Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Sander Magnussen Neraas

Artificial Intelligence in Projects:

Using machine learning approaches to assert whether variation orders can predict time delay of individual activities

Master's thesis in Engineering and ICT
Supervisor: Nils Olsson
June 2022

NTNU
Kunnskap for en bedre verden

Sander Magnussen Neraas

Artificial Intelligence in Projects:

Using machine learning approaches to assert whether variation orders can predict time delay of individual activities

Master's thesis in Engineering and ICT
Supervisor: Nils Olsson
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Preface

This master's thesis marks the end of the five-year Master of Science programme *Engineering and ICT* at the Norwegian University of Science and Technology (NTNU). The thesis is written as a specialisation in *Project and Quality Management*. The thesis was conducted in the spring of 2022 and counts towards 30 credits.

Through a varied selection of courses within artificial intelligence and machine learning, I have developed a fascination for real-world use cases of AI. The technology behind it, the vast number of applications, and the sometimes incredible accuracy are all factors that I am truly fascinated by. The topic *Artificial Intelligence in Projects* was for me a possibility to combine my knowledge with a field of study that has great potential for AI and ML applications. Project Management is a field where a significant amount of data is recorded, which makes it perfect for ML analyses. This combination is an opportunity for me to apply my knowledge and experience to a real-world application.

Acknowledgement

First, I would like to thank to my supervisor, Nils Olsson, for his invaluable guidance for this master's thesis. Nils Olsson has enabled me to work efficiently and targeted on this thesis by guiding when necessary. He has trusted me with working independently, but has also guided me to specific areas of research and related literature. Furthermore, he has put me in contact with the collaborating companies and gathered a selection of key personnel. With this, he laid the foundation for this master's thesis and has ensured an interesting and relevant topic.

I would also like to thank the collaborating companies. Through biweekly meetings, they have helped me understand the dataset and given a foundation on which the analyses are conducted. Furthermore, I would like to thank the company that owns the project data for sharing it. It has been a great motivation to work with real project data.

Thanks to my family for supporting me during these five years. You have motivated me in times of need and helped me focus on the end goal.

Lastly, I would like to thank my friends at NTNU. Without you, my five years in Trondheim would not have been the same.

Sander Magnussen Neraas

June 2022

Abstract

Variation orders (VOs) are generally accepted as inevitable in construction projects and can affect the scope of the project. Among the major effects of variation orders is time delay. Several papers have investigated whether it is possible to predict the delay of a project. However, little research has been found on using VOs to predict time delays. Furthermore, most of the research predicted the delay of the project and not for individual activities. This master's thesis investigates whether it is possible to use VOs to predict the delay of individual project activities. A compound dataset has been created from individual tables of a project database. The tables were extracted from a project management software. Four tree-based models were trained on the dataset; Decision Tree, Random Forest, AdaBoost and Gradient Boosting. A permutation feature importance analysis and a shapley additive explanations analysis were conducted to quantify whether the variation orders contributed to the predictions.

The overall best performing model was Random Forest with a recall on `DELAYED_START` of 92.7% and 91.8% on `DELAYED_FINISH`. Both importance analyses showed that the features extracted from VOs were insignificant for the predictions. Thus, this thesis concludes that classifying delayed activities from variation orders is implausible in this dataset. The most important features in predicting the delay of the activities were the date features, indicating that for this project the delays follow a trend. This master's thesis has shown that an early warning system for activity delays can be created with this method. Further work includes training a similar model on a multitude of projects and testing on multiple projects.

Sammendrag

Endringsordre er sett på som uunngåelige i byggeprosjekter og kan påvirke omfanget av prosjektet. Blant de viktigste effektene av endringsordre er forsinkelser. Flere forskningsartikler har undersøkt om det er mulig å predikere forsinkelsen av et prosjekt. Det er imidlertid funnet lite forskning på bruk av endringsordre for å predikere tidsforsinkelser. Felles for det meste av forskningen er at de predikerer forsinkelsen av prosjektet i sin helhet, og ikke enkelte aktiviterer i prosjektet. Denne masteroppgaven undersøker om det er mulig å bruke endringsordre til å predikere forsinkelsen av individuelle prosjektaktiviteter. Et sammensatt datasett er opprettet fra individuelle tabeller i en prosjektdatabase. Tabellene ble hentet fra en prosjektstyringsprogramvare. Fire trebaserte modeller ble trent på datasettet; Decision Tree, Random Forest, AdaBoost og Gradient Boosting. To viktighetsanalyser er implementert for å kvantifisere viktigheten av endringsordre i prediksjonene; permutation feature importance og shapley additive explanations.

Den mest presise modellen testet var Random Forest med en *recall* på `DELAYED_START` på 92,7% og 91,8% på `DELAYED_FINISH`. Begge viktighetsanalysene viste at variablene hentet fra endringsordrene var ubetydelige for prediksjonene. Dermed konkluderer denne oppgaven med at klassifisering av forsinkede aktiviteter ut i fra endringsordre er usannsynlig i dette datasettet. De viktigste variablene i å forutsi forsinkelsen av aktivitetene var datovariabler, noe som indikerer at for dette prosjektet følger forsinkelsene en trend. Denne masteroppgaven har vist at et tidligvarslingssystem for aktivitetsforsinkelser kan lages med denne metoden. Videre arbeid inkluderer videre utvikling av en lignende modell på en rekke prosjekter som burde testes på flere prosjekter.

Table of Contents

List of Figures

List of Tables

Acronyms

AI artificial intelligence. 3, 4, 10, 11, 21, 22, 44

ANN artificial neural network. 11, 12

APM autonomous project management. 11

CA chatbot assistant. 11

EDA exploratory data analysis. 27

EF early finish. 1, 2, 8, 34, 42, 47, 48, 54

ES early start. 1, 2, 8, 34, 40, 42, 47, 48, 53, 54

fmGA fast messy genetic algorithm. 11

FN false negative. 13, 14, 32, 43, 55

FP false positive. 13, 14, 32, 44, 55

IA integration and automation. 11

LF late finish. 2, 8, 34

LS late start. 2, 8, 34

MAPE mean absolute percentage rrror. 12

ML machine learning. 1–4, 10–14, 20, 22, 23, 30, 33, 44, 54

MLBPM machine learning-based project management. 11, 12, 22, 44, 45, 51, 55

NaN not a number. 27, 28, 30

PFI permutation feature importance. vii, viii, 21, 22, 32, 35, 37–40, 45–47, 49, 50, 54, 55

PM project management. 1, 3–5, 10, 11

SHAP shapley additive explanations. vii, 21, 32, 35, 41, 42, 54, 55

SVM support vector machine. 11, 12

TF total float. 8

TN true negative. 13, 32

TO time overrun. 12, 56

TP true positive. 13, 32, 43

VO variation order. viii, 1–3, 5, 7, 9, 10, 22–24, 26, 29, 32, 40, 44–47, 51, 53–55

VR variation request. 7

WBS work breakdown structure. 2, 5–7, 53

XAI explainable artificial intelligence. 13, 20, 22, 30

1 Introduction

This master's thesis will investigate the importance of variation orders (VOs) in project time delay. More specifically, it will investigate whether VOs can be used to predict the time delay of individual project activities in terms of delay in relation to early start (ES) and early finish (EF).

1.1 Current Situation

In project management (PM), VOs are generally accepted as inevitable (Akinsola et al., 1997; Sunday, 2010). There are a multitude of factors that might cause VOs, such as change in plan by the owner (Al Hammadi, 2009) and inadequate project objectives (Keane et al., 2010). The effects of VOs are many and varied. VOs can lead to an increase in project cost and duration (Al Hammadi, 2009), degrade relations between owner and contractor through claims and disputes (Benachour, 2018) and cause loss of productivity (Bower, 2000; Oyewobi et al., 2016).

VOs have been widely studied with research on the causes of VOs (Al Hammadi, 2009; Hsieh et al., 2004), the effects caused by VOs (Al Hammadi, 2009; Alsuliman et al., 2012; Benachour, 2018; Jergeas, 2008), the costs of variations (Bower, 2000; Hanif et al., 2016; Oladapo, 2007) and many more subfields. This thesis will focus on the time effects of VOs.

Although VOs and delays in projects are widely studied, there are not many studies on predicting project delay at the activity level. El-Kholy (2013), Gondia et al. (2020) and Peško et al. (2017) all investigated project delays with the use of machine learning methods. However, this was the delay of the project as a whole. Aarvold and Hartvig (2021) investigated if it was possible to predict whether activities started on time or not. However, none of these studies included VOs. This study will close the gap by investigating whether it is possible to predict the delay of an activity, both for start and finish times by including VOs. In addition, it will highlight the predictive ability of VOs on activity delays.

1.2 Problem Scope

The main objective of this thesis is to investigate whether VOs can be used to identify the time delay of individual project activities. The thesis will implement four common tree-based machine learning (ML) models on a single project dataset to classify whether an activity will start and or finish on time. With a compound dataset of combined features, it should be possible to identify whether the VO features are important for the classifications.

Since VOs are a common occurrence in construction projects, it is imperative to fully understand their effects. ML algorithms are well-studied methods capable of capturing complex interactions and conditions, perfect for understanding VOs. There are some studies investigating the effects of VOs. However, this study is intended to provide additional insight into how VOs contribute to activity delay and whether it can be used to predict delays. Predicting activity delays can be a great additional support for project managers and could be added as an early warning system for activities that are prone to delays. To evaluate whether VOs can be used to predict activity delay and to what extent delays can be predicted, the following research questions have been formulated:

RQ1 How can data from large project databases be extracted and processed to be used for an ML analysis?

RQ2 Can VOs be used to predict whether a related activity will be delayed?

RQ3 What are the most dominant features for predicting whether an activity will be delayed?

1.3 Delimitations

This master's thesis will not investigate the severity of the delays. It is restricted to classifying whether an activity will be delayed in terms of delay to ES and EF, and will not consider the amount that the activity is delayed by. Additionally, an activity can still be considered on time if it is within the late start (LS) and or late finish (LF). This thesis is delimited to considering that delay is defined by ES and EF.

In addition to the definition of a delay, the thesis will not consider any logical connections between activities. The sequencing of activities (e.g., an activity can only start when another is finished) will not be considered.

1.4 Applicability

The method presented in this paper can function as an early warning system for activities that have the potential to exceed the original ES and EF. Although this is not within the scope of the master's thesis, the application will be discussed to put the results into a real-world application. Activities and components of the work breakdown structure (WBS) might still be executed on time in relation to the LS and LF. However, this early warning system will signal which activities are prone to delays and has to be addressed further. By including VOs in these models, the early warning system can function while creating activities and updating resources and activities through

VOs. If applied to a multitude of projects, this model might give accurate predictions for activities that have the potential to be delayed.

1.5 Order of Information

The thesis will first give theoretical background in Section 2. This section is divided into four parts. The first part will elaborate on PM, specifically project scope management and project time management, and give a theoretical background for VOs. The second part will explain how artificial intelligence (AI) is implemented in PM today, before highlighting how it is applied to scope change management through related research. The third part will provide an important theoretical background for ML, specifically the methods used in this thesis. The last part will summarize the theoretical background. Section 3 will first highlight how the literature search was conducted. Furthermore, it will highlight how the data were extracted, cleaned, and preprocessed to be used in a ML analysis. Additionally, this section highlights how the training was performed and how the results were analysed. The last part of Section 3 assesses the quality of the method, specifically the reliability and validity. Section 4 presents the results of the training and secondary importance analyses. Section 5 discusses the performance, validity of the approach, and asserts the research questions before introducing limitations. Lastly, a conclusion to the research questions is made in Section 6, before suggesting further work.

2 Theoretical background

This section will introduce the theoretical background for this thesis, and is divided into four main parts. The first part will highlight key theory regarding PM for this thesis. The second part will give an overview of how AI is used in PM, and will provide relevant research related to this thesis. The third part will elaborate on the relevant ML algorithms, performance metrics, and discuss the explainability of ML models. The last part will summarise the theory.

2.1 Project Management

In the PMBOK Guide, Project Management Institute (PMI) defines PM as "the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements." (Project Management Institute, 2001). Project governance involves planning of the project and the monitoring of the progress and success of the project (Rolstadås et al., 2014). Figure 1 shows the project governance cycle. The figure shows the key steps in order to start and complete a project, among these are monitoring which is essential for project governance.

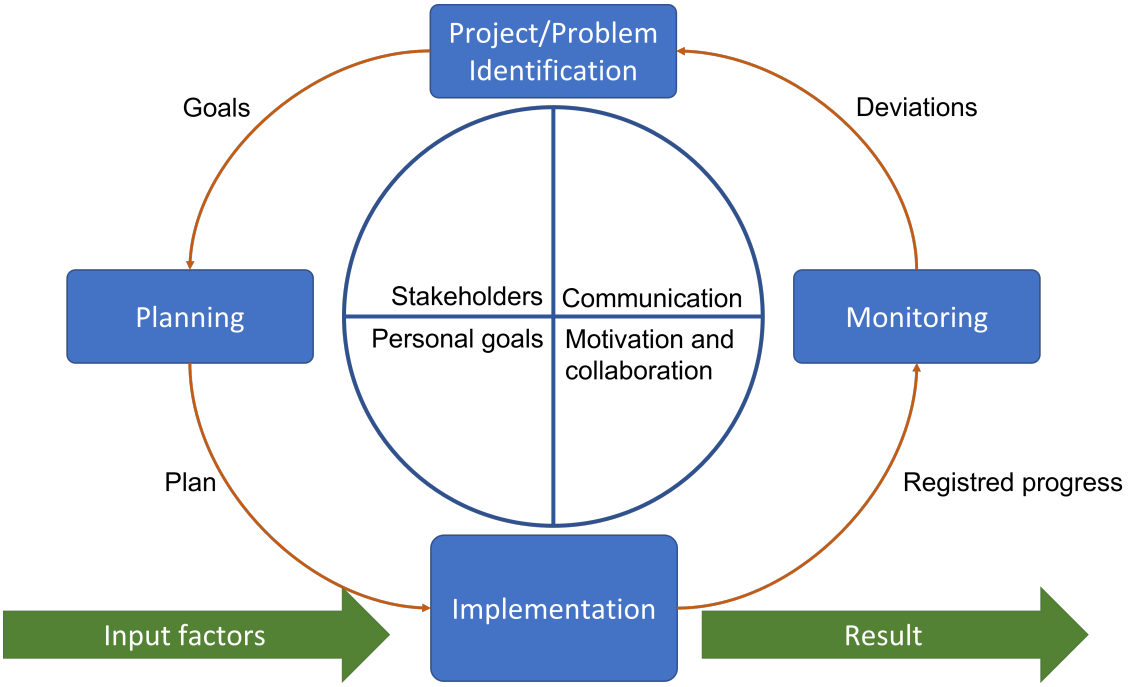


Figure 1: Project governance cycle. Adopted from (Rolstadås et al., 2014, p. 57).

The key objective of PM is generally considered to be to complete the project deliverables within budget, time and scope (De Wit, 1988). These three success criteria have been key to evaluating the project's success for several decades. In addition to these three success criteria, several others have been proposed, such as stakeholder satisfaction (Atkinson, 1999; Baccarini, 1999; Khang & Moe,

2008; Kumaraswamy & Thorpe, 1996), PM quality (Baccarini, 1999; Kumaraswamy & Thorpe, 1996), and many more. Even though the objectives for projects have mostly been constant, many projects still fail to reach its goals. PMI's 2016 *Pulse of the Profession* report found that only 53% of projects completed within the original budget and 49% of projects completed on time (PMI, 2016). This report is based on respondents from organisations worldwide with 51% of respondents from North America. Both delays and cost overruns are considered a universal phenomenon in the construction industry (Zidane & Andersen, 2018).

Flyvbjerg (2013) indicated that the root cause of underperformance is due to project planners systematically underestimating the risk of complexity and scope changes. He highlights that managers often overestimate the benefits of a project and underestimate the cost and time aspects. Project delay is a common phenomenon in the construction industry. The amount of delay will vary depending on the project, but it is present globally (Gondia et al., 2020). Zidane and Andersen (2018) categorised eleven major delay factors for the Norwegian construction industry. Of the 202 respondents, 189 listed poor planning and scheduling as one of the main causes of delay. Furthermore, 30% of the respondents listed variation orders as a major delay factor.

The following sections will detail the theoretical background for PM on which this thesis is based. It will focus on project time management and project scope management. Additionally, it will focus on VOs and causes and effects of them. In light of the research goal and especially RQ2 and RQ3, the time aspect of PM is the most important and will be the focus of the following chapters.

2.1.1 Project Scope Management

The project scope, and specifically the managing of it, is one of the most important parts of PM (Khan, 2006). "Project Scope Management includes the processes required to ensure that the project includes all the work required, and only the work required, to complete the project successfully." (Project Management Institute, 2001, p. 412). It includes several processes that are used to define and control what the project includes:

i. Collect requirements

ii. Define Scope

iii. Create WBS

iv. Verify Scope

v. Control Scope

After (i.) defining and documenting the needs of the stakeholders and (ii.) developing a detailed description of the project, the WBS is created to divide project work and deliverables into manageable parts. The separate elements, components, and services of the WBS must be systematically and logically divided (Rolstadås et al., 2014). In addition, it is a crucial part for effective follow-up on the project, as it is a framework for planning work. There are several ways to decompose the WBS, including functional, physical and geographical (Ibrahim et al., 2009). Whatever way the WBS is structured, the smallest structure of the WBS should be the work package (Smith, 2016). A simple WBS can be seen in Figure 2.

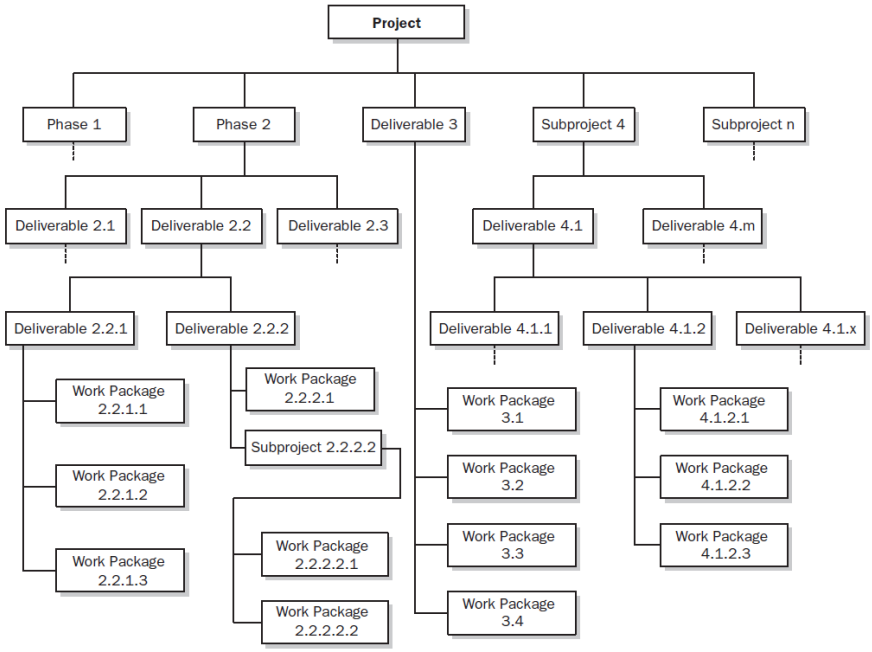


Figure 2: A sample WBS decomposed down to work packages (Project Management Institute, 2001, p. 119).

Effective project scope management is linked to effective management of other parts of the project and will thus increase the likelihood of project success. Key to project scope management is baselines. A baseline is a fixed schedule that is used to measure the performance of the project. All variations in the project will be tracked using the baseline scope. Establishing the baseline indicates the formal end of project planning and the beginning of project execution and control (Upland Software, 2022). The initial scope of the project will likely not remain unchanged, as changes are a key aspect of projects. Due to the complexity of construction processes, Oyewobi et al. (2016) argues that changes to the original scope are almost inevitable. Since all projects are unique in nature, they are prone to changes during the project lifetime. Changes in construction projects are generally accepted as inevitable (Akinsola et al., 1997). Hanna et al. (2002, p. 1) defined change as "any event that results in a modification of the original scope, execution time,

or cost of work". One key component of project scope management is scope change management.

To remedy changes in the scope of the project, scope change management is an important tool. The project scope will clarify the project objectives, making sure that all members are on the same page (Millhollan, 2008). Additionally, the project scope will clearly define project completion. After the preliminary scope and a WBS are set, the occurrence of changes is almost inevitable. VOs are the most common and recommended way to handle changes in projects. Focusing on the definition of change made by Hanna et al. (2002), a variation order is a written change to the original scope of the project that may impact the scope of work, schedule, cost, or quality of the project. Any change to the work originally agreed upon is treated as a variation. Correct handling of changes in projects is key to a successful project. Millhollan (2008) accredits scope change management for protecting projects against scope creep and contributing to managing stakeholder expectations. Here, a scope creep is the unauthorised scope changes that may *creep* into the scope through verbal or written instructions (Khan, 2006). When a change is requested, it is recorded as a variation request (VR). These VRs must be properly documented in order to be approved. When the VR later has been reviewed, it will be included in the schedule as a VO. When baselines are updated (i.e., the creation of a new baseline), updates from VOs are added to the baseline. VOs will be discussed further in Section 2.1.3.

2.1.2 Project Time Management

As one of the key objectives of PM, ensuring that a project is completed on time is of great concern to project managers. However, the complexity of construction projects can lead to delays (Gondia et al., 2020). Project time management includes several processes that are in place to manage the completion of the project within the set time (Project Management Institute, 2001):

i. Define activities
ii. Sequence activities
iii. Estimate activity resources
iv. Estimate activity durations
v. Develop schedule
vi. Control schedule

The activities of the project are the specific actions needed to complete the work. Rolstadås et al. (2014) defines activities as a fitting collection of work tasks that require resources to be

executed, where a resource can be humans executing a task, materials, or machines. The activities of a project are the basis for scheduling. After activities have (i.) been defined, they need to be (ii.) sequenced. Project Management Institute (2001) defines four logical relationships between activities to sequence them; finish-to-start, finish-to-finish, start-to-start, and start-to-finish. Of the four logical relationships, most activities are sequenced by the finish-to-start method, i.e. linked activities can only start when its precursor has finished. Activities are usually sequenced by the critical path methodology with the precedence diagramming method (Project Management Institute, 2001).

A key component of project time management is (iv.) estimating the activity duration to find when activities can start and finish. There are four common attributes to an activity; ES, EF, LS, and LF. ES defines the earliest start time of an activity. EF is defined as the earliest time an activity can end and is calculated as ES plus the duration of the activity ($EF = ES + duration$). The LF is the latest time an activity can finish and is calculated as the LF minus the duration ($LS = LF - duration$) (Rolstadås et al., 2014). These four attributes are used from the first linked activity to the last. Figure 3 shows how the critical path method can be used to determine scheduling flexibilities. The critical path is the longest sequence of activities that must be finished on time in order to complete the project (here A-C-D-E). The total float (TF) is the duration that the start or finish of an activity can vary by, i.e. the difference between the start times and finish times. The TF is defined by $TF = LS - ES = LF - EF$.

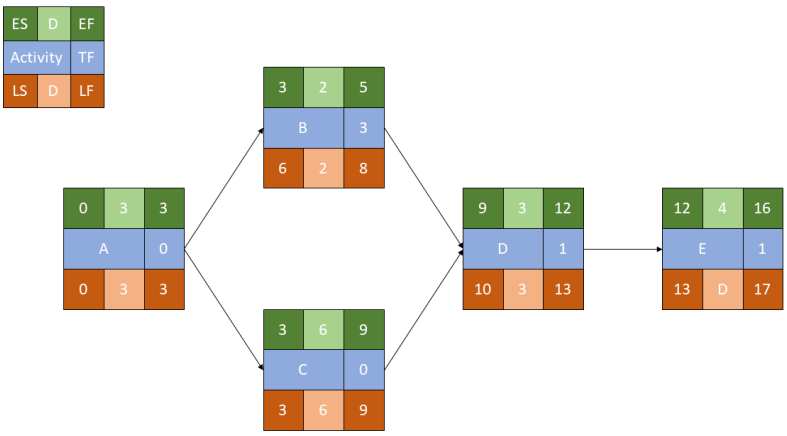


Figure 3: Critical Path Method for four linked activities.

After resources and durations are estimated, the schedule is (v.) developed and (vi.) controlled. Activities are usually controlled with project management software. Since the 1980's, project management software has become increasingly prevalent in the industry. A survey conducted

by Liberatore and Pollack-Johnson (2003) indicated that more than 90% of the respondents used project management software to some extent, with the most common tools being Microsoft Projects and Primavera Project Planner. Project management software can be used for both planning and controlling, and many offer a great variety of features and the ability to handle large projects. A project management software usually has a user interface which the user can use for planning and controlling, while the raw data is stored and processed in the database. Such tools are well suited for both time and scope management.

2.1.3 Variation Orders

VOs are common in all phases of a construction project, but the proportion of VOs can vary significantly throughout the project (Alsuliman et al., 2012). VOs typically contain addition, omission, substitution, or alteration to work in the initial scope (Akinsola et al., 1997). Arain and Pheng (2005) separates between beneficial and detrimental VOs. Beneficial VOs improve quality, reduce cost and time, or even reduce the complexity of the project. Detrimental VOs, on the other hand, are those that reduce owner value or have a negative impact on the project.

In the literature, there has been identified many different causes for variation orders. Through a literature study, Keane et al. (2010) grouped causes of variation orders into three categories; owner-related, consultant-related and contractor-related. Owner-related variations included change of plans or scope, inadequate project objectives and replacement of materials/procedures. Consultant-related variations included change in design, technology change and poor coordination. Contractor-related variations included lack of involvement in design, poor procurement process, differing site conditions and lack of communication. In addition to the three main categories, Keane et al. (2010) identified five other variations; weather conditions, safety considerations, change in economic conditions, sociocultural factors and unforeseen problems. Al Hammadi (2009) found through an exploratory study using a series of interviews that the main cause of variation for oil and gas construction projects is change of plan by the owner. The second major contributor to variation was concluded to be the contractor due to changes in site condition and design conflicts.

VOs can have a multitude of different effects on the project. Al Hammadi (2009) found that the two main effects of variation orders in oil and gas construction projects were an increase in the cost and duration of the project. The degredation of labour productivity and disputes was found to be less prevalent. Various other sources have also included cost and time overruns as direct effects of variation orders (Alsuliman et al., 2012; Benachour, 2018; Hsieh et al., 2004; Jergeas, 2008). In addition to cost and time overruns, variation orders can often end in claims and

disputes (Benachour, 2018; Bower, 2000) and degrade relationship between owner and contractor (Benachour, 2018). There are also several indirect effects of VOs. These can be either the result of tasks being performed at the same time or be "ripple" effects from logical links, i.e., connected tasks (Bower, 2000). The same article includes the following indirect effects; rework, time lost from pausing activities, financial costs, loss of productivity, revisions to project reports, and increased sensitivity to delay.

2.2 Artificial Intelligence in Project Management

There have been many substantial leaps in technology during the information age. In particular, processors have become minuscule in size with exponentially increasing power every second year (Kaul, 2017). The vast amount of computing power available without enormous financial costs has led to significant developments in AI and ML. The following historical events capture the rapid development of AI in the last few decades. In 1997, Garry Kasparov was beaten by IBM's supercomputer DeepBlue (Auth et al., 2019). In 2016, Google's AlphaGo beat 18-time world champion Lee Sedol in Go. In 2019, Google's AlphaStar was ranked Grandmaster in all three races of the real-time strategy game Starcraft II, thus beating 99.8% of all players (Vinyals et al., 2019). The increasingly complex problems solved by AI over time show great potential. AlphaStar is an AI agent that handles real-time visual input, controls hundreds of units, and can withstand players' random behaviour. Although these AI agents have exhibited immense capabilities, AI in PM is not at the same level. The Google Duplex system is one of the latest developments within AI in PM (Auth et al., 2019). Duplex can handle appointments automatically using natural language processing. This is surely a great improvement over paper-based PM, but there is still room for improvement.

Although AI has made immense progress in other fields, the use of AI in PM is still limited. There are some distinct reasons for this. Firstly, many companies are still not at a necessary readiness level to properly implement AI solutions (AlSheibani et al., 2018). Additionally, there is some controversy as to whether or not to implement AI. Wang (2019) identified three factors for this controversy; people fear that AI will take their job away, people fear that AI will fail and people fear abuse of AI. The last reason for the limited progress in PM is that it is a field where the project is highly dependent on the knowledge of the project manager (Project Management Institute, 2001, p.13). Much like a game of Go, each project demands creativity, strategies and intuition (Auth et al., 2019, p. 29). All factors that are backed by knowledge of the game. Despite the challenges, there is a continuous AI development within PM, albeit to a lesser degree compared to other fields. The following section will address current implementations of AI in the field of PM.

2.2.1 Implementation of Artificial Intelligence in Project Management

Mark Lahmann, Partner and Leader Transformation Assurance of PwC Switzerland, identified four steps in which AI will be implemented in PM (PwC, 2018); integration and automation (IA), chatbot assistant (CA), machine learning-based project management (MLBPM) and autonomous project management (APM). These four steps represent the evolution of AI in PM, and are presented as a step-by-step integration of AI. While IA and CA focus primarily on simplifying work by automating repetitive tasks, streamlined processes, and complete simple human-performed tasks, MLBPM is the first step where the power of AI can truly enhance PM. MLBPM helps the project manager analyse the project and will have an impact on the decisions (Wang, 2019). Through ML models, the project manager could access information and analyses collected from previous projects that will advise the project manager. This could give the project manager a reliable prediction to support in monitoring and can aid in decision making. According to PwC, "predictive project analytics will be the most disruptive innovation in PM in the next ten years" (PwC, 2018).

There have been several different applications of MLBPM in recent decades. They vary in scope and in AI approaches. Wauters and Vanhoucke (2014) used a support vector machine (SVM) to forecast time and cost variables. Cheng et al. (2010) created a Evolutionary Support Vector Machine Inference Model to estimate the final project cost with Estimate at Completion. The learning model combines fast messy genetic algorithm (fmGA) and SVM where the first technique is used for feature selection. There are several articles exploring the use of artificial neural network (ANN) in PM. Dursun and Stoy (2016) explores whether ANN can be used to estimate the cost using the multistep ahead approach. Heravi and Eslamdoost (2015) used ANN to estimate labour productivity in construction projects. Jin and Zhang (2011) implemented ANN to model the risk allocation decision-making process in public–private partnership projects.

The final step of the evolution, APM, could almost completely remove all workers from PM. As Mark Lehmann explains it: "Similar to self-driving cars, autonomous PM would only need limited input and intervention from a human project manager" (PwC, 2018). However, he considers it unlikely that we will see fully autonomous PM within the next 10-20 years. One application that touches the field of APM is Deloitte's Predictive Project Analytics (Auth et al., 2019). In addition to complexity and success analyses and risk assessments, the application can perform employee selection for project teams. This product is one of several components needed for APM and shows that APM might be possible in the future.

2.2.2 AI in Scope Change Management & Related research

Zidane and Andersen (2018) found that 60% of project managers contributed delay to a slow or poor decision-making process. More specifically, the respondents listed late decisions, wrong decisions and re-play on decisions as key delay factors in the Norwegian construction industry. As elaborated in the previous section, MLBPM can help the project manager in decision making by forecasting algorithms and predictive models. This section will highlight current MLBPM research that could aid project managers and could contribute to project success.

Peško et al. (2017) used ANN and SVM to estimate the cost and duration of 198 Serbian construction projects. Their results indicated that the SVM performed best in estimating cost with a mean absolute percentage rrror (MAPE) of 7.06%. However, estimating the duration proved to be more difficult, with the best model performing at a MAPE of 22.77%. This difference also reflects itself in the literature, where there is a significant amount of research on modelling and predicting the cost of projects compared to the duration or delays of projects. However, there are several papers that have explored ML principles to predict delays in projects. El-Kholy (2013) presented two models to predict the delay percentage of construction projects in Egypt. The approach consisted of collecting data from 20 Egyptian construction projects and predicting the delay percentage from 14 causes of delay. The best model to predict delay percentages was the modular neural network with a MAPE of 39.8%. Gondia et al. (2020) used both a decision tree and a naive Bayes model to predict time overrun (TO) over construction projects. The paper classifies projects into three classes; $< 30\%$ TO, $30\% - 60\%$ TO and $> 60\%$ TO. The predictions are made from a structured multivariate dataset consisting of 51 projects with nine risk sources. The nine risk sources are categorised by their probability. The recall obtained from their best classifier (Naive Bayes) obtained results between 75% and 83.3% recall on the labels.

Although there is some research on the delay of construction projects as a whole, little research is published on predicting individual activity delay. Aarvold and Hartvig (2021) classified delay of individual activities in a single project of a service provider in the industry. The activities were classified as start hit or start miss based. The best model was the Random Forest model with an F1 score of 89.8%. However, in other sectors there is more research. Choetkiertikul et al. (2015) developed a model to predict the delay of a subset of tasks for software project management. In addition to individual software project tasks, they used relations between them (i.e. networked data). This addition significantly improved the predictions compared to traditional approaches that classified each task individually. The results gave an F1 score of 56%-76%, a 37% improvement compared to not having networked data.

2.3 Machine Learning Theory

At the forefront of making intelligent machines and software programs is machine learning. As explained by Mohri et al., "Machine learning can be broadly defined as computational methods using experience to improve performance or make accurate predictions." (Mohri et al., 2018, p. 1). In essence, ML is just a set of mathematical matrix operations that, over time, will update and make stronger and stronger predictions. This section will detail three main components of this thesis. It will elaborate on how the performance of classification models is measured. It will detail the specific ML models and algorithms that were used for this thesis. Lastly, the choice of models and methods for analysing the results will be justified by concepts of explainable artificial intelligence (XAI).

2.3.1 Classification Performance Metrics

A key aspect of ML is measuring the degree of correctness in the model's predictions. In machine learning, it is common to split the dataset into two parts, one for training and one for testing. After training a model, it can be used to predict a new unseen set. This is the test dataset. The predictions is then compared with the actual labels. Instances where the predicted label is equal to the actual label are marked as true positive (TP) or true negative (TN) depending on the value of the label. Instances where the predicted label is true and the actual label is false are marked as false positive (FP). Instances where the predicted label is false while the actual label is true are marked as false negative (FN). The differences can be seen in Table 1. In classification theory, there are a multitude of different performance measures. This section will introduce the most relevant.

Predicted	Actual	Marking
1	1	TP
0	0	TN
1	0	FP
0	1	FN

Table 1: Naming convention for classification.

Accuracy measures how many instances that were correctly classified over the total number of instances (see Equation 1). Although the accuracy describes how often the model is right, a high accuracy can be misleading. In the case of binary classification, an unbalanced distribution (i.e., one label has well over half of the representation) of the target variables can lead to a high accuracy even if the model always gives the same output.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision measures the percentage of classified positive instances that actually were positive instances (see Equation 2). If the precision is 0.7, then 30% of the instances classified as true were actually false. In use cases where FP is more important to reduce than FN, precision is the preferred performance metric to use.

$$precision = \frac{TP}{TP + FP} \tag{2}$$

Recall measures the percentage of correctly classified positive instances (see Equation 3). If the recall is 0.7, then 30% of the positive instances were not correctly classified. In the use cases where FN is more important to reduce than FP, recall is the preferred performance metric to use.

$$recall = \frac{TP}{TP + FN} \tag{3}$$

For cases where both recall and precision are important, the F1 score is a common precision metric to use. The F1 score takes the harmonic mean of both precision and recall (see Equation 4). An advantage of using the F1 score as supposed to either recall or precision is that it will solve problems where either of them are high. For instance, a model that always predicts true will have a recall of 1, while the precision is low. Thus, the F1 score is a good metric to use to avoid misinterpreting seemingly good results.

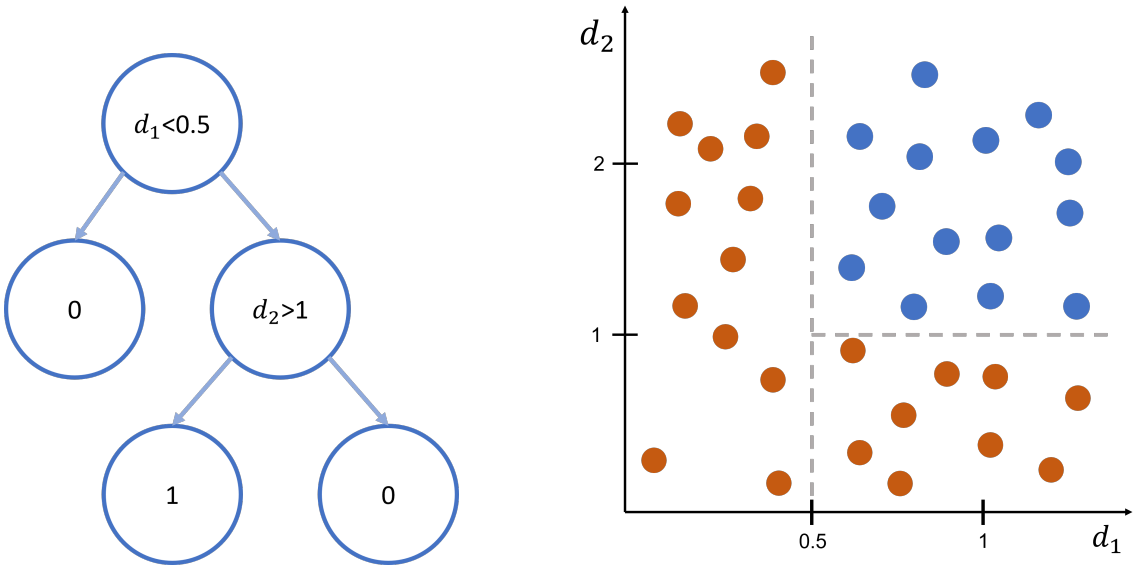
$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

2.3.2 Models and Algorithms Used

The ML models and algorithms used in this thesis are all tree-based classifiers. Firstly, tree-based models are models which utilise a tree structure made of nodes and edges, a directed cyclic graph, where the leaf nodes are the final predictions. Secondly, classifiers are models that predict a discrete label, in this case, delay or not delay. There are four models presented in this paper; Decision Tree, Random Forest, AdaBoost, and Gradient Boosting. The argument for choosing tree-based classifiers will be further elaborated in Section 2.3.3. All models can be used for regression analysis as well, but the focus for this thesis will lie on the classification counterpart.

Decision Trees

The decision tree algorithm is a divide-and-conquer approach that has been widely used for over four decades (Myles et al., 2004). A great advantage of this approach is the intelligibility of the model. A decision tree consists of nodes, where each node is a feature test called *split*. The data received by the node will be split into different subsets depending on their values (Zhou, 2012). The leaf nodes represent a label. Thus, all instances falling into this leaf node will be classified as that label. A simple example of a decision tree can be seen in Figure 4. This illustration has the features d_i where $i = 1, 2$. All instances are received by the first node; those instances with $d_1 < 0.5$ will be labelled 0, and the other will be tested with their d_2 value. Instances with $d_2 > 1$ will be labelled 1 and the rest will be labelled 0.



(a) Internal structure

(b) Classification

Figure 4: Illustration of a simplified decision tree with two features and pure leaf nodes.

The way decision trees learn is by recursively testing different features and splits. Not all leaf nodes will be pure like in Figure 4, instead the leaf nodes may have some instances of each label. The two most common ways to evaluate partitions are with the *information gain* or the *gini index* (Myles et al., 2004). Information gain uses the entropy of the feature subset. The entropy of the training data D is defined as

$$Entropy(D) = -\sum_{\forall y \in Y} P(y|D) log(P(y|D)) \tag{5}$$

where $P(y|D)$ denotes the probability of the class y given the dataset. The information gain is calculated as the reduction in entropy when dividing the training set into smaller subsets $D_1, ..., D_k$

$$InformationGain(D; D_1, ..., D_k) = Entropy(D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} Entropy(D_i) \tag{6}$$

The feature value pairs with the highest information gain are chosen for the split. When using the Gini index, the goal is to reduce class impurity by partitioning the feature space. The Gini index indicates the probability of misclassifying new data. The impurity of the training data D is defined as

$$Impurity(D) = 1 - \sum_{y \in Y} P(y|D)^2 \tag{7}$$

And the Gini index is calculated on the training subsets as

$$Gini(D; D_1, ..., D_k) = Impurity(D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} Impurity(D_i) \tag{8}$$

The equations are inspired by Zhou (2012) with minor notation changes. Decision trees are, however, prone to overfitting. A decision tree that fits perfectly to the training data will be worse at generalising than a decision tree with a poorer fit to the training data (Zhou, 2012).

Random Forest

Ensemble methods are techniques that combine several models. The resulting model has better predictive performance than any of the models it is built upon (base models). Bagging and Boosting are two of the most common ensemble methods. Breiman (1996) formalised bagging, or bootstrap aggregating. The original idea was to split the training set D into n several bootstrapped samples without replacement. Each bootstrapped sample, consisting of a unique set of features from D, is then used to grow a decision tree with cross-validation. Each instance in the test set is then classified by all decision trees, and their predictions are then aggregated to give a final prediction.

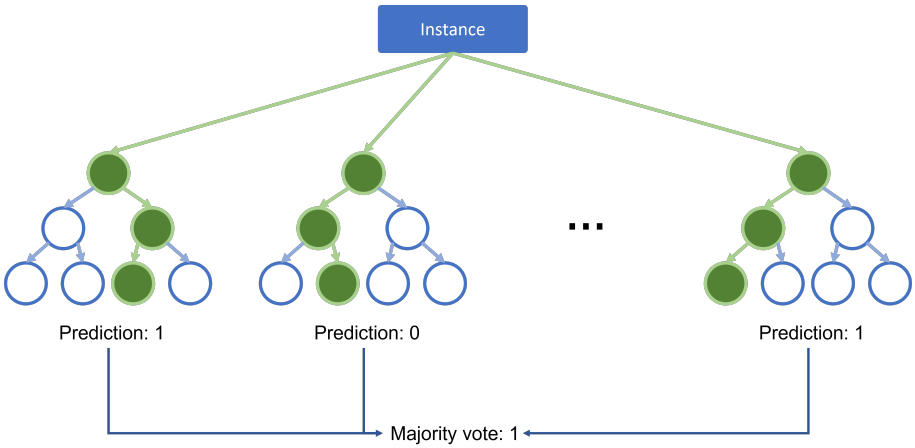


Figure 5: Illustration of prediction by Random Forest of decision trees of depth 3.

The random forest is a further development of bagging in which the bootstrapped samples are selected with replacement. Thus, the bootstrapped samples can have a random selection of the features, and one feature can be present more than once in a bootstrapped sample. Breiman (2001) argued that randomness was injected to reduce correlation and thus improve accuracy. The Random Forest algorithm creates a set of decision trees, each created with a unique subset of the dataset. When the Random Forest is trained, instances are predicted by the majority vote of all decision trees in the Random Forest. Figure 5 shows the Random Forest prediction process. While decision trees tend to be highly sensitive to training data, an ensemble of them reduces the sensitivity. Additionally, the random forest classifier cannot overfit due to the Law of Large Numbers (Breiman, 2001).

AdaBoost

AdaBoost (Adaptive Boosting) is another ensemble method, but one that uses boosting instead of bagging. The principle behind boosting is that it is easier to find many rough rules than a single highly accurate rule (Schapire, 2003). Thus, boosting combines a large set of weak learners, which are inaccurate, to produce highly accurate results. The AdaBoost model creates a set of decision trees with one parent node and two leaf nodes, also called a stump. A Decision Stump takes in one feature to predict the label. The core idea of boosting is to create new base models based on previous mistakes. In its essence, boosting is a technique designed to eliminate prior mistakes (Schapire, 2013). AdaBoost does this by iteratively updating the weight of each decision stump until all labels are classified correctly or the maximum number of iterations is reached.

Algorithm 1 AdaBoost algorithm. Adapted from (Schapire, 2013) and (Amini, 2015).

Input:

- A training set $S = (x_1, y_1), ..., (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$.
- The maximum number of iterations T.

1: Initialise weight distribution $\forall i \in \{1, ..., m\}, D_1(i) = \frac{1}{m}$.

2: **for** t = 1, ..., T **do**

3: Train weak learner $f_t : \mathbb{R}^d \to \{-1, +1\}$ using distribution D_t

4: Set $\epsilon_t = \sum_{i:f_t(x_i) \neq y_i} D_t(i)$ ▷ Weighted error

5: Choose $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$

6: Update the weight distribution over examples: $\forall i \in \{1, ..., m\}, D_{t+1}(i) = D_t(i) exp(-\alpha_t y_t h_t(\mathbf{x}_i))/Z_t$

7: **end for**

Output: The voted classifier $\forall \mathbf{x}, F(\mathbf{x}) = sign(\sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}))$

Algorithm 1 shows the AdaBoost algorithm. Initially, the number of iterations T and the weight distribution are set. For the first iterations, all instances from 1 to m are given equal weight, which will be updated each iteration. For each iteration, a set of weak learners is created. This is usually a set of Decision Stumps. Then the weighted error ϵ_t is calculated. For binary classification, all instances where weak learners miss will have an error of 1. All correctly predicted instances will have an error of 0. The weighted error ϵ_t is then the sum of all errors over all instances, times the weight of that instance. Then α_t is calculated from the weighted error ϵ_t. This is then used to update the weights for all weak classifiers. By updating the weights of each weak classifier for each iteration t, AdaBoost can learn from previous mistakes. The final output is the voted classifier. Figure 6 shows how AdaBoost learns from previous mistakes by updating the weights for each iteration t. This also shows how the final prediction is a vote of all iterations.

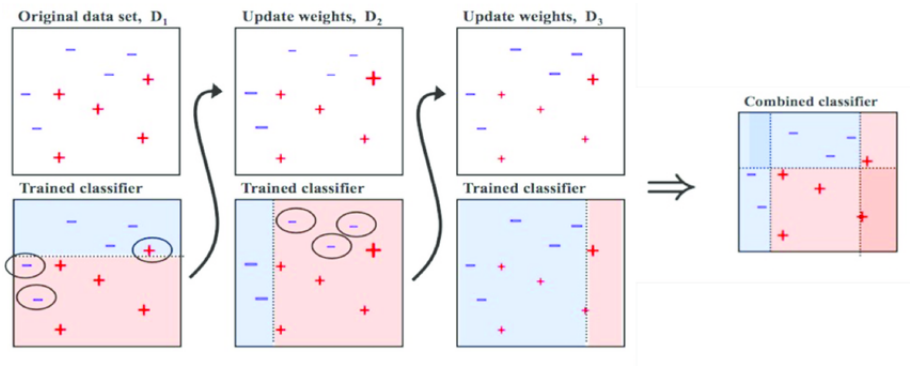


Figure 6: Illustrations of AdaBoost learning over three iterations ($t = 1, 2, 3$). (Alto, 2020).

Gradient Boosting

Gradient Boosting is another boosting ensemble which shares similarities with AdaBoost. Like AdaBoost, in Gradient Boosting the decision trees are fitted iteratively to correct the predictions of the prior models. However, the models are fitted using any differentiable loss function and the gradient descent optimisation algorithm. Here, the loss function is minimised as the ensemble is training. Additionally, the base learner can be larger than stumps, making it more computationally demanding.

Algorithm 2 Gradient Boosting algorithm. Adapted from (Natekin & Knoll, 2013)

Input:

- A training set $S = (x_1, y_1), ..., (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$.
- The maximum number of iterations T.
- Choice of the loss-function $L(y, f)$
- Choice of the base-learner model $h(x, \Theta)$

1: Initialise $\hat{f}_0$ with a constant

2: **for** t = 1, ..., T **do**

3: Compute the negative gradient $g_t(x)$

4: Fit a new base-learner function $h(x, \Theta_t)$ ▷ Θ_t is the incremental parameter estimate

5: Find the best gradient descent step-size ρ_t:

$$\rho_t = argmin_\rho \sum_{i=1}^{m} L[y_i, \hat{f}_{t-1}(x_i) + \rho h(x_i, \Theta_t]$$

6: Update the function estimate: $\hat{f}_t \leftarrow \hat{f}_{t-1} + \rho_t h(x, \Theta_t)$

7: **end for**

Output: The voted classifier $\forall \mathbf{x}, F(\mathbf{x}) = sign(\sum_{t=1}^{T} \hat{f}_t(\mathbf{x}))$

Like AdaBoost, the model is initialised with a set number of iterations. Additionally, the choice of loss function and base-learner is set. Like AdaBoost, the algorithm is trained for T iterations or until all instances are classified correctly. For each iteration, the new base-learner is trained with respect to the error of the whole ensemble. This is done by constructing base-learners to be maximally correlated with the gradient descent of the loss function (Natekin & Knoll, 2013).

2.3.3 Explainable Artificial Intelligence

As ML algorithms have improved in terms of predictive power and accuracy, they have also become increasingly more complex. Kamath and Liu (2021) argues that improvements in ML have led to a trade-off between improved quality and transparency. The phenomenon of reduced or even lack of transparency is known as the Black-Box problem, where for a given set of inputs, a Black-Box model will return outputs without the user or creator of the model being able to explain how. The field of XAI seeks to provide insight into how the models work, and how and why predictions are made (Kamath & Liu, 2021). The field of XAI has four goals closely related to the complexity of the model:

- **Understandability:** The AI model must be understandable to humans without the need to understand the internal algorithmic structure.
- **Comprehensibility:** The ability of an AI model to represent and convey its learnt knowledge in a human-understandable fashion.
- **Interpretability:** The ability to explain the model assumptions, i.e., explaining the structure of an AI model.
- **Transparency:** The degree to which the internal structure and algorithm of the AI model are understandable.

Of the algorithms presented in Section 2.3.2, the Decision Tree is the most interpretable algorithm. Furthermore, its simple structure with splitting nodes and criterion makes it more understandable, comprehensible, and transparent compared to the others. The added complexity of Random Tree, AdaBoost, and Gradient Boosting reduces the explainability of the algorithms. However, since tree-based models require less preprocessing, the models can easily be explained with methods for post hoc interpretability. Kamath and Liu (2021) proposed several methods for post-hoc interpretability (i.e., after the model has been fitted to the training data) of AI models. To address the low explainability of the ensemble method models, this thesis will utilise two methods for post-hoc interpretability; an instance level method with shapley additive explanations (SHAP) plots and a dataset-level method with permutation feature importance (PFI). Since both approaches are post hoc, they will be performed on the test set, and the model does not have to be trained again to perform the analysis. In addition to these two, Kamath and Liu (2021) has proposed several example-based methods, which will not be addressed in this thesis.

Shapley additive explanations analysis

When calculating the importance of a single feature value, the ordering of the value will have an impact (Kamath & Liu, 2021). The SHAP method reduces this variance by summarising the value attribution. This way, for a specific instance in the test set, one can see how each value of a feature affected the prediction (Biecek & Tomasz, 2020). Shapley values decompose the instance predictions into contributions for each feature value and are a great way to explore how different values affect the prediction. Positive Shapley values indicate that the specific value contributed to an increase in the prediction (i.e., a higher likelihood of being true for classification).

Permutation feature importance analysis

The idea behind permutation feature importance is that the importance of a feature is directly related to the increased error when the feature values are shuffled (Kamath & Liu, 2021). The

implemented analysis compares the F1 score of the model with one permuted feature (i.e., shuffled feature variables) and compares it to the original model. To reduce variance, the feature is permuted several times. The average decrease in F1 score for the model with a permuted feature will be the PFI for that feature. This analysis indicates how important a feature is for the model. If the PFI is zero, the feature has no impact on the prediction. Positive permutation importances indicates that the feature is important for the prediction. If a feature has a negative permutation importance the model loses predictive performance with that feature included, i.e., the model performs better on the permuted dataset than the original.

2.4 Summary

Section 2.1.3 highlights that the main effects of VOs are increase in project cost and duration. Thus, in respect to project scope and time management, controlling VOs is a key success factor as it directly influences two of the three main success criteria. In Section 2.2.2, recent research on scope change management was presented and showed that there is a variety of research on project delays. However, little research on the delay of individual activities is done. Furthermore, no research has been found studying the time delay of individual activities as a cause of VOs. Thus, there is a research gap.

There has been a significant development of AI in the last decades, and there are signs of AI and ML applications in project management. The development of AI in project management presented in Section 2.2.1 is present today, with several articles published on MLBPM solutions yearly. Furthermore, there has been an increased focus on XAI in recent years, where the explainability of the models used is more important than ever. This master's thesis will link the research gap on the time effect of VOs on individual activities with current developments on AI by predicting activity delays through individual activity features and VO features. This is believed to be an important field of study, as MLBPM solutions can provide substantial assistance to project managers in decision making and controlling. Applying the ML practises to project scope and time management can be a significant step towards more effective project management.

3 Methodology

The main goal of this master's thesis is to investigate whether VOs can contribute to predict activity delays. To investigate this research problem, a machine learning approach was applied. This section will explain in detail all steps taken from raw data to predicted delays. To properly address RQ1, this section will focus heavily on the transformation from raw data to merged and processed data that can be used for an ML analysis. The first part of this section details how the literature search was conducted. Next, this section includes data gathering, exploratory data analysis, data cleaning, preprocessing, model development, and training. Additionally, this section will elaborate on how the results obtained from the machine learning models were analysed and evaluated. Lastly, the quality of the method will be assessed.

The method presented in this master's thesis was created using the programming language *Python*. Python supports a wide range of libraries that offers predefined functions to streamline the processes. The libraries used in this master's thesis include *Pandas* for processing, cleaning and exploring the dataset, *Scikit-learn* for the creation of ML models, preprocessing and splitting of the dataset, and *Matplotlib* and *Seaborn* for data visualisation.

3.1 Literature Search

The theoretical background in Section 2 laid the foundation for the method and analyses in this master's thesis. The literature found has been crucial to gain insight into how VOs have been studied. Additionally, the literature has provided a sound theoretical background. The majority of the literature presented was found through Google Scholar. As Section 2 is divided into three major categories, a wide specter of search terms have been used. Searches include main terms such as "variation orders", "construction project management", "time delay", "project time management", and "project scope management". In addition, search terms for specific methods used in this paper were searched directly. The search engine has then redirected to publishers including Emerald Insight, CiteSeerX, and Taylor & Francis Group. Using the snowball method, where referenced articles are tracked, a varied selection of articles was found. Articles with a significant number of citations were prioritised. In addition to the snowball method, *connectedpapers.com* was used to find related articles by building a networked graph of citations from articles.

In addition to published research, reports and articles from consulting firms and project management organisations were used to provide additional insight into trends and statistics. Since little of the presented research has come from the last five years, these sources were a good addition to

find statistics and trends.

3.2 Data Extraction

The data used in this thesis come from a large service provider in the energy sector. It contains one project that spans several years and has several contributors. All data are stored in a third-party project planning software and had to be extracted with SQL queries. All data are extracted raw from the tables, i.e., `SELECT * FROM <table_name>`.

All project data from the company are divided into a multitude of different tables. There are 192 different tables in total with different use cases. The tables are used to present specific data in the project management software's user interface. Although they exist in the database, the project manager does not interact directly with the tables. Some tables are used for configurations like custom user fields and data plotting, some are used for forecasting and reporting, while others store project-specific data. Most of the project data is stored either by baseline (biannual updates), by weekly status updates, or as unique data points. After a detailed analysis of the available tables and interviews with the company, four tables were selected to include enough information about all activities. The selected tables are `activities`, `resources`, `period_status_a` and `vo_reg`. Three of which store instances as unique data points, while instances in `period_status_a` are stored by weekly period status updates.

The `activities` table stores information on all activities in the project, but some features in the table are overwritten as the project progresses. To obtain the initial features or values on specific dates, the table `period_status_a` can be used. These tables store much of the same information as `activities`, but store the information for each week of the project. Together, these two tables will give a proper overview of all the activities of the project. The `resources` tables store all the resources used in the project. Every resource is linked to activities where one activity can be linked to one or more resources. Furthermore, resources can be linked to a specific VO if they are included in a VO. Lastly, the `vo_reg` table is a record of all VOs, both requests and accepted, and contains information about which resources are affected by the VO. All tables will be further elaborated in the next section by an exploratory data analysis.

3.3 Exploratory Data Analysis

3.3.1 Activities

The `activities` contains 27764 unique instances with 308 features. Each project activity is represented in the table `activities`. These features are of varying types and include flags, dates, categorical features, and text. Of the 308 features, 200 of them are defined by the user, i.e., the project owner. These features allow the project owner to customise activity features. Table 2 shows the distribution of the user defined features. Even though there exist 200 user defined features, only 113 is defined for this project. These 113 can be filled in by the project owner. The other 87 features are not defined and thus not used.

Feature name	Description	# features	# active features
r[1-60]	Reference	60	30
d[1-40]	Date	40	20
l[1-40]	Flag	40	20
n[1-20]	Decimal	20	20
u[1-10]	Duration	10	5
o[1-30]	Outline codes	30	18

Table 2: User defined features.

An important aspect of project management is to track progression. The software has several date fields that record start and finish times. The feature `ES` (early start) is used to track when an activity should start. Since the `activities` table is overwritten as the project progresses, some stored date values are changed several times during the course of the project. Due to this, the `ES` feature has to be extracted from the `period_status_a` table where it is possible to obtain the first planned `ES`. `ACS` (actual start) is logged when an activity starts and will equal the last time the activity was set to have begun. Thus, this feature can be extracted from the `activities` table. For the finish time of an activity, `EF` (early finish) is like `ES` stored in both tables, where `period_status_a` stores the original `EF`. Like `ACS`, `ACF` (actual finish) is stored in the `activities` table. After interviews with the company, they highlighted `d13` as an important feature in the `activities` table. This is defined as the activity transfer date, which is the date when the activity was either created in the software or imported from another software.

Not all 308 features in this table can be used for an analysis. After interviews with the company, a limited set of features from the `activities` table was agreed to give a good overview of the activities.

3.3.2 Resources

The project contains 48998 individual resources. All resources are described by 35 features. Of these features, five features do not have any values, seven features have constant values and 13 features have more than 48.8% missing values. Excluding these features leaves ten features that are of value. The remaining ten features include two flags, two features regarding the progress of the resource, two features regarding the progress and four features which are unique IDs.

3.3.3 Variation Orders

In the table `vo_reg` there are 123 VOs stored. Of these, 9 are variation order requests leaving 114 VOs. Several of the variation orders are used at the end of each baseline for budgeting. There are 58 change orders of this type. Excluding both types leaves 56 change orders in total.

One interesting thing to look at is the distribution of change orders over time. Figure 7 shows that most of the distribution of VOs are front loaded. This figure shows the distribution for the remaining 56 change orders. 40 of the VOs were issued in 2015, 14 in 2016 and only two in 2017.

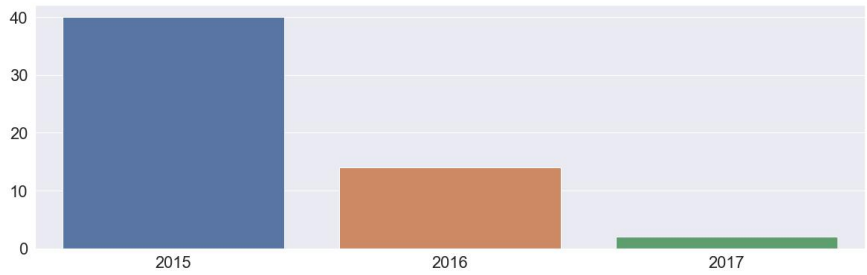


Figure 7: Distribution of issue dates for variation orders.

3.4 Data Cleaning

In order to use the extracted data for machine learning, the tables had to be joined and features had to be carefully selected. As Section 3.3 showed, many of the features present are currently not used by the company, many features are of a singular value and many features have a significant portion of missing values. In order to extract the most useful information out of the tables it is important to clean the data. Doing this before merging the tables ensures that the features present from each table are of importance for the model performance and predictive quality.

3.4.1 Activities

As Section 3.3.1 displayed, there are several features that needs to be removed from the activities table. The first part of data cleaning for this table was to remove individual activities that could not be included. This included removing cancelled activities and where the description is empty or contains the string "DUMMY". Additionally, activities that was not connected to a calendar had to be removed. The `activities` table has a feature named `on_target`. This is a flag that is set to true for all activities that will finish on time. The activities set to have `on_target` true are usually activities where the progress cannot be tracked. Thus, they are set to finish on time and was removed. Milestones for the project is also stored as activities, these had to be removed as well. Lastly, all activities that had a duration of zero was removed. The process of removing single activities reduces the `activities` table from (27764, 308) to (12661, 307). The single column that was removed was the `cancelled` column.

A prerequisite for machine learning algorithms is that all data points contain actual values, and not not a number (NaN) values. There are several solutions to deal with features that have a large percentage of NaN values. They can be set a specific value, most commonly zero or the mean of the feature. However, doing this on features that has a large percentage of NaN values can greatly affect the pureness of the data. It was chosen to remove all features that had at least 30% of NaN values present. Additionally, several features had a constant value. These features will not add any value to the models and were removed. The process of removing feature with a high NaN percentage and single value features reduced the number of features from 307 to 111.

As mentioned in Section 3.3.1, the `ES` feature of the `activities` table is overwritten for each weekly status update. To get the first written instance of `ES`, the table `period_status_a` was used. Since `ES` was one of the features removed by the NaN cut off, the number of features was increased from 111 to 112.

Finally, the remaining features had to be selected. From the exploratory data analysis (EDA) and interviews with the company, it was agreed to keep only a limited set of features that were believed to contribute the greatest to the analysis. The final size of the `activities` table were (12624, 34) and included five date features, 12 categorical features, 14 flags, a description, the duration and the unique ID of the activity. The final version of the `activities` table can be seen in Table 3.

Feature	Description	Type
seq	Unique ID	int
du	Activity duration	float
wpn	Calendar	Category
subnet_id	Subproject ID	Category
r (x10)	Reference fields	Category
l (x14)	Flags	Flag
es	Early start	Date
ef	Early finish	Date
acs	Actual start	Date
acf	Actual finish	Date
d13	Activity transfer date	Date
des	Activity description	Text

Table 3: Remaining features of `activities` after data cleaning. Full version can be seen in Table A.1.

3.4.2 Resources

The original `resources` table was of size (48998, 35). After removing columns with a `NaN` percentage of 30% or more, and removing features with constant value the size was reduced to (48998, 10). Of the remaining features, only four were kept; the activity ID, the VO ID and the quantity of the resource. The remaining features can be seen in Table 4.

Feature	Description	Type
an_seq	Unique ID of activity	int
vo_seq	Unique ID of variation order	int
qty	Quantity	float

Table 4: Remaining features of `resources` after data cleaning

3.4.3 Variation Orders

The original size of the `vo_reg` table was (123, 13). After removing columns with a `NaN` percentage of 30% or more, and removing features with constant value the size was reduced to (123, 9). As Section 3.3.3 explained, some of the instances in the table are either variation order requests or included for budgeting purposes. After removing these instances, the table was reduced to 56 instances. Finally, a smaller subset of the features were chosen, resulting in a table of size (56, 4) which can be seen in Table 5.

Feature	Description	Type
seq	Unique ID	int
description	Description of VO	Text
responsible	The responsible of the VO	Category
baseline_id	The baseline the VO was added to	Category

Table 5: Remaining features of `vo_reg` after data cleaning.

3.5 Merging the Tables

After the data was cleaned, the tables were merged on the IDs. Each instance in the merged table (hereafter *dataset*) was comprised of an activity-resource pair, i.e., one activity connected to four resources would be four distinct instances in the dataset. If the resource was connected to a variation order, the features from the table `vo_reg` would also be complete. Since the activities are of interest, the dataset was grouped on each activity and aggregated on the different features. This resulted in a table with the same number of instances as the cleaned `activities` tables. Features from the `resources` and `vo_reg` would not always match within the same activity, thus, these features were aggregated on different rules. The feature `qty` was summed, i.e. an activity with four resources would have the sum of the qty as a single feature. Categorical features such as `baseline_id` and `responsible` were aggregated to include all instances. Lastly, the description from each VO was appended to each other and the number of VOs for each activity was counted.

3.6 Preprocessing

Due to the complexity of merging and grouping the tables, some preprocessing was done before grouping the tables while the rest was done after. However, all preprocessing will be discussed this section.

To assert whether the activity were delayed, two target variables were created. If the actual start of an activity was after the early start, the activity was delayed. Likewise with finish, if the actual finish was after the early finish, the activity was delayed. The target values were created as two new features to the dataset, and `ACS` and `ACF` was removed from the dataset to avoid information leakage. The rule for creating the features can be seen in Equation 9 and 10.

$$DELAYED\_START := ES \leq ACS \tag{9}$$

$$DELAYED\_FINISH := EF \leq ACF \tag{10}$$

After the removal of `ACS` and `ACF`, three datetime features remained; `d13`, `ES` and `EF`. Datetime features contain a significant amount of information, but are not readable for a ML algorithm. The three datetime features were converted into separate features, `year`, `month`, `week`, `day` and `dayofweek`.

In order to focus on XAI, this thesis has chosen to implement tree-based ML models. One major advantage of these models is that they can interpret categorical features without the need of encoding. Thus, the categorical features were kept as is. The dataset contained two textual features `des` from `activities` and `description` from `vo_reg`. Neither can be interpreted raw, thus preprocessing had to be applied to these columns. First, they were vectorised where the top ten appearing words were selected. Stop words were removed to improve the predictive performance. Applying this process removed each textual feature and replaced it with ten features for the most common words and whether that is appearing for the activity. Lastly, all `NaN` values were replaced with a zero, the scale features was normalized and all IDs were removed. The final dataset was of size (11194, 72) and the features can be seen in Table 6.

Feature	Description	Type
du	Activity duration	float
wpn	Calendar	Category
subnet_id	Subproject ID	Category
r (x9)	Reference fields	Category
l (x14)	Flags	Flag
baseline_id_x (x2)	Corresponding baseline the VO was added to	Encoded
responsible_x (x4)	Responsible person of the VO	Encoded
qty	Quantity of the resources	Float
(es/ef/d13)_year	(es/ef/d13) year	Category
(es/ef/d13)_month	(es/ef/d13) month	Category
(es/ef/d13)_week	(es/ef/d13) week	Category
(es/ef/d13)_day	(es/ef/d13) day of month	Category
(es/ef/d13)_dayofweek	(es/ef/d13) day of week	Category
des_x (x10)	Activity description	Text
description_x (x10)	VO description	Text
DELAYED_START	Target for delayed start	Target
DELAYED_FINISH	Target for delayed finish	Target

Table 6: Overview of features in the final dataset.

3.7 Model Training

The preprocessed dataset consisted of 11194 instances with 72 features, where two features were the created targets. Initially, the dataset was split 80% into training and 20% into testing. To create reproducable models, the splitting of the dataset is done with a fixed seed so that the

seperation of instances is equal every time. In order to create robust models with high predictive performance, a grid search was performed over all models for a given set of hyperparameters (see table Table 7). The grid search approach is time consuming, but ensures that a wide range of models and hyperparameters was tested. To ensure that the models did not overfit (i.e. memorize the targets), only the training set was used for the grid search. Figure 8 shows the procedure for splitting the dataset.

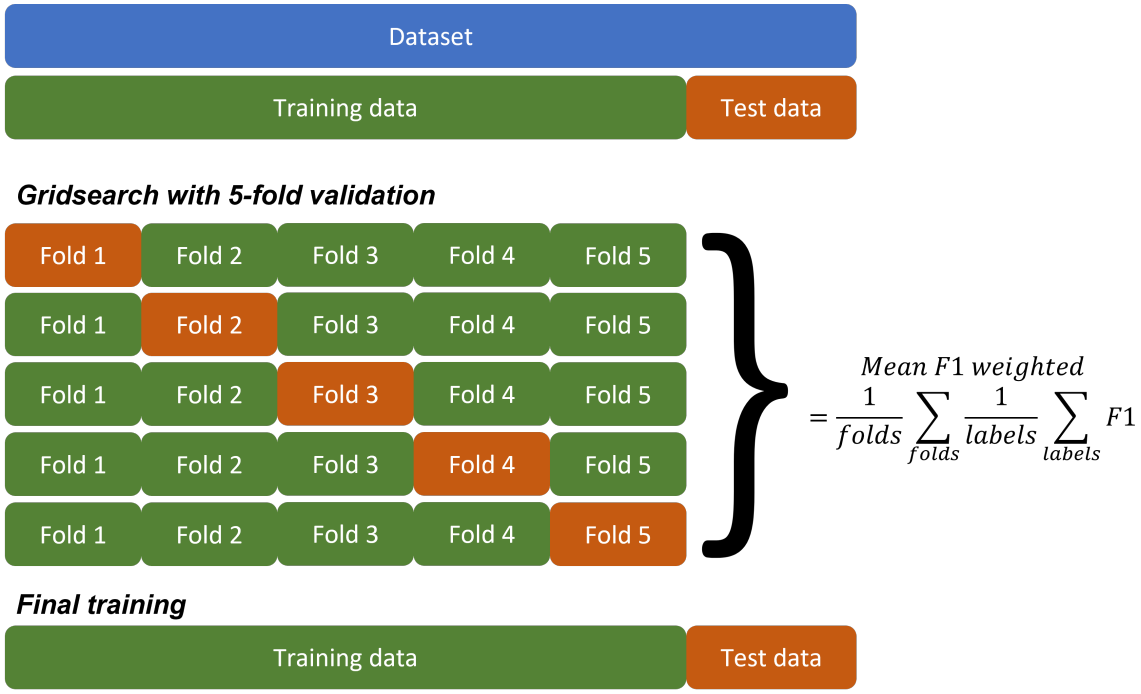


Figure 8: The process for training each model visualised. The grid search is performed once for each hyperparameter configuration.

For the main analysis, the top performing model from each class was selected by their mean weighted F1 score. This is the mean F1 score over the 5 folds, weighted over both labels. The best performing model from each class was then trained on the training set and tested on the testing set.

Classifier	Hyperparameter	Tested values
AdaBoost	n estimators	64, 128, 512
	learning rate	0.2, 0.5, 0.8, 1.0
	algorithm	SAMME, SAMME.R
Decision Tree	criterion	gini, entropy
	splitter	best, random
Gradient Boosting	loss function	deviance, exponential
	n estimators	64, 128, 512
	learning rate	0.2, 0.5, 0.8, 1.0
Random Forest	n estimators	64, 128, 512
	criterion	gini, entropy

Table 7: Models and hyperparameters selected for grid search.

3.8 Analysis of Predictions

With the target values `DELAYED_START` and `DELAYED_FINISH`, a TP and TN prediction indicates that a delay or not was predicted correctly. A FP prediction indicate that an activity that was on time was classified as being delayed, and a FN prediction indicates that a delayed activity was classified as being on time. In light of the research goal, a FN prediction is of higher concern than a FP prediction. If activities that is going to be delayed is not discovered, they can affect the project greatly. However, activities that will be on time but is marked as delayed (FP) will only lead to controlling an activity more than needed. To reduce the number of FN predictions the performance metric recall is of highest concern. Additionally, all models were analysed and validated on the performance metrics accuracy, precision and F1 score. In addition to the performance metrics, a confusion matrix plot was included to give insight into what type of errors the model does.

To quantify RQ3, both the SHAP and PFI analyses were performed on the test set. Both analyses can quantify to what degree the features were important in the prediction. By including both, one can quantify the attributions on both the instance level and the dataset level. Additionally, the results of these analyses will be used to quantify RQ2, whether or not VOs are important in predicting delays. Both analyses will increase the explainability of the models, and are important tools in understanding the results.

3.9 Assessing the Quality of the Method

3.9.1 Reliability

The reliability of the method is important to evaluate. Fellows and Liu (2003) argues that reliability concerns the consistency of a measure. To address the reliability of the method, this section will focus on the consistency and reproducability of the method. In other terms, will the same procedures described in the method yield the same dataset, models and results? Due to the extent to which the method has been described, it is considered reproducible. However, some parts of the method, like Section 3.5 where the tables are merged, are not described in full detail. Thus, writing code with a slightly different approach may yield different results. Even though the method is reproducible for this project, other project will likely yield different results. As the dataset is split into train and test data with a seed, the final datasets used in the models are also reproducable.

Tjora (2021) argues that a particularly vulnerable aspect of qualitative studies is the selection and presentation of citations from interviews or the selection of observations. This aspect is also applicable for quantitative research and especially for a ML-based method. The method presented selects a limited number of features (observations) from the original tables. Different subsets of the feature space may change the meaning of the variables. The method has detailed the processing of the data and has given a complete overview of the final features in Table 6. Eventhough the final dataset is reproducable, a different selection of features might yield different results.

3.9.2 Validity

Fellows and Liu (2003, p. 157) states that "validity concerns how well a measure does measure the concept it is supposed to measure." To evaluate the validity of the method, the dataset created is of highest concern. The dataset was sourced from four different datasets. Each of these are populated and moderated by employees of the company. Section 3.4 showed that some tables had dummy instances or instances that were not used. The procedures detailed in that section removed all instances of this type, however, as the final dataset contains 11194 instances, the data was not controlled manually. This reduces the validity of the final dataset as some instances that should not be included may be included. However, the process in Section 3.4 captured and removed all known instances, so the remaining amount of faulty instances is believed to be low. The features of the final dataset, and the removals detailed in Section 3.4 was found by EDA and interviews with the company. However, as the dataset was constructed by a single person without ownership of the data, some faults may be present.

Another aspect of the dataset important to evaluate is the definition of delay. For this thesis, a delayed activity was defined as whether or not the actual start or finish was before or on the date of the early start or finish. There is, however, different ways of defining delay. A delayed activity can be defined in relation to the LS and LF. Activities that started within the LS and LF will still be in bounds of the schedule. Additionally, activities that are delayed within the same week may not be of a big concern. In these cases, the delay can be defined by checking the actual start in relation to the week of the ES and EF.

4 Results

The results of the method described in Section 3 are intended to answer the last two research questions. This section will highlight the results from the grid search and what hyperparameter tuning led to the best predictors. It will also highlight how each model performed on both labels. The last part will address the importance of each feature, both by analysing the importance over the dataset with PFI and on single instances with SHAP.

4.1 Grid Search

In order to tune the hyperparameters of each model, a grid search was implemented. As mentioned in Section 3.7, the grid search was performed on the training set (i.e., 80% of all instances) and the top performing model of each class was selected from the weighted F1 score. This is the mean of the F1 score on both labels. The hyperparameters that produced the highest weighted F1 score for each model can be seen in Table 8. All ensemble method models produced better results when the number of estimators was increased.

Hyperparameter	Classifier			
	Decision Tree	Random Forest	AdaBoost	GradientBoost
algorithm	-	-	SAMME.R	-
criterion	entropy	entropy	-	-
splitter	best	-	-	-
learning rate	-	-	0.5	0.2
loss	-	-	-	exponential
n estimators	-	512	512	512

Table 8: Final hyperparameters selected for each classifier based on mean f1 score.

Table 9 shows the mean weighted F1 score of the top models and the standard deviation. This shows that Random Forest is the top performing model with a mean weighted F1 score of 0.902. The difference in performance between the best and worst predictor, Decision Tree, is 0.035.

Classifier	Mean weighted F1 score	SD
AdaBoost	0.886	0.006
Decision Tree	0.867	0.004
Gradient Boosting	0.898	0.005
Random Forest	**0.902**	0.004

Table 9: F1 score for top performing model of each class. Note that this is on 5-fold cross validation on the training set (80% of the dataset). The f1 scores are weighted on both labels.

4.2 Model Results

After the top performing model from each class was selected, they were trained again. This training was done on the training data. The scores for each model can be seen in Table 10, where the highest score for each metric is written in bold. The Random Forest model outperformed all other models in predicting the delay in activity start. However, in predicting delayed finish, AdaBoost outperformed Random Forest in recall, while Decision Tree performed the best in precision. The low difference in F1 score on `DELAYED_FINISH` between the models indicates that all models perform quite similarly on this label.

Classifier	`DELAYED_START`				`DELAYED_FINISH`			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
AB	0.832	0.868	0.907	0.887	0.825	0.856	**0.924**	0.889
DT	0.817	0.875	0.875	0.875	0.819	**0.889**	0.869	0.879
GB	0.858	0.891	0.917	0.904	0.839	0.873	0.920	0.896
RF	**0.870**	**0.899**	**0.927**	**0.913**	**0.841**	0.877	0.918	**0.897**

Table 10: Final performance metrics on the dataset for both labels.

Figure 9 shows the confusion matrices for the Random Forest model. Overall, this was the best model. Although the model was outperformed by AdaBoost in recall on `DELAYED_START` the difference is quite small at 0.006. The figure shows that the model failed to predict 120 activity starts and 139 activity finishes as delayed. These are marked as FN in the confusion matrix. 170 FP for `DELAYED_START` indicates that 170 activities were classified as delayed start when, in fact, they were on time. Similarly for `DELAYED_FINISH`, the model classified 217 activities that finished on time as delayed.

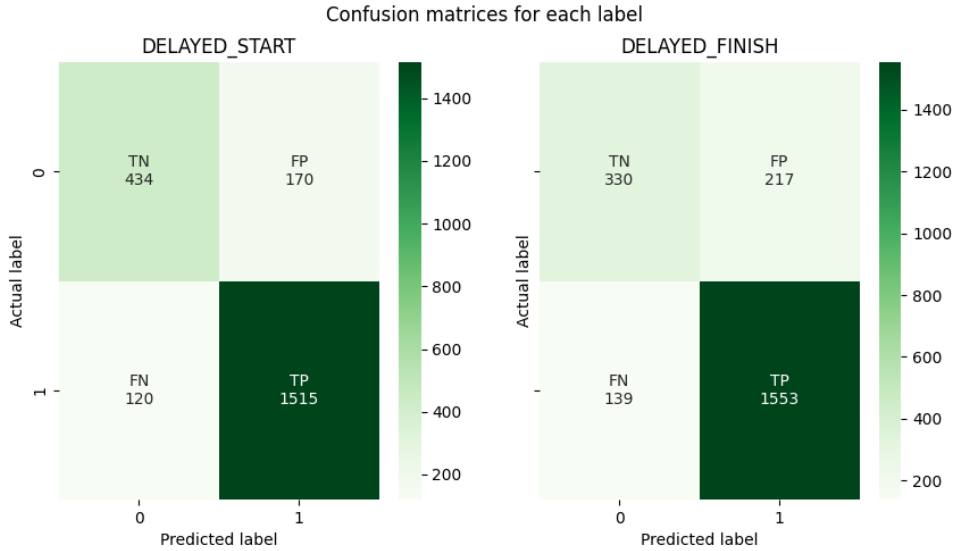


Figure 9: Confusion matrices for `RandomForest`.

Another important aspect of Figure 9 is that 73% of the activities in the test dataset had a delayed start, and 76% a delayed finish. This makes the test dataset somewhat imbalanced. If any model only predicts true, then they would have an accuracy of 73% and 76% respectively. This shows that recall is a better metric to use, as it limits the number of delayed activities classified as on time.

4.3 Permutation Feature Importances

The results of the PFI analysis are key to answering RQ2 and RQ3. Figure 10, 11, 12 and 13 show the top ten features of each model. The importance is measured in terms of loss to the F1 score when that feature is permuted.

For each model, the date features had a high occurrence in the top ten. In predicting `DELAYED_START`, all models assigned high importance to the ES features. Figure 10 shows that four of the ES features, `es_year`, `es_day`, `es_week` and `es_month`, had the highest PFI for the AdaBoost model. Furthermore, these features were significantly more important than the rest. Except for the Decision Tree model, the ES features were significantly more dominant than other features. Figure 11 shows that the Decision Tree model relies on more features than just the ES features.

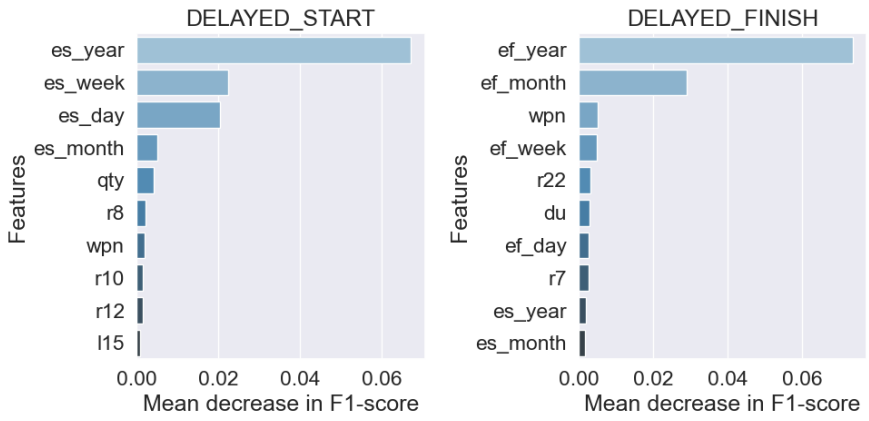


Figure 10: PFI for AdaBoost.

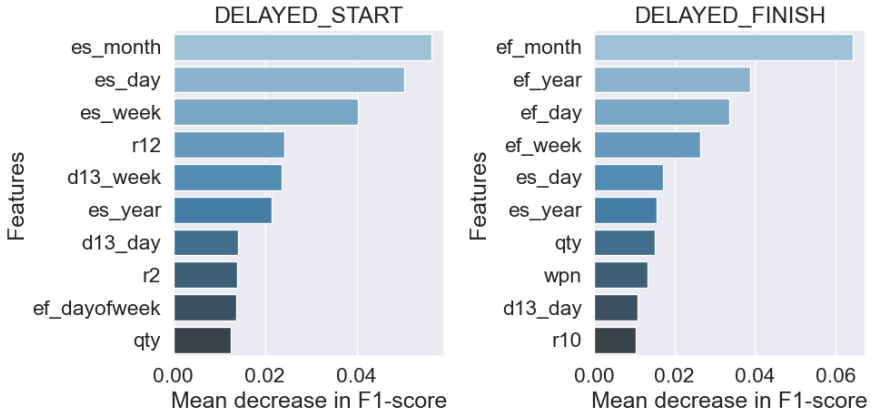


Figure 11: PFI for DecisionTree.

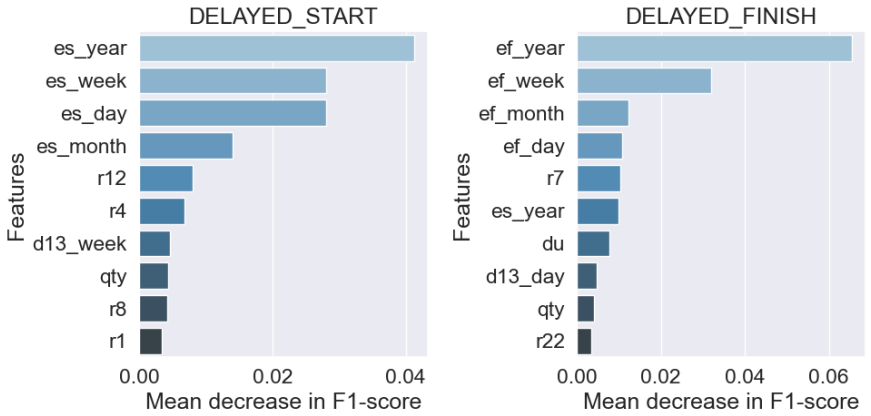


Figure 12: PFI for GradientBoosting.

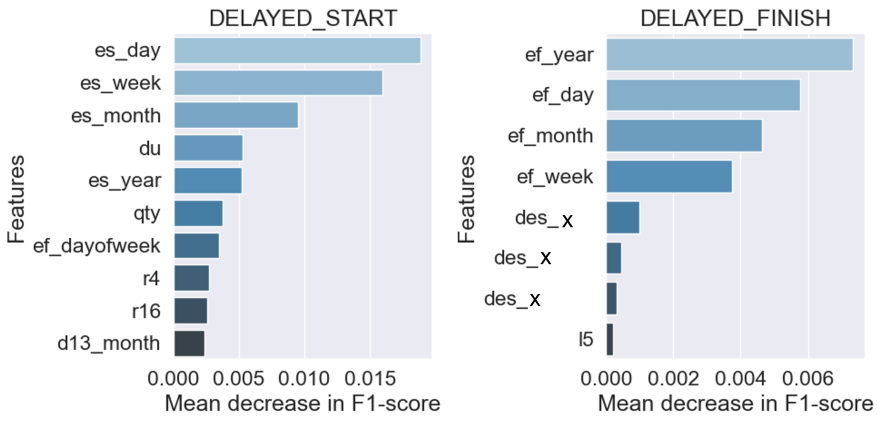


Figure 13: PFI for RandomForest. Note that the descriptive features `des_x` are anonymised.

In addition to the ES and EF features, the `d13` feature appeared frequently for both labels. Recall that this feature is the date when the activity was created in the software or imported from another

software.

In addition to the date features, the categorical features are frequently represented in the top 10 PFI plots. The categorical features that appeared in the top ten plots are shown in Table 11.

Feature	Description	DELAYED_START	DELAYED_FINISH
R2	Discipline	✓	
R3	WBS		✓
R4	Sub discipline	✓	✓
R7	Organisation		✓
R8	Area		✓
R10	Section		✓
R12	Category	✓	✓
R16	Building block	✓	✓
R22	WBS2	✓	✓
WPN	Calendar	✓	✓
SUBNET_ID	Sub project	✓	✓

Table 11: Overview of categorical features that made an appearance in the top ten PFI over all models for both labels.

When predicting `DELAYED_FINISH`, the feature `ef_year` is the most dominant. Table 12 shows that this feature is included in the top ten most important for each model. Additionally, in all models except Decision Tree, this feature scores the highest in PFI. With a mean of 0.0463, this feature lead to almost a double decrease in the F1 score compared to the second most important feature, `ef_month`.

Feature	DELAYED_START		
	n	Mean	SD
es_year	4	0.0338	0.0267
es_day	4	0.0294	0.0145
es_week	4	0.0267	0.0104
es_month	4	0.0212	0.0237
r12	2	0.0088	0.0108
d13_week	2	0.0076	0.0109
qty	3	0.0062	0.0043
r4	3	0.0053	0.0057
r2	2	0.0042	0.0066
d13_day	1	0.0039	0.0069

Feature	DELAYED_FINISH		
	n	Mean	SD
ef_year	4	0.0463	0.0299
ef_month	4	0.0275	0.0265
ef_week	4	0.0167	0.0145
ef_day	4	0.0132	0.0140
es_year	1	0.0066	0.0075
wpn	2	0.0048	0.0063
du	2	0.0042	0.0047
qty	0	0.0038	0.0081
es_day	0	0.0036	0.0091
r7	1	0.0030	0.0055

Table 12: Top 10 features for each label based on the mean Permutation Feature Importance. n is the number of occurrences in top 10.

4.3.1 PFI of the VO Features

To answer RQ2, the PFI scores of the VO features must be investigated. None of the features in the `vo_reg` table was included as the top 10 features based on PFI. The analysis showed that only three VO features had a mean PFI score different from zero for `DELAYED_START`. These features are shown in Table 13. This table shows that all three features had a slight importance for the Gradient Boosting and Random Forest model, and that the `responsible_2` feature had a slight importance for all models. However, these PFI of these features are insignificant compared to the top 10 features given in Table 12.

Feature	AB	DT	GB	RF	Mean
n_vos	0	0	-6.6×10^{-5}	1.7×10^{-5}	3.3×10^{-5}
baseline_id_1	0	0	-6.6×10^{-5}	1.7×10^{-5}	3.3×10^{-5}
responsible_2	-6.7×10^{-5}	5.3×10^{-5}	3.9×10^{-8}	-3.0×10^{-6}	4.9×10^{-5}

Table 13: PFI for `DELAYED_START` on all models of the three features from `vo_reg` that had a mean different from zero.

In predicting the label `DELAYED_FINISH` only two features had a mean PFI different from zero. These two features are shown in Table 14. Both features, `description_10` and `baseline_id_1`, had a PFI score different from zero for only one model each. The PFI analysis for `DELAYED_START` shows that the VO features are insignificant compared to the top features in Table 12. Although the top importances of `DELAYED_START` in Table 12 are slightly lower than those of `DELAYED_FINISH`, the PFI of the VO features in Table 14 are less than a hundredth of those in Table 12.

Feature	AB	DT	GB	RF	Mean
description_10	0	-3.4×10^{-4}	0	0	-8.4×10^{-5}
baseline_id_1	0	0	3.2×10^{-4}	0	8.0×10^{-5}

Table 14: PFI for `DELAYED_FINISH` on all models of the three features from `vo_reg` that had a mean different from zero.

4.4 SHAP Analysis

Figure 14 and 15 shows the Shapley value for the Random Forest model on each label. Figure 14 shows that the four features with the highest scores from the PFI analysis, are also the features with the highest Shapley value. For this particular instance, the ES of the activity was on the 5th of November 2016. The day of the month, week, and month all contributed significantly to reduce the chance of this activity having a delayed start. Furthermore, since the ES of the activity was in 2016, the chance of having a delayed start increased. One thing to note also is that several categorical features, like shown in Figure 13, have an impact on the prediction.

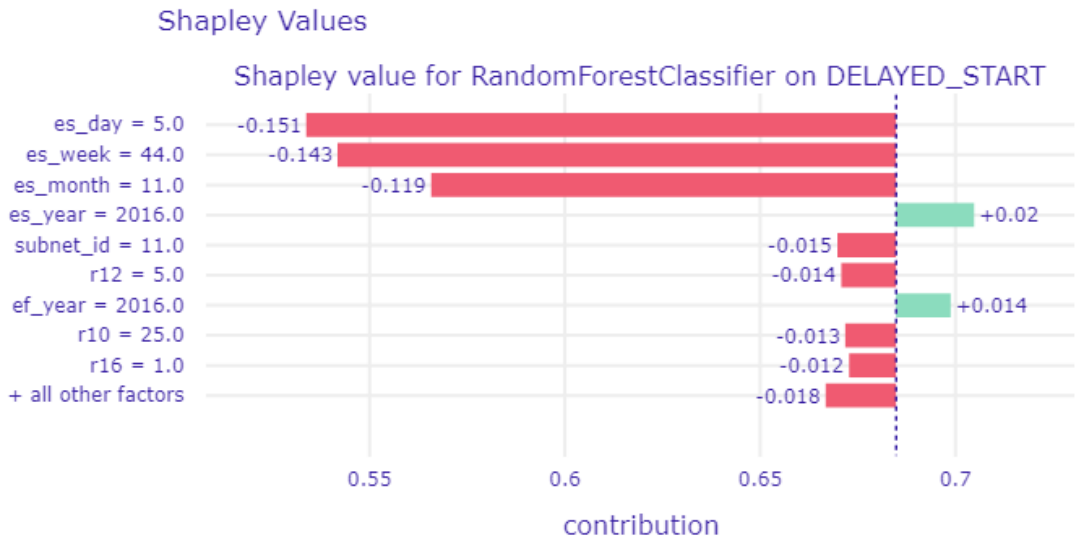


Figure 14: SHAP analysis of a single feature for `DELAYED_START`.

Figure 15 shows that the top features had a significantly lower impact on the prediction of `DELAYED_FINISH` than on `DELAYED_START`. The feature value combination with the highest Shapley value was `ef_year = 2016` with a Shapley value of +0.053, significantly lower than the third most influencal feature value combination from Figure 14. The low Shapley values for `DELAYED_START` compared to `DELAYED_FINISH` mirrors itself in Table 10 where the F1 score of the Random Forest model is lower for `DELAYED_FINISH` than for `DELAYED_START`.

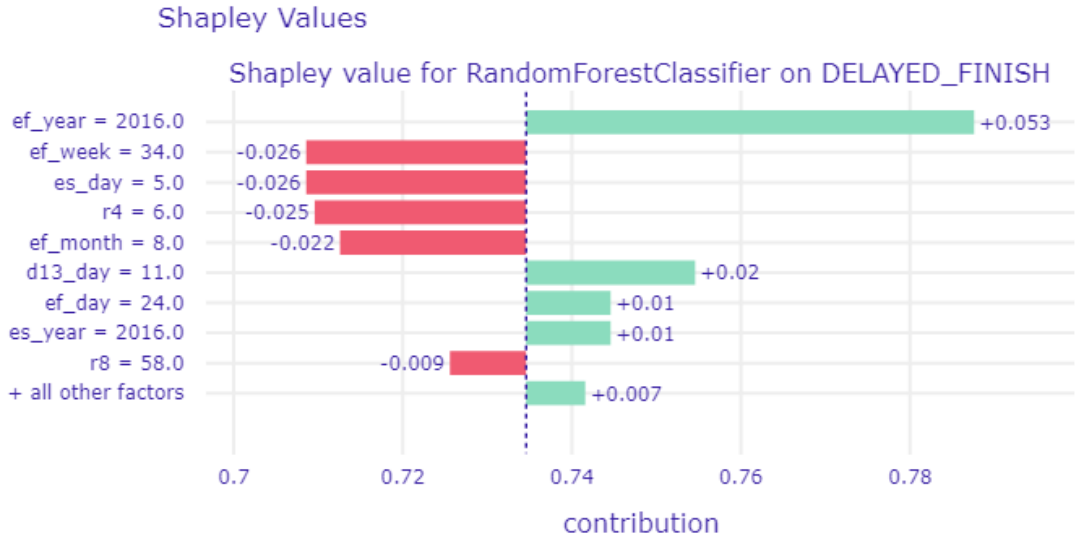


Figure 15: SHAP analysis of a single feature for `DELAYED_FINISH`.

Worth noting is that this is the Shapley values of a single instance in the test dataset. The Shapley

values change from instance to instance and can only be used to investigate how the model makes a prediction. Figure 16 and 17 show the Shapley values for another instance in the test set. This is an instance with ES the 30th of January 2017 and a EF the 3rd of June 2017. Notice that the Shapley values are smaller than in Figure 14. In addition, there are several new features which were not among the top contributors for the other instance.

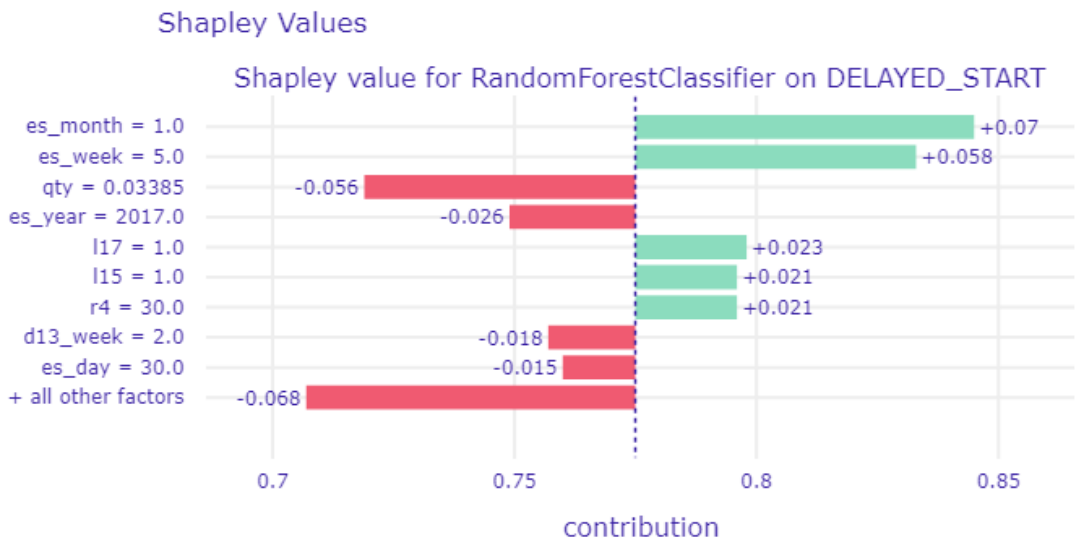


Figure 16: SHAP analysis of a single feature for `DELAYED_START`.

The same instance, but predicted for `DELAYED_FINISH` (Figure 17), shows that `qty = 0.03385` was the feature value combination with the highest Shapley value.

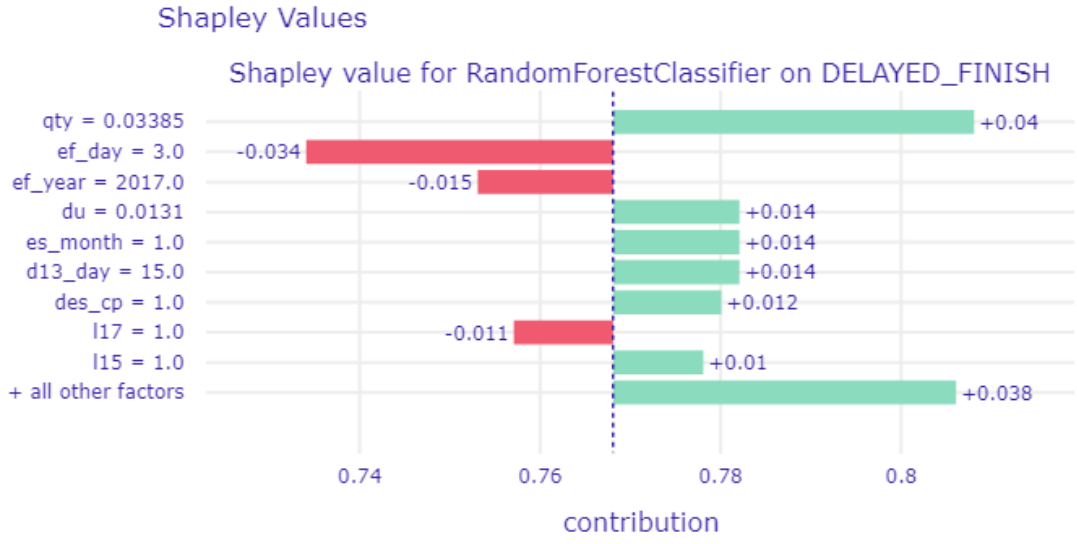


Figure 17: SHAP analysis of a single feature for `DELAYED_FINISH`.

5 Discussion

In this section, the research questions presented in the introduction will be discussed. This section will first address the performance of the models, then address each research question chronologically. Lastly, the limitations of the approaches used in this thesis will be discussed.

5.1 Model Performance

The four models presented in Section 3.7 all performed with satisfactory results. The Random Forest model scored highest in mean weighted F1 score over the grid search. When retrained on the training set, this model outperformed all other on `DELAYED_START` and performed the best in terms of accuracy and F1 score on the `DELAYED_FINISH` target. The results from training and validation can be seen in Table 10. In the complete dataset, 72% of the instances have a delayed start, while 74% have a delayed finish. Therefore, the accuracy of all models has to be compared with these percentages to give a real indication of performance. For `DELAYED_START`, the Random Forest model had an accuracy of 87.0%. This indicates that 13% of all predictions were wrong. In predicting `DELAYED_FINISH`, the precision was 84.1% compared to 74% of positive instances. This result is significantly worse than that of the other label. Both results show that the model is not purely guessing, but making somewhat accurate predictions. However, the relatively low result indicates that there is a significant room for improvement.

In order to use a model like this as an early warning system, the most important performance metric to maximise is the recall. Section 3.8 arguments that in order to reduce the number of FN predictions, recall is the most important performance metric. Equation 3 shows that the recall takes the number of TP predictions over the sum of TP and FN. The recall of the Random Forest model is 92.7% on the `DELAYED_START` target and 91.8% on the `DELAYED_FINISH` target. This means that the model misclassifies 6.3% and 8.2% of the positive instances respectively. On the latter, it was marginally beaten by both AdaBoost and Gradient Boosting. Figure 9 showed the confusion matrix for Random Forest. The model had only 120 FN predictions. That is, of the 1635 instances with a delayed start, the model only misclassified 120 of them. Had this model been implemented as an early warning system on this project, 1515 activities would receive an early warning flag that was correct, and 120 activities would not have received an early warning flag when it should have. Although Random Forest had a lower recall score than both AdaBoost and Gradient Boosting, the model outperformed both in terms of precision. This is shown in the F1 score, where Random Forest was the model with the highest F1 score on the `DELAYED_FINISH` label.

One can argue that the resulting models are too unprecise to be implemented in a real-world application. This thesis focused on tree-based machine learning models for two reasons; (1) it would give a relative comparison between models that are effective on categorical data, and (2) the models can be explainable with either intrinsic explanations or ad hoc methods. The following sections will highlight whether the model could be implemented as an early warning system.

5.2 Could this be Implemented?

RQ1 asked "How can data from large project databases be extracted and processed to be used for an ML analysis?". To answer this question, the earlier steps in Section 3 will be discussed. Firstly, to implement AI and ML solutions in a company, there must be a clear vision for its use and need. These solutions, more specifically MLBPM solutions, can be of high value to the project manager and other key personnel. However, there is no use for undefined solutions that will likely remain unused. The key component of an MLBPM early warning system is that the project manager will get insight into which activities are prone to delays. This can be used during scheduling and controlling. For scheduling, the project manager can schedule activities, set durations, and start times for activities. After the activities are scheduled, the software can inform the project manager of which activities are prone to delays, which project managers can use to input more slack into specific activities and linked activities. Additionally, this system can be used when setting new baselines, specifically when implementing VOs. New activities added to the current baseline would come with a flag that notifies the project manager whether they are prone to delays.

Had the Random Forest model been implemented as an MLBPM early warning system in the third party software as is, the results would have been of use to the project manager. As highlighted in Section 5.1, the Random Forest model had a recall of 92.7% for `DELAYED_START` and 91.8% for `DELAYED_FINISH`. In other words, the model will capture 92.7% of all activities that will have a delayed start and 91.8% of all activities that will have a delayed finish. However, the predictions come with some noise. As Table 10 shows, the precision of the Random Forest model is lower than the recall. The model had a precision of 89.9% on `DELAYED_START` and 87.7% on `DELAYED_FINISH`. In the real-world application, this means that of all the activities that the model predicts as having a delayed start, 10.1% of them will in fact be on time. This is a significant noise. In the event that all flagged activities will be further controlled by the project manager, over a tenth of all controlled activities are unnecessary to investigate. Similarly for delayed finish, 12.3% of all flagged activities are, in fact, on time. Although the goal was to minimise FN, the amount of FP comes with a cost. The increase in working hours from controlling leads to an increase in project cost. Furthermore, the high amount of noise might lead to project managers mistrusting the system and decrease the

likelihood that project managers want to use it.

Implementing the MLBPM early warning system is not an overly complicated process. The steps elaborated at the beginning of Section 3 can be easily streamlined and automated. Had this system been developed during the project, all completed activities could have been used as input for the model. The data could have been processed as explained in Section 3.6, and the trained models would have been implemented as a system. When new activities are imported or created in the software, the model would make two predictions for that activity; Will it have a delayed start and will it have a delayed finish? If either of the questions is asserted, then the activity will be flagged. The probability of the delay, i.e. the model's confidence, could also have been added to give insight into how likely the delay is. Additionally, if the model had been implemented on tree-based algorithms like the ones suggested in this thesis, simple splitting rules could have been added to the output to give the project manager insight into why the model predicted a delay for an activity. Alternatively, using more complex models as neural networks would increase the needed amount of preprocessing, and the model would have been more complex. This would have been a black-box method were no intrinsic insight into how the predictions were made could be given.

There are, however, some issues and concerns with implementing this system as is. Firstly, the model is only trained on one project, making it specialised for that project and not able to generalise. Additionally, the flags say nothing about the degree of delay. Some activities could be delayed within the same week, which is not a very big problem. Other activities could be delayed for several weeks or months. Section 5.5 will discuss the limitation of this method as is, while Section 6.2 will assess suggested improvements to the method.

5.3 Importance of VO Features

Section 4.3.1 addressed the PFI of all variables from VOs, and showed that only three of the features, `n_vos`, `baseline_id_1` and `responsible_2`, had a PFI score different from zero in predicting `DELAYED_START`. In predicting `DELAYED_FINISH`, only two features had a PFI score different from zero; `description_10` and `baseline_id_1`. In total, only 4 of the 17 features from the `vo_reg` table had any importance in terms of PFI. The low PFI scores on all features from the `vo_reg` table are undoubtedly a factor of the low occurrence of these features. Only 0.18% of all activities in the dataset were affected by VOs. Of the 11194 activities in the dataset, only 20 activities were affected by VOs. Of these, 15 were affected by one VO (i.e., `n_vos` = 1), 4 were affected by 2 VOs and 1 was affected by three VOs. This explains why the models did not use any of the features

extracted from the `vo_reg` table.

However, since the PFI analysis showed that some of the models attributed importance to these four features, there may be some predictive power. Section 5.5.1 will address how the dataset could be altered so that there is more predictive power in the features from VOs. Although it might be possible to use these features for predicting delays, the results show that these features are inessential for this constructed dataset. Since the permutation of these features yield little to no difference in performance, the removal of said features will likely not degrade the predictive performance.

Features	DELAYED_START		DELAYED_FINISH	
	VO instances	All instances	VO instances	All instances
n_vos	0	0.022	0	-0.019
baseline_id_1	0.096	0.010	0.099	-0.003
baseline_id_2	0.053	0.006	-0.254	-0.016
responsible_1	-0.053	0.021	0.254	-0.015
responsible_2	0.096	0.010	0.099	-0.003
responsible_3	0.053	0.006	-0.254	-0.016
responsible_4	0.124	0.021	-0.235	-0.022
description_1	0.150	0.003	0.373	-0.003
description_2	0.150	0.004	0.373	-0.001
description_3	0.150	0.003	0.373	-0.003
description_4	0.150	0.003	0.373	-0.003
description_5	0.150	0.003	0.373	-0.003
description_6	0.150	0.004	0.373	-0.001
description_7	0.150	0.003	0.373	-0.003
description_8	0.150	0.003	0.373	-0.003
description_9	0.150	0.003	0.373	-0.003
description_10	0.150	0.006	0.373	-0.010

Table 15: Correlation between the VO features and the labels. The correlation is plottet both in relation to all instances, and the 20 instances affected by VOs.

One factor that might explain why some of the features had a PFI score different from zero is its correlation with the target. Table 15 shows the correlation between each VO feature and the targets. Those marked in blue is the ones that had a PFI score different from zero. The correlations show no significant difference in correlation between the selected features and the others. Although they are among the highest in terms of correlation, other features score higher. For instance, the correlation between `responsible_4` for all instances on `DELAYED_START` is higher than that of `responsible_2` and `baseline_id_1`. For `DELAYED_FINISH`, both features marked in blue are outscored in terms of correlation by 5 other features. Thus, the correlation between the VO features and the targets does not give an indication as to why these four features should

outperform the others in terms of the PFI score. An explanation for this is that the features are under-represented. Thus, the 20 instances affected by VOs are more random than correlated, and the correlation between the VO features and targets on the complete dataset gives no indications because the VO features are heavily under-represented. With a more equally weighted distribution of VO features, i.e., more activities affected by VOs, both the PFI analysis and correlation would probably have given totally different results. Section 5.5.1 will further discuss how the dataset could have been altered to give more weight to the VO features.

5.4 Most Important Features

5.4.1 Date Features

Both the permutation feature importance analysis and the SHAP analysis provided clear results on which features were the most dominant. For both `DELAYED_START` and `DELAYED_FINISH`, the date features were the most dominant. For `DELAYED_START`, four of the ES features were at the top of the PFI analysis (see Section 5.4). The fifth most important feature was `r12`, with less than half of the importance of the fourth most important feature `es_month`. Similarly, for `DELAYED_FINISH`, the top four features were EF features where the fourth feature, `ef_day`, was more than twice as important as the fifth feature, `es_year`. Additionally, the creation date of the activity, `d13`, appeared in two of the top 10 most important features for `DELAYED_START`. Figure 18 shows the distribution of the label for each year and month of the project. This plot shows a high correlation between the features and the label. For instance, only 9.4% of all activities with an ES in 2018 were delayed, while in 2016 80.1% of all activities had a delayed start. The same applies for months, where 36.6% of all activities with an ES in December were delayed, while 93.5% of all activities with an ES in January were delayed. This high correlation is probably connected to the high PFI of these two features.

Figure 18: Distribution of `DELAYED_TARGET` over all project years and months.

The SHAP analysis also supports that the date features were the most important. Figure 14 shows that the Shapley value for an instance with `es_month` in January increased by 0.07, while the Shapley value for the instance in Figure 14 with `es_month` in November sunk by 0.143. Comparing these results to Figure 18, the increase in January and the decrease in November can be explained by the amount of delayed activities by these two months. Furthermore, the Shapley value for the same instance increased by 0.02 due to having `es_year` in 2016. Neither the feature `es_dayofweek` nor `ef_dayofweek` were as important as the other ES and EF features. This shows that the distribution of delays follows the specific date rather than weekly trends. Figure 19 shows the percentage of delayed activities for each month in terms of ES. This shows a strong trend in the dataset which is probably why the model relies so much on the ES features.

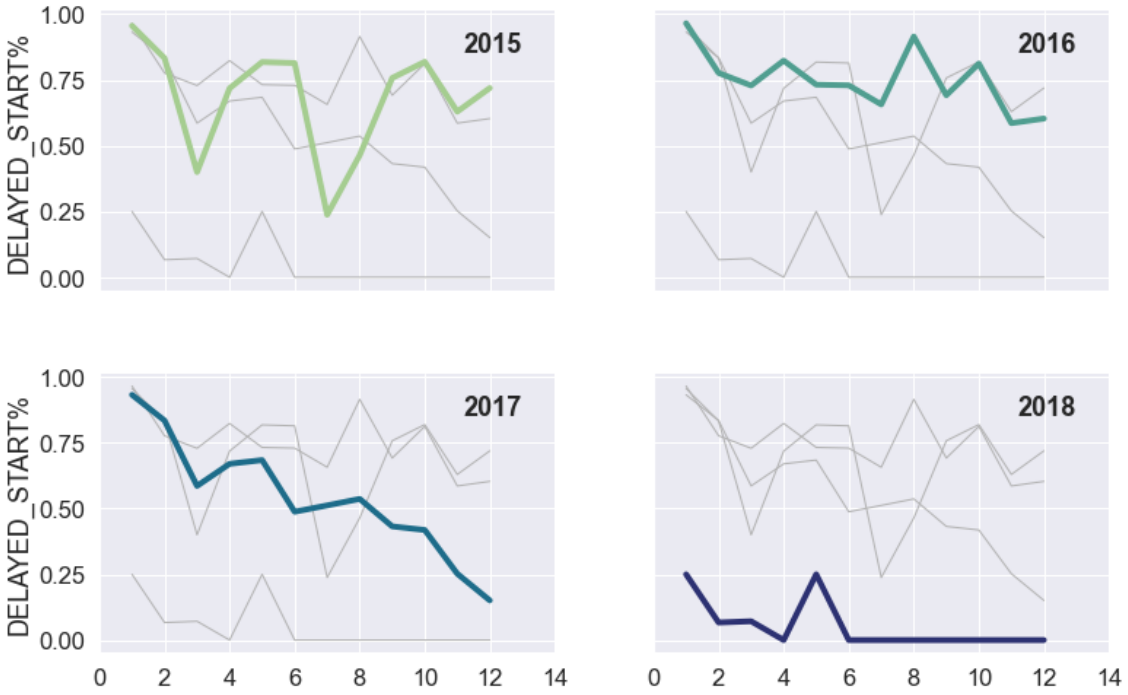


Figure 19: Distribution showing the percentage of `DELAYED_START` for each `es_year` and `es_month`.

One concern with the strong trend in the dataset is that the models do not generalise. Applying these models to other projects would likely yield worse results than the ones presented in this thesis. Section 5.5.3 will go into this issue more thoroughly.

5.4.2 Categorical and Flag Features

Second only to the date features in PFI are the categorical features shown in Table 11. Although these features were outranked by date features, several were included in individual models in the top ten features, as shown in Table 11. The categorical features were generally assigned a higher PFI score than the flag features, with a mean PFI of 2.6×10^{-3} on `DELAYED_START` and 1.5×10^{-3} on `DELAYED_FINISH`. The flag features scored generally low in PFI with 1.6×10^{-4} on `DELAYED_START` and -3.7×10^{-5} on `DELAYED_FINISH`. Note that a negative PFI score indicates that the removal of that feature leads to an improvement in the F1 score. Thus, the flag features generally reduced the predictive performance of the models on the `DELAYED_FINISH` target.

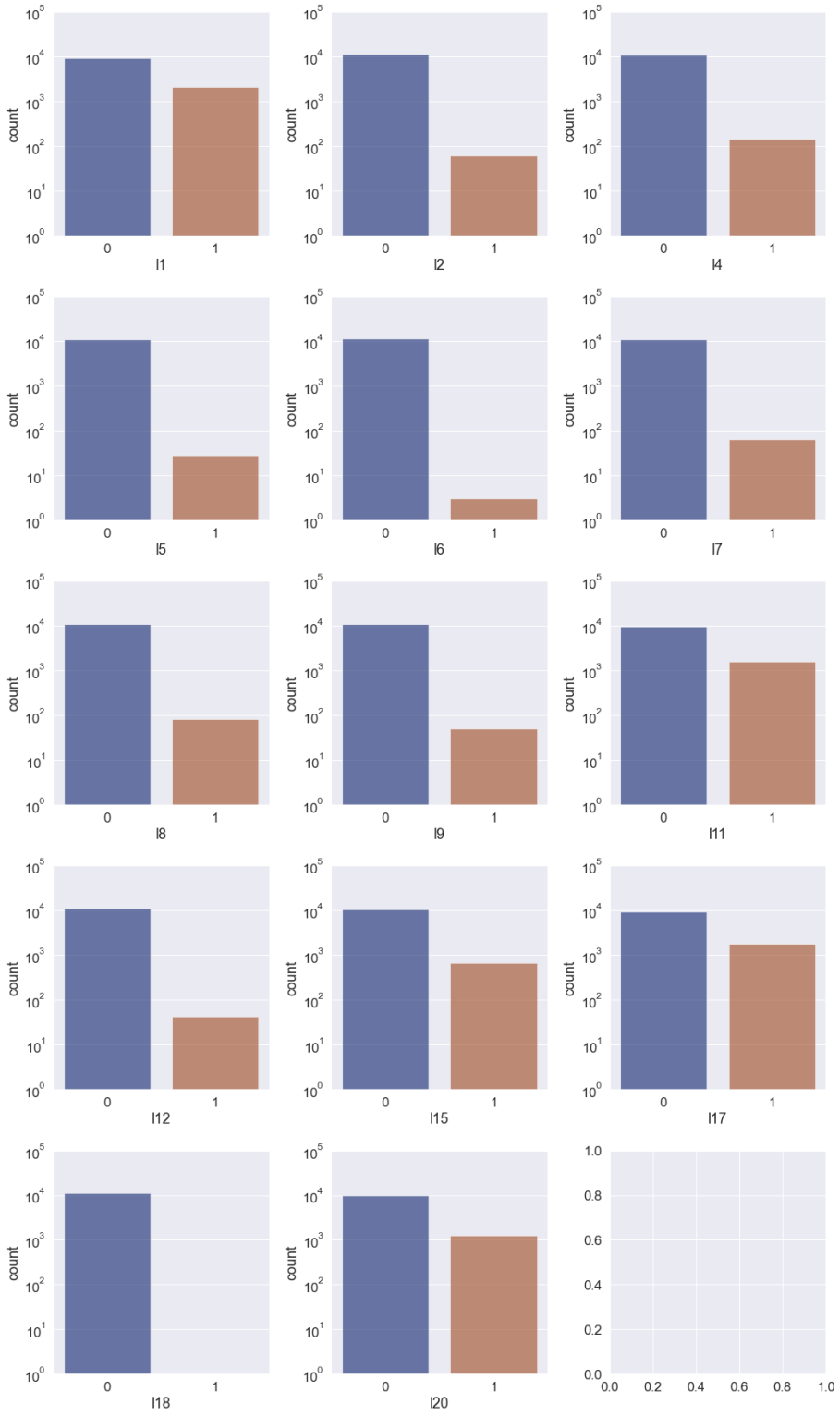


Figure 20: Distribution of the flag features. Note that the y-axis is logarithmic.

One reason why flag features scored significantly low in PFI and in some cases led to a model improvement when removed is that features are severely imbalanced. Figure 20 shows the distribution of the flag features. This plot shows that instance 1/true is severely under-represented for

several instances. 6 of the 15 features have less than 100 positive instances while 2 features have less than ten positive instances.

5.5 Limitations

5.5.1 The Amount of Activities Affected by VOs

The main issue in asserting RQ2 is this low occurrence of features affected by VOs. With 0.18% of activities affected by one or more VOs, these features are severely imbalanced. As discussed in Section 5.3, the amount of activities affected by VOs was too low to give the VO features any significant importance. Traditionally, it is recommended to remove features that are highly imbalanced and do not have any significant correlation with the target. Table 15 showed that the VO features are uncorrelated to the label on all instances. However, when calculating the correlation only on the 20 instances where VOs are present, the correlation was significantly higher for some features, specifically description features. Further investigation into these instances showed that 6 instances had a value of 1 for all description features, while the other instances had a value of 0. This shows that the original descriptions, before preprocessing, were imbalanced and should be removed. Thus, the remaining VO features have a lower correlation with the labels. Given that the correlation between the remaining features and the labels is relatively low, the VO features could have been removed without affecting the model performance.

Although the VO features were insignificant with this dataset, a different distribution could have yielded completely different results. There are several solutions that might have improved the importance of the VO features. For instance, the models could have used either the ensemble technique voting or stacking. In voting, several base models are created, and the final prediction is the average or weighted average of the base models' predictions. In stacking, the same procedure for base models is followed, only that their predictions are used as input for a logistic regression algorithm. In this way, a model can be created that combines predictions from a dataset on only VO free activities and a dataset on only VO affected activities. This should in theory capture the effect of VOs to a greater extent.

Another approach that could provide more information on the importance of VOs is to over-sample the instances that are affected by VOs. This could be from copying instances where VOs are present or generating slightly different instances through synthetic data generation. Although there exist methods that could have given more insight into RQ2, the practical implementation is questioned. If a MLBPM early warning system could operate without using VO features as input, then there

is no need to significantly complicate the model or generate synthetic data that could produce impossible instances.

5.5.2 Feature Selection

The features used in the dataset are a combination of features that were complete over most instances, and was found through interviews with the company to be of high importance to the individual activities. Section 5.4.1 proved that there is a high correlation between the date features and the targets, additionally, the distribution of delays is imbalanced as discussed in Section 5.4.2. However, there might be different subsets of features that would yield a higher predictive performance.

The process of creating a dataset with the fewest number of features that yield the highest predictive performance, feature selection, is highly studied, and there are several techniques that could aid in selecting the right features. Shardlow (2016) arguments that feature selection aims to reduce the size of the problem (i.e., cumpute time and space), improve classifiers by removing noisy or irrelevant features, and reduce the chance of overfitting to noisy data. Section 5.4.1 discussed that the model is highly sensitive to date features. This could be a sign of the models overfitting and relying to much on the date features in their predictions. Shardlow (2016) gives several examples of methods that could yield better results by feature selection. He structures the methods into three main methods:

- **Filter methods:** Applying ranking to the features, either by correlation, mutual information, relief, or data permutation. Then select the n top features. For some models, the lowest test error (or the highest recall for this application) may be from a dataset with fewer features.
- **Wrapper methods:** These methods wrap the classifier in a feature selection algorithm. These are typically search algorithms like Greedy Forward Search which iteratively adds one more feature and only keeps it if the improvement is significant, or Exhaustive Search which brute forces all possible feature combinations.
- **Hybrid methods:** This method combines filtering and wrapping, to obtain a better classification scheme. Methods include Ranked Forward Search, which combines ranking and Greedy Forwards search, and Refined Exhaustive Search, which first selects a subset of N features from ranking, and then performs Exhaustive Search on those.

Although these methods were not applied for this thesis, there are certain examples where these

methods could be used to yield better predictive performance. For instance, applying the filter method with ranking on mutual information would eliminate the feature `r3` (WBS) as the feature `r22` (WBS2) is a subset of that feature. The WBS2 feature is a further breakdown of the WBS. A specific value for `r22` would give the value for `r3`, thus this feature is uneccessary. Although there are more examples of features that might have been removed due to a feature selection analysis, the improvements that might be made from those removals were outweighed by the main goal of this analysis. As the main objective was to address whether it was possible to predict activity delays from VOs, feature selection was given a low priority.

5.5.3 Diversification

The largest limitation of this thesis is that the models, and the methodology as a whole, is based on a single project. Firstly, this violates the idea of using this method as an early warning system. To train a model, a substantial amount of training data must be present. The size of the dataset (i.e. the number of instances) is directly correlated to the model's tendency to overfit and not generalise. If the model was, for instance, trained on 100 activities, then it would likely be overfitting and not able to generalise for other activities. So, to gather enough data on which activities that actually were delayed, the data had to be collected long after the project started. Therefore, the early warning system would only be available for later stages of the project. Additionally, all activities that the model is trained on are from the earlier stages of the project. There might be almost zero correlation between activities in the early stages of a project compared to those in the later stages, making the early warning system useless.

In order to create a robust model that does not overfit to the training data and that is able to generalise, the training data has to be collected from a multitude of different projects. Section 5.2 discussed that this method can be implemented as an early warning system, however, there may be a need for major changes in preprocessing and feature creation. The date features that signifies the year (i.e. `es_year`, `ef_year`, and `d13_year`) are all specific for this project. For instance, Figure 18 showed that the concentration of delayed activities by ES is highest in 2016. This begs the question: Is this because this is the second project year or because 2016 was a year where activities were more prone to be delayed? Questions like this were not relevant when creating a model based on just one project, but when collecting a multitude of different project data, they are highly relevant. Although the method implemented in this thesis is applicable, the creation of a more robust model based on several projects requires different preprocessing.

6 Conclusion

The main research objective of this thesis was to investigate whether variation orders can be used to predict the time delay of individual activities. In this master's thesis, a compound dataset of four different datasets has been created. This dataset was used to train four different tree-based classification models to predict whether an activity had a delayed start or finish based on the ES and EF. Both a PFI and SHAP analysis was used to evaluate the feature importance, both on dataset level and instance level. The conclusion is structured in two parts. The first parts will address the main findings, specifically in relation to the research questions proposed in Section 1.2. The second part will suggest further work in relation to the main findings of this master's thesis.

6.1 Main findings

The first research question concerns how data from large project databases can be extracted and processed to be used for an ML analysis. To assess this research question, the procedures described Section 3.2, 3.4, 3.5 and 3.6 are presented as a solution. In summary, project data from large projects are usually distributed over a project management software that separates data into a multitude of different tables. The softwares are primarily developed for the front-end use, i.e. the user interface is prioritised. However, Section 3.2 showed that the data can be extracted from this software. As project management software typically hosts a wide variety of functionality, most of the extracted data will come with unnecessary features. Some feature removal was done in Section 3.4, but the discussion on feature importance in Section 5.4 argued that several other features could be removed. The four models implemented required a limited amount of preprocessing, which increased the explainability of the models. This thesis has provided a suggestion on how data from large project databases can be used for an ML analysis. Additionally, Section 5.2 argued that this process can be pipelined and implemented in a real-world application.

The second research question concerns whether VOs can be used to predict activity delays. The permutation feature importance analysis showed that the VOs had little to no contribution to the predictions made by the four models. Section 5.5.1 accredited the low contribution of the VO features to the low occurrence of VOs in the constructed dataset. Furthermore, several of the VO features were imbalanced for the 20 instances present. Although the VOs had little to no contribution to the prediction of activity delays for this thesis, Section 5.5.1 argued that this contribution can be increased by over-sampling VO instances or using other ensemble methods like voting and stacking.

The third research question concerns what features are most dominant in predicting the activity delay. Both thePFI analysis and the SHAP analysis proved that the date features were the most important in predicting the activity delay. Section 5.4.1 discussed that the high importance of these features may be attributed to the distribution of activity delays. Figure 19 showed that for `DELAYED_START`, the variance in the distribution of delays is high between the different years and months. In addition to date features, several categorical features presented in Table 11 were assigned a high PFI score. This indicates that the categorical features were also important in the predictions, although outranked by the date features.

6.2 Further Work

To fully understand the influence of VOs, the dataset has to be constructed differently. The models presented in this thesis attributed little to no importance to the VO features and indicated that they are irrelevant to the models' predictive performance. However, as discussed in Section 5.3, the features might be used to predict activity delays. A suggestion to further work is to reduce the imbalance of activities that are affected by VOs. Additionally, since the date features were given a high PFI score, this indicated that these features were highly relevant in predicting activity delays. By reducing some of the features with a high PFI score, the models will have to rely more on the VO features. In this way, the degree to which VOs can be used to predict an activity delay may be evaluated more precisely.

Another suggestion for further work is to analyse in more depth whether a similar approach can be used as an MLBPM early warning system implemented in a project management software. To properly address this theory, a dataset based on a multitude of different projects will have to be created. The possible improvements by this approach was discussed in Section 5.5.3. Additional model selection, preprocessing, and hyperparameter tuning may have to be investigated to reduce the number of FP and FN predictions. Additonally, it is recommended to do a feature importance analysis as a part of feature selection, this way, the model can be fitted to a limited subset of features that yield a great predictive performance by reducing correlation between features and misleading features. Alternatively, other feature selection algorithms, such as fmGA, may be included, similar to the approach of Cheng et al. (2010). To assess model performance, a small variety of projects is recommended to test on. Testing on a single project might be too limited and will not truly show whether the developed system is truly generalising on projects.

In addition to the major alteration to the approach, two minor suggestions for further work were found. The first is to estimate the amount of delay. Gondia et al. (2020) investigated whether it

is possible to predict the amount of project delay and classified the delay into three class labels; $< 30\%$ TO, $30\% - 60\%$ TO and $> 60\%$ TO. Unlike this paper, the suggestion is to estimate the delay of individual activities by the number of days overdue, i.e., to predict the actual date. In project management, an activity that is one day delayed is significantly less of a concern than an activity that is a month delayed. This approach would capture the significance of the delays, which will give the project manager even more insight. The other minor modification is to include the interaction between activities. Choetkiertikul et al. (2015) introduced this approach for individual software project tasks by creating a network of tasks. The tasks were connected if, for instance, they were developed by the same developer. Thus, if a precursory task for the developer was delayed, then the probability of the current task being delayed increased. To implement this approach on construction data could be to connect project activities that are linked in the scope, activities that are on the same building block, or activities that are performed by the same subcontractor. These interactions are already somewhat present by the categorical features, but there might be improvements made from networking activities.

References

Aarvold, M. O. & Hartvig, W. J. (2021). *Machine learning on complex projects: Multivariate time series data analysis through utilization of the sequential algorithm lstm* (Master's thesis). NTNU.

Akinsola, A., Potts, K., Ndekugri, I. & Harris, F. (1997). Identification and evaluation of factors influencing variations on building projects. *International Journal of Project Management*, *15*(4), 263–267. https://doi.org/10.1016/S0263-7863(96)00081-6

Al Hammadi, A. S. (2009). *Variation order (change order) in oil & gas projects* (Doctoral dissertation). British University in Dubai.

AlSheibani, S., Cheung, Y. & Messom, C. (2018). Artificial intelligence adoption: Ai-readiness at firm-level. *Twenty-Second Pacific Asia Conference on Information Systems*, 37.

Alsuliman, J., Bowles, G. & Chen, Z. (2012). Current practice of variation order management in the saudi construction industry. In S. Smith (Ed.). Association of Researchers in Construction Management.

Alto, V. (2020). Understanding adaboost for decision tree. Retrieved 21st May 2022, from https://towardsdatascience.com/understanding-adaboost-for-decision-tree-ff8f07d2851

Amini, M. R. (2015). Adaboost. Retrieved 25th May 2022, from http://ama.liglab.fr/~amini/AdaBoost/

Arain, F. M. & Pheng, L. S. (2005). The potential effects of variation orders on institutional building projects. *Facilities*, *23*(11/12), 496–510. https://doi.org/10.1108/02632770510618462

Atkinson, R. (1999). Project management: Cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International journal of project management*, *17*(6), 337–342. https://doi.org/10.1016/S0263-7863(98)00069-6

Auth, G., JokischPavel, O. & Dürk, C. (2019). Revisiting automated project management in the digital age–a survey of ai approaches. *Online Journal of Applied Knowledge Management (OJAKM)*, *7*(1), 27–39. https://doi.org/10.36965/OJAKM.2019.7(1)27-39

Baccarini, D. (1999). The logical framework method for defining project success. *Project management journal*, *30*(4), 25–32. https://doi.org/10.1177/875697289903000405

Benachour, M. (2018). Benchmarking change orders (variation order) in oil and gas contract versus fidic, aia, ejcdc and consensus docs. *PM World Journal*, *7*(7), 1–12.

Biecek, P. & Tomasz, B. (2020). Shapley additive explanations (shap) for average attribution. Retrieved 10th February 2022, from https://ema.drwhy.ai/shapley.html

Bower, D. (2000). A systematic approach to the evaluation of indirect costs of contract variations. *Construction Management & Economics*, *18*(3), 263–268. https://doi.org/10.1080/014461900370636

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123–140. https://doi.org/10.1007/BF00058655

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Cheng, M.-Y., Peng, H.-S., Wu, Y.-W. & Chen, T.-L. (2010). Estimate at completion for construction projects using evolutionary support vector machine inference model. *Automation in Construction*, *19*(5), 619–629. https://doi.org/10.1016/j.autcon.2010.02.008

Choetkiertikul, M., Dam, H. K., Tran, T. & Ghose, A. (2015). Predicting delays in software projects using networked classification (t). *2015 30th IEEE/ACM international conference on automated software engineering (ASE)*, 353–364. https://doi.org/10.1109/ASE.2015.55

De Wit, A. (1988). Measurement of project success. *International journal of project management*, *6*(3), 164–170. https://doi.org/10.1016/0263-7863(88)90043-9

Dursun, O. & Stoy, C. (2016). Conceptual estimation of construction costs using the multistep ahead approach. *Journal of Construction Engineering and Management*, *142*(9), 1–10. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001150

El-Kholy, A. M. (2013). Modeling delay percentage of construction projects in egypt using statistical-fuzzy approach. *IOSR Journal of Mechanical and Civil Engineering*, *7*(5), 47–58.

Fellows, R. F. & Liu, A. M. (2003). *Research methods for construction*. Blackwell Science Ltd.

Flyvbjerg, B. (2013). Over budget, over time, over and over again: Managing major projects, 321–344. https://doi.org/10.1093/oxfordhb/9780199563142.003.0014

Gondia, A., Siam, A., El-Dakhakhni, W. & Nassar, A. H. (2020). Machine learning algorithms for construction projects delay risk prediction. *Journal of Construction Engineering and Management*, *146*(1), 1–16. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001736

Hanif, H., Khurshid, M. B., Lindhard, S. M. & Aslam, Z. (2016). Impact of variation orders on time and cost in mega hydropower projects of pakistan. *Journal of Construction in Developing Countries*, *21*(2), 37. https://doi.org/10.21315/jcdc2016.21.2.3

Hanna, A. S., Camlic, R., Peterson, P. A. & Nordheim, E. V. (2002). Quantitative definition of projects impacted by change orders. *Journal of Construction Engineering and Management*, *128*(1), 57–64. https://doi.org/10.1061/(ASCE)0733-9364(2002)128:1(57)

Heravi, G. & Eslamdoost, E. (2015). Applying artificial neural networks for measuring and predicting construction-labor productivity. *Journal of Construction Engineering and Management*, *141*(10), 1–11. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001006

Hsieh, T.-y., Lu, S.-t. & Wu, C.-h. (2004). Statistical analysis of causes for change orders in metropolitan public works. *International Journal of Project Management*, *22*(8), 679–686. https://doi.org/10.1016/j.ijproman.2004.03.005

Ibrahim, Y. M., Kaka, A., Aouad, G. & Kagioglou, M. (2009). Framework for a generic work breakdown structure for building projects. *Construction Innovation*, *9*(4), 388–405. https://doi.org/10.1108/14714170910995930

Jergeas, G. (2008). Analysis of the front-end loading of alberta mega oil sands projects. *Project Management Journal*, *39*(4), 95–104. https://doi.org/10.1002/pmj.20080

Jin, X.-H. & Zhang, G. (2011). Modelling optimal risk allocation in ppp projects using artificial neural networks. *International journal of project management*, *29*(5), 591–603. https://doi.org/10.1016/j.ijproman.2010.07.011

Kamath, U. & Liu, J. (2021). *Explainable artificial intelligence: An introduction to interpretable machine learning*. Springer. https://doi.org/10.1007/978-3-030-83356-5

Kaul, A. B. (2017). *Microelectronics to nanoelectronics: Materials, devices & manufacturability*. CRC Press.

Keane, P., Sertyesilisik, B. & Ross, A. D. (2010). Variations and change orders on construction projects. *Journal of legal affairs and dispute resolution in engineering and construction*, *2*(2), 89–96. https://doi.org/10.1061/(ASCE)LA.1943-4170.0000016

Khan, A. (2006). Project scope management. *Cost engineering*, *48*(6), 12–16. https://www.proquest.com/docview/220450731

Khang, D. B. & Moe, T. L. (2008). Success criteria and factors for international development projects: A life-cycle-based framework. *Project management journal*, *39*(1), 72–84. https://doi.org/10.1002/pmj.20034

Kumaraswamy, M. & Thorpe, A. (1996). A computerised construction project management evaluation system. *Advances in engineering software*, *25*(2-3), 197–206. https://doi.org/10.1016/0965-9978(95)00097-6

Liberatore, M. J. & Pollack-Johnson, B. (2003). Factors influencing the usage and selection of project management software. *IEEE transactions on Engineering Management*, *50*(2), 164–174. https://doi.org/10.1109/TEM.2003.810821

Millhollan, C. (2008). Scope change control: Control your projects or your projects will control you! *PMI Global Congress–Denver, Colorado, USA*.

Mohri, M., Rostamizadeh, A. & Talwalkar, A. (2018). *Foundations of machine learning*. MIT press. http://www.jstor.org/stable/j.ctt5hhcw1

Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A. & Brown, S. D. (2004). An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, *18*(6), 275–285. https://doi.org/10.1002/cem.873

Natekin, A. & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, *7*, 21. https://doi.org/10.3389/fnbot.2013.00021

Oladapo, A. (2007). A quantitative assessment of the cost and time impact of variation orders on construction projects. *Journal of Engineering, Design and Technology*, *5*(1), 35–48. https://doi.org/10.1108/17260530710746597

Oyewobi, L. O., Jimoh, R., Ganiyu, B. O. & Shittu, A. A. (2016). Analysis of causes and impact of variation order on educational building projects. *Journal of Facilities Management*, *14*(2), 139–164. https://doi.org/10.1108/JFM-01-2015-0001

Peško, I., Mučenski, V., Šešlija, M., Radović, N., Vujkov, A., Bibić, D. & Krklješ, M. (2017). Estimation of costs and durations of construction of urban roads using ann and svm. *Complexity*, *2017*, 1–13. https://doi.org/10.1155/2017/2450370

PMI, P. (2016). Pulse of the profession report. *Pulse of the Profession report*.

Project Management Institute. (2001). *Project management body of knowledge (pmbok® guide)* (4th ed.).

PwC. (2018). Ai will transform project management. are you ready? Retrieved 3rd February 2022, from https://www.pwc.ch/en/insights/risk/ai-will-transform-project-management-are-you-ready.html

Rolstadås, A., Johansen, A., Olsson, N. & Langlo, J. A. (2014). *Praktisk prosjektledelse: Fra idé til gevinst*. Fagbokforlaget.

Schapire, R. E. (2003). The boosting approach to machine learning: An overview. *Nonlinear estimation and classification*, 149–171. https://doi.org/10.1007/978-0-387-21579-2_9

Schapire, R. E. (2013). Explaining adaboost. *Empirical inference* (pp. 37–52). Springer. https://doi.org/10.1007/978-3-642-41136-6_5

Shardlow, M. (2016). An analysis of feature selection techniques. *The University of Manchester*, *2016*, 1–7.

Smith, W. (2016). *The effect of variation orders on project cost and schedule overruns* (Doctoral dissertation). Stellenbosch: Stellenbosch University.

Sunday, O. A. (2010). Impact of variation orders on public construction projects. *26th Annual ARCOM Conference, Leeds, UK*.

Tjora, A. (2021). *Kvalitative forskningsmetoder i praksis* (Vol. 4). Gyldendal akademisk Oslo.

Upland Software. (2022). Baseline. Retrieved 13th May 2022, from https://uplandsoftware.com/psa/resources/glossary/baseline/

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P. et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354. https://doi.org/10.1038/s41586-019-1724-z

Wang, Q. (2019). How to apply ai technology in project management. *PM World Journal*, *8*(3), 1–12.

Wauters, M. & Vanhoucke, M. (2014). Support vector machine regression for project control forecasting. *Automation in Construction*, *47*, 92–106. https://doi.org/10.1016/j.autcon.2014.07.014

Zhou, Z.-H. (2012). *Ensemble methods: Foundations and algorithms* (1st ed.). CRC Press. https://doi.org/https://doi.org/10.1201/b12207

Zidane, Y. J.-T. & Andersen, B. (2018). The top 10 universal delay factors in construction projects. *International Journal of Managing Projects in Business*, *11*(3), 650–672. https://doi.org/10.1108/IJMPB-05-2017-0052

Appendix

A `Activities` table after data cleaning and feature removal

Feature	Description	Type
seq	Unique ID	int
du	Activity duration	float
wpn	Calendar	Category
subnet_id	Subproject ID	Category
r1	Sub contractor	Category
r2	Discipline	Category
r3	WBS	Category
r4	Sub discipline	Category
r7	Organisation	Category
r8	Area	Category
r10	Section	Category
r12	Category	Category
r16	Building block	Category
r22	WBS2	Category
l1	Anonymous	Flag
l2	Anonymous	Flag
l4	Anonymous	Flag
l5	Anonymous	Flag
l6	Anonymous	Flag
l7	Anonymous	Flag
l8	Anonymous	Flag
l9	Anonymous	Flag
l11	Anonymous	Flag
l12	Anonymous	Flag
l15	Anonymous	Flag
l17	Anonymous	Flag
l18	Anonymous	Flag
l20	Anonymous	Flag
es	Early start	Date
ef	Early finish	Date
acs	Actual start	Date
acf	Actual finish	Date
d13	Activity transfer date	Date
des	Activity description	Text

Table A.1: Remaining features of `activities` after data cleaning.

B Confusion matrices

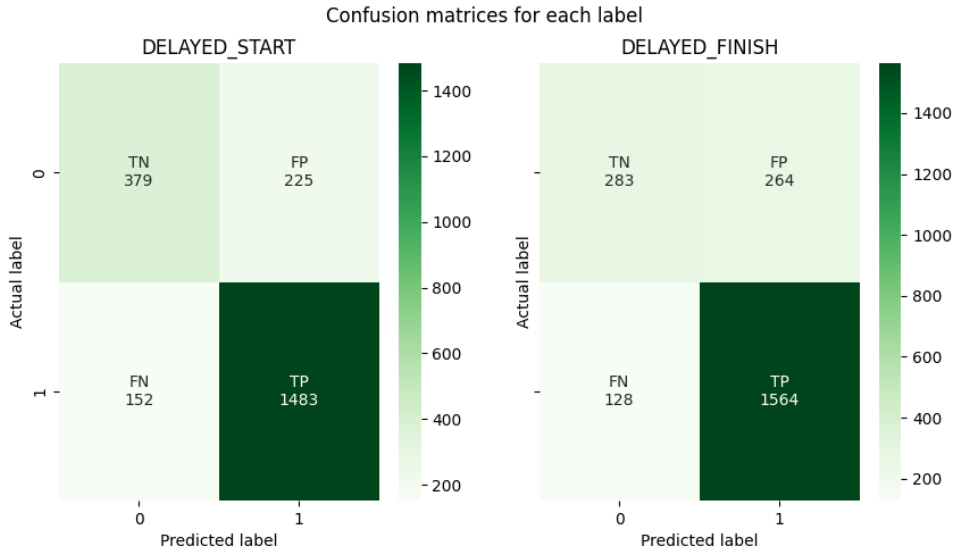


Figure B.1: Confusion matrices for `AdaBoost`.

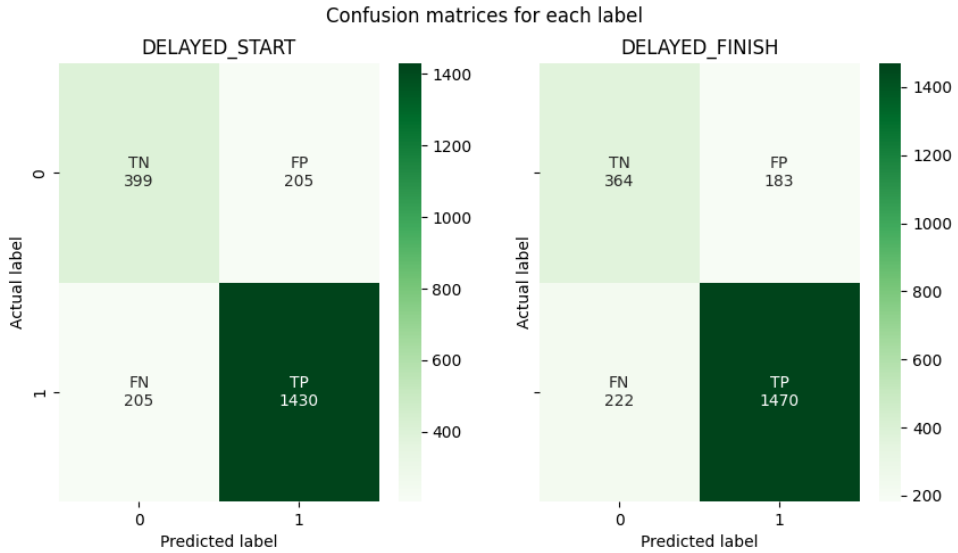


Figure B.2: Confusion matrices for `DecisionTree`.

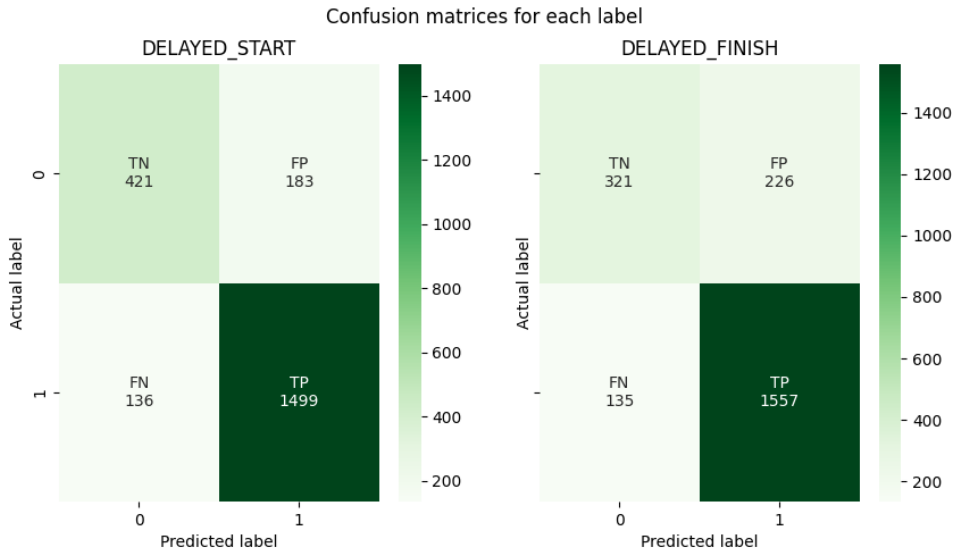
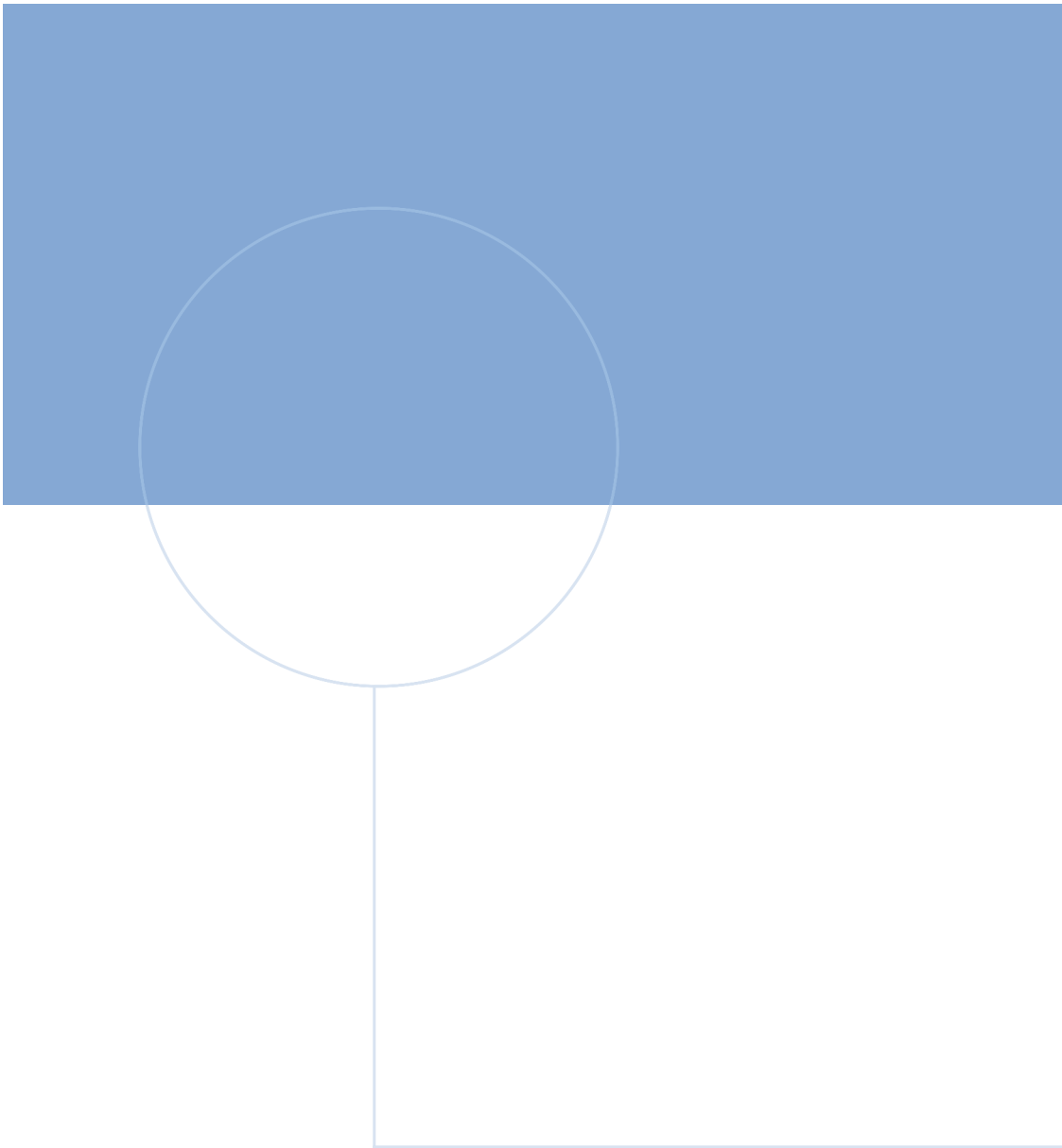


Figure B.3: Confusion matrices for `GradientBoosting`.



NTNU

Kunnskap for en bedre verden