Aalborg University

Master's Thesis

Time-Series Anomaly Detection for Industrial Screwdriving Task with Machine Learning Algorithms

Written by:
Ozren Vučićević

Supervised by:
Chen Li
Mohammadali Zakeriharandi

Group 14

Department of Materials and Production

Spring semester 2023

Jun 1st 2023

AALBORG UNIVERSITY
STUDENT REPORT

Aalborg University
Department of Materials and Production
http://www.mp.aau.dk

Title:

Time-Series Anomaly Detection for Industrial Screwdriving Task with Machine Learning Algorithms

Theme:

Classification of screwdriving with the use of machine learning algorithms

Project period:

Spring semester 2023

Project group:

Group 14

Participant:

Ozren Vučićević

Supervisors:

Chen Li

Mohammadali Zakeriharandi

Number of pages: 78

Date of completion:

Jun 1st 2023

Abstract

This project presents a comprehensive approach to anomaly detection in industrial screwdriving processes using machine learning (ML) algorithms. The primary hypothesis postulates that ML, applied to various process data from the screwdriving operation, can proficiently identify system performance irregularities, serving as a dependable anomaly detection instrument. The project follows several stages, beginning with data collection. A dataset consisting of time-series sensor data from the screwdriver, data from a UR10 robot, and audio data from a microphone was amassed to capture the aspects of the screwdriving process. Following this, preprocessing was employed to clean and format the data for further analysis. Subsequently, feature selection techniques were employed to identify the most informative attributes from the data, strengthening the predictive power of the ML models. The final step was the model-building phase, where classification algorithms were devised to distinguish between different classes of screws. This research is a collaborative effort carried out at Aalborg University, with a partnership from VELUX. They provided a real-world manufacturing context where this anomaly detection approach was derived from. In the future the aim is to enhance the quality and efficiency of their automated screwdriving production line by minimizing manufacturing anomalies. The successful execution of this project would validate the viability of integrating ML into industrial applications, but also sets the stage for future research focused on optimizing the performance of the algorithm and broadening its applicability to other manufacturing sectors.

Table of Contents

Preface

This paper was developed by a university student, as his Master's degree project in the Department of Materials and Production at Aalborg University. This paper is part of a bigger research project managed by a PhD Fellow from Department of Materials and Production Mohammadali Zakeriharandi in collaboration with the company VELUX. Paper encompasses the process of data collection and processing for machine learning applications and model building. The data which has been collected on this project, together with the different programs used during the project can be accessed in the GitHub repository trough this URL:

https://github.com/ozrenv/ScrewingCell

Reading instructions

This paper is using the IEEE source guidance. This means that a number inside of a square parenthesis [number] is used when citing a source in the paper. The sources are provided at the end of the paper in the bibliography. Figures and tables hold the name of their corresponding chapter. The first Figure in chapter 2 has a number 2.1. Underneath every figure and table there is a text which holds the description. As a addition to the project, this paper has appendices, which can be found at the end of the paper.

Acknowledgements

First I would like to thank Aalborg University for accepting me on their Master's program, which was a great opportunity to further develop my education and skills. On the university I found a new passion for mechanical engineering by using new technologies across different fields of research, which I will take forward to my work life. Aalborg University is filled with great mentors and professors who are always happy and available to help students and have a passion for teaching.

A huge thanks to my supervisors Chen Li and Mohammadali Zakeriharandi for guiding me trough this project with a positive attitude and constructive guidance in the field of machine learning, robotics and mechanical engineering. Also, I would like to thank Lasse Due Hornshøj, a student assistant who helped me find my way around the screwdriving cell with his knowledge in robotics.

Trough my education I always had support from my family and friends, to whom I will be forever grateful. Special thanks to my mother Snježana, who supported me trough education and encouraged me when I needed it the most.

This Master's thesis is the final product of my education, in which I endured trough the highs and the lows. As a final remark, I dedicate this thesis to my grandmother Nina, a physicist who will be happy to see her grandson finally finishing his education.

Ozren Vučićević

1 Introduction

In today's manufacturing industry, automated screwdriving is one of the most important parts of the manufacturing process. It is used for assembly of a variety of different products, which include electronic products, automotive components, different wood products etc. Recently, the American magazine ASSEMBLY did an annual Capital Equipment Spending Survey, which included American manufacturing companies that did assembly. The results showed that automated screwdriving was included in 58% of the companies. [24]

The process of automated screwdriving involves the use of different robotic systems to pick up screws, insert them into material or pre-drilled holes and tightening them to a specific torque value. When doing this process in a larger volume efficiently, where automation of the process is a big upside, there can be problems. The screwdriving process is prone to errors and anomalies that negative impact product quality and production efficiency.

There are different reasons why anomalies arise in the automated screwdriving process. Some of them include incorrect screw insertion, stripped or cross-threaded screws, over-tightening of the screw, under-tightening of the screws, missing screws and many more. [4] All of this anomalies can create product defects, rework, stop of production and even costly recalls. Because of this, there is a growing need for effective methods to detect and mitigate anomalies in the automated screwdriving process.

Detecting anomalies in the automated screwdriving process has some promising approaches trough the use of anomaly detection techniques. Anomaly detection is a field of machine learning which focuses on pattern identification and outliers in data. Algorithms can detect deviations from normal screwdriving behaviour and alert operators to potential issues by analyzing data from the screwdriving process. Current research on the topic focuses on developing novel algorithms and techniques for anomaly detection in real-time. Some of the methods are statistical process control, machine learning and deep learning approaches. Using different sensors that can detect the screwdriving force, torque, angle in combination with this algorithms gave high levels of accuracy in anomaly detection in the screwdriving process. [4]

Currently automated screwdriving is a critical aspect of manufacturing industry that is prone to anomalies and errors. Implementation of anomaly detection techniques is proving to be a promising approach to detecting anomalies in real time and improving the quality and efficiency of the screwdriving process. Because of that, there is increasing need for additional research and development to improve the effectiveness and accuracy of these techniques.

1.1 Part of a larger project

This thesis is a part of a bigger project from Aalborg University, run by PhD Candidate Mohammadali Zakeriharandi, who is also the co-supervisor. This project called: "Data-Driven Anomaly Detection in Industrial Robotic Processes" is done in collaboration with multiple companies shown in Figure 1.1 and focuses on anomaly detection in manufacturing processes.

Figure 1.1: *Collaborators*

Project focuses on collecting different types of data from the factory floor and/or from the AAU laboratory, building machine/deep learning models, and deployment of those models. Some of the processes that this project encompasses are:

- Automated Screwdriving Process
- Arc-Welding Process
- Robotized Press-Fitting Process
- Ultrasonic Welding Process
- PU Molding Process

Project of this report is based on one of the manufacturing plant from the company VELUX, which is a international company that produces a variety of products, like windows. [26] VELUX wants to automate their manufacturing further by implementing AI solutions to their production line. More specifically regarding this report, for the screwdriving robots used to screw together wooden window frames. The company wishes to achieve a better control of the quality of their products by the use of AI solutions and avoid sending faulty windows to the costumers.

This report focuses on the task of automated screwdriving and anomaly detection of the screwdriving process. A screwdriving cell was created in the AAU laboratory to mimic the factory floor setup of the Velux manufacturing plant. Idea of the project is to collect different data types on the cell, create a new dataset, process and analyse the data and build machine learning models to detect types of anomalies that happen during the screwdriving process. The goal is to investigate how can the collection of different data types influence the accuracy of the models, and see what models work best for anomaly detection in the task of screwdriving.

2 Problem Analysis

To get a better understanding of the report problem, a problem analysis will be conducted. First, as the focus of the project is the use of AI in manufacturing, it will be introduced together with machine learning. Afterwards, the AAU screwdriving cell will be described in detail, together with all of the parts in the process of screwdriving. As a conclusion, different problems that the report will be covering will be presented as project challenges.

2.1 Introduction to AI

Artificial Intelligence (AI) is a branch of computer science that focuses on the development of intelligent machines, which are capable of performing tasks which would require human intelligence. [1] AI encompasses a lot of different areas that are shown in Figure 2.1. One of the bigger branches is called machine learning, which is the focus of this project. To speak broadly, AI's focus is in development of algorithms and computer programs that can make informed decisions based on the provided data used for learning. In this regard machine learning is particularly important, because it involves the use of data in training computer programs to identify patterns and relationship in the data. Afterwards, they use this information to take action or make predictions.

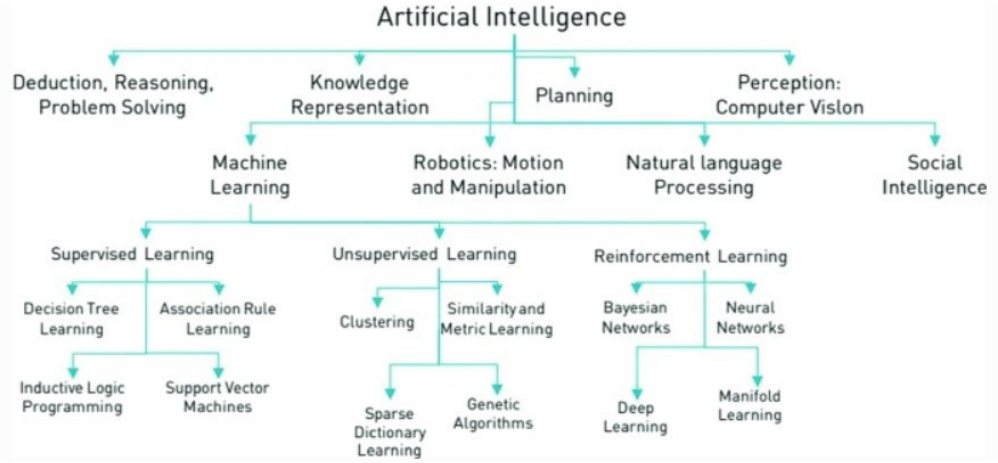


Figure 2.1: *Branches of AI [1]*

AI has the ability of automating routine and repetitive tasks, allowing humans to focus on more complex work. That is one of the reasons why VELUX wants to use AI in their manufacturing line. AI can be used to optimize their production processes, detect defects and predict equipment failures. Another use of AI is in developing intelligent agents that interact with humans in natural language and also exhibit human behaviour or emotions. The agents are usually named virtual assistants or chatbots are common in today's customer service and websites. While there are big potential benefits of the usage of AI today, there are challenges and risks related to it. To train the algorithms, you need good data sources. If the data is incomplete or problematic, there is a

potential for bias and discrimination. On the other hand, AI can be used for malicious purposes involving cyber attacks and surveillance.

With further AI development the impact on different areas of our lives will increase. Whether AI is shaping our education and work, or healthcare and transportation the difference will be noticeable. It will be important to use this technology with caution and foresight. That way its benefits will be maximized, with the minimum amount of risk.

AI in manufacturing

Short term impact in the business, with a positive impact on energy consumption and environmental aspects of the global production are one of the reasons why companies look towards new AI technologies. One of the leaders in the application of AI technology is the manufacturing sector, where introduction of AI created significant cuts in unplanned downtime and better quality of products and production. AI-powered analytic brings manufacturers new insights in to their operations and improve efficiency of their companies and safety of the employees. [3]

AI algorithms are creating market demand estimations by finding patterns, which link socioeconomic and macroeconomic factors, location, weather, consumer behaviour, political status and many more. This information is proving to be invaluable as it allows manufacturers to optimize energy consumption, staffing, inventory control and the supply of raw materials.

Introducing AI in manufacturing is providing massive leaps forward in productivity, environmental friendliness, quality of production and safety of the employees, but current research shows that only 12% of manufacturers are implementing AI. 58% of manufacturers are expressing active interest, but for different reasons decide not to. [3]

While AI holds the key to future growth and success in manufacturing, there are concerns about its ethical implications, particularly in relation to job losses and algorithmic decision-making. However, with proper regulation and ethical considerations, AI has the potential to revolutionize manufacturing and improve efficiency, product quality, and employee safety. [3]

2.1.1 Machine learning

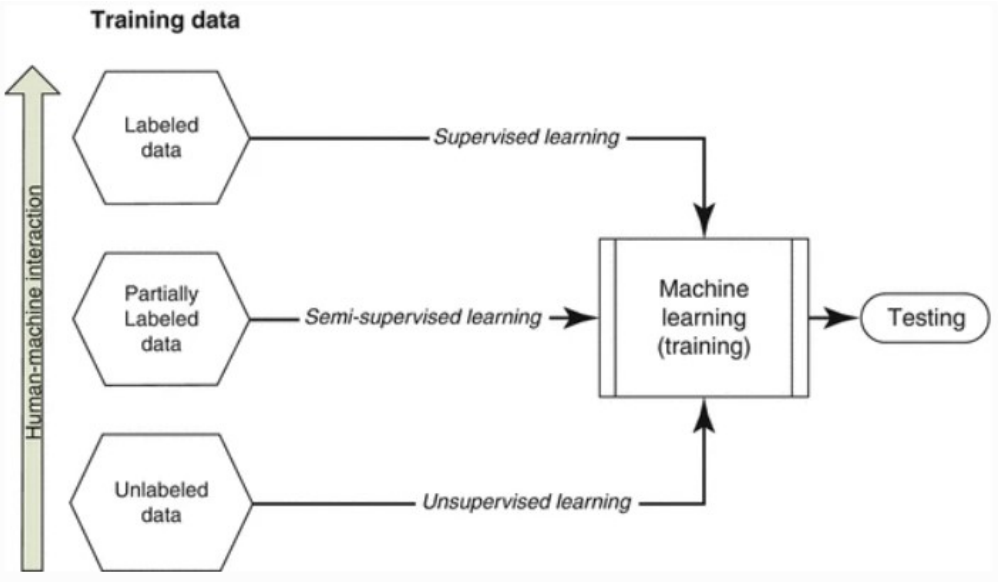


Figure 2.2: *Categories of machine learning algorithms according to training data nature [6]*

As a part of AI, Machine Learning (ML) is a way to emulate how human beings process sensory (input) signals to process data. Machine learning algorithm represents a computational process that builds its own architecture trough experience and not by being hard-coded for a specific outcome. The process of "teaching" the algorithm and providing it experience is named training. In the training input data is given together with the desired outcome, and the goal for the algorithm is to learn and optimize itself to new and unseen data in predicting the desired outcome. [6]

There are several types of ML systems. Figure 2.2 shows the division of machine learning by the type of training data provided to the training of the algorithm. The main division of machine learning is to supervised and unsupervised ML.

Supervised learning

Supervised machine learning builds a model that will make predictions based on previous data provided in training. A known set of input data and known responses to the data, also called "labeled data" from the Figure 2.2, is taken by the supervised learning algorithm to train the model. The goal of the model is to generate reasonable predictions when responding to new, unseen data. Supervised learning is the focus of this project, because there is a large amount of known data for the output which needs predictions. [11] In the case of automatic screwdriving input would be different sensor measurements, and the output or the "label" would be if the screwdriving was successful or not. This is a simple example of the "label" data, in the project different options will be explored.

Some of the techniques for building machine learning models in supervised learning are:

- Classification techniques: used for predicting discrete responses. A common example is determining if the email is genuine or spam, or if a tumor is benign or cancerous. An example of using classification in the screwdriving process would be to determine if the screw was inserted correctly or not. This technique uses input data and classifies it into different categories. If the data can be categorised, separated and tagged to fit different groups, classification is a good way to achieve that. If there are only two classes, term used is binary classification.
- Regression techniques: used for predicting continuous responses. Often used for predictions of hard-to-measure physical quantities, such as battery state of charge, load on the electricity grid or prices. These techniques are used when working with a data range or if the nature of the response is a real number. Examples of that are temperature and time until failure of different pieces of equipment.

Unsupervised learning

Opposed to the mentioned supervised, unsupervised learning analyzes unlabeled datasets which reduces the need for human involvement, also called a data-driven process. Some of the uses are to identify meaningful structures and trends, grouping results, exploration and extraction of generative features. The tasks of unsupervised learning include density estimation, dimensionality reduction, looking for associations, anomaly detection, etc. [7]

Some of the techniques for building machine learning models in unsupervised learning are:

- Cluster analysis (clustering): used for grouping and identifying data points of large datasets which are related based on similar metrics. The grouped objects are collected in categories called clusters, in a way that they are similar to other data in the same cluster by some metrics. This technique is used to find interesting correlations between the data, which would not be simple to see otherwise. That is why this technique is often used in cybersecurity, behavioral analysis, user modeling, consumer patterns, etc.
- Dimensionality Reduction and Feature Learning: common problem in data science and machine learning is processing high-dimensional data. This is why dimensionality reduction is

a important tool to lower computational costs and avoid overfitting by simplifying the models. The idea is to combine different sets of features and create new ones. Dimensionality reduction can feature selection and feature extraction, with the distinction in the act that "feature selection" keeps the subset of original features and "feature extraction" creates new features.

2.1.2 Machine learning pipeline

The machine learning pipeline of the bigger project, which this thesis is a part of consists of 3 steps, shown in Figure 2.3. The first step called "Data Acquisition" refers to the collection of data from the factory floor or the AAU screwdriving cell. Advantages of collecting the data in the AAU cell are that here the anomalies can be forcefully generated, and a better dataset can be collected. This is why it is important to have a good representation of the factory floor setup, so that the data collected can be representative of the real-world. This project only includes the data from the AAU cell, with a goal to collect data from different sensors and areas of the process.

Second step "Model building" refers to data handling, processing and machine learning model building. After the data is collected and processed, features are extracted which are used as the input to models in order to make predictions.

The last step "Model Deployment" refers to deploying the model in a real-world setup, to detect anomalies and predict the outcomes on unseen data. This representation can be used for other machine learning scenarios in general.

This thesis focuses on first two parts of the pipeline, "Data Acquisition" and "Model building".

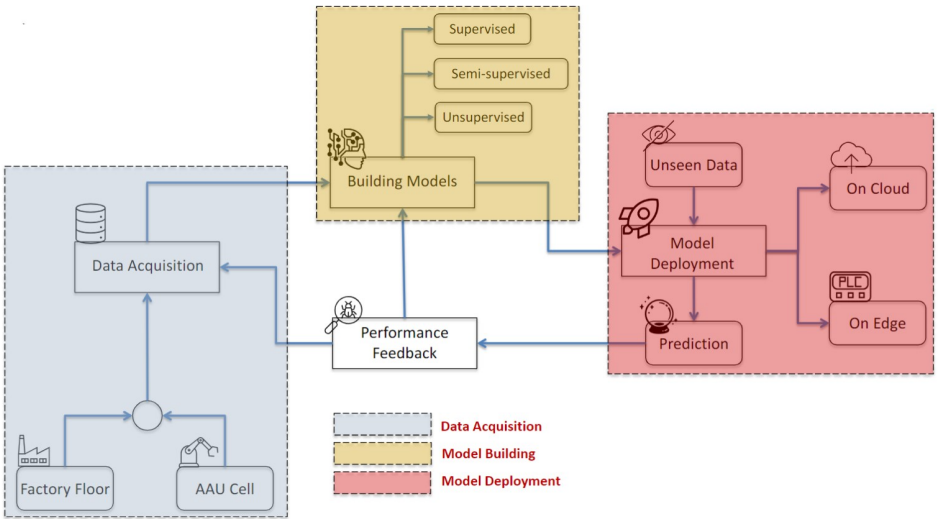


Figure 2.3: *Machine learning pipeline [10]*

Model building

The different steps of building a model will be explained later in the project, since finding out how the models are "built" is one of the purposes of this thesis. But as an introduction, different terminology of the process will be described. [17]

Target function: When dealing with predictive modeling, the goal is to model a particular process. This means that the idea is to approximate a specific, unknown function. The target function itself $(f(x) = y)$ is the function that needs to be approximated.

Hypothesis: Hypothesis is the researchers estimated function, which is presumed to be close or similar to the target function that needs to be modeled. In terms of classification, this is a classification rule proposed that will allow the algorithm to separate data. More specific a rule on which the screwdriving task is separated.

Model: Model and hypothesis are used sometimes interchangeably in the machine learning field. In other sciences a hypothesis would be an estimation by the scientist, and the model the manifestation of the hypothesis.

Learning algorithm: The purpose of the learning algorithm is to approximate or find the target function. The learning algorithm consists of a set of instructions that will try to model the target function with the use of the training data.

Hyperparameters: Hyperparameters are different parameters of a machine learning algorithm. Exampl of a hyperparameter is the regularization strenght of an L2 penalty in the loss function of logistic regression, or the number of neighbours in a KNN algorithm.

2.2 Screwdriving cell

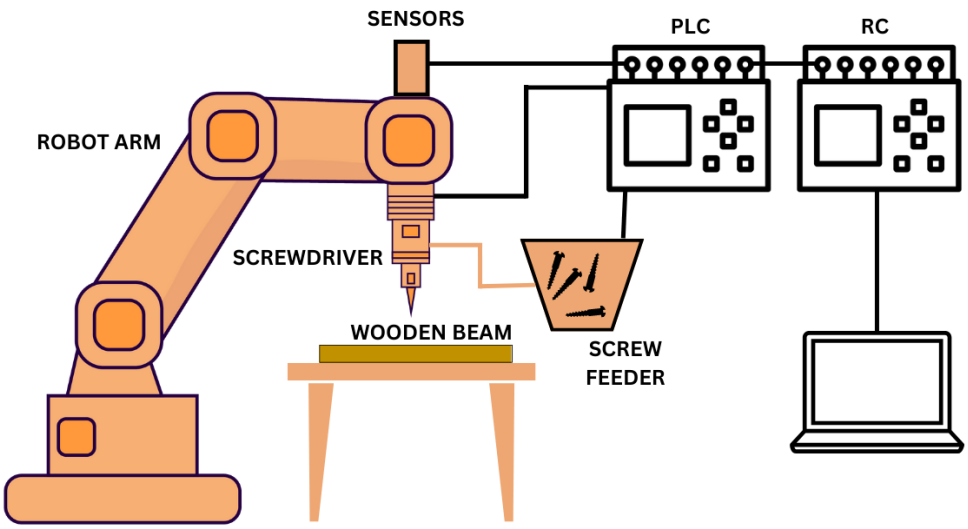


Figure 2.4: *Setup of the screw cell*

Figure 2.4 shows the current setup of the screwdriving cell as a sketch. The cell was made as a collaboration between Aalborg University and VELUX, where VELUX was responsible for providing the parts and the university of assembly.

The setup includes a robotic arm with an automatic screwdriver attached on the end effector of the robot as the tool. The automatic screwdriving system has an automatic feeder, which feeds the tool with screws trough a pressurised tube system, screwdriver controller (RC), which is in charge of the sensors, controls the screwdriver and the screw feeder. There is also a programmable logic controller (PLC) which is connected with the robotic arm, SC and the pneumatic system. The pneumatic system uses pressurised air to hold the wood test beam in place, and feeds the screws trough a tube to the screwdriver. On the screwdriver there are different measuring sensors which will be elaborated further later in the report.

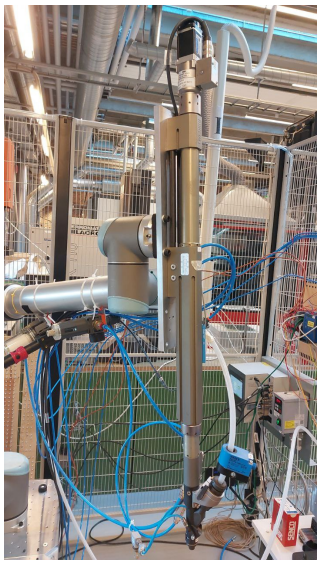
2.2.1 Robot arm

The robot arm set up in the cell is a Universal Robots robot UR10 shown in Figure 2.5. This model of the collaborative industrial robot has 6 joints and a versatile range of motion. It has a high payload (12.5 kg) and a long reach (1300 mm), making it suitable for a wide range of applications within machine fitting, palletizing and packaging. [15] In the cell the purpose of UR10 is to carry the automatic screwdriver and position it over each screwdriving point.

Figure 2.5: *Universal Robots UR10 model*

2.2.2 Automatic screwdriver

The tool of the UR10 is the automatic screwdriver. It consists of the screwdriver, screwdriver controller (C30S) and step-feeder (model ZEL). All parts were produced by company Weber. [27]



(a) *Screwdriver attached to the robot arm*

(b) *C30S controller*

(c) *Automatic feeder*

Figure 2.6: *Parts of the automatic screwdriver setup*

The screwdriver shown in Figure 2.6a is attached as the end effector of the UR10 robot and is connected to the automatic feeder, shown in Figure 2.6c, by a white tube. Trough this tube the screws are brought to the screwdriver before every screwing operation.

The screwing operation starts with the robot moving to the programmed location. The feeder then feeds the screwdriver with a screw, where the spindle moves down until the bit connects to the screw head. Inside of the spindle a vacuum is made to keep the screw in place on the bit. After the screw is secured and in position, the spindle moves to the wooden plank to the insertion point. Screw is then screwed in to the plank with a specified torque, which can be adjusted on the screwdriver controller, shown in Figure 2.6b. After the operation is done, the spindle moves back, the robot changes the position of the screwdriver and the process is repeated.

As the idea is to replicate real-world wooden frame manufacturing, experiment will include test wooden beams, which will be used for screwdriving of the screws. Figure 2.7 shows the test beams, and the pattern in which they are screwed in.

Figure 2.7: *Test beams for screwdriving*

The feeder, shown in Figure 2.8. Is used for automation of the process, by bringing the screws to the screwdriver in a regulated time intervals. That way the screwdriver always has a screw to start the screwdriving process.

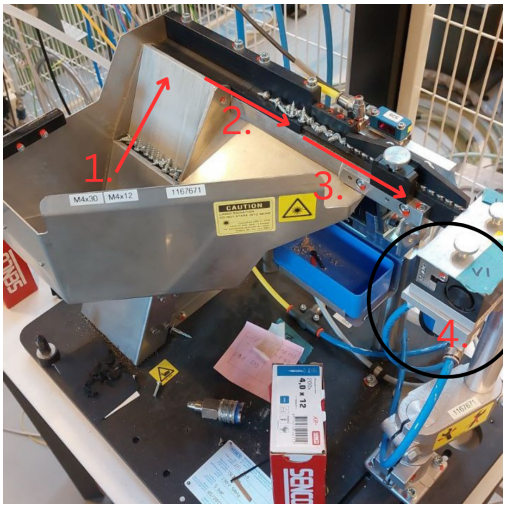


Figure 2.8: *Test beams for screwdriving*

This system has a screw holder (1.) where the screws are placed. A slider transfers the screws to the moving belt (2.). This belt transfers the screws to the alignment track (3.) This track either corrects the screws and aligns them, or blows them to the screw holder if they are not in a good

position. The screws that are aligned move to the partition unit (4.) by the use of vibration. This unit is in charge of sending the screws one by one in a correct time interval to avoid overloading of the screwdriver. The screws are transferred to the screwdriver by the use of pneumatic pressure.

2.2.3 PLC and safety features

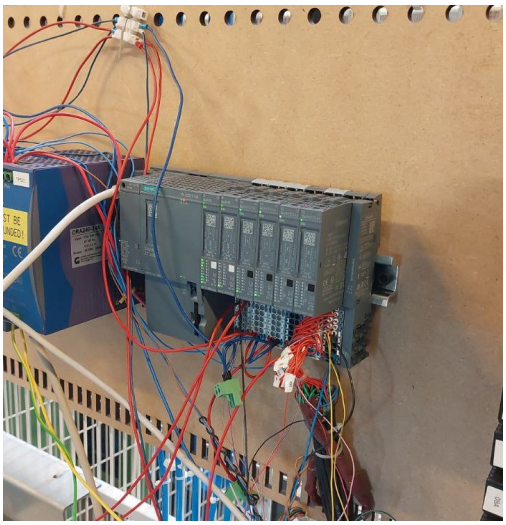


Figure 2.9: *Programmable logic controler (PLC)*

To synchronise all of the different parts in the setup, there is a need for a control system. The SC uses a programmable logic controller (PLC) for that task, shown in Figure 2.9. It is connected to every part of the process, and controls when to activate and deactivate different system components. Together with the different electrical components, it also controls the pneumatic system. As a safety precaution, there is a cage around the robot cell. When the cage opens and the contact is lost, PLC stops everything until the door is closed again.

2.2.4 Sensors

On the screwdriver there are different sensors for measuring data of the screwdriving process. This data is transmitted to the SC. Here are the sensor measurements that are currently collected over the screwdriving controller:

- Torque: The force used to screwdrive a screw into the wood
- Nset: the RPM of the spindle
- Current: the current that the screwdriver receives
- Angle: angle of the screwdriver
- Depth: how deep is the screwdriver on the Z-axes

2.3 Project challenges

This section is describing what will be the challenges in different areas of the project. That will later be used as a basis for the problem description.

Problems that this report will focus on are:

1. **Data collection and storage:**

   What will be the important data to detect anomalies in the screwdriving process? How will this data be collected, stored and managed? What are the challenges associated with collecting data in real-time from the automated screwdriving process?

2. **Data processing:**

   How will the collected data be preprocessed and cleaned to prepare it for analysis? How will outliers, missing values and noise be handled in the data? How will different data types (intrinsic, extrinsic and task data) be combined and normalised?

3. **Feature selection and engineering:**

   After data processing, what features will be extracted to later be used as input into a machine learning model? How will different features be combined and selected to maximize the models success?

4. **Model selection:**

   What machine learning model(s) will be best in detecting anomalies in the process? How will the performance be evaluated?

2.3.1 Data collection and storage

When talking about the data collection and storage, there is a lot to consider. For anomaly detection the type of data collected which will be used for the algorithm will have a big impact on the outcome. Getting the relevant data in a proper format is important to give the user easy access to the information and processing. After the metrics and data types are determined, a problem of collecting them arises. Collection of data is specific for different data types and different setups of experiments. Collection of different data types in real time can be complicated task. Dealing with system integration will be of big importance of the quality of data that will be collected.

Relevant metrics of the screwdriving process

As touched upon in the introduction, anomaly detection is one of the ways to improve the manufacturing process of automated screwdriving. To achieve this it is important to find and collect the correct data, which has direct influence on the screwdriving process and that can predict the outcome of the screwdriving quality. There are a lot of different types of data that can be collected for this problem. As an example three monitoring methods of the insertion process were proposed in Smith's paper. [23] They include the use of torque and/or the use of insertion depth.

1. *"Torque-only"* method includes the use of high-low torque band within which the signature signal is required to finish
2. *"Torque-angle"* method includes the use of the maximum and minimum torque and rotation angle (or insertion depth), where the correct scenarios are measured with experiments and compared to subsequent fastenings

3. *"Torque-rate"* method is similar to torque-angle method, where it looks into torque and angular rate signals, creating boundaries and comparing it to subsequent insertions

In more recent papers as [9], focus is put on using the data regarding the electrical current of the screwdriver in combination with torque, as a more cost effective method of determining the quality of screwdriving. It is a more cost effective method, because in the torque-current relationship there is no need for a additional sensor which will measure the angle. However, this method showed that the relationship between the torque and current is not linear in all instances.

Another option would be to analyse the screwdriving process with computer vision to detect anomalies. This would be done by implementing camera(s) that would record the process and detect anomalies by image analysis or machine/deep learning. However, by using this approach it is hard to detect screws that are under or over-tightened, with problems related to setting up the camera.

Anomaly types of the screwdriving process

When a screw is screwed in to the wood, different outcomes can occur. The screw can be screwdrived successfully, or anomalies can happen. Research paper [14], classifies events specific for their setup as:

- **Success:** Screw successfully driven into hole and tightened
- **No screw:** Failed to acquire screw
- **No hole found:** Screw acquired but never dropped into hole
- **Crossthreaded:** Screw entered hole but threads crossed so current limit hit before rundown completed
- **Stripped:** Screw successfully run down into hole but bit slip-page prevented full tightening
- **Partial:** Screw driven partly into hole but time limit reached before operation completed
- **Stripped (no engage):** Screw was so stripped it never engaged hole

In the case of the AAU cell, a different set of outcomes will be needed to describe the events of the screwdriving process. Outcomes that are not wanted, for example over and under-tightened screws, can cause delays in production, problems with quality of the end product and also a safety hazard. If the screws are under-tightened, different parts can get loose and ruin the product. If the screws are over-tightened, they can cause damage to the wood and be hard to remove which will also cause problems in production. When a whole batch of screws are defective, this can cause the stop of production for a longer period of time and create a big loss for the company. Because of that, it is important for manufacturing companies to regularly do quality control checks, which takes time and resources to do.

If an automated detection tool, that could detect the problematic screws instantly would be implemented, this would relieve the quality control part of the process, increase efficiency and quality of products.

Types of Real-World Data

The data availability is considered key in the construction of a machine learning model. Data comes in different forms and formats, notably as structured, semi-structured, unstructured and metadata. [7]

- Structured: data with a well-defined structure. This type of data is following a standard order which is organized and can be easily accessed. It comes in well-defined schemes and is usually stored in a tabular format containing names, addresses, dates, time and more.
- Unstructured: for this type of data there is no pre-defined format or organization. Because of that it is more difficult to work with, process and analyze. Best example of this are texts and different multimedia formats.
- Semi-structured: unlike the structured data, this data format is not stored in a relation database, but it contains certain organization structure, which helps in analysis. Some of examples are of semi-structured data is XML, kXML, JSON, HTML, NoSQL databases, etc. The robot controller in the screwdriving cell outputs a kXML data format, which will have to be converted to a structured format in order to work with it. Figure 2.10 shows a picture opened in VS code, from the data received by the robot controller in kXML file format.

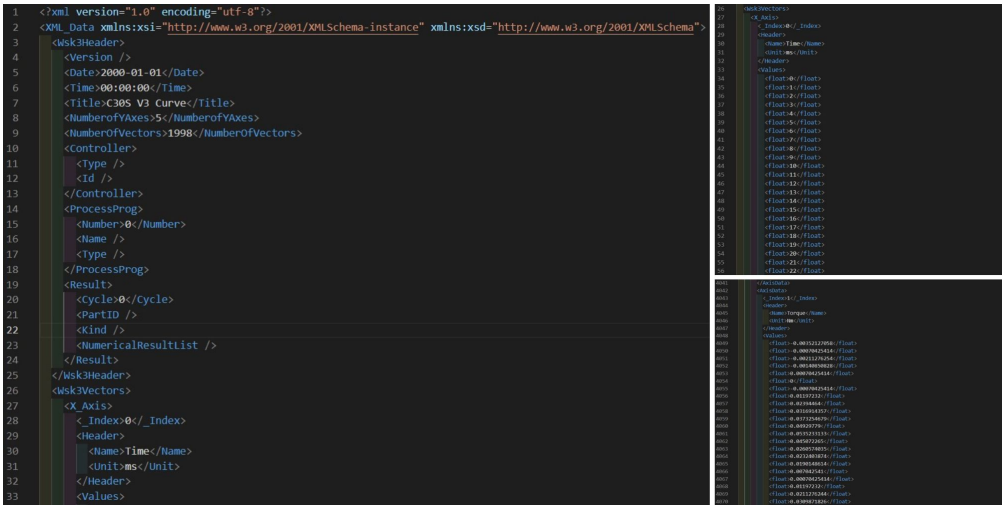


Figure 2.10: *kXML data data from the robot controller*

- Metadata: referred to as the "data about data", this data is not the same as the data types above. While the data is material that measures, classifies and documents relevant information, metadata is used to give the user better understanding of the data. The types of information in metadata would be the author, file size, date generated, etc.

Data management and storage

After the collection of the data, data needs to be properly managed and stored. Every screw has to have its proper ID for labeling and backtracking. It is important to later check which data correlates to which screw, and to properly identify it. When anomalies occur, this has to be correctly represented in the data as labels.

Data collection challenges of the screwdriving process

One of the critical steps in anomaly detection is the data collection. There are several reasons why it can be a challenge.

For starters, sensors can generate a large volume of data, which requires carefully management and processing. Additionally, noisy data can contain irrelevant information and errors related to sensor malfunctions and the environment.

Further on, choosing the correct sensors which will not be affected by factors such as vibration, temperature, outside noise, electromagnetic interference etc. All of these factors can alter the reliability and accuracy of the data. The data collected needs to be accurate, reliable, and synchronized, enabling efficient and effective anomaly detection. With the use of machine learning algorithms, the system integration can help manufacturers ensure the consistency and quality of their products by detecting anomalies.

Lastly, if the data is distributed across different devices, such as the PLC, robot and the sensors, it can be a challenge to integrate simultaneous data collection and cohesive data analysis. To do data collection correctly, there has to be a careful planning of the process and execution. This includes choosing the appropriate sensors, signal processing techniques and data integration methods.

2.3.2 Data processing

There are various steps involved in data processing and cleaning in preparation for analysis. The outliers, missing values and noise need to be addressed. The data also needs to be normalized and combined to be used as the input in to the machine learning algorithms. [16]

Preprocessing and cleaning

The data needs to be preprocessed and cleaned to achieve a reliable analysis. These are the problems that this report will face:

- Data format conversion: the output from the screwdriver controller is in a semi-structured data format, shown in Section 2.3.1. This data will need to be transformed to a structured format in order to work with them
- Removing the non-relevant features: if the data contains features that are not relevant, they have to be removed
- Handling missing values and outliers: if the data contains missing values, they need to be removed or filled with different values. Outliers can be removed by the use of different techniques like Z-score or IQR
- Removing noise: noise in the data can happen because of different sensor errors and fluctuations in the data. If sound is recorded, a good approach would be to filter the outside noise when handling the files

Combining and Normalizing Different Data Types

When collecting data from different data sources, especially data that is collected in a time series, it is important to correlate. This can be done in the data collection phase, by recording the data in the same time. If there is a misalignment, to properly combine the data later it needs to be corrected.

In terms of data normalization it is important to note that different data come in different unit measurements, so they will need to be transformed and scaled to fit together. This is done in order to ensure that one feature does not dominate the analysis. One of the ways to do that is to scale every feature to have a mean of 0 and standard deviation of 1. This can be done with the use of different techniques, like Min-Max scaling.

2.3.3 Feature selection and engineering

Different definitions of feature selection exist. On of them is provided by Hall: "Feature subset selection is a process of identifying and removing as much irrelevant and redundant information as possible." [16] Based on this definition the goal of feature selection is to gather a subset of features that will represent the problem appropriately. In this case features that will describe the screwdriving process and anomalies that happen in it. Some of the benifits of feature selection are:

- More accurate models: features that are redundant and irrelevant can cause accidental correlations in the algorithm, which will result in poor generalization capabilites. Removing this features will improve the models.
- Reduction of the search space: all features together make the search space of an algorithm, which it explores. With the use of dimensionality reduction, mentioned in Section 2.1, the research space is reduced, and a quicker learning process is achieved
- Saving storage: without the unnecessary features, the storage requirements are reduced. In this project, this will not be a problem
- Reduction of costs: this step is important for the data collection task. If some features are not important, this will lead to reduction in the need for sensors, workers and time consumption, which will lead to the reduction of cost
- Better understanding and visualization of data: with less features the models are simpler and with them results are easier to percieve
- Decrease of the over-fitting risk: the problem of over-fitting occurs when the ML algorithm adjusts to the training data too well, and achieves bad results on unseen cases

2.3.4 Model selection

When talking about the model selection, we need to first answer the question: "How do we estimate the performance of a machine learning model?" A simplified answer would be to collect and process the data. The data is then fed to a machine learning model and the labels of our test set are predicted. The correct and wrong predictions are counted and the accuracy of the model is determined. In reality this process has a lot of different sub-steps. To select the best option for the given hypothesis space, and determining the appropriate algorithms to use can be a challenging task which requires experimentation.

Using an ML algorithm with different hyperparameters will result in a different model, with different accuracy scores. Also, the usual approach is not to use only one "best" model, but more of them and compare their results and computational expenses.

The main points in evaluating the performance of the model [17]:

1. Estimation of the generalization performance, or the predictive performance of our model on future (unseen) data

2. Increasing the predictive performance of the model with changing the hyperparameters of the learning algorithm and selecting the model with the best performance in the given hypothesis space
3. Identifying the algorithm that is "best" for the problem. This requires comparison of different algorithms, selection of the algorithms by performance and selection of the best performing model

3 Problem Definition

After the problem analysis, where the setup of the screwdriving cell was presented, together with different problems and challenges that the report will face, the problem statement will be described. This will include a summary of the problem analysis, the main hypothesis of the report, the requirements of the project and the limitations. To create introduction to the solution phases of the project, as the final part of the problem definition, section called project solution workflow will show in what way will the problem be solved, together with the different solution phases.

3.1 Summary of the problem analysis

This thesis is made as a part of a bigger project which uses data from the AAU smart lab and data from the factory floor in hopes to improve the manufacturing process of a company. In this case AI tools are used as a anomaly detection tool, and it's goal is to detect irregularities and mistakes in the screwdriving process, to ensure better quality control of the products and improve efficiency of the manufacturing process. A screwdriving cell was created in the university's laboratory, in hopes to force the gathering of anomaly data which will be later used for machine learning purposes. The report deals with the data collection and machine learning model building, but not with the deployment of the model. The goal of the project is to collect a new dataset containing different available process data, build machine learning models and compare them.

3.2 Problem statement

Promising approach to detect anomalies in automated screwdriving is trough the use of various data types, which will in this case be collected on the robot screwing cell during the experiments. This data types will include information from the sensors on the screwdriver which is considered as intrinsic process data. There will also be an effort, to collect additional data that can be collected. Approach to detect anomalies will be trough the use of classification with machine learning. To start of with the problem statement, a hypothesis is made:

„By utilizing machine learning on the different process data of the automated screwdriving process it is possible to effectively identify deviations, irregularities in the performance of the system, and use this information as a reliable anomaly detection tool."

This statement will try to be answered by conducting a comprehensive analysis of the different data types collected during the screwdriving process and developing different machine learning models. The models will be compared by results that will be gathered. To achieve this goal, different research objectives have to be achieved.

3.3 Requirements

Table 3.1 will present the main research objectives and tasks included in each one of them:

Research objectives and related requirement specifications	
RO1: Design and implement a data collection and storage system for the robot screwing cell and create a new dataset which will be used for ML model training	
· find what process data can be collected (intrinsic, extrinsic, task) and decide on which will be collected	
· decide on the anomaly types that will be looked into	
· create a data collection system which will:	· collect different types of data · collect the data simultaneously · collect the data in a proper format · collect the data automatically
· create a labeling and backtracking structure for the data gathered which will:	· have a systematic division of data location · sort the data by the screwdriving type (or anomaly) · create a system for backtracking the data · make a system where every screw has an ID, together with its data
· create a new dataset by conducting a series of experiments, that will be used as the basis for training of the machine learning models	
RO2: Preprocess and clean the data collected	
· convert the kXML files to a more manageable data format	
· find a way to combine different data types with different sampling frequencies and delays in the data collection process	
· remove the unnecessary data from the dataset	
· remove the noise from the data	
· handle missing values from the data	
· visualise the data	
RO3: From the data collected, extract the relevant features and make different dataframes which will be used for ML.	
· from intrinsic, extrinsic and task data, extract the features relevant to the process	
· create different combinations of features and find the ones which give best results in the ML algorithms	
· label the data	
RO4: Develop a classification ML/DL models using the data from the screwing cell	
· decide on the algorithms that will be used in classification	
· create a system for measuring the results of different ML models	
· test separately intrinsic extrinsic and task data and compare the accuracy results of classification with different ML algorithms	
· test different combinations of features, and find the best "fit" for classification	
· create a deep learning model with separate or combined data types and measure the results gathered	

Table 3.1: *Project requirements*

3.3.1 Detailed description of the requirements

RO1: Design and implement a data collection and storage system for the robot screwing cell and create a new dataset which will be used for ML model training

Idea is to create a system for gathering the relevant intrinsic, extrinsic and task data from the cell, that will be used later to train machine learning models. Goal is to create a new dataset for analysis and training.

-find what process data can be collected (intrinsic, extrinsic, task) and decide on which will be collected

There is a large amount of data that can be collected during the screwdriving process. It is important to find out what data can describe the process best and be used for ML purposes. The data will be divided into intrinsic, extrinsic, task data and explained later.

-decide on the anomaly types that will be looked into

To use supervised machine learning, we need to label the data. In this case the labels will be different anomalies that happen during the screwdriving process.

-create a data collection system

To collect the data from different sources of the robot screwdriving cell, a data collection procedure has to be configured, to collect the data simultaneously in the proper format. It is important to find a way of integrating different parts of the system for a structured and smooth data collection.

-create a labeling and backtracking structure for the data gathering

Data from every screw has to have a unique ID and location of storage. It also has to later be properly labeled and backtracked. This will help if there will be errors in the data, so that it can later be checked and corrected if there will be errors.

-create a new dataset by conducting a series of experiments, that will be used as the basis for training of the machine learning models

The end product of the data collection will be a new dataset, which will be properly structured, labeled and stored.

RO2: Preprocess and clean the data collected

To achieve a reliable analysis the data needs to be cleaned and preprocessed and converted to the right format. The key in further data analysis will be combining the data with different sampling frequencies and delays.

-convert the kXML files to a more manageable data format

As mentioned in Section 2.3.1, the data from the screwdriver controller comes in a kXML file format, which is a semi-structured format. This data needs to be converted to a proper format for further analysis.

-find a way to combine different data types with different sampling frequencies and delays in the data collection process

After the data is collected, if there will be differences between the data in terms of sampling frequency or delays, there will be problems in combining the data for the machine learning. Ways to mitigate this will be explored.

-remove the unnecessary data from the dataset

If there is data in the dataset which is not needed for the training of the machine learning algorithms, it will be removed.

-remove the noise from the data

Noisy data can create problems in the performance of a model, so it will be processed and

removed. Good example of that is background noise of a sound recording.

-handle missing values from the data

If the datasets are missing values, or differ in length, the missing values have to be filled or removed.

-visualise the data

In order to better understand the data presented, visualisation of the data will be presented.

RO3: From the data collected, extract the relevant features and make different dataframes which will be used for ML

Depending on the type of ML model, extract the features necessary to achieve effective results in the models.

-from intrinsic, extrinsic and task data, extract the features relevant to the process

Different sensor measurements collect raw process data. To be able to use classification, features need to be extracted from some of the measures. This features need to represent the process of screwdriving effectively.

-create different combinations of features and find the ones which give best results in the ML algorithms

The choice of features and data that will be used for training of the ML models will effect how the models perform. With experimentation the goal will be to find the best combinations of data and features to achieve the most accurate models.

-label the data

To train the model, the dataset which is used for training has to be labeled correctly.

RO4: Develop a classification ML/DL models using the data from the screwing cell

After the data collection, the idea is to create a set of machine learning algorithms, use them for classification of anomalies and compare the results.

-decide on the algorithms that will be used in classification

When building the model, the choice of the algorithms will be important. Based on the data and purpose, multiple ML algorithms will be trained, tested and compared based on the performance.

-create a system for measuring the results of different ML models

After the models are trained and tested in classification of the anomalies, a system on which they will be compared has to be devised. This will help in determining which metrics are important and which model performes "best".

-test separate intrinsic, extrinsic and task data and compare the accuracy results of classification with different ML algorithms

The data comes from 3 different sources. In order to determine which source is better for classification purposes, ML models will be trained with separate data, and compared based on the performance.

-test different combinations of features, and find the best "fit" for classification

Will combination of different features or sources of data yield with better model accuracy?

-create a deep learning model with separate or combined data types and measure the results gathered

After testing different supervised machine learning models, an effort will be put into creating a deep learning model which will take into account the time-series data from the screwdriving operation.

3.4 Limitations

Do better describe the scope of the project, together with the different constraints which occurred, a section will be dedicated to explaining the limitations of the process.

Screwdriver controller

The setup of the screwdriving cell was made with the help of companies, outside of university by experts. While some modifications to the cell were made, focusing more on the software side of the process, there are some constraints. One of the bigger ones is that the sensory data, which is monitoring different sensors on the automatic screwdriver is transmitted to the screwdriver controller. The only way currently to receive this data with a device is trough a USB cable using a program from the company Weber called WSK3, which is a software for screwdriving graphs. Without the knowledge of the complicated setup which includes the controller and the PLC, idea was to find a different path in data collection from different sources, which comes with a price of a small delay in the different data collected.

UR10 robot

The robotic arm used on this project, referenced in Section 2.2.1, is an older version of the robot with an older operating system CB2. This will limit the data that can be gathered from it, together with the speed of communication between devices. This means that the data collected from the robot will be recorded in a smaller sampling frequency then the rest of the data. Also, this limited the amount of different measurements that can be recorded from the robot.

Audio recording

The location of the screwdriving cell in the AAU Smart Lab is next to the 5G internet servers, which produce noise. Also, the pneumatic system of the cell produces a lot of noise created by the air inside of the tubes and the screwdriver. This can create problems when recording sound with the microphone which is not specified for industrial use. Normally in the production plant there will be outside noise, so one of the problems report will face is how to deal with this noise.

Available wood for the dataset creation

The screwdriving cell operates with specific type of wooden beams, and the lab assistant adjusted the robot to operate based on the specific wood frames. There is a finite amount of wood left for dataset creation and testing, and implementation of new wood will require additional time and resources. Implementation of new wood will require time and also it could influence the dataset, since the new wood could have different specifications then the old one. For now, the plan is to create the dataset on the old wood frames.

3.5 Project solution workflow

As the transition from the problem analysis and definition to the solution phase of the project, this chapter will give an overview of the different operations and parts in the solution of the project. The solution of the project will come in 3 different phases:

A) Data collection

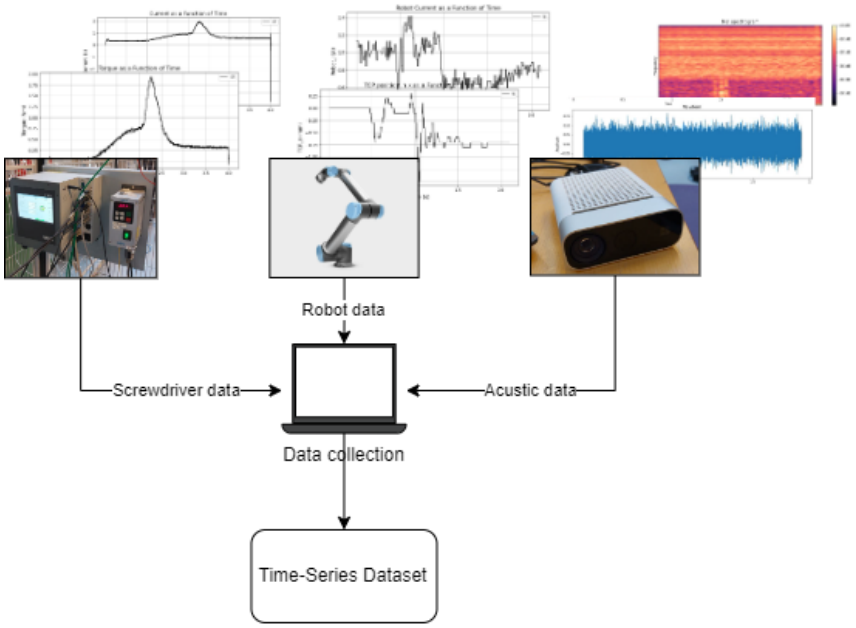


Figure 3.1: *Data collection phase*

The first phase builds a data collection system which will create a time-series dataset. It will integrate data from different sources, and create a dataset which will be used in classification. This part of the project will answer questions from the first research objective (RO1).

B) Preprocessing and feature selection

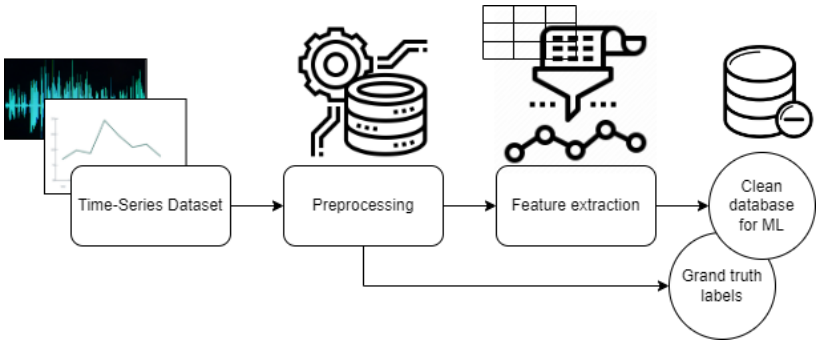


Figure 3.2: *Preprocessing and feature selection phase*

Second phase will deal with preprocessing of the data and feature extraction, which will come in raw sensory format and audio format, to achieve research objectives RO2 and RO3. The goal of this faze is to process and adjust the time-series dataset received from the data collection faze and prepare it for ML applications.

C) Model building and results

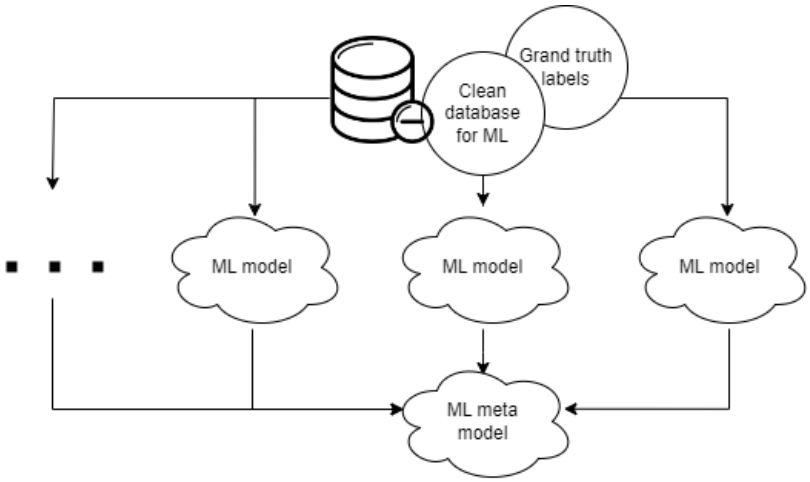


Figure 3.3: *Model building phase*

The final phase of the solution will involve using the clean database in building machine learning models. Different data sources will be used in models, and afterwards the data sources will be combined to create meta ML models. Model building process will be explained together with thee results of the models. This will answer the questions of the final research objective (RO4).

After the solution phase, there will be discussion based on the results of the research. The discussion will include the overview of the project research objectives.

4 Data collection

As the first part of the solution process, this chapter will explain the different data sources, define the test parameters of the experiment and explain the data collection procedure with all of the specifications. As the final product of the data collection step, the final dataset will be presented.

The data collection procedure includes the process of collection different data types, sorting and storage, by creating a program which receives information and creates dataframes and audio files. The data will be collected from 3 different sources. Afterwards, a ID and backtracking system will be implemented and explained in the report. Every screw will have a unique ID, and every data source will also be properly labeled. The data will be stored based on the screw type. Figure 4.1 shows the overview of the process.

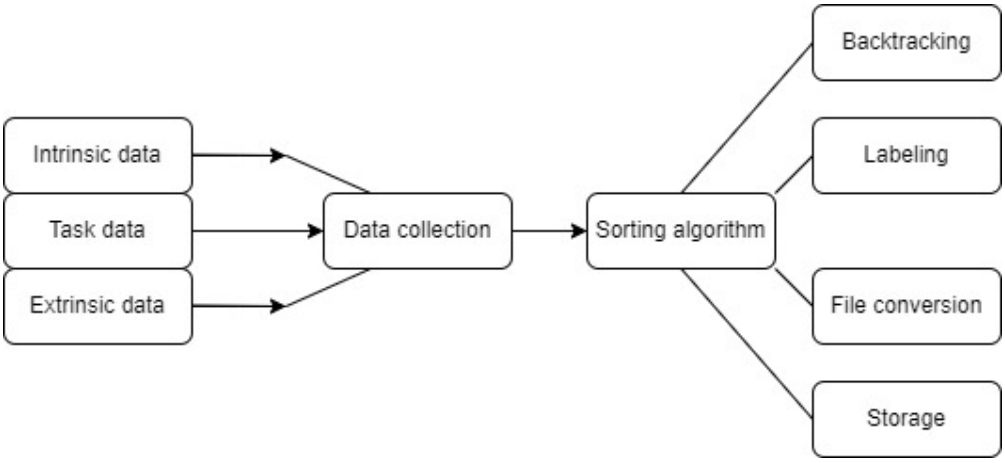


Figure 4.1: *Diagram of the data collection workflow*

The data collection will happen in two fazes. The first will be collection of data in the raw format from different sources. Afterwards, an algorithm will be made that will make sure to convert the data in the required format for later use, properly store the data, label the data and enable backtracking.

The end result of the data collection will be a full dataset containing the 3 data sources which will be visualised and analysed. The code for the data analysis will be provided in a gitub repository.

4.1 Test parameters

Before going into the data collection procedure different parameters of the screwdriving operation must be defined. It is important to know which screw types will be analysed, and what anomalies can occur. Different failure types will be described, together with explanations on how this modes can be artificially produced.

4.1.1 Screw types

This was not the first project on the screwdriving cell, so there was multiple examples of the screws in the wood beams. Based on them and scientific literature [14], different types of screwdriving have been determined. There is more classes of anomalies that could be looked into, but this list is based on the size of the dataset. To go deeper in classification of the screwdriving task, preferably we would have a larger amount of data.

To get an overview of the screw types that will be selected for classification a Table 4.1 shows the names of the different types and the abbreviations of them.

Screw type	Abbreviation
Normal screw	N
Over-Tightened screw	OT
Under-Tightened screw	UT
Pose anomaly	P
No-screw	NS

Table 4.1: *Table of screw types*

Normal screwdriving

In the process of normal screwdriving, the screw head comes directly in contact with the wood, the threading stays intact, and the screw position is aligned with the axis perpendicular to the wood's surface.

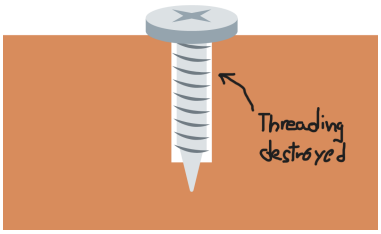
(a) *Normal screw image* **(b)** *Normal screw sketch*

Figure 4.2: ***Normal screw***

Over-Tightening

In the process of over-tightening, the screw head does come directly in contact with the wood and sometimes brake the wood surface, the threading is broken, and the screw position is aligned with the axis perpendicular to the wood's surface. The broken threading means that the screw is lose inside of the hole for the torque momentum.

(a) *Over-Tightened screw image*



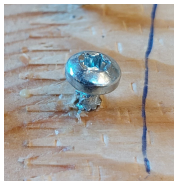
(b) *Over-Tightened screw sketch*

Figure 4.3: ***Over-Tightened screw***

Under-Tightening

In the process of under-tightening, the screw head does not come directly in contact with the wood while the threading stays intact. A misalignment can happen, but if the screw was screwed in it is also considered a under-tightened screw as shown in the following Figures 4.4.

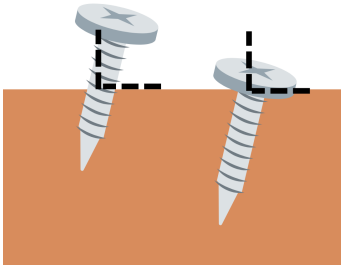
(a) *Under-Tightened screw image 1.*



(b) *Under-Tightened screw image 2.*

(c) *Under-Tightened screw image 3.*

(d) *Under-Tightened screw sketch 1.*



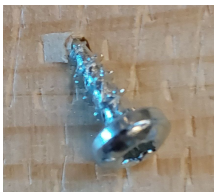
(e) *Under-Tightened screw sketch 2.*

Figure 4.4: ***Under-Tightened screw***

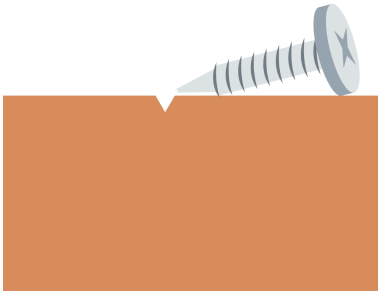
Pose anomaly

When the screw "slips" during the screwdriving process it is classified as pose anomaly. This means that the screw penetrated the wood, but did not penetrate significantly and was left on the surface of the wood, causing the drill bit to go out of position.

(a) *Pose anomaly image 1.*



(b) *Pose anomaly image 2.*



(c) *Pose anomaly sketch*

Figure 4.5: ***Pose anomaly***

No-screw

When after screwdriving there is no-screw, it means that the screw was not part of the screwdriving operation and did not penetrate the wood. The wood will not have marks on it. This often occurs when the feeder gets jammed, and no screws gets fed into the screwdriver.

(a) *No-screw image*

(b) *No-screw sketch*

Figure 4.6: ***No-screw***

4.1.2 Settings of the screwdriver controller

To achieve different screw types and artificially create anomalies, the screwdriver controller (C30S) was set up with different values of torque. There were different tests done on the screwdriving cell, from which these torque values were established and explained by the lab assistant. The controller is set up in 3 different screwdriving modes, also shown in Table 4.2:

Screwdriving type	Normal	Under-Tightening	Over-Tightening
Torque settings	1.1 Nm	0.5 Nm	2 Nm

Table 4.2: *Screwdriving modes*

These modes differ in the torque value that the screwdriver will produce during the screwdriving process. From these 3 modes, all of the screwdriving types can be achieved. It is also worth

mentioning that for screw tipe where the screw is stuck in the feeder, there is no need for one of these modes. The Out-of-Plane screw can not be generated forcefully, but can happen in any of these 3 modes.

4.2 Data sources

Figure 4.7 shows different sources of data that were used in the experiment. They are intrinsic, task and extrinsic sources.

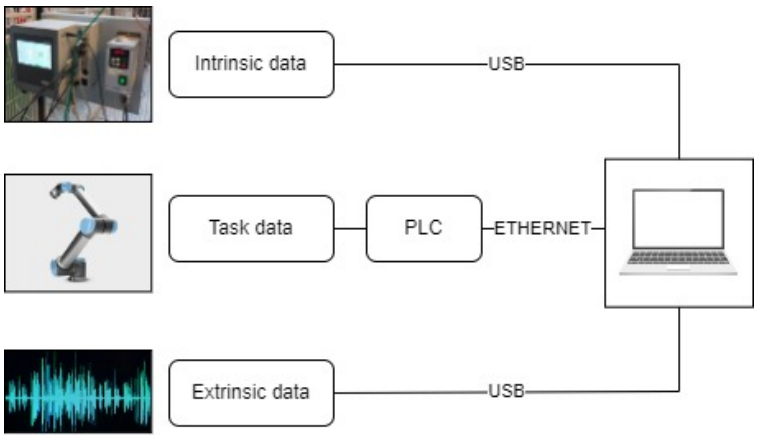


Figure 4.7: *Diagram of the data collection workflow*

Intrinsic data

Intrinsic data sources refer to the inherent information withing the process. In this case the different sensor measurement of the automatic screwdriver are considered intrinsic or native. This data should yield with the best basis for the analysis, because it is the most accurate, precise and derives from the process itself. With intrinsic data the potential of biases or external influences is minimized.

Intrinsic data comes from sensors which send the data to the screwdriver controller (C30S). This data is then sent with a USB cable to the laptop with the help of the WSK3 program. The sensors record 5 different measurements in a time-series format. Table 4.3 shows those measurements with units:

Time	Nset	Torque	Current	Angle	Depth
(ms)	(1/min)	(Nm)	(V)	(rad)	(mm)

Table 4.3: *Screwdriver sensor measurements*

The Nset values represent the rotation of the screwdriver per minute, the Torque represents the torque value of the screwdriver in Newton meters, the Current represents the current to the screwdriver in Volts, the Angle represents the rotation angle of the screwdriver which is used to control the precision and accuracy of the torque applied by the screwdriver during tightening or loosening operations in radians and the Depth value which represents the depth of the screwdriver in millimeters.

Task data

Task data refers to the data received from the robotic arm (UR10), that holds the automatic screwdriver. As mentioned in the Section 3.4 there were limitations with the amount of data that that can be collected. After consultation with the supervisors, it was decided to prioritise the

sampling frequency over the number of measurements that will be collected. Table 4.4 shows the measurements of the task data with the units:

Time	TCP_x	TCP_y	TCP_z	TCP_rx	TCP_ry	TCP_rz	Robot_I
(ms)	(mm)	(mm)	(mm)	(rad)	(rad)	(rad)	(A)

Table 4.4: *Robot data measurements*

The TCP stands for Tool Center Point. The TCP is a reference point on the tool or end-effector that the robot uses to perform tasks. Te TCP_x, TCP_y and TCP_z represent the location in the 3D robot base coordinate system in millimeters and the TCP_rx, TCP_ry, and TCP_rz represent the rotation components of the Tool Center Point with respect to the robot base coordinate system in radians. Robot_I is the current that the robot is receiving in Ampers.

Extrinsic data

As the extrinsic data source, it was decided to record sound of the screwdriving process. There were different options which included using images and image recognition and recording the vibrations of the process, but with discussions with the sensors and available recording devices, it was decided to record sound. The microphone of the Azure Kinect DK was selected as the device for recording sound, shown in Figure 4.8.

Figure 4.8: *Azure Kinect DK*

Azure Kinect DK is a spatial computing developer kit with sophisticated computer vision and speech models, advanced AI sensors, and a range of powerful SDKs that can be connected to Azure cognitive services. [2] In case of this project this was the best microphone available.

4.3 Data collection procedure

Before going in to the procedure of the data collection the setup of the experiment will be explained. First, the parts of the setup are:

- The robot screwdriving cell
- Laptop with the data collection program
- Wood and screws
- Azure Kinect DK microphone

- Automatic feeder and screwdriver

After consulting the supervisors, and also with limitations on the amount of wood available, it was decided to collect at least information about 1000 screws in total. This data should include the 3 different modes mentioned in Section 4.1.2.

The data collection happens in two phases:

1. **Data collection phase**
2. **Data sorting phase**

This process is divided in two phases based on the implementation of the data collection. First the data is recorded in the data collection phase and then the data is sorted in the data sorting phase.

4.3.1 Data collection phase

Data was collected from 3 sources: robot, screwdriver and microphone. Task data, which is the data from the UR10 robot, and extrinsic data, which is recorded by the microphone data, is received by the use of a python script.

Screwdriver (intrinsic) data

The data from the screwdriver comes with a USB cable from the screwdriver controller directly to the laptop. As explained earlier, it comes trough a program called WSK3, shown in the following Figure 4.9.

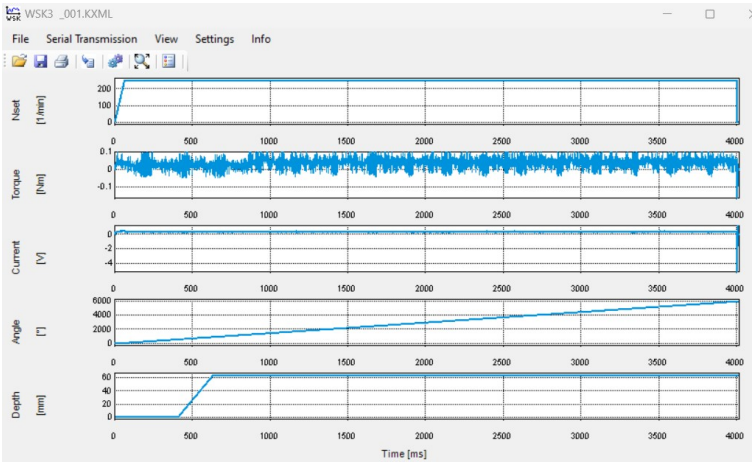


Figure 4.9: *WSK3 program recieving data*

When the program is started, the screwdriver controller automatically sends the data to the laptop for each screw separately, and saves them in the .KXML file format as _000, _001, _002 and so on. This process happens separately from the data collection of the robot and the microphone. This data is named intrinsic process data.

UR10 (task) data

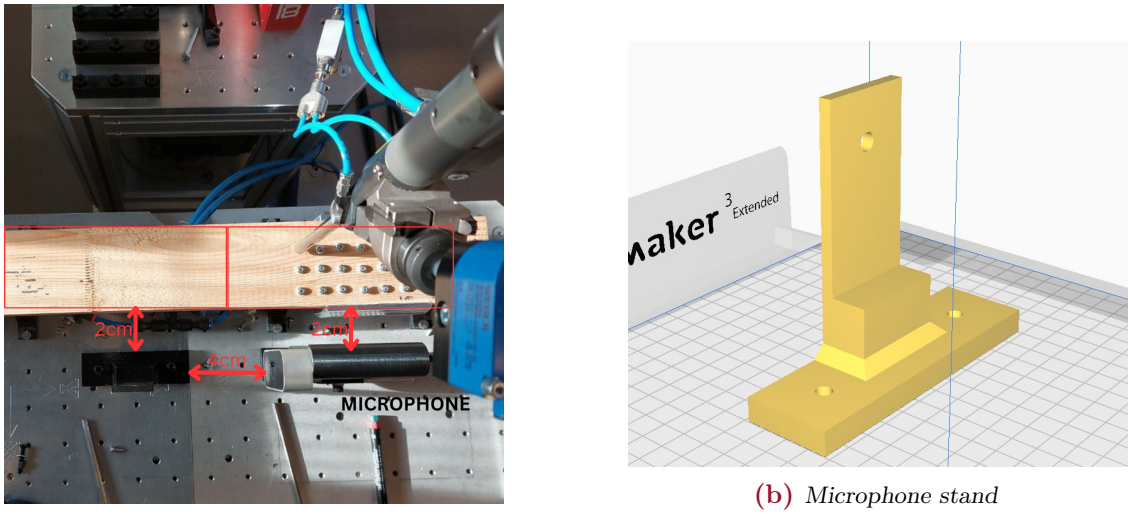
The UR10 robot was connected to the laptop trough Modbus. Modbus is a serial communication protocol, which was developed by Mdoicon in 1979. for use with its programmable logic controllers (PLCs). As a description, Modbus is a method for transmitting information trough serial lines between different electronic devices. There is Modbus Client, device which requests the information and Modbus Server, device which supplies information. Usually there is one Modbus Client and

multiple Modbus Servers. The purpose of Modbus is to transmit signals and data from control devices and instrumentation back to the main controller or data gathering system, in this case from the UR10 robotic arm to the Laptop. [5]

The data is stored in a .csv format, with the information mentioned in Table 4.4. It is stored in tabular format, with a time mark for every data point.

Microphone (extrinsic) data

Sound information from the screwdriving operations is received by the microphone. To achieve a more standardised, fixed position, frames for the microphone were 3d printed and screwed in to the table. Because the robot already had troubles with handling the weight of the automatic screwdriver, it was decided that fixing the microphone to the tool would not be feasible, even if this would be the most standardised location for it. Figure 4.10 shows the 3d model of the microphone frame, and the microphone setup.



(a) *Setup of the microphone*

(b) *Microphone stand*

Figure 4.10: *Microphone setup*

In Figure 4.10a, the microphone is on the right stand. As the screwdriving operation is moving the microphone is recording the process. After the screws enter the zone of the next stand, the microphone is removed from the first one and transferred to the next one. That way the distance between the screw and the microphone is relatively consistent.

The data is stored in a .wav format.

Python data collection script

At first, recording the data from the robot and microphone together presented a problem, because the sampling frequency was low (data from the robot was received at a sampling frequency of 100Hz). This was happening because the program was using a lot of resources to read the different register values with the Modbus protocol. To mitigate this, threading was introduced.

Threading in Python is a technique that creates multiple threads of execution, which run concurrently together within a single process. The threads are lightweight and independent units of execution that share the same memory space. This allows the script to multitask and improves it's performance. [13]

In the context of the project, threading was used to read the data from the registers and data from the microphone separately from the main process of data collection. By separating these two tasks, both operations were performed in parallel, without one process blocking another. After threading was introduced, the sampling frequency of the UR10 was consistent, at around 400Hz.

The script reads the Modbus register values from the PLC and records the sound from the microphone. The register values are saved in a Pandas dataframe, and stored to a .csv file. The values that are recorded are shown in Table 4.4. The task dataframes and sound recordings are saved after every screwdriving process.

To be able to synchronise collecting the data from different sources, a signal from the PLC was used. In the PLC programming, a function was inputted which is sending notifications on the status of the screwdriving process. A Boolean value of True is set when the screwdriving is happening and False when it is not. This location of the strobe signal is shown in the following Figure 4.11.

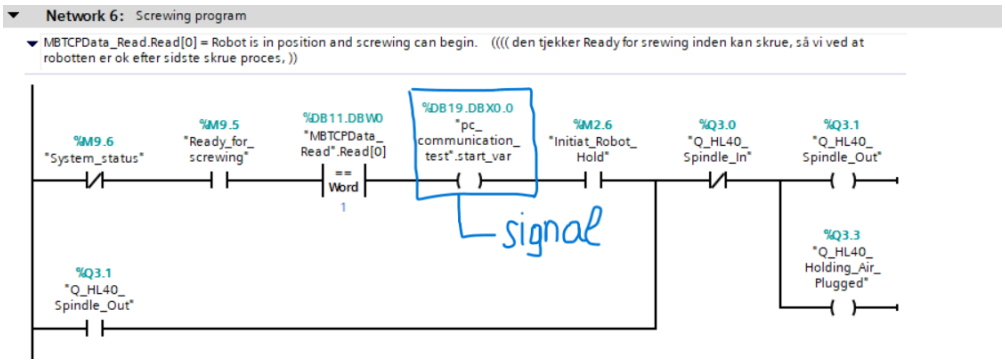


Figure 4.11: *Strobe signal in the PLC*

The screwdriver starts the operation with the instructions from the PLC. With this signal, together with the intrinsic data, the robot data and the microphone data were recorded in unison.

4.3.2 Data sorting phase

After the data collection phase, where the laptop was connected to the different data sources and recorded the relevant process data, a second phase called data sorting starts. This phase includes the inspection of screws, storage of the data and labeling.

After the screwdriving is done and the data is collected, the operator comes to the test wood beam and inspects it. By a simple visual inspection it is clear which screw belongs to which category.

Over-tightened screws have to be checked with the help of a screwdriver, because the broken thread is not visible with the visual inspection. If the screw does not resist when screwing it in, the threading is broken.

Labeling, backtracking and storage

As mentioned in Section 2.3.1, it is important to be able to identify each individual screw, with the information regarding it (metadata). Intrinsic, task and extrinsic data is recorded for each screw separately. To be able to keep track of the data related to the process, the data about every screw has a unique ID number. This was done to later work with this data and to enable backtracking of the data in case of mistakes. Figure 4.12 shows how was a unique ID made for every screw.

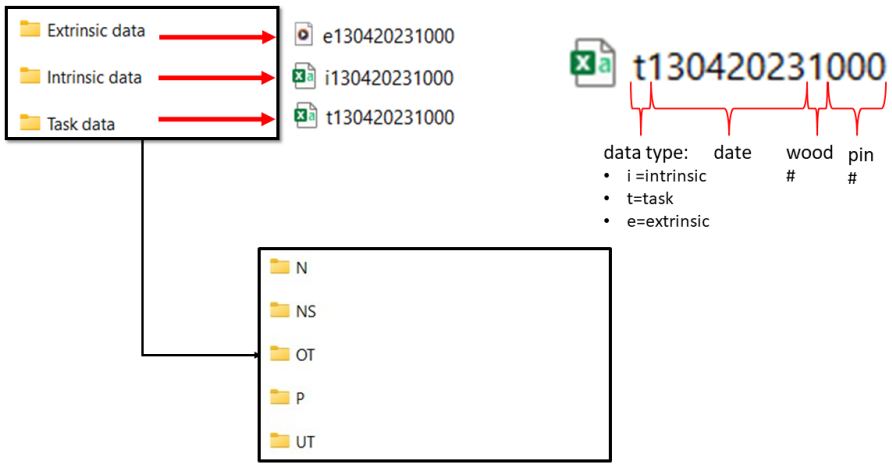


Figure 4.12: *Data ID and storage*

Every data source has a folder, where it is saved. Every folder has sub folders which represent the anomaly class of the screwdriving, and they are named with the abbreviations of the classes. The sorting algorithm saves the appropriate screw data to the class folders.

The screw ID contains a letter which describes the data type, a date, number of the wood and number of the pin. Later, when the data will be inputted to ML models, this ID will be correlated to the specific class folder as the label or "grand truth", mentioned in Section 2.1.2.

Python data sorting script

```
Type number of missing screws:6
Type the number of no-screws:3
Type number of out-of-plane screws:6
Type number of under-tightened screws:0
Type number of normal screws:0
Enter mising pins #1: 005
Enter mising pins #2: 022
Enter mising pins #3: 054
Enter mising pins #4: 059
Enter mising pins #5: 061
Enter mising pins #6: 079
Enter no pins #1: 073
Enter no pins #2: 074
Enter no pins #3: 075
Enter bent pins #1: 008
Enter bent pins #2: 017
Enter bent pins #3: 045
Enter bent pins #4: 053
Enter bent pins #5: 066
Enter bent pins #6: 077
```

Figure 4.13: *Sorting program input*

When running a python script to sort the data, input is the wood number, and the classes, shown in Figure 4.13. After the screwdriving on the whole piece of wood is finished, the operator checks the wood and the screws and writes down the pins with anomalies. This is used as the input of the program. After the inputs, the program takes all of the recorded files, renames them and distributes them in the proper folders.

It is important to mention that the data from the screwdriver comes in a semi-structured format mentioned in Section 2.3.1. The script automatically converts all of the semi-structured .KXML files to .csv format, before the files are distributed to the proper folders.

4.4 Data collection results

As the final result of the data collection, a dataset was created. This dataset contains 1341 recorded screwdriving operations. This section will show the distribution of the different classes in the dataset, together with the visualisation of the data collected.

The goal was to gather enough relevant and diverse data to effectively train, validate, and test a model that can predict outcomes or identify patterns in the screwdriving process. The data collected must be of good quality, meaning it should be accurate, consistent, and free from noise or errors as much as possible. Poor quality data can lead to poor model performance or misleading results. Machine learning models often require large amounts of data to learn effectively, so it is preferable to collect as much data as feasible. If some classes are over-represented in the dataset, the model might become biased towards predicting those classes. It's important to have a balanced dataset, or to use techniques to handle imbalanced data effectively.

4.4.1 Final dataset

As mentioned, the final dataset contains 1341 screwdriving files, for each of the 3 data sources. There is 5 classes in the dataset shown in Figure 4.14. This figure shows the class distribution and the number of screws in each class.

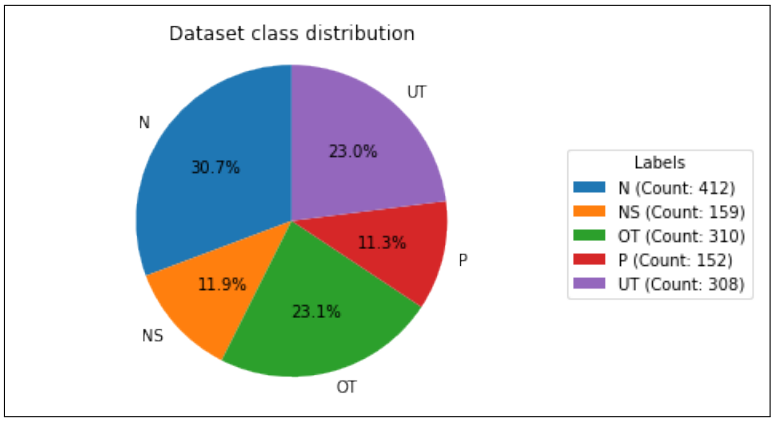


Figure 4.14: *Dataset class distribution*

The balance of the dataset is logical, because the N (normal screwdriving), UT (under-tightened) and OT (over tightened) classes are forcefully created and have a similar amount of representative data. Classes NS (no-screw) and P (pose anomaly) are not created forcefully, which is why there is a smaller amount of the representative operations.

The dataset was created on 17 different wood beams, and the class distribution of the every beam can be seen in Appendix A.

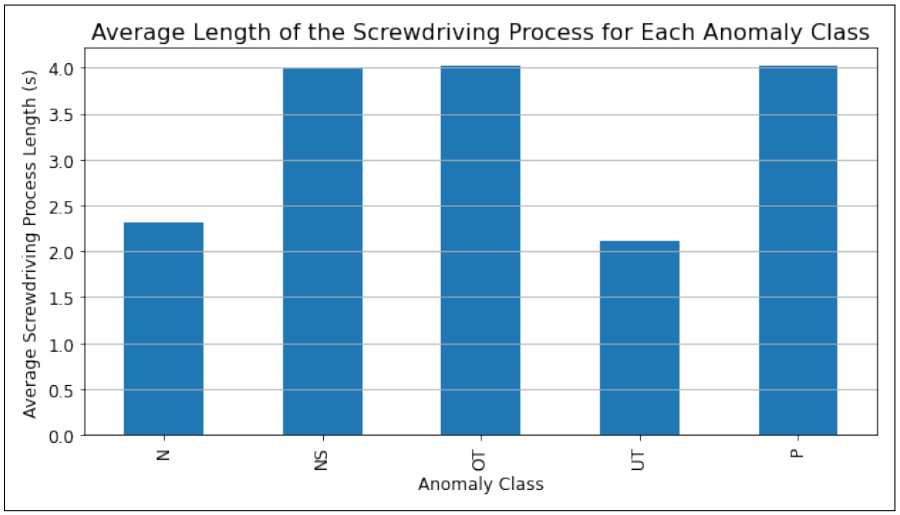


Figure 4.15: *Screwdriving process average process time*

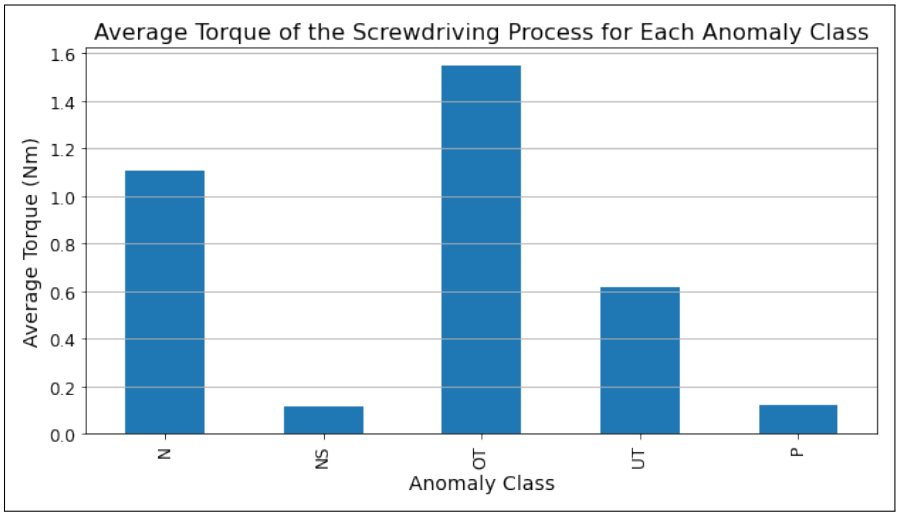


Figure 4.16: *Screwdriving process average torque*

Figure 4.15 shows the average time it takes to screwdrive each class of screws. It can be noticed that on average class N and class UT have the same length of approximately 2 seconds, and class NS, class OT and class P length of approximately 4 seconds. This will be a problem if the ML model will take time-series dataframes, because every input will have to be of the same length.

Figure 4.16 shows the average torque that the screwdriver produces for each class of screws. It can be noticed that on average class OT has the highest average torque, together with N. This correlates to the settings and the setup of the experiment. Also, class NS and P has the smallest torque value, which is also correlating correctly.

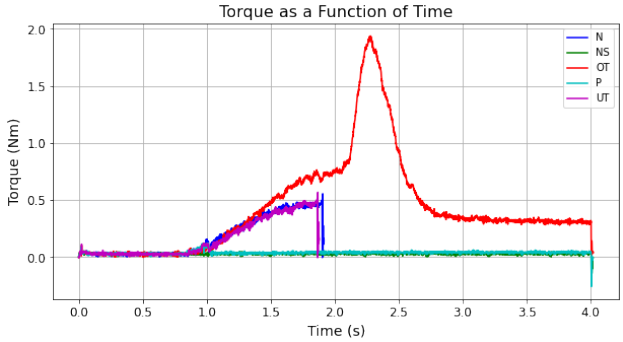
Data from these graphs can be an indication that the general classification of screws in different classes shows results that correlate to the class definitions and screwdriver controller setup.

4.4.2 Data visualisation

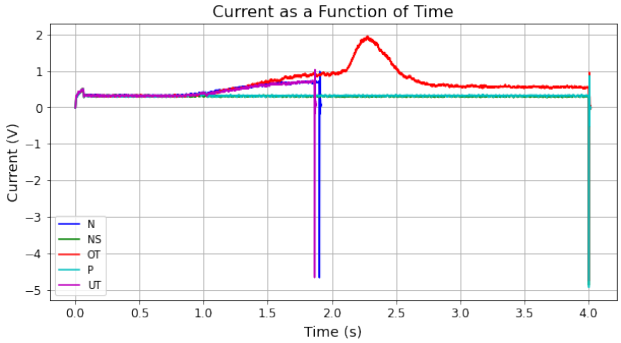
To get a visualisation of the sensory data collected from all data sources, this section will show the graphs of the recorded data.

Intrinsic sensor data by class

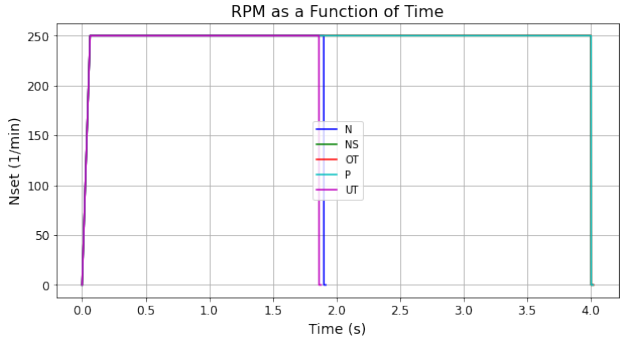
Figure 4.17 shows the sensor measurements of the screwdriver. Every graph contains data about 5 different screws, each representing a one screwdriving type or class.



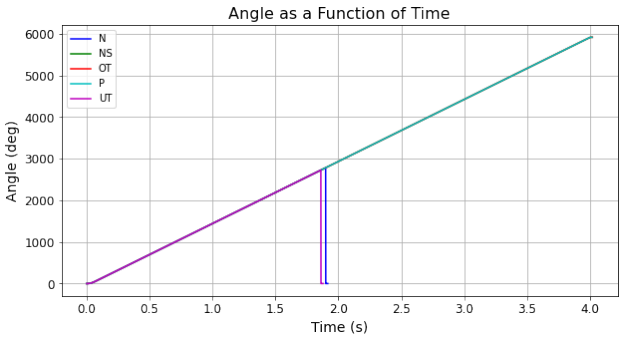
(a) *Torque graph as a function of time of the different classes*



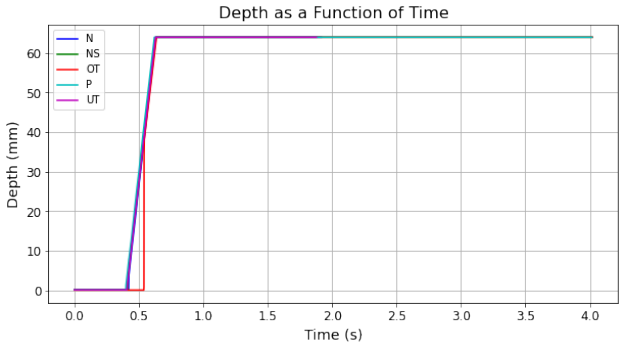
(b) *Current graph as a function of time of the different classes*



(c) *RPM graph as a function of time of the different classes*



(d) *Angle graph as a function of time of the different classes*



(e) *Depth graph as a function of time of the different classes*

Figure 4.17: *Intrinsic sensor data visualisation*

Figure 4.17a shows a variation in the torque data. It is noticable that the value of torque for the over-tightening class (OT) comes to a maximum torque value, after which it falls off. This effect happens when the threading is destroyed and the friction of the wood decreases together with the torque. "Normal screwdriving" (N) and "under-tightening" (UT) raises to the specified torque and

does not fall off, because the threading is not destroyed. The "no-screw" (NS) and "pose anomaly" (P) class does not show a increase in the torque value, because the bit of the screwdriver does not come in to contact with the screw, or comes in a weak contact.

Figure 4.17b shows the variation of the current to the screwdriver. As the value of torque, the current shows a similar behaviour.

Figue 4.17c and Figure 4.17d show that the RPM is the same for every screwdriving type and that the angle follows the same line for every screwdriving type. This correlates, probably because when rotation is constant, then the rotation angle of the screw increases linearly with time.

Task data visualisation

Data from the robot is straightforward. Figure 4.18 shows the movement of the robots Tool Center Point in x, y and z axis, for 5 different screw types. The movement of the robots TCP is not long, and as can be seen from the figure it is messy for every screw class, instead of the no-screw class. No-screw class obviously shows movement which is less sporadic the the rest of the classes, which is to be expected, since no screw is used.

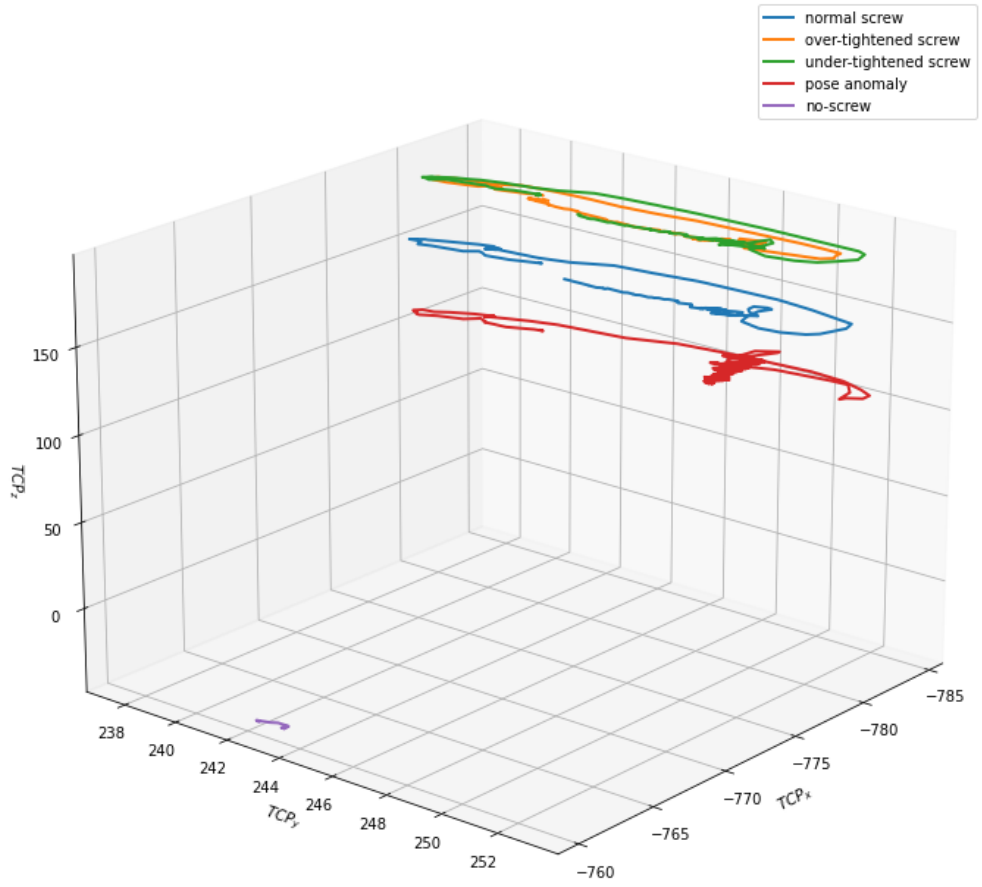


Figure 4.18: *TCP 3D position of the classes*

In the Figure, TCP_z represents the horizontal axis, which means the axis which is parallel with the table and the beam. Because the screw values diverge in the values of TCP_z, it will be important to somehow get the different screws in the same frame. One way of doing that would be to look at

offset in the axes, instead of the positions.

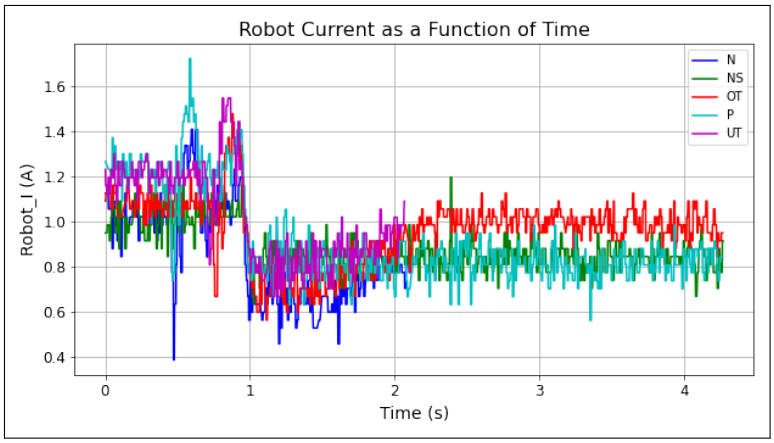


Figure 4.19: *Current to the robot as a function of time*

Another value that can be visualised is the current received by the robot. Figure 4.19 shows the current that the robot received during the 5 different screwdriving processes, and 5 different classes. The current is fluctuating, and does not have a visible pattern. In the figure it is hard to find patterns and correlations of different classes. This will be the task of machine learning algorithms.

A case could be made for the over-tightening, where the current was stronger during the screwdriving process after the 2 second mark. The reason for that could be the fact that the screw types were randomly selected from the dataset, so no conclusive evidence can be derived from the figure.

Extrinsic data visualisation

Figure 4.20 shows the recorded sound of one screwdriving process in a Mel spectrogram and waveformat. Mel spectrogram is a type of spectrogram where the frequency scale is converted to Mel scale. This scale is the perceptual scale of different pitches, who's idea is to approximate the human ear response of different frequencies. It is computed by the usage of Fast Fourier Transformation (FFT) of a signal, which creates the frequency spectrum. Afterwards, a set of filters is used to sum the energy. Using this spectrogram can be useful to use as a feature for different audio signal processing tasks.

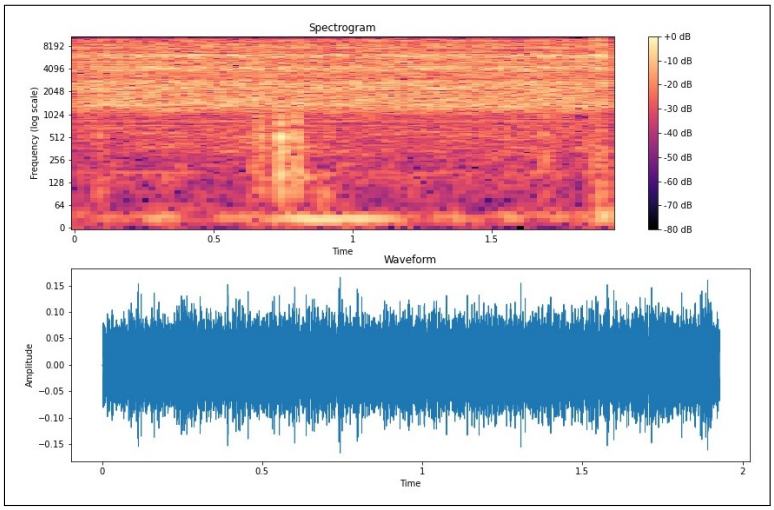


Figure 4.20: *Representation of the audio recording as Mel spectrogram (top), and waveform (bottom)*

From the visualisation of the sound, it is clear that the audio has a lot of noise. This is due to outside noise from the server and air from the pneumatic system, mentioned in Section 3.4. To work with the sound information, it will be a good idea to clean this noise.

5 Preprocessing and feature selection

After the data collection, where the final dataset for ML was created and analysed, data has to be preprocessed, cleaned, organized, and structured and the features for the algorithm have to be selected. This chapter will show what was done to the data, and what are the features extraced for the models.

As shown in Figure 5.1, the data will be separated in 3 sources, and combined between them. This will result in dataframes which will be ready for the stage of feature selection.Since the data gathered is in raw sensory time-series or audio format, for machine learning purposes the relevant features will be extracted using different libraries.

Together with the dataframes, there will be a dataframe containing the labels and metadata, that will correlate screwdriving classes with the individual screws.

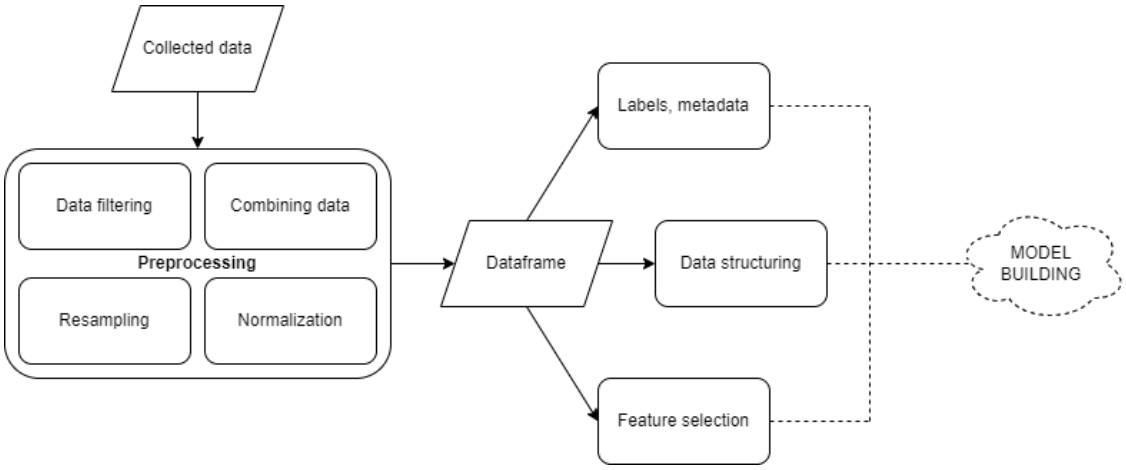


Figure 5.1: *Diagram of preprocessing and feature selection*

5.1 Preprocessing

A large part of prepreocessing happened in the data collecting phase. The data from the screwdriver was immediately converted with the python sorting script to a manageable table .csv format. Also, the robot data was during the collection converted to units that match the data from the screwdriver controller:

- TCP position to millimeters
- TCP rotation to radians
- current to the robot to amperes

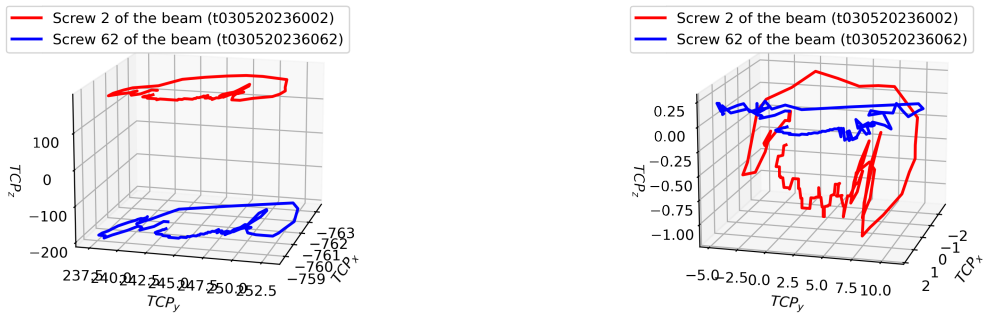
The task data from the robot and extrinsic data from the microphone was recorded in the same time-frame as the data from the screwdriver explained in Section 4.3.1. In the extrinsic data example, one of the way would be to record a long audio segment of screwdriving different screws, which would

follow by cutting the audio on segments that would match every screw specifically. By implementing the strobe signal to start and stop the recording, this was avoided.

Ideally, all of the data collected would be collected trough a data collection system or program. In case of this project, data from the screwdriver controller was collected with a different program then the robot and microphone data. This resulted in a time dilation of the different sources of data by 200ms on average. Usually the task and extrinsic data is recorded for 200ms more then intrinsic data. The reason for this is unknown and it is hard to decide why did this happen, but it could have to do with the laptop operating system and the delays in the python script. However, with keeping this in mind, it should not have a big influence on the final results of the classification.

5.1.1 Robot TCP position calibration

During the data collection, the UR10 robot is used to position the automatic screwdriver in position for every screwdriving operation. This means that every screw has a different starting position in the 3D space, as TCP position in x, y and z axis. Before the feature extraction, there was a need to uniform this data from the robot. A way to do that was to make every screw's starting position as 0, by subtracting every TCP position column by the first data point in the respective column. The before and after data is shown in Figure 5.2.



(a) *TCP position before calibration* **(b)** *TCP position after calibration*

Figure 5.2: *TCP position calibration*

5.1.2 Dataset management

The data for this project was collected from 3 sources: intinsic data, extrinsic data and task data. After all of the separate data was collected in the data collection process and the final dataset was made, a new file containing intrinsic and task data was created. As shown in Figure 6.1, this file contains all of the separate screw measurements of the screwdriver and the robot, with the screw ID ("Source") connected to the sensor type ("Type"), with the time and value. This dataframe was saved as a Hierarchical Data Format version 5 (HDF5). HDF5 is an open source file format that supports large, complex, heterogeneous data.

Since extrinsic data comes in the waveform format, it will be connected with the intrinsic and task data after the feature selection process if needed.

Source	Time (ms)	Type	Value
i030520235006	0.000	Nset (1/min)	0.000
i030520235068	0.000	Nset (1/min)	0.000
i050520238018	0.000	Angle (deg)	0.000
i030520237070	0.000	Angle (deg)	0.000
i280420232085	0.000	Angle (deg)	0.000
...	...	...	...
t1005202314051	5001.519	TCP_y (mm)	2.600
t1005202314051	5001.519	TCP_x (mm)	0.000
t1005202314051	5001.519	TCP_rz (rad)	-1.204
t1005202314051	5001.519	TCP_z (mm)	-4.600
t1005202314051	5001.519	TCP_rx (rad)	1.234

Figure 5.3: *Combined dataframe*

One of the requirements of this project, shown in Table 3.1, is to test different data on the machine learning algorithms. This can be an indicator on how using different sensory data can influence the classification of anomalies. The following data combinations will be used in building the models:

1. **Intrinsic data**
2. **Task data**
3. **Extrinsic data**
4. **Intrinsic data + Task data**
5. **Intrinsic data + Task data + Extrinsic data**

5.1.3 Data cleaning

After the data is collected and after features are extracted, there can be problems and errors. During the data cleaning of the intrinsic, task and extrinsic data some measures were taken to clean the data of these problems. Most notable measures are:

1. **Check for missing (NaN) values:** checking if there is missing values in the dataframe by the use of numpy 'isna()' command
2. **Check for infinite values:** checking if there is infinite values in the dataframe by the use of numpy 'isinf()' command
3. **Check for values too large for 'float64':** the maximum finite presentable floating-point number in 'float64' is approximately 1.8e308. There is a check on if the dataframe contains values larger than this

After the measures check the data for the problems, there is a question on dealing with them. No errors or missing values have been found in the data received from the sensors.

Cleaning noise from the audio

As mentioned in Section 3.4, audio was recorded in a environment that contains a large amount of external noise. This "noisy" audio is not the part the process itself, hence it is labeled as noise.

Every day during the data collection a 10 minute audio recordings of the external noise were recorded. This "noise" audio files will be used in the noise reduction process.

Implementation of noise reduction is conducted with the following steps:

1. **The Short-Time Fourier Transformation (STFT) is calculated from the "noise", 10 minute audio recordings.**

   STFT as a sequence of Fourier transforms of a windowed signal, it provides a time-localized frequency information in situations where frequency components of a signal vary over time. [12] This allows us to see how the frequency content of the signal changes over time, which is helpful in identifying and removing time-varying noise.

2. **Calculating the average frequency range.**
3. **Applying a low pass filter.**
4. **Noise reduction of the audio dataset.**

Figure 5.4 shows a example of a screwdriving audio file before and after noise reduction. It is visible how the noise audio which occupies the higher frequency range is removed from the audio file, while the important middle and lower frequency range stays.

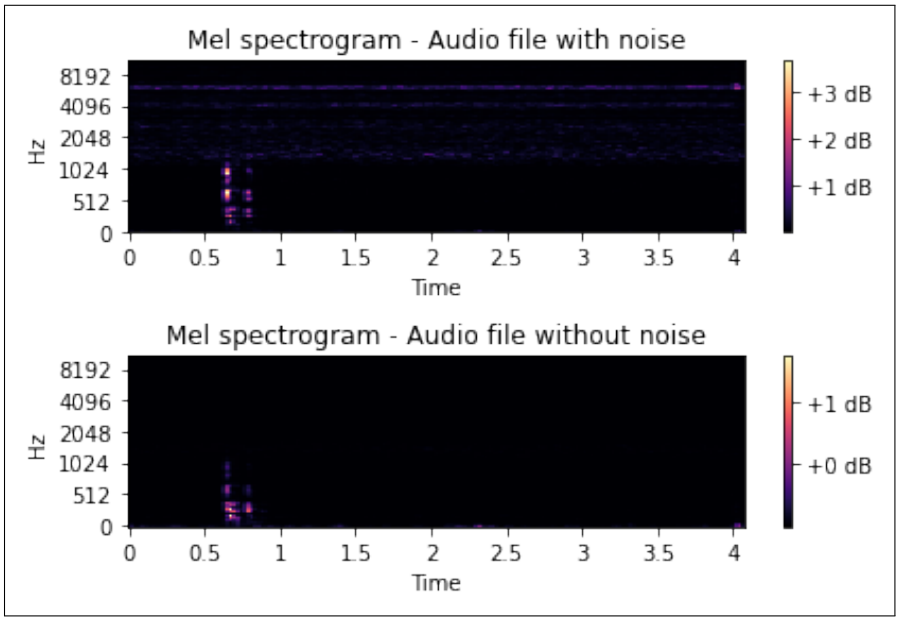


Figure 5.4: *Mel spectrograms before and after the noise removal process*

5.2 Feature selection

After the datasets that we want to use for machine learning are selected and preprocessed, the next step is to select features. Currently the data from the sensors of the screwdriver and data from the robot is in raw sensory format. Every screw has a time-series dataframe which shows the sensor measurements trough the screwdriving process. Raw time series data can often be noisy, high-dimensional, and contain complex temporal dependencies, making it challenging for machine learning models to learn from directly. Feature extraction can help alleviate these challenges.

5.2.1 Intinsic and task data features (TSFresh)

The features can be extracted manually, but there are different libraries that help with this process. One of them is called TSFresh. It is a python library designed for automatic feature extraction from a time-series dataset. It extract the relevant data and provides a comprehensive set of features. Some of feature extraction methods are basic statistics (like mean, variance), trends, seasonality and temporal dependencies.

Advantage of using a library like TSFresh is the ability to automatically select relevant features, shown with Figure 5.5. The library uses a hypothesis test to determine what is the best selection of features for the given problem. This reduces the risk of overfitting and removes the need for manual feature selection.

Figure 5.5: *Automatic feature extraction*

When using automatic feature extraction there are different settings that can be selected in the TSFresh library. The two settings that are selected are:

- **MinimalFCParameters:** (Intrinsic data - 50 features, task data - 70 features)
- **EfficientFCParameters:** (Intrinsic data - 3885 features, task data - 5439 features)

These two settings specify how many features will be extracted from the dataset. There are settings with a even larger amount of features, but extracting larger number of features would require stronger hardware then the one used for this project. THat is the reason of only using these settings.

In case of the time-series data, TSFresh is capable of handling multivariate time series and can automatically extract features from each variable. This is useful in this scenario, where the time series is characterized with multiple parallel streams of data from different sensors. [25]

Here are some examples of the data that the TSFresh features cover:

- **Distribution of the time series:** statistical properties including mean, median, standard deviation, variance, skewness, kurtosis, quantiles, etc.
- **Linear trends:** features related to a linear regression model fitted to the series
- **Non-linear trends:** features obtained by fitting a variety of non-linear models to the time series data
- **Complexity of the time series:** features that show the time-series complexity, including number of fluctuations, number of crossings above and below the mean, etc.
- **Stochastic properties:** features founded on autoregressive models, including autocorrelation, partial autocorrelation, etc.
- **Fourier Transform coefficients:** features derived from the coefficients of the Fourier Transform
- **Seasonality:** time-series seasonal decomposition
- **Statistical tests:** outcomes of various statistical tests, including Augmented Dickey-Fuller test (stationarity), The Anderson-Darling test (normality), etc.
- **Change detection:** capturing fast changes in the time-series

- **Wavelet Transform coefficients:** time-series at various scales

5.2.2 Extrinsic data features

For the audio extrinsic data, extracting features has to be configured separately. TSFresh can only automatically extract features from a specific format, and is not adjusted to extract features from audio files. The extrinsic data is recorded in .wav format, so it first needs to be converted, after which the features are extracted.

To extract features from the audio "librosa" library was used. Librosa is a Python open-source library, which is used for audio and music processing and analysis. This library provides necessary tools to retrieve and create audio and music information with a user-friendly interface. [8] Key features of the librosa library are:

- Audio feature extraction
- Audio visualization
- Audio manipulation
- Integration with other Python libraries

After research, these are the features that were extracted from the audio:

1. **'chroma_stft':** Chroma features can be used as a tool for representation of music audio, where the entire spectrum of the audio file is projected onto 12 bins which represent 12 semitones (or chroma) of the musical octave. In music theory, notes that are one octave apart are perceived as similar, so knowing the distribution of pitch class can be a good feature.
2. **'spec_contrast':** Spectral contrast is used to show the difference in amplitude between peaks and valleys inside of the sound spectrum. High and low contrast indicate clear harmonic structure and inharmonic structure (noise). This contrast is used as a tool in quantifying prominence of the harmonic peaks.
3. **'tonnetz':** Tonnetz is a feature that presents a geometric representation of musical pitch classes (notes), often used to visualize tonal relationships in music. This can indicate harmonic relations of different notes and is often used in music analysis.
4. **'spectral_bandwidth':** Spectral bandwidth refers to the width of the band of frequencies in a sound. While a sound having a wider bandwidth usually means that the sound is "noisy" or complex, if a sound has a more narrow bandwidth might indicate a more pure tone.
5. **'zero_crossing_rate':** Zero crossing rate is the rate at which a signal changes from positive to zero to negative, or from negative to zero to positive. It's often used as a simple measure of the perceptual 'pitch' of a sound - higher zero crossing rates can often be indicative of higher pitch.
6. **'spectral_rolloff':** Spectral rolloff is a specific frequency, underneath which is a specified percentage of the total spectral energy. If the roloff is at 85%, this is the frequency below which 85% of the energy is contained. This is a good indicator for harmonic (low roloff) and noisy (high roloff) sounds. The best way to describe it is as the 'brightness' of the sound.
7. **'mfcc':** Mel-frequency cepstral coefficients (MFCCs) are a set of features (40 features) which are used to concisely describe the overall shape of a spectral envelope. They provide a high-level representation of the power spectrum of an audio signal in a way where they capture the phonetic content carrying elements of the signal.

5.3 Feature normalisation

The extraction of features results with different features from the data which are diverse in the format and scale. Because of that feature normalisation can be an important step in the preprocesing stage, because it brings all features to a similar scale or distribution. Normalisation helps in avoiding problems, which arise by varying units, ranges and magintudes between features. That can have a negative influence on the performance of different ML algorithms. In the project the main normalisation technique is the popular sikit-learn StandardScaler. [21]

The StandardScaler is a widely used feature normalisation technique, which works with a simple principle forcing features to have a zero mean and unit variance. It functions by independently scaling each feature and transforming the data in a way that it has a mean of zero and a standard deviation of one. This is achieved by the following mathematical equation:

$$X_{\text{scaled}} = \frac{X - \text{mean}(X)}{\text{std}(X)}$$

The X is representing original values of features, mean(X) is the mean of these values and std(X) is the standard deviation of the feature values. By subtracting X with the mean of X, StandardScaler creates transformed feature vales with a mean of zero and a standard deviation of one.

In conclusion, the benefits of using the feature normalisation techniques are:

1. Preserving information by keeping the distribution and shape of the original feature while forcing them to the same scale.
2. Model improvement, because some machine learning algorithms perform better and converge faster with the use of normalised features. By using normalisation technique like StandardScaler, the model is not biased towards the features with larger ranges and scales.
3. Features are more interpretive, since they are all on the same scale. Because of this their coefficients and feature importance can be compared directly.

Another thing to note is that ensemble methods have a built-in mechanisms which are making them robust varying scales and distributions. Internal processes like relative comparisons and model aggregation can enable them to not need feature normalisation.

6 Model building

As the final chapter of the solution phase, model building and results will show how do different models deal with the classification of the data that was collected. This report has specified the different screwdriving types or classes that will be used as the "label" in classification. The goal will be to see how accurate are the different models in recognising which screw belongs to which screwdriving type. As a starting point, this chapter will give an overview of the method in which the results will be compared and interpreted.

6.1 Researched models

This chapter will use the finalised sets of features and data to build machine learning models which will classify the data as the result. After the preprocessing and feature selection, different combinations of features are created. Models will use these feature combinations and show the results of the classification. Figure 6.1 shows how different sources of data will be used in creating the models. The models are divided in 3 stages:

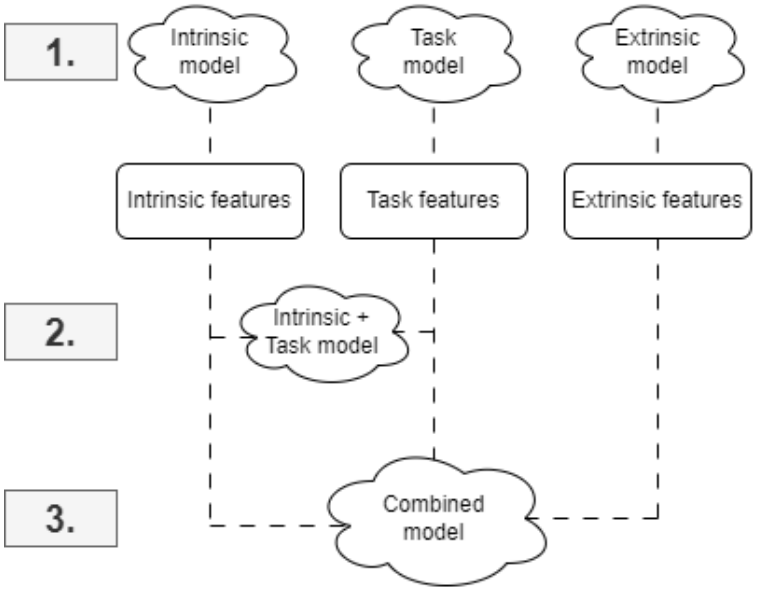


Figure 6.1: *Diagram of model building*

1. **Baseline models:** this models will be created from every data source separately. There will be 3 models from intrinsic, task and extrinsic data. Baseline models will be used as the basis for further model building. On the baseline models different algorithms are tested, with only small hyperparameter changes to achieve the optimal model accuracy's.
2. **Intrinsic + Task model:** this model will combine the intrinsic screwdriver data and task robot data together. The goal is to see if there are improvements on the accuracy of classification using these two data sources in combination.

3. **Combined "meta" model:** this final model will include all of the data sources together and explore how do they work together. The best algorithm will be explored together with the best hyperparameters.

6.2 Splitting the data

After the data needed for classification including the training data and the "grand truth" labels is loaded, it needs to be separated in data for model training (training set) and data for model testing (test set). It is sparted using the sikit-learn's train-test-split function shown in Figure 6.2.

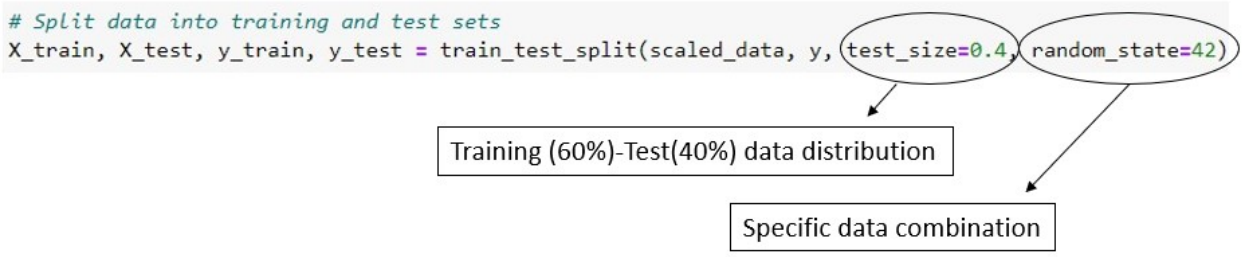


Figure 6.2: *Train-test-split function*

To keep results of different models comparable it is decided to split the data in 60% - 40% ratio. This ratio was decided based on the smaller size of the dataset. The parameter "random_state", keeps the data combination same every time the model is trained, so the results can be replicated. As the result, Table 6.1 shows how the test data is distributed based on the 5 classes.

N	NS	OT	P	UT	TEST SET
165	66	138	59	109	537

Table 6.1: *Training set distribution by class*

6.3 Algorithm selection

When building the models, different ML algorithms are used. The selected algorithms are tested on the baseline models, to achieve the optimal results of the classification. Most of the machine learning algorithms are created with the scikit-learn library. [19] The selected algorithms are:

1. **K-Nearest Neighbours (KNN):** KNN algorithm belongs to the group of algorithms using instance-based or memory-based learning. This algorithm does not learn a model from training data but memorizes the dataset instead. The classification of new instances is made based on similarity measures (like distance) with instances stored in memory. The distance of the neighbors can be calculated using different metrics like Euclidean, Manhattan, or Minkowski, which correlates to the specific problem. Because the KNN algorithm makes no assumptions about the distribution of underlying data, it is suitable for non-linear data. [18]
2. **Support Vector Machines (SVM):** SVM algorithms are supervised learning algorithms primarily used for classification. Main goal of an SVM is finding a hyperplane in an N-dimensional space (N - the number of features) that separates and classifies the data points. If the dataset is separable linearly, SVM will find the optimal hyperplane from the margin

of the nearest data point of all classes, also known as support vectors. Usually SVM is good in dealing with high dimensional data. If the data is non-linear, SVM can use a kernel trick by transforming the data into a higher-dimensional feature space. It is important to carefully select the kernel function and associated parameters. [22]

3. **Random Forrest (RF):** RF is a machine learning algorithm often used that builds on the simpler decision tree algorithm. It belongs to a group of ensemble learning methods, meaning that it takes the concept of a single decision tree and creates a group of them 'forest' and arrives at the best solution. RF works on the principle of generating multiple decision trees and aggregating their results. Every sample is drawn with replacement (bootrstrap sample) from the training set and builds a tree in the forest. When splitting each node during the tree construction, the best split is found from a random subset of features size m, or from all input features. The randomness makes the model more robust then a single decision tree which helps with overfitting problems. In classification the final prediction is based on the majority voting principle in a way where the class with most votes from all trees in the forest becomes the prediction of the algorithm. Ensemble learning models do not require feature scaling, and the key parameter is the adjustment of the number of trees, number of features sampled at each split and the depth of the trees. [20]

6.3.1 Algorithm settings

Adjusting the hyperparameters of the algorithms to improve their ability of classification is one of the important steps in building machine learning models. The models selected do not have a large amount of parameters that can be changed, but the most notable ones are:

KNN:

- N-Neighbours - the number of neighbours is a parameter of the KNN algorithm which sets the number of data points that the target data point will look for when selecting a class it belongs to. Often, it is a good practice to use the KNN algorithm with different number of neighbours to see which number gives the best algorithm precision.

SVM:

- Kernel - kernel, or the kernel function is used to transform the input data into a higher-dimensional space which allows easier classification with the linear classifier. Often used choices for the kernel function are "linear", "rbf" (Radial basis function), "sigmoid" and "poly".
- Gamma - parameter used in 'rbf', 'poly' and 'sigmoid' kernels. As an example in "rbf" kernel, it defines the influence of a single training data point. Low values have bigger influence, and high values lower.
- C - regularization parameter, also named cost parameter. The purpose of this parameter is determining the trade-off between achieving low error on the training data and minimizing the model complexity. Lower C creates a smoother decision surface and the large C tries to classify all training data points correctly, in a way of giving the model more freedom to select more samples as support vectors.

RF:

- N_estimators - this value defines the number of decision trees in the random forest algorithm. By increasing the number of estimators, the model's performance increases with an increase in computational requirements.

- min_samples_split - this value represents the minimum number of samples that is required to split an internal node. If the node has less then this value, it is not split and it is turned to a leaf node by default. This hyperparameter is often used to help in the overfitting control. Small value of this parameter makes the model more complex, which could cause oferfitting. Larger value of this parameter leads to a simpler model, which could cause underfitting.
- max_depth - this parameter shows the maximum depth of the tree. This measure is the maximum distance between the root and any leaf. By not specifying this parameter, nodes will be expanded until all leaves are pure or containing lessthen 'min_sample_split' samples.

7 Results

As shown, there are 3 model divisions in machine learning. Basic models, Intrinsic + Task model and Combined "meta" model. This divisions include different models created, so the results will give information about the accuracy of every model, and the ranking of the models in the division. In the Results section, the most important information is presented. The classification reports, confusion matrix and other relevant data for the models are presented in the Appendix.

7.1 Results format

The main format of the machine learning results of models will be a confusion matrix. Confusion matrix is presented as a table used in evaluation of the performance of a classification model, usually in supervised learning. In summary it represents the model's predictions compared to the true values.

7.1.1 Binary classification problem

As an example of a binary classification, Table 7.1 shows the following measurements:

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

Table 7.1: *Confusion matrix for binary classification*

- **True positive (TP):** this measure shows the number of instances where the positive class is predicted correctly as a positive
- **False positive (FP):** this measure shows the number of instances where the positive class is predicted incorrectly as positive.
- **False negative (FN):** this measure shows the number of instances where the negative class is predicted incorrectly
- **Negative (TN):** this measure shows the number of instances where the negative class is predicted correctly

From the classes presented, the metrics that are needed to express the model's performance are calculated:

Recall: indicator of the model's ability to identify all instances of the positive class correctly

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

Precision: indicator of the model's ability to classify the positive class correctly

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

F1 Score: metric that combines recall and precision. It is defined as the harmonic mean of precision and recall

$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$

7.1.2 Multi-class classification problem

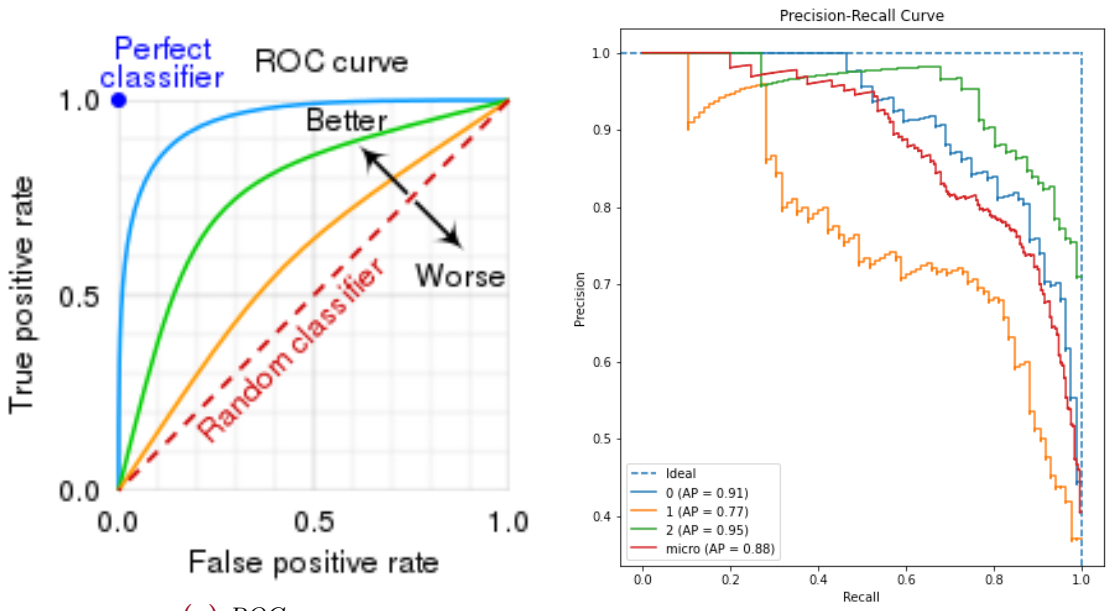
In the case of this project the confusion matrix will have 5 classes, shown in Table 7.2.

	Actual class A	B	C	D	E
Predicted Class A	AA	AB	AC	AD	AE
B	BA	BB	BC	BD	BE
C	CA	CB	CC	CD	CE
D	DA	DB	DC	DD	DE
E	EA	EB	EC	ED	EE

Table 7.2: *Confusion matrix with 5 classes*

- The top-left diagonal to bottom-right (AA, BB, CC, DD, EE) shows the number of instances predicted correctly for each class. AA is the number of instances in class A that are correctly predicted as class A, BB is the number of class B instances that are correctly predicted as class B, and so on. This is the same as True Positives in the binary matrix, but for each class separately.
- Off-diagonal elements show the instances which are missclassified. AB represents the instances which are of class A but were predicted incorrectly as class B by the classifier.

To better understand and visualise different model results, Figure 7.1 shows two diagnostic curves which help in the visualisation and comparison of different classification models. During the results the ROC curve and Precision-Recall curve are presented for every model.



(a) *ROC curve*

(b) *Precision-Recall curve*

Figure 7.1: *Model diagnostic curves*

Receiver Operating Characteristic (ROC)

ROC curve is a graphical plot often used in diagnostics of a machine learning model. It illustrates the classification model's diagnostic ability as its discrimination threshold is varied. It is created by plotting the rate of the true positives (recall) against the rate of the false positives (fall-out) at various threshold settings. Area under the ROC curve is called AUC (Are Under the Curve). It summarises the overall model performance as a single scalar value. 1.0 is ideal and it indicates that the classifier shows the prefect discriminatory ability. The closer the curve is following the left border and the top border of the plot, the more accurate the model.

Precision-Recall

A Precision-Recall curve is a graphical plot of the precision (y-axis) against the recall (x-axis), similar to the ROC curve. This diagnostic tool shows the tradeoff between recall and precision. Bigger area under the curve shows a high recall and high precision.

7.2 Baseline models

First results are based on the baseline models containing data types separated by the source of data. The results of different models are presented in the Appendix B, and contain information on the KNN, SVM and RF model for every data source in a form of classification reports and confusion matrices.

Table 7.3 shows the accuracy comparisons of every model based on the algorithm used and data source provided. The intrinsic baseline model and the task baseline model performed the best with the RF algorithm, while the extrinsic baseline model performed the best with the SVM model.

	INTRINSIC BASELINE MODEL			TASK BASELINE MODEL			EXTRINSIC BASELINE MODEL		
Algorithm	KNN	SVM	RF	KNN	SVM	RF	KNN	SVM	RF
Accuracy	0.9404	0.9572	0.9683	0.9311	0.9255	0.9479	0.6406	0.7821	0.7803
Precision	0.9423	0.9576	0.9691	0.9321	0.9295	0.948	0.6771	0.8085	0.8026
Recall	0.9404	0.9572	0.9683	0.9311	0.9255	0.9478	0.6406	0.7821	0.7803
F1-Score	0.9404	0.9572	0.9684	0.9313	0.926	0.9479	0.6305	0.7787	0.7759
AUC	0.977	0.9941	0.9945	0.9839	0.9928	0.9937	0.8761	0.9632	0.9535

Table 7.3: *Combined baseline model results*

7.2.1 Intrinsic baseline model

From the results, the Random Forest algorithm was best in classification of the screwdriver intrinsic data. The algorithm showed an accuracy of 97% which shows the classification was successfully in general. Model shows strong performance with high precision, recall, and F1 score across multiple classes. Model is able to classify the majority of instances correctly. For a deeper look in to the results of different algorithms, data of the algorithms is available in Appendix B.1.

Algorithm settings:

- KNN: number of neighbours sekected in the KNN algorithm is n=3. Figure B.1 shows how does the algorithm perform with different number of neighbours.
- SVM: kernel selected for the SVM algorithm is "rbf".
- RF: N-estimators was selected to be N=100.

Results:

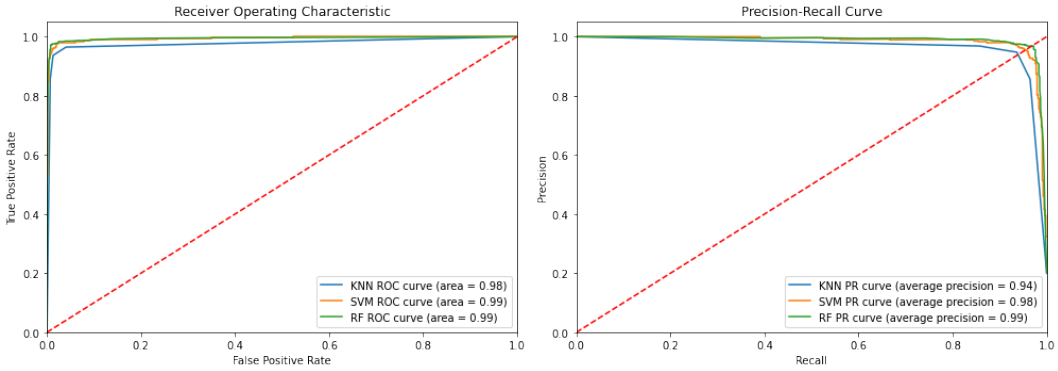


Figure 7.2: *ROC and Precision-Recall graph for intrinsic data*

Figure 7.2 shows the ROC and Precision-Recall curves for algorithms that are using intrinsic data. An important metric is the area under these curves, where it is also visible that RF models shows the best results. RF and SVM results show similar classification accuracy, while the KNN algorithm has a lower accuracy. All of the algorithms have good performance.

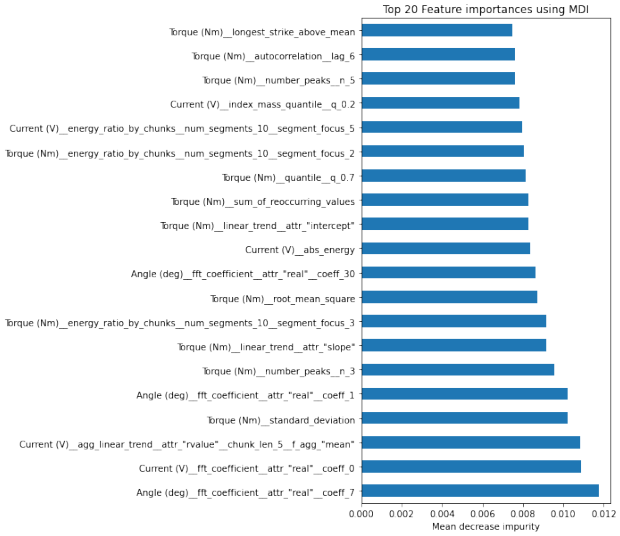


Figure 7.3: *Feature importance of the intrinsic baseline model*

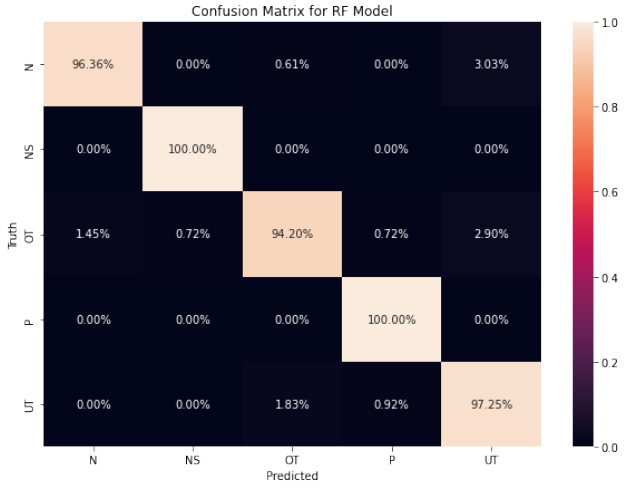


Figure 7.4: *Confusion matrix of intrinsic RF model*

Figure 7.3 shows the feature importance of the trained Random Forest algorithm. The bar graph contains a set of top 20 best performing features of this model. Feature importance's are calculated using the .feature_importance's attribute and they represent the relative importance of each feature in the classification process. The feature importance can be used to adjust the models and reduce overfitting by using only the important features, but it was not done for the baseline models. All of the top 20 features include torque, current and angle features.

Figure 7.4 shows the confusion matrix for the best algorithm, which is in this case RF. The algorithm showed a good classification capability, with the only notable mistakes are 3.03% predictions of the normal screwdriving as under-tightening and 2.9% predictions of over-tightening as under tightening.

7.2.2 Task baseline model

From the results, the Random Forest algorithm was best in classification of the robot task data. The algorithm showed an accuracy of 95% which shows the classification accuracy was a little lower then the accuracy of the intrinsic dataset, but is still successful in general. Model shows strong performance with high precision, recall, and F1 score across multiple classes. It is able to classify the majority of instances correctly. Specific results of the algorithms are available in Appendix B.2.

Algorithm settings:

- KNN: number of neighbours selected in the KNN algorithm is n=10. Figure B.5 shows how does the algorithm perform with different number of neighbours.
- SVM: kernel selected for the SVM algorithm is "rbf".
- RF: N-estimators was selected to be N=100.

Results:

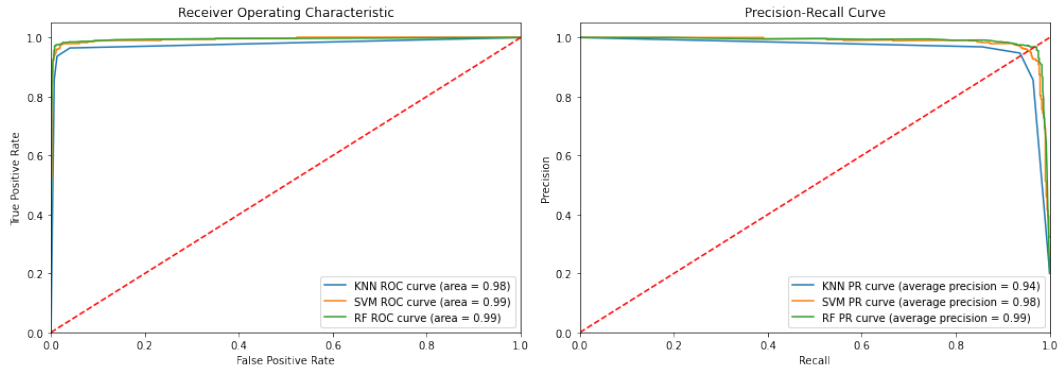


Figure 7.5: *ROC and Precision-Recall graph for task data*

Figure 7.5 shows the ROC and Precision-Recall curves for algorithms that are using task data. As in the case of intrinsic data, the RF and SVM algorithm are closer together in the AUC.

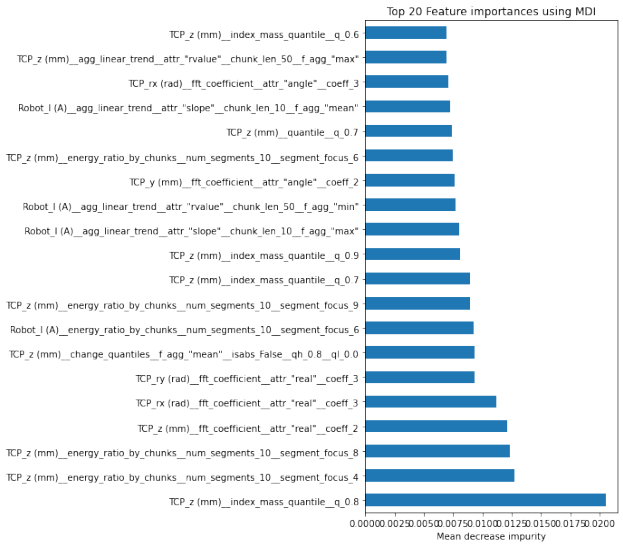


Figure 7.6: *Feature importance of the task baseline model*

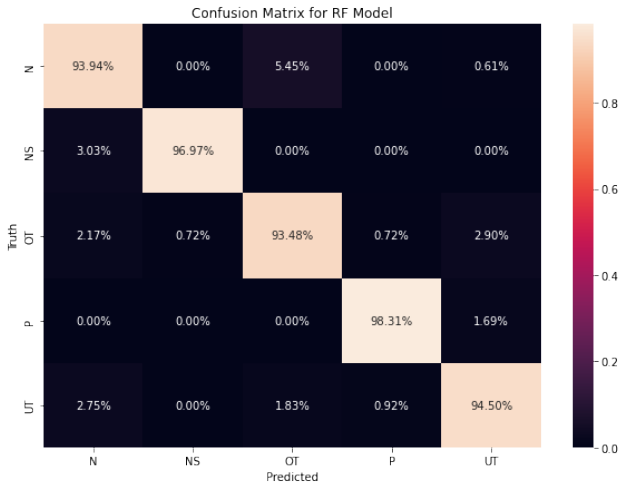


Figure 7.7: *Confusion matrix of the task RF model*

Figure 7.6 shows the feature importance of the trained Random Forest algorithm with task data. The bar graph contains a set of top 20 best performing features of the task RF model. Tool Centerpoint Position (TCP) features in z axis occupy the majority of the top 20 features. One feature has close to double influence of the other features, and it is TCP_z index mass quantile. This tsfresh feature, gives insights into the distribution and concentration of values in the TCP_z data of the robot. It is giving information about the index value below which 80% of the mass or weight of the data is located.

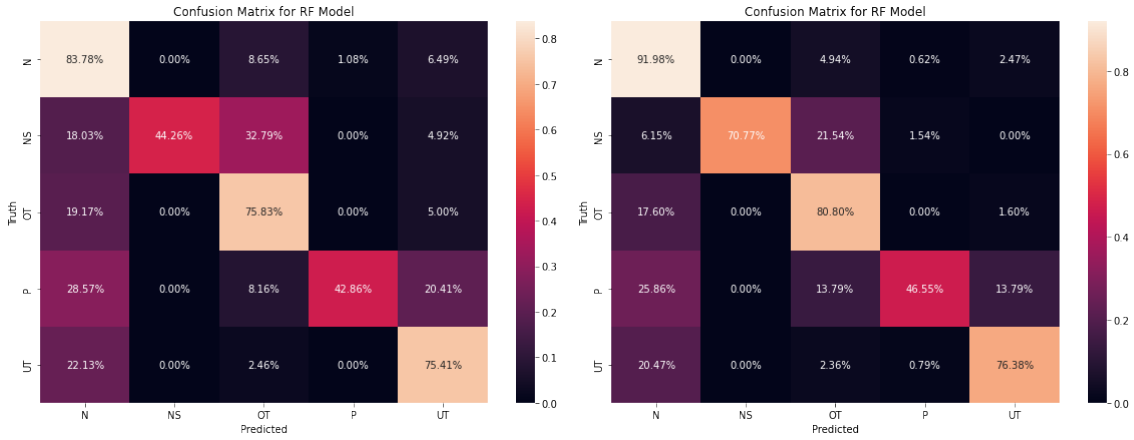
Figure 7.7 shows the confusion matrix for the best algorithm, which is in this case again RF. The algorithm showed a good classification capability, with the only notable mistakes are the 5.45% predictions of the normal screwdriving as over-tightening. This makes sense, because the robots position should stay in a similar position, without an offset, during these 2 screwdriving types.

7.2.3 Extrinsic baseline model

Before presenting the results of the extrinsic baseline model, metrics of the data before and after cleaning will be presented. The goal was to clean the audio data from noise in the preprocessing stage, explained in Section 5.1.3. The following results in Table 7.4, show how successful was the cleaning process by comparing the results of the RF algorithm on audio data before and after "noise" removal process. Figures 7.8 show the confusion matrices of data before and after noise removal.

RF algorithm	accuracy	precision	recall	F1-score
"Noisy" audio data pre-cleaning	*70.95*	*72.98*	*70.95*	*70.04*
"Clean" audio data after-cleaning	*78.03*	*80.25*	*78.03*	*77.59*

Table 7.4: *Comparison of the RF algorithms for extrinsic data before and after cleaning*



(a) *"Noisy" data confusion matrix* **(b)** *"Clean" data confusion matrix*

Figure 7.8: *Comparison of RF algorithm's confusion matrices of extrinsic data*

From the results of classification, there is an increase in RF algorithm accuracy on "clean" audio data of around 8% compared to the "noisy" data. This is a good indicator that the noise removal process was successfully in removing the outside noise, and the result is audio files which provide better data for classification.

From the main results in Table 7.3, the Support Vector Machines algorithm was best in classification of the robot extrinsic data. The algorithm showed an accuracy of 78.21% which shows the classification

accuracy was a significantly lower then the accuracy of the intrinsic and task dataset, but still has promising results, when all the problems with recording sound of the process are taken into consideration. Specific results of the algorithms are available in Appendix B.3.

Algorithm settings:

- KNN: number of neighbours selected in the KNN algorithm is n=4. Figure B.9 shows how does the algorithm perform with different number of neighbours.
- SVM: kernel selected for the SVM algorithm is "rbf".
- RF: N-estimators was selected to be N=100.

Results:

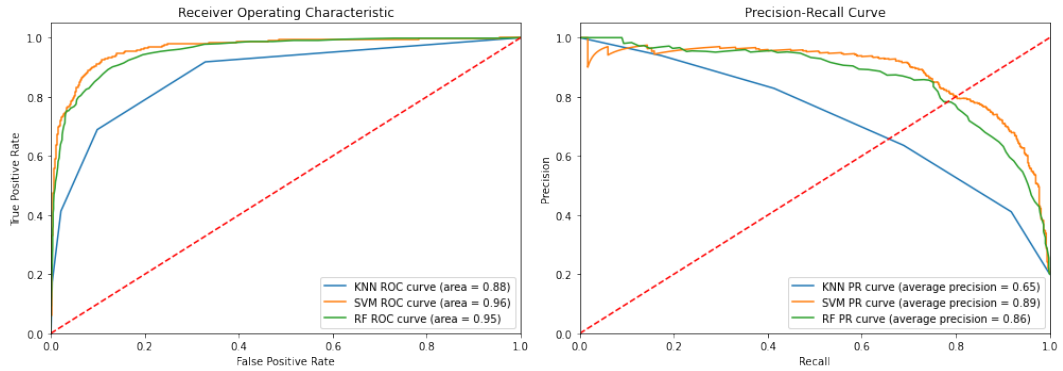


Figure 7.9: *ROC and Precision-Recall graph for extrinsic data*

Figure 7.9 shows the ROC and Precision-Recall curves for algorithms that are using extrinsic data. Extrinsic data source gave best results of the classification with the SVM algorithm, together with the RF algorithm. KNN algorithm showed worse results by a big margin.

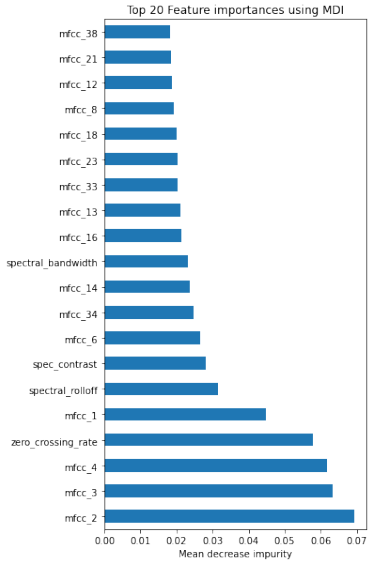


Figure 7.10: *Feature importance of the extrinsic baseline model*

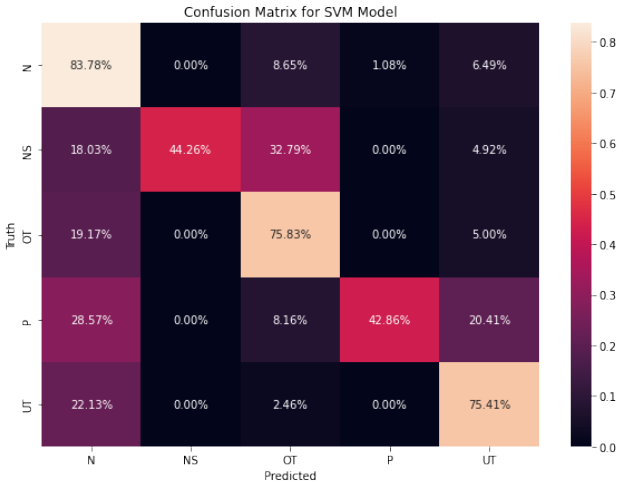


Figure 7.11: *Confusion matrix of the extrinsic SVM model*

Figure 7.6 shows the feature importance of the trained Random Forest algorithm with extrinsic data. The bar graph contains a set of top 20 best performing features of the extrinsic RF model. The

features with the most influence are the different MFCC features, mentioned in the Section 5.2.2. They provide a compact representation of the spectral characteristics of audio signals, making them effective for audio classification tasks. Also, all of the features extracted from the audio data are providing an influence in classification.

Figure 7.11 shows the confusion matrix for the best algorithm, which is in this case SVM. The algorithm showed a worse classification capability compared to the other data sources. This makes sense, because the audio was not recorded in the ideal conditions. Biggest problems in the classification were during the classification of the under-tightening, pose anomaly and over-tightening screw as the normal screw type. Other predictions showed better results.

7.3 Intrinsic + Task model

All of the baseline models show that the KNN algorithm unperformed during the classification compared to the rest. Intrinsic and Task models showed best performance with the use of SVM and RF algorithm, with RF algorithm giving the best results. Because of that the Intrinsic + Task model will focus on maximizing these algorithm's performance. As mentioned, this model includes the screwdriver intrinsic data and robot task data combined.

7.3.1 Random forest algorithm (intrinsic + task)

To achieve the most optimal results with the RF algorithm, a process called grid search was conducted. By using grid search, the optimal hyperparameters of the RF algorithm are determined. Figure 7.12 shows the 3D visual representation on how do different hyperparameters influence the model accuracy. Specific metrics of this model can be found in Appendix C.

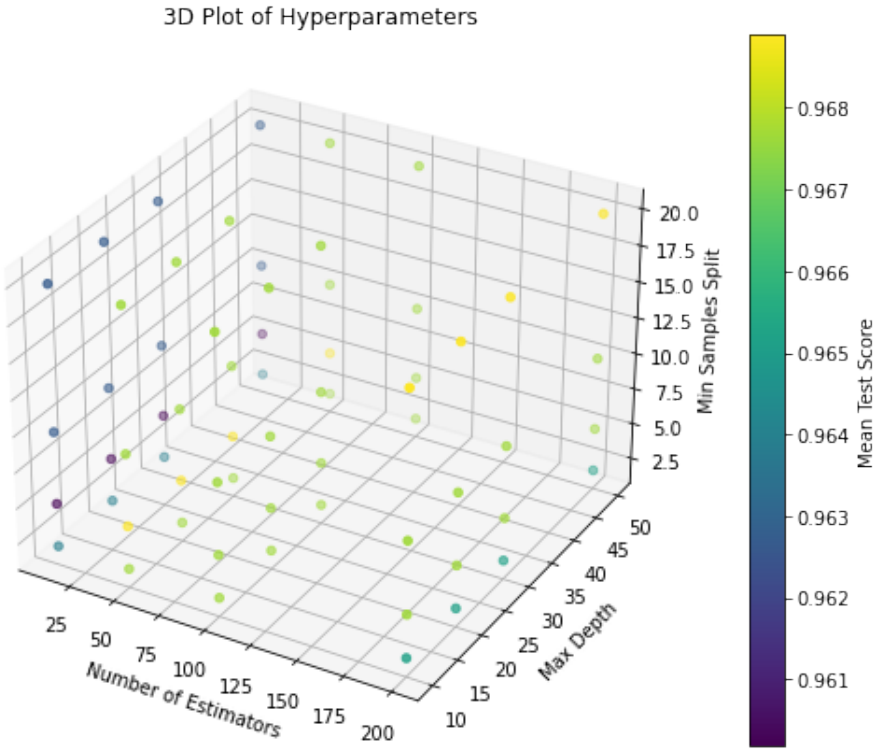


Figure 7.12: *3D graph of hyperparameter grid search of the RF model*

Algorithm settings:

From the grid search the following hyperparameter values were decided:

- N-estimators = 100
- max_depth = None
- min_samples_split = 2

This showed that using the default hyperparameters of the RF algorithm gave the best results from the start, and there is no need for further tuning.

Results

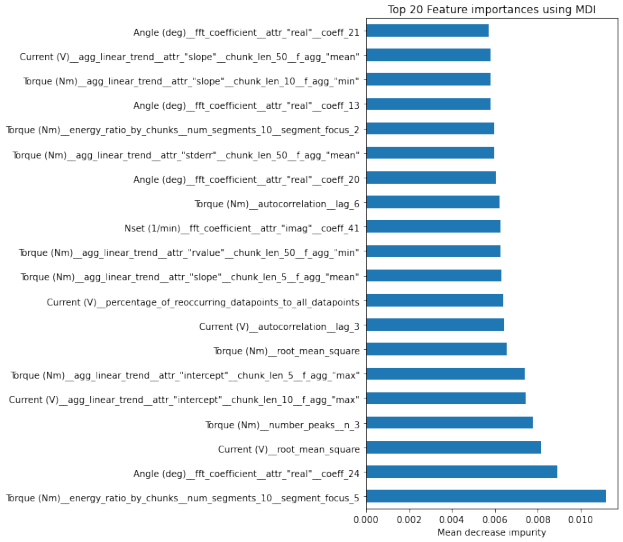


Figure 7.13: *Feature importance of the intrinsic + task model*

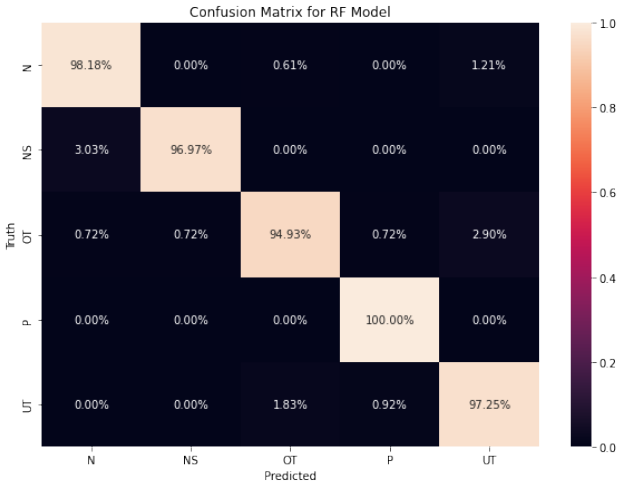


Figure 7.14: *Confusion matrix of the intrinsic + task RF model*

Figure 7.13 shows the feature importance of the trained Random Forest algorithm with intrinsic and task data combined. The bar graph contains a set of top 20 best performing features of the intrinsic + task RF model. From the features shown it is important to notice that all of the 20 features belong to intrinsic data source. This means that the data from does not have a big influence on the model, resulting in a similar accuracy score to the intrinsic model.

Figure 7.14 shows the confusion matrix for the RF algorithm. The algorithm showed a good classification capability, similar to the intrinsic model. As an example, even some missclassifications are of the same percentage, like the wrong classification of over-tightening to under-tightening of 2.9%. This can be seen by comparing the RF confusion matrix of intrinsic and task data in Figure 7.14 and RF confusion matrix of intrinsic data in Figure 7.4

7.3.2 TSFresh feature extraction settings

Since the results of the Intrinsic + Task model are giving good classification results, without making big improvements on the best performing intrinsic data baseline model, this section will explore how using a smaller number of features extracted from the TSFresh package influences the classification.

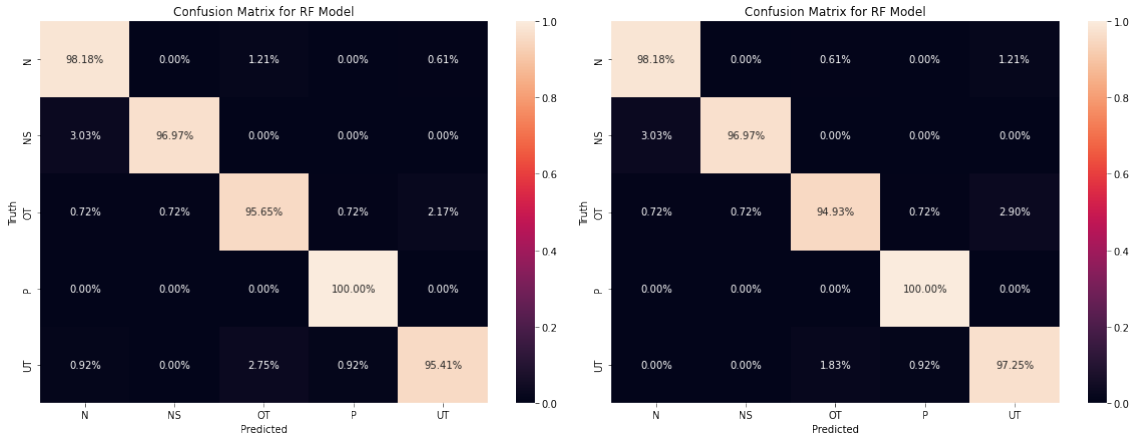
As mentioned in Section 5.2.1, the features were extracted with two TSFresh setting instances, MinimalFCParameters and EfficientFCParameters. Since the results show that for intrinsic and task data, big amount of features have little or no importance in the model performance, the plan is to

test what results would a smaller set of features provide in terms of the classification ability of the RF algorithm.

RF algorithm was trained with minimal and efficient features and the results are shown in the Table 7.5, together with the confusion matrices in Figure 7.15.

RF algorithm	accuracy	precision	recall	F1Score
MinimalFCParameters	*97.02*	*97.02*	*97.02*	*97.01*
EfficientFCParameters	*97.21*	*97.23*	*97.21*	*97.20*

Table 7.5: *Comparison of the RF algorithms for intrinsic + task model before and after cleaning*



(a) *Model with minimal features* **(b)** *Model with efficient features*

Figure 7.15: *Comparison of RF algorithm's confusion matrices of Intrinsic + Task model with different feature sets*

The models perform good in both cases, showing a strong classification capability. Since the difference in the number of features is big, this is an indicator how a vast amount of efficient features have no impact on the performance in classification.

7.4 Combined model

As the final model, the Combined model includes all of the data sources combined. This was done by combining the features of all 3 data sources together. Since intrinsic and task data performed the best on the RF algorithm, and audio data on SVM algorithm, both of them are tested with the combined data. Further information on the different algorithms can be found in Appendix D.

Before going into the models, ROC and Precision-Recall graph is presented in Figure 7.16, for the KNN, SVM and RF algorithm of the combined model.

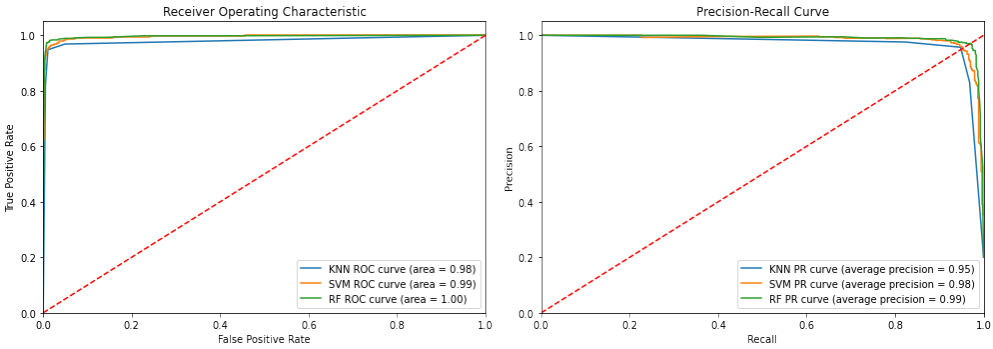


Figure 7.16: *ROC and Precision-Recall graph for combined data*

Since all of the algorithms performed good, as can be seen in the AUC measure from the curves, only the RF algorithm will be presented in the report.

7.4.1 RF algorithm (combined)

The RF algorithm again performed similar to the intrinsic baseline model. From that it can be concluded that the screwdriver data intrinsic to the process has such a big impact on the models, that other sources of data do not bring a big impact on the model's performance. Confusion matrix of the algorithm is shown in Figure 7.18.

Algorithm settings:

From the grid search the following hyperparameter values were decided:

- N-estimators = 100
- max_depth = None
- min_samples_split = 2

Results:

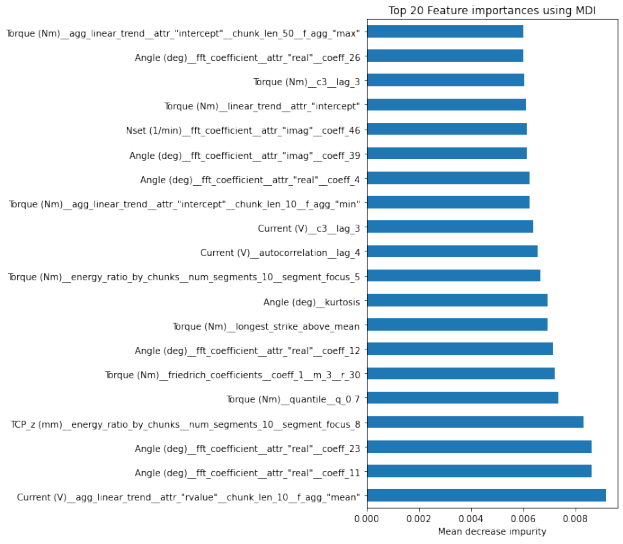


Figure 7.17: *Feature importance of the combined model*

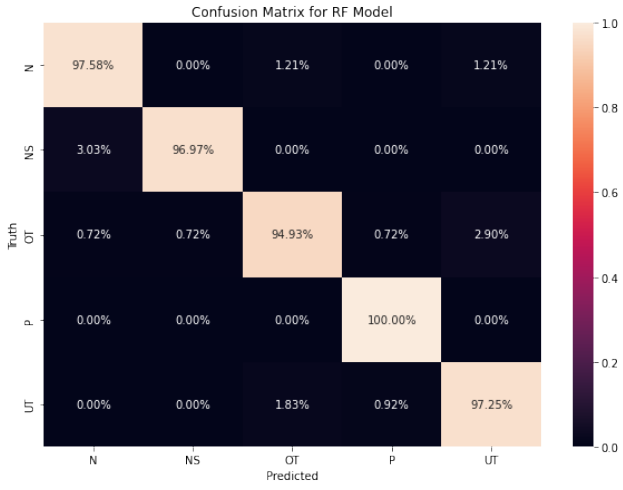


Figure 7.18: *Confusion matrix of the combined RF model*

From the Figure 7.17 it is again clear how much influence do intrinsic features provide. However one feature has a big impact, which is TCP_z energy ratio by chunks. The energy ratio could refers to the ratio of the energy in one chunk to the energy in another chunk, or to the total energy. For example, it could be used to identify whether certain periods of time (chunks) contain more "activity" (higher energy) than others. In this case it refers to the sum of the squares of the signal values.

With the results of the models, often times features related to the Tool Centerpoint Position in z-axis have a impact comparable to the intrinsic screwdriver data. This is a good indicator that from the task dataset, TCP_z is the most important value recorded.

7.5 Results summary

After presenting the results of different ML models, a summary will show what can be concluded from the results provided after the model building. The results encompassed models with different data sources and data combinations, algorithms and algorithm hyperparameters.

7.5.1 Baseline models

The baseline models tested how different algorithms classify screw classes by the use of different sources of data. All of the models showed a good classification capability, with some outperforming the others.

The best performing model, intrinsic baseline model, gave the best results with only wrongly classifying a few of the screw classes. This was to be expected, since the data intrinsic to the process should be best in giving information about the screwdriving process itself. Also, the sensors mounted on the screwdriver gave precise measurements of the torque, current and similar values. The algorithm that worked best with this data was a RF algorithm, although other algorithms also showed a good performance. After calculating the feature importance, it was concluded that a lot of features extracted with the TSFresh package did not have a big impact on the model's performance. The best performing features were features related to the torque, current and angle.

The second best performing model was the task baseline model. This model showed surprisings results with a good classification accuracy, especially with the RF algorithm. Again similar to the intrinsic baseline model, task model was able to classify the majority of instances correctly. When looking trough the top 20 features by feature importance, It was concluded that the most important feature was TCP_z, or the robots Tool Center Point in z-axis.

The third performing model was the extrinsic baseline model. This model showed a decent classification capability, but unlike the task and intrinsic models, extrinsic baseline model showed best results with the SVM algorithm. During the model building, this model was tested with the raw unprocessed "noisy" data and audio data after processing. This showed a substantial improvement on the model's performance which showed that the preprocessing was done successfully. After feature importance calculation, it was clear that the features related to the mel-frequency cepstral coefficients (MFCCs), were the most influential in the model's performance.

7.5.2 Intrinsic + Task model

This model included data from the screwdriver (intrinsic data) and data from the UR10 robot (task data) in combination. This combination was achieved by combining the two sets of features. The

main algorithm used in this model was the RF algorithm, where the hyperparameters were tuned to achieve the best performance, but it was concluded that the initial default parameters showed the best results. The results of this RF algorithm were close to same to the intrinsic baseline model. Also, the most important features included only the intrinsic screwdriver data features. This means that the features from the robot have a small impact on the final classification precision.

Because there was a large amount of features that had zero to no importance on the model, a new approach was to use a smaller subset of features from the TSFresh package. The MinimalFCParameters with less features and EfficientFCParameters with more features models were created and compared. The results showed that the model with minimal features provides a strong performance, close to the model with more features.

7.5.3 Combined model

The final models created were the combined "meta" models. These models included features of all 3 data sources with 3 algorithms, which resulted in 3 models. Again all of the models showed an excellent performance during classification, with the RF model performing best. Like it was the case with the Intrinsic + Task model, the intrinsic screwdriver data features were the most important features during classification and dictated the results of the algorithms. Because of that the performance of the combined model is similar to the performance of the intrinsic baseline model. Only one feature from the task robot data was in the top 20 features by importance, and it is again related to the TCP_z. This shows how the TCP_z feature can benefit even the models with intrinsic data source.

8 Conclusion

This chapter will give a discussion and overview on the completion of the project, by answering the Research Objectives. This will include answering the different sub questions for every objective and summarise the steps of the project. Afterwards, a final conclusion of the project will be presented, with the thoughts on the discussion and results.

As the final part of the project, further steps and improvements will be presented in project continuation segment. This segment is meant to show what could be the further uses with this project and results.

8.1 Research objectives completion

As a start of the conclusion the answers and completion of the Research Objectives, that are introduced in the Section 3.3, are provided.

Research Objective 1 (RO1)

The first research objective was to design and implement a data collection and storage system for the screwdriving cell and create a dataset for ML purposes. The majority of objectives were completed, with an exception of the simultaneous data collection. Table 8.1 shows the completion of RO1.

RO1: Design and implement a data collection and storage system for the robot screwing cell and create a new dataset which will be used for ML model training		Completion (yes/no)
·find what process data can be collected (intrinsic, extrinsic, task) and decide on which will be collected		yes
·decide on the anomaly types that will be looked into		yes
·create a data collection system which will:	·collect different types of data	yes
	·collect the data simultaneously	yes
	·collect the data in a proper format	yes
	·collect the data automatically	no
·create a labeling and backtracking structure for the data gathered which will:	·have a systematic division of data location	yes
	·sort the data by the screwdriving type (or anomaly)	yes
	·create a system for backtracking the data	yes
	·make a system where every screw has an ID, together with its data	yes
·create a new dataset by conducting a series of experiments, that will be used as the basis for training of the machine learning models		yes

Table 8.1: *Completion of Research Objectives 1*

-find what process data can be collected (intrinsic, extrinsic, task) and decide on which will be collected

The sources of data collection were screwdriver sensors (intrinsic data), UR10 robot (task data) and Azure microphone (extrinsic data). It was decided to collect data from all 3 sources after consideration. There were other possibilities, like images or vibration sensors, but the data that was decided to collect was the best fit for this project, when the available resources and timeframe are taken into account.

-decide on the anomaly types that will be looked into

The goal was to use classification for different types of screwdriving that happen during the manufacturing process. In the first iteration, there was 7 screw types, but in the end it was decided to have 5 classes of screw types, by combining three classes in to one. This was agreed upon by discussion with the supervisors. The reason for this number of anomaly types was the size of the dataset. If the dataset is expanded, more anomalies can be included into classification. The classes selected were normal screwdriving, over-tightening, under-tightening, under tightening, pose anomaly and no-screw.

-create a data collection system which will:

- *collect different types of data*

  The data collection system was created which included collection of data from 3 sources by the use of computer WSK3 program and python scripts.

- *collect the data simultaneously*

  In the ideal situation all of the data collection would happen trough the same python program. On this project, because of the setup complications and the older version of the UR10 robot's operating system this was not possible without making modifications to the screwdriving cell. To avoid that a signal from the PLC was used to synchronize the collection of the data with multiple programs. The data from the robot and microphone was collected simultaneously, but the intrinsic data from the screwdriver was collected with a separate computer program. This resulted in a small delay between the data and different sampling frequencies, but it did not have a big impact on the project results.

- *collect the data in a proper format*

  The data collection system was implemented in two stages, where the data was collected and stored in different formats. Screwdriver data is collected in a .KXML format, robot data in .csv format and audio data in .wav format. All of the formats functioned as ML training data after processing.

- *collect the data automatically*

  The data was not collected automatically. The whole process of data collection had to include a operator, which would have to step in multiple times into the process. Because of the robot often malfunctioning, implementing a automatic data collection system would not be possible.

-create a labeling and backtracking structure for the data gathering

The second stage of the data collection system included formatting, labeling and storage based on the screw ID and anomaly type. After every beam was screwdrived, the operator had to inspect every screw and provide input to the data sorting python script. This metadata can be used to backtrack to every individual screw if needed.

-create a new dataset by conducting a series of experiments, that will be used as the basis for training of the machine learning models

As the final part of the RO1, a state-of-art dataset including screwdriver, robot and microphone data was created. This was the most time-consuming part of the process, because it included a lot of trial an error during the screwdriving process. The final result was a dataset which included 1341 recorded screws with 3 sources of data for each screw.

Research Objective 2 (RO2)

The second research objective was to preprocess and clean the data which was collected in the data collection. All of the objectives were completed. Table 8.2 shows the completion of RO2.

RO2: Preprocess and clean the data collected	Completion (yes/no)
·convert the kXML files to a more manageable data format	yes
·find a way to combine different data types with different sampling frequencies and delays in the data collection process	yes
·remove the unnecessary data from the dataset	yes
·remove the noise from the data	yes
·handle missing values from the data	yes
·visualise the data	yes

Table 8.2: *Completion of Research Objectives 2*

-convert the kXML files to a more manageable data format

During the data sorting phase of data collection, the .KXML files were converted to a more managable tabular .csv format by the data sorting script. This turned to semi-structured data format to a structured tabular format, where every column of the table belonged to a sensor measurement. The data was carefully renamed and saved so that it represents the screws appropriately.

-find a way to combine different data types with different sampling frequencies and delays in the data collection process

These problems were resolved by the use of the feature extraction process. The ML algorithms are taking input in terms of values that describe the time-series dataset, and not the values in the raw format. Because of this the sampling frequency and delays had little effect on the end results of the classification with machine learning.

-remove the unnecessary data from the dataset

During the preprocessing stage, the data was cleaned and prepared for feature extraction and classification. Robot data was configured to be fixed in the same point for every screw. From the audio data the unnecessary outside "noise"was removed. All of the data was checked for irregularities and errors. Also, during the ML process, models were tested with only the relevant features, by removing the features of low importance.

-remove the noise from the data

By the use of a low-pass filter, the audio was cleaned from the background noise generated by servers and other factors. This noise was recorded, averaged and then removed from the dataset.

-handle missing values from the data

All of the data was checked for missing values or infinite values during the preprocessing stage. Also, during the feature extraction with TSFresh package, the missing values created by the program were also filled or removed, before inputing them to a ML algorithm.

-visualise the data

At the end of the data collection phase, together with during the preprocessing and feature selection phase, the data was properly visualised and analysed. This was done to check the data for problems and errors, and to analyse if there can be improvements.

Research Objective 3 (RO3)

The third research objective was to extract the relevant features and make different dataframes from the data collected for ML applications. All of the objectives were completed. Table 8.3 shows the completion of RO3.

RO3: From the data collected, extract the relevant features and make different dataframes which will be used for ML.	Completion (yes/no)
·from intrinsic, extrinsic and task data, extract the features relevant to the process	yes
·create different combinations of features and find the ones which give best results in the ML algorithms	yes
·label the data	yes

Table 8.3: *Completion of Research Objectives 3*

-from intrinsic, extrinsic and task data, extract the features relevant to the process

Intrinsic screwdriver data and task robot data had the same feature extraction procedure. Since the data was time dependant, a package for feature selection from time-series data called TSFresh was used. This resulted in automatic feature extraction of intrinsic and task data.

Extrinsic data had feature extracted as a separate process, where features that are used in speech recognition and audio analysis were extracted.

-create different combinations of features and find the ones which give best results in the ML algorithms

The features were combined in different ways, to test how do they influence the ML models. Every data source was tested as a standalone set of features for the model. Afterwards, data sources were combined together. As the final model, features of all 3 data sources were combined in to one set.

-label the data

During the data collection process, more specifically data sorting phase, the files of different sensors and data sources were stored by the class they belong to. Afterwards, a dataframe which consisted of the screw names, which also acted as metadata, and class labels was created. This dataset was used as labels for the ML models.

Research Objective 4 (RO4)

The fourth research objective was to develop a classification ML/DL models by the use of the collected data. The majority of objectives were completed, with an exception of the creation of deep learning models. Table 8.4 shows the completion of RO4.

RO4: Develop a classification ML/DL models using the data from the screwing cell	Completion (yes/no)
·decide on the algorithms that will be used in classification	yes
·create a system for measuring the results of different ML models	yes
·test separately intrinsic extrinsic and task data and compare the accuracy results of classification with different ML algorithms	yes
·test different combinations of features, and find the best "fit" for classification	yes
·create a deep learning model with separate or combined data types and measure the results gathered	no

Table 8.4: *Completion of Research Objectives 4*

-decide on the algorithms that will be used in classification

For the classification task 3 different ML algorithms were implemented for all of the models. The algorithms are K-Nearest Neighbours (KNN), Support Vector Machines (SVM) and Random Forest (RF). These algorithms all belong to a different class of ML algorithms.

-create a system for measuring the results of different ML models

In the beginning of the Results chapter, a system for measuring results was elaborated. The most important metrics for the comparisons were the confusion matrices in combination with areas under the ROC and Precision-Recall curves.

-test separate intrinsic, extrinsic and task data and compare the accuracy results of classification with different ML algorithms

Every data source had 3 models containing different ML algorithms. This resulted in 9 baseline models. After testing it was concluded that intrinsic data source provided the best accuracy during classification, which was followed by task data source as second and extrinsic data source as the third. During the baseline models creation different parameters were used to improve models performance, but with not a lot of changes. By doing this the influence of different data sources on the accuracy of classification was tested.

-test different combinations of features, and find the best "fit" for classification

By combining the features of different data sources, the models yielded a minimal improvements in accuracy. In general, when data sources were combined, the data source that presented better accuracy by itself benchmarked the accuracy of combined models. For that reason the model that combines all 3 features had a similar outcome as the best performing intrinsic baseline model.

-create a deep learning model with separate or combined data types and measure the results gathered

After the results gathered from machine learning, there was not time to develop deep learning models. Part of the reason for that was that the whole structure of the report focused on multiple aspects of the whole data collection to machine model building process. The deep learning models were tested, but not introduced to the report, but they did show a promising start to the further development of the anomaly detection systems, especially on the audio data.

8.2 Discussion

After the research objectives, a discussion will cover the project segments, including the data collection, machine learning and results. Toughs on every segment will be presented, ending with the result interpretation.

8.2.1 Data collection

Data collection was one of the most important parts of the project, and also the most time-consuming part. Getting the correct data and building a dataset which is properly labeled is one of the key goals of a project like this that involves classification machine learning.

Screwdriving cell setup

The setup of the screwdriving cell is complicated, including a lot of different parts. The UR10 robot created a lot of problems with the connection and malfunctions. As a start the screwdriver

mounted on the robot is too heavy, which during the screwdriving operations often causes the robot to displace and shut down. The older operating system of the robot only allowed collection of 7 sensor measurements, by the use of Modbus. This proved to be sufficient, but the newer models can use more advanced communication protocols, which would yield more data with a higher sampling frequency. Also, the robot moves during the screwdriving process across the beam. The further the robot moves away from it's base, the more displacement it presents. This could be a problem in classification, if the algorithms recognise this displacement and classify the data based on that. However, this is just an observation and it is possible that it has no influence. A robot with a more advanced operating system and a higher payload capability could yield a dataset which represents the manufacturing process better.

Microphone setup

As mentioned, the robot already had problems with the screwdriver weight, so placing the microphone on it was not a possibility. The current microphone setup with 3D printed stands proved to be an effective solution for audio collection. However, the placed of the microphone on robot would provide a fixed position of the microphone for every screw to be the same. Currently every screw was distanced from the microphone, depending on its position on the wood beam. Also, Azure Kinect microphone is used for speech recognition and computer vision since it is primarily a camera. Using an industrial microphone for recording screwdriving audio would be a better solution, but also a more expensive one.

Simultaneous data collection

As explained in the report, the data collection of 3 sources was synchronised with the use of a strobe signal from the PLC. This resulted in a small delay between the screwdriver intrinsic data and the robot and microphone data. Currently intrinsic data is recorded over the WSK3 program, and task and extrinsic data is recorded over a python script. To collect all of the data over the python script, a reconfiguration of the setup should be made, by sending the data from the screwdriver controller to the PLC, and getting it from the PLC together with the task data.

Final dataset

In general final dataset creation was successful, considering the different challenges during the data collection process. The final dataset contains 1341 screws, with 3 data sources for each one of them. Collecting a bigger dataset would give a better representation of the different classes, but since the classes were properly balanced, the results of ML should be representative.

The dataset contains 5 screw classes. For bigger class variance, there could be additional screw types introduced to the dataset. The current dataset of 1341 screws should be enhanced with new screws, to introduce new classes, or there will be not enough examples for classification.

8.2.2 Machine learning

This discussion segment includes the preprocessing, feature selection and machine learning segments of the report. After the final dataset was collected, properly labeled and checked, it was time for data analysis by the use of ML/DL.

Preprocessing and data cleaning

The data from the sensors, robot and microphone was already collected in a proper format in the final dataset. It also did not require a lot of preprocessing and cleaning before the feature extraction process. The most important part of the preprocessing was cleaning the audio from the outside noise that was created by the scrwdriving cell's proximity to 5G server, together with the noise created by

the pneumatic system of the screwdriver and test beam setup. This noise is also expected in the industrial environment, so dealing with this noise will be a standard process in the manufacturing process.

Feature selection

Features of the intrinsic and task data were selected and extracted by the TSFresh library. This library provided a comprehensive set of features for both datasets, which reflected in the results. This is just one of many options for feature extraction, but it contains the usual features used in describing time-series data.

Librosa python library was selected for feature extraction of extrinsic audio data. The features were extracted, based on features often used for sound analysis, especially speech recognition.

Model building

Models were build with separate data sources or combinations between different data sources. There is a vast amount of different algorithms for machine learning, and this report only encompasses 3 of them. Since the results of the models gave high accuracy, the only model that would need more improvement would be the extrinsic model that analyses and classifies the audio.

With the current algorithms, changing their hyperparameters did not yields better performances of the models. The reason for this was that the available computational power was not sufficient in testing a lot of different parameter variations. If there was more time, a lot of different parameters could be tested to improve model performance, but it would not guaranty the improvement.

The next step would be to build deep learning models. No deep learning model has been introduced to this project, but they did show promising results, especially a Convolutional Neural Network (CNN) model of extrinsic data.

8.2.3 Results

All of the algorithms provided a good classification capability, with the combined model providing the best accuracy across the board. It showed best results, but almost the same as the Intrinsic + Task model and intrinsic model. This is due to intrinsic features having a overwhelming importance in classification.

Baseline models

In the baseline models, intrinsic model as mentioned showed the best results which was to be expected. The model is using precise data intrinsic to the process, with a high sampling frequency. The surprises in the results come with the task model and extrinsic model.

The task model results are surprising, because the UR10 robot had a lot of problems and malfunctions during the process of screwdriving, and could only output data in a sampling frequency of 400 Hz. With this high accuracy, the need for all of the sensors on the screwdriver becomes redundant, since gathering the data from the robot is free compared to the installation of expensive sensors. Because of this, it will be a good idea to create additional experiments on a different robots, which would confirm that the classification was successful, and that the algorithms are not recognising something in the data related to the experiment setup, and not the screw class.

The extrinsic models showed good results, when the microphone setup and the microphone are taken in to consideration. Like with the task data it will be important to test this in a different setup or environment. Implementation of microphones would also be a cheaper option for anomaly detection if successful, then the installation of sensors.

Intrinsic + Task model

When combining two data sources, the results of the classification will end up similar to the baseline model of the data source which had better accuracy. Even if this is true, the model did outpreform its baseline counterparts by a small margin, so combining the data would not be a bad practise, especially if the implementation would have little or no cost. If the dataset is larger and there is more classes, using more sources of data would make the model more robust and possibly better.

Using the RF algorithm provided the best results, but setting different hyperparameters on the algorithm ,then the basic ones, just lowered the accuracy of the model.

By the use of MinimalFCParameters, it was shown that a smaller subset of basic features worked similarly to the EfficientFCParameters set of TSFresh features. For the model implementation this would mean less computational power will be needed when extracting features from the incoming data. It is important to find a balance where the best results are achieved without the need for a vast amount of features, which is done by the use of feature importance. With the feature importance, the features that have zero to none influence in the model can be discarted.

Combined model

The final results provided were for the combined model which included 3 data sources combined. Again all of the algorithms performed exceptionally well on the classification task, with the RF algorithm providing the best classification results. This model provided the best classification accuracy, but by a slim margin more then its intrinsic baseline model counterpart. Like the Intrinsic + Task model, this model showed that the intrinsic features provide best classification and overwhelm the other data sources.

Even if intrinsic data provided most of the models performance, an observation after the calculation of the feature importances of different models was that the Tool Center Point in z-axis (TCP_z) measurement, and it's features from the task data source are of big importance even when combined with the intrinsic data. For the extrinsic data that feature is the mel-frequency cepstral coefficients (MFCCs).

8.3 Final conclusion

The goal of this thesis was to create anomaly detection models with machine learning, which will be able to classify different types of anomalies that happen during the automated screwdriving process. Idea was to enter a replica of the VELUX manufacturing screwdriving cell, with similar setup to the setup of the company, pass the machine learning pipeline where the data is collected from different sources, processed and inputted into different machine learning algorithms as anomaly detection tools in classification. The hypothesis of the project was:

„By utilizing machine learning on the different process data of the automated screwdriving process it is possible to effectively identify deviations, irregularities in the performance of the system, and use this information as a reliable anomaly detection tool."

The hypothesis is proved with the results of classification. In the end algorithms were successful in recognising which anomaly belongs to which class of the screw.

This project gives an overview on the different parts of the machine learning pipeline. The data collection part yileded a diverse dataset containing 1341 screws, with the each screw containing 3 data sources. This data was visualised and analysed.

Afterwards, different steps needed to preprocess and clean the data were implemented. This resulted in a clean dataset, whic is ready for machine or deep learning applications. For the machine learning a way to select features was provided for every data source.

As the final result, the classification of the screw types was conducted by the use of machine learning algorithms. The results showed that the use of any data source tested would give good classification results. Intrinsic and task data showed an excellent anomaly detection capability, while the extrinsic data is able to classify a majority of the screws to a correct class.

8.4 Project continuation

This section will give an overview on what could be done to improve upon the results of the project, and what would that mean for the anomaly detection of the VELUX manufacturing, and research of anomaly detection with different data sources.

8.4.1 Expanding the data sources

In this report only 3 sources have been touched upon. Inside of these sources of data additional information could be gathered, especially from the UR10 robot. The current data was collected with keeping in mind the robot communication limitations. For newer version of the robots operating system a lot more data could be collected.

When thinking about additional data sources that could be collected there is a variety of options. The one that would seem beneficial for the process of screwdriving would be collection of data with vibration sensors. Vibration sensors are not expensive, but could provide an excellent information regarding the process of screwdriving.

Implementation of computer vision systems, where the use of blob detection from the screw images could give additional information on the screw types. Images however would not be able to distinguish if the threading has been broken in case of over-tightening, so most likely it would missclassify normal screws as over-tightened and vice-versa. This would mean that this data source should be combined with some other type of data.

There is more possibilities that would for example involve the use of x-ray images. But this, like the implementation of cameras, would require a lot of adjustments and fine-tuning of the manufacturing environment. It would also require a lot of expenses for the equipment.

8.4.2 Creation of deep learning models

Developement and use of deep learning models on time-series data, expecially convolutional neural networks (CNNs) and recurrent neural networks (RNNs), has provent to be exceptionally effective at discovering and understanding of complex patterns within the data. This means that the models could potentially uncover deeper and intricate relationships between the different data sources tha the simpler machine learning models might have overlook. First looks into these models is proving to be likely capable of enhancing the accuracy of the screw type classification.

As an example, using Long-Short-Term Memory (LSTM), which is a variant of RNN, could allow the system to learn not just from the current data point, but also from the points before it, which would create a memory of past patterns. In a process like screwdriving, a time dependant process where it is important to capture different parts of the process in time, this could be beneficial.

Using deep learning could also open the possibility of transfer learning. This would give an opportunity of the use of pre-trained models on a vast amount of datasets on this specific case. This way an extensive dataset would not be needed.

The problem with deep learning from the dataset currently amased is that deep learning models ofter require big datasets compared to simpler machine learning models, so implementation of deep learning will depend on the resources available.

8.4.3 Use of unsupervised learning

By the use of unsupervised learning models, like dimensionality reduction techniques and clustering algorithms, could provide a new approach in understanding the data from the process. It could be used in detection of latent structures, patterns or groups in the data. This could be used to refine the screw type classification. As an example, the use of clustering could help to identify previously unknown anomalies or types of screws. This would lead to improvements on the classification.

Another use of unsupervised learning is in feature learning or extraction. The algorithms have a capability to learn new sets of features from the input data, which could prove more informative for classification then the current features. This would include techniques like autoencoders.

8.4.4 Model deployment

Deploying the models involves a process where the trained models become available for use in broader systems and applications. To do that the model would need to be put into an application program interface (API) that would allow it to receive data, process data and create predictions.

For a manufacturing setting like VELUX's manufacturing line, deploying the model could involve embedding the model into the existing industrial control systems. To achieve this, a collaboration with control engineers and IT professionals would be needed to ensure the model can operate in real-time and that it has the necessary computational resources, and can interface correctly with other systems.

Model deployment is not the end of the process. After the model is in use, it will require continual monitoring of performance. Sometimes models may degrade because of the change of the underlying process, which is known as concept drift. A robust monitoring system would allow the operator to track the performance of the model and tune it if or when needed.

Bibliography

[1] A. Lele. "Artificial Intelligence (AI). In: Disruptive Technologies for the Militaries and Security. Smart Innovation, Systems and Technologies". In: *vol 132. Springer* (2019).

[2] Azure. *Azure Kinect DK*. https://azure.microsoft.com/en-us/products/kinect-dk. Accessed on 2023-08-05. May 2023.

[3] B. Buchmeister, I. Palcic, R. Ojstersek. *Artificial Intelligence in Manufacturing Companies and Broader: An Overview*. 2019.

[4] A. L. Blazej Leporovski Daniella Tola. "AURSAD: Universal Robot Screwdriving Anomaly Detection Dataset". In: *Research Gate* (2021).

[5] S. Electric. *What is Modbus and How does it work?* https://www.se.com/us/en/faqs/FA168406/. Accessed on 2023-11-05. May 2023.

[6] I. El Naqa, M.J. Murphy. *Machine Learning in Radiation Oncology*. 2015.

[7] I. H. Sarker. "Machine Learning: Algorithms, Real-World Applications and Research Directions". In: *SN Computer Science* (2021).

[8] Librosa. *Librosa*. https://librosa.org/doc/latest/index.html. Accessed on 2023-23-05. May 2023.

[9] M. Matsumura, S. Itou, H. Hibi. "Tightening torque estimation of a screw tightening robot". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (1995).

[10] M. Zakeriharandi. *Powerpoint slides provided by supervisor*. 2023.

[11] MathWorks. *How Machine Learning Works*. https://www.mathworks.com/discovery/machine-learning.html. Accessed on 2023-25-04. Apr. 2023.

[12] N. Kehtarnavaz. *Digital Signal Processing System Design (Second Edition)*. 2008.

[13] Python. *threading — Thread-based parallelism*. https://docs.python.org/3/library/threading.html. Accessed on 2023-11-05. May 2023.

[14] Reuben M. Aronson, Ankit Bhatia, Zhenzhong Jia. "Data-driven Classification of Screwdriving Operations". In: *Robotics Institute, Carnegie Mellon Universit* ().

[15] U. Robots. *UR10e*. https://www.universal-robots.com/da/produkter/ur10-robot/. Accessed on 2023-20-04. Apr. 2023.

[16] S. García, J. Luengo, F. Herrera. "Tutorial on practical tips of the most influential data preprocessing algorithms in data mining". In: *Knowledge-Based Systems* (2016).

[17] S. Raschka. "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning". In: *University of Wisconsin–Madison* (2018).

[18] Scikit-learn. *KNeighborsClassifier*. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. Accessed on 2023-26-05. May 2023.

[19] Scikit-learn. *Machine Learning in Python*. https://scikit-learn.org/stable/. Accessed on 2023-30-05. May 2023.

[20] Scikit-learn. *RandomForestClassifier*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed on 2023-26-05. May 2023.

[21] Scikit-learn. *StandardScaler*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Accessed on 2023-29-05. May 2023.

[22] Scikit-learn. *SVC*. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. Accessed on 2023-26-05. May 2023.

[23] S. K. Smith. "Use of microprocessor in the control and monitoring of air tools while tightening thread fasteners". In: *In Proceedings of the Society of Manufacturing Engineers* (1980).

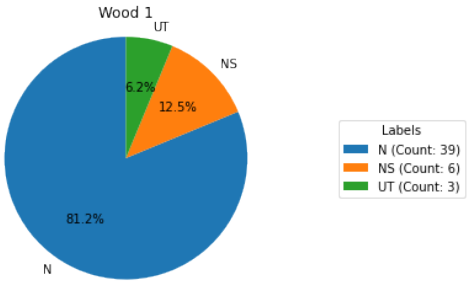
[24] J. Sprovieri. *New technology for automatic screwdriving*. https://www.mathworks.com/discovery/machine-learning.html. Accessed on 2023-25-04. Apr. 2023.

[25] tsfresh. *tsfresh*. https://tsfresh.readthedocs.io/en/latest/text/introduction.html. Accessed on 2023-22-05. May 2023.

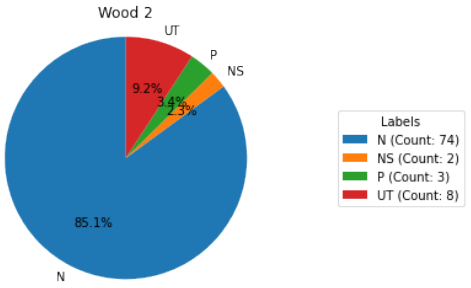
[26] *VELUX*. https://www.velux.dk/. Accessed on 2023-19-04. Apr. 2023.

[27] Weber. *Weber ZEL Feeding system*. https://www.weber-online.com/en/feeding-systems/step-feeder-zel/. Accessed on 2023-20-04. Apr. 2023.

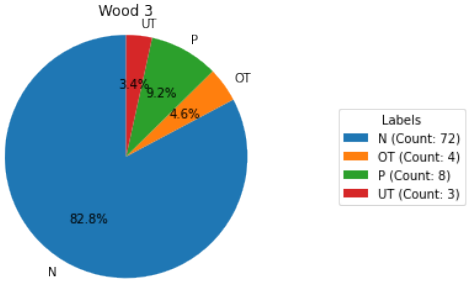
A Class distribution of wood profiles



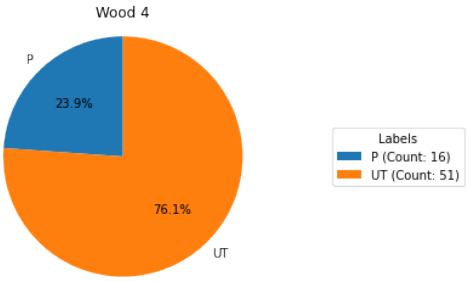
(a) *Wood 1 class distribution*



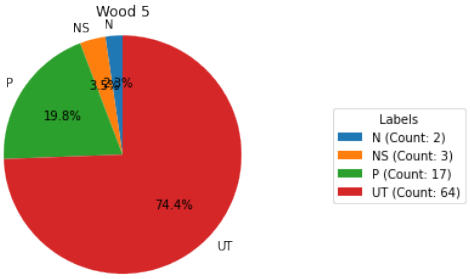
(b) *Wood 2 class distribution*



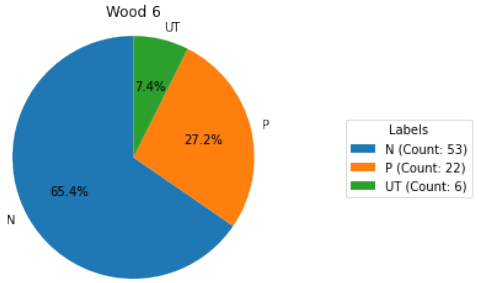
(c) *Wood 3 class distribution*



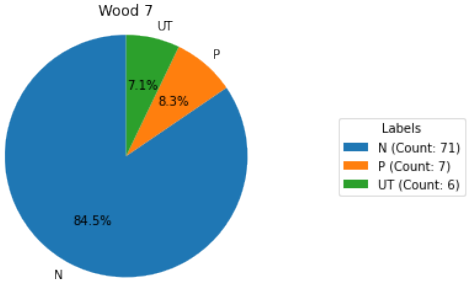
(d) *Wood 4 class distribution*



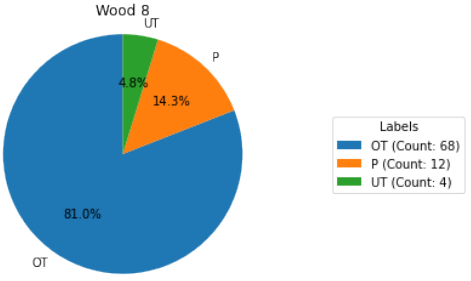
(e) *Wood 5 class distribution*



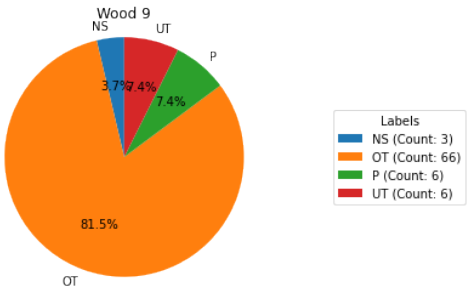
(f) *Wood 6 class distribution*



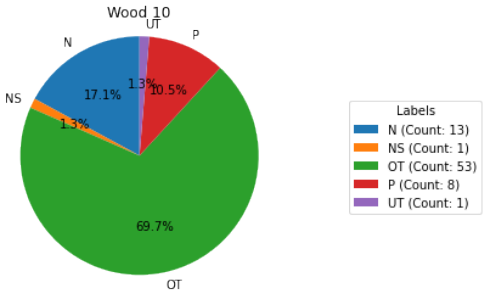
(g) *Wood 7 class distribution*



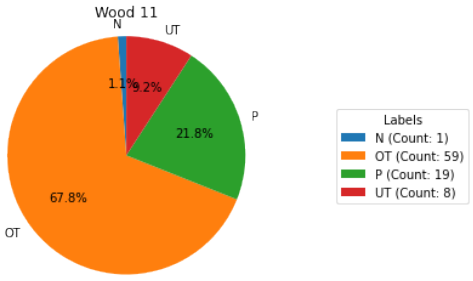
(h) *Wood 8 class distribution*



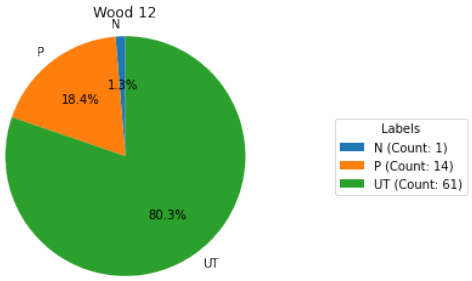
(a) *Wood 9 class distribution*



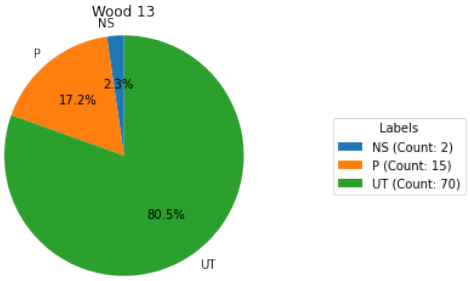
(b) *Wood 10 class distribution*



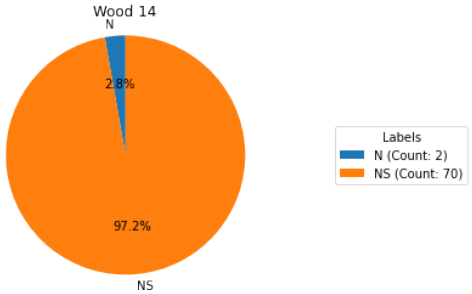
(c) *Wood 11 class distribution*



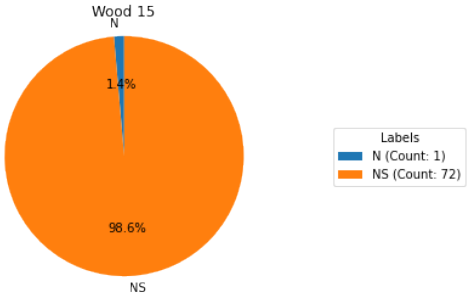
(d) *Wood 12 class distribution*



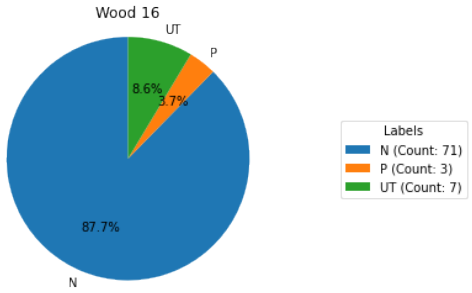
(e) *Wood 13 class distribution*



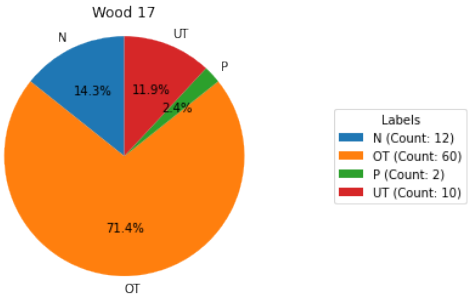
(f) *Wood 14 class distribution*



(g) *Wood 15 class distribution*



(h) *Wood 16 class distribution*



(i) *Wood 17 class distribution*

B Baseline models

The accuracy of the confusion matrices is shown in instances of true or false predictions and not in percentages, like the matrices from the report Results Chapter.

B.1 Intrinsic baseline model

B.1.1 KNN algorithm (intrinsic)

	precision	recall	F1-score	support
N	*0.98*	*0.93*	*0.96*	*165*
NS	*0.88*	*0.98*	*0.93*	*66*
OT	*0.92*	*0.96*	*0.94*	*138*
P	*0.96*	*0.86*	*0.91*	*59*
UT	*0.94*	*0.94*	*0.94*	*109*
accuracy			**0.94**	**537**
macro average	**0.94**	**0.94**	**0.94**	**537**
weighted average	**0.94**	**0.94**	**0.94**	**537**

Table B.1: *KNN classification report of intrinsic model*

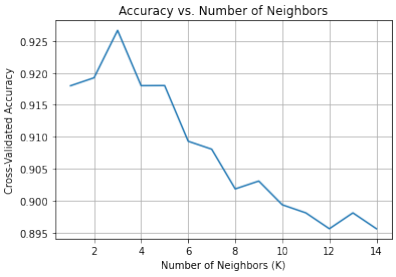


Figure B.1: *Cross-validation of the optimal neighbor number (n=3)*

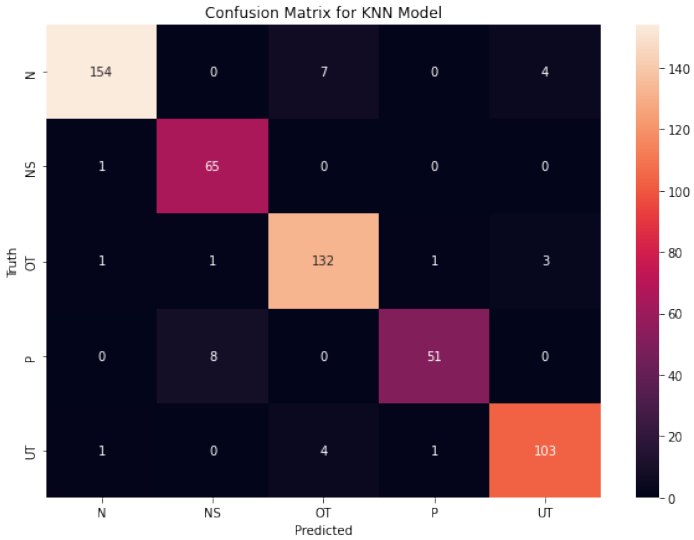


Figure B.2: *Confusion matrix of intrinsic KNN model*

Accuracy: 94.04%

B.1.2 SVM algorithm (intrinsic)

	precision	recall	F1-score	support
N	*0.98*	*0.94*	*0.96*	*165*
NS	*0.97*	*1.00*	*0.99*	*66*
OT	*0.94*	*0.94*	*0.94*	*138*
P	*0.97*	*0.98*	*0.97*	*59*
UT	*0.93*	*0.96*	*0.95*	*109*
accuracy			**0.94**	**537**
macro avg	**0.96**	**0.97**	**0.96**	**537**
weighted avg	**0.96**	**0.96**	**0.96**	**537**

Table B.2: *SVM classification report of intrinsic model*

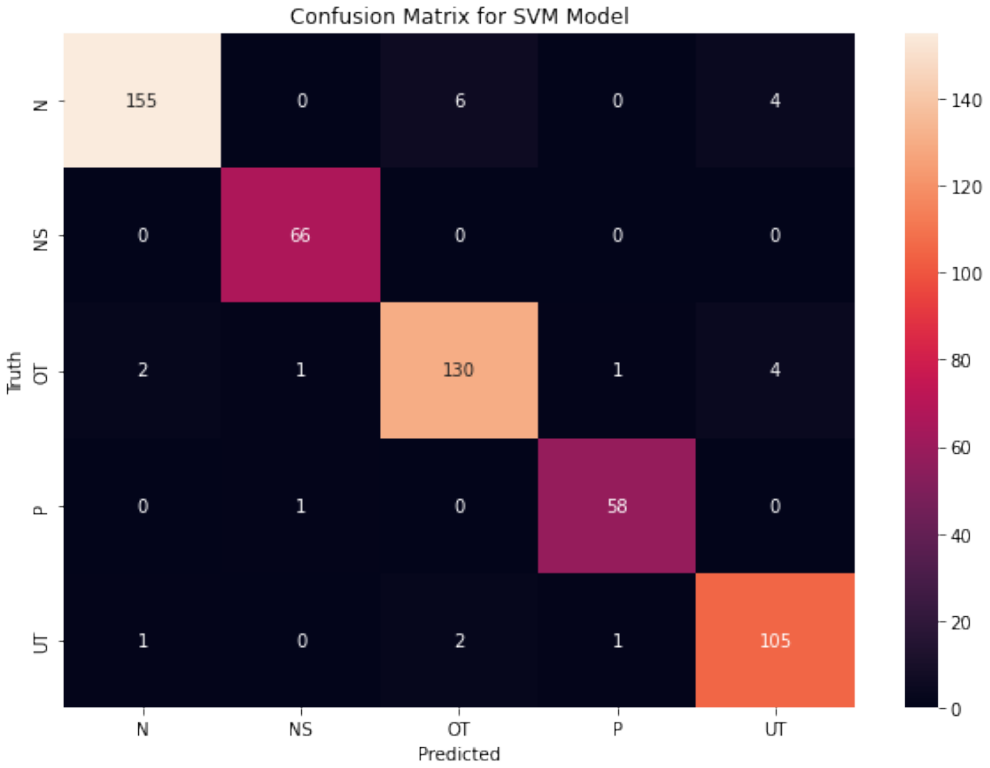


Figure B.3: *Confusion matrix of intrinsic SVM model*

Accuracy: 95.71%

B.1.3 RF algorithm (intrinsic)

	precision	recall	F1-score	support
N	*0.99*	*0.96*	*0.98*	*165*
NS	*0.99*	*1.00*	*0.99*	*66*
OT	*0.98*	*0.94*	*0.96*	*138*
P	*0.97*	*1.00*	*0.98*	*59*
UT	*0.92*	*0.97*	*0.95*	*109*
accuracy			**0.97**	**537**
macro avg	**0.97**	**0.98**	**0.97**	**537**
weighted avg	**0.97**	**0.97**	**0.97**	**537**

Table B.3: *RF classification report of intrinsic model*

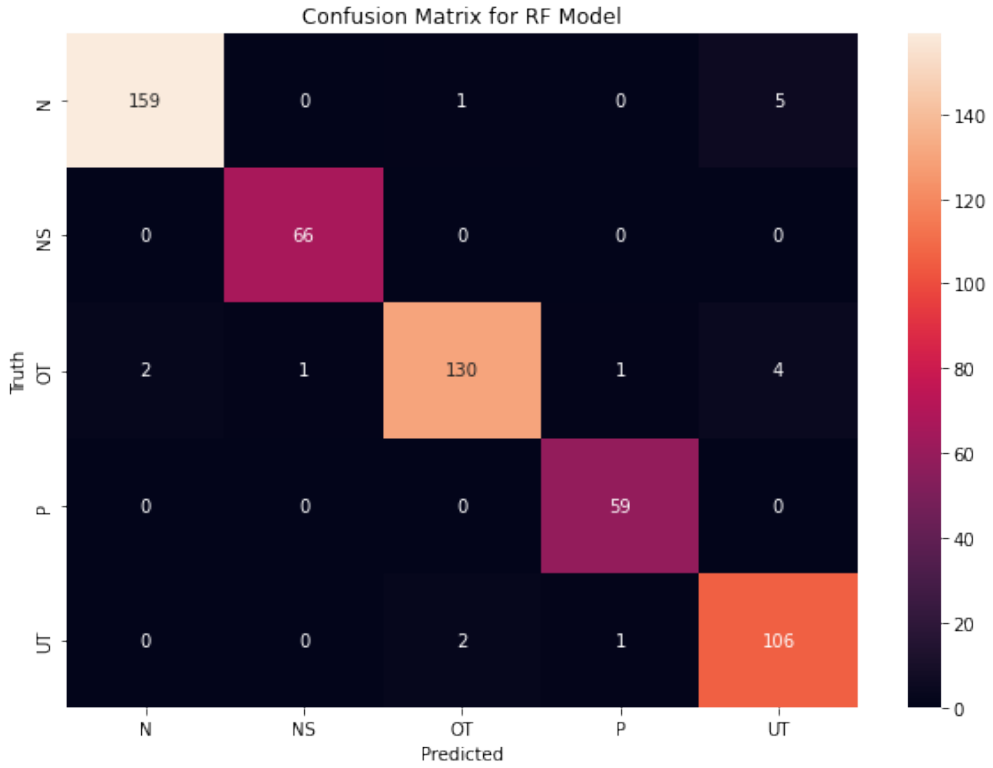


Figure B.4: *Confusion matrix of intrinsic RF model*

Accuracy: 96.83%

B.2 Task baseline model

B.2.1 KNN algorithm (task)

	precision	recall	F1-score	support
N	*0.92*	*0.92*	*0.92*	*165*
NS	*0.98*	*0.98*	*0.98*	*66*
OT	*0.88*	*0.92*	*0.90*	*138*
P	*0.97*	*0.98*	*0.97*	*59*
UT	*0.97*	*0.90*	*0.93*	*109*
accuracy			**0.93**	**537**
macro avg	**0.94**	**0.94**	**0.94**	**537**
weighted avg	**0.93**	**0.93**	**0.93**	**537**

Table B.4: *KNN classification report of task model*

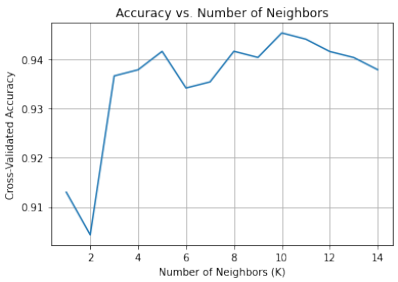


Figure B.5: *Cross-validation of the optimal neighbor number (n=10)*

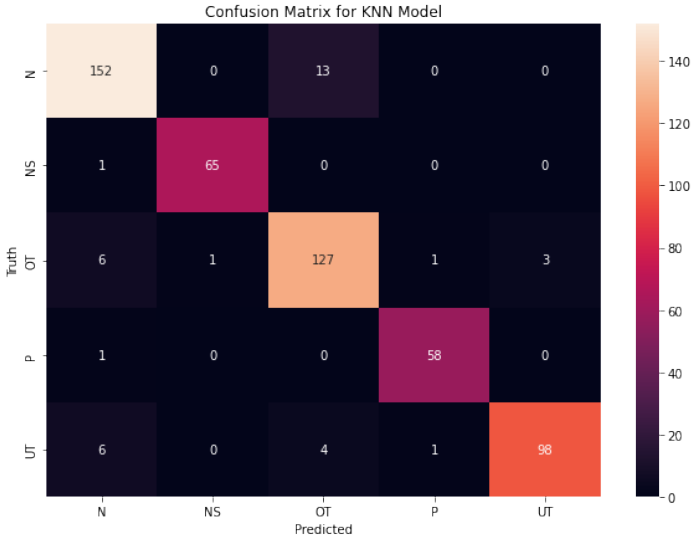


Figure B.6: *Confusion matrix of task KNN model*

Accuracy: 93.11%

B.2.2 SVM algorithm (task)

	precision	recall	F1-score	support
N	*0.97*	*0.89*	*0.93*	*165*
NS	*0.98*	*0.95*	*0.97*	*66*
OT	*0.85*	*0.93*	*0.89*	*138*
P	*0.89*	*1.00*	*0.94*	*59*
UT	*0.96*	*0.91*	*0.93*	*109*
accuracy			**0.93**	**537**
macro avg	**0.93**	**0.94**	**0.93**	**537**
weighted avg	**0.93**	**0.93**	**0.93**	**537**

Table B.5: *SVM classification report of task model*

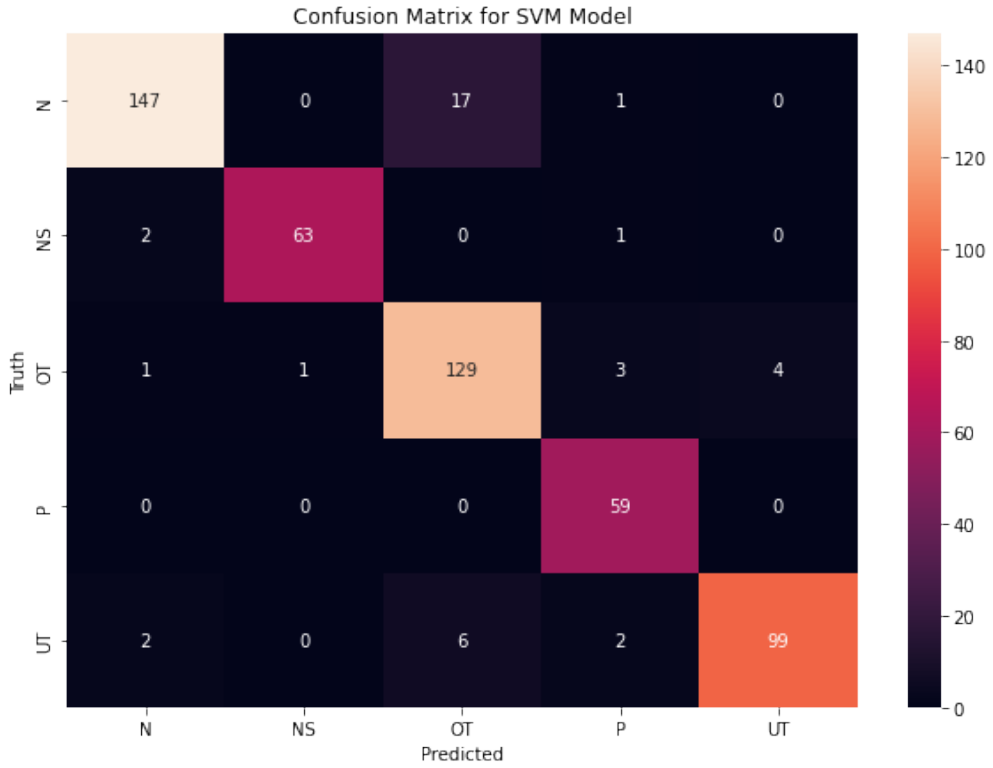


Figure B.7: *Confusion matrix of task SVM model*

Accuracy: 92.55%

B.2.3 RF algorithm (task)

	precision	recall	F1-score	support
N	*0.95*	*0.94*	*0.95*	*165*
NS	*0.98*	*0.97*	*0.98*	*66*
OT	*0.92*	*0.93*	*0.93*	*138*
P	*0.97*	*0.98*	*0.97*	*59*
UT	*0.94*	*0.94*	*0.94*	*109*
accuracy			**0.95**	**537**
macro avg	**0.95**	**0.95**	**0.95**	**537**
weighted avg	**0.95**	**0.95**	**0.95**	**537**

Table B.6: *RF classification report of task model*

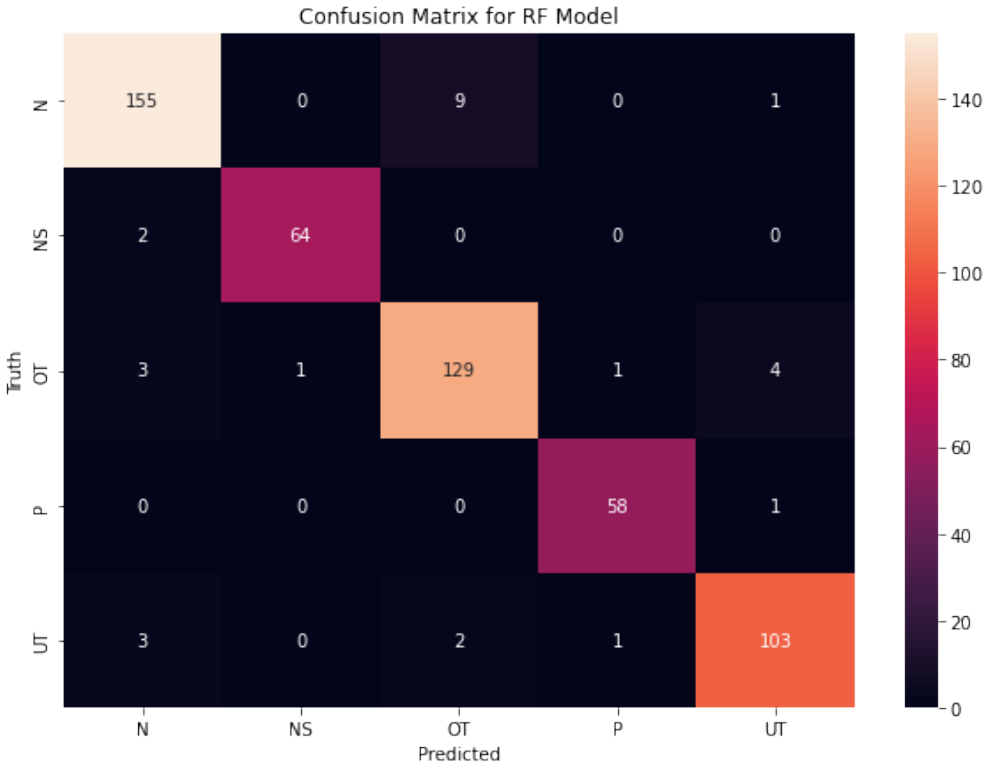


Figure B.8: *Confusion matrix of task RF model*

Accuracy: 94.78%

B.3 Extrinsic baseline model

B.3.1 KNN algorithm (extrinsic)

	precision	recall	F1-score	support
N	*0.56*	*0.83*	*0.67*	*162*
NS	*0.84*	*0.75*	*0.80*	*65*
OT	*0.59*	*0.66*	*0.63*	*125*
P	*0.68*	*0.29*	*0.41*	*58*
UT	*0.82*	*0.47*	*0.60*	*127*
accuracy			**0.64**	**537**
macro avg	**0.70**	**0.60**	**0.62**	**537**
weighted avg	**0.68**	**0.64**	**0.63**	**537**

Table B.7: *KNN classification report of extrinsic model*

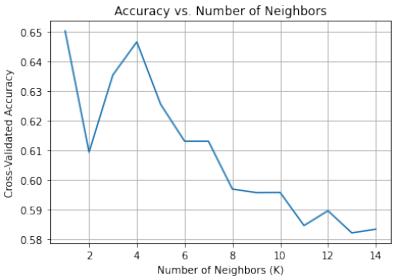


Figure B.9: *Cross-validation of the optimal neighbor number (n=4)*

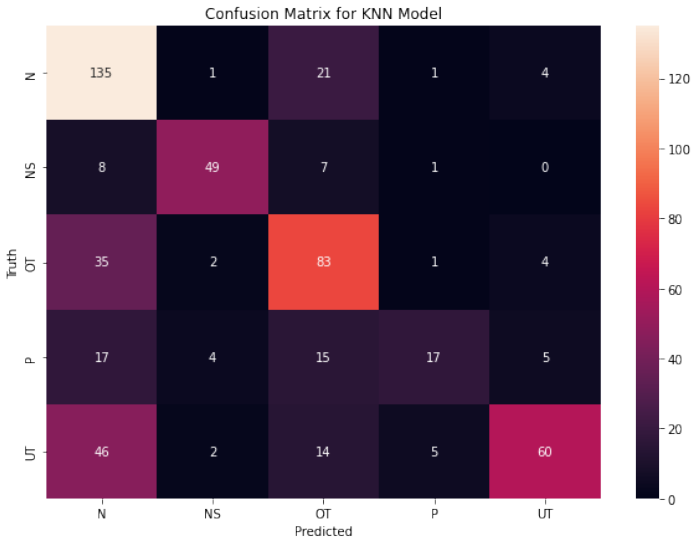


Figure B.10: *Confusion matrix of extrinsic KNN model*

Accuracy: 65.04%

B.3.2 SVM algorithm (extrinsic)

	precision	recall	F1-score	support
N	*0.69*	*0.92*	*0.79*	*162*
NS	*1.00*	*0.71*	*0.83*	*65*
OT	*0.75*	*0.81*	*0.78*	*125*
P	*0.90*	*0.47*	*0.61*	*58*
UT	*0.87*	*0.76*	*0.82*	*127*
accuracy			**0.78**	**537**
macro avg	**0.84**	**0.73**	**0.77**	**537**
weighted avg	**0.81**	**0.78**	**0.78**	**537**

Table B.8: *SVM classification report of extrinsic model*

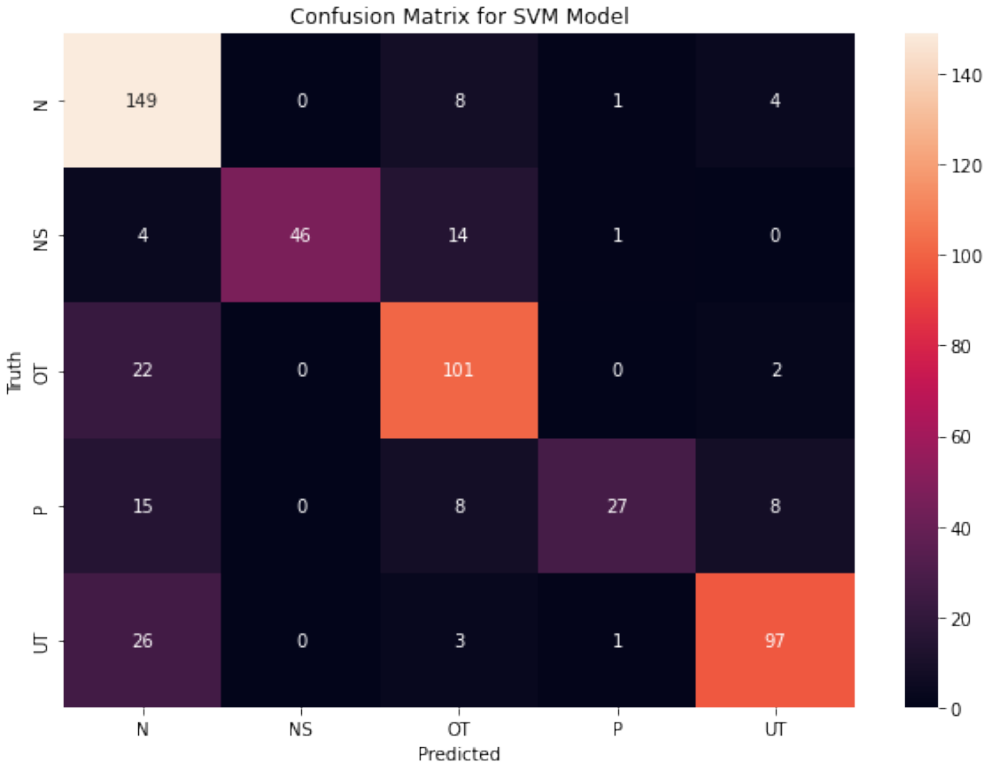


Figure B.11: *Confusion matrix of extrinsic SVM model*

Accuracy: 78.21%

B.3.3 RF algorithm (extrinsic)

	precision	recall	F1-score	support
N	*0.70*	*0.91*	*0.79*	*162*
NS	*1.00*	*0.69*	*0.82*	*65*
OT	*0.72*	*0.74*	*0.73*	*125*
P	*0.90*	*0.45*	*0.60*	*58*
UT	*0.87*	*0.85*	*0.86*	*127*
accuracy			**0.78**	**537**
macro avg	**0.84**	**0.73**	**0.76**	**537**
weighted avg	**0.80**	**0.78**	**0.78**	**537**

Table B.9: *RF classification report of extrinsic model*

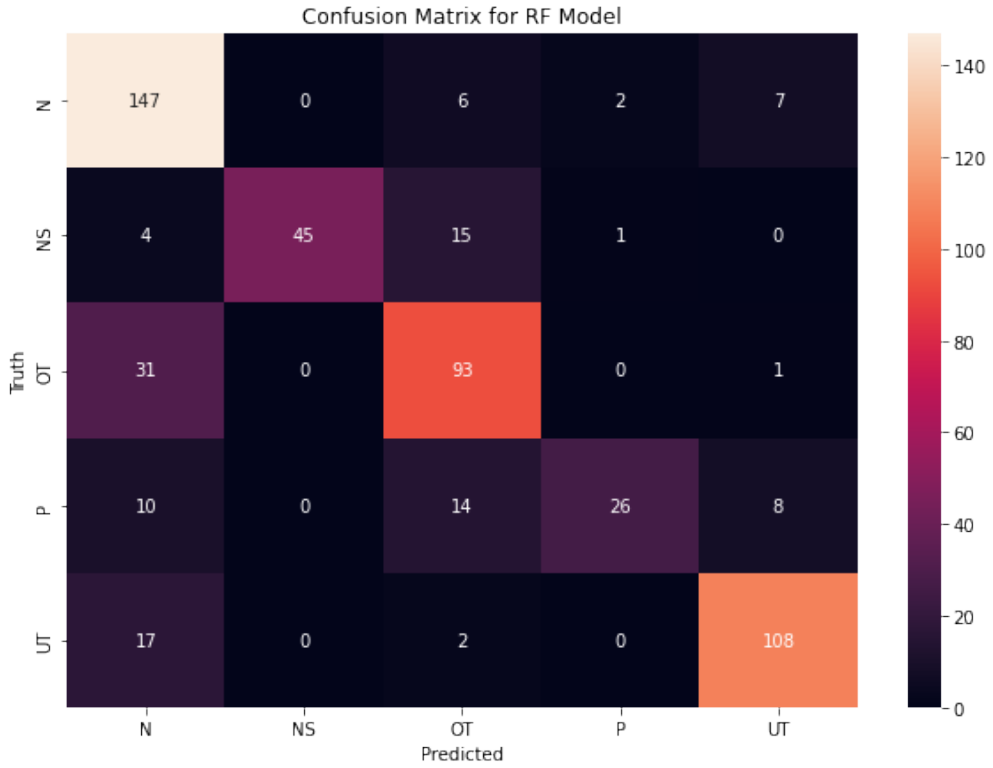


Figure B.12: *Confusion matrix of extrinsic RF model*

Accuracy: 78.03%

C Intrinsic + Task model

The accuracy of the confusion matrices is shown in instances of true or false predictions and not in percentages, like the matrices from the report Results Chapter.

C.1 RF algorithm (intrinsic + task)

	precision	recall	F1-score	support
N	*0.98*	*0.98*	*0.98*	*165*
NS	*0.98*	*0.97*	*0.98*	*66*
OT	*0.98*	*0.95*	*0.96*	*138*
P	*0.97*	*1.00*	*0.98*	*59*
UT	*0.95*	*0.97*	*0.96*	*109*
accuracy			**0.97**	**537**
macro avg	**0.97**	**0.97**	**0.97**	**537**
weighted avg	**0.97**	**0.97**	**0.97**	**537**

Table C.1: *RF classification report of intrinsic + task model*

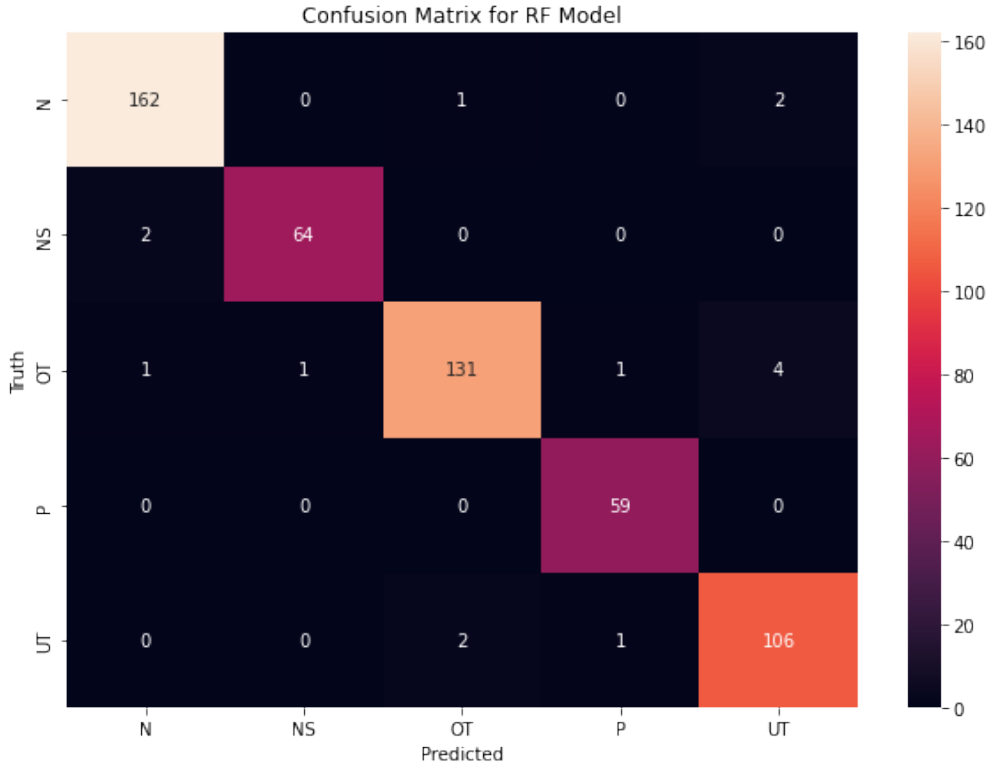


Figure C.1: *Confusion matrix of intrinsic + task RF model*

Accuracy: 97.21%

Cross validation accuracy: 96.64%

D Combined model

The accuracy of the confusion matrices is shown in instances of true or false predictions and not in percentages, like the matrices from the report Results Chapter.

D.1 KNN algorithm (combined)

	precision	recall	F1-score	support
N	*0.96*	*0.93*	*0.94*	*165*
NS	*0.98*	*0.97*	*0.98*	*66*
OT	*0.92*	*0.94*	*0.93*	*138*
P	*0.97*	*1.00*	*0.98*	*59*
UT	*0.95*	*0.95*	*0.95*	*109*
accuracy			**0.95**	**537**
macro avg	**0.96**	**0.96**	**0.96**	**537**
weighted avg	**0.95**	**0.95**	**0.95**	**537**

Table D.1: *KNN classification report of combined model*

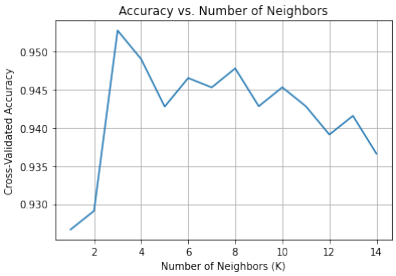


Figure D.1: *Cross-validation of the optimal neighbor number (n=3)*

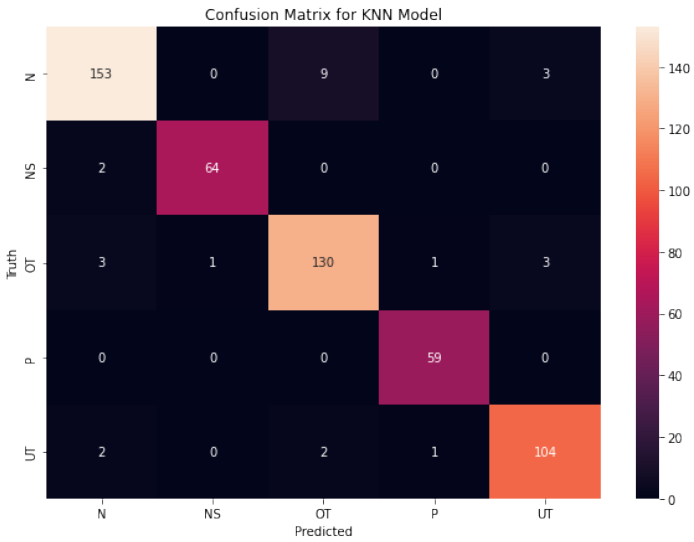


Figure D.2: *Confusion matrix of combined KNN model*

Accuracy: 94.97%

D.2 SVM algorithm (combined)

	precision	recall	F1-score	support
N	*1.00*	*0.92*	*0.96*	*165*
NS	*0.98*	*0.98*	*0.98*	*66*
OT	*0.94*	*0.94*	*0.94*	*138*
P	*0.88*	*1.00*	*0.94*	*59*
UT	*0.93*	*0.96*	*0.95*	*109*
accuracy			**0.95**	**537**
macro avg	**0.95**	**0.96**	**0.95**	**537**
weighted avg	**0.95**	**0.95**	**0.95**	**537**

Table D.2: *SVM classification report of combined model*

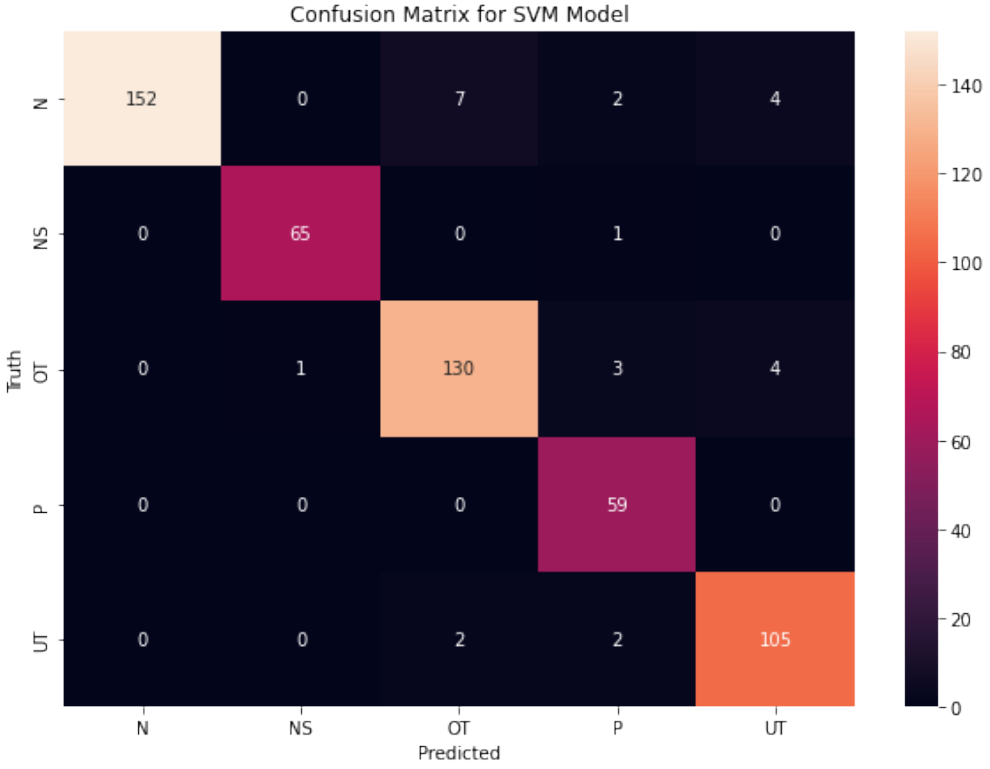


Figure D.3: *Confusion matrix of combined SVM model*

Accuracy: 95.16%

D.3 RF algorithm (combined)

	precision	recall	F1-score	support
N	*0.98*	*0.98*	*0.98*	*165*
NS	*0.98*	*0.97*	*0.98*	*66*
OT	*0.97*	*0.95*	*0.96*	*138*
P	*0.97*	*1.00*	*0.98*	*59*
UT	*0.95*	*0.97*	*0.96*	*109*
accuracy			**0.97**	**537**
macro avg	**0.97**	**0.97**	**0.97**	**537**
weighted avg	**0.97**	**0.97**	**0.97**	**537**

Table D.3: *RF classification report of combined model*

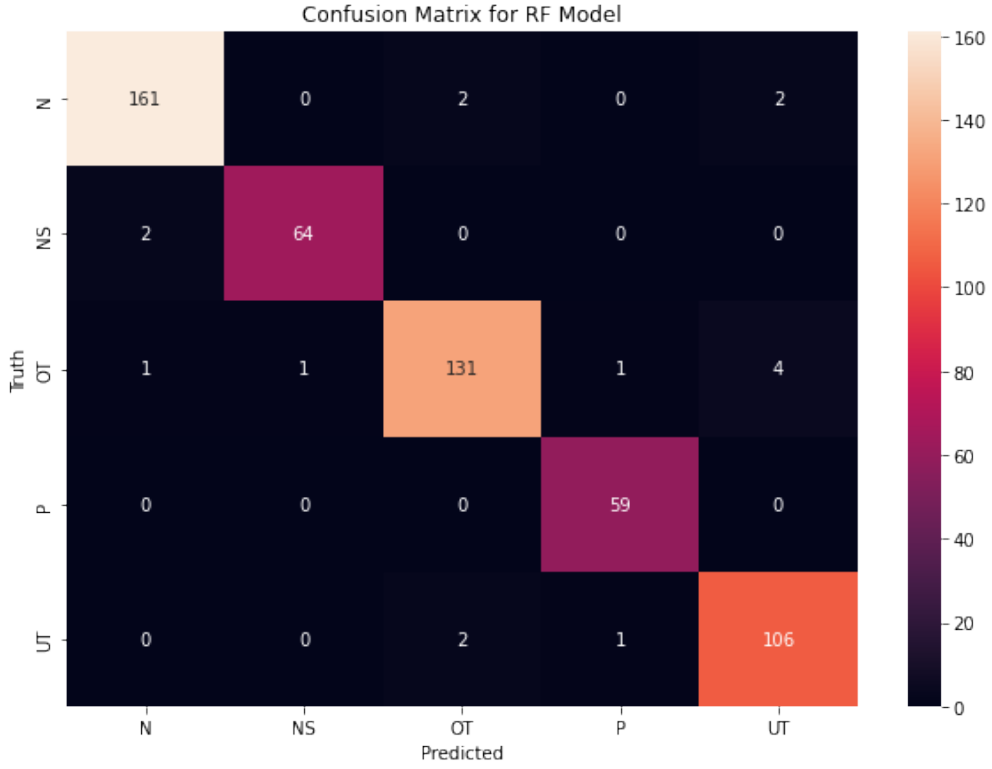


Figure D.4: *Confusion matrix of combined RF model*

Accuracy: 97.02%