunnskap i preprosesseringsfasen, slik at man unngår feilaktig fjerning av viktige kodede attributter som kan brukes til å ekstrahere mer informasjon fra datasettet. Ytelsen til den beste klassifiseringsmodellen er for øyeblikket ikke god nok til å automatisk merke arbeidsordre med en feilkode, men den er god nok til å foreslå feilkoder når en operatør oppretter nye arbeidsordre.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) as a part of the fulfillment of a Master of Science degree in Informatics. The research is conducted by Eirik Nilsen Fosse and Sondre Sæterli Hjetland at the Department of Computer Science (IDI). It is supervised by professor Jon Atle Gulla.

Acknowledgements

First and foremost we would like to thank our supervisor, Jon Atle Gulla, for invaluable feedback and formidable ideas throughout the entire duration of the research.

We would also express our sincere gratitude towards Que Tran, Geir Engdahl, Erlend Vollset, and the rest of the team at Cognite, for providing us with a challenging task and for promptly assisting us with any questions we had and problems we encountered.

Last but not least, we want to thank our fellow students from the study hall Sule, for insightful discussions and for proofreading our thesis.

Table of Contents

List of Tables

List of Figures

Acronyms

API	Application programming interface.
AVA	All-versus-all.
BoW	Bag-of-Words.
CSV	Comma-separated values.
DT	Descision Tree.
E&P	Exploration & Production.
ECML-PKDD	The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.
ERP	Enterprise Resource Planning.
FMC	Failure Mode Code.
FN	False Negatives.
FP	False Positives.
FPSO	Floating Production, Storage and Offloading.
GBM	Gradient Boosting Machine.
IDF	Inverse Document Frequency.
IDI	Department of Computer Science.
MB	Megabyte.
ML	Machine Learning.
NB	Naïve Bayes.
NLP	Natural Language Processing.
NORSOK	Norsk Sokkels Konkurranseposisjon.
NTNU	Norwegian University of Science and Technology.

OVA	One-versus-all.
RF	Random Forest.
ROC	Receiver operating characteristic.
SE	Stacked Ensemble.
SFI	Skipsteknisk Forskningsinstitutt.
SMOTE	Synthetic Minority Over-sampling Technique.
SVD	Singular value decomposition.
SVM	Support Vector Machine.
TF	Term Frequency.
TF-IDF	Term Frequency-Inverse Document Frequency.
TN	True Negatives.
TP	True Positives.

Chapter 1

Introduction

"There is no such thing as a free lunch."

– David H. Wolpert

The no free lunch theorem states that there is no general purpose algorithm to solve all problems, and that there is no right answer on optimal data preparation when working with machine learning [70, 71]. This research focuses on finding suited algorithms and selecting optimal feature sets, in the context of classifying an imbalanced dataset that originates from the oil and gas industry.

1.1 Background and Motivation

A focus on a digitalization of the oil and gas sector has emerged in the past few years. World Economic Forum recently published a white paper, naming digitalization the new era for the oil and gas industry [59]. Digitalization is defined as the use of digital technologies to change a business model and provide new revenue and value-producing opportunities; it is the process of moving to a digital business. The industry has not yet utilized the opportunities that derive from technology and the large amounts of data that industrial companies possess. Industrial environments are data intensive settings, where data is generated from sensors, observations, reports and other instrumental readings. For instance, a single drilling rig at an oilfield can generate terabytes of data daily, but only a small part of the data is used for decision making. World Economic Forums' value-at-stake analysis estimates that a digitalization of the oil and gas industry could unlock approximately $1.6

trillion of value for the industry, its customers and wider society.

1.2 Context and Problem Description

This paper is written as part of a cooperation with Cognite, who are currently working on digitalizing the industry of Aker BP. Aker BP is one of Europe's largest independent oil and gas companies. The dataset used in this research consists of work orders from Alvheim FPSO (Floating Production, Storage and Offloading), an offshore oil and gas production ship operating on the Norwegian continental shelf.

A work order is a manual entry of a malfunction or other types of events, and it contains both structured and unstructured data. Most work orders contain a *failure mode code (FMC)*, specifying which type of malfunction that caused the work order. About 20% of all work orders are missing an FMC. It is beneficial for Aker BP to have FMCs on all work orders for the sake of further analysis and decision making. Being able to predict FMCs based on other data in a work order, can also be used to suggest an FMC when operators submit new work orders.

The main objective of this work is to train a classifier that is capable of predicting FMCs in work orders where the FMC is missing. This involves experimentation with multiple classification techniques and various approaches to feature extraction.

The dataset used in the study can be characterized as highly imbalanced when the training samples are grouped by their target class (FMC). Minority classes contain very few samples, making it hard to generalize a classifier and correctly predict unseen samples. Consequently, the research will also focus on finding informative evaluation metrics that works well for multiclass problems and takes class imbalance into consideration. This is important in order to train and select a classifier that performs well at the classification task at hand.

1.3 Research Questions

In order to put emphasis on the primary goals of the research, the following research questions have been defined:

RQ1 To what extent do new domain-driven features extracted from existing fields of industrial datasets contribute to a classifier's performance?
We want to investigate how domain knowledge can be utilized in the process of extracting new features from existing fields of the dataset. Features extracted using different types of domain knowledge will be tested both singlehandedly, and in combination. The features will be assessed using different classification algorithms, and evaluated based on multiple evaluation metrics. Cognite and Aker BP provided us with explanations and manuals with descriptions of the different features in the dataset. This also included a description of how values are encoded in some features. We only had one meeting with a domain expert from an oil platform throughout the research, which limited the amount of domain knowledge accessible to us. Data scientists from Cognite were available during the entire research, and contributed with domain knowledge to some extent. The results will not be evaluated in terms of computational efficiency. There are no requirements on training time, as the models do not need to be retrained frequently.

RQ2 How does features extracted from unstructured and structured fields compare in terms of improving a classifier's performance when classifying industrial data?
The dataset used in the study contains both structured and unstructured fields. We want to investigate how features extracted from structured and unstructured fields contribute to enhancing the performance of classifiers. The work on unstructured features will mainly focus on extracting more features using domain knowledge, and not using NLP. TF-IDF will be applied in order to form a basis to evaluate the effect of predefined domain terms in the process of extracting new features. A limited amount of work will go into optimizing TF-IDF and applying other NLP techniques.

RQ3 What are informative evaluation metrics when dealing with imbalanced multi-class datasets?
When classifying samples of an imbalanced dataset into multiple classes, it is important to use an evaluation metric that evaluates the classifier correctly based on the characteristics of the classification task. If it is crucial to classify minority classes correctly, some metrics might be more fitting than others. Other metrics may be more appropriate if the task is to classify as many correctly as possible. In our task, minority class samples are slightly more important to classify correctly than majority class samples. We want to investigate which existing evaluation metrics that are informative to use in our task. A limited number of metrics will be appraised, and those that will be considered are decided by similar research that involve imbalanced datasets. No effort will be put into developing new metrics or modifying existing metrics.

RQ4 How does a Stacked Ensemble compare with its constituent models when classifying highly imbalanced datasets?
A Stacked Ensemble is a meta-learner that combines knowledge from multiple classifiers in its predictions. We want to assess how a Stacked Ensemble performs compared to the constituent classifiers used in the ensemble. The base learners will be selected from a set of models built using different classification algorithms.

The content of the thesis is centered around these research questions, and they lay ground for a common thread throughout this work.

1.4 Approach

This section gives an overview of the approach taken in this research. The task of classifying work orders can be summarized as a three-step process that includes data preprocessing, feature extraction and training of classifiers.

Data preprocessing is done in order to improve the quality of the data and to make it more suited for the purpose of machine learning. In this step irrelevant rows are removed, as well as duplicate and highly correlated features. Some preprocessing is also done on the textual fields to make feature extraction more applicable and efficient.

The feature extraction step includes extracting features from both structured and unstructured fields using domain knowledge and TF-IDF. Domain expertise has been provided by Cognite and Aker BP through their employees, manuals, protocol descriptions and technical reports. These assets are exploited and used to identify additional features that can improve the predictive performance of the classifiers. Features extracted using TF-IDF are used as a comparison to the domain-driven features extracted from unstructured fields.

In the classification phase, four algorithms are employed and evaluated. These are Random Forest, Gradient Boosting Machine, Stacked Ensemble and näive Bayes. The algorithms were chosen on the basis of a run of H2O's AutoML feature. H2O is an open source machine learning and predictive analytics platform which has been used throughout this research. AutoML is a method that automatically trains and tunes several of the most popular supervised machine learning algorithms on the same dataset, in order to give an indicator on which algorithm best fits the problem. Hyperparameters were found by doing a *Random Grid Search*.

In order to evaluate the effect of the various approaches to feature extraction, a baseline model for each of the tested algorithms is trained. These baseline models are trained on the preprocessed dataset without the additional extracted features, with the same hyperparameters as used in the rest of the experiments.

The experiments are conducted such that all of the approaches to feature extraction are tested in combination and compared to the baseline. Multiple evaluation metrics are used in order to give a better overview of the predictive performance of the classifiers.

1.5 Results

The results of the models that yielded the greatest predictive performance in terms of Cohen's Kappa, are shown in Figure 1.1. Cohen's Kappa is an evaluation metric that takes class imbalanced into consideration, by weighting the impact of each class relative to their size. The models evaluated in the figure utilize features extracted from both structured and unstructured fields in the original dataset. Features extracted using domain knowledge had a significant impact on the performance of all the tested algorithms, increasing the score from the baseline experiment by an average of ≈ 0.07 measured by Cohen's Kappa. This equals an average relative improvement of 25.2%.

An evaluation of feature importance in one of the best performing models, showed that

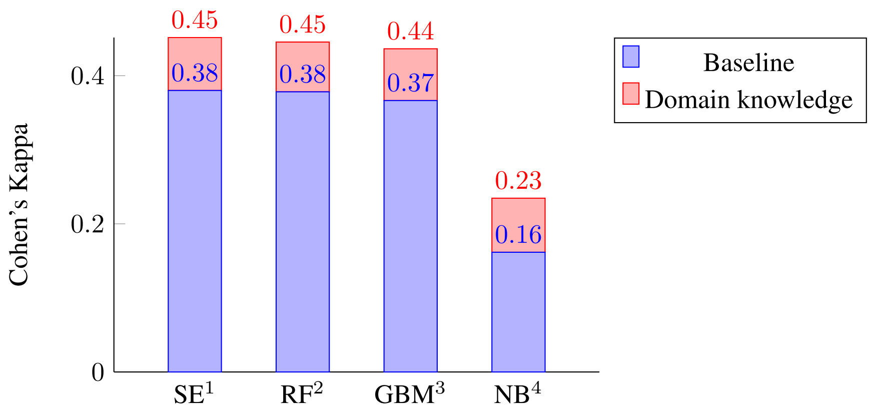


Figure 1.1: Summary of results (baseline models vs. models with incorporated domain knowledge). [1]Stacked Ensemble, [2]Random Forest, [3]Gradient Boosting Machine, [4]Naïve Bayes

six of the extracted features are among the top 23 most important features. Combined, the features that were extracted using domain knowledge constitute 38.97% of the total feature importance (100%). The two most important features were *Equipment_type* and *System*, both extracted from the structured field *FunctionalLocation*. *Equipment_type* refers to the type of equipment that a work order concerns, while *System* specifies the system that the equipment resides in.

The classifier that had the best performance was a Stacked Ensemble that incorporated features extracted from both structured fields and from unstructured features using domain terms. The model achieved a Cohen's Kappa of 0.4515 and an accuracy of 0.4851. This is not accurate enough to automatically label work orders with an FMC, but it is accurate enough to suggest FMCs when an operator submits new work orders.

1.6 Research Paper

As a part of this master's thesis, we have written a research paper on *Using Domain Knowledge in Classifying Industrial Data from the Oil and Gas Sector*. The paper is attached in Appendix A. It analyzes the effect of incorporating domain knowledge when training a classifier on an imbalanced high dimensional industrial dataset. Three different classification algorithms were used in the experiments. The performance of all the classifiers improved with ≈ 0.07 measured by Cohen's Kappa when domain knowledge was incorporated in the preprocessing phase. The paper was submitted to The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2018 (ECML-PKDD). At the time of writing, we have not gotten feedback on whether or not the paper is accepted to the conference.

1.7 Report Structure

The rest of the report is structured as follows: Chapter 2 gives a theoretical overview of the techniques and key concepts related to the task at hand. Related work is presented and discussed in relation to our problem in Chapter 3. Chapter 4 outlines the the origin of the dataset and context of the research. The dataset, along with a thorough analysis of its key characteristics and challenges, is presented in Chapter 5. In Chapter 6 the specific methods and implementations used in the research are presented. The conducted experiments along with their results are provided in Chapter 7. Chapter 8 and 9 concludes the research by discussing the results and findings, addressing the research questions, and providing recommendations for further work.

Chapter 2

Background

This chapter presents the background theory of the experiments conducted in this project.

2.1 Supervised vs. Unsupervised Learning

Supervised learning is the machine learning task of learning a mapping function f, that maps an input vector X to an output variable Y, based on *labeled* training data [30]. The training data consists of pairs on the form (X, Y), where X often is a vector of descriptive features and Y is the class (label) associated with that feature vector. In other words, a supervised learning algorithm uses labeled examples in order to approximate $f(X) = Y$. The inferred mapping function can then be used to classify new unseen input vectors. Supervised learning can be further divided into regression and classification based on the prediction method (real values versus classes).

Unsupervised learning is the task of learning a function to describe the underlying structure in data based on *unlabeled* input vectors. The most common unsupervised learning method is clustering, which is used for exploratory data analysis in order to find hidden patterns or groupings in the data.

The problem in this study is a supervised learning task. More precisely a multiclass classification problem. In other words, our dataset consists of labeled records belonging to a discrete number of classes.

2.2 Structured, Semi-Structured, and Unstructured Data

Structured data is data that has an unambiguous and defined interpretation. It has a high degree of organization, and a predefined data model. It is often organized into a formatted repository, such as a table where each column impose information about the data stored in the column. In the context of machine learning, the term column is used interchangeably with the term feature. Semi-structured data is data that lacks the same level of organization and predictability as structured data. The data does not reside in fixed fields or records, but it does contain elements that can separate the data into various hierarchies. Contrary to structured data, unstructured data does not have a predefined data model, and it is not organized in a specific manner. An example of unstructured data is a chunk of raw text, such as: *"On the 23rd of February 2018, John Doe conducted an inspection of the valve"*. Table 2.1 shows how this data can be structurized.

Table 2.1: Example of structurized data.

Date	**Task**	**Equipment**	**Conducted by**
2018-02-23	Inspection	Valve	John Doe

Unstructured data must be organized and structured in order to be useful in classification tasks. If it is left untouched it will be interpreted as a label, meaning that two unstructured values must be equal in order to have a relationship. When datasets consist of both structured and unstructured features, a common approach is to apply text mining to extract features from the unstructured text. The extracted features are then added to the existing set of structured features. This process is often used in for instance email spam filtering, since an email consists of both structured metadata and unstructured textual content [6]. Some of the most common approaches to extract features from unstructured fields are presented in Section 2.4.

2.3 Features

In machine learning, a feature can be described as an attribute or property of the input set, that can be used to build an accurate predictive classifier. For instance, in the task of training a model that predicts tomorrow's weather, natural features would be *Temperature*, *Humidity* and *Pressure*. By using these features, a classifier is to some extent able to differentiate between the various output classes.

2.3.1 Data Types

Features can be of various types, ranging from simple numerical values (e.g., today's temperature) to facial features extracted from an image for the purpose of facial recognition. There are four distinguishable types of data that are often encountered in machine learning:

Nominal A Nominal feature consists of values that are simply used for labeling. In other words, a nominal value is not assigned an order in relation to any other value. For instance, names are nominal values; it would make little sense to rank them in relation to one another.

Ordinal An Ordinal feature consist of values where the order matters; they are ranked relatively to one another. Students' grades (A-F) are examples of ordinal data, as the grades have a rank; an A is superior to a B and so forth.

Interval An interval feature consist of values where the difference between two of them is meaningful. An example of an interval feature is temperature; the difference between $20°$C and $30°$C, is the same as the difference between $30°$C and $40°$C.

Ratio A Ratio feature shares the properties of an interval feature, but additionally, any two values have a meaningful ratio. It is meaningful to perform multiplication and division on ratio variables. Temperature measured in Celsius is not a ratio feature due to the fact that $0°$C does not mean that there is no temperature, and $20°$C is not twice as hot as $10°$C. The Kelvin temperature scale on the other hand, assigns 0 to no temperature (absolute-zero) and is therefore considered a ratio feature.

It can be important to identify the data types of the feature set when training a classifier. Some implementations of machine learning algorithms requires numerical values and treats nominal data as ordinal data. This imposes an ordinal relationship between nominal values, which is often undesirable. For instance, the names *Leo* and *Jerome* might be encoded as the numerical values 1 and 2. With this encoding, one could evaluate $1 < 2$, or $Leo < Jerome$, which in reality is a nonsensical comparison.

2.3.2 Curse of Dimensionality

Datasets with a lot of features (dimensions) are prone to what is often called *The Curse of Dimensionality*. The "curse" is a phenomenon, that informally states that the complexity of a dataset increases rapidly with increasing dimensionality [11]. This is due to the fact that as dimensions grow, dimension space grows exponentially. In other words, the sparisty of the data points increase exponentially with regards to the dimensionality. Assume that a dataset consists of 10 samples and a single feature. This can be represented with only 10 values for x in one-dimensional space. If another feature were to be added, the data would be represented in two dimensions, and the dimension space would increase to 10^2. If yet another feature is added, the data would be represented in three dimensions, increasing the dimension space to 10^3. It can be seen that the dimension space increases by a magnitude of 10 for every additional feature (dimension), increasing the sparsity of the data points. This is a problem, as the difficulty of searching through the space gets a lot harder. Various dimensionality reduction techniques can be used in order to deal with the phenomenon. This includes automatic feature selection methods such as *backward feature elimination* and *correlation filtering* [27]. Domain knowledge can also be utilized to identify the most relevant features.

2.4 Unstructured Features

In order to be useful for supervised learning, unstructured features need to be preprocessed and organized in some way. There are multiple techniques to structurize data, for instance by building a *Bag-of-Words model*.

2.4.1 Bag-of-Words Model

The Bag-of-Words model (BoW) is a way to represent and make textual data more suited for the purpose of machine learning. In the Bag-of-Words model, a document is represented as a multiset (bag) of its words. With this representation, various document features can be extracted and used further in the machine learning process. A common feature to calculate using BoW is *term frequency* (i.e., the number of times a term occurs in a document).

As an example, assume the following corpus of documents $D = \{d_1, d_2, d_3\}$, where

d_1 = John likes art. He also likes tennis.
d_2 = Eric and Lisa like tennis. Lisa dislikes art.
d_3 = Tom dislikes tennis.

A vocabulary V is constructed from the distinct words in the corpus. This vocabulary defines the basis for a vector space.

$$V = \{john, likes, art, he, also, tennis, eric, and, lisa, like, tom, dislikes\}$$

As the vocabulary consists of all the words in the corpus, every document can be represented as a fixed-length integer vector of size $|V|$, where the position indicates the term and the value indicates the term frequency. The $|V|$-dimensional vector representations of d_1, d_2, and d_3 are shown in Table 2.2. Each column represents a term in V, and each row indicates a document. Index $a_{i,j}$ of the matrix indicates the term frequency of term t_j in document d_i.

Table 2.2: Example of document representation using Bag-of-Words.

	john	likes	art	he	also	tennis	eric	and	lisa	like	tom	dislikes
d_1	1	2	1	1	1	1	0	0	0	0	0	0
d_2	0	0	1	0	0	1	1	1	2	1	0	1
d_3	0	0	0	0	0	1	0	0	0	0	1	1

2.4.2 TF-IDF

One of the pitfalls of simply using the term frequency as in the example above, is that common terms such as articles and prepositions are heavily weighted. *Term Frequency-Inverse Document Frequency* (TF-IDF) is a simple and effective weighting scheme, often used in text mining and information retrieval in order to deal with this shortcoming [54]. The main motivation behind TF-IDF is to give a numerical statistic on how important a term is in a given document, based on term frequency (TF) and *inverse document frequency* (IDF). TF-IDF is defined as $TF \cdot IDF$.

Term Frequency

Term frequency is generally defined as the number of times a term occurs in a document. The term frequency may be weighted in several ways (e.g., Boolean, log scaled, or normalized over document length). When dealing with variable document text lengths, the term frequencies are commonly higher in longer documents, and they should be normalized. A commonly used norm is the euclidean norm (ℓ^2 norm) [42]. An advantage of euclidean normalization is that it takes document length into consideration.

Given a document d and a term t, let the term frequency $tf_{t,d} = f_{t,d}$ be the exact number of times t appears in d. This can be represented as a vector, $\vec{x} = (x_1, x_2, \ldots, x_n)$, where each vector element x_i denotes the term frequency of term i. The term frequency vector can be normalized using Equation (2.1).

$$\hat{f}_{t,d} = \frac{\vec{x}}{\|\vec{x}\|_p} \tag{2.1}$$

where $\hat{f}_{t,d}$ is the normalized TF vector, $\vec{x}$ is the vector that is being normalized, and $\|\vec{x}\|_p$ is the norm of vector $\vec{x}$ in the ℓ^p space (Lebesgue space).

The euclidean norm uses $p = 2$, which makes the denominator in Equation (2.1) equal to $\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$ for a TF vector $\vec{x}$. To normalize, each vector element x_i are divided by the euclidean norm. Table 2.3 shows normalized term frequencies using the example from the previous section.

Table 2.3: Example of document representation after normalizing term frequency.

	john	likes	art	he	also	tennis	eric	and	lisa	like	tom	dislikes
d_1	0.33	0.67	0.33	0.33	0.33	0.33	0	0	0	0	0	0
d_2	0	0	0.32	0	0	0.32	0.32	0.32	0.63	0.32	0	0.32
d_3	0	0	0	0	0	0.58	0	0	0	0	0.58	0.58

Inverse Document Frequency

The inverse document frequency is a measure of how rare a term t is across all documents in a document corpus D. It decreases the weight of commonly used words, and increases the weight of terms that are rarely used in the corpus. The IDF of a term t given a corpus D is calculated by Equation (2.2).

$$idf_{t,D} = \log \frac{|D|}{f_{t,D}} \tag{2.2}$$

where $|D|$ is the number of documents in the corpus and $f_{t,D}$ is the number of documents in D where t is present.

Combining TF and IDF

Using the definitions above, it is apparent that TF-IDF works by determining the frequency of a term in a specific document compared to the inverse proportion of that term over the entire document corpus [54]. In other words, the importance of a term is determined by how frequently it appears in a given document, and how rare it is in the corpus as a whole. By using the definitions of TF and IDF just introduced, Equation (2.3) can be derived for calculating TF-IDF.

$$tf_{t,d} \cdot idf_{t,D} = \hat{f}_{t,d} \cdot \log \frac{|D|}{f_{t,D}} \tag{2.3}$$

where $t \in d$, $d \in D$, and $\hat{f}_{t,d}$ is the normalized term frequency of term t in document d, .

An Example of TF-IDF weighting

Assume that we have the same corpus of documents as in the example given in Section 2.4.1, $D = \{d_1, d_2, d_3\}$. As the term frequencies are already given in Table 2.3, the task that remains is to calculate IDFs using Equation (2.2). After the IDFs are calculated, Equation (2.3) is used to determine the TF-IDF weights. The result is the document representations shown in Table 2.4. The term *tennis* was considered an important term in the TF representation, but by using TF-IDF the weight is reduced to 0 since it is present in all the documents.

Table 2.4: Example of document representation using TF-IDF.

	john	likes	art	he	also	tennis	eric	and	lisa	like	tom	dislikes
d_1	0.16	0.32	0.06	0.16	0.16	0	0	0	0	0	0	0
d_2	0	0	0.06	0	0	0	0.15	0.15	0.30	0.15	0	0.06
d_3	0	0	0	0	0	0	0	0	0	0	0.28	0.10

2.5 Imbalanced Datasets

When talking about imbalanced datasets in classification, it refers to an uneven distribution of samples when the samples in the dataset are grouped by their target class. Small differences in distribution can be manageable, but as the differences increase, the more challenging the classification task becomes. Large classes can quickly become very general, since many samples often contain a lot of distinct values for each feature [66]. Thus, many different feature values are associated with the large classes. A minority class requires new samples to be almost identical in order to be labeled with that class, since the classifier is trained on few samples with few distinct feature values.

There are many suggested approaches to combat the problem of imbalanced datasets, but there are three main approaches that cope with the problem on different levels. Sampling techniques can be used to even out the class distribution (data level), cost-sensitive learning can be used to weight smaller classes higher during model training (algorithm level), and changing the evaluation metric can favour classifiers where smaller classes are prioritized (evaluation level) [38, 31].

Applying sampling techniques on the dataset can even out the differences in the class distribution. The main sampling techniques are:

Undersampling Samples are deleted from the larger classes to even out the differences.

Oversampling Samples in minority classes are duplicated in order to get more training data.

Syntetic sampling Minor classes are oversampled by using synthetic sampling. SMOTE is a method where synthetic samples are generated based on features in the minority classes [13].

Accuracy might not be optimal when evaluating classifiers on imbalanced datasets. Changing the performance metric to for instance Cohen's Kappa, g-mean or macro average F_1-Score, can indirectly weight smaller classes higher when training a classifier. The rationale behind the choice of evaluation metrics and further discussions on evaluation metrics are presented in Sections 7.1 and 8.2.

With imbalanced datasets it is especially important to test multiple types of algorithms, since particular algorithms can perform better on irregular datasets. Tree based algorithms generally work well on imbalanced datasets [31].

2.6 Preprocessing

The representation and quality of the data is one of the most important factors for success. Machine learning algorithms often produce less accurate results, or fail to discover anything useful at all if the data contains extraneous or irrelevant information [39]. Data preprocessing is therefore a crucial step in order to produce a dataset that maximizes the performance of machine learning algorithms. There are many different techniques for preprocessing data, and it is often repeated in an iterative manner [18].

Generally more data is better, but in some cases datasets include data that is irrelevant for a classification task and cause noise. Such data should be discarded. Data must also be formatted in a suitable way for further processing. This includes joining data from multiple sources. If some samples or features have high sparsity (are missing many values), it must be considered if they contribute to training a classifier or if they should be discarded. Cleaning is the process of removing, adding or generating missing data. A commonly used technique for coping with missing data is sampling, which was presented in the previous section.

Preprocessing methods that only concern structured data are normalization and discretization of continuous features. Techniques that only applies to unstructured textual features are stop word removal, stemming and tokenization [19]. Using the example from Section 2.4.1, the following procedure shows how stop word removal and stemming can be applied to reduce the size of the vocabulary and simplify the document representation:

1. Initial vocabulary:
   $V = \{john, likes, art, he, also, tennis, eric, and, lisa, like, tom, dislikes\}$
2. After stop word removal:
   $V = \{john, likes, art, tennis, eric, lisa, like, tom, dislikes\}$
3. After stemming:
   $V = \{john, like, art, tennis, eric, lisa, tom, dislike\}$

Words such as he and and were previously considered important terms, but these are removed during preprocessing. After stemming, $like$ and $likes$ are considered to be the same word. As shown in Table 2.5, the TF-IDF weights are different after terms are removed during preprocessing.

Table 2.5: Example of document representation using TF-IDF after preprocessing.

	john	like	art	tennis	eric	lisa	tom	dislike
d_1	0.18	0.13	0.07	0	0	0	0	0
d_2	0	0.06	0.06	0	0.16	0.32	0	0.06
d_3	0	0	0	0	0	0	0.28	0.10

Another important step of preprocessing is engineering new features based on the existing data. Unstructured features such as text fields often contains ”hidden” information, meaning that machine learning algorithms cannot utilize the features unless they are extracted and structurized first (e.g., using Bag-of-Words). Structured fields may also contain encoded information that should be extracted.

2.7 Classification

In machine learning, classification is the task of approximating a function that maps every feature set X to a class Y. The goal is to be able to classify new unseen samples correctly.

2.7.1 Binary vs. Multiclass Classification

Binary classifiers are classifiers that maps an input to one of two output classes. For instance *yes or no*, and *positive or negative*. Multiclass classification on the other hand, maps an input to one of three or more output classes. Various algorithms can handle multiple classes by design, while others are binary classifiers by nature. Some binary classifiers can be adapted to handle multiclass problems.

There are three categories of multiclass classifiers [1]. *Extensible algorithms* are binary classification algorithms that are extended to solve multiclass problems. Decision trees, neural networks, k-Nearest Neighbor and naïve Bayes are examples of such algorithms. Neural networks can for instance be extended by having N binary neurons instead of one in the output layer.

The second category is *decomposing the problem into binary classification tasks*, and is for instance used for multiclass SVMs (Support Vector Machine). Decomposing a K-class classification problem into K binary classification task is the simplest approach [55]. This is called one-versus-all (OVA). When training a binary classifier for a class, the samples belonging to the class are considered positive, while the samples from all the $K-1$ other classes are considered negative. The winner is the classifier that produces the maximum output. All-versus-all (AVA) is another decomposition, where each class is compared to all the other classes [22]. This generates $\frac{K(K-1)}{2}$ binary classifiers. When a new sample is classified, it is classified by all the binary classifiers, and each classifier casts a vote for the class the sample should be labeled with. The sample is ultimately labeled with the class that received the most votes. Chih-Wei and Chih-Jen [32] compared the two decompositions against each other using SVM, and showed that AVA performs best in general.

Hierarchical classification is the third method that can be used to tackle multiclass problems. This method is inspired by the behaviour of decision trees, but each parent node is an actual binary classifier. The output space is structured as a binary tree, where each leaf node represents a class. Each parent node is divided into a cluster of classes, until the leaf nodes only consist of one class. The division of classes must be done in such a way that the discrimination between the clusters is as high as possible. By using training samples from each of the clusters in every parent node to train models, new samples traverse its way down the tree as it is classified on every level. Eventually it ends up in a leaf node, representing the class the sample should be labeled with.

2.7.2 Decision Tree

A decision tree (DT) is a supervised learning method used for classification and regression for both continuous and discrete data [57]. The method forms the basis for most of the more advanced techniques used in this study. Decision trees follow a procedure where the data is recursively partitioned into smaller subsets based on simple decision rules at each non-leaf node of the tree. The internal nodes have exactly one parent node and two or more child nodes. Every leaf node is associated with a class, and the input vectors are labeled according to the node they end up in. An example of a decision tree is shown in Figure 2.1. In this particular example, the goal is to decide whether or not an applicant is eligible for a mortgage loan. The decision is based on features such as the applicant's income, age, debt and employment status. This is a binary classification problem, however, decision trees are able to handle multiclass classification by nature. An advantage of decision trees is that new samples are very quick to classify. The decision tree algorithm produces highly interpretable models, allowing humans to inspect paths in the tree to understand the behaviour of the classifier. This is important in problems where it is valuable to understand how decisions are made, for instance in predicting medical conditions and the weather [11].

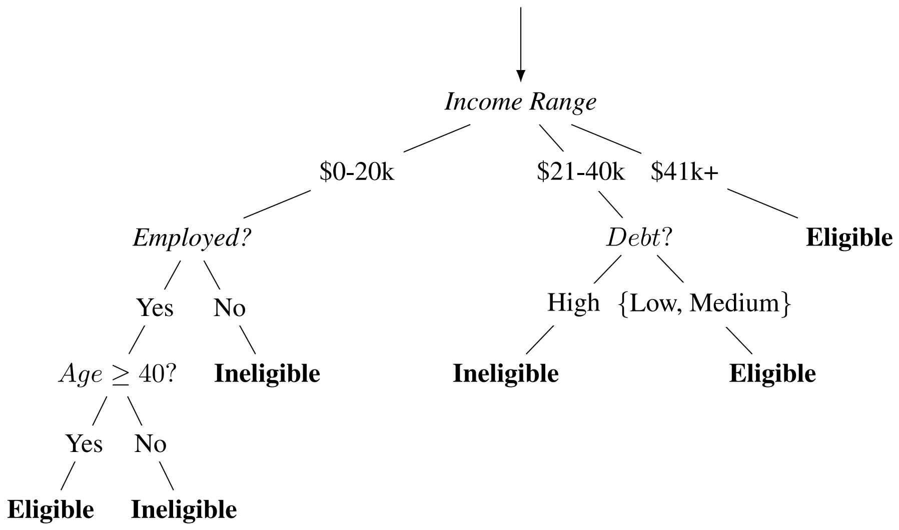


Figure 2.1: Example of a decision tree where the goal is to decide whether or not an applicant is eligible for a mortgage loan.

Decision Tree Induction

There are various algorithms that are used to construct decision trees. Some of the most well-established are CART, ID3 and C4.5 [11, 51, 52]. The implementations are mainly different with regards to the split criterion, how overfitting is addressed, and how they deal with missing data [72]. The algorithms construct the trees in a similar greedy manner, using a recursive top-down and divide-and-conquer approach. This is also the idea behind one of the earliest algorithms, *Hunt's Algorithm*, that forms the basis for some of the more complex algorithms. In order to build a decision tree, Hunt's Algorithm recursively subdivides the training data, X, into *purer* subsets, $X_1, X_2, \ldots, X_t$, where t denotes the node [35]. A high level 3-step description of the procedure is as follows:

1. IF X_t consists of records of the same class C_y exclusively, THEN label node t as a leaf node associated with C_y (terminal condition).
2. IF X_t consists of records of multiple classes, THEN split X_t into smaller partitions based on the attribute which discriminates best between the classes.
3. Recurse on each partition.

The recursive splitting of partitions continues until one of the following criteria is met: (1) When all samples in a partition belongs to the same class; (2) When all samples in a partition have the same attributes; (3) Another predefined stopping criterion is met. In the case where a partition contains a mixture of classes with no available split attributes, the node is assigned the majority class of that partition.

Assume that our training data consist of n records and m features, and that the depth of our decision tree is $O(\log n)$. At each level of the tree the entire set of n records are evaluated ($O(n \log n)$), and at each node all m features must be considered. In other words, the computational complexity for Decision Tree induction without pruning is $O(mn \log n)$ [68]. Classifying a new sample is done by conducting a single test at each level of the tree, meaning that the computational complexity for classifying a new sample is $O(\log n)$.

Split Criterion

An attribute selection measure is used in order to find the attributes that results in the most homogeneous class distribution in the nodes [57]. Two of the most commonly used attribute selection measures are Gini Index and Entropy [56].

Assume that before splitting, a node contains 10 samples from class C_0 and 10 samples from C_1. The best splitting criterion can be found using a greedy approach. A homogeneous class distribution in the nodes is desirable (e.g. 9 C_0 and 1 C_1 in a child node), with a lower impurity than the parent node. Gini Index and Entropy, shown in Equation (2.4) and (2.5), measures the impurity between two splits.

$$GINI(t) = 1 - \sum_{i=0}^{c-1} \left[p\left(i|t\right)\right]^2 \tag{2.4}$$

$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t) \tag{2.5}$$

where $p\left(i \mid t\right)$ is the number of samples in class i in node t.

In order to calculate GINI of the split, the GINI of all the nodes are combined using Equation (2.6)

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i) \tag{2.6}$$

where k is the partitions of the parent node, n_i is number of samples in child node i, and

n is the total number of samples in the child nodes.

The information gain of the split can be calculated by subtracting GINI of the split from GINI of the parent node p, as shown in Equation (2.7).

$$GAIN_{split} = GINI(p) - GINI_{split} = GINI(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} GINI(i) \right) \tag{2.7}$$

If there are multiple split options, the one with the highest $GAIN_{split}$ is selected. The same approach can be used for calculating Entropy gain.

Pruning

Pruning is a technique to combat overfitting, and it reduces the size and complexity of the decision tree [58]. It removes sections of the tree that provides little value to the classification task. A prepruning technique is to employ an early stopping rule, for instance [52]:

- The information gain for a split is less than a given threshold.
- The number of samples is less than a given threshold.
- The class distribution of the remaining samples is independent of their features.

After the decision tree is fully constructed, error-based pruning can be utilized:

- Remove nodes from the bottom and up.
- If the generalization error is reduced, replace subtree with a node.
- The class label for the new leaf node is the majority class in the pruned subtree.

2.7.3 Naïve Bayes

Naïve Bayes classifiers are a family of probabilistic classifiers based on Bayes' theorem (2.8). They all share the assumption that all features of a sample are independent of each other, regardless of any correlation between them. Even though this assumption is almost always wrong, practical comparisons have proven naïve Bayes to perform surprisingly well [29]. The independence assumption simplifies the learning significantly, since the parameters for each attribute can be learned separately. It also helps to alleviate problems that might arise from the curse of dimensionality. Popular naïve Bayes classifiers are gaussian naïve Bayes, multinomial naïve Bayes and multi-variate Bernoulli naïve Bayes [45, 34].

Naïve Bayes has been successfully used in spam filtering, text classification, and for predicting medical diagnosis [46, 45, 37]. It is popular due to its simplicity, and it often outperforms more complex models when it is applied on small datasets [65].

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{2.8}$$

where A and B are events and $P(B) \neq 0$.

With regards to a dataset, Bayes can be applied by using Equation (2.9).

$$P(c \mid X) = \frac{P(X \mid c)\, P(c)}{P(X)} \tag{2.9}$$

where X is a feature vector $(x_1, x_2, x_3, \ldots, x_n)$, and c is the class variable.

Given a set of target classes C and an unseen sample s, naïve Bayes algorithms calculate the probability of s belonging to each $c \in C$. Figure 2.2 shows how the probability for class c is calculated based on a feature vector.

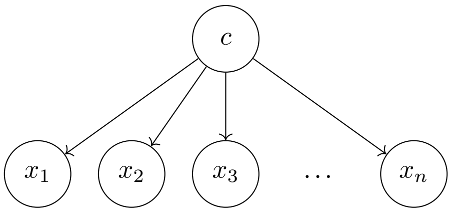


Figure 2.2: Naïve Bayes Classifier, each feature x_n from the feature vector X independently contributes on the probability for class c.

The final prediction is the target class with the highest probability, and can be calculated using Equation (2.10).

$$c_m = \underset{c_j \in C}{\arg\max} \left(P(c_j) \prod_{i=1}^{n} P(x_i \mid c_j) \right) \tag{2.10}$$

Calculating the initial probabilities requires a simple scan through all the training instances N and their features k [65]. In other words, the computational complexity of building the classifier is $O(Nk)$. Classifying new instances requires calculating probabilities for all the $|C|$ classes based on k features - i.e., the computational complexity is $O(|C|\, k)$.

2.8 Ensemble Learning

In order to better apprehend how ensemble learning works, it is important to understand the relationship between *bias* and *variance*. The total error of a model is based on bias, variance, and irreducible error (intristic noise that can not be reduced by algorithms) [23]. Bias occurs when a model generalizes well, but has limited flexibility to learn the true signal from the training data (known as underfitting). Variance refers to a model's sensitivity to small fluctuations in the training data [23]. A high variance implies that a model is too specific and that it models noise in the training data instead of the signal (known as overfitting).

Assume that six subsets are randomly subsampled from a training dataset. Now imagine that the six subsets are used to train six classifiers with the same algorithm and parameters. Bias can be thought of as the average performance of the classifiers, while variance is the consistency between them. Figure 2.3 is inspired by Fortmann-Roe [20], and shows four bulls-eye diagrams with possible scenarios of how the six classifiers perform. Each dot in the figure represent an individual realization of a model. A dot that is close to the bulls-eye is a classifier with a high accuracy. As we move away from the bulls-eye, the predictions get worse. Low variance (and high bias) algorithms tend to train models that are consistent, but inaccurate on average. Algorithms that produce classifiers with low variance are usually less complex, such as naïve Bayes. Low bias (and high variance) algorithms are often more complex and tend to train models that are accurate on average, but inconsistent. Decision trees are known to suffer from high variance, since one wrong choice cascades further down the tree and has an impact on all subsequent choices.

Training classifiers that focus on reducing the bias as much as possible will often result in higher variance, and vice versa. The bias–variance dilemma is the problem of simultaneously minimizing both bias and variance in order to reduce the total error [26].

The goal of ensemble learning is to reduce the total error by combining predictions of several base estimators in a way that reduces bias and variance [16]. Bagging and boosting are examples of such methods.

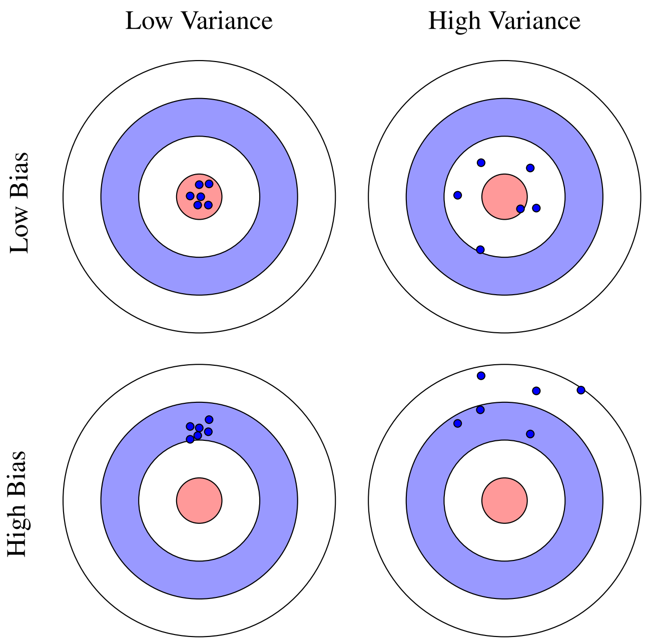


Figure 2.3: Bias vs. variance. Each dot represents a classifier that is trained using the same algorithm, but with different subsets of training data.

2.8.1 Bagging and Boosting

Bagging is an ensemble method where the base learners are built in parallel, independently of each other [8]. A randomly sampled subset of the data is used for each base learner, such that different subsets are used for training each learner. The same sample might occur in multiple subsets. After the base learners are trained, the validation samples are tested in all the classifiers, and ultimately labeled based on a majority vote between the classifiers. The main purpose of bagging is to reduce the error by reducing the variance.

Boosting is an ensemble method where the base learners are built sequentially [21]. It differs from bagging, since each iteration takes previous classifiers' success into consideration. Random sampling is used for generating subset of the data, but wrongly classified data is weighted higher in each iteration in order to emphasize the hardest cases. In boosting, the base learners are assigned a weight based on how accurate their results are. The weighted average of the base learners are used when predicting new samples, so that accurate learners have more influence. The main purpose of boosting is to reduce the error by reducing the bias, but it can also reduce the variance to some extent.

Both bagging and boosting reduce variance, but only boosting makes an effort to reduce the bias [3]. Bagging is generally better at reducing variance than boosting, thus making it more robust to overfitting [53, 21]. This is because bagging methods builds uncorrelated classifiers with random samples from the training data, causing different classifiers to make different errors.

2.8.2 Random Forest

Random Forest is an ensemble learning method that utilize bagging. It is frequently used for multiclass classification. Characteristics of Random Forest include robustness to noise, outliers and overfitting [10]. It works by generating a large number of tree-structured classifiers and fitting them to a random sample (with replacement) of the training set. The mode of the output classes of the individual trees are then used to conclude a final classification. Random Forest can in other words be defined as a collection of N tree-classifiers $\{T_k(\boldsymbol{x}),\ k = 1, 2, \ldots, N\}$, where $\boldsymbol{x}$ is the input vector of features. Each tree then predicts a class c_k based on the input vector, resulting in a total of N individual classifications, $\{T_k(\boldsymbol{x}) = c_k,\ k = 1, 2, \ldots, N\}$. The classification output c of the Random Forest, is the most popular class among these predictions. As Random Forests are ensembles of tree-based classifiers, they also support direct use of categorical variables. This is convenient due to the nature of the dataset used in this study.

Random Forest builds fully constructed decision trees, resulting in trees with a high variance and a low bias. Since it is a bagging method, its strengths is that the variance is reduced during the majority vote, as special cases are downvoted by other more general learners. General learners are learners that were not assigned data with the noise or outliers that caused the overfitting during the random subsampling. This makes it more robust to overfitting than boosting methods such as Gradient Boosting Machine. Random Forest does not reduce the bias, but it aims to keep a low bias by using fully constructed trees.

2.8.3 Gradient Boosting Machine

Gradient Boosting Machine (GBM) is a machine learning technique that uses boosting and steepest gradient optimization, in order to produce highly accurate, robust and interpretable classification models [24]. Boosting, as introduced above, works by combining the predictions of several weak classifiers with high bias and low variance, in order to make a final prediction based on a weighted majority vote [30]. The weak classifiers are built by sequentially applying a classification algorithm to repeatably weighted modified versions of the input data. In steepest gradient optimization, the loss function of the model is minimized by iteratively moving towards the steepest descent (negative of the gradient).

Gradient Boosting Machine works by first constructing an initial model $f_0(x)$ (optimal constant model) on basis of the training data. It then follows an iterative procedure where a regression tree (weak learner) is fitted to the remaining errors (*pseudo-residuals*) of the current model [30]. These pseudo-residuals are the gradient values of some differentiable cost-function L, that we want to minimize. For each leaf node of the newly constructed regression tree, a step is taken in the opposite direction of the average gradient (calculated from samples with similar features), and *line search* is used to determine the step magnitude γ. The result is an updated model. After a predefined number of iterations M, the final model $f_M(x)$ is returned. Pseudocode for Gradient Boosting Machine is shown in Figure 2.4.

```
1. $f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$
2. for $m = 1$ to $M$
    (a) Compute $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$ for $i = 1, \ldots, N$
    (b) Fit a regression tree to the pseudo residuals $r_{im}$.
    (c) Compute step multiplier $\gamma_{jm}$ for every leaf node $R_{jm}, j = 1, \ldots, J_m$
    (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I\left(x \in R_{jm}\right)$
3. Return $f_M(x)$
```

Figure 2.4: Gradient Boosting Machine regression algorithm.

Similar to the Random Forest, a strength of Gradient Boosting Machine is that it reduces the variance by using multiple learners. Another strength of Gradient Boosting Machine that does not apply to bagging techniques, is that it contributes to reducing the bias. It reduces the bias by starting with very general classifiers, before sequentially training new base learners with increased expressive power [63]. The new base learners are trained with increased weights on previously misclassified samples, making the base learners more specialized.

2.8.4 Stacked Ensemble

Stacking (also known as Super Learning or Stacked Regression) is a concept that aims to reduce overfitting and minimize the error rate, by deducing the biases of multiple generalizers with respect to a provided learning set [69]. Instead of selecting the best predictor in a set of classifiers, the idea of stacking is to combine several classifiers into a more accurate predictor [9]. Unlike bagging and boosting, stacking is a second-layer meta learner that intends to ensemble the base learners. Stacking can combine base learners built using simple classification algorithms, as well as learners built using bagging and boosting classification algorithms. The particular stacking algorithm used in this work is the Super Learner developed by van der Laan et al. [64]. The strength of a Stacked Ensemble is that it can utilize knowledge from multiple predictors, that can be constructed using different classification algorithms.

2.9 Evaluation Metrics

There are several methods to evaluate the performance of a classifier. Accuracy is commonly used, but recall, precision, F-Score or Cohen's Kappa might be better suited depending on the classification task. They can all be calculated from a confusion matrix.

2.9.1 Confusion Matrix

The performance of a classifier can be evaluated using a confusion matrix. A confusion matrix for a binary classification task is shown in Table 2.6. The true positives (TP) are the correctly classified samples belonging to the class. True negatives (TN) are the samples that was correctly classified as not belonging to the class. The samples that are incorrectly classified as members of the class are false positives (FP). Samples that actually belong to the class, but classified as not belonging to the class, are false negatives (FN).

In a multiclass classification problem, the confusion matrix should be interpreted slightly different. Table 2.7 shows an example of a confusion matrix where weather conditions are predicted. The main diagonal is now only representing correctly classified samples while the rest of the cells are wrong predictions. The confusion matrix can be interpreted by looking at the cells in the main diagonal one by one. For cell$_{ii}$, the number of TPs are

Table 2.6: Confusion matrix.

		Predicted class	
		P	N
Actual class	P	*True positives (TP)*	*False negatives (FN)*
	N	*False positives (FP)*	*True negatives (TN)*

the number in the cell, while the rest of the diagonal are TNs. The other cells in row i are FNs with regards to the predicted class, while the cells in column i are FPs. A confusion matrix gives detailed insight into how a classifier performs, but it does not yield a single value metric.

Table 2.7: Multiclass confusion matrix for weather predictions.

		Predicted class			
		Rainy	Cloudy	Sunny	Snowy
Actual class	Rainy	**17**	2	3	0
	Cloudy	1	**10**	2	3
	Sunny	1	4	**22**	1
	Snowy	5	2	3	**11**

2.9.2 Accuracy

Accuracy measures the fraction of predictions that are correctly classified. It is defined by Equation (2.11).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.11}$$

In a multiclass problem, it can be calculated from a confusion matrix by dividing the sum of the TPs in the main diagonal by the total number of samples.

Accuracy is a commonly used metric for evaluating classifiers. It can however be misleading when dealing with imbalanced classes, as it is very sensitive to class distribution [31]. Assume a classification problem where the goal is to identify disastrous malfunctions in an oil drilling system. Ten samples belong to the malfunction class, while 990 samples belong to the class where nothing is wrong. A classifier that predicts all the ten malfunction samples correctly, but additionally predicts 15 FPs, would get an accuracy of 98.5%. A majority classifier that always predicts that nothing is wrong would get an accuracy of

99%. This is referred to as the *accuracy paradox* [74]. The truth is that the majority classifier performs terrible for the problem at hand, while the first classifier could be considered to perform pretty well.

2.9.3 Recall & Precision

Recall and precision are metrics that originates from information retrieval, and both yield a value between 0 and 1. In a multiclass classification problem, recall and precision can be calculated for each class, before all the values are averaged to get a single value for both recall and precision. Recall measures the proportion of all the actual samples that are labeled correctly. It is not affected by incorrect positive predictions. Recall is defined by Equation (2.12).

$$Recall = \frac{TP}{TP + FN} \tag{2.12}$$

The FNs are samples that are labeled as not belonging to a class when they actually do. Recall is the probability that a randomly selected target class sample is actually classified as belonging to the target class.

Precision measures the proportion of classifications that is actually correct. It is defined by Equation (2.13).

$$Precision = \frac{TP}{TP + FP} \tag{2.13}$$

The FPs are cases where the classifier positively labels a sample, when it is not actually belonging to the class. Precision is the probability that a randomly selected sample that is classified in the target class is actually correctly classified.

In general, getting a higher precision often results in a lower recall, and vice versa [15]. Labeling all samples as belonging to a class, would give a recall of 1, but the precision will only be the proportion of correctly predicted samples. Setting a very high threshold for actually making a prediction in a binary classification task would give a high precision, but a low recall.

2.9.4 F-Score

Since precision and recall measures two different properties of a classifier, and maximizing one of the scores can negatively affect the other, it is useful to have a metric that combines both of them into a single value. The F-Score is the harmonic mean of precision and recall, and it is defined by Equiation (2.14).

$$F_\beta = \left(1+\beta^2\right) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{2.14}$$

A commonly used value for β is 1, where the precision and recall is weighted equally. This is known as the F_1-Score. When calculating the F-Score in a multiclass problem, it can either be macro or micro averaged to get a single value metric. The macro average is an average of the F-Score of each class. This implies that each class is equally weighted, which means that samples in minority classes are much more significant than samples in larger classes. Assume that one class has one sample and another class has 100 samples. The punishment for mislabeling the one sample class is equal to mislabeling all the 100 samples in the other class when using macro averaging.

Micro averaging sums up all the TPs, FNs and FPs, before calculating precision, recall and F-Score based on these values. Micro averaging does thereby not take class imbalance into consideration, and each sample is weighted equally.

2.9.5 Cohen's Kappa

Cohen's Kappa, shown in Equation (2.15), is a coefficient of interjudge agreement for nominal scales, where mere chance is excluded [14]. In order to remove random chance from the performance evaluation, Ben-David [4] argues for using Cohen's Kappa instead of accuracy. Its original intent was to measure the agreement between two or more people observing the same phenomenon, but in classification it measures the agreement between the classifier and the truth. It ranges from -1 (disagreement) through 0 (random) to 1 (agreement) [5].

$$\kappa = \frac{P_0 - P_c}{1 - P_c} \tag{2.15}$$

where P_0 is the relative observed agreement among raters, and P_c is the hypothetical probability of chance agreement.

In a multiclass classification task, Cohen's Kappa can be computed from a confusion matrix using Equation (2.16).

$$\kappa = \frac{n \sum_{i=1}^{C} x_{ii} - \sum_{i=1}^{C} x_{.i} x_{i.}}{n^2 - \sum_{i=1}^{C} x_{.i} x_{i.}} \tag{2.16}$$

where x_{ii} is the number of TPs (the main diagonal), n is the total number of samples, C is the number of classes, and $x_{.i}, x_{i.}$ are the column and row counts respectively [25].

Landis and Koch [41] have suggested that the Cohen's Kappa can be interpreted as follows: (< 0): No agreement, $(0.00 - 0.20)$: Slight agreement, $(0.21 - 0.40)$: Fair agreement, $(0.41 - 0.60)$: Moderate agreement, $(0.61 - 0.80)$: Substantial agreement, $(0.81 - 1.00)$: Almost perfect agreement. Let us revisit the example that was presented when discussing the accuracy paradox, where ten samples belong to a malfunction class, while 990 samples belong to a class where nothing is wrong. A classifier that always predicts the majority class in this example will have an accuracy of 99%, but the Cohen's Kappa will be 0. A classifier that predicts all the ten malfunction samples correctly, but additionally predicts 15 false positives, would get a Cohen's Kappa of 0.57. The reason that Cohen's Kappa punishes the classifier by 0.43 is that the class imbalance is so extreme that the minority class is much higher weighted than the majority class. Since 15 of 25 predictions in the minority class are wrong, the Cohen's Kappa is somewhat reduced. The new classifier is still considered to outperform the majority classifier. In addition to giving majority classifiers a score of 0, classifiers that randomly classify samples according to class distribution will statistically get a Cohen's Kappa of 0 [4]. A classifier that has a Cohen's Kappa of > 0, is considered to perform better than majority and random classifiers.

2.10 Validation

When training a classifier using supervised learning, you need to have a training set where the samples are labeled with the correct class. The goal is to train the classifier so that it generalize from the training data in such a way, that it later can classify unseen samples correctly based on the features of the samples. In order to do this, the training data is typically split into a training set and a test set. The training data is used to fit a model, which later will be used to classify new samples. Since we have a test set where the actual class is known, we can run it through the classifier and validate if it predicted the correct class or not.

In some cases it is common to have both a validation set and a test set. The validation set is then used to tune the hyperparameters of the classification algorithm in use. It does not modify the weights of the classifier that was learned during the fitting. Since the dataset used in this study was small relative to the number of classes, we only used a training set and a test set. Instead of tuning the hyperparameters using a validation set, we used a Random Grid Search in order to find the best hyperparameters.

2.10.1 k-Fold Cross-Validation

A common method used to validate the predictive performance of a model is the *holdout* method, where for instance 80% of the dataset is used for training, while the remaining 20% is used for testing the model. Assuming that the performance of a model improves in relation to the size of the training dataset, the holdout method is not optimal since a portion of the data is never used for training [36]. Fewer samples used as test data will widen the confidence interval for the accuracy. Increasing the number of test samples can lead to overfitting [62].

As depicted in Figure 2.5, *k-fold cross-validation* is a technique where the dataset is split into k mutually exclusive subsets of approximately equal size. The model is trained and tested k times, each time with a different fold held out as testing data. This ensures that all the samples are used both in training and in testing, and guarantees that minor classes are also used for testing. It is more accurate than the holdout method, without reducing the number of training samples [7].

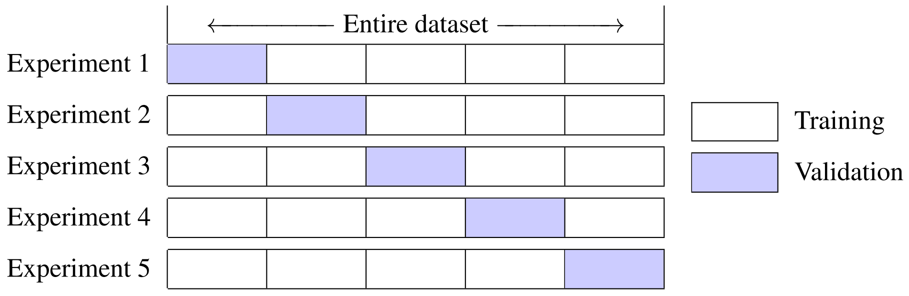


Figure 2.5: Dataset split into 5 folds for cross-validation.

In order to get a valid evaluation, it is crucial that the holdout sets for each of the k models are representative of the entire dataset. If a holdout set is missing samples from one class, it is not possible to validate samples that are classified as that class when the set is used for validation. This will have an impact on the final classifiers ability to predict the class. For large and evenly distributed datasets, random or modulo-based splitting are both fine since the probability of missing a class in a holdout set is close to zero. With an imbalanced dataset it is important to get samples from minority classes in each holdout set, and stratified sampling should be used. Stratification will try to evenly distribute observations from the different classes when splitting a dataset, thus making each holdout set representative of the whole dataset. It does however violate the principal that test labels should not be looked at before they are used for validation, and it may lead to a loss of diversity. Stratified sampling is generally a better scheme, both in terms of bias and variance, when compared to regular cross-validation [36].

In our environment, the k holdout predictions are combined into a single prediction for the full training dataset. This is done in such a way that all the samples are predicted, but the model that has made the prediction for a particular sample has not seen that sample during training. The overall cross-validation metrics of the model are computed by scoring the final model against the true labels.

Chapter 3

Related Work

We found some similar work in the manufacturing industry [61, 2, 44, 50, 47], but the datasets and classification tasks were not similar enough to be comparable to our problem. One paper was found on clustering and classification of maintenance logs from pump stations for dams and weirs, and will be reviewed in this chapter.

We also found some related work from other domains. Section 3.2 reviews a research paper on medical text report classification, which incorporates domain knowledge in their learning process. Section 3.3 reviews a paper where the task was to detect oil spills in satellite radar images. The study used a dataset with an imbalanced class distribution, and the researchers encountered many of the same challenges as we do with our dataset.

3.1 Clustering and Classification of Maintenance Logs using Text Data Mining

Edwards et al. have conducted experiments on clustering and classification of industrial maintenance logs, where text mining was used for feature extraction [17]. A company that maintains and repairs pump stations for dams and weirs provided the data used in the study. The objective of the work was to binary classify maintenance logs as scheduled work (65%) or an unscheduled fault repair (35%). The dataset consists of textual maintenance logs from 1994-2006, and covers work on pump motors, electrical systems, fire extinguishing systems, air conditioning systems, and external buildings and grounds. All the features in the dataset were unstructured free text, except for the target class. Some free text columns indicated the cause of the entry and if there was damage to the equipment.

The columns were joined together with the other text columns because of inconsistent values and a high level of sparsity.

In order to get the data ready for mining, a significant amount of work went into preprocessing. This included joining multiple text columns, correcting misspellings, removing punctuation marks and converting all text to lowercase. Phrases with similar meanings were transformed to a single word in order to embody important semantics, for instance $\{$*pump station, pump stat, pstn*$\} \rightarrow$ *pumpstation*. Stemming and stop word removal were also performed.

The output of the preprocessing was a dataset where each row consisted of one text string. Text mining was performed on the text by using TF-IDF, which generated term weights. Singular value decomposition (SVD) was used on the term weights to reduce the dimensionality by projecting them onto a smaller set of dimensions. Only the top 100 weighted terms were kept. The vectors were then clustered based on the SVD dimensions. Experiments showed that the optimal number of clusters was 14. Labels (type of maintenance job) were manually assigned to each cluster, based on their samples. Decision trees and neural networks were then trained to predict the cluster of new samples. The clusters provided useful information to the client, but they could not singlehandedly predict if a sample represented a scheduled or unscheduled job.

Decision trees and neural networks were also trained to classify if maintenance jobs were scheduled or not. The predicted cluster labels of samples were not used in this classification. Instead, the term weights were used when building the decision tree, and 17 SVD dimensions were used for training the neural network. The neural network and the decision tree achieved an accuracy of 82.8% and 85.0% respectively. The main contribution of the research was to show how information can be extracted from a low quality dataset using text mining, clustering and classification.

The researchers have thoroughly experimented with text mining in unstructured textual features, and extracted feature vectors that had a positive impact on the performance of the classifiers. More effort on extracting structured features could have improved the results further. The original dataset included columns that indicated the cause of the entry, if there was damage to the equipment, and if the work was part of a larger job. Even though the columns consists of free text, the researchers could have tried to structurize these columns and use them as features. The predicted cluster label could also be used as a feature when predicting if a job was scheduled or not. These techniques could have contributed to training more accurate classifiers.

Many of the challenges encountered in the research are challenges that we also ran into, such as low quality data, multiple data sources, a sparse dataset, and many grammatical errors. The reviewed research is however facing a binary classification task, while our target class set consists of far more classes. The dataset used in the research was somewhat imbalanced, but not to the same extent as our dataset.

3.2 The Role of Domain Knowledge in Automating Medical Text Report Classification

Healthcare is an information intensive environment, and an important task in healthcare informatics is to facilitate access to and improve quality of data. Wilcox and Hripcsak have conducted experiments on extracting clinical observations contained in medical text reports [67]. Structured data in medical reports was not sufficient for classifying hospital admissions. Wilcox and Hripcsak analyzed the effect of incorporating domain knowledge in the inductive learning process. Natural language processing (NLP) was used to convert unstructured textual features into structured features in the preprocessing phase. Specific attributes or features that were relevant to the classification task were then selected by domain experts. The dataset used in the study consists of 200 medical reports, categorized into six clinical conditions.

Experiments were conducted using varying degrees of domain knowledge utilization, and using classifiers from multiple paradigms (decicions trees, naïve Bayes, rule induction, nearest neighbor and decision tables). Domain knowledge was shown to be the most significant factor affecting inductive learning performance, outweighing differences in learning algorithms. The cost of acquiring domain knowledge was also a lot less than trying to learn said knowledge inductively. Numbers or tables on the exact results are not included in the paper, but by looking at a bar chart presented, we observe that domain knowledge improved the performance of all the classifiers. On average, the performance went from ≈ 0.82 to ≈ 0.91 measured by area under the ROC curve. That is a relative improvement of $\approx 11\%$. Wilcox and Hripcsak suggests that domain knowledge also can be used to combine multiple features together or to create new features, but they do not conduct any experiments on this matter.

The results of the study can be considered as strong and trustworthy, since the researchers have conducted experiments using learning algorithms from multiple classification paradigms.

All experiments showed an improvement when domain knowledge was incorporated. In order to further improve their results, it could have been interesting to experiment more with utilizing the domain knowledge for extracting features in the preprocessing phase.

3.3 Machine Learning for the Detection of Oil Spills in Satellite Radar Images

Kubat et al. have researched the use of machine learning in the detection of oil spills in satellite radar images [40]. The motivation behind the study was to create a system that could give an early warning on oil spills, which could have significant environmental impact. This is a binary classification problem, and the goal was to create a system that is capable of deciding whether or not a specific dark region of an image is an oil spill. A set of feature vectors describing dark regions of the image were used for training the classifiers. The results of the study were similar to earlier research, but the main contribution of was to identify issues deserving attention of the research community.

The design of the proposed system is heavily affected by key characteristics of the dataset, such as data scarcity and an imbalanced class distribution. Oil spills are relatively uncommon, meaning that the study had only access to a sample of 9 images containing a total of 41 spills. On the other hand, images that contain lookalikes of oil spills are easily accessible. This resulted in an uneven distribution of the classes. As a result of class imbalance, the classification system could achieve a great predictive score in terms of accuracy (96%) by simply classifying every record as a lookalike. This is a misleading result, as the classifier fails completely at its fundamental goal of classifying oil spills. The study argues that accuracy is an inappropriate measure in applications where the classes are unequally represented in the training set, and suggests that alternative metrics such as *geometric mean (g-mean)* and *F-score* are far more guiding.

The study relied completely on domain experts to define useful features for the classification task. A total of 49 features were constructed and used to describe the regions. The feature set consists of both generic features and features that implicitly represents domain knowledge through theoretical considerations. Many of the features proved to be insignificant in the predictive model, and the research speculates that the domain knowledge could have been better utilized if it was captured explicitly.

Using the decision tree algorithm C4.5, the researchers were able to achieve a g-mean of

0.811, and an accuracy of 76.6% on the positive samples. 1-nearest neighbor achieved a g-mean of 0.672 and an accuracy of 54%.

The dataset used in the study has many of the same characteristics as the dataset used in our study. This includes data scarcity, an imbalanced class distribution, and the fact that feature engineering using domain knowledge is required. In contrast to their work, our study uses explicitly predefined features in combination with features induced from domain knowledge. Our study includes an additional evaluation metric, *Cohen's Kappa*, that is more suited for our classification task, since our main goal is not to identify the minority class samples. The metric could potentially give a better indicator on the predictive performance of a classifier trained using an imbalanced training set, where minority classes should be prioritized slightly higher than larger classes.

3.4 Summary of Related Work

The majority of similar research focused on unstructured textual features. Some of the research had datasets that contained structured fields in addition to the textual features, that were not utilized in the classification task. We suspect that exploiting more than just the unstructured fields could have improved the performance of the classifiers. Especially in our case, since the dataset used in this study includes many structured features. We were unable to find similar research that utilize features extracted from structured fields in a high dimensional industrial dataset. This is especially important to examine in industrial datasets, as fields may contain values that are encoded using certain protocols and coding schemes. The fact that domain knowledge could be used to extract new features from existing structured features was pointed out by Wilcox and Hripcsak, but they did not conduct any experiments on the matter.

The points mentioned in the previous paragraph is what our research will focus on. We will extract features from both structured and unstructured fields with the use of domain knowledge. The domain knowledge will be applied in the preprocessing- and feature engineering phase, instead of being used for creating rules in a rule-based classifier. Kubat et al. highlights the importance of choosing correct performance metrics when dealing with imbalanced datasets. A thorough analysis will go into the selection of performance evaluation metrics in our research.

Chapter 4

Research Context

"Aker BP is collaborating with Cognite to make our data a strategic resource for accelerated performance, innovation and decision making. This partnership is a key enabler for our quest to digitize the E&P value chain."

– Karl Johnny Hersvik, CEO Aker BP

This master's thesis is written as part of a cooperation with Cognite AS, who are currently working on digitalizing Aker BP's oil and gas industry. Aker BP is one of the largest independent oil and gas companies in Europe measured in production.

Located in the central part of the North Sea lies the greater Alvheim area, where production started in 2008. It consists of the main field Alvheim, and the satellite fields Bøyla, Vilje and Volund. The depth of the sea in the area is between 120 and 130 meters. Alvheim FPSO (Floating Production, Storage and Offloading) is the production ship that is operating in the Alvheim area. The FPSO has an overall length of 252.16m, or 233m between perpendiculars, as well as a 42m moulded breadth and a 23.2m moulded depth [49]. It has the capacity to store 560.000 barrels of oil. Oil is stabilized and stored on the FPSO, before it is transported using tank ships. Gas is transported to an existing UK Scottish Area Gas Evacuation system using a cross-border pipeline.

The dataset used in this work originates from the Alvheim FPSO. It consists of work orders, which are prioritized tasks and maintenance logs with hundreds of different features. A work order is a manual entry of a malfunction or other types of events, and it contains both structured and unstructured data. About 80% of the work orders contains an FMC, which specifies the cause of the work order. As assigning an FMC to a work order is not mandatory in the ERP system used at Alvheim, the code is missing in some of them. The objective of this work is to improve the performance in the task of classifying FMCs in work orders. It is beneficial for Aker BP to have FMCs on all work orders for the sake of further analysis and decision making. The classifier can also be used to suggest FMCs when operators submit new work orders.

4.1 Work Order Life Cycle

The filing of a work order is a strict process where several different people are involved. Figure 4.1 shows a high level overview of the life cycle of a work order. A work order is primarily created on basis of two types of events:

1. Routine work that is scheduled at a fixed interval in the ERP system. For instance, inspection of equipment every third month.
2. Failures or other abnormal observations that requires attention. For instance, a leaking valve.

When a failure is first discovered, a notification is created and submitted to the ERP system. The notification is then put through a review process. If the observation is deemed insignificant, the notification is discarded and no further work is required. If not, a work order with priority, earliest start date and other formalities is created. It also includes the discipline (electrical, mechanical, process, etc.) that is responsible for fixing the failure. After the work order is created, it goes to planning. In the planning phase, the approach is planned in detail. This includes determining a date of fulfillment, deciding whether or not contractors must be used, checking if the required material is in stock or if it must be ordered, and so forth. The information is then added to the work order, and the work can begin. As work is conducted, the work order is updated by the disciplines that are involved in the process. The update includes the status of the work order after their work is completed, as well as an optional text that provides relevant information for later work. This is an iterative process that keeps going until the failure is identified and fixed. Finally, the work order is marked as completed and kept in the ERP system for further reference. This includes statistical analyses on company, plant, equipment and supplier level. The stored work orders are also utilized when the same or different equipment are exposed to similar failures, in order to see how it was previously solved.

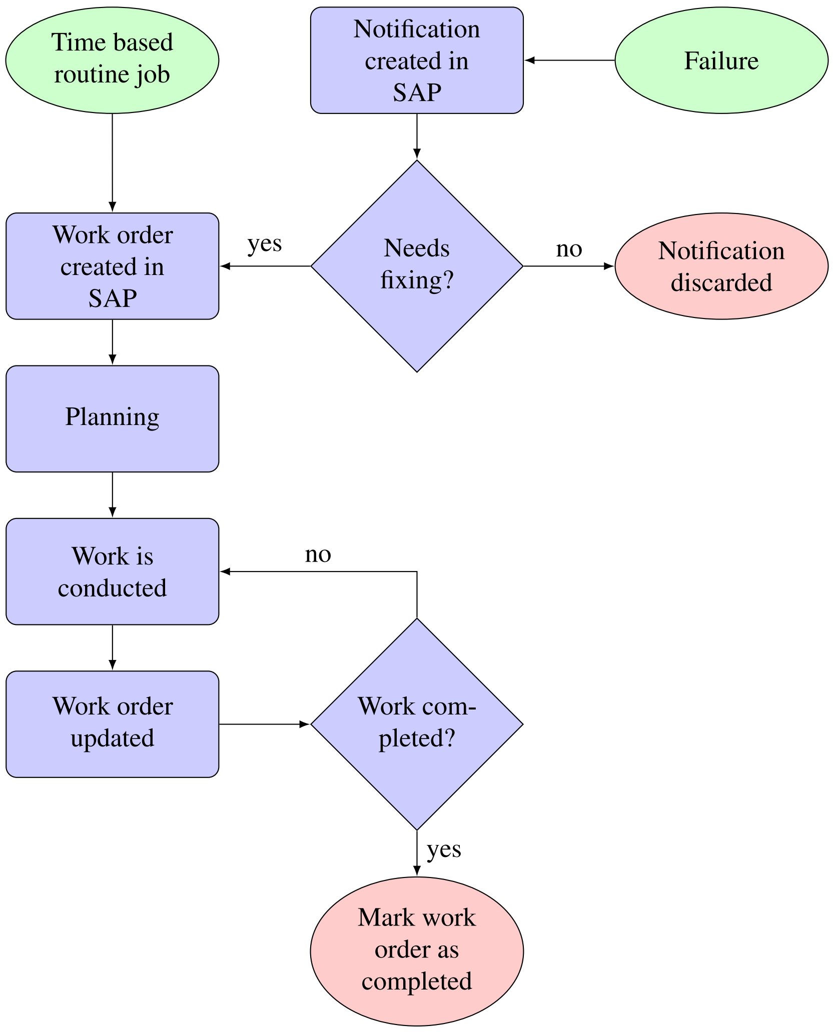


Figure 4.1: The life cycle of a work order.

4.2 The Importance of FMCs

The FMCs are not significant for Aker BP in the day-to-day operational work, but they are important for long-term statistics on equipment problems. Statistical analyses are conducted using FMCs, in order to identify the leading causes of failure, the types of equipment failure that are costing the company the most money, and in which aspects the company can improve. Work orders missing an FMC are not valuable in these statistical analyses, as their type of failure is not explicitly given. Consequently, Aker BP is interested in being able to automatically identify and predict the FMC of these work orders. Such a classifier would need to have an accuracy of at least 80%, in order to precisely classify FMCs so that they are applicable for statistical analysis. It is also beneficial for Aker BP to be able to automatically suggest relevant FMCs for their operators when they are filing a work order. This would guide them in the process of assigning the correct FMC, and could potentially mitigate confusion and reduce the frequency of mislabeling. Such a system does not require the same predictive performance as an automatic classifier, since human interaction is required to actually select the correct FMC from a set of suggestions.

Chapter 5

Data

In this study, a restricted real-world dataset provided by Cognite (on behalf of Aker BP) has been used. The dataset consists of 20 163 maintenance and repair requests, also called work orders. Work orders are filed for any work that needs to be done on a platform – ranging from changing a light bulb to repairing a sub-sea pump. An FMC is manually assigned by an inspector or operator when a work order is filed. The code indicates the type of failure that led to the work order. FMCs are exclusive; a work order may only be assigned a single code. The dataset used in this study is a subset of all work orders, and consists of those that are concerned with actual failures. A subset of FMCs that concerns actual failures is listed in Table 5.1.

Table 5.1: Subset of FMCs (10 of 40) with description and count. Listing the five largest and smallest classes, ordered by size. The full table can be found in Table B.2 in Appendix B.

Code	Description	Count	% of total
OTH	Other	1 178	18.50%
ELU	External leakage utility medium	466	7.32%
SER	Minor in-service problems	414	6.51%
SPO	Spurious operation	413	6.49%
FTF	Fail to function on demand	383	6.02%
...	...	...	...
VLO	Very low output	4	0.06%
FRO	Fail to rotate	4	0.06%
FTR	Fail to regulate	2	0.03%
STP	Fail to stop on demand	1	0.02%
FDC	Fail to disconnect	1	0.02%

5.1 Data Source

The majority of the data used in this research originates from a database export from a system called SAP. Alvheim FPSO uses SAP for creating and modifying work orders. The dataset was delivered by Cognite as a CSV-file, and it is approximately 68 MB. Cognite also has an API, where more information about work orders can be fetched. 42 MB worth of data was exported from the API, before it was joined with the original dataset.

5.2 Dataset Analysis

In this section, the dataset at hand is analyzed and described in detail. Key features and characteristics of the dataset are presented and discussed. The dataset poses several challenges for machine learning in terms of the amount of data, the number of target classes, the class distribution, and the general quality of the data. These characteristics must be understood and addressed in order to perform a sensible analysis.

5.2.1 Amount of Data

The initial dataset consists of 20 163 rows of work orders from different categories, where 4 737 are missing an FMC. Due to an *M:N* relationship between *CodeGroup* (described in Section 5.2.2) and FMC, the dataset has a total of 3 000 possible target classes. The dataset does not contain all of these combinations, and after removing rows missing *CodeGroup* and FMC, there are 372 distinct target classes left. This proved to be a very challenging classification task, and Cognite decided that we should rather focus on the *Failure mode* category. This reduced the number of target classes drastically. After more pruning (described in Section 6.1), the final dataset consists of 4 960 work orders with 29 distinct FMCs.

5.2.2 Features

The feature set of the dataset originally consists of 343 features, ranging from time stamps and identifiers automatically set by the ERP system – to textual descriptions manually entered by an operator. In other words, a work order is made up of a diverse set of features which includes nominal, ordinal and interval variables. Examples of the various feature types found in the dataset:

Nominal The feature *Enteredby* indicates the person that filed the work order. This value is limited to the persons working on the platform.

Ordinal The feature *Priority* assigns a numerical value of 1-4 to a work order. This value suggests the level of urgency, where 1 is high priority and 4 is low priority.

Interval The feature *CreatedOn* indicates the date of which the work order was filed. These values are on the format *YYYY–MM–DD HH:ss*

Since the feature set has a cardinality of 343, it would be inconvenient to give an in-depth description of all of them. Instead, the most significant features along with their relationship is presented in the list below. A complete list of the features can be found in Table B.1 in Appendix B.

FailureModeCode (FMC). This feature is the classification target. An FMC is a code indicating the type of equipment failure that led to the work order. Examples of FMCs and their description is shown in Table 5.1. The FMCs and their distribution are further described in Section 5.2.3.

CodeGroup. A *CodeGroup* is a grouping of FMCs that briefly indicates the type of equipment that the work order concerns. For instance, a work order assigned with the group *EM000001* suggests that the failure is related to an Electric Motor. Further, assume that the FMC of the work order is *OHE (Overheating)*. It is then apparent that the work order is filed due to an overheating electric motor. There are a total of 50 unique *CodeGroups* in the dataset, all of which are in relation to a subset of the FMCs. The majority of the *CodeGroups* are linked to 5-20 different FMCs. There is an *M:N* relationship between FMCs and *CodeGroup* – a FMC may belong to multiple *CodeGroups*, and a *CodeGroup* may consist of multiple FMCs. For instance, Compressors (*CO000001*) may overheat as well, so this *CodeGroup* is also in relation with *OHE*. *CodeGroup* is not present in work orders lacking an FMC, meaning it must be predicted as well.

CatalogProfile. A *CatalogProfile* is a code indicating a grouping of *CodeGroups*. The relationship between *CatalogProfile* and *CodeGroup* is *1:N*. In reality, the ratio is close to *1:1* as there are few *CatalogProfiles* related to multiple *CodeGroups*. This means that the *CodeGroup* of a work order can be accurately predicted (with $\approx 95\%$ accuracy) given the *CatalogProfile*.

FunctionalLocation. The *FunctionalLocation* of a work order is a specific identifier referring to the involved equipment. *FunctionalLocation* follows standardized protocols that are used on the platform, and it is the key to many of the results achieved in this study. The structure and semantics of these identifiers are thoroughly explained in Section 5.3.

Text. The *Text* of a work order is a manually entered text value. It often describes the failure, the measures that have been taken, the work that remains, and the equipment involved. They vary from being empty or a few words, to lengthy semi-structured documents. The longest text consists of 1 276 words, while the average length is 88 words (577 characters). Operators incrementally update the text as work related to the work order is conducted. The characteristics of the *Text* feature are further discussed in Section 5.2.4.

Three work orders are shown in Table 5.2. They are limited to their most significant features as described above. In the first example, the work order is concerned with a hydraulic leak. The FMC (*ELU*) and *CodeGroup* (*VA000001*) suggests that this is filed due to a *Valve* with *External leakage (utility medium)*, and the text feature confirms this with a more specific description. The second work order is also concerned with a valve, however, in this case no textual description is provided. In the last example, it is apparent that the language of the texts vary.

Table 5.2: Example of truncated work orders limited to their most significant features.

FMC	CodeGroup	CatalogProfile	FuntcionalLocation	Text
ELU	VA000001	113	NOAF-50XV0144	Hydraulic leak …
SPO	VA000001	113	NOAF-23FV5250	–
FTF	IP000001	86	NOAF-97XY7532	Spjeld virker ikke …

5.2.3 FMC Distribution and Challenges

As mentioned above, the final dataset includes 29 distinct FMCs after the preprocessing phase. This is a drastic reduction from the original dataset, and the reasoning behind this decision is discussed in Section 6.1.1.

The total of 29 FMCs poses a major challenge in the supervised classification task. On average, each FMC has $\frac{4\,960}{29} \approx 171$ representative work orders. This could arguably be considered an insufficient amount of data for the learning task, as data is crucial in order to train an accurate classifier. The average work order distribution is however rather misleading, as the classes are imbalanced and the distribution is skewed. In Figure 5.1, the distribution of the final 29 FMCs is shown. There are relatively huge classes such *ELU*, *SER*, and *SPO*, as well as seemingly insignificant classes such as *IHT*, *PDE*, and *TEX*. This imposes another challenging problem; the smaller classes will be underrepresented in the training phase. This is a problem due to the fact that general classifiers often tend to favor huge classes by design, leading to inaccurate models with regards to the small classes [66].

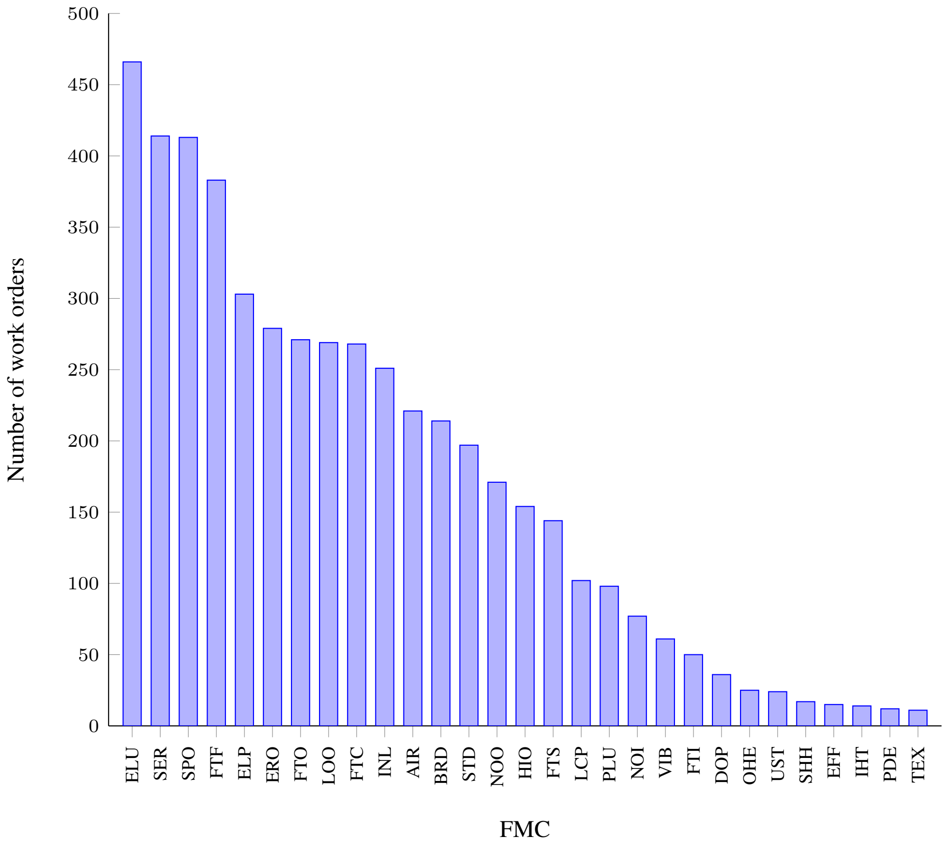


Figure 5.1: The distribution of FMCs in the final preprocessed dataset.

The effect of class imbalance when training a Random Forest classifier can be seen in Figure 5.2. This is a confusion matrix showing the actual class versus predicted class for each of the FMCs, where the row index indicates the actual class and the column index indicates the predicted class. Each row has been normalized such that each value is replaced by itself over the sum of the values in the row – i.e., the percentage of the total predictions for that class. This percentage is represented by the color brightness of each cell. A bright cell represents a high percentage and vice versa. Figure B.1 in Appendix B shows a more detailed version of Figure 5.2 with all the percentages included.

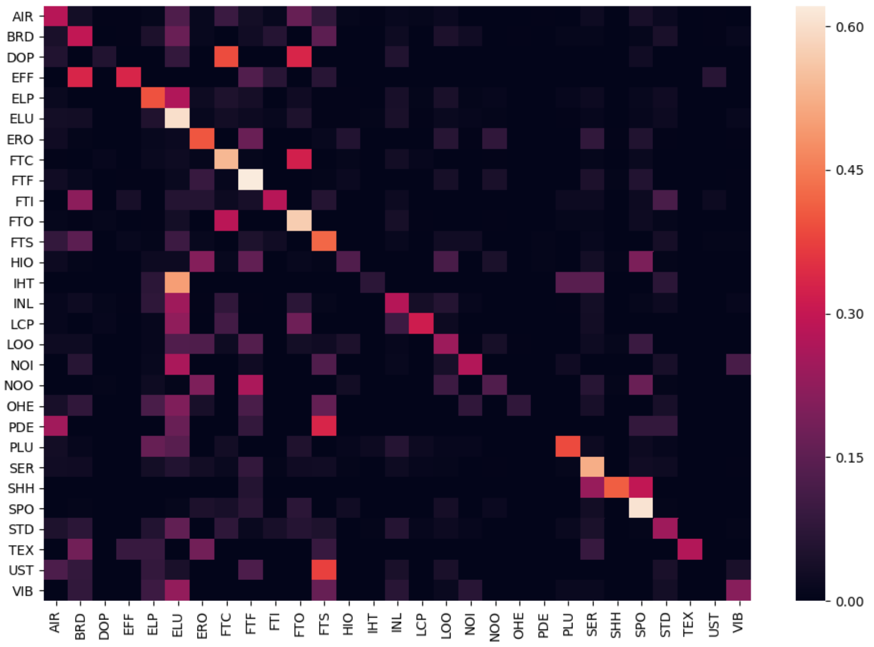


Figure 5.2: Row normalized confusion matrix based on the predictions of a Random Forest classifier.

By inspecting the matrix, it is evident that small classes (see Figure 5.1) are misclassified to a far greater extent than the big classes. For instance, the biggest class, *ELU*, is relatively accurately predicted, while *PDE* is not predicted correctly once. The *ELU* prediction column is bright compared to the others, suggesting that a lot of work orders are misclassified as *ELU*. This implies that work orders belonging to the small classes are often predicted to the big classes. That is the case for *IHT*, where the majority of the associated work orders are classified as *ELU*. It is also interesting to see how some of the classes correlate. It can be seen by cell [FTO, FTC] and [FTC, FTO] that the classes *FTO (Fail to Open)* and *FTC (Fail to Close)* are often mistaken for one another. This suggests that these classes are highly correlated in terms of their features.

Table 5.3 gives a more detailed overview of the data displayed in Figure 5.2. It shows the TPs of each FMC, both as a raw number and as a percentage of the total number

Table 5.3: FMCs along with their TPs (raw number and % of the class instances) resulting from the predictions of a Random Forest classifier. The class that the FMC is most frequently misclassifed as is also included.

FMC	TP	% TP	Most misclassified as
AIR	62	28%	FTO (16%)
BRD	63	29%	ELU (17%)
DOP	2	6%	FTC (39%)
EFF	5	33%	BRD (33%)
ELP	120	40%	ELU (27%)
ELU	280	60%	ELP (5%)
ERO	112	40%	FTF (17%)
FTC	144	54%	FTO (32%)
FTF	238	62%	ERO (9%)
FTI	14	28%	BRD (22%)
FTO	155	57%	FTC (28%)
FTS	61	42%	BRD (15%)
HIO	20	13%	ERO (21%)
IHT	1	7%	ELU (50%)
INL	70	28%	ELU (25%)
LCP	32	31%	ELU (23%)
LOO	65	24%	ERO (13%)
NOI	21	27%	ELU (26%)
NOO	22	13%	FTF (26%)
OHE	2	8%	ELU (20%)
PDE	0	0%	FTS (33%)
PLU	38	39%	ELP (16%)
SER	217	52%	FTF (8%)
SHH	7	41%	SPO (29%)
SPO	251	61%	FTO (7%)
STD	48	24%	ELU (16%)
TEX	3	27%	BRD (18%)
UST	0	0%	FTS (38%)
VIB	13	21%	ELU (23%)

of class instances. The class that each FMC is most frequently misclassified as is also provided, including the percentage of misclassification in parentheses. For instance, *AIR* was predicted correctly 62 times, which constitutes 28% of the total number of predicted *AIR* instances. The instances were most frequently misclassified as *FTO* (16% of the instances).

In 9 out of 29 cases, *ELU* is the FMC most instances are misclassified as. This suggests that *ELU* is a general class in terms of its feature values, causing a significant overlap with the other FMCs. There are cases where the majority class of the predictions differs from the actual class. This is true for *VIB*, *UST*, *PDE*, *OHE*, *NOO*, *IHT*, *HIO*, and *DOP*, all of which can be found in the bottom 16 with regards to the number of representative work orders (see Figure 5.1). The majority class is in these cases *ELU*, *FTS*, *FTF*, *ERO*, or *FTC*, which are all found on the other end of the spectrum. This further underlines the effect of class imbalance.

5.2.4 Data Quality

The overall quality of the data can be considered moderate in terms of structure. There are however several aspects of the dataset that poses problems for supervised machine learning. This includes varying quality of textual data, missing and duplicate data, and mislabeled data. In this section, each of these challenges are presented and discussed in relation to the dataset.

Manually Entered Text

As mentioned in Section 5.2.2, the *Text* feature of the work orders is a manually entered textual description of steps taken in the process of fixing the failure. These texts are of varying quality in terms of their content, structure and length. Some of the texts are well structured and encapsulate important aspects of the failure, while others are significantly less profound. There are several problems related to the content of the texts that make them hard to utilize in the machine learning task. Several languages such as Norwegian, English and Danish are used in the text field, and sometimes the records are multilingual, meaning that they contain multiple languages. The sentences that make up the texts are often grammarless and shortened, and contains industrial terms, abbreviations, identifiers and jargon. These problematic characteristics of the texts, make feature extraction using natural language processing a very challenging task. An example of a typical text:

”Damper not closing 27.09.2010 23:00:06 JOHN DOE (JDOE) damper not closing during test, either from SAS or local 29/9 smørt opp tregt spjeld, testet x10, nå OK.”.

In this example, it can be seen that multiple languages are used (Norwegian and English), as well as industrial terms and abbreviations such as *SAS*. *JDOE* is an unique identifier referring to the person John Doe.

Missing and duplicate data

Feature values are missing throughout the entire dataset. The initial dataset without preprocsessing has a sparsity of 67.34%. This is assuming that features consisting of uniform values are treated as missing values. The reasoning behind this assumption is that uniform values cannot be used to differentiate between classes in the classification task. In other words, 67.34% of the feature values are either non-existent or uniform. It is not completely known why that many values are missing, but a fair assumption is that the absence of some data is due to unmeasured or missing information about the failures. This does not have a detrimental effect on the classification task since many features have a coverage of 100%, but it is still important to decide what to do with the missing values.

The dataset also consists of duplicate features that are encoded versions of one another. A total of 14 features out of the original 343 are simply duplicates of other features. These duplicates are present as a result of how the original dataset was constructed. The original dataset is a joint dataset made up of smaller datasets. Some of these smaller datasets have features in common, resulting in an overlap of features in the joint dataset. It is important to address this issue, since a feature that is essentially the same as another feature (*1:1* mapping) are indirectly weighted double by for instance naïve Bayes if both are included [43]. The approach used to deal with missing and duplicate data is further explained and discussed in Section 6.1.2.

Mislabeled data

As the FMCs are manually entered and assigned by an operator, it is safe to assume that errors and mislabeling occurs to some extent. There are 29 distinct FMCs in the final preprocessed dataset concerned with failure modes. A few of the FMCs have similar denotations, such as *Fail to Open* and *Fail to Close*. Based on the predictions of the Random Forest classifier depicted in Figure 5.2 from Section 5.2.3, it is evident that some of the FMCs are highly correlated. A fair assumption is that in some cases, the operators might be confused as to which FMCs to use. This might result in mislabeling, or simply taking the easy way out and assigning the order either *OTH (Other)* or *UNK (Unknown)*. A fact that suggests this, is that the single biggest class is *OTH* as shown in Table 5.1. It might also be the case that different operators have their own subjective interpretation of the FMCs, leading to an overlap of the classes. In an analysis where 10 work orders were examined, one sample was mislabeled. The content of the work order explicitly describes a door that fails to close, but it was assigned *FTO (Failed to open on demand)* instead of *FTC (Fail to close on demand*). How frequently mislabeling occurs can not be concluded from this experiment alone, however, it proves that mislabeled data is present in the dataset.

5.3 Encapsulated Data

The feature set includes *FunctionalLocation*, a feature that specifies an equipment tag. At first glance it looks like an arbitrary sequence of alphanumeric characters, but it is in fact an encoded reference to the specific equipment that a work order concerns. Equipment tags at the platform have been issued using two well-established coding systems: SFI Group System [73] and NORSOK Z-DP-002 [60]. The SFI Group System is used for older hull equipment, while the rest of the tags are issued according to the NORSOK standard. In the preprocessed dataset, 15% of the work orders are issued using SFI and 85% using NORSOK. Features encoded in the NORSOK equipment tags are:

Platform. Specifies on which platform the equipment resides.

System. Logical grouping category of the parent systems (e.g. *Oil Storage*).

Equipment type. Logical grouping category of the equipment types.

Equipment id. The first level in the hierarchy related to a physical asset.

Subunit. Unit within the equipment boundary defined by the equipment ID.

Maintainable item. Lowest level within the equipment boundary defined by the equipment ID.

Features encoded in the SFI equipment tags are:

Platform. Specifies on which platform the equipment resides.

SFI 3-digit group system. A ship/rig is divided into 10 primary groups (1st digit, e.g. 7 – Machinery main components). Each primary group consists of 10 secondary groups (2nd digit, e.g. 3 – Compressed air systems). Finally, each secondary group is divided into 10 tertiary groups (3rd digit, e.g. 1 – Starting air systems). Each digit represent a step down in the hierarchy.

SFI Detail- & material code. Not standardized, but used to define individual components in a system.

How these features are extracted will be presented in Section 6.3.

Chapter 6

Methods

Our method for training an FMC classifier is divided into three steps. First, we do basic data preprocessing such as class filtering, class grouping and some natural language processing. Secondly, the data is enriched by utilizing domain knowledge and data from external sources. Finally, models are trained using naïve Bayes, Random Forest, Gradient Boosting Machine and Stacked Ensemble. Since our classification algorithms are able to handle the amount of features present in our dataset, using other feature selection techniques is not relevant in this study.

6.1 Dataset Preprocessing

Preprocessing is one of the measures taken in order to deal with the challenges of the dataset that were presented in the previous section. The final preprocessed dataset used in our experiments differs a lot from the initial dataset. This is a result of many steps of data preprocessing, which will be explained in this section.

6.1.1 Filtering Out Irrelevant Rows

The original dataset consists of 20 163 rows of work orders from different categories, where 4 737 are missing an FMC. The *M:N* relationship between *CodeGroup* and FMC makes the classification task very difficult with 372 different classes. A work order belongs to one of the following categories: (1) Attribute (2) Task (3) Decision (4) Event (5) Root cause (6) Results(def) (7) Activity QM (8) Defect type (9) Activities (10) Maint.

items (11) Failure mech (12) Coding (13) Defect loc. (14) Dec. SPM (15) Defect - SPM (16) Effort - SPM (17) Activity SPM (18) Insp verdict (19) Failure mode. The category is decided by the combination of the *CodeGroup* and the FMC of a work order. Because of the high number of target classes, Cognite decided that we should rather focus on the *Failure mode* category. This left a total of 9 721 rows distributed among 329 classes. However, each row has a *CatalogProfile* that now maps almost *1:1* to the *CodeGroup*, which means that only the FMC must be predicted. This reduces the number of classes to 46.

The code group *NOSPEC01* is used when a work order is missing a *CatalogProfile* or when the FMC is not specified (0100), and should thus be discarded. The same goes for *OTH (Other)* and *UNK (Unknown)*, which are removed due to the fact that these classes symbolize that the operator was unable to find a fitting FMC. This leaves us with a total of 4 999 work orders and 43 distinct FMCs.

FMCs with less than ten representative work orders are also removed. This decision is based on experiments using oversampling, undersampling and upsampling with synthetic samples. The sampling techniques had zero or negative impact on the final results, implying that there is insufficient data for these FMCs. This leaves us with the total of 29 distinct FMCs. Properties of the dataset during the filtering are listed in Table 6.1.

Table 6.1: Number of records and target classes after steps of filtering.

Step	Action	# Records	# Classes
1	All records	20 163	372
2	Keeping failure mode records only	9 721	46
3	Removing the *CodeGroup NOSPEC01*	6 364	45
4	Removing *OTH* and *UNK*	4 999	43
5	Removing classes with $<$ 10 records	4 960	29

Highly correlated *CodeGroups* are merged together. This is done in order to increase classifier performance when predicting *CodeGroup* based on *CatalogProfile*. The following groups are assembled:

Fire and Gas Detectors Composed of *FD (Fire Detectors)*, *GD (Gas Detectors)* and *FG (Fire and Gas Detectors)*.

Input Devices Composed of *ID (Input devices)* and *IP (Input devices)*.

Basic natural language processing operations are applied to the *Text* field of the work orders. Punctuation characters are removed, Norwegian special characters are encoded, and all text is made lowercase. These operations are done in order to prepare the data for tokenization and feature extraction using a predefined domain term dictionary and TF-IDF (described in Section 6.3.3 and 6.4, respectively). Stemming and stop word removal were initially tested, but later discarded due to the *Text* field being written in different languages.

6.1.2 Filtering Out Irrelevant Columns

In order to reduce the number of features in the dataset, irrelevant columns are discarded. The method used to reduce the column dimensionality lowered the sparsity of the dataset substantially. Since a column with only one distinct value for all rows provides zero knowledge to the classification task, these columns are treated as missing when calculating the sparsity. Table 6.2 lists the steps taken when filtering out irrelevant columns.

Table 6.2: Number of columns after steps of filtering.

Step	Action	# Columns	Sparsity
1	Initial dataset	343	67.34%
2	Remove empty columns	210	46.60%
3	Remove columns with only one distinct value	141	20.56%
4	Remove duplicate columns	127	15.72%
5	Remove columns with a density of $< 80\%$	108	0.21%

The initial dataset consisted of 343 columns and had a sparsity of 67.34%. An analysis of the dataset showed that many of the columns are either completely empty, or consist of the same value. The first step was therefore to remove all empty columns. This pruned 133 columns, and the sparsity went down to 44.60%. Removing columns with only one distinct value filtered out 69 more columns, and the sparsity was reduced to 20.56%. A correlation analysis between the columns, proved that many of them had exactly the same information. In many cases one column held the value, while another column held an id of the value. 14 new columns were removed by pruning duplicate columns. As listed in Table B.1 in Appendix B, there are very few columns with a density between 5% and 80%. Initial experiments showed that the effect of including the features with a density of $< 80\%$ had close to zero impact on the performance of the classifiers. Based on this analysis, all columns with a density of $< 80\%$ were pruned. The final dataset has a total of 108 features, with a sparsity of 0.21%. Filtering out irrelevant rows and columns lowers the number of data points from 6 734 442 to 535 680.

6.2 Baseline Configuration

The baseline models used in the experiments are trained using the preprocsessed dataset, meaning that irrelevant rows and features have been filtered out. In other words, the baseline classifiers will be trained on a dataset with the following properties:

- 4960 records (work orders)
- 108 features
- 29 target classes (FMCs)

These models will be used to assess the effect of additional features that are extracted using domain knowledge and TF-IDF. Random Forest, Gradient Boosting Machine, Stacked Ensemble and naïve Bayes are the algorithms that will be used to train the baseline models. These algorithms are also used in all the other experiments. The choice of algorithms along with the selection of hyperparameters (configurations) is discussed in Section 6.6.

6.3 Utilizing Domain Knowledge

In the second step of data preprocessing, domain knowledge is incorporated. The utilization of domain knowledge and enrichment of the data consist of three separate methods. In the first method, applicable data is extracted from *FunctionalLocation*. The second method uses external information about the equipment tags to construct new features. Domain knowledge used in the first two methods originates from descriptions and manuals received from Aker BP. In the third method, knowledge about each of the FMCs is exploited in order to extract useful information from the *Text* field.

6.3.1 FunctionalLocation Breakdown

FunctionalLocation consists of encoded values as previously described in Section 5.3. A series of string operations are used in order to decode and extract additional features from the values. After each step, the extracted part of the *FunctionalLocation* is removed from the string in order to simplify further extractions. Examples of *FunctionalLocation* breakdowns for both SFI and NORSOK are given in Table 6.3a and 6.3b, respectively. The approach of the feature extraction is as follows:

1. Extract platform info
2. Identify if *FunctionalLocation* is SFI or NORSOK
3. If NORSOK:
   (a) Extract component
   (b) Extract subunit
   (c) Extract system
   (d) Extract equipment type
   (e) Extract equipment id
4. If SFI:
   (a) Extract primary, secondary and tertiary group
   (b) Extract detail code
   (c) The meaning of the remaining part of *FunctionalLocation* is unknown

Table 6.3a: *FunctionalLocation* breakdown for NORSOK standard.

FunctionalLocation	Platform	System	Eq. type	Eq. id	Subunit	Component
NOAF-24VD003	NOAF	24	VD	003	-	-
NOIA-70-BS-32020-L42B	NOIA	70	BS	32020	L42	B
NOAF-97XY10439A	NOAF	97	XY	10439	-	A
NOAF-63ACD123-M01	NOAF	63	ACD	123	M01	-

Table 6.3b: *FunctionalLocation* breakdown for SFI standard.

FunctionalLocation	Platform	Gr.[1]	Gr.[2]	Gr.[3]	Detail code	Unknown
NOAF-662.27D-2-DE006	NOAF	6	62	662	27D	2-DE006
NOAF-713.24C-1	NOAF	7	71	713	24C	1
NOAF-769.21-1	NOAF	7	76	769	21	1

[1]Primary group
[2]Secondary group
[3]Tertiary group

6.3.2 Extracting Additional Information From FunctionalLocation

Additional information about the *FunctionalLocations* exists in Aker BP's internal ERP-system. This includes further grouping of the tags, along with a short textual description of each asset. An export of this data has been made available for use in this study. In the ERP-system, a *FunctionalLocation* belongs to exactly one of the following groups: (1) Cables (2) Electrical Equipment (3) Fire and Gas (4) Instrument (5) Junction Box (6) Lifting Lug (7) Line (8) Manual Valve (9) Master Equipment (10) Miscellaneous (11) Signal (12) Special (13) Telecom (14) Unknown. Based on this information, a new feature *Tag_type* is constructed. Each *FunctionalLocation* belongs to exactly one of the groups above. The textual descriptions of the tags are also extracted, and the values are stored as a feature named *Tag_description*.

The new features are added to the original dataset by the following procedure: A dictionary D is constructed from the additional dataset and filled with key-value pairs on the form $\{FunctionalLocation : (Tag\_type,\ Tag\_description)\}$. For each of the work orders in the original dataset, D is queried with the *FunctionalLocation* to extract the corresponding values of *Tag_type* and *Tag_description*. The values are then appended to the original dataset as separate fields.

6.3.3 Domain Term Dictionary & Binary Feature Vectors

This section describes our approach to extracting features from the unstructured *Text* field of the work orders using domain knowledge. It is a two-step approach that consists of manually defining a domain term dictionary of representative terms, followed by adding these terms as features using the Bag-of-Words model with a binary weighting scheme.

Constructing the Domain Term Dictionary

By manually inspecting the data at hand, several descriptive domain terms for each of the FMCs were identified. The manual process of extracting these terms consisted of thoroughly examining the content and structure of the *Text* field of the work orders, in light of the domain. By grouping the work orders based on their FMC, representative and general terms for each of the classes could be identified. Terms for the FMCs were selected on the basis of two criteria:

(1) How frequently does the term appear in work orders that are assigned the FMC?
(2) Based on our understanding of the domain, how relevant is the term in context of the FMC?

In other words, the terms representing an FMC appears frequently among its associated work orders, and are by our understanding related to, or a direct cause of, the failure that the FMC represents. For instance, the term *block* appears often in work orders assigned with *PLU*, and it may also be considered a direct cause of the filing of these work orders, as this FMC is used when equipment is plugged or choked. The output of this process were 217 distinct terms that represents the 29 FMCs. A term may be closely associated with one or more of the FMCs. Examples of FMCs and their related terms are shown in Table 6.4. The full list of terms can be found in Table B.3 in Appendix B.

Table 6.4: Example of FMCs and related domain terms.

FMC	Terms
INL - Internal Leakage	[internal, leakage, sweat, oil, diesel,. . .]
LOO - Low Output	[low, output, pressure, flow, pump,. . .]
PLU - Plugged/Choked	[plugged, choked, drain, flow, block,. . .]

The exact number of terms (217) was not an obvious choice, but rather a consequence of how many representative terms we were able to identify. This is most likely not an optimal number of domain terms. Adding more terms might help discriminate between the FMCs to a greater extent, but it may also cause the models to overfit. Decreasing the number of terms could reduce a potential overfitting caused by the 217 terms, but it could also reduce the performance boost.

Calculating Binary Feature Vectors

The domain term dictionary is used in order to extract additional features from the *Text* field. Our approach results in Boolean features, where the values indicate whether or not the term is present in the work order. This is achieved by first defining the vocabulary V as the set of domain terms, $|V| = 217$. Each of the documents (work order texts) are then represented as a Bag-of-words. The Bag-of-words are then used to calculate and represent the document as a feature vector with binary term weights, where the vector space model is defined by V. This results in binary feature vectors of length $|V|$, that represents the presence and absence of domain terms in the corresponding document. The terms are added as features to the dataset, where the values are given by the feature vectors.

Binary weighting of terms was chosen because it outperformed term frequency as a weighting scheme in all the conducted experiments. TF-IDF was not used for weighting due to the fact that decisions trees, the base learners for most of the tested techniques in this study, are invariant under all monotonic transformations of individual ordered variables [11]. In other words, node splits based on term features will be the same regardless of whether term frequency or TF-IDF is used.

6.4 TF-IDF

TF-IDF was employed in order to compare the use of manually selected domain terms to terms extracted using text mining. There are 20 741 unique terms when looking at all the text fields in the dataset. It is unattainable to use that many features in a classifier, and therefore only the most important terms were selected as features. Since the defined domain term dictionary consists of 217 terms, a decision was made to keep the 217 highest TF-IDF weighted terms as well. This was done in order to get an equal base when comparing the performance of classifiers incorporating TF-IDF terms and domain terms. The terms were selected by averaging the TF-IDF weights for each term, and only keeping the 217 highest average weighted terms. Table B.4 in Appendix B lists the highest weighted TF-IDF terms.

The documents used in TF-IDF were constructed in two different ways. At first samples were grouped by their target class, and each document consisted of the joined text from each group. Due to the class imbalance, the text was much longer in the larger classes and included a lot of different words. Since the large documents had many more terms, normalizing the term frequency was necessary. This led to higher term frequency scores on terms in the minority classes, and the extracted terms proved to be of little to no value in the classification task. In the second approach, the text in each sample was treated as individual documents. This reduced the difference in text length substantially. Adding the extracted terms to the feature set resulted in a perceptible improvement in the classification task.

A minimum and maximum document frequency can be set when applying TF-IDF. They specify how large proportion of the documents can contain a term in order for the term to be included in the results. The weights were decided by doing experiments where the output terms were manually inspected. Ultimately, we decided not to set a minimum document frequency, and a maximum document frequency of 5%. As the text fields are of variable length, the term frequency was normalized using euclidean norm (ℓ^2 norm).

There is an intersection of 29 terms between the domain terms and the TF-IDF terms. These terms are *aapne, aapner, batteri, close, closed, detektor, diesel, drain, fast, feedback, flow, fuel, instrument, intern, leak, lekasje, level, nivaa, olje, open, problemer, sensor, service, start, stenge, stengt, tar, trykk* and *virker*.

The selection of the top features could have been done differently, since picking the ones with the highest average TF-IDF weights might have excluded discriminating terms. Assume that a term is actually very important for a small class and has a high TF-IDF weight for samples in that class. If the term is mentioned in a few other samples from larger classes with much text, the TF score will negatively affect the TF-IDF score for the said term in those documents. This will drastically reduce the average TF-IDF weight of the term that in fact would have been very discriminating for the small class. This is a problem due to the varying text lengths and the imbalanced classes.

The representation of terms in the feature set were a binary value indicating if a term was present in the text or not. Changing this to a normalized term frequency could possibly have improved the results. A binary value was chosen in order to have the same basis as the features extracted using domain terms. A more sophisticated weighting scheme could have been applied on the domain terms instead of binary weighting the TF-IDF terms. Doing more experiments on parameter selection for TF-IDF might have had an impact on the results as well. We did not put much effort into optimizing TF-IDF, as the focus of the research was not centered around extracting features using NLP.

6.5 Environment

All of our experiments are conducted using H2O, an open source machine learning and predictive analytics platform [28]. An attractive feature of H2O is that it supports the direct use of categorical variables in tree-based algorithms. By using for instance *scikit-learn*, we would have had to *one-hot encode* or *label encode* the features first. One-hot encoding makes the feature space orders of magnitude larger, while label encoding imposes ordinal relationships that are not necessarily true.

H2O offers a feature called AutoML, which automatically trains and tunes several models using multiple algorithms with different hyperparameters on the same dataset. AutoML currently supports Random Forest, Gradient Boosting Machine, Deep Learning, Extremely Randomized Trees, Generalized Linear Modeling, and a Stacked Ensemble of all the models. We used AutoML to get an overview of how a large selection of candidate models performed on our dataset.

6.6 FMC Classification

As the task of classifying FMCs is a multiclass classification problem, there are several prominent techniques and algorithms that may be deployed. The choice of algorithm is dependent on the task at hand, meaning that it is often necessary to test multiple algorithms to see what best fits the problem.

H2O's AutoML was initially used to get an intuition of which algorithms worked best on our dataset. It is considered good practise to test multiple classification algorithms, and it contributes to get a broader assessment of the effects of incorporating additional features. A comparison of the different algorithms is listed in Table 6.5. These results are only meant to give an overview of the algorithms, and are not used as baseline. Naïve Bayes is not included in AutoML, but it was included in the experiments in order to have a probabilistic classifier as well. Three algorithms stood out after running AutoML, and we decided to employ these methods in further experiments. Thus, the following four algorithms were used and evaluated in this study:

1. Naïve Bayes (NB)
2. Random Forest (RF)
3. Gradient Boosting Machine (GBM)
4. Stacked Ensamble (SE)

Table 6.5: Comparison of algorithms using AutoML.

	Cohen's Kappa	Accuracy	Macro average F_1-Score
Stacked Ensemble	0.3794	0.4176	0.3003
Random Forest	0.3755	0.4133	0.3113
Gradient Boosting Machine	0.3638	0.4020	0.2914
Extremely Randomized Trees	0.3282	0.3728	0.2610
Deep Learning	0.2685	0.3146	0.1859
Generalized Linear Modeling	0.2417	0.2965	0.1377

A Random Grid Search was used in order to find the best hyperparameters for both Random Forest and Gradient Boosting Machine. The difference in performance using various hyperparameters turned out to be insignificant, and the best baseline results were very similar to the results from H2O's AutoML. The final hyperparameters used in the experiments are listed in Table 6.6.

Table 6.6: Hyperparameters after Random Grid Search.

	Random Forest	Gradient Boosting Machine
Number of trees	50	50
Maximum tree depth	10	13
Minimum samples in leaf node	1	30
n-bins (numerical values histogram)	10	20
Learn rate	-	0.1
Row sample rate per tree	-	0.8
Column sample rate	-	0.7
Score model interval	-	5
Distribution function	-	Multinomial

Chapter 7

Experiments and Results

In order to examine the effect of domain knowledge, this chapter presents experiments that are conducted using several supervised learning techniques. The experiments compare the performance of classifiers incorporating features extracted with domain knowledge and TF-IDF, with models trained using no form of extracted knowledge (baseline).

7.1 Evaluation Metric

Based on the class imbalance in the dataset and feedback from Cognite, it was preferable to use an evaluation metric that weights minority classes higher than larger classes. It should however not be weighted to the extent where each class contributes equally to the score, since the main goal is not to only correctly classify the minority classes.

Accuracy is a commonly used metric for evaluating classifiers. Unfortunately accuracy can be misleading when dealing with imbalanced classes, as it is very sensitive to class distribution [31]. Based on the accuracy paradox that was presented in Section 2.9.2 and the imbalanced dataset used in this project, we decided not to use accuracy as the primary metric. Other metrics such as Cohen's Kappa and F-Score punish classifiers harder for mislabeling small classes.

Cohen's Kappa was chosen as the primary metric over macro average F_1-Score, since macro averaging weights the F_1-Score of each class equally. Misclassified samples belonging to minority classes would then negatively affect the score more than desired. Cohen's Kappa has been used as a metric in many multiclass classification problems involving imbalanced datasets [12, 33].

We have also included accuracy and macro average F_1-Score in our results for a more comprehensive understanding of the performance, but Cohen's Kappa is the primary evaluation metric.

To ensure that all minority classes are represented in the testing set, we use 5-fold cross-validation. The final score presented is averaged from the five folds.

7.2 Baseline

Baseline results were computed using all the initial features, without utilizing domain knowledge. The results are presented in Table 7.1. Using Landis and Koch's interpreation of the Cohen's Kappa, the performance of the baseline classifiers can be considered as *Fair*, except for naïve Bayes. The naïve Bayes baseline classifier has a *Slight agreement* with the ground truth.

Table 7.1: Baseline results.

	Cohen's Kappa	Accuracy	Macro average F_1-Score
Stacked Ensemble	0.3800	0.4185	0.2982
Random Forest	0.3782	0.4165	0.3087
Gradient Boosting Machine	0.3664	0.4044	0.2902
Naïve Bayes	0.1616	0.1889	0.1463

7.3 Incorporating Extracted Features

This section presents the results achieved when extracted features were added to the dataset the classifiers are trained on. The results include experiments using features that were extracted using domain knowledge and features that were extracted using TF-IDF. Each table that presents the results are structured in the same way. It shows the performance of each of the four classifiers, with Cohen's Kappa, accuracy and macro average F_1-Score as evaluation metrics. The results are shown using abbreviations for the applied algorithms; SE (Stacked Ensemble), RF (Random Forest), GBM (Gradient Boosting Machine) and NB

(Naïve Bayes). Included in the tables are also the improvement from the baseline results, wrapped in parentheses.

7.3.1 Features Extracted From Structured Fields

In total, 14 different features were extracted from the *FunctionalLocation* field of the work orders. Since a *FunctionalLocation* is encoded using either the NORSOK or SFI coding standard, a maximum of nine features can be extracted per sample. No effort was made to find connections between features extracted from NORSOK and SFI. This implies that the extracted features only will contribute to finding correlations internally in the two coding scheme groups.

Experiment 1: Structured Fields

Table 7.2 shows the performance of the classifiers using the original features in addition to features extracted from *FunctionalLocation*. The performance metrics increased for all the classifiers, compared to the baseline results. Naïve Bayes had a higher relative improvement than the other algorithms, suggesting that the extracted features are good independent features. The increase in performance of the other three algorithms were far less, which indicates that some of the knowledge in the extracted features had already been learned from combinations of other features.

Table 7.2: Experiment 1: Performance of classifiers using the original features in addition to features extracted from structured fields using domain knowledge.

	Cohen's Kappa	Accuracy	Macro average F_1-Score
SE	0.3899 *(+0.0099)*	0.4277 *(+0.0092)*	0.3068 *(+0.0086)*
RF	0.4012 *(+0.0112)*	0.4378 *(+0.0213)*	0.3191 *(+0.0104)*
GBM	0.3894 *(+0.0230)*	0.4259 *(+0.0215)*	0.3286 *(+0.0384)*
NB	0.2081 *(+0.0471)*	0.2373 *(+0.0484)*	0.1732 *(+0.0269)*

7.3.2 Features Extracted From Unstructured Fields

In the following two experiments, features were extracted from the *Text* field of the work orders. The two approaches are distinct in the way that the features are extracted. TF-

IDF is used in the first approach in order to automatically extract key terms from the document corpus, while the second approach uses a predefined list of descriptive terms, that was manually assembled using knowledge about the domain. Both of these extended datasets were constructed using the method described in Section 6.3.3, where the features are appended using binary feature vectors that represents the absence or presence of each term in the work orders.

Experiment 2: TF-IDF

This experiment was conducted using a dataset with 217 additional term features extracted using TF-IDF. The results are shown in Table 7.3. They show that SE performs slightly better than RF in terms of Cohen's Kappa and accuracy, however, RF and GBM are superior with respect to macro average F_1-Score. All the algorithms had a significant improvement on all evaluation metrics compared to the baseline.

Table 7.3: Experiment 2: Performance of classifiers using additional features extracted with TF-IDF.

	Cohen's Kappa	Accuracy	Macro average F_1-Score
SE	0.4049 *(+0.0249)*	0.4423 *(+0.0238)*	0.3060 *(+0.0078)*
RF	0.4048 *(+0.0266)*	0.4421 *(+0.0256)*	0.3236 *(+0.0149)*
GBM	0.3872 *(+0.0208)*	0.4236 *(+0.0192)*	0.3225 *(+0.0323)*
NB	0.1892 *(+0.0276)*	0.2175 *(+0.0286)*	0.1666 *(+0.0203)*

Experiment 3: Domain Terms

In this experiment, 217 additional predefined domain term features were appended to the original dataset. The results are shown in Table 7.4. It is evident that the overall performance of the classifiers improved significantly compared to the baseline. This approach also performed better than the TF-IDF approach in every aspect, except for the GBM. The overall improvement suggests that the predefined domain terms are more descriptive and discriminating than the TF-IDF terms.

Table 7.4: Experiment 3: Performance of classifiers using additional predefined domain term features.

	Cohen's Kappa	Accuracy	Macro average F_1-Score
SE	0.4115 *(+0.0315)*	0.4482 *(+0.0297)*	0.3153 *(+0.0171)*
RF	0.4098 *(+0.0316)*	0.4470 *(+0.0305)*	0.3292 *(+0.0205)*
GBM	0.3797 *(+0.0133)*	0.4166 *(+0.0112)*	0.3206 *(+0.0304)*
NB	0.2070 *(+0.0454)*	0.2369 *(+0.0480)*	0.1778 *(+0.0315)*

7.3.3 Combination of Extracted Features

Four experiments were conducted where features from unstructured fields and features from structured fields were used in combination. Extracted structured features originates from the structured field *FunctionalLocation*. Experiment 4 uses these features in combination with the terms found using TF-IDF, while Experiment 5 use the extracted structured features in combination with features extracted using predefined domain terms. Experiment 6 combines features extracted from the unstructured fields, using both TF-IDF and domain terms. At last, all the three feature sets are combined in Experiment 7.

Experiment 4: TF-IDF and Structured Fields

Table 7.5 shows the results of the classifiers when both the features extracted using TF-IDF and the features extracted from *FunctionalLocation* were added to the dataset. The combination of the feature sets performs better than when using them separately. This suggests that both of the feature sets add value to the classification task, and that the knowledge they infer does not fully overlap.

Table 7.5: Experiment 4: Performance of classifiers incorporating additional features extracted using TF-IDF as well as features extracted from the structured field *FunctionalLocation*.

	Cohen's Kappa	Accuracy	Macro average F_1-Score
SE	0.4260 *(+0.0460)*	0.4611 *(+0.0426)*	0.3266 *(+0.0284)*
RF	0.4152 *(+0.0370)*	0.4510 *(+0.0345)*	0.3479 *(+0.0392)*
GBM	0.4128 *(+0.0464)*	0.4478 *(+0.0434)*	0.3458 *(+0.0556)*
NB	0.2138 *(+0.0522)*	0.2429 *(+0.0540)*	0.1837 *(+0.0374)*

Experiment 5: Domain Terms and Structured Fields

Table 7.6 shows the performance of the classifiers when both the features extracted using domain terms and the features extracted from *FunctionalLocation* were added to the dataset. This was the feature set that yielded the best performance for all the classifiers.

Table 7.6: Experiment 5: Performance of classifiers incorporating additional features extracted using domain terms as well as features extracted from structured fields.

	Cohen's Kappa	Accuracy	Macro average F_1-Score
SE	0.4515 *(+0.0715)*	0.4851 *(+0.0666)*	0.3641 *(+0.0659)*
RF	0.4454 *(+0.0672)*	0.4796 *(+0.0631)*	0.3665 *(+0.0578)*
GBM	0.4363 *(+0.0699)*	0.4700 *(+0.0656)*	0.3674 *(+0.0772)*
NB	0.2346 *(+0.0730)*	0.2651 *(+0.0762)*	0.2032 *(+0.0569)*

The classifiers using domain terms performs slightly better than the classifiers using TF-IDF. When they are combined with features extracted from *FunctionalLocation*, TF-IDF is outperformed by domain terms. This suggests that the classification value inferred from the domain terms does not overlap with the value inferred from structured features to the same extent as it does for TF-IDF.

Experiment 6: TF-IDF and Domain Terms

This experiment tests how all features extracted from unstructured fields perform in combination. Table 7.7 shows the performance of the classifiers when features extracted using both domain terms and TF-IDF were added to the dataset. The performance of the classifiers in this experiment is significantly higher than when the two feature sets are used independently.

Table 7.7: Experiment 6: Performance of classifiers incorporating additional features extracted using both TF-IDF and domain terms.

	Cohen's Kappa	Accuracy	Macro average F_1-Score
SE	0.4341 *(+0.0541)*	0.4685 *(+0.0500)*	0.3473 *(+0.0491)*
RF	0.4213 *(+0.0431)*	0.4579 *(+0.0414)*	0.3452 *(+0.0365)*
GBM	0.4138 *(+0.0474)*	0.4488 *(+0.0444)*	0.3380 *(+0.0478)*
NB	0.2106 *(+0.0490)*	0.2395 *(+0.0506)*	0.1769 *(+0.0306)*

Experiment 7: TF-IDF, Domain Terms and Structured Fields

Table 7.8 shows the results of the classifiers when all extracted features were added to the dataset. This includes features extracted from the *Text* field using both TF-IDF and domain terms, as well as the features extracted from the structured field *FunctionalLocation*. The results are similar to Experiment 6 that uses almost the same feature set, but without features extracted from structured fields. In Experiment 5, where only structured features and features extracted from unstructured fields using domain terms are applied, the results are significantly better. This may indicate that the feature set used in Experiment 7 has caused the classifiers to overfit. A more thorough discussion of this will be presented in the next chapter.

Table 7.8: Experiment 7: Performance of classifiers incorporating additional features extracted using both TF-IDF and domain terms, as well as features extracted from structured fields.

	Cohen's Kappa	Accuracy	Macro average F_1-Score
SE	0.4319 *(+0.0519)*	0.4665 *(+0.0507)*	0.3441 *(+0.0459)*
RF	0.4366 *(+0.0584)*	0.4714 *(+0.0549)*	0.3666 *(+0.0579)*
GBM	0.4046 *(+0.0382)*	0.4395 *(+0.0351)*	0.3398 *(+0.0496)*
NB	0.2291 *(+0.0675)*	0.2589 *(+0.0700)*	0.1891 *(+0.0428)*

7.4 Summary of Results

The difference in metrics between the baseline classifiers and the best performing classifiers with incorporated domain knowledge (Experiment 5) is illustrated in Figure 7.1. Cohen's Kappa has increased with ≈ 0.07 for all algorithms.

Table 7.9 shows the performance of the baseline classifiers and the classifiers from the experiment that yielded the best result. The last section of the table shows how much each metric has improved from the baseline to the best classifiers from Experiment 5. The relative improvement from the baseline is wrapped in parentheses. Cohen's Kappa shows that the performance of the ensemble algorithms has increased to *Moderate*, while the naïve Bayes classifier has improved to *Fair*. The fact that Cohen's Kappa has increased more than the macro average F_1-Score, implies that the classifier has a greater improvement on the larger classes than the smaller ones.

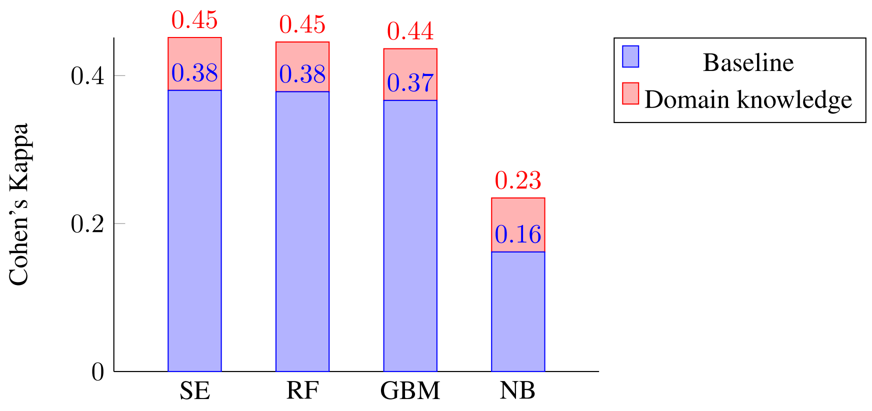


Figure 7.1: Cohen's Kappa for baseline compared to models with incorporated domain knowledge.

Table 7.9: Results from the baseline models and the best performing models from Experiment 5, along with the improvement.

	Cohen's Kappa	Accuracy	Macro average F_1-Score
Baseline			
Stacked Ensemble	0.3800	0.4185	0.2982
Random Forest	0.3782	0.4165	0.3087
Gradient Boosting Machine	0.3664	0.4044	0.2902
Naïve Bayes	0.1616	0.1889	0.1463
Domain knowledge (Experiment 5 – structured features + domain terms)			
Stacked Ensemble	0.4515	0.4851	0.3641
Random Forest	0.4454	0.4796	0.3665
Gradient Boosting Machine	0.4363	0.4700	0.3674
Naïve Bayes	0.2346	0.2651	0.2032
Improvement from baseline			
Stacked Ensemble	0.0715 (18.8%)	0.0666 (15.9%)	0.0659 (22.1%)
Random Forest	0.0672 (17.8%)	0.0631 (15.2%)	0.0578 (18.7%)
Gradient Boosting Machine	0.0699 (19.1%)	0.0656 (16.2%)	0.0772 (26.6%)
Naïve Bayes	0.0730 (45.2%)	0.0762 (40.3%)	0.0569 (38.9%)

Adding features extracted from unstructured fields using domain terms in addition to features extracted from the structured field *FunctionalLocation* to the feature set, yielded the best improvement for all the classification algorithms. All of the extra features used in the top performing classifiers were extracted using domain knowledge. When features extracted using TF-IDF were added in addition to the domain-driven features, the performance decreased slightly. This may indicate that the models have overfitted.

Figure 7.2 lists the feature importances of the best Random Forest classifier (from Experiment 5). Feature importance can easily be computed from a Random Forest because of the characteristics of decision trees. It is calculated from the relative influence of each feature. A feature has influence if it is selected during the splitting of a leaf node, and if it contributes to reducing the squared error over all trees.

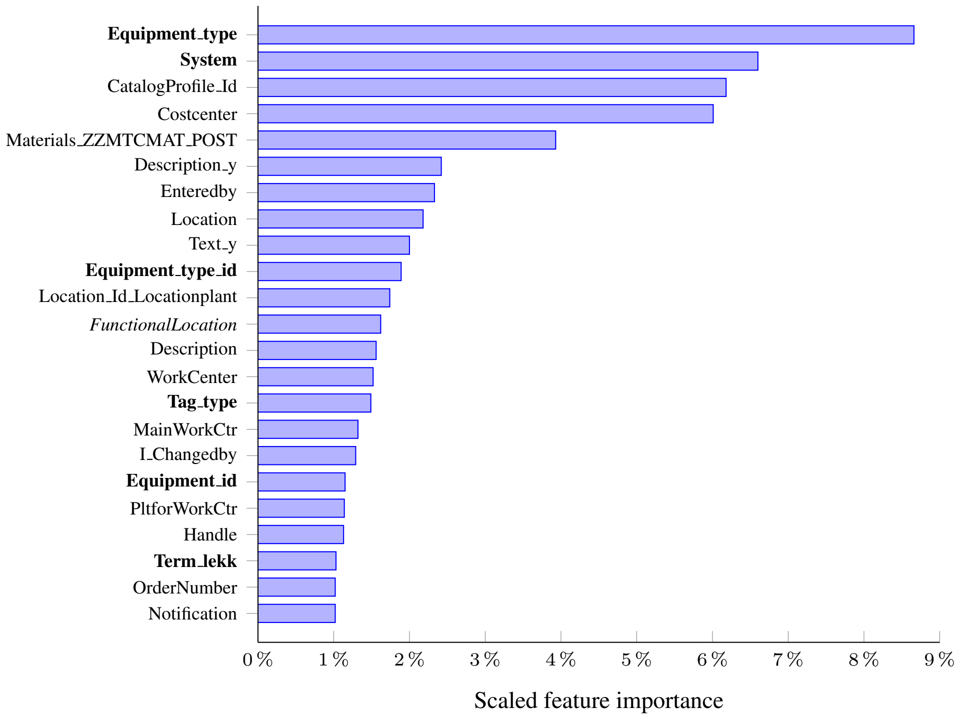


Figure 7.2: Features with a scaled importance of $> 1\%$ from the best performing Random Forest model with domain knowledge incorporated. The features with bold text were extracted using domain knowledge.

As shown in the figure, six of the extracted features are among the top 23 most important features in the improved model. The features that were extracted using domain knowledge constitute 38.97% of the total feature importance (100%). *Equipment_type*, *System*, *Equipment_type_id*, *Equipment_id* and *Tag_type* are all extracted from the structured *FunctionalLocation* field.

Chapter 8

Discussion

This chapter contains a discussion of the results and other takeaways from the project. The discussion is written with the research questions and related work in mind.

8.1 The Effect of Domain Knowledge

The idea of experimenting with feature extraction using domain knowledge, was the result of several unsuccessful approaches that yielded zero to very slight improvement in performance. It was further motivated by the fact that *FunctionalLocation* was discovered to be an encoded feature consisting of relevant information about the equipment involved in the work orders. The *Text* field was first utilized by extracting terms using TF-IDF, however, these terms were manually inspected and several of them were deemed unrepresentative with regards to the FMCs. This led to experimentation with manually extracted domain terms that represents the FMCs to a greater extent.

In this section, the effect of the various approaches are reviewed and discussed with basis in the experiments. From the conducted experiments it is evident that the features extracted using domain knowledge had a significant impact on the classifiers in terms of predictive performance.

8.1.1 Features Extracted From Structured Fields

In Experiment 1 where only features extracted from the *FunctionalLocation* field were added to the feature set, the naïve Bayes classifier had a much higher relative improvement than the other algorithms. Interpreting these results together with the naïve assumption that all features are independent of each other, suggests that the extracted features are valuable independently of other features. This means that a classifier can discriminate samples from different classes using the extracted features, without looking at features in combination. This is also supported by Figure 7.2 that was presented in the previous chapter, showing feature importances of the best Random Forest model (from Experiment 5). Five out of the top 20 features are extracted from *FunctionalLocation*, indicating that they are discriminating features that for instance would be good to use far up in decision trees.

The improvement of the other three algorithms were not as substantial, probably because some of the knowledge extracted from *FunctionalLocation* already had been learned from combinations of other features. Naïve Bayes fails to find knowledge from combined features because of the independence assumption.

When only features extracted from structured fields were added to the training set, Random Forest had the best performance measured by Cohen's Kappa. Experiment 1 was one of the two experiments where Random Forest performed better than the Stacked Ensemble. We were not able to interpret why this was the case, but one possible explanation is that the dataset were more suited for bagging techniques (Random Forest) in this experiment. In the other experiments, far more features were extracted from unstructured fields, which might have improved the performance of boosting and stacking.

8.1.2 Features Extracted From Unstructured Fields

Two methods were considered for extracting features from the unstructured *Text* field. The first method uses the TF-IDF weighting scheme in order to extract the most important terms from the document corpus, while the second method utilizes the manually constructed domain term dictionary.

Incorporating TF-IDF features in Experiment 2 improved the performance of all classifiers by an average of 0.025 measured in Cohen's Kappa, an average relative improvement of 9.1%. The experiment showed that Stacked Ensemble was slightly better when considering Cohen's Kappa and accuracy, but Random Forest and Gradient Boosting Machine performed better in terms of macro average F_1-Score. Since macro average F_1-Score weights smaller classes higher, the results indicate that Stacked Ensemble is better at predicting the larger classes, while Random Forest and Gradient Boosting Machine are capable of classifying the smaller classes to a greater extent. As the class distributions are highly imbalanced, it is preferable that a classifier is able to identify the smaller classes, even at the cost of accuracy.

In Experiment 3 where predefined domain terms were used, the predictive performance of the classifiers increased by an average of 0.03 measured in Cohen's Kappa, an average relative improvement of 13.2%. The results are similar to that of the TF-IDF method when it comes to the performance of the algorithms, but the overall performance increased slightly more when using the domain terms. An exception is Gradient Boosting Machine, which for an unknown reason scored better on all metrics when the TF-IDF terms were used. Random Forest had an almost equal Cohen's Kappa as Stacked Ensemble on unstructured terms, but Random Forest had a noticeable higher macro average F_1-Score. This indicates that Random Forest is better at classifying minority classes, while maintaining an overall performance equal to Stacked Ensemble when using features extracted from unstructured fields.

A comparison of the performance between the two methods is shown in Figure 8.1. Although the domain term method performed better than the TF-IDF method, they both gave a noteworthy performance boost when compared to the baseline experiment. One hypothesis is that the domain term method performed slightly better due to the fact that it guarantees that every FMC is represented by at least one term. This causes all FMCs to be considered in the classification processes. The TF-IDF method does not guarantee this, as the terms are simply selected on basis of their score.

One could argue that to give a fair assessment of the domain terms, features extracted using TF-IDF should be used as a baseline. By this definition, the average gain from the method is reduced to 0.005 measured by Cohen's Kappa.

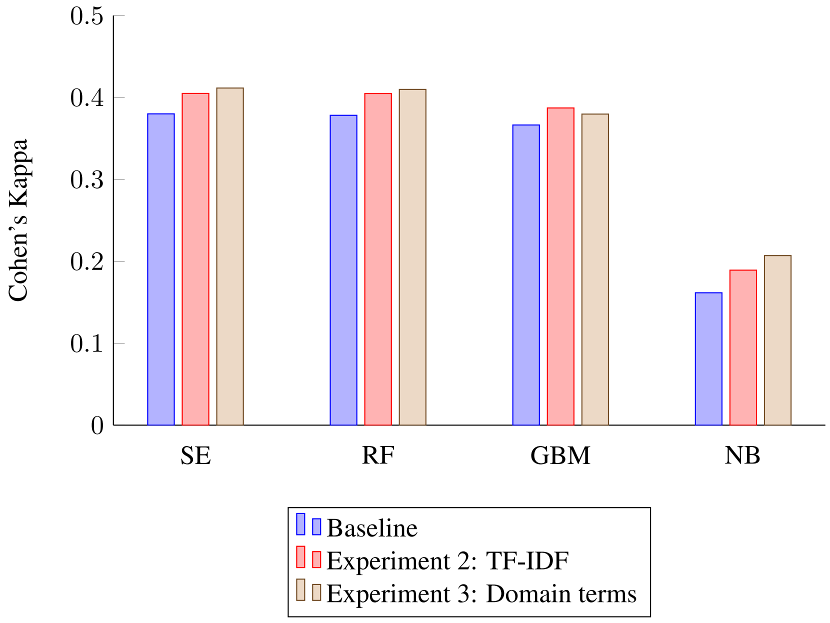


Figure 8.1: Comparison of performance in Cohen's Kappa between the two methods used for extracting features from unstructured fields (TF-IDF vs. domain terms).

8.1.3 Comparing the Effect of Structured and Unstructured Fields

Features extracted from structured and unstructured fields gave a notable increase in the predictive performance of the classifiers. As domain terms gave the best results for unstructured features, these are the results we refer to when talking about unstructured features in this section. A comparison between the two approaches in terms of performance measured in Cohen's Kappa is shown in Figure 8.2. Baseline results are also included for reference. The algorithms improved with an average of 0.023 (relative improvement 10.9%) measured in Cohen's Kappa by using the features extracted from structured fields, versus an average increase of 0.03 (relative improvement 13.2%) from the unstructured fields. In other words, the features from unstructured fields contributed slightly more to the overall performance gain.

Although the features extracted from unstructured fields affected the performance the most, there are 217 of these features, compared to the 14 obtained from *FunctionalLocation*. The performance gain per *FunctionalLocation* feature is therefore substantially bigger than that of the features extracted using domain terms. This is most likely due to the fact that these are nominal features, whereas the domain term features simply consists

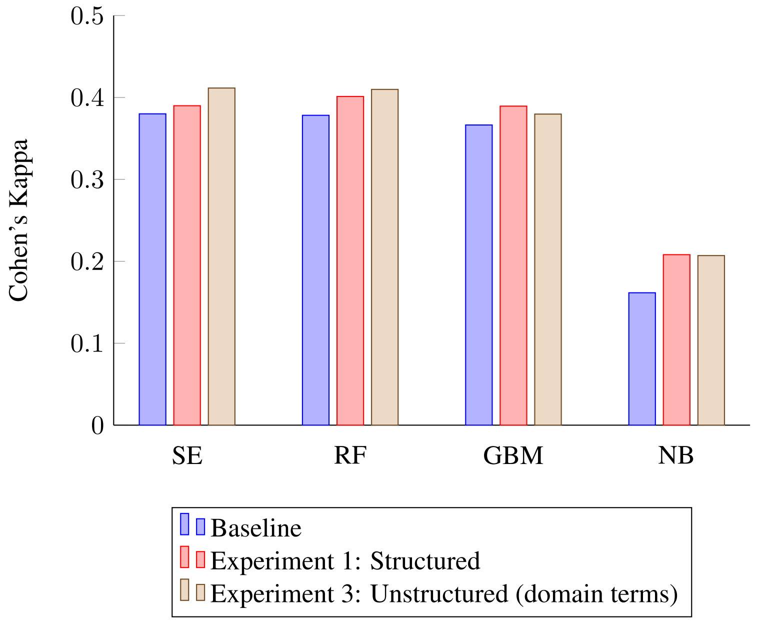


Figure 8.2: Comparison of performance in Cohen's Kappa between the two approaches of utilizing domain knowledge.

of binary values. The significance of the features extracted from *FunctionalLocation* is further emphasized by the feature importance ranking shown in Figure 7.2. It shows that five of the 20 most important features were extracted from *FunctionalLocation*. This implies that these features are very discriminating. In contrast, single domain term features are not important by themselves, however, when combined, they were able to boost the predictive performance of all the classifiers.

8.1.4 Combining the Approaches

The best results in this research were achieved when utilizing features extracted from both structured and unstructured fields. Figure 8.3 shows the results of adding different combinations of extracted features to the training data. While the results from using TF-IDF and domain terms alone are quite similar, the difference was much bigger when features extracted from *FunctionalLocation* were included in the training. Experiment 4 and 5 show that domain terms perform significantly better than TF-IDF in combination with features from *FunctionalLocation*. This suggests that the classification value inferred from TF-IDF does not complement the value inferred from structured features to the same extent as the domain terms does. The performance decreased slightly when all the extracted features were used in combination in Experiment 7. A possible explanation is that the classifiers have overfitted due to the large amount of features used. This hypothesis is supported by the fact that the performance of Gradient Boosting Machine decreased the most. Gradient Boosting Machine is a boosting method and thus more prone to overfitting. Naïve Bayes is not affected to the same degree as Gradient Boosting Machine, and it is more robust to noise and overfitting [48]. Random Forest is also only experiencing a minor decrease in Cohen's Kappa. Since Random Forest is a bagging technique, it is better at reducing variance, thus making it more robust to overfitting. The Stacked Ensemble is indirectly prone to overfitting since it uses a Gradient Boosting Machine as a base learner, but not to the same extent. This is because the variance is reduced by using multiple different base learners. These are all characteristics that matches the results in Figure 8.3.

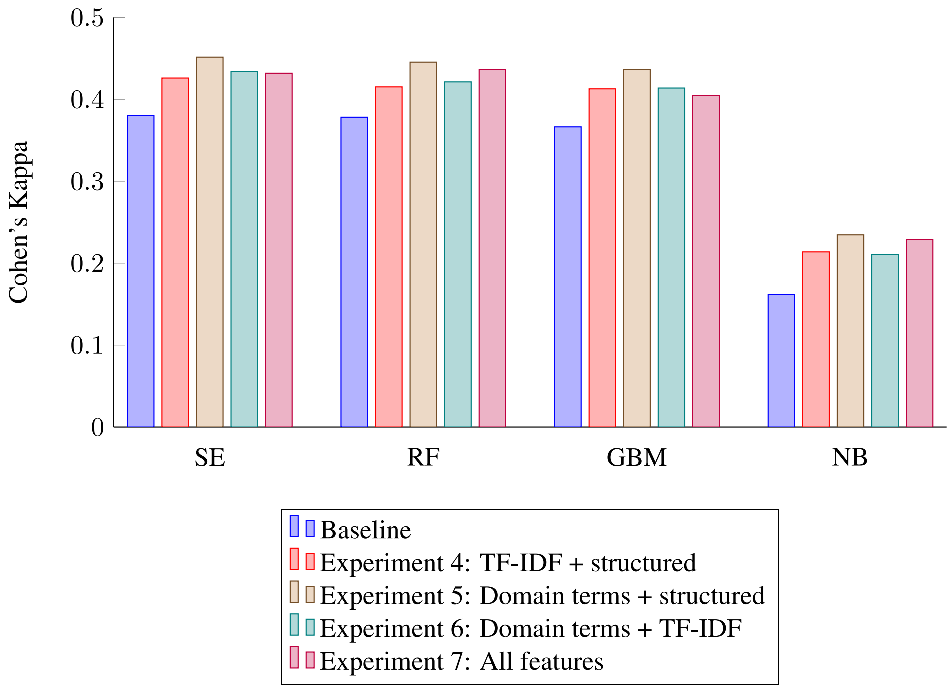


Figure 8.3: A comparison of results when the extracted structured features were added in addition to TF-IDF and domain terms.

Experiment 6 utilize features extracted from unstructured fields using both TF-IDF and domain terms in combination. The performance of the classifiers in this experiment are significantly higher than when the feature sets are used alone. This suggests that the classification value inferred from the different methods complement each other. Together with the implications that the classifiers overfitted in Experiment 7, this indicates that conducting more experiments where feature selection techniques are applied to all the extracted features could have increased the performance of the classifiers.

8.2 Choice of Evaluation Metrics

There are almost as many evaluation metrics as there are classification algorithms. Some metrics, such as area under the ROC curve only works for binary classifiers, while others work for multiclass problems as well. Accuracy is traditionally a commonly used evaluation metric, due to its simplicity and fairness. It does however give little information about how the performance is distributed among classes in classification tasks using imbalanced datasets. The accuracy paradox sheds light on this shortcoming.

A confusion matrix presents a detailed overview of how a classifier performs on all classes, and can be useful for inspecting the predictions of a single classifier. It does not yield a single value metric, which makes it unsuitable for comparison of multiple classifiers.

The experiments presented in the previous chapter lists Cohen's Kappa, accuracy and macro average F_1-Score. Cohen's Kappa and macro average F_1-Score were chosen based on earlier research and recommendations. An attractive characteristic of Cohen's Kappa is that it always removes random chance from evaluation, regardless of class distribution. Majority classifiers and classifiers that predicts randomly based on class distribution, will get a Cohen's Kappa of ≈ 0.

Accuracy was also included in the results due to its widespread use. It should not be used as a metric singlehandedly in imbalanced class problems, since majority classifiers and random classifiers tend to get a high score even though the classifiers do not solve the classification task at hand. This was demonstrated with the accuracy paradox in Section 2.9.2.

The use of multiple evaluation metrics proved to give some insights that would have been missed using only one metric. While accuracy weights each sample equally, Cohen's Kappa weights classes relative to their size. Consequently, samples from smaller classes are weighted higher than samples from larger classes. This makes Cohen's Kappa suitable as a primary metric when you are dealing with an imbalanced dataset where smaller classes are important, but the main task is not to correctly predict samples in small classes. Macro average F_1-Score is more suited when correctly predicting small classes is crucial, since it makes each class contribute equally to the final score. This means that a class with one sample is equally important to classify correctly as all 1 000 samples in another class. These characteristics can implicate how a classifier performs when the metrics are examined in combination. Looking at the results from Experiments 1-5, we see that Gradient Boosting Machine always had a higher improvement on the macro average F_1-Score than on accuracy and Cohen's Kappa when domain knowledge was added. This implies that the domain knowledge made Gradient Boosting Machine perform better on smaller classes, since the macro average F_1-Score favours small classes. On the contrary, Naïve Bayes saw a greater improvement in Cohen's Kappa and accuracy, meaning that the classifiers got better on larger classes. The insight that the combination of these evaluation metrics provide, can give an understanding of how the classifiers evolve when features are added or removed.

8.3 Assessment of Algorithms

There were very small differences in the performance of the top three classification algorithms. A possible explanation to this is that both Gradient Boosting Machine and Random Forest are ensembles of tree learners. Stacked Ensemble is a second level meta learner that was trained using Gradient Boosting Machine and Random Forest as base learners, meaning that all knowledge eventually is extracted using decision trees. Another possible explanation is that the classifiers are closing in on the best possible performance that can be achieved with the dataset at hand. The naïve Bayes algorithm performed substantially worse, which may be caused by the fact that it ignores dependencies between features. It is more dependent on feature selection compared to the other algorithms, and a limited amount of time went into optimizing the feature set for naïve Bayes. Nevertheless, naïve Bayes had the highest relative improvement from the basesline. This indicates that the other algorithms already had learned some of the extracted knowledge from combinations of other features.

Stacking generally led to an increase in Cohen's Kappa and slight decrease in macro average F_1-Score when compared to the base learners. For instance, the best Random Forest and Gradient Boosting Machine classifiers had a Cohen's Kappa of 0.4454 and 0.4363, and a macro average F_1-Score of 0.3665 and 0.3674, respectively. When these classifiers were combined in the Stacked Ensemble, the Cohen's Kappa increased to 0.4515 while the macro average F_1-Score decreased to 0.3641. This pattern was found in Experiments 2-5. In Experiment 1 where only features extracted from structured fields were incorporated, Random Forest performed slightly better than the Stacked Ensemble, but the macro average F_1-Score still decreased in the Stacked Ensemble. In spite of this, the overall impression is that stacking results in a more accurate prediction of larger classes at the expense of smaller classes. As Stacked Ensemble was the classifier that produced the highest Cohen's Kappa in the study, it is considered to be the most accurate predictor for the task at hand.

As discussed in Section 8.2, Gradient Boosting Machine generally had a higher improvement on the macro average F_1-Score than on accuracy and Cohen's Kappa when more features were added to the dataset. The results in Experiment 6 and 7 show that the performance of Gradient Boosting Machine suddenly drops below the performance of Random Forest and Stacked Ensemble measured by all of the evaluation metrics. Since Gradient Boosting Machine is a boosting technique, it is more prone to overfitting. This suggests that the Gradient Boosting Machine overfits when features extracted using both TF-IDF and domain terms are used in combination. The Stacked Ensemble is also affected by this, since Gradient Boosting Machine is used as a base learner. Based on these results, it is evident that Random Forest is better suited than Gradient Boosting Machine and Stacked Ensemble if high variance and overfitting is a problem in the classification task.

The Gradient Boosting Machine from Experiment 5 was the classifier with the highest macro average F_1-Score. This speaks for using Gradient Boosting Machine instead of Random Forest and Stacked Ensemble when the small classes are substantially more important than larger classes. This is supported by the fact that Gradient Boosting Machine is a boosting algorithm, where misclassified samples (often belonging to minority classes) are gradually weighted higher.

8.4 Comparison to Related Work

Many of the algorithms and evaluation metrics that were used in the related work, were also used in our study. It did however prove to be challenging to compare the results, due to the nature of the datasets and the classification tasks. Industrial classification tasks are often atypical, and usually requires specific approaches. High dimensional industrial datasets have unconventional characteristics that peculiarly influence how different techniques perform at classification tasks. This makes comparison with related work challenging.

The related studies that involved clustering and classification of maintenance logs and detection of oil spills were both binary classification tasks. In the study involving detection of oil spills, the dataset consisted of images, which made it infeasible to use the same techniques for applying domain knowledge. The research on maintenance logs did not include a baseline, and it is therefore hard to evaluate how much their preprocessing techniques contributed to increasing the performance compared to the techniques that we applied.

We tested a few of the techniques that Wilcox and Hripcsak [67] used in the study of automating medical text report classification. Both naïve Bayes and decision tree based algorithms were tested, and domain knowledge were utilized in a similar way. Wilcox and Hripcsak applied text mining before domain experts selected relevant terms from the output of the mining. When we tested the same approach, a few of the minority classes were not represented in the the set of output terms. Consequently, the selection of domain terms in our research was done without utilizing text mining. Compared to the baseline, our best results had an improvement of 0.07 measured by Cohen's Kappa, with a relative improvement of 18.8%. Wilcox and Hripcsak were able to achieve an improvement of 0.11, and a relative improvement of 13.6%, measured by area under the ROC curve. The classification task in Wilcox and Hripsack's research had six target classes, while our had 29. The large difference in the number of target classes is one possible explanation for their better overall performance. Our classifiers still had a better relative improvement compared to the baseline. This might be related to the fact that we extracted features from both structured and unstructured fields, while they only focused on text mining unstructured fields. Due to the large dimensionality differences of the datasets, it is hard to draw any conclusions. It is difficult to do any further comparisons of the results, as different evaluation metrics were used in the studies.

Our research was conducted in a similar manner as Wilcox and Hripcsak's, considering that we ran experiments using multiple classification algorithms and that varying degrees of domain knowledge were incorporated in the training. Strengths in our research were that we also applied domain knowledge on structured fields in addition to unstructured fields, which enabled the extraction of more features. We also included multiple evaluation metrics, which gave a more comprehensive understanding of the results.

A major part of the research conducted by Kubat et al. [40], focused on selecting a fitting evaluation metric when dealing with an imbalance dataset. Their objective was to correctly identify minority class samples in a binary classification task. This differs from our problem, since we rather wanted minority class samples to be slightly higher weighted in the evaluation.

Kubat et al. ultimately decided to use *g-mean* as their primary evaluation metric. In a binary classification task, g-mean is given by $\sqrt{acc+ \cdot acc-}$ where $acc+$ is the accuracy of positive samples and $acc-$ is the accuracy of negative samples. In a multiclass problem, the g-mean can be calculated by multiplying the accuracy of each class before finding the n-th root of the product where n is the number of classes. That implies that if the accuracy of one class is 0, the overall g-mean will also be 0. A class with an accuracy close to 0 will also have a huge impact on the final score. This is unfortunate when dealing with 29 classes, where some of the classes are very small. As we have argued for earlier in this chapter, Cohen's Kappa is more suited when dealing with a multiclass problem with an imbalanced dataset. Misclassified samples belonging to minority classes will have a bigger impact on the final score than samples from large classes, but a few errors in the smallest classes will not bring the final score to the ground as g-mean does.

Table 8.1 shows an example of a confusion matrix for a classification task with five target classes. The class distribution in the example resembles the distribution in our dataset, but with fewer classes. The confusion matrix shows that the classifier has done a poor job in classifying samples from minority classes.

Table 8.1: Example of confusion matrix for a classifier using an imbalanced dataset with five target classes.

		Predicted class				
		A	B	C	D	E
	A	**1000**	50	30	10	5
	B	70	**100**	20	5	2
Actual class	C	20	10	**20**	4	0
	D	6	3	1	**4**	0
	E	2	1	0	0	**0**

Figure 8.4 shows different single value metrics computed from the confusion matrix. It gives an overview of how much each evaluation metric penalize misclassifying minority class samples. Although the differences were not that big in our actual results, the trends look the same as in the figure. When you are dealing with many classes and an imbalanced dataset, g-mean is not suited due to the high degree of penalization of misclassifying minority samples. Unless the goal is to correctly classify samples from the smaller classes, misclassifying minority samples will often also affect the macro average F_1-Score too much since each class is weighted equally in the score. Accuracy does not care about

class distribution at all, and the larger classes will often have a higher impact on the score than desired. Cohen's Kappa weights each class relative to their size, which satisfies the requirements we have for evaluating the models in our classification task.

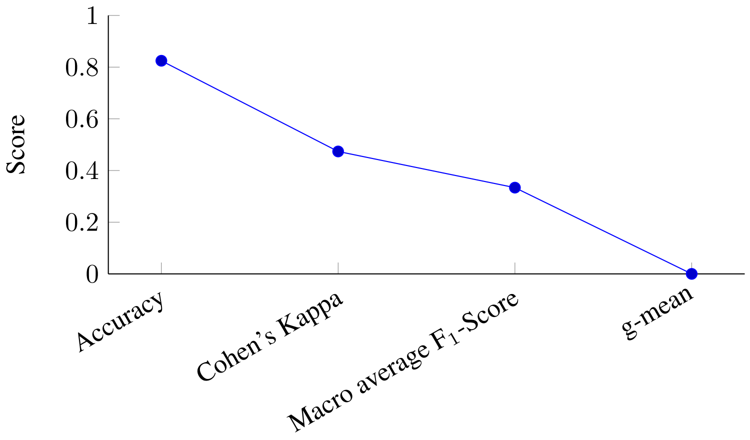


Figure 8.4: Evaluation metrics based on the confusion matrix presented in Table 8.1.

Edwards et al. [17] were the most similar research we were able to find, since it concerns classifying maintenance logs in an industrial setting. The study does however not include any baseline metrics, it is a binary classification task, and it does not utilize domain knowledge in the classification task. This makes it hard to compare results. The study encountered many of the challenges as we did, and preprocessing was a major part of the research. It only focused on extracting features from unstructured fields, although the dataset included three structured fields that possibly could have enhanced the performance of the classifier. TF-IDF was used for extracting features from the text. Edwards et al. used SVD in order to reduce the dimensionality, and generalize the TF-IDF vectors. We created binary feature vectors based on the top 217 TF-IDF words, and it could have been interesting to experiment more with SVD, alternative weighting of extracted features, and more advanced NLP techniques. Opposed to Edwards et al., our main effort was put into understanding the entire dataset and extracting info from both structured and unstructured fields, instead of applying advanced NLP techniques. We do believe that Edwards et al. could have benefited from extracting and utilizing more features. The output of their *maintenance type* clustering could have been included when training a classifier for predicting if a log entry was scheduled or not. Structured fields that indicated the cause of the entry and damage were merely joined with the rest of the text, instead of trying to normalize and utilize the structured fields such as we did.

Chapter 9

Conclusion

Finding good features for supervised learning in high dimensional industrial settings is challenging. Using feature extraction techniques to generate new features had a significant impact on classifying work orders. It is crucial to obtain a thorough understanding of the dataset at hand, in order to perceive where features can be encoded.

According to Cognite, the best classifier is currently not accurate enough to automatically label work orders with an FMC, but it is accurate enough to suggest FMCs when an operator submits new work orders. A classifier used to automatically label work orders with an FMC, would need to have an accuracy of at least 80% in order to provide value for Aker BP. Reaching a satisfactory performance level in this task proved to be infeasible with the techniques that we applied. While the best classifier had a relative improvement of over 18% compared to our baseline results, a Cohen's Kappa of 0.45 is not accurate enough to be used for automatic labeling. It is however good enough to be used for suggesting FMCs when operators on oil platforms submit new works orders. Since human interaction is required to actually select the correct FMC from a set of suggestions, it is not necessary to achieve the same predictive performance as an automatic classifier. Statistically, our best classifier will suggest the correct FMC in almost one out of two cases.

This research has given Cognite a better understanding of the dataset, such as the relationships between *FailureModeCode*, *CodeGroup* and *CatalogProfile*, as well as the meaning of several features. We have also shed light on some techniques that can be employed to improve the classification results, which can be used in combination with other techniques in further experiments.

9.1 Research Questions

This section contains final conclusions for the research questions. The conclusions are based on the results, as well as the discussions from the previous chapter.

RQ1 To what extent do new domain-driven features extracted from existing fields of industrial datasets contribute to a classifier's performance?

The conducted experiments showed that using domain knowledge to extract new features had a significant impact on the predictive performance of the tested classifiers. Using features extracted from unstructured fields in combination with features extracted from structured fields, resulted in an increase in performance by up to 0.073 measured in Cohen's Kappa. The average improvement of the best models were 25.2%. Performance gain from the extracted domain-driven features outweighed differences in the top three classification algorithms used. Out of the top 20 features ranked by importance, five were extracted using domain knowledge, including the top two features.

Understanding how and where domain knowledge could be utilized and applied was a time consuming task, however, we were unable to achieve similar results using other techniques. While the exact procedure used to extract new features in this work is very specific, the overall approach is quite generic. Canvassing the dataset for encoded values and extracting features from these fields, can be applied in similar work as well as in other domains.

RQ2 How does features extracted from unstructured and structured fields compare in terms of improving a classifier's performance when classifying industrial data?

Features extracted from structured and unstructured fields both contributed to a significant increase in the predictive performance of all the tested classifiers. Experiments 1-3 test feature sets extracted from structured and unstructured fields on their own. It is apparent that the average performance of all the tested algorithms increased more from the 217 features extracted from unstructured fields, than from the 14 features extracted from structured fields (0.0277 versus 0.0228 measured in Cohen's Kappa). However, the performance gain per feature was greater from the features obtained from *FunctionalLocation*, indicating that these are more independent and discriminating. As we are dealing with a multiclass classification problem, this result is as expected. The domain term features are simply binary features, whereas the *FunctionalLocation* features are polytomous features.

When structured features were utilized in combination with unstructured features, the performance was always better than when they were applied on their own. The highest performance boost was achieved in Experiment 5. This experiment utilized features extracted using domain terms from the unstructured *Text* field, in combination with features extracted from the structured field *FunctionalLocation*. The average performance of the classifiers increased by 0.0704 measured in Cohen's Kappa in this experiment, an average relative improvement of 25.2% from the baseline. It is therefore evident that the approaches work substantially better in combination than as standalone feature extraction methods. As the features extracted from structured and unstructured fields fulfill each other, it is difficult to conclude which of them contributes the most in improving the classifiers performance.

RQ3 What are informative evaluation metrics when dealing with imbalanced multiclass datasets?

As a single value metric, Cohen's Kappa proved to be the most fitting evaluation metric for the classification task at hand. That is because Cohen's Kappa weights minority classes higher than larger classes, but not to the extent where each class contributes equally to the score. Characteristics of similar problems that could benefit from using Cohen's Kappa, are problems where the dataset is highly imbalanced, contains multiple target classes, and where correctly classifying samples in small classes should be weighted higher than classifying samples in large classes. If the most important task is to correctly classify the minority class samples, macro average F_1-Score should be used instead. This finds the harmonic mean between precision and recall for each class, and weights the classes equally in the final score. In binary imbalanced classification tasks, $g - mean$ is an informative metric. It is not suited for multiclass problems, since a low accuracy in one of the classes have a detrimental effect on the final score.

Including both Cohen's Kappa and macro average F_1-Score in our results provided additional value, since it gave insight into how the classifiers evolved when new features were added. A higher increase in Cohen's Kappa than macro average F_1-Score implies that the classifiers got better on larger classes, and vice versa.

RQ4 How does a Stacked Ensemble compare with its constituent models when classifying highly imbalanced datasets?

There were very small differences in the performance of the Stacked Ensemble and its two constituent algorithms, Random Forest and Gradient Boosting Machine. In all experiments except one, stacking led to an increase in Cohen's Kappa and a slight decrease in macro average F_1-Score when compared to the base learners. This indicates that stacking results in a more accurate prediction of larger classes at the expense of smaller classes.

9.2 Further Work

It would be interesting to further investigate techniques to extract features from textual fields. Instead of just looking at each word separately, building n-grams of words could impact the results. The domain term dictionary can also be extended to include more terms. Features could also be extracted using *Named Entity Recognition*, *word2vec* or *doc2vec*. Extracted terms could be weighted using for instance TF-IDF weights, and a more sophisticated technique could be used when selecting which terms to include in the feature set.

It would also be interesting to test different numbers of terms to include in the feature set, to see how the size of the feature set affects the performance. Acquiring more data from other oil platforms for training and validation could improve the performance of the models as well. It would also be interesting to see how the system performs when highly correlated FMCs are grouped together. The measures mentioned above could further improve the predictive performance of the classifiers.

Since the *FunctionalLocation* field encapsulates encoded data using two different encoding schemes, these features only contribute to finding correlations internally in the two coding scheme groups. Finding a full or partial mapping between the two coding schemes could provide great value when training the classifiers. This would help the classifiers to find correlations between samples from the different coding schemes. More domain knowledge is required to create this mapping.

Late in the research, Cognite conveyed that it would be useful to know how the classifiers performed, if a sample were considered correctly classified if the sample's actual class were among the top five output classes ordered by probability. Naïve Bayes is a probabilistic classifier, and calculates the probabilities that a record belongs to each of the target classes. As ensemble learners use voting to decide the predicted class, it is possible to get probabilities for a sample belonging to each class. There was however no easy way to add this functionality to our system, and we could not find time to implement it.

Bibliography

[1] Mohamed Aly. Survey on Multiclass Classification Methods. Technical report, California Institute of Technology, 01 2005.

[2] Mouna El Attar and Xavier Hamery. Industrial Expert System Aquired by Machine Learning. *Applied Artificial Intelligence*, 8(4):497–542, 1994.

[3] Eric Bauer and Ron Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine learning*, 36(1-2):105–139, 1999.

[4] Arie Ben-David. A Lot of Randomness is Hiding in Accuracy. *Engineering Applications of Artificial Intelligence*, 20(7):875 – 885, 2007.

[5] Arie Ben-David. Comparison of Classification Accuracy Using Cohen's Weighted Kappa. *Expert Systems with Applications*, 34(2):825 – 832, 2008.

[6] Enrico Blanzieri and Anton Bryl. A Survey of Learning-Based Techniques of Email Spam Filtering. *Artificial Intelligence Review*, 29(1):63–92, 2008.

[7] Avrim Blum, Adam Kalai, and John Langford. Beating the Hold-out: Bounds for K-fold and Progressive Cross-validation. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, COLT '99, pages 203–208, New York, NY, USA, 1999. ACM.

[8] Leo Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[9] Leo Breiman. Stacked Regressions. *Machine Learning*, 24(1):49–64, Jul 1996.

[10] Leo Breiman. Random Forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[11] Leo Breiman, Jerome. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

[12] Devendra Singh Chaplot et al. Predicting Student Attrition in MOOCs using Sentiment Analysis and Neural Networks. In *AIED 2015 Workshop Proceedings*, volume 3. AIED, 2015.

[13] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[14] Jacob Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[15] Jesse Davis and Mark Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM.

[16] Thomas G. Dietterichl. Ensemble Learning. In M. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 405–408. MIT Press, 2002.

[17] Brett Edwards, Michael Zatorsky, and Richi Nayak. Clustering and Classification of Maintenance Logs Using Text Data Mining. In *Proceedings of the 7th Australasian Data Mining Conference - Volume 87*, AusDM '08, pages 193–199, Darlinghurst, Australia, Australia, 2008. Australian Computer Society, Inc.

[18] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Advances in Knowledge Discovery and Data Mining. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter From Data Mining to Knowledge Discovery: An Overview, pages 1–34. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[19] Ronen Feldman and James Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, December 2006.

[20] Scott Fortmann-Roe. *Understanding the Bias-Variance Tradeoff*, 2012.

[21] Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.

[22] Jerome H. Friedman. Another Approach to Polychotomous Classification. Technical report, Department of Statistics, Stanford University, 1996.

[23] Jerome H. Friedman. On Bias, Variance, 0/1-Loss, and the Curse-of-Dimensionality. *Data Min. Knowl. Discov.*, 1(1):55–77, January 1997.

[24] Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Statist.*, 29(5):1189–1232, 10 2001.

[25] S. García et al. A Study of Statistical Techniques and Performance Measures for Genetics-Based Machine Learning: Accuracy and Interpretability. *Soft Computing*, 13(10):959, Dec 2008.

[26] Stuart Geman, Elie Bienenstock, and René Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1):1–58, 1992.

[27] Isabelle Guyon and André Elisseeff. An Introduction to Variable and Feature Selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[28] H2O. Welcome to H2O 3. `http://docs.h2o.ai/h2o/latest-stable/h2o-docs/welcome.html`. [Accessed 23 March 2018].

[29] David J. Hand and Kerning Yu. Idiot's Bayes – Not So Stupid After All? *International Statistical Review*, 69(3):385–398, 2001.

[30] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2 edition, 2008.

[31] H. He and E. A. Garcia. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, Sept 2009.

[32] Chih-Wei Hsu and Chih-Jen Lin. A Comparison of Methods for Multiclass Support Vector Machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, Mar 2002.

[33] László A. Jeni et al. Facing Imbalanced Data–Recommendations for the Use of Performance Metrics. In *Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, ACII '13, pages 245–251, Washington, DC, USA, 2013. IEEE Computer Society.

[34] George H. John and Pat Langley. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, pages 338–345, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[35] G. Kesavaraj and S. Sukumaran. A Study on Classification Techniques in Data Mining. In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pages 1–7, July 2013.

[36] Ron Kohavi. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[37] Igor Kononenko. Inductive and Bayesian Learning in Medical Diagnosis. *Applied Artificial Intelligence an International Journal*, 7(4):317–337, 1993.

[38] Sotiris Kotsiantis, Dimitris Kanellopoulos, and Panayiotis Pintelas. Handling Imbalanced Datasets: A Review.

[39] Sotiris Kotsiantis, Dimitris Kanellopoulos, and Panayiotis Pintelas. Data Preprocessing for Supervised Learning. *International Journal of Computer Science*, 1(2):111–117, 01 2006.

[40] Miroslav Kubat et al. Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*, 30(2):195–215, Feb 1998.

[41] J. Richard Landis and Gary G. Koch. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174, 1977.

[42] Edda Leopold and Jörg Kindermann. Text Categorization with Support Vector Machines. How to Represent Texts in Input Space? *Machine Learning*, 46(1):423–444, Jan 2002.

[43] David D. Lewis. Naive (Bayes) at forty: The Independence Assumption in Information Retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, pages 4–15, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[44] Shuangfang LU, Wenbiao HUANG, Fangwen CHEN, Jijun LI, Min WANG, Haitao XUE, Weiming WANG, and Xiyuan CAI. Classification and Evaluation Criteria of Shale Oil and Gas Resources: Discussion and Application. *Petroleum Exploration and Development*, 39(2):268 – 276, 2012.

[45] Andrew McCallum, Kamal Nigam, et al. A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[46] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam Filtering With Naive Bayes – Which Naive Bayes? In *CEAS*, volume 17, pages 28–69, 2006.

[47] László Monostori. AI and Machine Learning Techniques for Managing Complexity, Changes and Uncertainties in Manufacturing. *Engineering Applications of Artificial Intelligence*, 16(4):277 – 291, 2003. Intelligent Manufacturing.

[48] Vivek Narayanan, Ishan Arora, and Arjun Bhatia. Fast and Accurate Sentiment Classification Using an Enhanced Naive Bayes Model. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 194–201. Springer, 2013.

[49] Offshore Technology. Alvheim Field, Norway. `https://www.offshore-technology.com/projects/alvheim/`. [Accessed 14 May 2018].

[50] D T Pham and A A Afif. Machine-learning Techniques and Their Applications in Manufacturing. In *IMechE, Part B: The Journal of Engineering Manufacture*, pages 395–412, Manufacturing Engineering Centre, Cardiff University, Cardiff, UK, 2005. SAGE.

[51] John Ross Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, Mar 1986.

[52] John Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1 edition, January 1993.

[53] John Ross Quinlan et al. Bagging, Boosting, and C4.5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996.

[54] Juan Pramyr Ramos. Using TF-IDF to Determine Word Relevance in Document Queries. Technical report, Department of Computer Science, Rutgers University, 2003.

[55] Ryan Rifkin and Aldebaro Klautau. In Defense of One-Vs-All Classification. *J. Mach. Learn. Res.*, 5:101–141, December 2004.

[56] L. Rokach and O. Maimon. Top-down Induction of Decision Trees Classifiers - a Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487, Nov 2005.

[57] S. R. Safavian and D. Landgrebe. A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674, May 1991.

[58] Cullen Schaffer. Overfitting Avoidance as Bias. *Machine Learning*, 10(2):153–178, Feb 1993.

[59] Mark Spelman, Bruce Weinelt, Pedro Gomez, Renée van Heusden, Reema Siyam, Muqsit Ashraf, Vivek Chidambaram, James Collins, Adnan Khan, Wolfgang Popp, Anand Shah, Pratik Agrawal, and Shishir Shroff. Digital Transformation Initiative Oil and Gas Industry. Technical report, World Economic Forum, 01 2017.

[60] Standard Norge. NORSOK STANDARD Z-DP-002. `http://www.standard.no/pagefiles/942/z-dp-002r1.pdf`, 1995. [Accessed 15 March 2018].

[61] David Stirling and Wray Buntine. Process Routings in a Steel Mill: a Challenging Induction Problem. *Artificial Intelligence Developments and Applications*, pages 301–313, 1988.

[62] David RB Stockwell and A Townsend Peterson. Effects of Sample Size on Accuracy of Species Distribution Models. *Ecological modelling*, 148(1):1–13, 2002.

[63] Yuk Lai Suen, Prem Melville, and Raymond J Mooney. Combining Bias and Variance Reduction Techniques for Regression Trees. In *European Conference on Machine Learning*, pages 741–749. Springer, 2005.

[64] Mark J. van der Laan et al. Super Learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007.

[65] Geoffrey I Webb, Janice R Boughton, and Zhihai Wang. Not so Naive Bayes: Aggregating One-Dependence Estimators. *Machine learning*, 58(1):5–24, 2005.

[66] Gary Weiss and Foster Provost. The Effect of Class Distribution on Classifier Learning: An Empirical Study. Technical report, 2001.

[67] Adam B. Wilcox and George Hripcsak. The Role of Domain Knowledge in Automating Medical Text Report Classification. *Journal of the American Medical Informatics Association*, 10(4):330–338, 2003.

[68] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[69] David H. Wolpert. Stacked Generalization. *Neural Networks*, 5:241–259, 1992.

[70] David H. Wolpert. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[71] David H. Wolpert. *The Supervised Learning No-Free-Lunch Theorems*, pages 25–42. Springer London, London, 2002.

[72] Xindong Wu, Vipin Kumar, John Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 Algorithms in Data Mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[73] Xantic. SFI Group System - Product Description. `https://www.xantic.net/internet/files/products/amos/sfi/supportdocuments/ProductDescription.pdf`, 2005. [Accessed 15 March 2018].

[74] Xingquan Zhu and Ian Davidson. *Knowledge Discovery and Data Mining: Challenges and Realities*. IGI Global, Hershey, PA, USA, 2007.

Appendix A

Research paper

A.1 Using Domain Knowledge in Classifying Industrial Data from the Oil and Gas Sector

On the following 16 pages are the research paper we submitted to The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2018 (ECML-PKDD). At the time of this writing, we have not gotten feedback on whether or not the paper is accepted to the conference.

Using Domain Knowledge in Classifying Industrial Data from the Oil and Gas Sector

Sondre Hjetland[1], Eirik Fosse[1], Que Tran[2], and Jon Atle Gulla[1]

[1] Norwegian University of Science and Technology - Department of Computer Science, Trondheim, Norway
`sondrehj@stud.ntnu.no,eirik@bustbyte.no,jon.atle.gulla@ntnu.no`
[2] Cognite AS, Oslo, Norway
`que.tran@cognite.com`

Abstract. Finding good features for performing supervised learning on high dimensional industrial datasets can be challenging, as your feature set typically consists of hundreds to thousands of features. Specific features might follow protocols or custom coding standards that, unless decoded, are unusable by machine learning algorithms. This is often the case in industrial environments, where you need domain knowledge to interpret the semantics of the data. This paper analyzes the effect of domain knowledge on the supervised learning process in creating classifiers for industrial work orders. Our experiments, using various supervised learning algorithms on a high dimensional real-world dataset, show that incorporating domain knowledge in the pre-processing phase, improves the performance of a classifier substantially. By utilizing domain knowledge we were able to increase the performance of the classifier with 7% measured by Cohen's Kappa. The two most important features in the resulting model were features extracted using domain knowledge (*System* and *Equipment type*). The sum of the importance of the extracted features were 38.97% in the final model. This implies that domain knowledge is crucial in order to avoid erroneous pruning of important encoded features.

Keywords: Industrial data, Supervised machine learning, Domain knowledge, Imbalanced data set

1 Introduction

A focus on digitizing the oil and gas sector has emerged in the past few years. World Economic Forum recently published a white paper, naming digitization the new era for the oil and gas industry [16]. The industry has not yet utilized the opportunities that derive from technology and the large amounts of data industrial companies possess. For instance, a single drilling rig at an oilfield can generate terabytes of data daily, but only a small part of the data is used for decision making. World Economic Forums' value-at-stake analysis estimates that a digital transformation in the oil and gas industry could unlock approximately $1.6 trillion of value for the industry, its customers and wider society.

This paper is written as part of a cooperation with Cognite AS, who are currently working on digitizing the oil and gas industry.

Industrial environments are data intensive settings, where data is generated from sensors, observations, reports and other instrumental readings. The dataset used in this research originates from one of Europe's largest independent oil and gas companies, hereafter referred to as *the oil and gas company.* It consists of work orders from an offshore oil and gas production platform operating on the Norwegian continental shelf. A work order is a manual entry of a malfunction or other types of events, and it contains both structured and unstructured data. Most work orders contain a *Failure Mode Code (FMC)*, specifying which type of malfunction that led to the work order. The objective of this study is to analyze how domain knowledge affects the performance in the task of classifying failure mode codes in work orders. It is beneficial for the oil and gas company to have failure mode codes on all records for the sake of further analysis and decision making. The classifier can also be used to suggest failure mode codes when operators submit new work orders.

With regards to the distribution of failure mode codes, the dataset is highly imbalanced and sparse, which presents a challenge in the classification task. In this research, we will experiment with multiple classification algorithms and test different types of domain knowledge.

There is limited work in the oil and gas industry that examines the impact of domain knowledge in classifying industrial data. There is however more research on this topic in other fields such as medicine and economy. The majority of the related research focus on rule-based learning and use domain knowledge to define rules based on features in unstructured textual data [18,8]. We experimented with similar approaches in our study, but the best results were achieved when encoded features were extracted from both structured and unstructured fields.

The main contribution of this work is to show the importance of understanding the dataset at hand, and how utilizing domain knowledge can be used to extract important encoded features in order to enhance the performance of a classifier.

Related work is presented in the next chapter. Chapter 3 then describes the dataset at hand. In Chapter 4 we go through the methods used in the research. Chapter 5 contains results and a discussion, before a conclusion is made in the final chapter.

2 Related Work

Wilcox and Hripcsak have analyzed the effect of domain knowledge on the inductive learning process in creating classifiers for clinical observations in medical text reports [18]. Structured data in medical reports were not sufficient for classifying hospital admissions. In the pre-processing phase of the inductive learning, natural language processing (NLP) was used to convert unstructured textual features into structured features. Domain experts then selected specific attributes or features that were relevant to the classification task. Domain knowledge was shown to be the most significant factor affecting inductive learning performance, outweighing differences in learning algorithms. The cost of acquiring domain knowledge was also a lot less than trying to learn said knowledge inductively. Wilcox and Hripcsak suggests that domain knowledge can be used to combine multiple features together or to create a new features, but they do not conduct any experiments on this matter.

Kubat et al. have researched the use of machine learning in the detection of oil spills in satellite radar images [13]. This is a binary classification problem, and the system the research is based on aims to identify if oil spills are present or not in regions of an image. The dataset used in the study shares many of the same characteristics as the dataset in our work, such as class imbalance, data sparsity and the fact that feature engineering is required. Domain experts defined 49 features describing each region in Kubat et al.'s study. A significant amount of time was spent defining these features, and other feature engineering techniques were not employed. The results of the study were similar to earlier research, but the main contribution of the study was to identify issues deserving attention of the research community. This include selecting descriptive performance metrics, handling imbalanced and sparse datasets, and performing feature engineering.

The majority of similar research we examined, utilizes domain knowledge by incorporating the knowledge in a rule-based learner. Most of the research also focused on unstructured textual features. As Wilcox and Hripcsak pointed out, domain knowledge could be used to extract new features from the existing feature set. That is what this study will focus on. Domain knowledge will be applied in the pre-processing and feature engineering phase, instead of directly in the classifier. We will extract encoded features from both structured and unstructured fields with the use of domain knowledge.

3 Dataset

In this study, a restricted real-world dataset provided by Cognite (on behalf of an oil and gas company) has been used. The dataset consists of 6364 maintenance and repair requests, also called work orders. Work orders are filed for any work that needs to be done on a platform – ranging from changing a light bulb to repairing a sub-sea pump. The dataset used in this study is a subset of all work orders, and consists of orders that are concerned with actual failures. Work orders are categorized into one of 40 classes or failure mode codes, manually assigned by an inspector or operator when the order is filed. The code indicates the type of failure that led to the work order. FMCs are exclusive; a work order may only be assigned a single code. A subset of the FMCs are listed in Table 1.

3.1 Features

A work order is described by 312 features, ranging from time stamps and identifiers automatically set by the system, to textual descriptions manually entered by an operator.

Table 1: Subset of Failure Mode Codes (10 of 40) with description and count. Listing the five largest and smallest classes, ordered by size.

Code	Description	Count	% of total
OTH	Other	1178	18.50%
ELU	External leakage utility medium	466	7.32%
SER	Minor in-service problems	414	6.51%
SPO	Spurious operation	413	6.49%
FTF	Fail to function on demand	383	6.02%
...	...	...	...
VLO	Very low output	4	0.06%
FRO	Fail to rotate	4	0.06%
FTR	Fail to regulate	2	0.03%
STP	Fail to stop on demand	1	0.02%
FDC	Fail to disconnect	1	0.02%

The feature space of the dataset consists of nominal and ordinal variables, with the most significant features being:

FailureModeCode (FMC). The classification target. A code indicating the type of equipment failure that led to the work order.

CodeGroup. A code indicating a grouping of FMCs. There are 50 unique CodeGroups in the dataset. Note that there is an *N:N* relationship between FMC and CodeGroup – a FMC may belong to multiple CodeGroups, and a CodeGroup may have multiple FMCs. For instance, amongst work orders with *ELU* as FMC, there exist 18 distinct CodeGroups. The majority of the CodeGroups are linked with 5-20 different FMCs. CodeGroup is not present in work orders lacking a FMC, meaning it must be predicted as well.

CatalogProfile. A code indicating a grouping of CodeGroups. The relationship between CatalogProfile and CodeGroup is *1:N*. However, the ratio is close to *1:1* as there are few CatalogProfiles with multiple CodeGroups.

FunctionalLocation. An identifier referring to the involved equipment (explained in Section 3.3).

Text. A manually entered free text value, often describing the problem, what has been done, what needs to be done and the equipment involved. The text has often been edited multiple times as work related to the work order is conducted.

3.2 Challenges

The work orders are manually entered by operators working on the platform. It poses several challenges for machine learning with regards to data quality.

Insufficient amount of data. The dataset initially consists of a mere 6364 rows. This is further reduced, as the classes *OTH (Other)* and *UNK (Unknown)* are removed due to the fact that these classes symbolize that the operator was unable to find a fitting FMC. That leaves us with a total of 4999 work orders and 38 distinct FMCs.

Numerous classes & imbalance. The total of 38 FMCs poses a challenge due to the limited amount of data. Each FMC has on average 4999/38 = 132 representative work orders. However, as seen in Table 1, the FMC distribution is skewed and the classes highly imbalanced, leaving us with relatively huge classes such as *ELU* and seemingly insignificant classes such as *FDC (Fail to Disconnect)*. There is also high correlation between certain FMCs, such as *NOO (No Output)* and *LOO (Low Output)*, making them hard to differentiate.

Missing and duplicate data. Attribute values are missing throughout the dataset. The missing data is in most cases due to unmeasured and missing information about the failure. Features with a coverage of less than 80% were pruned, which reduced the number of features to 141. Furthermore, 32 out of the 141 features are either duplicates of other features, or consist of uniform values. Pruning them leaves us with a total of 109 features.

Mislabeled data. As the FMCs are manually entered by an operator, errors and mislabeling occurs. Numerous classes, some of them hard to differentiate, may also cause the operator to be confused as to which FMC to use. This might result in a mislabeling, or simply taking the easy way out assigning it either *OTH* or *UNK*. Different operators may also have their own subjective interpretation of the FMCs, leading to an overlap of the classes.

Varying quality. The manually entered fields of the work orders, especially the failure description, are of varying quality. The texts vary from being a few words long, to being many sentences. Several languages such as Norwegian, English and Danish are used in the text fields, sometimes even in the same record. Some work orders contain well structured text and encapsulates important aspects of the order, but most of them are not. The written sentences are often grammarless and shortened, and contains industrial terms, abbreviations, identificators and jargon. All of these properties make it very challenging to perform natural language processing to extract features. An example of a typical text: *"Damper not closing 27.09.2010 23:00:06 JOHN DOE (JDOE) damper not closing during test, either from SAS or local 29/9 smørt opp tregt spjeld, testet x10, nå OK."*

3.3 Encoded Data

The feature set includes FunctionalLocation, that specifies an equipment tag. At first glance it looks like an arbitrary sequence of alphanumeric characters, but is in fact an encoded reference to the specific equipment that a work order concerns. Equipment tags at the platform have been issued using two well-established coding systems: SFI Group System [20] and NORSOK Z-DP-002 [17]. The SFI Group System is used for older hull equipment, while the rest of the tags are issued according to the NORSOK standard. Features encoded in the NORSOK equipment tags are:

Platform. Specifies on which platform the equipment resides.

System. Logical grouping category of the parent systems (e.g. *Oil Storage*).
Equipment type. Logical grouping category of the equipment types.
Equipment id. The first level in the hierarchy related to a physical asset.
Subunit. Unit within the equipment boundary defined by the equipment ID.
Maintainable item. Lowest level within the equipment boundary defined by the equipment ID.

Features encoded in the SFI equipment tags are:
Platform. Specifies on which platform the equipment resides.
SFI 3-digit group system. A ship/rig is divided into 10 primary groups (1st digit, e.g. 7 – Machinery main components). Each primary group consists of 10 secondary groups (2nd digit, e.g. 3 – Compressed air systems). Finally, each secondary group is divided into 10 tertiary groups (3rd digit, e.g. 1 – Starting air systems). Each digit represent a step down in the hierarchy.
SFI Detail- & material code. Not standardized, but used to define individual components in a system.
How these features are extracted will be presented in Section 4.2.

4 Method

Our method for training a FMC classifier is divided into three steps. First, we do basic data pre-processing such as class filtering, class grouping and some natural language processing. Secondly, the data is enriched by utilizing domain knowledge and data from external sources. Finally, models are built using Random Forest, Gradient Boosting Machine and Stacked Ensemble. Since our classification algorithms are able to handle the amount of features present in our dataset, and by design identifies the most important features, using other feature selection techniques is not relevant in this study.

4.1 Data Pre-processing

Pre-processing is one of the measures taken in order to deal with challenges of the dataset presented in the previous section. The FMCs *OTH (Other)* and *UNK (Unknown)* are filtered out due to being of little to no value. FMCs with less than 10 representative work orders are also removed. This decision is based on experiments using oversampling, undersampling and upsampling with synthetic samples. The sampling techniques had zero or negative impact on the final results, implying that there is insufficient data for these FMCs. This leaves us with the total of 29 distinct FMCs. Properties of the dataset during the filtering are listed in Table 2.

Highly correlated CodeGroups are merged together. This is done in order to increase the classification accuracy when predicting *CodeGroup* based on *CatalogProfile_Id*. The following groups are assembled:
Fire and Gas Detectors Composed of *FD (Fire Detectors)*, *GD (Gas Detectors)* and *FG (Fire and Gas Detectors)*.

Table 2: Number of records after steps of filtering.

Step	Action	# Records	# Classes
1	All records	6364	40
2	Removing OTH and UNK	4999	38
3	Removing classes with < 10 records	4960	29

Input Devices Composed of *ID (Input devices)* and *IP (Input devices)*.
Basic natural language processing operations are applied to the *Text* field of the work orders. Punctuation characters are removed, Norwegian special characters are encoded, and all text is made lowercase. These operations are done in order to prepare the data for tokenization and matching against a domain term dictionary as described in Section 4.2.

4.2 Utilizing Domain Knowledge

In the second step of training the classifier, domain knowledge is incorporated. The utilization of domain knowledge and enrichment of the data consist of three separate methods. In the first method applicable data is extracted from the FunctionalLocation. The second method uses external information about the equipment tags to construct new features. Domain knowledge used in the first two methods originates from descriptions and manuals received from the oil and gas company. In the third method, knowledge about each of the FMCs is utilized in order to extract useful information from the *Text*-field. Although the process of incorporating the knowledge in the pre-processing phase was straightforward, understanding how the domain knowledge could be used took some effort.

FunctionalLocation Breakdown FunctionalLocation consists of encoded values as previously described in Section 3.3. A series of string operations are used in order to decode and extract additional features from the values. After each step, the extracted part of the FunctionalLocation is removed from the string in order to simplify further extractions. Examples of FunctionalLocation breakdowns for both SFI and NORSOK are given in Table 3a and 3b. The approach of the feature extraction is a follows:

1. Extract platform info
2. Identify if FunctionalLocation is SFI or NORSOK
3. If NORSOK:
   (a) Extract component
   (b) Extract subunit
   (c) Extract system
   (d) Extract equipment type
   (e) Extract equipment id

4. If SFI:
   (a) Extract primary, secondary and tertiary group
   (b) Extract detail code
   (c) The meaning of the remaining part of FunctionalLocation is unknown

Table 3a: FunctionalLocation breakdown for NORSOK standard

FunctionalLocation	Platform	System	Eq. type	Eq. id	Subunit	Component
PLAT-24VD003	PLAT	24	VD	003	-	-
PLAT-70-BS-32020-L42B	PLAT	70	BS	32020	L42	B
PLAT-97XY10439A	PLAT	97	XY	10439	-	A
PLAT-63ACD123-M01	PLAT	63	ACD	123	M01	-

Table 3b: FunctionalLocation breakdown for SFI standard

FunctionalLocation	Platform	Gr.[1]	Gr.[2]	Gr.[3]	Detail code	Unknown
PLAT-662.27D-2-DE006	PLAT	6	62	662	27D	2-DE006
PLAT-713.24C-1	PLAT	7	71	713	24C	1
PLAT-769.21-1	PLAT	7	76	769	21	1

Extracting Additional Information About FunctionalLocation Additional information about the FunctionalLocations exists in the oil and gas company's internal ERP-system. This includes further grouping of the tags, along with a short textual description of each asset. An export of this data has been made available for use in this study. In the ERP-system a FunctionalLocation belongs to exactly one of the following groups: (1) Cables (2) Electrical Equipment (3) Fire and Gas (4) Instrument (5) Junction Box (6) Lifting Lug (7) Line (8) Manual Valve (9) Master Equipment (10) Miscellaneous (11) Signal (12) Special (13) Telecom (14) Unknown. Based on this information, a new feature *Tag_type* is constructed. Each FunctionalLocation belongs to one of the groups above, and is matched based on part of the FunctionalLocation. The textual descriptions of the tags are also extracted, and the values are stored as a feature *Tag_description*.

[1] Primary group
[2] Secondary group
[3] Tertiary group

The new features are added to the original dataset by the following procedure: A dictionary D is constructed from the additional dataset and filled with key-value pairs on the form $\{FunctionalLocation : (Tag\_type,\ Tag\_description)\}$. For each of the work orders in the original dataset, D is queried with the FunctionalLocation to extract the corresponding values of Tag_type and Tag_description. The values are then appended to the original dataset.

Domain Term Dictionary & Binary Feature Vectors By inspecting the data at hand, several descriptive domain terms for each of the FMCs were identified. These are terms that are often used in the *Text* field of work orders concerned with a specific failure. A term may be closely associated with one or more of the FMCs. An example of FMCs and their related terms are shown in Table 4. The terms are used as features in the classification, where the values indicate whether or not the term is present in the text of the work order. This is achieved by first defining the vocabulary V as the set of domain terms, $|V| = 217$. Each of the documents (work order texts) are then represented as a *Bag-of-words*, which is an unordered list of the unigram terms contained in the document. The Bag-of-words are then used to represent the document as a feature vector with binary term weights, where the vector space model is defined by V. This results in binary feature vectors of length $|V|$, that represents the presence and absence of domain terms in the corresponding document. The terms are added as features, where the values are given by the feature vectors.

Table 4: Example of FMCs and related domain terms

FMC	Terms
INL - Internal Leakage	[internal, leakage, sweat, oil, diesel,. . .]
LOO - Low Output	[low, output, pressure, flow, pump,. . .]
PLU - Plugged/Choked	[plugged, choked, drain, flow, block,. . .]

4.3 FailureModeCode Classification

Random Forest, Gradient Boosting Machine and Stacked Ensemble were used for classifying FMCs in our experiments. This section explains the different algorithms used.

Random Forest Random Forest is an ensemble learning method frequently used for multi-class classification. Characteristics of Random Forest include robustness to noise, outliers and overfitting [4]. It works by generating a large number of tree-structured classifiers, then taking the mode of the output classes of the individual trees to conclude a final classification. Random Forest can in other words be defined as a collection of F tree-classifiers $\{T_k(\boldsymbol{x}),\ k = 1, 2, \dots, F\}$, where $\boldsymbol{x}$ is the input vector of features, in our case, features that describe a work order. Each tree then predicts a class c_k based on the input vector, resulting in a total of F individual classifications, $\{T_k(\boldsymbol{x}) = c_k,\ k = 1, 2, \dots, F\}$. The classification output c of the Random Forest, is the most popular class among these predictions. As Random Forests are ensembles of tree-based classifiers, they also support direct use of categorical variables. This is convenient due to the nature of the dataset.

Gradient Boosting Machine Gradient Boosting Machine (GBM) is a machine learning technique that uses *boosting* and steepest gradient optimization, in order to produce highly accurate, robust and interpretable classification models [7]. Boosting is a method used to increase the performance of a given learning algorithm. The method works by combining the predictions of several weak classifiers in order to make a final prediction based on a weighted majority vote [10]. The weak classifiers are built by sequentially applying a classification algorithm to repeatably weighted modified versions of the input data. In steepest gradient optimization, the loss function of the model is minimized by iteratively moving in the direction of the steepest descent (the negative of the gradient).

Stacked Ensamble Stacking (also known as Super Learning or Stacked Regression) is a concept that aims to reduce overfitting and minimize the error rate by deducing the biases of multiple generalizers with respect to a provided learning set [19]. Instead of selecting the best predictor in a set of classifiers, the idea of stacking is to combine several classifiers into a more accurate predictor [3]. Unlike bagging and boosting, stacking is a second-layer meta learner that intends to ensemble the base learners. The particular stacking algorithm used in this work is the Super Learner developed by van der Laan et al. [14]. The Super Learner ensemble was proven to be an asymptotically optimal system for learning. In our experiments, models from Random Forest and Gradient Boosting Machine were used as base learners.

4.4 Environment

All of our experiments are conducted using H2O, an open source machine learning and predictive analytics platform. An attractive feature of H2O is that it supports the direct use of categorical variables in tree-based algorithms. By using e.g. scikit we would have had to one-hot encode or label encode the features first. One-hot encoding makes the feature space orders of magnitude larger, while label encoding imposes an ordinal relationship that are not necessarily true.

H2O offers a feature called AutoML, which automatically trains and tunes several modals using multiple algorithms on the same dataset. AutoML currently supports Random Forest, Gradient Boosting Machine, Deep Learning, Extremely Randomized Trees, Generalized Linear Modeling, and a Stacked Ensemble of all the models. We used AutoML to get an overview of how a large selection of candidate models performed on our dataset.

5 Experiments

In order to examine the effect of domain knowledge, experiments were done using popular supervised machine learning techniques. The experiments compare the performance of models with incorporated domain knowledge, with models constructed using no form of domain knowledge (baseline).

5.1 Evaluation metric

Accuracy is a commonly used metric for evaluating classifiers. Unfortunately accuracy can be misleading when dealing with imbalanced classes, as it is very sensitive to class distribution [11]. Assume a classification problem where the goal is to identify disastrous malfunctions in an oil drilling system. Five records belong to the malfunction class, while 995 records belong to the class where nothing is wrong. A majority classifier that always predicts that nothing is wrong would get an accuracy of 99.5%. This is seemingly a good result, but the truth is that the classifier performs terrible for the problem at hand. Other metrics such as Cohen's Kappa and F-score punish classifiers harder for mislabeling small classes.

In order to remove random chance from the performance evaluation, Ben-David [1] argued for using Cohen's Kappa instead of accuracy. Cohen's Kappa (1) is a coefficient of interjudge agreement for nominal scales, where mere chance is excluded [6]. It's original intent was to measure the agreement between two or more people observing the same phenomenon, but in classification it measures the agreement between the classifier and the truth. It ranges from -1 (disagreement) through 0 (random) to 1 (agreement) [2]. If a classifier always predicts the majority class, Cohen's Kappa will be 0, while a classifier with Cohen's Kappa > 0 performs better than the majority classifier.

$$K = \frac{P_0 - P_c}{1 - P_c} \tag{1}$$

where P_0 is the relative observed agreement among raters, and P_c is the hypothetical probability of chance agreement. In a multi-class classification task, Cohen's Kappa can be computed from the resulting confusion matrix using Equation (2)

$$K = \frac{n \sum_{i=1}^{C} x_{ii} - \sum_{i=1}^{C} x_{.i} x_{i.}}{n^2 - \sum_{i=1}^{C} x_{.i} x_{i.}} \tag{2}$$

where x_{ii} is the number of true positives (the main diagonal), n is the total number of samples, C is the number of classes, and $x_{.i}, x_{i.}$ are the column and row counts respectively [9].

Cohen's Kappa has been used as a metric in many multi-class classification problems involving imbalanced datasets [5,12]. Landis and Koch [15] have suggested that the Cohen's Kappa can be interpreted as follows: < 0: No agreement, 0.00-0.20: Slight agreement, 0.21-0.40: Fair agreement, 0.41-0.60: Moderate agreement, 0.61-0.80: Substantial agreement, 0.81-1.00: Almost perfect agreement.

We have also included accuracy and macro average F_1-score in our results for a better understanding of the performance, but we will focus on Cohen's Kappa in the discussion of the results. Cohen's Kappa was chosen as the primary metric over macro average F_1-score, since macro averaging weights the F_1-score of each class equally. Misclassified records belonging to minor classes would then negatively affect the score more than desired.

To ensure that all minor classes are represented in the testing set, we use 5-fold cross-validation. The metrics presented are the average score of the five folds.

5.2 Results & Discussion

H2O's AutoML was initially used to get an intuition of which algorithms worked best on our dataset. A comparison of the different algorithms is listed in Table 5. These results are only meant to give an overview of the algorithms, and are not used as baseline. Stacked Ensemble, Random Forest and Gradient Boosting Machine stood out in the results, and we decided to employ these methods in further experiments.

Table 5: Comparison of algorithms using baseline dataset

	Cohen's Kappa	Accuracy	Macro average F_1 score
Stacked ensamble	0.3794	0.4176	0.3003
Random Forest	0.3755	0.4133	0.3113
Gradient Boosting Machine	0.3638	0.4020	0.2914
Extremely randomized trees	0.3282	0.3728	0.2610
Deep learning	0.2685	0.3146	0.1859
Generalized linear modeling	0.2417	0.2965	0.1377

Baseline In order to get a broader assessment of the effects of incorporating domain knowledge, the experiments were conducted using three classification algorithms. A random grid search was used in order to find the best hyperparameters for both Random Forest and Gradient Boosting trees. The difference in performance using various hyperparameters turned out to be insignificant, and the best baseline results were very similar to the results from H2O's AutoML. The final hyperparameters used in the experiments are listed in Table 6.

Baseline results were computed using all the initial features, without utilizing domain knowledge. The baseline results are presented at the top of Table 7. Using Landis and Koch's interpreation of the Cohen's Kappa, the performance of the baseline classifiers can be considered as *Fair*.

Table 6: Hyperparameters after random grid search

	Random Forest	Gradient Boosting Machine
Number of trees	50	50
Maximum tree depth	10	13
Minimum samples in leaf node	1	30
n-bins (numerical values histogram)	10	20
Learn rate	-	0.1
Row sample rate per tree	-	0.8
Distribution function	-	Multinomial

Table 7: Results from the baseline models and the models that incorporates domain knowledge, along with the improvement.

	Cohen's Kappa	Accuracy	Macro average F_1 score
Baseline			
Stacked Ensemble	0.3800	0.4185	0.2982
Random Forest	0.3782	0.4165	0.3087
Gradient Boosting Machine	0.3664	0.4044	0.2902
With domain knowledge			
Stacked Ensemble	0.4515	0.4851	0.3641
Random Forest	0.4454	0.4796	0.3665
Gradient Boosting Machine	0.4363	0.4700	0.3674
Improvement			
Stacked Ensemble	0.0715	0.0666	0.0659
Random Forest	0.0672	0.0631	0.0578
Gradient Boosting Machine	0.0699	0.0656	0.0772

With domain knowledge Table 7 also shows evaluation metrics after domain knowledge was incorporated. The Cohen's Kappa shows that the performance of the classifiers has increased to *Moderate*. The difference in metrics between the baseline model and the model with incorporated domain knowledge is illustrated in Figure 1. The Cohen's Kappa has increased with $\sim 7\%$ for all algorithms. The fact that Cohen's Kappa has increased more than the accuracy, implies that the classifier has a greater improvement on the smaller classes than the large ones.

As shown in Figure 2, many of the extracted features (*Equipment_type*, *System*, *Equipment_type_id*, *Equipment_id* and *Tag_type*) are of great importance in the improved model. The sum of the importance of all features equals 100%.

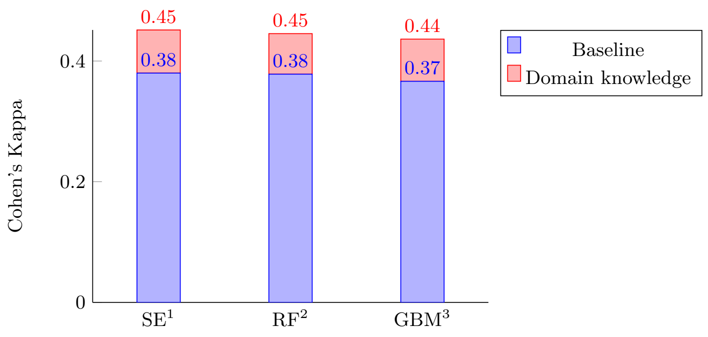


Fig. 1: Cohen's Kappa for baseline compared to models with incorporated domain knowledge. [1]Stacked Ensemble, [2]Random Forest, [3]Gradient Boosting Machine.

The features that were extracted using domain knowledge constitute 38.97% of the total feature importance.

Since the classification problems discussed in related work are not directly comparable to our problem, it is hard to compare the results. Contrary to the related work, we extract new features from structured fields with the use of domain knowledge. In combination with features extracted from unstructured fields, the performance of the classifiers increased significantly.

6 Conclusion

Finding good features for supervised learning in high dimensional industrial settings is challenging. Using domain knowledge to extract new features had a significant impact on classifying work orders in the dataset used in this study. The increase in performance outweighed differences in the top three classification algorithms used. It is crucial to obtain a thorough understanding of the dataset at hand, in order to perceive where features can be encoded. Although it took quite some time to understand how and where domain knowledge could be applied, we were unable to achieve similar results using other techniques. While the exact procedure used to extract new features in this work is very specific, the overall approach is quite generic. Canvassing the dataset for encoded features and using predefined domain term dictionaries, can be applied in similar work as well as in other domains.

It would be interesting to further investigate techniques to extract features from textual fields, and compare results for techniques with and without making use of domain knowledge. Instead of just looking at each word separately, building n-grams of words could impact the results. The domain term dictionary can also be extended to include more terms. Features could also be extracted using named entity recognition, word2vec, doc2vec or tf-idf. Acquiring more data from

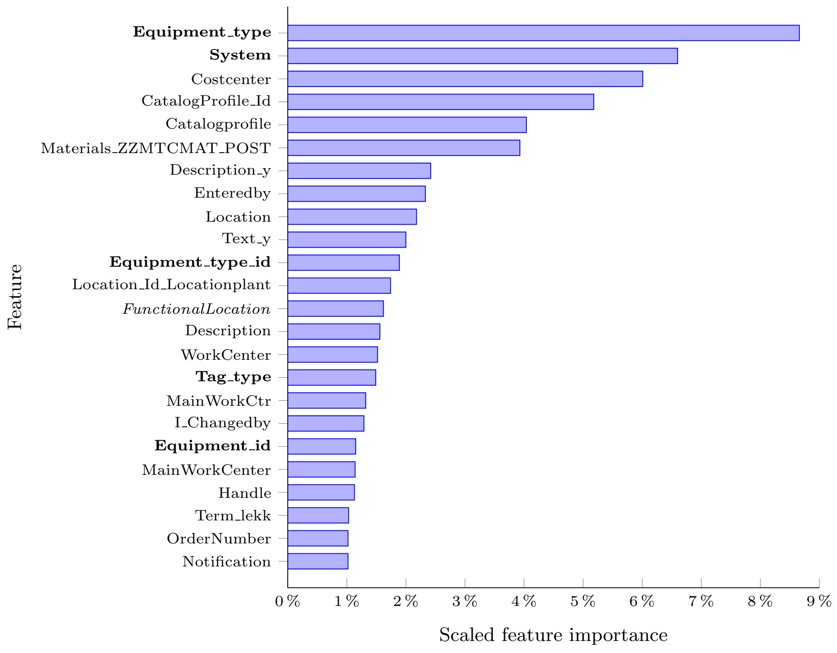


Fig. 2: Features with a scaled importance of $> 1\%$ from the Random Forest model with domain knowledge. The features with bold text are the ones that were extracted from FunctionalLocation.

other oil platforms for training and validation could improve the performance of the models as well. It would also be interesting to see how well the system performs when highly correlated FMCs are grouped together. The measures mentioned above could improve the classifiers extensively.

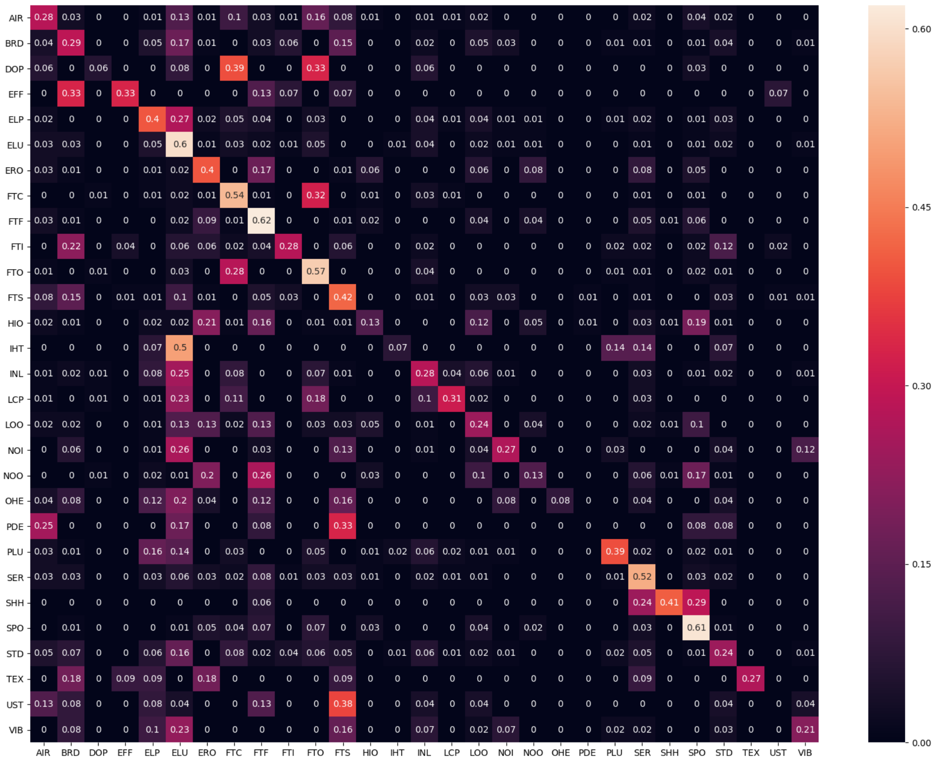
References

1. Ben-David, A.: A lot of randomness is hiding in accuracy. Engineering Applications of Artificial Intelligence **20**(7), 875 – 885 (2007). https://doi.org/10.1016/j.engappai.2007.01.001, `http://www.sciencedirect.com/science/article/pii/S0952197607000061`
2. Ben-David, A.: Comparison of classification accuracy using cohen's weighted kappa. Expert Systems with Applications **34**(2), 825 – 832 (2008). https://doi.org/10.1016/j.eswa.2006.10.022, `http://www.sciencedirect.com/science/article/pii/S0957417406003435`

3. Breiman, L.: Stacked regressions. Machine Learning **24**(1), 49–64 (Jul 1996). https://doi.org/10.1023/A:1018046112532
4. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (Oct 2001). https://doi.org/10.1023/A:1010933404324
5. Chaplot, D.S., et al.: Predicting student attrition in moocs using sentiment analysis and neural networks. In: AIED 2015 Workshop Proceedings. vol. 3. AIED (2015), `http://ceur-ws.org/Vol-1432/islg_pap2.pdf`
6. Cohen, J.: A coefficient of agreement for nominal scales. Educational and Psychological Measurement **20**(1), 37–46 (1960). https://doi.org/10.1177/001316446002000104, `https://ci.nii.ac.jp/naid/30026339172/en/`
7. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Ann. Statist. **29**(5), 1189–1232 (10 2001). https://doi.org/10.1214/aos/1013203451
8. Gaines, B.: An ounce of knowledge is worth a ton of data: Quantitative studies of the trade-off between expertise and data based on statistically well-founded empirical induction. In: Proceedings of the Sixth International Workshop on Machine Learning. pp. 156–159. Morgan Kaufmann (1989)
9. García, S., et al.: A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Computing **13**(10), 959 (Dec 2008). https://doi.org/10.1007/s00500-008-0392-y
10. Hastie, T., et al.: The Elements of Statistical Learning. Springer Series in Statistics, Springer New York Inc. (2001)
11. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering **21**(9), 1263–1284 (Sept 2009). https://doi.org/10.1109/TKDE.2008.239
12. Jeni, L.A., et al.: Facing imbalanced data–recommendations for the use of performance metrics. In: Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction. pp. 245–251. ACII '13, IEEE Computer Society, Washington, DC, USA (2013). https://doi.org/10.1109/ACII.2013.47
13. Kubat, M., et al.: Machine learning for the detection of oil spills in satellite radar images. Machine Learning **30**(2), 195–215 (Feb 1998). https://doi.org/10.1023/A:1007452223027
14. van der Laan, M.J., et al.: Super learner. Statistical Applications in Genetics and Molecular Biology **6**(1) (2007). https://doi.org/10.2202/1544-6115.1309
15. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. Biometrics **33**(1), 159–174 (1977), `http://www.jstor.org/stable/2529310`
16. Spelman, M., et al.: Digital Transformation Initiative Oil and Gas Industry. Tech. rep., World Economic Forum (01 2017)
17. Standard Norge: NORSOK STANDARD Z-DP-002. `http://www.standard.no/pagefiles/942/z-dp-002r1.pdf` (1995), [Accessed 15 March 2018]
18. Wilcox, A.B., Hripcsak, G.: The role of domain knowledge in automating medical text report classification. Journal of the American Medical Informatics Association **10**(4), 330–338 (2003)
19. Wolpert, D.H.: Stacked generalization. Neural Networks **5**, 241–259 (1992)
20. Xantic: SFI Group System - Product Description. `https://www.xantic.net/internet/files/products/amos/sfi/supportdocuments/ProductDescription.pdf` (2005), [Accessed 15 March 2018]

Appendix B

Data



	AIR	BRD	DOP	EFF	ELP	ELU	ERO	FTC	FTF	FTI	FTO	FTS	HIO	IHT	INL	LCP	LOO	NOI	NOO	OHE	PDE	PLU	SER	SHH	SPO	STD	TEX	UST	VIB
AIR	0.28	0.03	0	0	0.01	0.13	0.01	0.1	0.03	0.01	0.16	0.08	0.01	0	0.01	0.01	0.02	0	0	0	0	0	0.02	0	0.04	0.02	0	0	0
BRD	0.04	0.29	0	0	0.05	0.17	0.01	0	0.03	0.06	0	0.15	0	0	0.02	0	0.05	0.03	0	0	0	0.01	0.01	0	0.01	0.04	0	0	0.01
DOP	0.06	0	0.06	0	0	0.08	0	0.39	0	0	0.33	0	0	0	0.06	0	0	0	0	0	0	0	0	0	0.03	0	0	0	0
EFF	0	0.33	0	0.33	0	0	0	0	0.13	0.07	0	0.07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.07	0
ELP	0.02	0	0	0	0.4	0.27	0.02	0.05	0.04	0	0.03	0	0	0	0.04	0.01	0.04	0.01	0.01	0	0	0.01	0.02	0	0.01	0.03	0	0	0
ELU	0.03	0.03	0	0	0.05	0.6	0.01	0.03	0.02	0.01	0.05	0	0	0.01	0.04	0	0.02	0.01	0.01	0	0	0	0.01	0	0.01	0.02	0	0	0.01
ERO	0.03	0.01	0	0	0.01	0.02	0.4	0	0.17	0	0	0.01	0.06	0	0	0	0.06	0	0.08	0	0	0	0.08	0	0.05	0	0	0	0
FTC	0	0	0.01	0	0.01	0.02	0.01	0.54	0.01	0	0.32	0	0.01	0	0.03	0.01	0	0	0	0	0	0	0.01	0	0.01	0	0	0	0
FTF	0.03	0.01	0	0	0	0.02	0.09	0.01	0.62	0	0	0.01	0.02	0	0	0	0.04	0	0.04	0	0	0	0.05	0.01	0.06	0	0	0	0
FTI	0	0.22	0	0.04	0	0.06	0.06	0.02	0.04	0.28	0	0.06	0	0	0.02	0	0	0	0	0	0	0.02	0.02	0	0.02	0.12	0	0.02	0
FTO	0.01	0	0.01	0	0	0.03	0	0.28	0	0	0.57	0	0	0	0.04	0	0	0	0	0	0	0.01	0.01	0	0.02	0.01	0	0	0
FTS	0.08	0.15	0	0.01	0.01	0.1	0.01	0	0.05	0.03	0	0.42	0	0	0.01	0	0.03	0.03	0	0	0.01	0	0.01	0	0	0.03	0	0.01	0.01
HIO	0.02	0.01	0	0	0.02	0.02	0.21	0.01	0.16	0	0.01	0.01	0.13	0	0	0	0.12	0	0.05	0	0.01	0	0.03	0.01	0.19	0.01	0	0	0
IHT	0	0	0	0	0.07	0.5	0	0	0	0	0	0	0	0.07	0	0	0	0	0	0	0	0.14	0.14	0	0	0.07	0	0	0
INL	0.01	0.02	0.01	0	0.08	0.25	0	0.08	0	0	0.07	0.01	0	0	0.28	0.04	0.06	0.01	0	0	0	0	0.03	0	0.01	0.02	0	0	0.01
LCP	0.01	0	0.01	0	0.01	0.23	0	0.11	0	0	0.18	0	0	0	0.1	0.31	0.02	0	0	0	0	0	0.03	0	0	0	0	0	0
LOO	0.02	0.02	0	0	0.01	0.13	0.13	0.02	0.13	0	0.03	0.03	0.05	0	0.01	0	0.24	0	0.04	0	0	0	0.02	0.01	0.1	0	0	0	0
NOI	0	0.06	0	0	0.01	0.26	0	0	0.03	0	0	0.13	0	0	0.01	0	0.04	0.27	0	0	0	0.03	0	0	0	0.04	0	0	0.12
NOO	0	0	0.01	0	0.02	0.01	0.2	0	0.26	0	0	0	0.03	0	0	0	0.1	0	0.13	0	0	0	0.06	0.01	0.17	0.01	0	0	0
OHE	0.04	0.08	0	0	0.12	0.2	0.04	0	0.12	0	0	0.16	0	0	0	0	0	0.08	0	0.08	0	0	0.04	0	0	0.04	0	0	0
PDE	0.25	0	0	0	0	0.17	0	0	0.08	0	0	0.33	0	0	0	0	0	0	0	0	0	0	0	0	0.08	0.08	0	0	0
PLU	0.03	0.01	0	0	0.16	0.14	0	0.03	0	0	0.05	0	0.01	0.02	0.06	0.02	0.01	0.01	0	0	0	0.39	0.02	0	0.02	0.01	0	0	0
SER	0.03	0.03	0	0	0.03	0.06	0.03	0.02	0.08	0.01	0.03	0.03	0.01	0	0.02	0.01	0.01	0	0	0	0	0.01	0.52	0	0.03	0.02	0	0	0
SHH	0	0	0	0	0	0	0	0	0.06	0	0	0	0	0	0	0	0	0	0	0	0	0	0.24	0.41	0.29	0	0	0	0
SPO	0	0.01	0	0	0	0.01	0.05	0.04	0.07	0	0.07	0	0.03	0	0	0	0.04	0	0.02	0	0	0	0.03	0	0.61	0.01	0	0	0
STD	0.05	0.07	0	0	0.06	0.16	0	0.08	0.02	0.04	0.06	0.05	0	0.01	0.06	0.01	0.02	0.01	0	0	0	0.02	0.05	0	0.01	0.24	0	0	0.01
TEX	0	0.18	0	0.09	0.09	0	0.18	0	0	0	0	0.09	0	0	0	0	0	0	0	0	0	0	0.09	0	0	0	0.27	0	0
UST	0.13	0.08	0	0	0.08	0.04	0	0	0.13	0	0	0.38	0	0	0.04	0	0.04	0	0	0	0	0	0	0	0	0.04	0	0	0.04
VIB	0	0.08	0	0	0.1	0.23	0	0	0	0	0	0.16	0	0	0.07	0	0.02	0.07	0	0	0	0.02	0.02	0	0	0.03	0	0	0.21

Figure B.1: Row normalized confusion matrix based on the predictions of a Random Forest classifier. The label in front of each row symbolize the actual class of the samples in the row, while the columns symbolize the predicted class. Cell value x, y indicate the percentage of samples that are actually class x, that were predicted as belonging to class y.

Table B.1: Column density on final 4960 rows.

	Column	Fields not null	Percent not null
1	id	4960	100.0%
2	Plants_Id_PltforWorkCtr	4960	100.0%
3	ValidFrom	4960	100.0%
4	WorkStart	4960	100.0%
5	NotificationInformation_Id	4960	100.0%
6	CatalogProfile_Id	4960	100.0%
7	GeneralObjectNumber_Id_Objectnumber_OBJNR	4960	100.0%
8	InspectionCatalogType_Id	4960	100.0%
9	Material_Id_Material	4960	100.0%
10	MaterialGroups_Id	4960	100.0%
11	NotificationInformation_Id_Notification	4960	100.0%
12	NotificationInformation_Id_Referencenotif	4960	100.0%
13	NotificationTypes_Id	4960	100.0%
14	UnitsOfMeasurement_Id	4960	100.0%
15	Timecreated	4960	100.0%
16	Workorder_Id_Order	4960	100.0%
17	Accepted	4960	100.0%
18	Catalogprofile	4960	100.0%
19	Catalogtype	4960	100.0%
20	Chance	4960	100.0%
21	i_Changedat	4960	100.0%
22	i_Changedby	4960	100.0%
23	i_Changedon	4960	100.0%
24	Codegroup	4960	100.0%
25	Coding	4960	100.0%
26	Complaintqty	4960	100.0%
27	Usage	4960	100.0%
28	startTime	4960	100.0%
29	LastPMR	4960	100.0%
30	Plannedrelease	4960	100.0%
31	MainWorkCtr	4960	100.0%
32	ObjectClass	4960	100.0%
33	Objectnumber	4960	100.0%
34	OrCostObj	4960	100.0%
35	Order_AUFNR	4960	100.0%
36	Ordercategory	4960	100.0%
37	OrderType_y	4960	100.0%
38	OriginalPlnDt	4960	100.0%
39	OutstandingQty	4960	100.0%
40	Plannedclodat	4960	100.0%
41	PlannedCompltn	4960	100.0%
42	Plant_y	4960	100.0%
43	Techcompletion	4960	100.0%
44	Processgroup	4960	100.0%
45	ProdProcNo	4960	100.0%
46	ProfitCenter	4960	100.0%
47	Reachedstatus	4960	100.0%
48	RecoveryIndic	4960	100.0%
49	Release	4960	100.0%

50	Respcostcntr	4960	100.0%
51	SalesOrdItem	4960	100.0%
52	Sequencenumber	4960	100.0%
53	Status	4960	100.0%
54	Statuschange	4960	100.0%
55	CompletedbySAP	4960	100.0%
56	Completiontime	4960	100.0%
57	Completndate	4960	100.0%
58	ReferenceTime	4960	100.0%
59	Order	4960	100.0%
60	Origin	4960	100.0%
61	OriginOfNotification	4960	100.0%
62	Plannooper	4960	100.0%
63	PltforWorkCtr	4960	100.0%
64	POdate	4960	100.0%
65	Primarylang	4960	100.0%
66	Priority_y	4960	100.0%
67	PriorityType	4960	100.0%
68	Proddat	4960	100.0%
69	Referencedate	4960	100.0%
70	Refquantity	4960	100.0%
71	ComplTimeatSAP	4960	100.0%
72	Reqendtime	4960	100.0%
73	Reqstart	4960	100.0%
74	Reqstarttime	4960	100.0%
75	Required	4960	100.0%
76	RequiredEnd	4960	100.0%
77	ReturnDelivQty	4960	100.0%
78	Returnedon	4960	100.0%
79	SAPNetnotif	4960	100.0%
80	Timestamp	4960	100.0%
81	WorkCenter	4960	100.0%
82	Text_y	4960	100.0%
83	ObjectTypeCR	4960	100.0%
84	Objectnumber_OBJNR	4960	100.0%
85	ObjectID_Profile	4960	100.0%
86	i_ObjectID	4960	100.0%
87	i_CostEstimateNo	4960	100.0%
88	Counter	4960	100.0%
89	Createdat	4960	100.0%
90	Createdby	4960	100.0%
91	i_Createdon	4960	100.0%
92	DefQty(exter)	4960	100.0%
93	Defqty(int)	4960	100.0%
94	Deliveryitem	4960	100.0%
95	i_Description	4960	100.0%
96	Estimated	4960	100.0%
97	Handle	4960	100.0%
98	ID	4960	100.0%
99	InspectionLot	4960	100.0%
100	Item	4960	100.0%
101	Itemmatdoc	4960	100.0%
102	Itempurdoc	4960	100.0%

103	Nodenumber	4960	100.0%
104	Notifdate	4960	100.0%
105	Notification	4960	100.0%
106	Notifictntype	4960	100.0%
107	Notiforigin	4960	100.0%
108	NotifTime	4960	100.0%
109	Notiftimezone	4960	100.0%
110	Locationplant	4960	100.0%
111	Failure code joined	4960	100.0%
112	CostEstimateNo	4960	100.0%
113	CostingSheet	4960	100.0%
114	Changedat	4960	100.0%
115	Changedby	4960	100.0%
116	Changedon	4960	100.0%
117	Client	4960	100.0%
118	Close	4960	100.0%
119	COArea	4960	100.0%
120	CompanyCode	4960	100.0%
121	PM_ActivityType	4960	100.0%
122	Costcollector_CCKEY	4960	100.0%
123	Createdon	4960	100.0%
124	BacklgDat	4960	100.0%
125	MainWorkCenter	4960	100.0%
126	Planningplant	4960	100.0%
127	PlannerGroup	4960	100.0%
128	Currency	4960	100.0%
129	Description_x	4960	100.0%
130	OrderNumber	4960	100.0%
131	OrderType_x	4960	100.0%
132	Workorder_Id	4960	100.0%
133	Description_y	4960	100.0%
134	BaselineCompletion	4960	100.0%
135	BasicStartDate	4960	100.0%
136	EndofWork	4960	100.0%
137	Workorder_Id_Order_AUFNR	4960	100.0%
138	Location_Id_Plant	4960	100.0%
139	Location_Id_Locationplant	4960	100.0%
140	GeneralObjectNumber_Id	4960	100.0%
141	CompanyCodes_Id_Reqcocode	4960	100.0%
142	CompanyCodes_Id_CompanyCode	4960	100.0%
143	eventReferenceKey	4960	100.0%
144	Plants_Id_Locationplant	4960	100.0%
145	Plants_Id_Plant	4960	100.0%
146	WBS_Elements_Id	4960	100.0%
147	Workorder_Id_Order_REFNR	4960	100.0%
148	BasicFinDate	4960	100.0%
149	eventTransformerId	4960	100.0%
150	tagIds	4960	100.0%
151	MaintenanceType	4960	100.0%
152	Description_OrderTypes	4960	100.0%
153	OrderTypes	4960	100.0%
154	Plant_x	4960	100.0%
155	Application	4960	100.0%

156	ApplicDate	4960	100.0%
157	FunctionalLocation	4960	100.0%
158	index	4960	100.0%
159	OrderTypes_Id	4960	100.0%
160	JointVenture	4960	100.0%
161	endTime	4960	100.0%
162	description	4960	100.0%
163	Equitytype	4960	100.0%
164	JVObjectType	4960	100.0%
165	Enteredby	4960	100.0%
166	type	4960	100.0%
167	originalId	4960	100.0%
168	Estimatedcosts	4960	100.0%
169	Priority_x	4959	99.98%
170	Reportedby	4936	99.516%
171	NotificationNumber	4925	99.294%
172	RAKey	4905	98.891%
173	Costcenter	4905	98.891%
174	Longtext	4772	96.21%
175	Longtxtexists	4758	95.927%
176	Text_x	4758	95.927%
177	Closed	4585	92.44%
178	Errorrecords	3277	66.069%
179	ApproverID_ZZAPPROVER	2342	47.218%
180	ApproverID_ZZAPPROVER_POST	2342	47.218%
181	ApprovDateandTime_ZZDATZEIT_POST	2342	47.218%
182	ApprovDateandTime_ZZDATZEIT	2342	47.218%
183	Location	2189	44.133%
184	SystemCondition	2171	43.77%
185	LaborST_ZZMTCLST_POST	665	13.407%
186	LaborST_ZZMTCLST	665	13.407%
187	Materials_ZZMTCMAT	596	12.016%
188	Materials_ZZMTCMAT_POST	596	12.016%
189	Completed	319	6.431%
190	JIB/JIBEClass	201	4.052%
191	JIB/JIBESbClsA	201	4.052%
192	Misc_ZZMTCMIS_POST	159	3.206%
193	Misc_ZZMTCMIS	159	3.206%
194	Released	56	1.129%
195	WBSelement	55	1.109%
196	FunctionalArea	55	1.109%
197	PlntWorkCenter	55	1.109%
198	NotifPhase	54	1.089%
199	SynergiReference	51	1.028%
200	Scenario	47	0.948%
201	Services_ZZMTCSVC	27	0.544%
202	Services_ZZMTCSVC_POST	27	0.544%
203	BusinessArea	20	0.403%
204	TMRNo	9	0.181%
205	Referencenotif	7	0.141%
206	Tasks	6	0.121%
207	Order_REFNR	5	0.101%
208	LaborOT_ZZMTCLOT_POST	1	0.02%

209	LaborOT_ZZMTCLOT	1	0.02%
210	Addressnumber	1	0.02%
211	Vendor	0	0.0%
212	VendorBatch	0	0.0%
213	ObjectType	0	0.0%
214	VendorMatNo	0	0.0%
215	Usageofparts	0	0.0%
216	Operatingsys	0	0.0%
217	Objectautom	0	0.0%
218	i_WBSElement	0	0.0%
219	Objnorealact	0	0.0%
220	Objectnumber_/ISDFPS/OBJNR	0	0.0%
221	ObjectTypeProfile	0	0.0%
222	PartnerType	0	0.0%
223	SalesGroup	0	0.0%
224	Plantformat	0	0.0%
225	SAPSystemID	0	0.0%
226	R/3Systemtype	0	0.0%
227	Referenceno	0	0.0%
228	MPNMaterial	0	0.0%
229	MainWorkCenterPlant	0	0.0%
230	SerialNumber	0	0.0%
231	ReportType	0	0.0%
232	SAPcomponent	0	0.0%
233	Plantversions	0	0.0%
234	SalesOrg	0	0.0%
235	SalesOrderLS	0	0.0%
236	i_SalesOrder	0	0.0%
237	SalesOffice	0	0.0%
238	RevisionLevel	0	0.0%
239	RSheader	0	0.0%
240	R/3Release	0	0.0%
241	QMOrder	0	0.0%
242	PurchGroup	0	0.0%
243	PurchasingOrg	0	0.0%
244	PurchasingDoc	0	0.0%
245	ProdVersion	0	0.0%
246	Prodorder	0	0.0%
247	Prodhierarchy	0	0.0%
248	StatusSAPNotif	0	0.0%
249	StLocILotStock	0	0.0%
250	StorLocation	0	0.0%
251	Devicedata_x	0	0.0%
252	Timezoneorig	0	0.0%
253	TSegmentexists	0	0.0%
254	POnumber	0	0.0%
255	UII	0	0.0%
256	Unitofmeasure	0	0.0%
257	Template	0	0.0%
258	Delete	0	0.0%
259	Matofversion	0	0.0%
260	DisTranGroup	0	0.0%
261	Reqcocode	0	0.0%

262	Reqcostcenter	0	0.0%
263	Reqorder	0	0.0%
264	Envirinvest	0	0.0%
265	SalesOrder	0	0.0%
266	Scale	0	0.0%
267	SettlementCE	0	0.0%
268	Statistical	0	0.0%
269	TaxJur	0	0.0%
270	CostingVariant	0	0.0%
271	Department	0	0.0%
272	Telephone_USER1	0	0.0%
273	Deletionflag	0	0.0%
274	CUOrder	0	0.0%
275	UserResponsible	0	0.0%
276	CUDesignNo	0	0.0%
277	Variancekey	0	0.0%
278	Workpermit	0	0.0%
279	EqRental_ZZMTCEQR	0	0.0%
280	Regindicator	0	0.0%
281	Refurbishment	0	0.0%
282	EqRental_ZZMTCEQR_POST	0	0.0%
283	LogicalSystem	0	0.0%
284	MultipleItms	0	0.0%
285	ObjectID	0	0.0%
286	Isolation	0	0.0%
287	IsoDescription	0	0.0%
288	Investreason	0	0.0%
289	Investprofile	0	0.0%
290	InterestProf	0	0.0%
291	Overheadkey	0	0.0%
292	Personresp	0	0.0%
293	Planlineitems	0	0.0%
294	Integplanning	0	0.0%
295	Identification	0	0.0%
296	HandlingType	0	0.0%
297	G/Laccount	0	0.0%
298	Processcat	0	0.0%
299	Extorderno	0	0.0%
300	Created	0	0.0%
301	Costcollector_PKOSA	0	0.0%
302	MaterialGroup	0	0.0%
303	Function	0	0.0%
304	Applicant	0	0.0%
305	Allocationset	0	0.0%
306	Devicedata_y	0	0.0%
307	DistrChannel	0	0.0%
308	Division	0	0.0%
309	Docrequired	0	0.0%
310	Equipment	0	0.0%
311	Frontend	0	0.0%
312	Functionalloc	0	0.0%
313	Costcollector_KSTEMPF	0	0.0%
314	AcctgIndicator	0	0.0%

315	Installation	0	0.0%
316	i_LogicalSystem	0	0.0%
317	Manufacturer	0	0.0%
318	MasterEquip	0	0.0%
319	MatDocYear	0	0.0%
320	Material	0	0.0%
321	MaterialDoc	0	0.0%
322	Delivery	0	0.0%
323	Databasesystem	0	0.0%
324	Customer	0	0.0%
325	CustMaterial	0	0.0%
326	ContLaborST_ZZMTCCST_POST	0	0.0%
327	ContLaborST_ZZMTCCST	0	0.0%
328	ContLaborOT_ZZMTCCOT_POST	0	0.0%
329	ContLaborOT_ZZMTCCOT	0	0.0%
330	Add-onID	0	0.0%
331	Add-OnRelease	0	0.0%
332	i_Adressnumber	0	0.0%
333	Author	0	0.0%
334	AuxAcctAsmnt_1	0	0.0%
335	Batch	0	0.0%
336	CMNumber	0	0.0%
337	ClientID	0	0.0%
338	CCtr:Truepost	0	0.0%
339	AutoEstCosts	0	0.0%
340	i_CostingVariant	0	0.0%
341	CriticalPart	0	0.0%
342	i_Currency	0	0.0%
343	Telephone_USER3	0	0.0%

Table B.2: Failure Mode Codes with description and count.

Code	Description	Count	% of total
OTH	Other	1178	18.50%
ELU	External leakage utility medium	466	7.32%
SER	Minor in-service problems	414	6.51%
SPO	Spurious operation	413	6.49%
FTF	Fail to function on demand	383	6.02%
ELP	External leakage process medium	303	4.76%
ERO	Erratic output	279	4.38%
FTO	Fail to open on demand	271	4.26%
LOO	Low output	269	4.23%
FTC	Fail to close on demand	268	4.21%
INL	Internal leakage	251	3.94%
AIR	Abnormal instrument reading	221	3.47%
BRD	Breakdown	214	3.36%
STD	Structural deficiency	197	3.10%
UNK	Unknown	188	2.95%
NOO	No output	171	2.69%
HIO	High output	154	2.42%
FTS	Fail to start on demand	144	2.26%
LCP	Leakage in closed position	102	1.60%
PLU	Plugged/choked	98	1.54%
NOI	Noise	77	1.21%
VIB	Vibration	61	0.96%
FTI	Fail to function as intended	50	0.79%
DOP	Delayed operation	36	0.57%
OHE	Overheating	25	0.39%
UST	Spurious stop	24	0.38%
SHH	Spurious high alarm level	17	0.27%
EFF	Electric failure	15	0.24%
IHT	Insufficient heat transfer	14	0.22%
PDE	Parameter deviation	12	0.19%
TEX	Loss of EX-integrity	11	0.17%
ELF	External leakage - fuel	9	0.14%
FOV	Faulty output voltage	8	0.13%
LOA	Load drop	5	0.08%
SLL	Spurious low alarm level	5	0.08%
VLO	Very low output	4	0.06%
FRO	Fail to rotate	4	0.06%
FTR	Fail to regulate	2	0.03%
STP	Fail to stop on demand	1	0.02%
FDC	Fail to disconnect	1	0.02%

Table B.3: Full list of domain terms used for feature extraction from the Text field.

aapne	ex	kjoeling	overoppheting	stop
aapner	ex-	knekt	paa	stopp
abnormal	ex-integrity	kommando	parameter	structural
alarm	external	kompressor	plugg	struktuell
alarmnivaa	fail	kvelt	plugged	stuck
avik	failure	last	plugget	svett
avlesning	falsk	lav	point	sweat
avvik	falskt	lavt	posisjon	tap
batteri	fast	leak	position	tar
block	fault	leakage	powersupply	temperatur
brann	faulty	lekasje	pressure	tett
breakdown	feedback	lekk	problemer	tilbakemelding
brent	feil	lekkasje	problems	transfer
broken	feiler	level	process	treg
bytte	feilmaaling	load	prosess	trip
choked	flow	loss	pumpe	trykk
clogged	forventet	low	punktsegment	turbin
close	fuel	lukke	reading	uberegnelig
closed	function	lukker	repair	ulyd
closing	grader	lukket	reparer	unormal
compressor	ground	lyd	sammenbrudd	unstable
cool	havari	maaling	sas	ustabil
corrosion	heat	maanedtlig	segment	utility
daglig	high	mangel	sensor	utilstrikkelig
daily	hoey	medium	sent	utsatt
damaged	hoeyt	minor	service	value
defect	hot	mistet	setting	valve
defekt	i/o	monthly	shutdown	varm
deficiency	ikke	moving	siger	varme
degree	in-service	mulighet	sikring	vent
delayed	indicator	next	skad	ventil
demand	indikator	nivaa	skade	verdi
detector	ingen	no	skru	vibrasjon
detektor	insp-monitor-repair	noise	slow	vibration
deviation	inspection	nytte	smaa	virker
diesel	inspeksjon	olje	spenning	voltage
drain	instrument	open	sprekk	weekly
drivstoff	insufficient	operasjon	sprukket	within
drop	integritet	operation	spurious	0,
e-	intended	opportunity	start	0.
ekstern	intern	output	starter	=
electric	internal	overfoering	stenge	
elektrisk	jordfeil	overhaul	stengt	
erratic	kjoeler	overheating	stoey	

Table B.4: TF-IDF terms, sorted by weight. Some terms are censored on request from the providers of the dataset.

[redacted]	skiftet	[redacted]	aft	previously
[redacted]	historik	new	geir	lagt
detektor	[redacted]	uten	gas	siden
ringo	hvad	aapner	johnny	solenoid
[redacted]	lt	tommy	[redacted]	tid
[redacted]	open	line	feilsoeking	resatt
dect	job	tu	isolere	problemer
dg	deck	oil	ute	[redacted]
tg	leak	needed	positioner	problem
ab	actuator	bs	kom	slange
vcm	morten	fejlen	parts	gear
observed	testet	nei	[redacted]	fungerer
seal	el	mellom	production	alvheim
pump	aapen	ba	eksempel	nytt
alt	stk	af	bestilt	till
tank	trond	noe	ref	closed
linje	rom	vaskat	kort	suggest
linser	arne	location	service	mye
vasket	vifte	drain	noen	bestilling
stengt	start	dekk	mekanisk	ccr
[redacted]	helt	ned	boer	activities
[redacted]	sjekk	bf	mangler	aktre
feedback	stb	close	nivaa	luft
sb	hv	brannpumpe	water	jobbet
stenger	mulig	fuel	flens	spjeld
kapasitet	sensor	sap	type	xv
[redacted]	flow	diesel	pakning	notifikation
damper	tor	safety	impact	proevd
pumpen	fast	flyttes	untill	valves
ronny	aktuator	feilsoeke	similar	level
lekasje	ar	magne	ser	lekkasjen
virker	sjekke	tanker	heater	fejl
vann	dine	[redacted]	jeg	manometer
olje	aapne	ingvar	samme	intern
[redacted]	instrument	[redacted]	repairing	skiftat
tar	stenge	mange	[redacted]	batteri
foerste	[redacted]	mot	vadoy	pries
ventiler	nr	bar	respect	kjoert
kah	erik	uke	gir	ola
change	rtl	lst	environment	felt
trykk	item	litt	sendt	room
filter	steam	bestille	signal	
garten	flere	fwd	pm	
vud	kalibrere	[redacted]	mech	