Joachim Ågotnes

Machine Learning and Image Processing for the Study of Fluid Particle Breakage in Turbulent Flow

Master's thesis in MTKJ
Supervisor: Hugo Atle Jakobsen
June 2020

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Chemical Engineering

Joachim Ågotnes

Machine Learning and Image Processing for the Study of Fluid Particle Breakage in Turbulent Flow

Master's thesis in Industrial Chemistry and Biotechnology
Supervisor: Hugo Atle Jakobsen
June 2020

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Chemical Engineering

Preface

The master thesis is conducted at the Institute of Chemical Engineering at the Norwegian University of Science and Technology in the spring of 2020 for the research group for environmental engineering and reactor technology.

I would like to thank my supervisor, Hugo Atle Jakobsen for always being interested. Further on I would like to thank my two co-supervisors, Eirik H. Herø and Nicolas La Forgia for making it possible to write a thesis on this subject and for always giving feedback and asking questions.

I would also like to thank all my friends and family for always being supportive, especially since the corona-epidemic has made writing a master thesis different, as most of the work was done from home.

Summary

The main object of the project is to create machine learning models for application in an already existing image analysis software to be able to track oil droplets in a turbulent flow of water. In addition, the project aims to clearly describe the procedure of creating machine learning models with high accuracy.

Experiments consisting of a column of turbulent water where one oil droplet is injected into the system has been previously conducted. Two high-speed cameras record images when the droplet passes through the column. The image analysis software attempts to track the original droplet in addition to any daughter droplets if the original droplet breaks. Although it requires manual corrections to successfully track all of the droplets.

This project aims to improve the current tracking software by introducing machine learning models. The previously recorded experiments were processed using an in-house developed image processing software. In addition, input was required to correct any software misclassifications. Two different data sets were created; the first one describes breakages and the second one describes if two droplets are the same.

A series of variables related to the droplets in every frame were saved for every experiment. These data sets were used to create machine learning models. The supervised machine learning models used for this thesis were logistic regression, discriminant analysis, K-nearest neighbors, support vector machines, and tree ensembles. For data exploration, an unsupervised method called principal component analysis (PCA) was used. All of the different machine learning models have hyperparameters that are optimized by the use of Bayesian optimization with the objective of increasing a measurement of performance, namely the area under the curve (AUC).

The resulting models show that tree ensembles models are the most appropriate models for both data sets and are improvements compared to the old image analysis software. These models were implemented in the already existing image analysis software and tested on three different cases with different complexity. The case studies show that the newly implemented machine learning models outperform the old tracking logic for all of the three cases. The input needed from the user to correct for any mistakes that the image analysis software does is greatly reduced.

Sammendrag

Hovedformålet med denne oppgaven er å lage maskinlæringsmodeller for bruk i en allerede eksisterende bildeanalysekode for å kunne spore oljedråper i en turbulent strøm av vann. I tillegg, har prosjektet som mål å beskrive prosedyren for å lage og teste maskinlæringsmodeller med høy nøyaktighet.

Eksperimentet består av en kolonne med turbuelent vann der en oljedråpe blir injisert i systemet. To høyhastighetskameraer blir brukt for å ta bilder av oljedråpene når de passerer kolonnen. Bildeanalysekoden prøver å spore dråpene i tillegg til datterdråpene hvis den første dråpen deler seg, men dette krever manuelle korreksjoner.

I dette prosjektet blir det forsøkt å forbedre sporingen på dråpen ved hjelp av maskinlæringsmodeller. Tidligere ekseperimenter ble prossessert ved hjelp av bildeanalysekoden i tillegg til manuelle korreksjoner. To datasett ble lagd der den første beskriver delingen av dråper og det andre datasettet beskriver hvis to dråper er den samme.

En rekke variabler relatert til dråpene i hvert bilde ble lagret. Denne dataen ble brukt for å lage maskinlæringsmodellene. Følgende maskinlæringsmodeller ble brukt i dette prosjektet: logistisk regresjon, diskriminerende analyse, K-nærmeste naboer, støttevektormaskiner og treensembler. For utforskning av data ble prinsipiell komponentanalyse brukt. Alle de forskjellige maskinlæringsmodellene har hyperparametere som er optimalisert ved hjelp av Bayesisk optimalisering med mål om å arealet under kurven (AUC) som er et mål på ytelse.

De konstruerte modellene viser at treensembler er de beste og mest nøyaktige modellene for begge datasettene i tillegg til at modellene er mer nøyaktig enn den tidligere implementerte koden. Disse modellene ble implementert i bildeanalysekoden og testet på tre forskjellige, nye eksperimenter med forskjellig kompleksitet. Alle tre testene viser at den nye implementasjonen med maskinlæringsmodeller gir bedre resultater enn den gamle bildeanalysekoden. Dette gjør at det kreves mye mindre manuelle korreksjoner i hvert eksperiment, noe som sparer mye tid.

Contents

List of Figures

List of Tables

Acronyms

ALS Alternating Least Squares. 19

AUC Area Under the Curve. vii, 21, 22, 44, 49, 52–62, 73

DA Discriminant Analysis. 61

EIG Eigenvalue Decomposition. 19

KNN K-Nearest Neighbors. 28, 54, 61

LDA Linear Discriminant Analysis. 24–26, 54

MSE Mean Squared Error. ii, 40

PBE Population Balance Equations. 1, 2

PCA Principle Component Analysis. 17, 18, 43, 44, 48, 49

PRC Precion Recall Curve. vi, vii, 22, 44, 50, 53, 55, 57, 59, 73

QDA Quadratic Discriminant Analysis. 24–26, 54

ROC Receiver Operating Characteristics. vi, vii, 22, 44, 50, 52, 54, 56, 58, 60, 73

SGD Stochastic Gradient Descent. 23

SVD Singular Value Decomposition. 19

SVM Support Vector Machine. 57

List of Symbols

α_i	Weight for SVM	–
β_i	Logistic regression coefficient i	–
δ_k	Discriminant score for class k	–
ε_i	Slack term	–
γ	Regularization parameter	–
$\hat{\Sigma}$	Predicted covariance matrix	–
$\hat{y}_i$	Predicted response	–
λ	Regularization parameter	–
μ	Mean	–
ϕ_{ij}	Loading coefficient i in principle component j	–
π_k	Prior probability for class k	–
Σ	Covariance matrix	–
σ^2	Variance	–
$\tilde{x}$	Standardized observation	–
a	Parameter for Minkowski distance	–
B	Number of bootstrap data sets	–
B	Number of decision trees	–
C	Budget for slack variables	–
C^*	Cost penalty on slack variables	–
D	Deviance	–
d	Distance between to observations	–
d_s	Polynomial number	–
E	Lowest classification error	–

Err	Classification error	–
F	Data set for boosting	–
G	Gini index	–
H	Model in boosting	–
K	Kernel	–
k	Number of folds in cross-validation	–
M	Margin	–
m	Number of randomly chosen variables for random forest	–
m	Number of variables chosen at each split in decision trees	
n	Number of observations	–
p	Number of variables	–
r	Rank	–
V	Diagonal matrix with standard deviations	–
X	Model matrix	–
X_i	Variable i	–
X_i^*	Variable with mean 0	–
x_{ij}	Variable j in observation i	–
Y	Response vector	–
y_i	Response from observation	–
Z_i	Principle component i	–

Chapter 1

Introduction

Within the oil and gas industry, separation is widely used in, for instance, separating oil from gas or water from oil. Although the mechanism of different separators may vary, all have a dispersed phase with a size distribution. The properties of the dispersed phase are therefore important in order to maximize the production by the separators. The population balance equations (PBE) describe the size distribution evolution using models for coalescence and breakage [2].

For the population balance the following is needed: D which is the size of the mother droplet, t_b which is the breakage time, P_D which is the size distribution of the daughter droplets, ν which is the average number of daughter droplets, as well as ε which is the turbulent kinetic energy dissipation rate [1]. To find the parameters for the PBE, an experimental procedure was developed by Herø et. al. [1] for conducting single droplet breakage experiments in turbulent flow with oil and water. In addition, an image analysis software was made that processes the images from the experiments.

The experimental setup consists of a closed loop with a continuous flow of water with a breakage channel with periodic structures to create turbulence. A small drop of oil is injected into the system with a syringe before the breakage channel. There are two high-speed cameras that will take images of the oil droplet when it goes through the breakage channel where it may or may not break.

The image analysis software that is described in [1] is an ongoing project that is improved over time as it is a comprehensive software. It still has challenges that need to be overcome, as it requires manual interpretation and input to process the images from the experiments correctly. One of the main issues related to the software is deciding upon when a droplet breaks. When the image analysis software predicts that a droplet breaks, the new daughter droplets will be classified as two new droplets. If the software detects that the droplets break, when in reality it does not, the software will mark the droplet as a new droplet. This mistake will need to be corrected manually. The second related issue that will be considered in this thesis is to track the droplets from one frame to another. The image analysis software also considers both the initial breakage definition and the cascade breakage definition. The initial breakage definition is based on only considering the first breakage of the mother droplet and thus neglecting any other breakages that may occur during the experiment. The cascade breakage definition also takes into account also the later breakages in addition to the initial breakage of the mother droplet. The breakage event is assumed finished if a droplet is circular for five or more consecutive frames.

The main topic of this thesis is to improve the tracking accuracy of the droplets within the image analysis software by using data from previously conducted experiments. The current version makes many mistakes that have to be interpreted and require manual correction. These corrections are extremely time-consuming. The tracking part of the image analysis software will be improved by creating two different models. The first model will detect breakages of droplets in a turbulent flow. The second model will track the droplets from one frame to another, classifying when a droplet in the previous frame is the same as a droplet in the current frame.

The models that will be explored in this thesis are based on machine learning models. These types of models use a previously obtained data set to make predictions of new observations. The data sets required are obtained from processing experiments with the image analysis software and then manually correcting the errors it makes. The hypothesis is that with these models, it is possible to track the droplets much better than the current tracking. The current tracking logic is based on using the difference in some of the variables from one frame to another and combining these. In the end, this new combination is compared to a threshold value that is found by trial and error. Reducing the required intervention by the user and thus decreasing the need for manual corrections will increase the consistency of the software. In addition, the risk of human error is decreased, and thus the accuracy of the calculated parameters for the PBE is increased.

The objective of the thesis will be to find the most accurate machine learning models for predicting breakages and to track the droplets from one frame to another with the optimal hyperparameters. In addition, concepts within machine learning will be presented. The obtained models from this thesis will be used as a part of the image analysis software, which in return will be used in the study of coalescence models for the use in the PBE.

Chapter 2

Experimental Setup

The purpose of the experiment is to study the breakage phenomena of single oil droplet in turbulent flows. For this study, which is a statistical study, many breakage cases must be obtained and analyzed to obtain breakage models for the PBE. The experiment is conducted by injecting an oil droplet which is transported by a continuous flow of water to a breakage channel where turbulence develops at the channel walls. An experiment consists of a single oil droplet going through the breakage column. Two high-speed cameras detect and record the movement of the oil droplets in the breakage channel. The images obtained from the experiment are processed in MATLAB. In every frame variables such as droplet position and droplet size are stored for all the droplets.

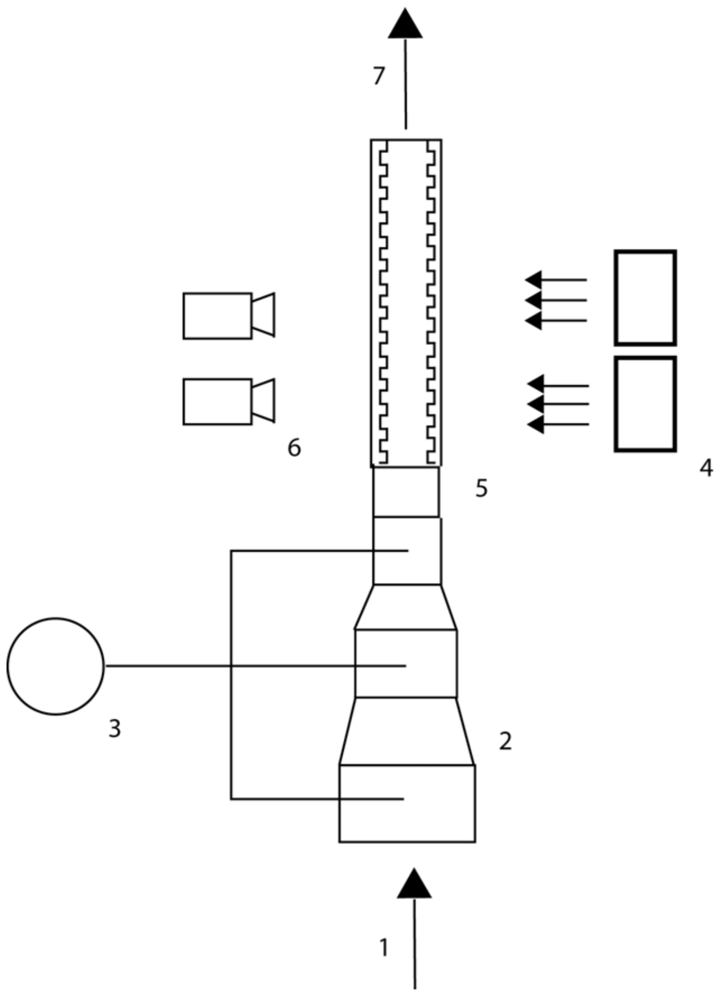


Figure 2.1: Experimental setup. 1. Water inlet. 2. Region for droplet generation. 3. Pump for syringe. 4. Lights for illumination. 5. breakage channel with structures. 6. Cameras. 7. Water outlet. [1]

The experimental setup for single droplet breakage experiments in turbulent flow is described more in detail by Herø et. al. [1]. The setup of the experiment is presented in Fig. 2.1. Here, a one meter long breakage channel with a square shape can be seen. The breakage channel has a cross-sectional area which is 24 mm by 30 mm. Inside the breakage channel, there are periodic structures to create turbulence. The droplet is generated by a syringe with a needle and a pump. The size of the droplets can be adjusted by injecting the droplets in different regions of the droplet generation region. The velocity of the water will be higher where the cross-sectional area is smaller and thus cause the droplet to detach from the needle of the syringe quicker. The droplet travels to the breakage channel which has two synchronized high-speed cameras that acquire images at 4000 frames per second. It should also be noted that both cameras are on the same side of the channel, meaning that the experiment only produces 2D images. The oil droplets consist of 1-octanol in addition to a dye, namely black sudan, to increase the contrast for detection of the droplets [1]. The turbulent flow characterization inside the breakage channel has been previously studied in the work of La Forgia et. al. [3].

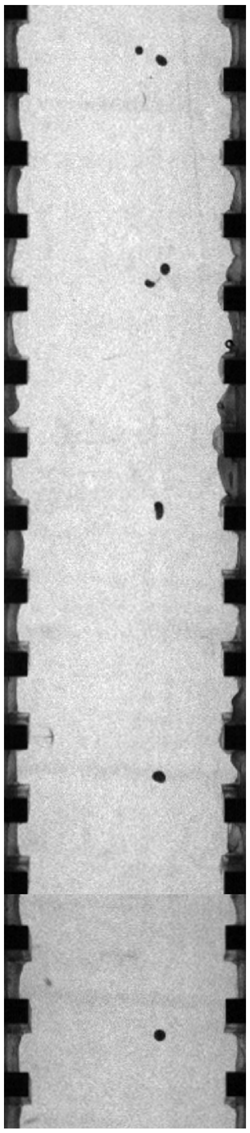


Figure 2.2: Collage of frames showing droplet breakage with 0.024 seconds between each image.

Fig. 2.2 shows succession of images, showing the effect of the turbulent flow on the droplet deformation and the resulting breakage event. The droplets enter the bottom of the image and exits through the top. This is described in Fig. 2.1 as 5. The dark areas to the right and left of the breakage channel shows the periodic structures that create turbulence within the breakage channel. The droplet always enters the breakage channel as one droplet. In the case where more than one droplet enters the breakage channel, the experiment is discarded. The droplet that enters the breakage channel is called the mother droplet. The mother droplet may, or may not break into several smaller droplets. In the case of Fig. 2.2, the droplet breaks into three smaller daughter droplets, although the third droplet is very small. The daughter droplets may also break again, creating many more droplets and complex experiments. If the droplets break in the wall, the experiment is also discarded.

Chapter 3

Image Analysis Software

The image analysis software processes one experiment at the time and starts by pre-processing the experiments. This includes removing the background of all the images and calibration. This makes it possible to look at the differences in the grey-scale values to prevent dirt or other dark areas to be detected as droplets. As the distance from the cameras and the column is not equal at all parts of the column, there is a need for calibration to correct for this. The two cameras take images at the same time of the lower and the upper part of the breakage column with a small overlap. The two images from the cameras are used to construct one large image of the column. The image analysis software takes into account both the initial and the cascade breakage definition, which is also the case for this thesis. The image analysis software is explained more in detail by Herø et. al. [1].

After pre-processing, the image analysis software starts to track the droplets of each experiment. When tracking, the software plots the trajectory of the droplet or droplets, distinguishing between the different droplets. For every frame, the software checks if there is a breakage or not by comparing the droplets in the previous frame by the droplets in the current frame. In addition to this, the software checks if the droplet is the same droplet as in the previous frame to be able to track the droplets. If it is neither a breakage nor the same droplet, the droplet is classified as a new droplet to be able to continue to track the droplet when the software fails.

For every frame, the software utilizes a function that compares a droplet in the current frame with a droplet in the previous frame by the use of differences in certain variables. The output of the equation is then evaluated against a set threshold to determine whether a comparison of two droplets indicates a breakage of if the droplet is the same. When testing for whether a droplet is the same the following equation is used:

$$T_1 = \frac{DD\_centroid \cdot DD\_area}{u} \tag{3.1}$$

Where *DD_centroid* is the euclidean distance from one centroid to the other centroid, *DD_area* is the change in area of the droplets and *u* is the velocity of the continuous water flow. For this thesis, the velocity of the continuous water flow was kept almost constant, and thus not used for the machine learning models. Adding different velocities is suggested for further work.

For testing the breakage a different procedure is used. The procedure estimates the area and the position of the combined droplets that are believed to be the daughter droplets. This is to approximate the area and the position of the droplets assuming the breakage did not happen. The approximated area and position is then compared to the area and position of the mother droplet with a threshold, using the function in Eq. (3.1), with a new threshold, T_2, to decide whether it is a breakage or not.

Both T_1 and T_2 are evaluated against a threshold value that is set by trial and error and if T_1 or T_2 are not below the threshold, the droplet will be classified as a new droplet. Although, this method sometimes fails when the changes in the deformation and the changes in the centroid placement is larger than anticipated. Because of this, a new approach is suggested by including machine learning models with more variables. The machine learning methods are believed to be more robust for identifying breakage events and tracking the droplets.

There are several objectives with the tracking. To get the breakage time, the first frame that the mother droplets start to oscillate in must be found. A droplet starts to oscillate when the droplet is no longer circular in the 2D image. All of the breakage instances must be found. The size of the daughter droplets is also estimated and a series of experiments is used to calculate the probability of a breakage. The position of the mother droplet in the frame where the oscillation starts is used to calculate the turbulent energy dissipation rate [1].

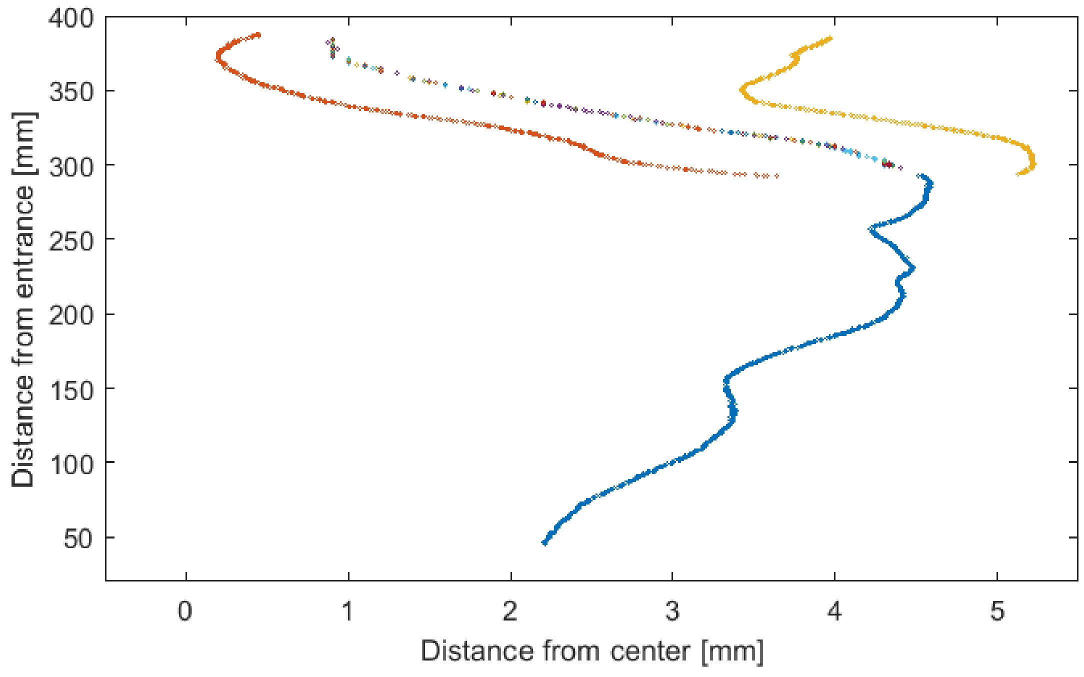


Figure 3.1: Plot showing droplet breakage.

In Fig. 3.1, the trajectories of the same experiment as shown in Fig. 2.2 is shown. The x-axis describes the position in mm from the middle of the column, while the y-axis describes the position from the entry of the column in mm. Although not shown in the plot, there is also a third dimension, describing the frame number of each individual droplet. Each point marks a specific droplet in a specific frame, and if the colours of two consecutive droplets are the same, it is the same droplet. From the collage in Fig. 2.2, it can be seen that the mother droplet breaks into three smaller droplets. However, the middle droplet in Fig. 3.1 shows that the software struggles to recognize which droplets are the same as the consecutive droplets are coloured differently. These misclassifications make the software classify the droplets as new droplets since the software did not classify the droplet as the same droplet or as a breakage.

Chapter 4

Creating a Data Set

Machine learning are methods that utilizes previously obtained data to create models that learn from patterns in the data. When the goal is to have a high prediction accuracy the amount of previously obtained observations is important. Having more observations will in most cases increase the predictive capabilities of the models. Although, as the number of observations increase, the increase in prediction accuracy diminishes.

Creating a data set to train a machine learning model with high-quality data is not only vital to be able to make accurate predictions, but it can also be very time-consuming. If the quality of the data is low, meaning that there are variables missing that explain a correlation between the observation and the true nature of the system, it will greatly influence the predictive accuracy of the model. It is therefore important to choose variables that are believed to have a correlation to the behaviour of the system.

The old image analysis software is based on comparing the differences in certain variables as described in Chapter 3 to a threshold value. The software compares every droplet in the current frame with every droplet in the previous frame. The new image analysis software is also based on comparing every droplet in the current frame with every droplet in the previous frame. Although, machine learning models are used to determine whether it is a breakage of if it is the same droplet instead of using T_1 and T_2.

In the data set, observations are stored. An observation is defined as a series of variables of a droplet in the current frame in addition to the same variables for a droplet in the previous frame. This means that the number of observations for one frame is the number of droplets in the current frame multiplied by the number of droplets in the previous frame. In addition, two responses for every observation are stored. The first response is whether the comparison of the two droplets shows a breakage and the second response is whether the droplet is the same droplet. In this thesis, two different data sets will be created, and therefore there will also be two responses and two machine learning models. The first data set, which from now on will be referred to as the breakage data set, has the breakage as the response. The second data set, which from now on will be called the droplet data set, checks if the droplet in the previous frame and the current frame is the same droplet or not.

Both of the data sets have the same variables, with the only difference being the response. Why is it needed to have two different models? An alternative approach would be to use one data set with three different responses instead of two. The first response describes the case where it is not the same droplet and there is no breakage. The second response describes not the same droplet and a breakage while the last response describes the same droplet. There are some problems with this approach as when there is a breakage forming between two droplets, the droplets are also not the same droplets, which would make it difficult for the algorithm to differentiate between the different classifications.

To create a data set, the old tracking algorithm is used to track the droplets in several experiments. Although, since the software makes mistakes, there is need for manual corrections. This is a very time consuming operation as there is a need for a large amount of data when training the machine learning models. To correct for the mistakes, there is a need to look at the specific frames to determine which droplets are which and when the droplets break. Depending on the situation, the droplets are merged if two droplets are in reality the same. The droplets are deleted if the tracking has made errors in the tracking itself or the droplets are split if there are droplets that are not the same droplet. In addition to this, there is a need to label all the observations with breakage or non-breakage and if it is the same droplet.

4.1 Choice of Variables

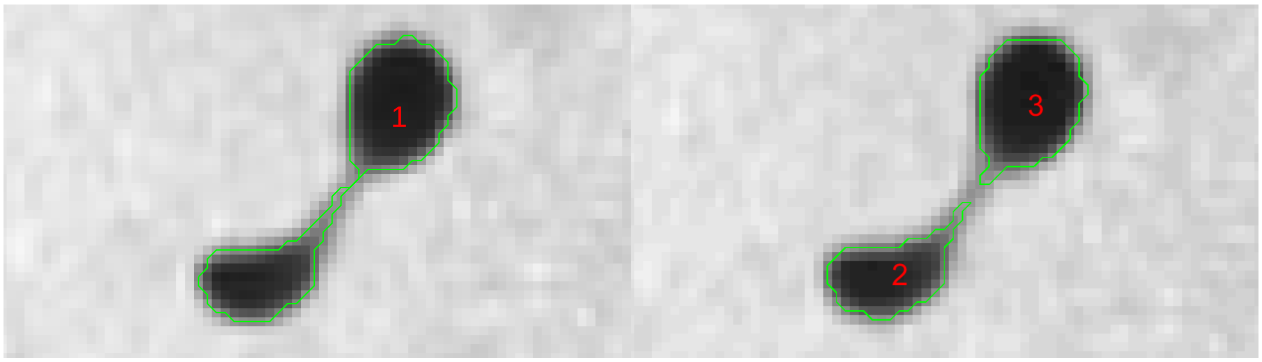
The cameras are recording the experiment has a set frame-rate of 4000 frames per second. This means that the time between two frames is 0.25 ms and since the breakage events are observed in between two frames, the accuracy of the time of breakage is associated with this value. The first frame after the breakage is therefore stored as the breakage frame.

Many of the variables are defined for both droplet i, which is a droplet in the previous frame, and for droplet j, which is a droplet in the current frame. In total 24 variables are stored for both the breakage and the droplet data set in addition to the corresponding response.

4.1.1 Responses

This subsection describes the stored responses for the two different data sets.

Breakage: The breakage point is stored in the data set as the response for creating the breakage model and is coded with 0/1, where 0 represents no breakage, and 1 represents a breakage.



(a) Droplet before breakage. (b) Droplet after breakage.

Figure 4.1: Two subsequent frames showing a breakage of a droplet.

An example of a breakage event is shown in Fig. 4.1. Here two consecutive images depicting the frame before and after the breakage is shown. The green line represents where the image analysis software finds the edges of the droplets. As seen, there is only one droplet in this figure. Fig. 4.1b shows the frame after and the first frame after the breakage happens. There are now two droplets, which are marked by two green regions by the image analysis software. As seen in Fig. 4.1, droplet one is marked by the green region and is split into droplet two and three. Droplet one will thus be checked against droplets two and three and in this case, both show breakage instances.

The Same Droplet: When checking the droplets in the current image with the droplets in the previous image, the droplets that in reality are the same will be classified 1, while the droplets that are not the same will be classified as 0. This information is used as the response when creating the droplet model.

4.1.2 Variables

This subsection briefly describes all the variables in the two data sets. The variables are equal for both data sets.

DD_contour and DD_contour_max: The DD_contour is defined as the minimum euclidean distance from the contour of droplet j to droplet i. DD_contour_max is defined as the maximum distance from the contour of droplet j to droplet i. The contour of the droplets is defined by all of the border points of the droplet, which is the outer pixels of the droplet.

DD_centroid: The Euclidean distance from the centroid of the first droplet to the centroid of the other droplet is given as the distance from the mass center of the centroid of the first droplet to the mass center of the centroid of the second droplet.

Area: The area of a droplet is the area captured by the border points, which is equal to the projected area. Although, care should be taken when looking at the area as the area can change when the droplet is stretched in the third dimension. It is given in mm^2.

x_pos: x_pos is the position of the centroid of the droplet on the x-axis which is the axis perpendicular to the breakage channel. It is given as the position in pixels from the left wall of the breakage channel.

y_pos: y_pos is the position of the centroid of the droplet on the y-axis which is the axis parallel to the breakage channel. It is given as the position in pixels from the bottom of the breakage channel.

aAxis: The aAxis for a droplet is half of the major axis for the droplet given in mm.

bAxis: The bAxis for a droplet is half of the minor axis for the droplet given in mm. Variable $bAxis_j$ was neglected as it was linearly dependent on $aAxis_j$.

Diameter: The diameter for a droplet is given in mm and is given by:

$$Diameter = (2 \cdot aAxis \cdot (2 \cdot bAxis)^2)^{\frac{1}{3}} \tag{4.1}$$

The diameter of the droplet is an approximation assuming a 3d elliptic shape of the droplet with the third axis equal to the minor axis. Although, in reality, the shape is arbitrary.

Deformation: The deformation is given in mm and is given by:

$$Deformation = \frac{aAxis - bAxis}{aAxis} \tag{4.2}$$

This is again an approximation assuming the droplet is elliptic.

Eccentricity: The eccentricity for a droplet is given in mm and is given by:

$$Eccentricity = (2 \cdot Deformation - Deformation^2)^{0.5} \tag{4.3}$$

The eccentricity describes how stretched out the ellipse is by being the ratio of the foci and the major axis of the droplet.

Equivalent diameter: The equivalent diameter is given in mm and is given by:

$$EquivDiameter = (4 \cdot aAxis \cdot bAxis)^{0.5} \tag{4.4}$$

The equivalent diameter gives the diameter of a circle with the same area as of the projected image.

Perimeter: The perimeter of a droplet is given in mm and describes the length of the boundary that the border points create.

Orientation: The orientation of a droplet is given in degrees and describes the orientation of the major axis of the droplet in comparison to the x-axis.

Summing up, the stored variables and responses are:

Table 4.1: Variables used in the data sets.

Variable/response	Variable/response name
Y	$Breakage$
Y	$Same\ droplet$
X_1	$DD\_Contour$
X_2	$DD\_Contour\_max$
X_3	$DD\_Centroid$
X_4	$Area_i$
X_5	$Area_j$
X_6	x_pos_j
X_7	y_pos_j
X_8	$Diameter_i$
X_9	$Deformation_i$
X_{10}	$Eccentricity_i$
X_{11}	$EquivDiameter_i$
X_{12}	$Perimeter_i$
X_{13}	$Orientation_i$
X_{14}	$Diameter_j$
X_{15}	$Deformation_j$
X_{16}	$Eccentricity_j$
X_{17}	$EquivDiameter_j$
X_{18}	$Perimeter_j$
X_{19}	$Orientation_j$
X_{20}	x_pos_i
X_{21}	y_pos_i
X_{22}	$aAxis_i$
X_{23}	$bAxis_i$
X_{24}	$aAxis_j$

Where the breakage data set consists of the breakage response in addition to the 24 other variables and the droplet data set consists of the same droplet response and the 24 variables.

Chapter 5

Machine Learning

Machine learning is a big field of research, involving many different disciplines such as statistics, computer science, and optimization. It should be noted that in this thesis, the goal is to utilize machine learning methods that predict the data with a satisfactory level of success. As there are entire books devoted to the specific models such as [4, 5], only an overview of the different models will be explained. Many of the different models have extensive use of optimization. Because optimization theory is a research field itself and is out of the scope of this thesis, the specific optimization methods will only be briefly explained with references to further information for the interested reader.

The different machine learning models may be divided into supervised and unsupervised learning. In supervised learning, the goal is to find relations between the variables X and the response Y and to be able to predict the response of a new observation as accurately as possible. In unsupervised learning, the goal is to find relations between the variables without the use of the response. The unsupervised methods can thus not be used for predictions. Since there is no access to the response, the purpose of unsupervised methods is to discover underlying connections between the variables. The only unsupervised method that will be looked into in this thesis is principal components analysis (PCA), which is explained in Section 5.1. The rest of the models are supervised methods.

The supervised learning methods may be divided into two different subgroups, regression, and classification. Regression methods are used when the response is quantitative, and the classification methods are used when the response is qualitative [6]. Both the data sets discussed in this thesis have a qualitative response, which represents the different classes of the observations and is denoted as an integer being either 1 or 0. This means that classification models will be utilized for solving the problem. The goal with these methods is to predict the response as accurately as possible such as getting a low classification error as possible which is given by:

$$Err = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) \tag{5.1}$$

Where n is the total number of observations. The term $I(y_i \neq \hat{y}_i)$ is one if $y_i \neq \hat{y}_i$ and zero otherwise.

For simplicity, further on in this thesis, the response of the observations which is breakage/no-breakage or if it is the same droplet or not will be denoted as Y, while the observed variables or features will all be in the matrix, X, which is referred to as the design matrix. The response is given by, Y is $y_1, y_2, ..., y_n$ where n is the number of observations. There are n rows in the design matrix, X, and the dimensions of each row in the design matrix is given by $X_1, X_2, ..., X_p$ where p is the total number of variables or features. x_{ij} denotes variable j in observation i and y_i denotes the response of observation i. The data set can be visualized in the following table:

Table 5.1: Visualization of a data set.

Y	X_1	X_2	X_3	$\cdots$	X_p
y_1	x_{11}	x_{12}	x_{13}	$\cdots$	x_{1p}
y_2	x_{21}	x_{22}	x_{23}	$\cdots$	x_{2p}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
y_n	x_{n1}	x_{n2}	x_{n3}	$\cdots$	x_{np}

The goal of the machine learning is to use the response Y in addition to the model matrix $X = [X_1, X_2, ..., X_p]$ to create a model that explains the connection between the variables and the response. This model is then used to predict the response of new observations that do not contain the response. This means that for a data set of new observations the first column containing the response Y in Table 5.1 is not known, while the variables X are measured. The goal is to predict the value of Y.

A training data set is a set of data that is used to create a machine learning model. A testing data set is a set of data that is used to test a machine learning model. In general, the testing data set is not used to train the model.

All the models that will be discussed in this thesis have some tuning or hyperparameters. These are parameters that can be changed in order to create the most accurate model. The hyperparameters is optimized by Bayesian optimization which is briefly described in Appendix A.12.

The different theoretical aspects of this thesis will be organized as follows:

- PCA is explained in Section 5.1.
- k-fold cross-validation is a method for predicting the classification error and is explained in Section 5.2.
- Model selection for choosing the best model is explained in Section 5.3.
- The different machine learning models are explained in Section 5.4, Section 5.5, Section 5.6, Section 5.7, and Section 5.8.
- For optimizing the hyperparameters of the different models, Bayesian optimization is used as explained in Appendix A.12.

5.1 Principal Component Analysis (PCA)

In Chapter 4, various of measurable variables were considered when creating the data set, although it was not considered which variables were the most important. The principal component analysis is a method that is used for both visualization and to create principal components in which most of the variance in the original data set is captured. Each principal component is a weighted sum of the variables in the original data set, which also makes it possible to see in which variables most of the variance is captured. For simplicity, only the first principal component will be shown:

$$Z_1 = \phi_{11}X_1^* + \phi_{21}X_2^* + ... + \phi_{p1}X_p^* \tag{5.2}$$

Where Z_1 is the first principal component and $\phi_{11}, \phi_{22}, ..., \phi_{p1}$ are the weightings often referred to as the loading for the first principal component. X_i^* is a vector of the variables, $x_{1i}^*, x_{2i}^*, ...x_{ji}^*$. Only the variance is of importance and therefore, the variables are changed such that the mean of the different variables are zero. This means that $X_i^* = X_i - \hat{X}_i$. It should be noted that the loadings are normalized such that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$ for the first principal component to constrain the variance. The first principal components, Z_1, consists of principal component scores, $z_{i1}, z_{i2}, ...z_{in}$. In addition to this, the objective for the first principal component is to maximize the variance, thus the following optimization problem arises [7]:

$$\max_{\phi_{11},...,\phi_{p1}} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 \text{ subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1 \tag{5.3}$$

The principal components can be found by various methods such as eigenvalue decomposition (EIG) or alternating least squares algorithm (ALS), but in this case, it has been found by singular value decomposition (SVD) which is described in Appendix A.7.

The first principal component, Z_1, represents the direction in which the data is most variable and the second principal component, Z_2, gives the highest variance that does not have any correlation to the first principal component, Z_1.

5.2 k-Fold Cross-Validation (CV)

k-fold cross-validation is an alternative method for estimating the classification error as shown in Eq. (5.1) or any other similar objective. The goal of the method is to estimate the classification error of a model that has been trained by all available data. The method is based on dividing the data set into k folds or k bins. The first fold is used as a testing data set, while the other folds are used to train a model. The model is then tested on the testing data set to get a classification error that is stored. Then the second fold is used as a testing data set and a new classification error is calculated. This is repeated until all of the folds have been used as a testing data set.

The classification error described in Eq. (5.1) or a similar objective for the machine learning models can be calculated from training the model with the training data set and then testing the model on the training data. Although this leads to an underestimation of the true classification error [7].

Another solution is to have a training set to train the model on and a different data set to test the model on. Although, to achieve the most accurate model, all of the data must be used to train the data. Because of this, k-fold cross-validation is a commonly used method for estimating the classification error.

1
2
3
4
5
1
2
3
4
5

Figure 5.1: Illustration of dividing a set into 5 folds.

Fig. 5.1 illustrates the first iteration of the cross-validation. The first iteration of the cross-validation is illustrated at the left of Fig. 5.1. Each number describes one fold, so this example describes five-fold cross-validation. The red marking means that the red fold is used as a testing set, while all the other folds are used to train the model. After calculating the test error for this fold, the next iteration is described to the right of Fig. 5.1. Now the test set is fold 2 while all the other folds are used to train the model. The total test error is thus given by the average over all the test errors from the k models:

$$Err_{CV} = \frac{1}{k}\sum_{i=1}^{k} Err_k \tag{5.4}$$

Where Err_k is the classification error for each fold as given in Eq. (5.1). The number of folds that are used for k-fold cross-validation is usually $k = 5$ or $k = 10$. The reasoning behind not using the number of observations as folds, namely, $k = n$, is that it would result in very high variance. Using all the observations without one to train the model on and using only one observation as a test set would result in higher variance as all the models would be highly correlated [7].

Although getting an accurate estimate of the test error is important for achieving the best model, it should also be noted that in this case, it is the performance in the image analysis software that is the most important.

Algorithm 1: Algorithm for doing cross-validation.

```
Randomize the placement of all observations;
Divide the data set into $k$ folds;
for $i=1:k$ do
    Let fold $i$ be the testing set;
    Let folds $1:k$ without fold $i$ be the testing set;
    Train machine learning model with the training set;
    Find the error rate by testing model on testing set;
    Store error rate;
end
Calculate final error rate according to Eq. (5.4);
```

5.3 Model Evaluation

The performance of the models can be evaluated in different ways. An intuitive way of measuring the performance of the model is to use the classification error which is given in Eq. (5.1). The classification error describes the percentage of misclassifications that the model has done. Usually, the classification error is calculated from testing the model on data that was not used to train the model, or estimated by k-fold cross-validation as explained in Section 5.2.

Although the classification error is commonly used, it has some drawbacks. Instead of this measurement, another commonly used performance indicator is the area under the curve (AUC). To understand the drawbacks of the classification error and what the AUC is, the confusion matrix is firstly shown:

Table 5.2: Confusion matrix for classification.

		Predicted	
		1	0
Real	1	True positive	False negative
	0	False positive	True negative

The confusion matrix given in Table 5.2, describes how many correct predictions of both 0s and 1s the algorithm has correctly classified and also how many incorrectly 0s and 1s that it has classified. The classification error can thus be found by:

$$\text{Classification error} = \frac{\text{False negative} + \text{False positive}}{n} \tag{5.5}$$

Where n is the total number of observations.

The sensitivity describes the percentage of correctly classified positives and is given by [7]:

$$\text{Sensitivity} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} \tag{5.6}$$

The specificity, on the other hand, describes the percentage of correctly classified negatives and is given by [7]:

$$\text{Specificity} = \frac{\text{True negative}}{\text{True negative} + \text{False positive}} \tag{5.7}$$

There are two main drawbacks with using the classification error. Firstly, when dealing with imbalanced data sets, the classification error will be very low if the classifier classifies all the observations to the major class, and thus gives the false indication that the model performs well [8]. Secondly, the classification error only consider whether the observation is classified to one class or the other. It does not consider the probability that the observation is classified to a certain class, meaning it does not consider all values of the threshold for the cut-off probability. The threshold for the cut-off is the probability cut-off that divides the classes. In general, it is normal to use 0.5 as the threshold for the cut-off, meaning that any value of 0.5 and higher would classify the observation to the positive class. This threshold can be set to any given value. When varying the threshold, sensitivity and sensitivity will change. Plotting the sensitivity against 1-specificity gives the receiver operating characteristics (ROC). The area under the curve (AUC) tells how accurate the model is for all values of the threshold. An AUC of 1 gives a model where for any given value of the threshold, the sensitivity will be one and the specificity will be one, meaning that the model has correctly classified all the positive values correctly and all the negative values correctly. Therefore an AUC of 1 gives the best possible model. On the other hand, an AUC of0.5 suggests that the model is no better than random guessing. The model with the highest AUC is selected. Instead of the classification error, the AUC gives a better indication of the performance of a model [9]. The AUC based on ROC is used for the droplet data set. Having a data set that has a reasonable amount of observations from both classes, doing model selection based on the AUC calculated from the ROC can be used. Although, in the case of having a lot more observations from one class, the AUC based specificity and sensitivity should not be used. Instead, for cases where the data set is greatly imbalanced, the precision-recall should instead be used [10]. The sensitivity is equal to the recall, as shown in Eq. (5.7) and the precision is given by:

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}} \tag{5.8}$$

The precision gives the fraction of correct positives predictions compared to the total number of positive predictions. Plotting precision against the recall gives the precision-recall plot (PRC). Taking the area under the curve gives the AUC. The model with the highest AUC is selected. The AUC based on PRC is used for the breakage data set as the data set is imbalanced.

Unlike the ROC curve, the baseline AUC for the PRC is not constant. The baseline for sensitivity in the PRC is a function of the positive and negative observations:

$$\text{Baseline} = \frac{\text{Positive}}{\text{Positive} + \text{Negative}} \tag{5.9}$$

Where the positives and negatives are the amounts of positive and negative observations in the data set.

5.4 Logistic Regression

Linear regression is a well known supervised regression method, but it is not directly suitable for classification problems. It is possible to fit a least squares estimate to the 0/1-coding, and thus predict $\hat{y} > 0.5$ gives a classification of 1 and vice versa. Although there is a problem that the regression may give predictions outside the [0,1] range, so interpreting this as a probability does not quite make sense. Logistic regression is a method of modeling the probability that an observation belongs to a specific class [7]. It is thus a classification method, even though the name suggests otherwise. An ordinary linear regression model can be described by the following:

$$f(X) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p \tag{5.10}$$

Where β is the regression coefficients and X describes different variables. Instead of using Eq. (5.10), in logistic regression the probability for the observations belonging to the positive class is given with the use of a logistic function [7]:

$$p(X) = \frac{exp(\beta_0 + \beta_1 X_1 + ... + \beta_p X_p)}{1 + exp(\beta_0 + \beta_1 X_1 + ... + \beta_p X_p)} \tag{5.11}$$

The logistic function given in Eq. (5.11) has bounds from [0,1] as the exponential term cannot be negative. The cut-off probability is the probability that divides the classes. An intuitive cut-off probability is 0.5, meaning that all observations that give a probability, $p(x) < 0.5$, is classified to one class while all observations that give a probability, $p(x) \geq 0.5$ is classified to the other class. The cut-off probability is set by the user and different cut-offs may give different test errors and different sensitivities and specificities as explained in Section 5.3.

The logistic function given in Eq. (5.11) can also be rewritten as:

$$\frac{p(X)}{1 - p(X)} = exp(\beta_0) \cdot exp(\beta_1 X_1) \cdot ... \cdot exp(\beta_p X_p) \tag{5.12}$$

The left term of Eq. (5.12) is called the odds. When increasing a variable X_i by one, the odds will be multiplied by $exp(\beta_i)$ [7].The fitting of β_0,β_1,...β_p is done by stochastic gradient descent (SGD) which is explained in Appendix A.11.

5.4.1 Regularization

For regression methods, it is possible to shrink the variable space by setting some of the regression coefficients close to, or even zero. Shrinking the variable space means that some of the variables are neglected in the model. By doing this, the variance is lowered at a little cost for the bias, thus being able to increase the prediction accuracy. Ridge regression and the lasso are two shrinkage methods that will be looked further into in this thesis. This is also called regularization which are methods to reduce overfitting of a machine learning method, which is equivalent to reducing the variance. Overfitting means that the model fits the data that is used to train the model too well.

5.4.2 Ridge Regression

Ridge regression is a shrinkage method in which the regression coefficients β are shrunken towards zero, but never reaches exactly zero. The goal of the machine learning methods is to reduce the classification error given in Eq. (5.1). Ridge regression is very similar, but adds an additional term:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{5.13}$$

Where $\lambda \geq 0$ is a tuning parameter [7]. The last term in Eq. (5.13) is a penalty term to the coefficients often called the shrinkage penalty. This is also called an L_2 penalty. The value of the second term is low when the coefficients are also low, meaning that it penalizes high values for the coefficients. But there is also a trade-off between the reduction in classification error and the penalizing term, meaning that the coefficients of the variables that have the least effect on the classification error can be driven close to zero quickly, without increasing the total value of Eq. (5.13). The penalty is although not applied to the intercept, β_0, as the goal is to only penalize the coefficient of the variables, thus decreasing the effect of the associated variables. When the penalty factor λ is zero, will just give back the ordinary logistic regression, but when λ becomes large, it will drive the coefficients towards zero.

5.4.3 The Lasso

The lasso is closely related to ridge regression, but has a few differences. While the ridge regression sets some of the regression coefficients to be close to zero, the coefficients will actually never be zero. This can cause some confusion when interpreting the model, so to prevent this, the lasso sets some of the coefficients to zero. It is also based on a penalty function, just like in the ridge regression in Eq. (5.13) and is given by:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) + \lambda \sum_{j=1}^{p} |\beta_j| \tag{5.14}$$

Where $\lambda \geq 0$ is a tuning parameter in the same way as for the ridge regression. The last term in Eq. (5.14) is the shrinkage penalty or an L_1 penalty. When the penalty parameter, λ, becomes large it will set some of the coefficients to zero. So the main difference between the ridge regression and the lasso is that while the ridge regression always keeps all the variables and only drives the coefficients towards zero, the lasso only uses some of the variables as it sets some of the coefficients to zero.

5.4.4 Hyperparameters

The following parameters will be treated as hyperparameters and thus subject to optimization:

- $\lambda \in [10^{-10}, 1]$
- Standardize $\in$ [yes,no]
- Regularization type $\in$ [ridge,lasso]

The logistic regression in itself if scale-invariant, but the regularization is not. It is therefore added as a hyperparameter. The standardizing of the data is done according to Appendix A.1.

5.5 Discriminant Analysis (DA)

The discriminant analysis is a model type and can be divided into two subcategories, namely linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA). The LDA assumes that all the classes share the same covariance matrix, which leads to a linear decision boundary. The QDA assumes that all the classes have different covariance matrices, which leads to a quadratic decision boundary.

5.5.1 Linear Discriminant Analysis (LDA)

In the case of more than one variable ($p > 1$), it is assumed that all variables have the multivariate normal distribution and that they all share the same covariance matrix, Σ. Meaning that $X_k \approx N(\mu_k, \Sigma)$ where each of the classes, k, share the same covariance matrix, but have an individual expected value, μ_k. The multivariate normal density function is given by [7]:

$$f(x) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right) \tag{5.15}$$

Where p is the number of variables, μ is the expected value and x is the current observation. The LDA classifies an observation, x, to the class that is most likely according to the Bayes' theorem [7]:

$$p_k(x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \tag{5.16}$$

Where $p_k(x)$ is the probability for an observation x to belong to class k. π_k is the prior probability which is the probability that an observation belongs to class k in the data set used to train the model [7]. Inserting Eq. (5.15) into Eq. (5.16) gives the following equation for a discriminant score:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + ln(\pi_k) \tag{5.17}$$

The full derivation is showed in Appendix A.2. The classifier classifies an observation to the class that gives the highest discriminant score, $\delta_k(x)$ [7]. Since the discriminant score $\delta_k(x)$, is linear in regards to x, the decision boundaries produced by the LDA will also be linear. μ_k, Σ and π_k are estimations given by:

$$\hat{\mu}_k = \frac{1}{n_k}\sum_{i:y_i=k} x_i \tag{5.18}$$

Where n_k is the number of observations related to the class, k [7].

$$\hat{\pi}_k = \frac{n_k}{n} \tag{5.19}$$

Where n is the total number of observations. The prior probability, π_k, is thus the number of observations that belong to class k, divided by the total number of observations [7]. The pooled covariance matrix between two of the vectors, A and B, is estimated by [6]:

$$\hat{\Sigma}(A,B) = \sum_{k=1}^{K} \sum_{i:y_i=k} \frac{1}{n-K}(x_i - \mu_k)^T (x_i - \mu_k) \tag{5.20}$$

Where K is the total number of different classes and Σ has the dimension p x p. μ_k is a row vector of the means of the variables: $X_1, X_2, ..., X_p$ and x_i is a row vector describing observation, i.

5.5.2 Quadratic Discriminant Analysis (QDA)

The quadratic discriminant analysis is very similar to the linear discriminant analysis as described in Section 5.5.1 as it is based on the same assumption that all classes belong to the multivariate normal distribution in addition to using the Bayes' theorem. The difference between the LDA and the QDA is that the LDA assumes the same covariance matrix, Σ, for all the classes, while the QDA does not. This means that $X_k \approx N(\mu_k, \Sigma_k)$. With this assumption in mind, the discriminant score for the QDA is given by:

$$\delta_k(x) = -\frac{1}{2}x^T \Sigma_k^{-1} x + x^T \Sigma_k^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma_k^{-1} \mu_k - \frac{1}{2}ln(|\Sigma_k|) + ln(\pi_k) \tag{5.21}$$

The full derivation is showed in Appendix A.4. The QDA classifies the observation to the class with the highest discriminant score as given in Eq. (5.21). From Eq. (5.17), it is possible to observe that the discriminant score is now a quadratic function in terms of x. μ_k and π_k are estimations given in Eq. (5.19) and Eq. (5.18). The covariance matrix for each class is estimated by:

$$\hat{\Sigma}_k = \frac{1}{n_k - 1} \sum_{i:y_i=k} (x_i - \mu_k)^T (x_i - \mu_k)^T \tag{5.22}$$

Where n_k is the number of observations belonging to class k.

5.5.3 Hyperparameters

The following parameters will be treated as hyperparameters and thus subject to optimization:

- $\gamma \in [0,1]$
- Standardize $\in$ [yes,no]
- Model type $\in$ [Linear, pseudo-linear, diagonal linear, quadratic, pseudo-quadratic, diagonal quadratic]

The pseudo-quadratic and pseudo-linear model are the same as the quadratic and linear model described in Section 5.5.2 and Section 5.5.1, with the only difference being how to calculate the inverse of the covariance matrix, Σ^{-1}. This is done by using the pseudo-inverse of the covariance matrix, which is explained in Appendix A.8. The pseudo-inverse is most commonly used when the covariance matrix, Σ, cannot be inverted. The diagonal quadratic and diagonal linear methods are also used when the covariance matrix, Σ, cannot be inverted. The methods are based on only using the diagonal entries when calculating the inverted covariance matrix, thus making it possible to invert it.

$\gamma \geq 0$ is the regularization that is added. In some cases, it is not possible to invert the covariance matrix given in Eq. (5.20). Then regularization can be added to be able to invert the covariance matrix. Regularization is added in the following way:

$$\hat{\Sigma}_{\gamma} = (1-\gamma)\hat{\Sigma}_k + \gamma diag(\hat{\Sigma}) \tag{5.23}$$

Having the highest regularization possible, $\gamma = 1$, reduces the covariance matrix given in Eq. (5.23) to a diagonal matrix. Regularization is only used for the linear model types.

5.6 K-Nearest Neighbors (KNN)

K-nearest neighbor is a classification method in which the K-nearest observations to a given observation is used to classify the given observation. The neighborhood containing the K-nearest observations is specified as $\mathcal{N}_0$. The model calculates a probability for a given observation belonging to a certain class as a fraction of the neighborhood belonging to that class. Or in other terms, the probability for an observation x to be belonging to class j is given by [7]:

$$p(Y = j|X = x) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j) \tag{5.24}$$

The K-nearest neighbors model classifies the observations to the class with the highest probability [7].

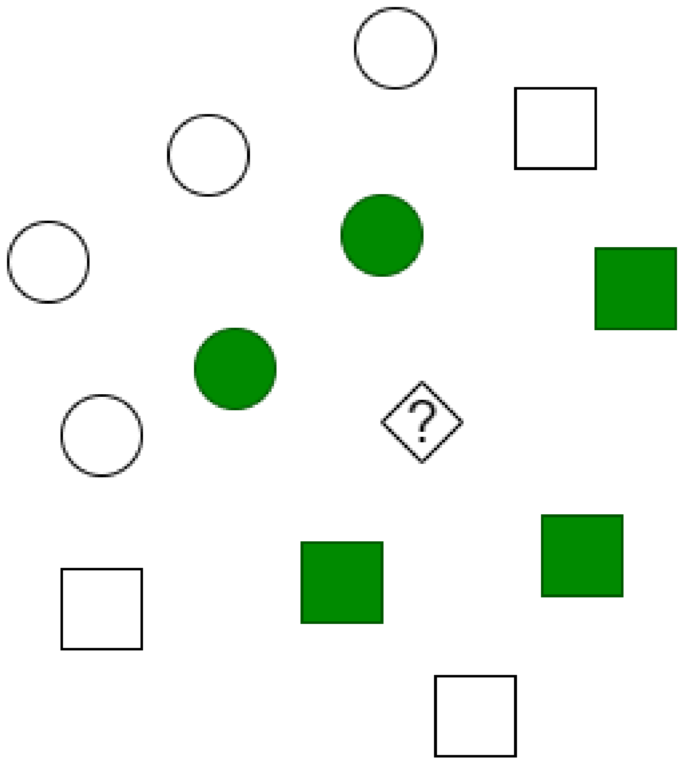


Figure 5.2: Illustration of the K-nearest neighbor classifier with 5 neighbors.

In Fig. 5.2, an example of how the KNN-classifier classifies the observations with 5 neighbors. The circles are one class, while the squares are another class. The diamond is an observation in the variable space without a label of either square or circle. The observations that are coloured green are the 5 closest neighbors by Euclidean distance to the observation that is predicted. As there are more rectangles than circles, the classifier will classify the unknown observation, the diamond, as a rectangle.

The decision boundary created by the KNN-classifier can be either very flexible or not. This is all dependent on the choice of the number of neighbors, K. Having a larger number of neighbors means that the model would average over a larger number of observations, and if taken to the extreme, namely having K as the same as the number of observations, the classifier would classify all new observations to the same class. On the other hand, having a K with low value would make the decision boundary more flexible, as a classification of a new observation would be very dependent on only the closest observations. Deciding upon the number of neighbors, K, is usually done by using k-fold cross-validation.

Defining near or closest can be done in many different ways, where the Euclidean distance may be the most intuitive distance and is given by [6]:

$$d = \sqrt{(x_1 - x_0)(x_1 - x_0)^T} \tag{5.25}$$

Where d is the Euclidean distance from observation x_0 to observation x_1. Other popular choices for determining the distance between two observations are:

The standardized Euclidean distance is given by [11]:

$$d = \sqrt{(x_1 - x_0)V^{-1}(x_1 - x_0)^T} \tag{5.26}$$

Where V is a diagonal matrix corresponding to the standard deviations of each variable. The procedure for calculating the standard deviations of each variable is shown in Appendix A.1.

The Mahalanobis distance is given by [12]:

$$d = \sqrt{(x_1 - x_0)\Sigma^{-1}(x_1 - x_0)^T} \tag{5.27}$$

Where Σ is the covariance matrix of X given by Eq. (5.20).

The Cityblock distance is given by [13]:

$$d = \sum_{j=1}^{p} |x_{1,j} - x_{0,j}| \tag{5.28}$$

The Minkowski distance is given by [12]:

$$d = \sqrt[a]{\sum_{j=1}^{p} |x_{1,j} - x_{0,j}|^a} \tag{5.29}$$

Where a is a chosen parameter.

The Chebychev distance is given by [14]:

$$d = max|x_1 - x_0| \tag{5.30}$$

The Cosine distance is given by [15]:

$$d = 1 - \frac{x_1 x_0^T}{\sqrt{(x_1 x_1^T)(x_0 x_0^T)}} \tag{5.31}$$

The Correlation distance is given by [16]:

$$d = 1 - \frac{(x_1 - \bar{x_1})(x_0 - \bar{x_0})^T}{\sqrt{(x_1 - \bar{x_1})(x_1 - \bar{x_1})^T}\sqrt{(x_0 - \bar{x_0})(x_0 - \bar{x_0})^T}} \tag{5.32}$$

The Hamming distance is given by [17]:

$$d = \frac{1}{p}\sum_{i=1}^{p}(x_0 \neq x_1) \tag{5.33}$$

The Jaccard distance is given by [18]:

$$d = \frac{\sum_{i=1}^{p}(x_0 \neq x_1) \cap (x_0 \neq 0 \cup x_1 \neq 0)}{(x_0 \neq 0 \cup x_1 \neq 0)} \tag{5.34}$$

The Spearman distance is given by [19]:

$$d = 1 - \frac{(r_0 - \bar{r_0})(r_1 - \bar{r_1})^T}{\sqrt{(r_0 - \bar{r_0})(r_0 - \bar{r_0})^T}\sqrt{(r_1 - \bar{r_1})(r_1 - \bar{r_1})^T}} \tag{5.35}$$

Where r is the ranking of observation x. The ranking vector r has the same size as x and the position in r describes which ranking the same position in x has from low to high. This means that for a vector $x = [10, 11, 2]$, the ranking vector would be $r = [2, 3, 1]$.

5.6.1 Hyperparameters

The following parameters will be treated as hyperparameters and thus subject to optimization:

- Number of neighbors $\in [1, 500]$
- Exponent for Minkowski distance $\in [3, 10]$
- Standardize $\in$ [yes,no]
- Distance type $\in$ [cityblock, chebychev, correlation, cosine, euclidean, hamming, hamming, minkowski, seuclidean, spearman, mahalanobis]

The exponent for the Minkowski distance does not include 1 since this is equivalent to the city block distance and it does not include 2 since this is equivalent to the Euclidean distance.

5.7 Support Vector Machine (SVM)

The support vector machine is a classification method that recently has become more popular. It is closely related to the maximal margin classifier and the support vector classifier. To better understand the mechanics of the support vector machine, the maximal margin classifier and the support vector classifier will briefly be explained.

If the classes are completely separable, there is a possibility to make a separating hyperplane between the observations. This means that if there are two variables there should be a line in between the variables that separate the classes completely. With three variables this line becomes a linear plane and so on. The margin is the distance of the hyperplane to the closest observations, meaning that the maximal margin is the hyperplane that is the furthest away from the closest observations by distance. These closest observations are called support vectors. Since the maximal margin is only dependent on the closest observations, it would not be affected by the move in any of the other observations, as long as this observation would not be closer to the hyperplane than the current support vectors [7]. The problem with this method arises quickly, as in most cases the classes are not separable.

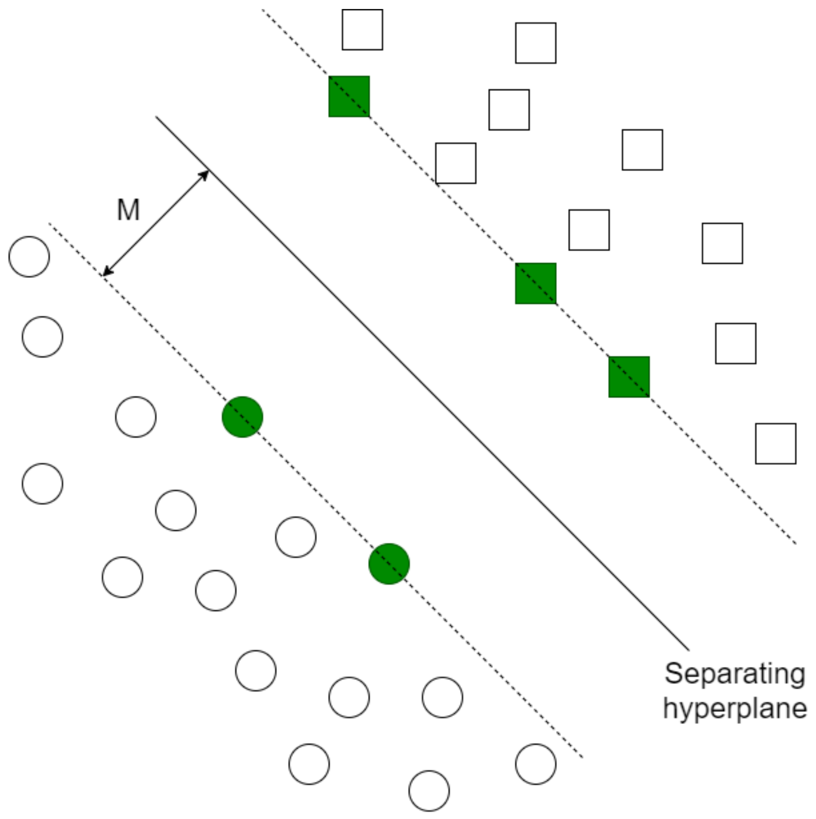


Figure 5.3: Maximal margin hyperplane.

Fig. 5.3 shows a maximal margin hyperplane. As seen from the figure, there is a separating hyperplane that perfectly separates the two classes which are shown as the circles and the squares. The support vectors for this hyperplane are marked with green. The width of the margin, M, is equally distanced from the separating hyperplane.

In the case of non-separable classes, there is a possibility to use the support vector classifier. It is also based on a hyperplane with the maximal distance from the closest observations, but with a soft margin where the margin is the distance from the hyperplane to the closes observations. The meaning of a soft margin is that there is an allowance for some of the observations to be on the wrong side of the margin. These observations are the support vectors in addition to the observations located on the margin itself, and as for the maximum margin classifier, the hyperplane is only dependent on the support vectors. The amount of observations that are allowed to be on the wrong side of the margin is a tuning parameter [7]. It is thus possible to make a hyperplane even though the classes are not completely separable. Although, both the maximal margin classifier and the support vector classifier produces linear decision boundaries, in many cases, this may not fit very well.

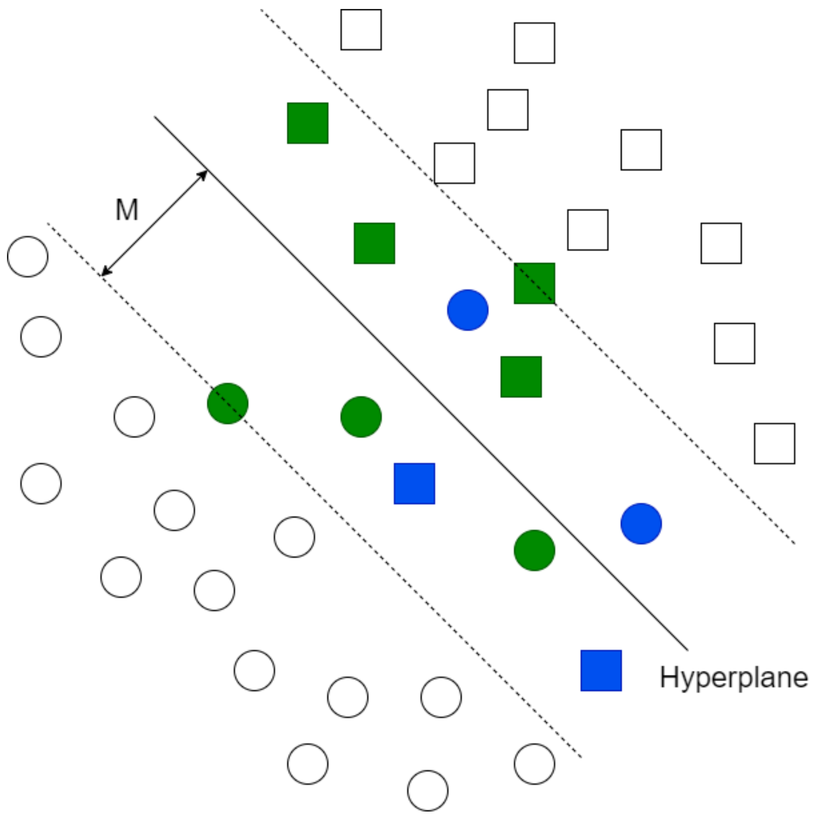


Figure 5.4: Support vector classifier.

Fig. 5.4 shows a support vector classifier with two classes, the squares, and the circles. The green and blue observations show the support vectors. The blue color specifically shows the observations that are located on the wrong side of the hyperplane.

The support vector machine utilizes a hyperplane in addition to a soft margin to create a non-linear decision boundary. An example of this is if there's a forest of trees with a small patch of flowers inside the forest. The flowers may grow a bit into the forest, but it will soon be too dark for the flowers to grow, meaning that the trees and flowers can not be completely separated. This small patch has an arbitrary shape that can be described by a function. Thus, the decision boundary between the flowers and the trees is not linear.

The shape of the hyperplane can be any shape, but the following example will show a linear shape of the hyperplane as shown in Fig. 5.3. A linear hyperplane can be described by:

$$f(x) = \beta_0 + \beta_1 x_1 \beta_2 x_2 + ... + \beta_p x_p = 0 \tag{5.36}$$

Where β describes the fitted coefficients for the hyperplane, x are the different variables and p is the number of variables. Let there be two classes which is either classified as $y_i = 1$ or $y_i = -1$. As $f(x) > 0$ on one side of the hyperplane and $f(x) < 0$ on the other side of the hyperplane, it is possible to define the following property for the hyperplane:

$$y_i(\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+...+\beta_p x_{ip})>0 \tag{5.37}$$

The goal of the support vector machine is to maximize the margin, M, namely the distance from the closest observations, but all observations must also be on the correct side of the margin, with some exceptions. These exceptions are described with slack variables, ε_i, which are variables added to the optimization problem to allow some of the observations to be on the wrong side of the margin. This is needed as in most cases, it is not possible to separate the classes completely with a hyperplane. If the slack variable for a given observation is $\varepsilon_i=0$, the observation will be located at the correct side of the margin, but if $\varepsilon_i>0$, the observation will be on the wrong side of the margin. And if $\varepsilon_i>1$ it will also be located at the wrong side of the hyperplane [7]. C is a tuning parameter that describes how much allowance there is for violations of the margin. Combining the objective function of the problem, to maximize the margin, M, the equation for the hyperplane given in Eq. (5.37) with the addition of slack terms, ε_i, in addition to the budget for the slack variables, C, gives:

$$\max_{\beta_0,\beta_1,...,\beta_2,...,\beta_p,\varepsilon_1,...,\varepsilon_n,M} M \tag{5.38a}$$

subject to:

$$y_i\left(\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+...+\beta_p x_{ip}\right)\geq M(1-\varepsilon_i) \tag{5.38b}$$

$$\sum_{i=1}^{n}\varepsilon_i\leq C \tag{5.38c}$$

$$\varepsilon_i\geq 0 \tag{5.38d}$$

$$||\beta||=1 \tag{5.38e}$$

With a linear hyperplane, the problem in Eq. (5.38) describes the support vector classifier as shown in Fig. 5.4. By adding Eq. (5.38e), the perpendicular distance from the observation to the hyperplane is given by $y_i(\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+...+\beta_p x_{ip})$ [7]. It is possible to include the constraint on $||\beta||=1$ by including it in Eq. (5.38c), which gives the following [6]:

$$\frac{1}{||\beta||}y_i(\beta_0+\beta_1 x_{i1}+...+\beta_p x_{ip})\geq M(1-\varepsilon_i) \tag{5.39}$$

Arbitrarily setting $||\beta||=\frac{1}{M}$ [6], gives:

$$\min_{\beta_0,\beta_1,\ldots,\beta_2,\ldots,\beta_p,\varepsilon_1,\ldots,\varepsilon_n} ||\beta|| \tag{5.40a}$$

subject to:

$$y_i(\beta_0+\beta_1 x_{i1}+\ldots+\beta_p x_{ip}) \geq (1-\varepsilon_i) \tag{5.40b}$$

$$\sum_{i=1}^{n} \varepsilon_i \leq C \tag{5.40c}$$

$$\varepsilon_i \geq 0 \tag{5.40d}$$

$$\tag{5.40e}$$

It is further possible to alter Eq. (5.40) to make it have better computational properties:

$$\min_{\beta_0,\beta_1,\ldots,\beta_2,\ldots,\beta_p,\varepsilon_1,\ldots,\varepsilon_n} \frac{1}{2}||\beta||^2 + C^* \sum_{i=1}^{n} \varepsilon_i \tag{5.41a}$$

subject to:

$$y_i(\beta_0+\beta_1 x_{i1}+\ldots+\beta_p x_{ip}) \geq (1-\varepsilon_i) \tag{5.41b}$$

$$\varepsilon_i \geq 0 \tag{5.41c}$$

$$\tag{5.41d}$$

Where a tuning parameter C^* is replacing the budget for the costs in Eq. (5.40) and is an L1 penalty on the slack variables. Having a large value of the tuning parameter, C^*, gives a larger penalty on the slack variables, thus reducing the margin. The example shown in Eq. (5.41) is linear, but with the introduction of kernels, it can also be extended to represent different shapes of the hyperplane. Solving the optimization problem given in Eq. (5.41) will not be described in this thesis, but it can be shown that the solution only consists of inner products [7]. With this in mind, the classifier can be described as:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \tag{5.42}$$

Where n is the number of training observations. Thus, Eq. (5.42), is exchanged with Eq. (5.38c). α can be estimated by having the inner products between all of the observations. The inner product of two observations are given by $\langle x_i, x_{i'j} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$. It should also be noted that $\alpha_i \neq 0$ only when the observations are support vectors. The inner product gives the linear decision boundary, but changing the inner product term in Eq. (5.42) to a general form gives:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(x, x_i) \tag{5.43}$$

Where $K(x,x_i)$ is the kernel function that describes how similar two observations are [7]. Thus, for the linear case, the kernel is given by:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{5.44}$$

A commonly used kernel is the polynomial kernel which is given by:

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^{d_s} \tag{5.45}$$

Where $d_s > 1$ is an integer. If $d_s = 1$, it will give back the linear kernel described in Eq. (5.44). The decision boundary with a polynomial kernel will be much more flexible than for the linear kernel. Another commonly used kernel is the radial kernel which is given by:

$$K(x_i, x_{i'}) = exp(-\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{5.46}$$

The exponential term in Eq. (5.46) gets very small when evaluating observations far from each other, meaning that the kernel has a very local behaviour [7]. There is also an advantage to use the radial kernel because it is more computationally effective [7].

For each observation, $f(x)$, is calculated as given in Eq. (5.43) along with one of the kernel functions. This is called the classification score and it also describes the distance from the hyperplane. If $f(x_j) > 0$, it means that the the observation is classified to the positive class, 1, while $f(x_j) < 0$ means that the observation is classified to the negative class, -1.

5.7.1 Hyperparameters

The following parameters will be treated as hyperparameters and thus subject to optimization:

- The penalty parameter for the slack variables: $C^* \in [0.001, 20]$
- Kernel scale $\in [0.001, 20]$
- Polynomial order $\in [2, 4]$
- Standardize $\in$ [yes, no]
- Kernel function $\in$ [gaussian, linear, polynomial]

Where the kernel scale is applied to the observations before calculating the kernels by dividing each observation by the value of the kernel scale.

5.8 Tree Ensembles

Tree ensembles are a collection of regression and classification methods that consists of using many decision trees to classify an observation. All of the methods that will be discussed are based on decision trees, so before discussing more complex methods, the decision tree will firstly be explained.

5.8.1 Decision Trees

A decision tree can be visualized as a tree. At the top of the tree, there is a root node. A node in the tree describes the point where a variable is chosen and split at a certain value. The split results in two branches which again leads to new nodes. In the end, there are leaf nodes, which decide in which class to classify an observation.

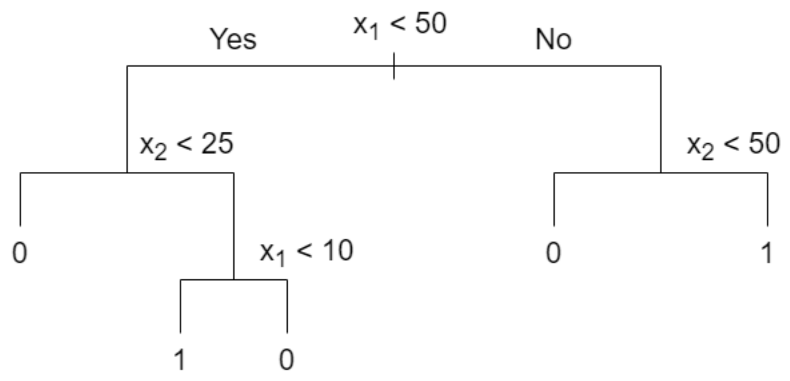


Figure 5.5: Illustration of a decision tree.

In Fig. 5.5, an example of a decision tree is illustrated. At a split of a node, the left branch describes true, while the right side is false. The tree starts at the root with the first node being $x_1 < 50$. If this is true, the algorithm will continue to the left. If not, it will continue to the right. The splitting of the nodes is repeated until one of the leaf nodes. The leaf nodes are all marked with the predicted outcome of the tree. As mentioned already, the decision tree starts from the top, but there is still a need to explain how the model chooses which variable to split and at which value to split it. At each split, the decision tree will search for the variable and the split that gives the lowest classification error:

$$E = 1 - \max_k(\hat{p}_{m,k}) \tag{5.47}$$

Where $\hat{p}_{m,k}$ is the probability for an observation to be in class k within the region m. Although other functions for deciding the split is used, such as the Gini index:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{m,k}) \tag{5.48}$$

As can be seen from Eq. (5.48), the Gini index will be close to 0 either if all of the $\hat{p}_{m,k}$ are close to 0 or close to 1 [7]. Another commonly used function is the entropy or deviance:

$$D = -\sum_{k=1}^{K} \hat{p}_{m,k} log(\hat{p}_{m,k}) \tag{5.49}$$

The entropy function in Eq. (5.49) will also be close to 0 if all of the $\hat{p}_{mk}$ are close to 0 or 1. Both the entropy in Eq. (5.49) and the Gini index in Eq. (5.48) can be seen as measurements of purity for each node [7], with a lower value meaning a more pure node. This means that the algorithm will choose the split that gives the purest node. Although these functions are used to decide the split for each node, the algorithm does not look further down in the tree. This means that the current split will only be made with information of the current node and does not take into account splits further down the tree and is therefore called a greedy approach. Although decision trees are a very visual approach and can easily be explained, this method generally performs worse than other models because of the decision trees having a high variance [7]. This means that the model is very sensitive to changes in the data.

Both the Gini index and the entropy can be used to make decision trees. Although, earlier studies show that there is no conclusive evidence of which type of split criterion is the best performing criterion [20, 21]. For the interested reader, information on the CART algorithm which is used to create the decision tree is given in [22].

5.8.2 Bagging

Bagging or bootstrap aggregating is a bootstrap based method that reduces the variance of a machine learning method. Since the decision trees suffer from high variance, it is very often used for this purpose.

The bootstrap is a resampling method that resamples with replacement [7]. Given n observations in a data set, n, randomly selected observations with replacement is chosen to create a new data set. This is repeated B times, to make B new data sets. The average of the mean is given by:

$$var(\bar{x}) = \frac{\sigma^2}{n} \tag{5.50}$$

The full derivation of Eq. (5.50) can be found in Appendix A.5. The variance of the mean is thus lower than the variance of a single data set. Thus, it is possible to train B different models with the B bootstrap data sets and use these models for prediction. Averaging over the predictions of these data sets will give a total prediction with higher accuracy, as the variance of mean is lower than the variance for one data set [7]. Using this approach, the bagging prediction is given by:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x) \tag{5.51}$$

Which again will give a prediction with lower variance and thus improve the prediction accuracy of the decision tree. Choosing the number of bootstrap data sets, B, is not treated as a hyperparameter, as it will not lead to overfitting by having too many data sets. Instead, a sufficiently high number is required, as having a too high number would increase computational time.

5.8.3 Random Forest

Random forest uses bagging on decision trees to make predictions, but there are also some differences. When building each separate decision tree, the algorithm is only allowed to choose from m randomly selected variables each time there is a split. These m randomly selected variables are updated each time. The number of variables selected are often chosen to be $m \approx \sqrt{p}$, where p is the total number of variables [7]. The reason for doing this is in the bagged trees, there will usually be a dominant variable at the top for all the bagged trees. This will cause all of the bagged trees to be similar and correlated. The mean of correlated variables does not give a high reduction in the variance of the mean [7]. By only allowing the model to choose from m variables each time, the trees will be less similar and thus less correlated.

5.8.4 Boosting

As mentioned, decision trees suffer from high variance. Bagging or random forest can be used to reduce the variance, but another method that is also used for this purpose is boosting. In random forest and bagging, bootstrap data sets are used to train decision trees. Since each data set is picked independently to the other data sets, the decision trees will be independent of the other decision trees. Boosting, on the other hand, grows the trees sequentially. The method is based on creating many decision trees, each one is fit to the residuals of the previous tree [7]. In comparison to the decision tree where the prediction is only based on one tree, boosting trees is a method that learns slowly from the errors of the previous trees. In boosting the number of decision trees, M, are treated as a hyperparameter as too many dependent decision trees may cause overfitting.

Algorithm 2: Basic algorithm for boosting. [5].

```
Randomly select without replacement, $m$ observations of the total $n$ observations in the data
  set to make a data set $F_1$ ;
Train a decision tree on the data set $F_1$ to make model $H_1$;
Randomly select half of the observations from misclassifications by $H_1$, to make $F_2$;
Train a decision tree on the data set $F_2$ to make model $H_2$;
Use all observations where $H_1$ and $H_2$ have different predictions to create $F_3$;
Train a decision tree on the data set $F_3$ to make model $H_3$;
The final model is based on a majority vote from the predictions of $H_1$,$H_2$ and $H_3$;
```

It is possible to extend Algorithm 2 by also choosing the individual decision trees by boosting [5]. In this thesis, a wide variety of different boosting methods will be presented.

AdaBoost Algorithm

The AdaBoost is an extension of Algorithm 2, but it differs from the first boosting algorithm by that it weights the original data set instead of making subsets [5] when creating the decision trees.

Algorithm 3: AdaBoost algorithm [5].

```
The weights $w$ have the initial conditions: $w_i = \frac{1}{n}, \quad i \in \{1,...,n\}.$ ;
$m = 1$ ;
while $m \leq M$ do
    Observation $i$ is weighted by $w_i$ for $i \in \{1,...,n\}$. Create a decision tree, $H_m$ with the
      weighted data set;
    The weighted error $err_m$ is found by: $err_m = \sum_{i=1}^{n} w_i \cdot h(-y_i H_m(x_i))$;
    Find $\alpha_m = \frac{1}{2} ln(\frac{1-err_m}{err_m})$;
    Update the weights for all of the observations: $v_i = w_i exp(-\alpha_m y_i H_m(x_i))$;
    Find $S_m = \sum_{j=1}^{n} v_j$ ;
    The normalized weights are given by: $w_i = \frac{v_i}{S_m}$;
    m = m+1;
end
The final model is given by the majority vote: $H = sign(\sum_{j=1}^{M} \alpha_j H_j(x))$;
```

Where h is the heaviside function. err_m is the weighted classification error and α_m is the weight of model H_m.

The following boosting algorithms are all variants of AdaBoost.

5.8.5 Logit Boost

The Logit Boost algorithm calculates the working response which is the training error for the current model [23]. The algorithm improves the error by fitting the errors by the weighted least squares estimator, which can be seen as using Newton steps [5].

Algorithm 4: Logit Boost algorithm [24].

```
The weights $w_i$ have the initial conditions: $w_i = \frac{1}{n}, \quad i \in \{1,...,n\}.$ ;
$p(x_i) = 0.5$;
for $m = 1 : M$ do
    The working response is given by: $f_i = \frac{y_i^* - p(x_i)}{p(x_i)(1-p(x_i))}$;
    Update the weights: $w_i = p(x_i)(1 - p(x_i))$;
    The decision tree $H_m$ is fitted by weighted least squares of $z$ and $x$ with the weights, $w$;
    $H(x) = H(x) + \frac{1}{2} H_m(x)$;
    $p(x) = \frac{exp(H(x))}{exp(H(x)) + exp(-H(x))}$
end
The final model is given by the majority vote: $H = sign(\sum_{j=1}^{M} \alpha_j H_j(x)).$ ;
```

Where $p(x_i)$ is the current probability of classifying observation i to class 1. y_i^* is the predicted class label denoted as $\{0,1\}$ instead of $\{-1,1\}$. Instead of using classification trees, the Logit Boost algorithm uses regression trees. The regression trees minimize the mean squared error (MSE). The model divides the observation space into regions R. The objective is to find the variable, X_j, and where to divide the region $X_j < s$, such that the MSE is as low as possible [7] which is given by Eq. (A.5).

5.8.6 Gentle AdaBoost

The Gentle Boost is quite similar to the Logit Boost algorithm as it uses Newton steps. Although it is more stable since it does not use logarithms of ratios to calculate the probabilities. The Gentle Adaboost algorithm is given by [24]:

Algorithm 5: Gentle AdaBoost algorithm [24].

```
The weights $w$ have the initial conditions: $w_i = \frac{1}{n}, \quad i \in \{1,...,n\}$. ;
for $m = 1 : M$ do
    The regression tree $H_m$ is fitted by weighted least squares of $y$ and $x$ with the weights, $w$;
    $H(x) = H(x) + H_m(x)$;
    Updating the weights: $w_i = w_i exp(y_i H_m(x_i))$. Normalize $w_i$ for $i \in \{1,...,n\}$;
end
The final model is given by the majority vote: $H = sign(\sum_{j=1}^{M} \alpha_j H_j(x))$. ;
```

The Gentle AdaBoost also uses regression trees instead of classification trees as explained in Section 5.8.5.

5.8.7 RUSBoost

RUSBoost is designed for imbalanced data sets where there are much more positive observations or the other way around. Instead of using weighted data sets of the original data set, it makes new data sets by undersampling the major class [25]. Undersampling means removing random observations belonging to a certain class, in this case, the major class. The amount of undersampling is decided by the percentage of observations belonging to the minor class, N. The RUSBoost algorithm is given by [25]:

Algorithm 6: RUSBoost algorithm [25].

```
The weights $w$ have the initial conditions: $w_i = \frac{1}{n}, \quad i \in \{1,...,n\}$;
for $m = 1 : M$ do
    Data set $F_1$ is consisting of $k$ observations from undersampling the original data set with
      the corresponding weightings $w^*$.;
    Use the data set $F_1$ to create a decision tree model, $H_m$;
    The weighted error $err_m$ is found by: $err_m = \sum_{i=1}^{k} w_i^*(1 - y_i - y_i^*)$;
    Find $\alpha_m = \frac{err_m}{1-err_m}$;
    Update the wights for all of the observations in $F_1$: $v_i = w_i^* \alpha_m^{-\frac{1}{2}w_i(1-y_i-y_i^*)}$;
    Normalize the weights: $w_i^* = \frac{v_i}{\sum_{j=1}^{k} v_j}$;
end
The final model is given by the majority vote: $H = argmax(\sum_{j=1}^{M} ln(\frac{1}{\alpha_j})H_j(x))$;
```

Where y_i^* are the class labels predicted by H_{m-1} which are decoded as $\{0,1\}$.

5.8.8 Hyperparameters

The following parameters will be treated as hyperparameters and thus subject to optimization:

- Method $\in$ [random forest, GentleBoost, LogitBoost, AdaBoost, RUSBoost]
- Number of trees $\in [10, 5000]$
- Learning rate $\in [0.00001, 1]$
- Minimum leaf size $\in [1, 138202]$
- Random variables chosen at split $\in [1, 24]$
- Ratio for RUSBoost $\in [1, 10]$

Where the maximum value for the range of the minimum leaf size is half of the number of total observations. It describes the minimum observations that each leaf node needs to contain at a minimum. Standardization is not added as a hyperparameter since the decision trees are scale-invariant. The ratio for RUSBoost is how many observations from the major class that is picked in comparison to the observation of the minor class.

Chapter 6

Results and Discussion

The results of this thesis are divided into three main parts. Firstly, a PCA is done in addition to data exploration. The second part shows the results from all of the tuned machine learning models with the optimal hyperparameters for the breakage and droplet data set. In the end, a model for both data sets is chosen and implemented in the image analysis software. In the last part, the implemented machine learning models are tested on new experiments to evaluate whether the new models are better than the old tracking logic.

Data was gathered for the machine learning part by using the old tracking logic and manual corrections. A total of 160 experiments were processed. These experiments were chosen from the already recorded experiments with a focus on variety in droplet sizes. Some of the droplets in the experiments were deleted due to the difficulty of accurately being able to classify the droplets correctly. The most complex experiments were also not processed due to the time constraint on the project, as some of the complex experiments would take days to process. Although, the models are still expected to be able to process some of these very complex experiments, as the most important part for the data set is to have a wide variety of data that accurately contains the responses with the related variables. The processed data consists of two different data sets with the following properties:

Table 6.1: Data set properties.

Data set	Number of observations	Positive rate	Negative rate
Droplet	276405	6.9×10^{-1}	3.1×10^{-1}
Breakage	276405	1.5×10^{-3}	1.0×10^{-1}

In Table 6.1, both the droplet data set and the breakage data set has a large number of observations, which will most likely increase the accuracy of the machine learning models. The positive rate and negative rates are the rates of positive and negative observations. For the droplet data set the positive rate and negative rate are pretty similar, but for the breakage data set, there are almost no positive observations compared to the negative observations. This means that if a model only predicts that all the observations are negative, it will still only have an error rate of 1.5×10^{-3}.

The Statistics and Machine Learning Toolbox in MATLAB was used to do the PCA and for creating the machine learning models [26]. To find the best suited hyperparameters for the different machine learning models, the hyperparameters for the different models were optimized with the use of Bayesian optimization as described in Appendix A.12 with 30 iterations. For the droplet model, AUC based on ROC is used as the objective function that is maximized for Bayesian optimization as described in Section 5.3. Since the breakage data set is heavily imbalanced, the objective that is maximized in this case is AUC based on PRC. Both types of AUC is calculated by 5-fold cross-validation as described in Section 5.2. Thus, the AUC is given by the average AUC for all of the folds:

$$A\hat{U}C = \frac{1}{k}\sum_{i=1}^{k} AUC_i \tag{6.1}$$

Where k is the number of folds.

The classification error was also calculated by k-fold cross-validation by Eq. (5.4) with a threshold of 0.5.

6.1 Principal Component Analysis and Data Exploration

A principal component analysis was performed as described in Section 5.1. The droplet and breakage set are very similar data sets. The only difference between the two sets is the response. Since PCA is an unsupervised learning method, the response is not used when creating the model. Thus both data sets will give the same result and only one principal component analysis was performed.

PCA can be used for decreasing the variable space by using the principal components that explain the most of the variance, but in this case, it is used for finding the variables that contain most of the variance. With having 24 different variables, it is a possibility to make up to 24 different PCAs, with the first one containing most of the variance, the second having the next most variance and so on. The variance captured for each principal component describes in which directions there is most variance in the data set.

In this thesis, the PCA is only used to consider in which directions the data has the largest variance, but it is also possible to create the principal components that capture most of the variance to create a new data set consisting of the principal components and the response. The principal components can be used directly for machine learning to reduce computational issues. Some information is lost when using the principal components for the machine learning models, which is why it is considered for this project. There is also no computational issue when there are only 24 variables. For this thesis, it is most important that the two machine learning models can predict the response with high accuracy or precision.

For the purpose of data exploration, some of the variables are plotted against each other. Although, due to the sheer amount of variables and having two types of responses, only some interesting combinations are shown. 1000 random negative and positive observations are used for both the droplet set. For the breakage set, all of the 417 positive observations were used in addition to 1000 random, negative observations.

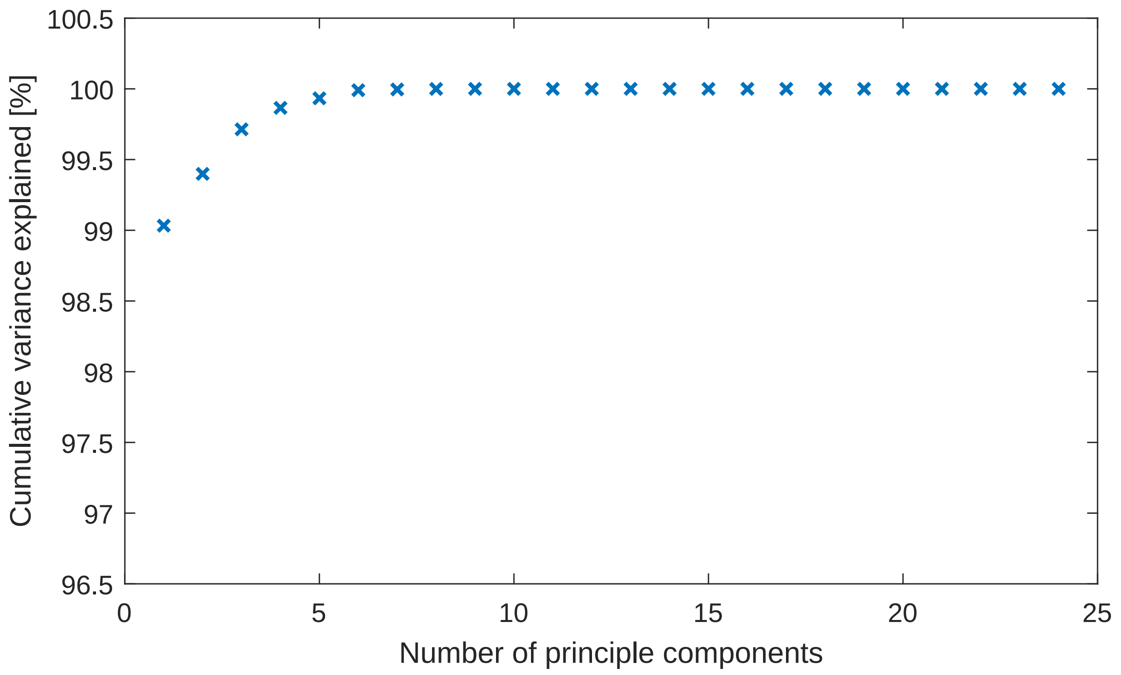


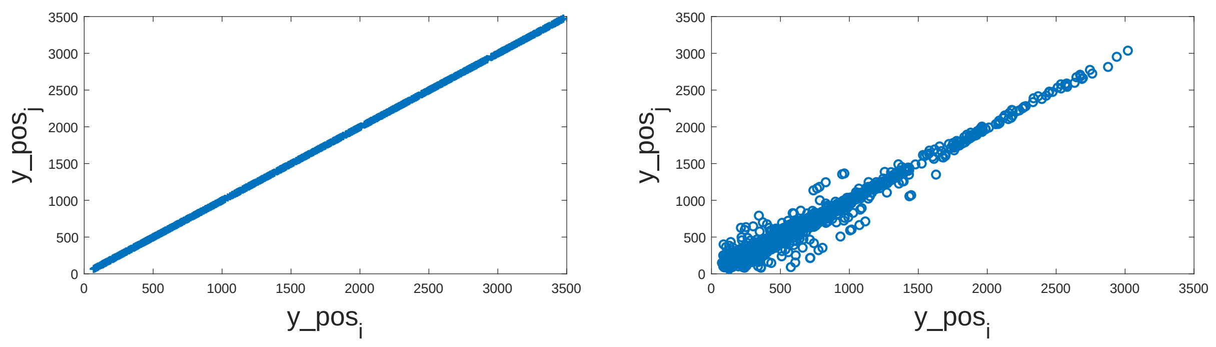
Figure 6.1: Cumulative variance explained.

From Fig. 6.1, the variance explained has almost approached 100% when using six principal components and there is little improvement when using seven or more principal components. The first principal component covers around 99% of the variance in the data set. The percentage of variance explained for the different principal components is given in Appendix A.6. The different principal components were calculated according to Section 5.1. This gave the following loadings for the first six principal components:

Table 6.2: Loadings for the first six principle components.

Variable	PC 1	PC 2	PC 3	PC 4	PC 5	PCA 6
$DD\_contour$	-0.01	0.10	0.56	0.00	0.00	0.00
$DD\_contour\_max$	-0.01	0.10	0.57	-0.03	0.00	0.00
$DD\_centroid$	-0.01	0.10	0.58	-0.01	0.01	0.00
$Area_i$	0.00	0.00	0.00	0.00	0.00	0.00
$Area_j$	0.00	0.00	0.00	0.00	0.00	0.00
x_pos_j	0.00	0.67	-0.11	0.21	0.04	0.00
y_pos_j	0.71	0.00	0.01	0.00	0.70	0.10
$Diameter_i$	0.00	0.00	0.00	0.00	0.00	0.00
$Deformation_i$	0.00	0.00	0.00	0.00	0.00	0.00
$Eccentricity_i$	0.00	0.00	0.00	0.00	0.00	0.00
$EquivDiameter_i$	0.00	0.00	0.00	0.00	0.00	0.00
$Perimeter_i$	0.00	0.00	0.00	-0.01	0.00	0.00
$Orientation_i$	0.00	-0.20	0.05	0.68	0.09	-0.70
$Diameter_j$	0.00	0.00	0.00	0.00	0.00	0.00
$Deformation_j$	0.00	0.00	0.00	0.00	0.00	0.00
$Eccentricity_j$	0.00	0.00	0.00	0.00	0.00	0.00
$EquivDiameter_j$	0.00	0.00	0.00	0.00	0.00	0.00
$Perimeter_j$	0.00	0.00	0.00	-0.01	0.00	0.00
$Orientation_j$	0.00	-0.20	0.05	0.68	-0.10	0.70
x_pos_i	0.00	0.67	-0.11	0.20	-0.05	0.00
y_pos_i	0.71	0.00	0.02	0.00	0.70	-0.10
$aAxis_i$	0.00	0.00	0.00	0.00	0.00	0.00
$bAxis_i$	0.00	0.00	0.00	0.00	0.00	0.00
$aAxis_j$	0.00	0.00	0.00	0.00	0.00	0.00

In Table 6.2, the highest loadings for the first principal component are y_pos_j and y_pos_i, which is the location of the droplet in the y-plane. i denotes a droplet in the previous frame, while j denotes a droplet in the current frame. All the recorded droplets enter the breakage channel at the beginning and exit through the top, so the position in the height plane is highly variable. Although, the position of the droplet does not indicate uniquely whether there is a breakage or not or if two droplets are the same, the probability of breakage or if it is the same droplet can change with the position in this specific column. The loadings for the first principal component for y_pos_j and y_pos_i are very similar in size. This indicates that there may be a correlation between these two variables. It makes sense as the position of the droplets in the current frame is dependent on the droplets in the previous frame.

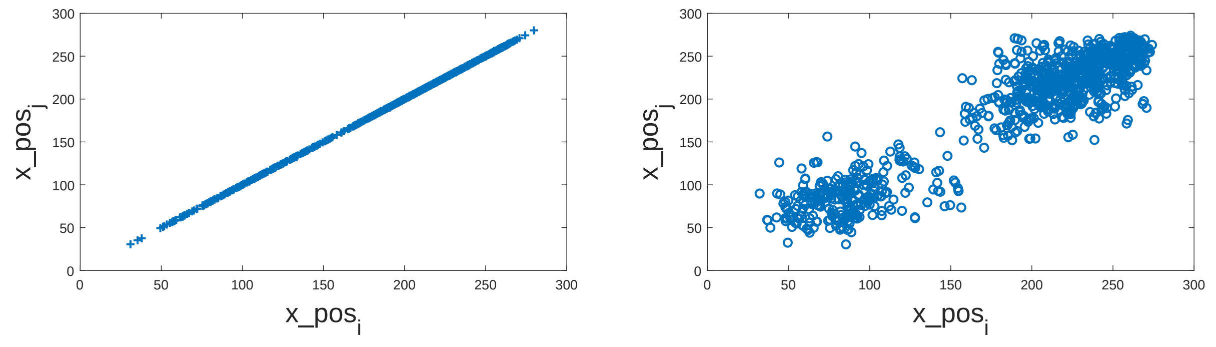


(a) Positive response. (b) Negative response.

Figure 6.2: y_pos_j plotted against y_pos_i with the same droplet response.

For all the positive responses given in Fig. 6.2a, there seems to be a linear correlation between the position in y_pos_i and y_pos_j. If a droplet is the same as the one in the previous frame, there is very little difference in y_pos_i and y_pos_j compared to if the droplet in the current frame is a different droplet than the one in the previous frame. For the negative responses given in Fig. 6.2b, there also seems to be a somewhat linear correlation, but with more observations for lower values of y_pos_i and y_pos_j. All droplets travel from the beginning to the column to the end of the column and even if the droplets break, there is not a very large difference in the positions in the y-direction. Therefore it is almost a linear correlation between the variables, with some observations being different.

The second principal component is mostly dominated by x_i and x_j. Together with y_i and y_j, the position of the droplets in the two-dimensional space is found.

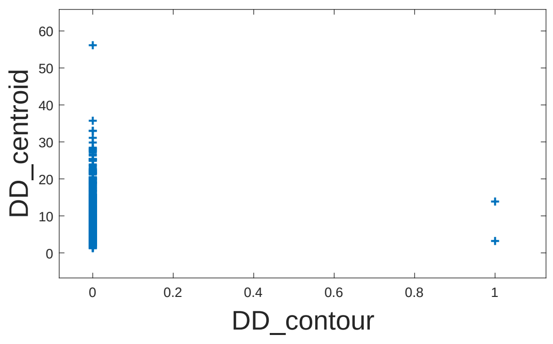


(a) Positive response. (b) Negative response.

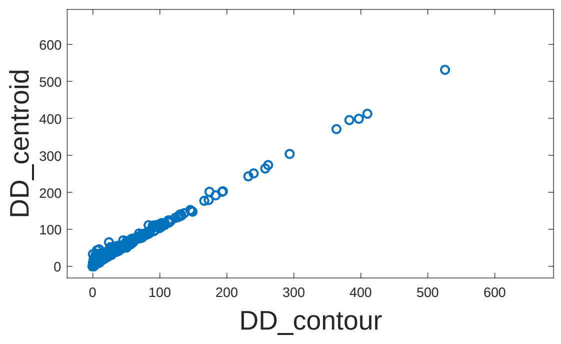
Figure 6.3: x_pos_j plotted against x_pos_i with the same droplet response.

Looking at Fig. 6.3a, there is a linear correlation between the position, x_pos_i and x_pos_j. If a droplet is the same as one in the previous frame, it has a very similar position in the x-plane compared to if it is not the same droplet. The negative response given in Fig. 6.3b, shows two clouds of observations, with very few observations in the middle. In the breakage channel, the turbulence most likely drags the droplets to one or the other side, thus causing fewer observations in the middle.

The third principal component is mostly dominated by *DD_contour*, *DD_contour_max* and *DD_centroid*, which all have similar values. With having similar loadings they are most likely correlated. When the contour of the droplet changes, the centroid also changes.



(a) Positive response.

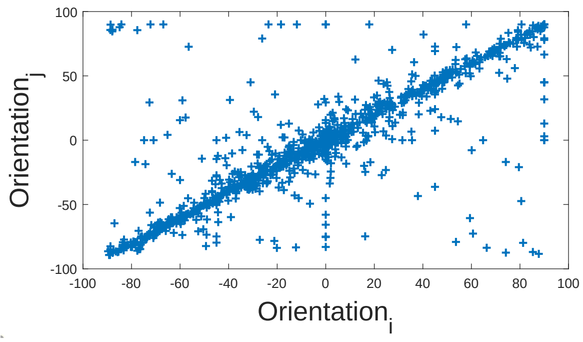


(b) Negative response.

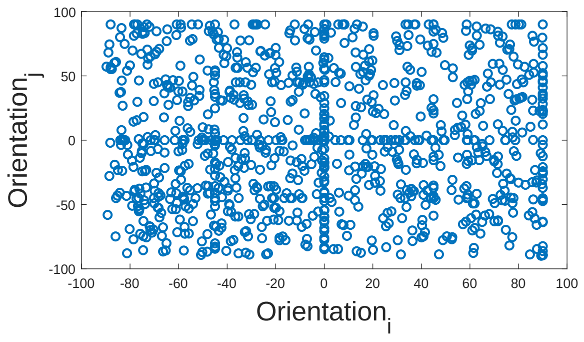
Figure 6.4: *DD_Centroid* plotted against *DD_Contour* with the breakage response.

From Fig. 6.4a, the *DD_contour* is 0 when there is a breakage, but for the negative responses there seems to be a linear correlation between the *DD_contour* and *DD_centroid*. Although the intuitive thought may be that the *DD_contour* should be large when there is a breakage, the data set is made from comparing all droplets in the current frame with all droplets in the previous frame. For two different droplets, the *DD_contour* is large, and when checking the droplet that breaks, the *DD_contour* is small in comparison. For the negative responses, as shown in Fig. 6.4b, there seems to be a linear correlation between the variables, which also confirms the suspicion of correlation from the PCA.

The fourth principal component is dominated by $Orientation_i$ and $Orientation_j$. There is a correlation between the orientation as two droplets that are equal in the previous and current frame has a very similar orientation. The fifth principal component is dominated by y_i and y_j while the sixth principal component is dominated by $Orientation_i$ and $Orientation_j$.



(a) Positive response.



(b) Negative response.

Figure 6.5: $Orientation_j$ plotted against $Orientation_i$ with the same droplet response.

As seen from Fig. 6.5a, for the positive responses, there seems to be a very linear dependency of $Orientation_i$ and $Orientation_j$. For the negative responses there seems to be less correlation between the variables. Although, there is a higher concentration of negatives and positives when either of the variables are 0, because in very many observations there is little to no change in the droplet orientation.

All in all, there are many variables that are correlated as seen from the PCA. In addition, the plots show a linear dependence on many of the variables. This is to be expected as many of the variables only change very slightly from a droplet in one frame to the same droplet in the next frame. Although, these changes are larger for droplets that are not the same.

6.2 Old Image Analysis Software

The old image analysis software was tested on the data set for comparison to the machine learning models. Although it should be noted that in the implementation of the old tracking, only certain observations are evaluated to check for breakages. The AUC was calculated for both the droplet and the breakage data set. The following results were obtained from testing the old image analysis software on the droplet and breakage data sets:

Table 6.3: The results from using the old image analysis software to predict the observations in the data set.

Data set	1-AUC	Classification error
Droplet	4.79×10^{-4}	9.87×10^{-1}
Breakage	9.99×10^{-1}	2.05×10^{-1}

From Table 6.3, the old image analysis software performs well for the droplet data set, having a very low 1-AUC, and a low classification error. For the breakage data set, the old model does not perform well as it has a very high 1-AUC in addition to a classification error of close to one. The reason why the old image analysis software does not perform well in this case is that it will predict breakages too often, giving a lot of false positives. The old model struggles in detecting the breakages, so to increase the chances of detecting the true positives, it allows for more false positives. The false positives can then be removed by manual corrections. The PRC plot and ROC plot is shown in Appendix A.10.

6.3 Logistic Regression

A logistic regression model was created as described in Section 5.4 with regularization and the hyperparameters given in Section 5.4.4. The following results was obtained from the optimization:

Table 6.4: Results from optimizing a logistic regression model for both data sets.

Data set	AUC	Classification error	Lambda	Standardize	Regularization
Breakage	4.84×10^{-2}	2.50×10^{-4}	5.07×10^{-10}	yes	ridge
Droplet	1.93×10^{-5}	9.66×10^{-4}	6.33×10^{-9}	yes	lasso

For both data sets, the best logistic regression model used ridge as regularization instead of the lasso. For the breakage data set, the classification error is much lower than the positive rate of the data set as given in Table 6.1 which means that the model is able to find patterns in the variables that describe the breakage.

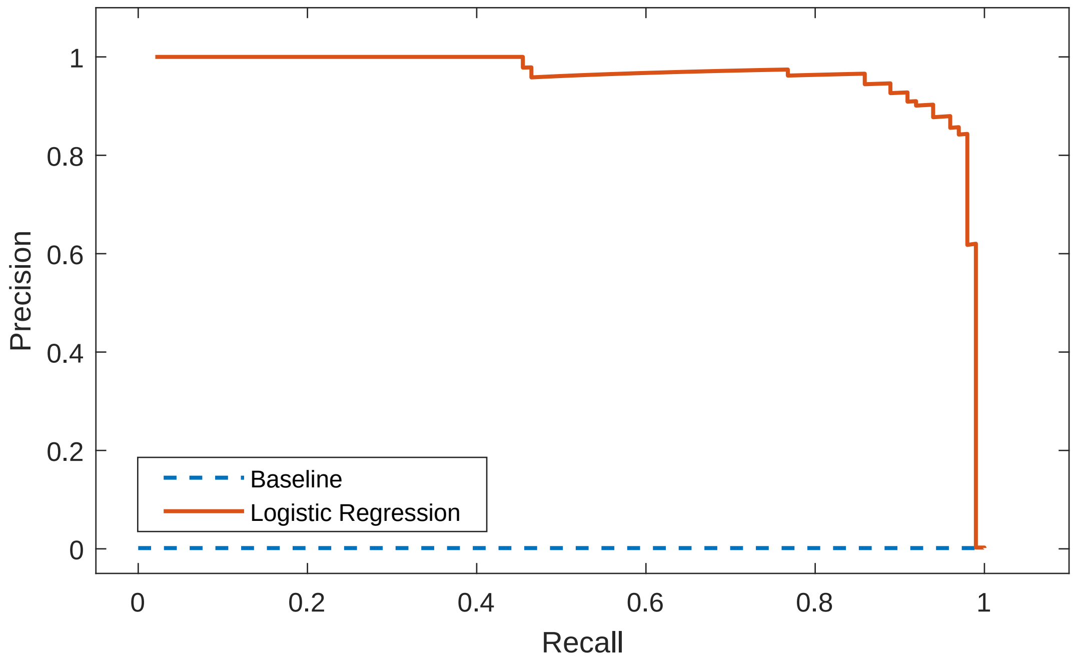


Figure 6.6: Precision-recall plot for the breakage data set using logistic regression.

Fig. 6.6 is made from using four folds for training and one for testing. The precision is lower when the recall is higher. When the recall approaches one, it means that the prediction by the model captures all the positive values, but at the cost of also predicting some of the true negatives to be positives. When this happens, the precision as given in Eq. (5.8) drops.

The odds for classifying a positive observation is given in Eq. (5.12) and the estimated coefficients, $\beta_0, \beta_1, ..., \beta_{24}$ are given in Table A.2. Comparing the coefficients, the magnitude will tell something about how much the variable is being weighed in terms of odds. Looking at Table A.2 for the breakage data set, β_1 which is the coefficient for *DD_contour*, has the largest magnitude without the intercept and β_3 which is the coefficient for *DD_contour_max* has the second largest magnitude. As *DD_contour* describes the distance from the contours of the droplets, it means that the most important factor for predicting a breakage is that the distance is small between the contours. Decreasing the variable *DD_contour* by one will multiply the odds by $\frac{1}{exp(\beta_1)}$, thus increasing the odds for a breakage. This makes sense, as the distances from the contours must be close for it to be a breakage. In the actual breakage, the distance from one contour to the other contour is zero, but as the experiment is based on images, the contours should be very close to zero. On the other hand, the variable *DD_centroid* contributes negatively to the odds when increased. The variable describes the length from the centroids of the droplets. Although it may be intuitive that the length from the centroids should have a positive effect on the odds for breakage, it is in this case opposite. Since all the droplets in the current frame are compared with all the droplets in the previous frame, the length from the centroid of one droplet to another will in most cases be very large. When a droplet breaks, on the other hand, the centroids will be relatively close.

For the droplet set, β_3, which is the *DD_centroid* variable that has the largest magnitude, while β_2, which is the *DD_contour_max* that has the second largest magnitude. The length from one centroid to the other has a large negative effect on the odds when increased. When one droplet is far away from the other, it is most likely not the same droplet, so it makes sense that it has a large coefficient. The *DD_contour_max* has also a negative effect on the odds. The variable describes the furthest distance from the contour of the two droplets and has a much larger coefficient than the minimum distance between the contours. In many cases, the droplets may be different, but still, be very close or even overlapping at certain points. When using the maximum distance between the contours, the algorithm will disregard the cases where the droplets are close at some points in the contour, but further away in other points as the maximum distance will be large and thus reduce the odds.

A classification error of 9.67×10^{-4} for the droplet data set is low, but in every experiment there are a lot of observations, meaning that even with such a low classification error, there would still be some misclassified observations, and even more in the more complicated experiments.

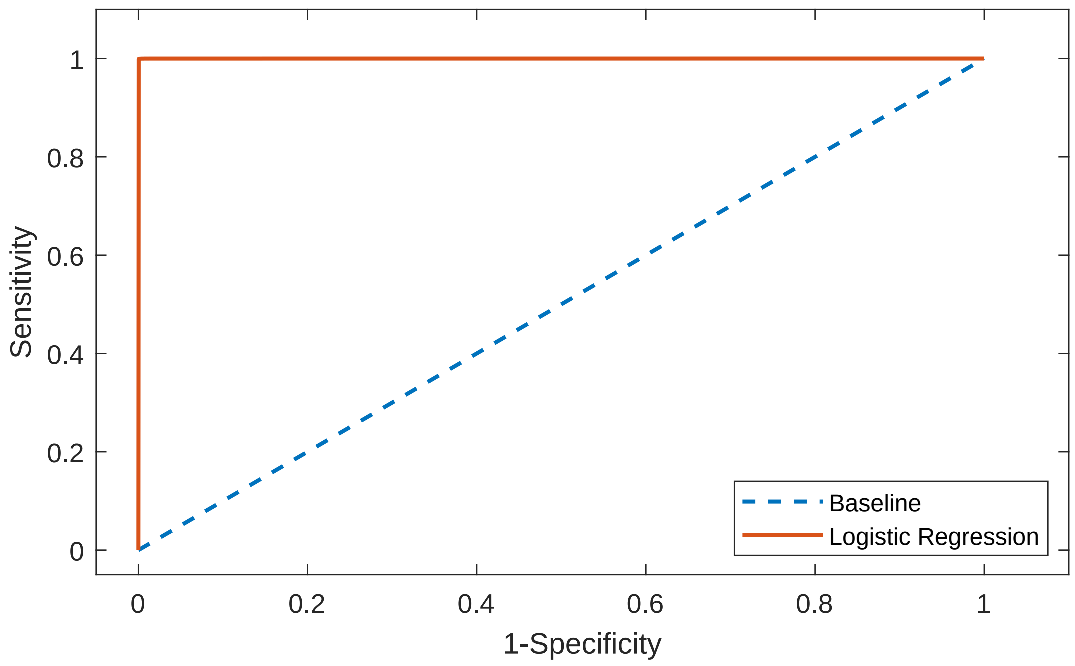


Figure 6.7: ROC curve for the droplet data set using logistic regression.

Fig. 6.7 shows that the sensitivity is 1 for almost all of the values of the specificity and the other way around which means that the model separates the true observations from the negative observations well for all values of the threshold for the cut-off. This is further confirmed by the 1-AUC which in this case is 1.93×10^{-5} as given in Table 6.4.

For the droplet data set, the largest positive coefficient is β_0 which is the bias term, meaning that if all variables are set to 0, the odds would be $exp(\beta_0)$. As opposed to the coefficient for the breakage data set, a lot of the coefficients are positive, but also with a low and similar value. This means that there are a lot of variables that contribute approximately the same to increase the odds, while a few contribute to decreasing the odds.

All in all, the logistic regression does well at explaining the responses in both data sets by using the available variables.

6.4 Discriminant Analysis

The discriminant analysis models were created according to Section 5.5 with optimizing the model type and γ as explained in Section 5.5.3. Bayesian optimization was used as explained in Appendix A.12.

Although the covariance matrix was invertible, the regularization parameter γ was still added as a hyperparameter. The performance of the models was much higher with some regularization. The reasoning behind this is that the covariance matrix given in Eq. (5.20) is only an estimate, thus being able to adjust it may improve the performance of the model.

Table 6.5: Results from optimizing a discriminant analysis model for both data sets.

Data set	1-AUC	Classification error	γ	Standardize	Model type
Breakage	8.27×10^{-1}	7.40×10^{-3}	6.43×10^{-2}	1	Linear
Droplet	1.11×10^{-4}	1.17×10^{-2}	0.00	1	Quadratic

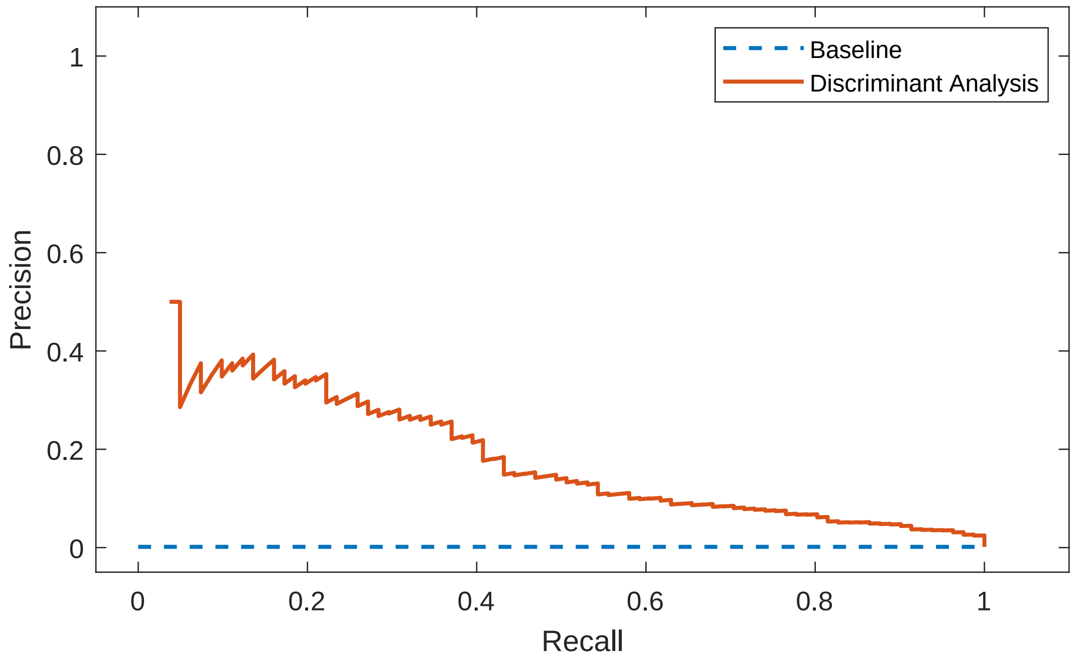


Figure 6.8: PRC curve for the breakage data set using discriminant analysis.

The AUC for the breakage data set is very low as can be seen from the PRC plot in Fig. 6.8. This means that the discriminant model struggles to separate the classes. When the recall which is the true positive rate approaches zero, the precision increases. This makes sense as when increasing the threshold for the cut-off, the recall will also decrease. Increasing the threshold for the cut-off will also increase the precision as only the observations that have a very high probability of being a breakage will be classified as one. For the breakage case, the linear discriminant analysis performed better than the quadratic discriminant analysis.

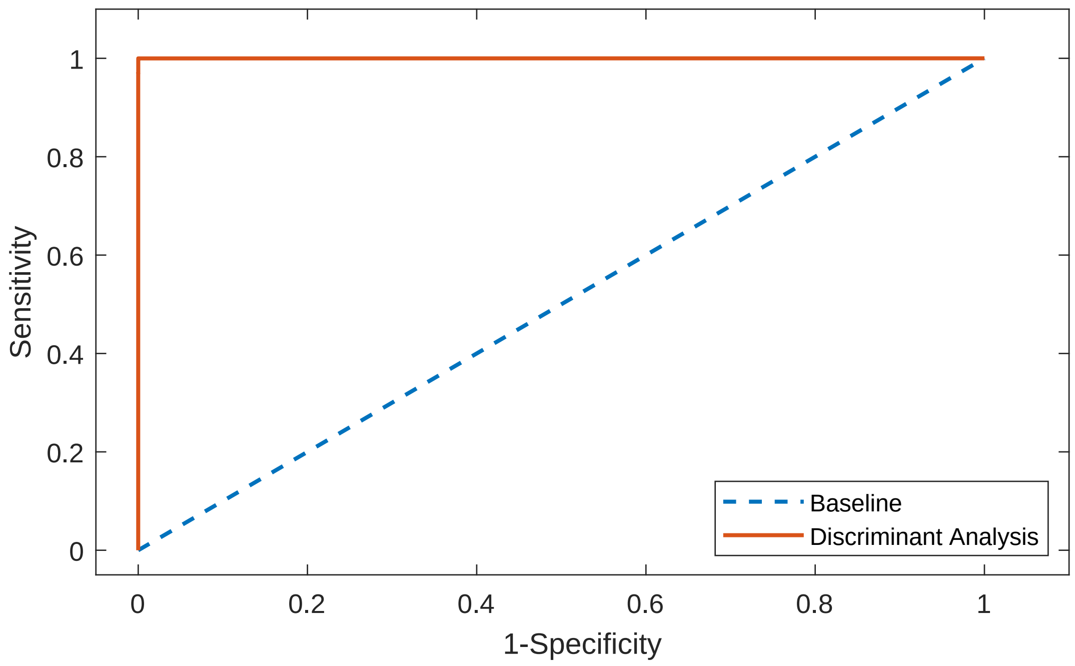


Figure 6.9: ROC curve for the droplet data set using discriminant analysis.

The 1-AUC for the droplet data set is low, which also can be seen in Fig. 6.9. Even though the classification error is pretty low at 1.17×10^{-2}, it will still make incorrect predictions on an experiment due to the number of predictions the model has to make. The best performing model type for the droplet data set was the QDA in comparison the breakage data set in which LDA was preferred. For the QDA, each class has a different covariance matrix as explained in Appendix A.4 which in this case is a better fit.

All in all, the discriminant analysis fails to separate the two classes for the breakage data set, but is more successful for the droplet data set. One of the assumptions behind the discriminant analysis models is that all the variables belong to a multivariate normal distribution. If the variables do not belong to the multivariate normal distribution, it may cause the discriminant analysis models to fail to separate the different classes.

6.5 K-Nearest Neighbors

The KNN-models for the breakage and droplet set were created according to Section 5.6 and optimized with the hyperparameters given in Section 5.6.1.

Table 6.6: Results from optimizing a KNN model for both data sets.

Data set	1-AUC	Classification error	Number of neighbors	Distance	Standardize
Breakage	1.13×10^{-1}	4.74×10^{-4}	46	Spearman	no
Droplet	1.11×10^{-4}	1.05×10^{-2}	107	Mahalanobis	yes

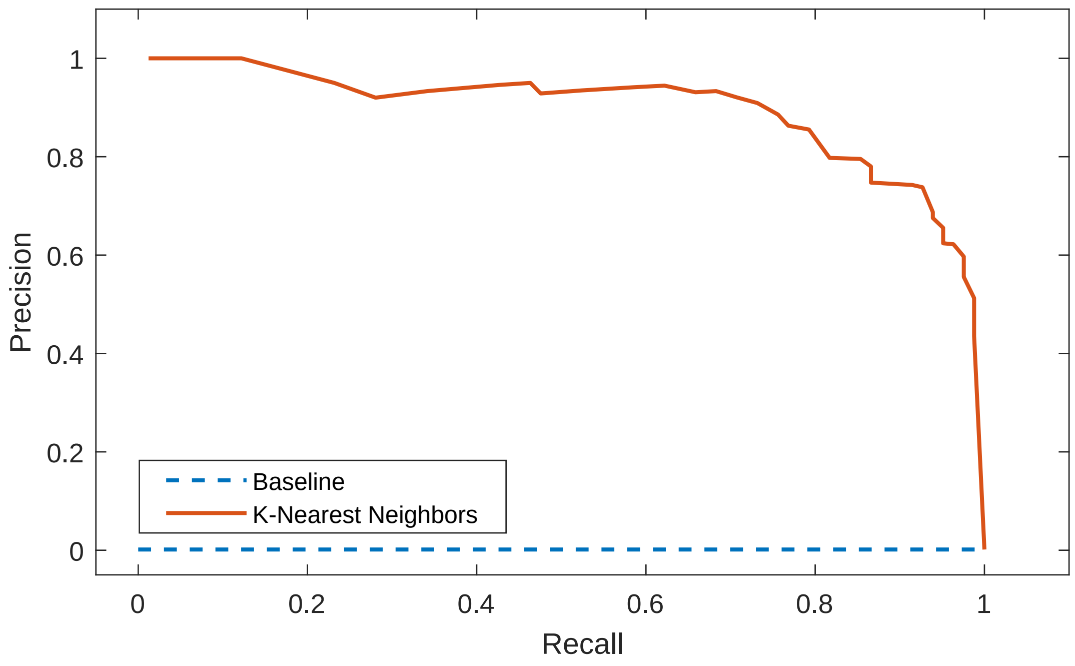


Figure 6.10: PRC curve for the breakage data set using K-nearest neighbors.

An 1-AUC of 1.13×10^{-1} means that the model manages to separate some of the cases, but it also makes some mistakes. Although the classification error is fairly low, the precision over most of the threshold is low which can be seen in Fig. 6.10. This means that a lot of the mistakes that the model makes is classifying non-breakage cases as breakage cases, instead of the other way around. The optimal number of neighbors was decided to be 46. This means that the algorithm will look at the 46 closest neighbors to the current observation to determine where to classify it. Although this can seem like a high number, there are a lot of observations that are closely related in the data set. The Spearman distance is based on ranking the different variables of the current observation instead of using the values of the variables. The closest observations to the current observations are those that have a similar ranking.

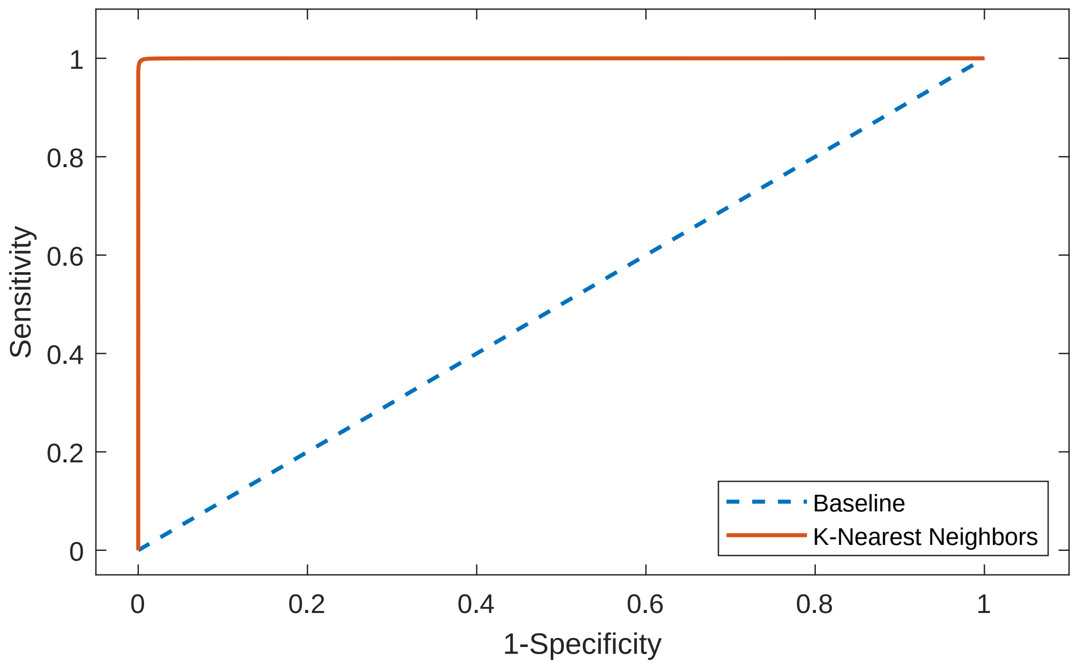


Figure 6.11: ROC curve for the droplet data set using K-nearest neighbors.

For the droplet data set, the model manages to separate the classes well, having an 1-AUC of 1.11×10^{-4}. The ROC curve given in Fig. 6.11 supports this. Although the AUC is high, the classification error is 1.05×10^{-2}, meaning that according to the 5-fold cross-validation, the model does misclassify around 1 out of 100 observations. Since the experiments can be thousands of observations, the number of misclassifications will become significant. The optimal number of neighbors was decided to be 107. The Mahalanobis distance is based on using the covariance matrix of the data set, thus the correlation is taken into account. As seen from the principle components analysis in Section 6.1, some of the variables are most likely highly correlated. The Mahalanobis distance takes this into account and is most likely why this distance succeeds in this case.

All in all, the K-nearest neighbors models manage to separate the classes to some extent for both the droplet and the breakage data set, although not perfectly. This may be due to the curse of dimensionality which states that even though observations may seem close, they can still be far away in the multi-dimensional space. Therefore, the K-nearest neighbor models often works better in cases where there are fewer variables.

6.6 Support Vector Machines

The support vector machines models were made according to Section 5.7 and optimized with the hyperparameters as described in Section 5.7.1. The models were optimized by Bayesian optimization as described in Appendix A.12 with the objective function to maximize the AUC. The optimization gave the following results:

Table 6.7: Results from optimizing a SVM model for both data sets.

Data set	1-AUC	Classification error	C^*	KernelScale	KernelFunction	Standardize
Breakage	4.03×10^{-2}	1.99×10^{-4}	0.28	1.51×10^{-2}	Linear	yes
Droplet	1.61×10^{-5}	2.24×10^{-4}	0.11	5.45×10^{-2}	Linear	yes

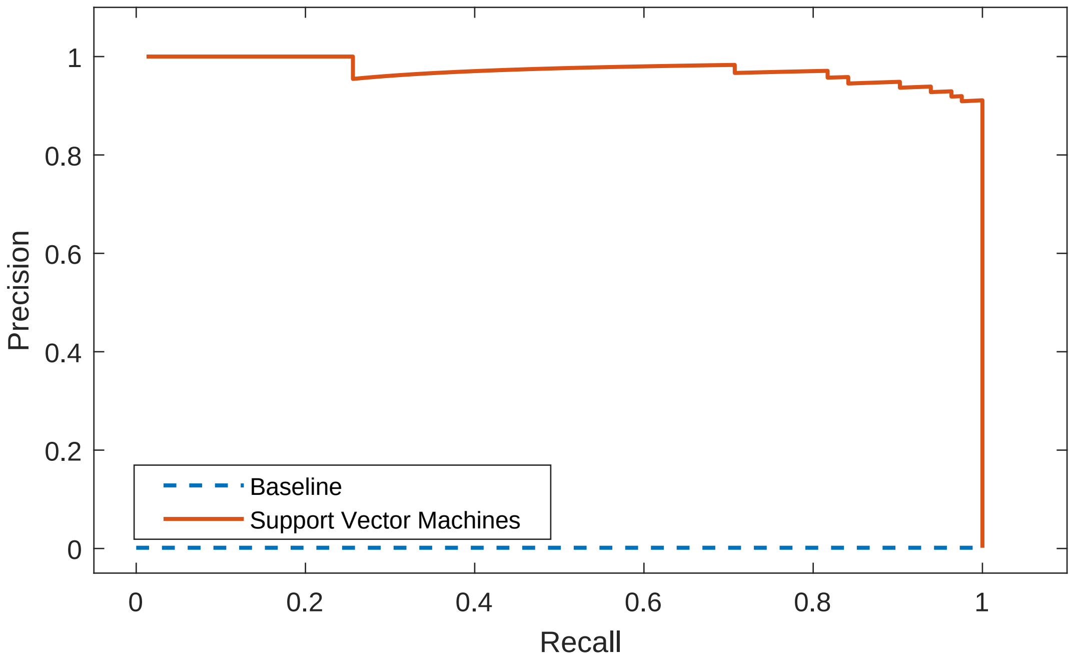


Figure 6.12: PRC curve for the breakage data set using support vector machine.

For the breakage model, an 1-AUC of 4.03×10^{-2} was obtained along with the PRC curve in Fig. 6.12. The precision is close to one for almost the whole range of the recall, which is also reflected by the high AUC. The 5 fold cross-validated classification error is also much lower than the prior probability for the data set, which is given in Table 6.1. This means that the model manages to separate the classes well with high precision for the breakages. In this case, there are 191 support vectors for this model. Like some of the other model type that has been considered in this thesis, a linear decision boundary explains the data set best. Since the kernel is linear, the support vector machine is reduced to the support vector classifier.

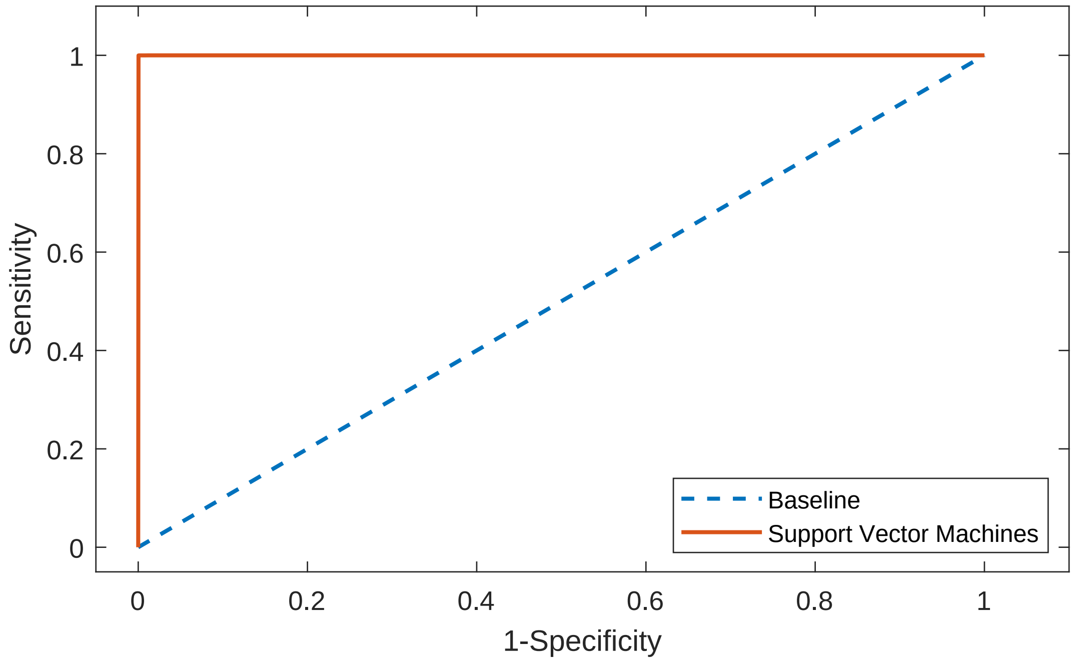


Figure 6.13: ROC curve for the droplet data set using support vector machine.

For the droplet mode, an 1-AUC of 1.61×10^{-5} was obtained. An ROC curve is also plotted in Fig. 6.13. The sensitivity is always very close to 1 for all values for the 1-specificity, which indicates that the model manages to separate the classes very well which is further confirmed by a classification error of 2.10×10^{-4}. The penalty for the slacks for the droplet model was lower than for the breakage model. This is also reflected in a higher number of support vectors, as a lower penalty gives the slack variables the ability to be increased further. The total number of support vectors, in this case, is 397. The best kernel function was also the linear just as for the breakage model. This reduces the support vector machine to the support vector classifier.

All in all, the support vector machine is successful in separating the classes for both the breakage and the droplet data set with both using a linear decision boundary.

6.7 Tree Ensembles

The tree ensembles were made and optimized according to Section 5.8 with the hyperparameters given in Section 5.8.8. Both boosting and bagging are methods that involve many decision trees. Representing them visually will not be done because of the number of decision trees and the size of the decision trees.

Table 6.8: Results from optimizing a tree ensemble model for both data sets.

Data set	1- AUC	Classification error	Number of trees	Learn rate	Minimum leaf size	Random variables	Max number of splits	Split criterion	Model type
Breakage	2.02×10^{-2}	1.88×10^{-4}	369	2.23×10^{-2}	1	12	40840	Deviance	GentleBoost
Droplet	1.90×10^{-6}	2.10×10^{-4}	822	9.93×10^{-1}	2	13	160	Gini	AdaBoost

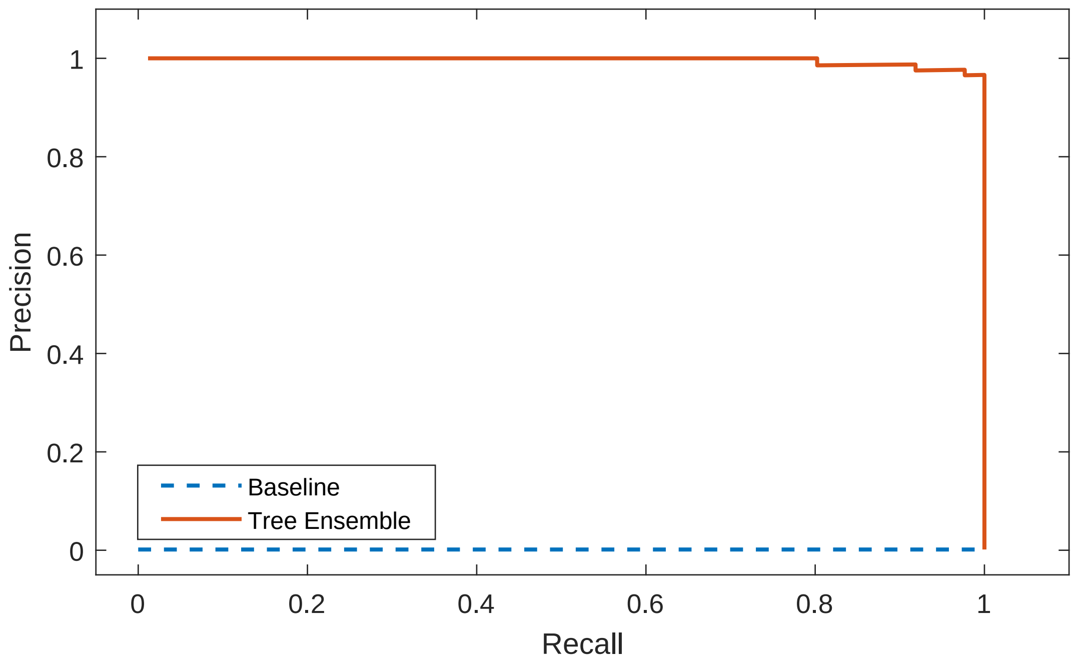


Figure 6.14: PRC curve for the breakage data set using tree ensembles.

For the breakage data set, the classification error is fairly low, much less the prior probability for a breakage. This means that the model successfully manages to separate the breakage cases from non-breakage cases to some extent. The AUC for the breakage case is high, although it is not perfect. The visualization AUC can be seen in Fig. 6.14. Having a low learning rate means that the model will learn much slower which again prevents overfitting the data. The minimum leaf size restricts the number of leaf nodes. Variables chosen at each split was 12, meaning that for each split in each decision tree, the algorithm will choose one variable of 12 random variables that it will split on. 12 random variables will decrease the correlation between the trees as each tree only can choose from 12 of the variables at each split. The type of split criterion for the decision trees was the deviance. As mentioned in Section 5.8, there is no evidence whether the deviance or gini split criterion is the best in general.

The GentleBoost algorithm given in Algorithm 5 was the best performing algorithm for the breakage data set. It outperformed the RUSBoost algorithm given in Algorithm 6 which is an algorithm made for imbalanced data sets, even though previous studies of different data sets claim that RUSBoost at least outperforms the AdaBoost algorithm given in Algorithm 3 when it comes to imbalanced data sets [25]. The reasoning behind this is most likely due to the nature of the variables in the data set. In RUSBoost the algorithm chooses random observations by undersampling. When creating the data set, all observations from each experiment were saved. This means there are a lot of observations, as for instance of the mother droplet where the droplet has not started to break or oscillate yet. These types of observations are a significant part of the data set, and thus many of these observations will be chosen randomly. On the other hand, it is also very important for the model to distinguish between almost breakage and breakage. There are a lot of observations before and after the breakage where the droplets will have almost the same properties. Many of these observations will be lost in the undersampling. All in all, the RUSBoost does not work because there is a correlation between the droplets that the algorithm does capture because of the undersampling. GentleBoost is a more gentle algorithm than the other algorithms as it does not use any kind of logarithm ratios which makes it more stable.

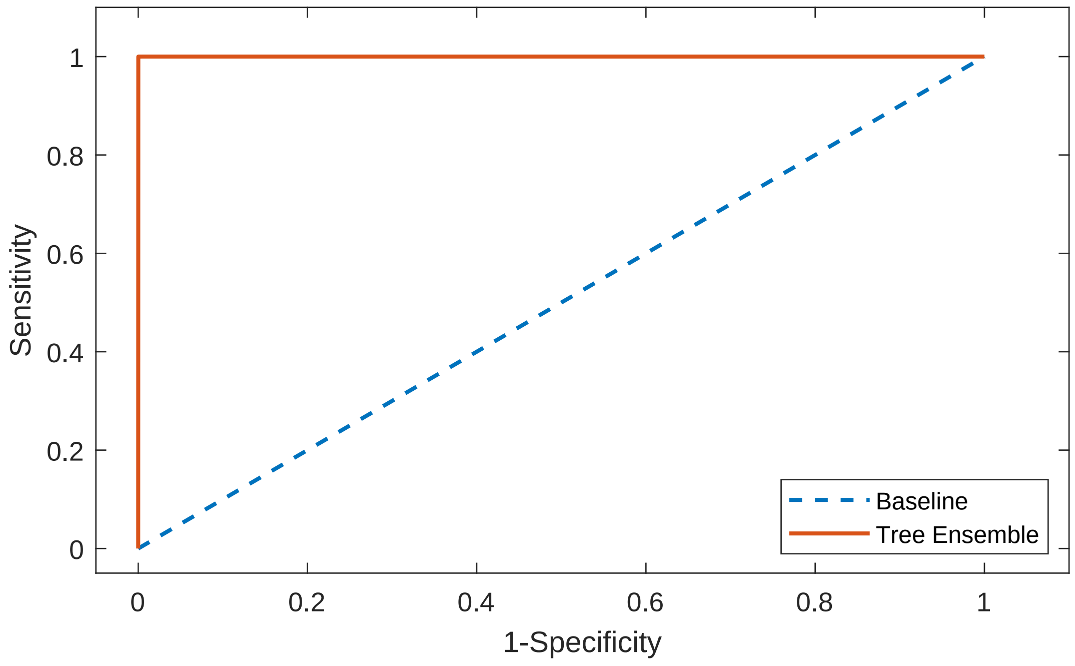


Figure 6.15: ROC curve for the droplet data set using tree ensembles.

For the droplet data set, the AUC is very high and the classification error is very low, meaning that the model successfully manages to separate the two classes. As with the breakage data set, it uses a low learning rate to create a slow learning process using 822 decision trees. For each split for every decision tree, only one of 13 random variables are chosen. The best performing algorithm, in this case, was AdaBoost which differs from the GentleBoost that it does not use Newton steps. As for the breakage data set, the best split criterion for the decision trees was decided to be the Gini in comparison to the deviance for the breakage data set.

The best model for the droplet and the breakage data set performs very well, being able to separate the classes well with a low 1-AUC and classification error.

6.8 Model Selection

For the image analysis software, only one model for checking if it is a breakage or not and one model for checking whether the current drop is the same as a drop in the previous frame is needed. There is thus a need to select the best model for each case.

Table 6.9: Summary of the results for the breakage data set.

Model	1-AUC	Classification error
Old software	9.99×10^{-1}	2.05×10^{-1}
Logistic regression	5.66×10^{-2}	3.04×10^{-4}
Discriminant analysis	8.20×10^{-1}	7.30×10^{-3}
K-nearest neighbors	1.13×10^{-1}	4.74×10^{-4}
Support vector machines	4.03×10^{-2}	1.99×10^{-4}
Tree ensemble	2.02×10^{-2}	1.88×10^{-4}

A summary of the results for each machine learning method for the breakage data set is given in Table 6.9. From comparing the machine learning models to the old model, it can be seen that all of the new models outperform the old image analysis software in both 1-AUC and in classification error. For the breakage data set, it is clear that the tree ensemble is the model that performs best in both 1-AUC and the classification error. This means that not only is the tree ensemble the most precise model, but it also makes the least mistakes in total. This is the reason why the tree ensemble with the GentleBoost boosting algorithm and the hyperparameters given for the breakage model in Section 6.7 is chosen.

Table 6.10: Summary of the results for the droplet data set.

Model	1-AUC	Classification error
Old software	4.79×10^{-4}	9.87×10^{-1}
Logistic regression	1.95×10^{-5}	9.59×10^{-4}
Discriminant analysis	1.11×10^{-4}	1.17×10^{-2}
K-nearest neighbors	1.11×10^{-4}	1.05×10^{-2}
Support vector machines	1.61×10^{-5}	2.24×10^{-4}
Tree ensemble	1.90×10^{-6}	2.10×10^{-4}

A summary of the best performing models for the droplet data set is given in Table 6.10. As for the breakage model, all of the machine learning models outperform the old image analysis software. Although both the 1-AUC and classification error of the old image analysis software are almost as low as for the DA and KNN models. All the machine learning models have a very low 1-AUC and can thus separate the classes well, but the best model is the tree ensemble. It should also be noted here that both the logistic regression and the tree ensemble does very well in separating the classes, but the tree ensemble just outperforms the logistic regression. The tree ensemble model is used with the AdaBoost boosting algorithm and the hyperparameters for the droplet data set given in Section 6.7.

Using the tree ensemble models for both data sets will achieve the highest AUC and classification rate of the tested models. A function that updates the data set was added to the image analysis software to further increase the accuracy and precision of the models. The algorithm for updating one of the data sets and one of the models is given by:

Algorithm 7: Algorithm for updating the machine learning models.

```
The old data set is given by $X$ and $Y$;
The old machine learning model is given by $H$;
for $i = 1 : experiments$ do
    Process experiments and save observations as $X^{(i)}$ and $Y^{(i)}$;
end
Add all new observations to the old data set, making $X^*$ and $Y^*$. Train a new tree ensemble
  model, $H^*$, reoptimizing all hyperparameters;
Find the 5-fold cross-validated AUC on data set $[Y^*, X^*]$ for $H^*$ and $H$, getting $AUC^*$ and
  $AUC$;
if $AUC^* > AUC$ then
    Include $H^*$ as the new model in the tracking logic;
else
    Continue to use $H$ as the model in the tracking logic;
end
```

In Algorithm 7, the model is only updated if the AUC obtained from the new model is better than the AUC obtained from the previous model. The classification error can not be treated as the expected classification error in an experiment and should be used only be used for comparison of different models. The difference in the individual experiments will affect the classification error greatly. Some experiments are very simple where the mother droplet only breaks once, but in some experiments, the mother droplets may break into very many smaller droplets. In cases where there are more droplets that are very similar, the models are suspected to perform worse. In addition to this, it is very easy to misunderstand the meaning of the classification error. Since the implemented machine learning models compare every droplet in this frame with every droplet in the previous frame, the amount of predictions that the models do has a quadratic growth. Due to this, the models are more likely to do a misclassification in experiments with many droplets.

Updating the machine learning models after processing experiments and getting a bigger data set will most likely also increase the predictive capabilities of the models over time. Although there is a theoretical maximum as there is also an inherent error of unknown size.

6.9 Implementation in the image analysis software

Having the two best machine learning models for both the droplet and the breakage data set, the models need to be implemented in the framework of the already existing image analysis software in MATLAB. The old tracking logic was based on checking all the droplets in the current frame with all the droplets in the previous frame and the new tracking logic will also take advantage of this. The tracking process is thus an iterative process where the frames are checked sequentially with the next frame being dependent on the result from the previous iteration.

Further on, the new tracking logic will check if any of the comparisons show a breakage, and classify it accordingly. If is not a breakage the software will check if it is the same droplet, and it is not the same droplet either, it will classify the droplet as a new droplet.

Algorithm 8: Implementation of the machine learning models.

```
The breakage model is defined as $H_1$;
The droplet model is defined as $H_2$;
There are $n$ droplets in the previous frame;
There are $m$ droplets in the current frame;
for $i = 1 : frames$ do
    Predict the response for all observations in frame $i$ with both $H_1$ and $H_2$, getting the
      predicted responses $h_1$ and $h_2$ with dimensions $n$ x $m$;
    Each droplet comparison has a position in $h_1$ and $h_2$, where the first entry for $j = 1 : m$ do
        $[test1, index1] = max(h_1(:, j))$;
        $[test2, index2] = max(h_2(:, j))$;
        if $test1 > parameter1$ then
            Droplet $index1$ in the previous frame and droplet $j$ in the current frame shows a
              breakage;
        else if $test2 > parameter2$ then
            Droplet $index2$ in the previous frame and droplet $j$ in the current frame is the same
              droplet;
        else
            Droplet $j$ in the current frame is a new droplet;
        end
    end
    if $break\_instances == 1$ then
        Check the next highest probability for a breakage of the same droplet in the previous
          frame: $[test3, index3] = max(h_1(index1, h_1 \;\; = J))$, where $J$ is the droplet in the
          current frame marked as a breakage;
        if $test3 > parameter1/2$ then
            Remove any label from droplet $index3$ in the current frame;
            Label droplet $[index1, index3]$ as a breakage;
end
```

Where *parameter*1 and *parameter*2 are probability threshold values for the cut-off and are set to 0.5. In cases where the algorithm only finds one breakage instance in a frame, the last part of the algorithm is evaluated. A droplet has to break into at least two different droplets, so either there is not a breakage, or another droplet also shows a breakage instance. The threshold of the probability is lowered as there is a large probability that, in this case, it is a breakage. The tree ensemble model predicts the breakage has high precision, meaning that it predicts few false positives. All of the boosting models and SVM models give a score with the range of $[-\infty, \infty]$. To get posterior probabilities between 0 and 1, all the scores are scaled with the logit function as given in Eq. (5.11), but with changing the linear regression term, $f(x)$, with the score from the model.

Chapter 7

Case Studies

In this part, a case study on experiments has been conducted by using both the old tracking logic to show its strengths and weaknesses in addition to the new and improved tracking logic. It should be noted that these experiments do not contain data that is used to train the machine learning model. Three different cases will be shown, where the mother droplet breaks into different amounts of daughter droplets to try to capture more of the different outcomes of the experiment. The first case will be with few breakages, the second more breakages, and the third with a lot of breakages. Further on, a breakage instance is defined as the observation where one droplet breaks away from the mother droplet, meaning there are two breakage instances for each breakage. A breakage is defined as a mother droplet splitting into two droplets.

Each case study consists of one experiment, which means that the case study is a sequence of consecutive frames from a single oil droplet going through the breakage column. Experiments where the old tracking logic failed in tracking the droplets were chosen for the case studies.

7.1 Case Study 1

The first case study will be looking into an experiment with few breakages, but also where the old tracking logic does not perform well enough. The goal is to show that the new and improved tracking logic performs well also in less complex cases.

The old tracking logic is based on using some of the variables that are believed to be related to the difference in the droplets. In addition, it also tries to predict where the droplet should be based on the trajectory from the previous frames. Although this approach works quite well, it has some weaknesses. A lot of the available variables were not used, meaning that some variables that are important may be neglected. Based on this, the old tracking logic will in some instances misclassify the observations. The old tracking logic will first look for which drops in the previous frame are the same as in the current frame and look for any breakages. After classifying these observations, it will classify the remaining unclassified droplets as new droplets. To prevent the software from classifying different droplets as the same droplets, the old tracking logic will classify a lot of the observations as new droplets. In addition to this, the old logic will also classify a lot of the droplets as breakages.

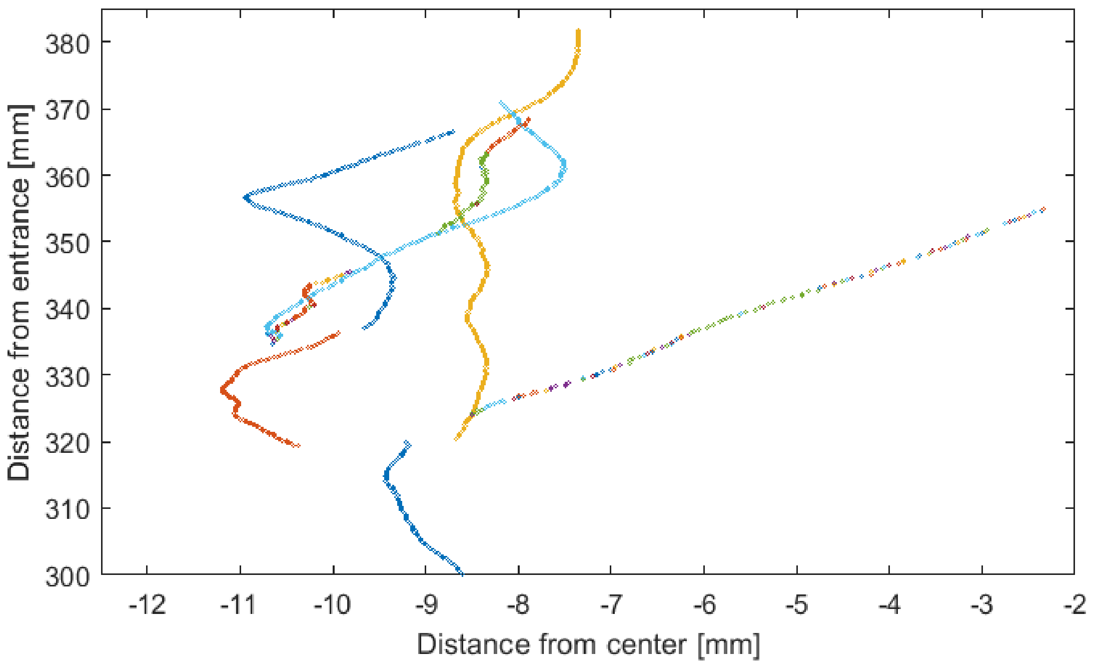


Figure 7.1: Case 1: Tracking with old tracking logic.

In Fig. 7.1, the old tracking logic is used along with the image analysis software to track a droplet for an experiment. Each dot represents a droplet in a given frame, and subsequent dots with the same colour represents the same droplet in different frames. By looking at the droplet to the middle and right at the plot in Fig. 7.1, it is possible to see that there are created new droplets that are most likely the same droplet. This can be seen by looking at the colours changing, but this was also checked by looking at the actual images of the droplet.

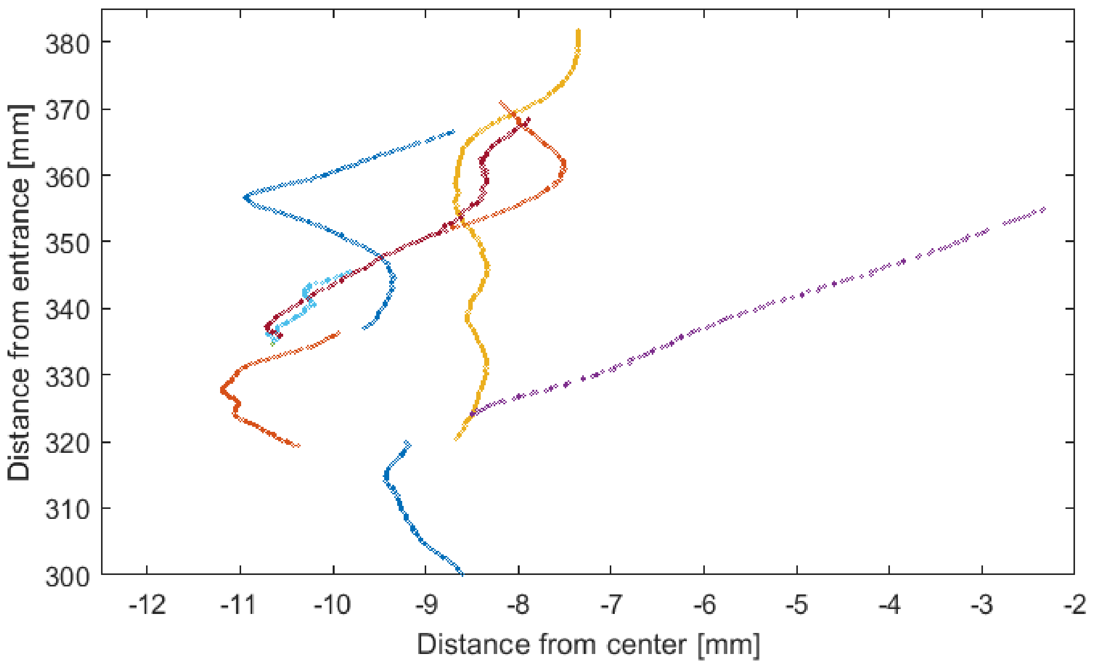


Figure 7.2: Case 1: Tracking with the use of machine learning.

In Fig. 7.2, the new and improved tracking logic with the use of the two machine learning models has been used. The plot in Fig. 7.1 and Fig. 7.2 are very similar as all the placements of the droplets are the same. This is to be expected, as the tracking itself was not altered. Thus it is only the colouring and finding the correct instances of breakage that was altered. From looking at the plot, it is very clear that the new tracking logic has made fewer mistakes. As opposed to the old logic, it did not make too many new droplets, and the droplet to the middle-right is now correct. On the other hand, it did also not make too few new droplets, meaning that in this case, this experiment seems to need almost no processing at all. The only mistake the new tracking logic made was when a droplet that was detected by the software disappeared for a few frames and reappeared. This caused the droplet to change too much, thus the machine learning model did not believe it was the same drop anymore.

Table 7.1: Case 1: Results for the breakage model.

Logic	Breakage precision	True positive rate of breakages
Old	0.05	0.50
New	1.00	1.00

In Table 7.1, a summary of the precision and true positive rate for both the new and old model for finding the breakages is shown for this specific experiment. The old tracking logic predicted 40 breakage instances, when in reality there were 4, meaning that there were a lot of false positives. Also, the old tracking logic missed one of the breakages, making two missed breakage instances and a low true positive rate. On the other hand, the machine learning model managed to capture all the breakage instances, meaning no false negatives, and did not have any false positives, meaning the precision is also 1. As the machine learning model for breakage managed to classify all breakages correctly, meaning no false negatives, the true positive rate is also 1. Compared to the old tracking logic this is a great improvement as the old tracking logic had a very low precision with a high number of false positives.

7.2 Case Study 2

The second case study is a more complex case study than in Section 7.1. In terms of complexity, this type of experiment is not uncommon to occur. The new and improved tracking logic has to perform well on all types of experiments, not only the less complex or the most complex experiments. This case study will show that the new tracking logic by using machine learning models will perform better than the old tracking logic.

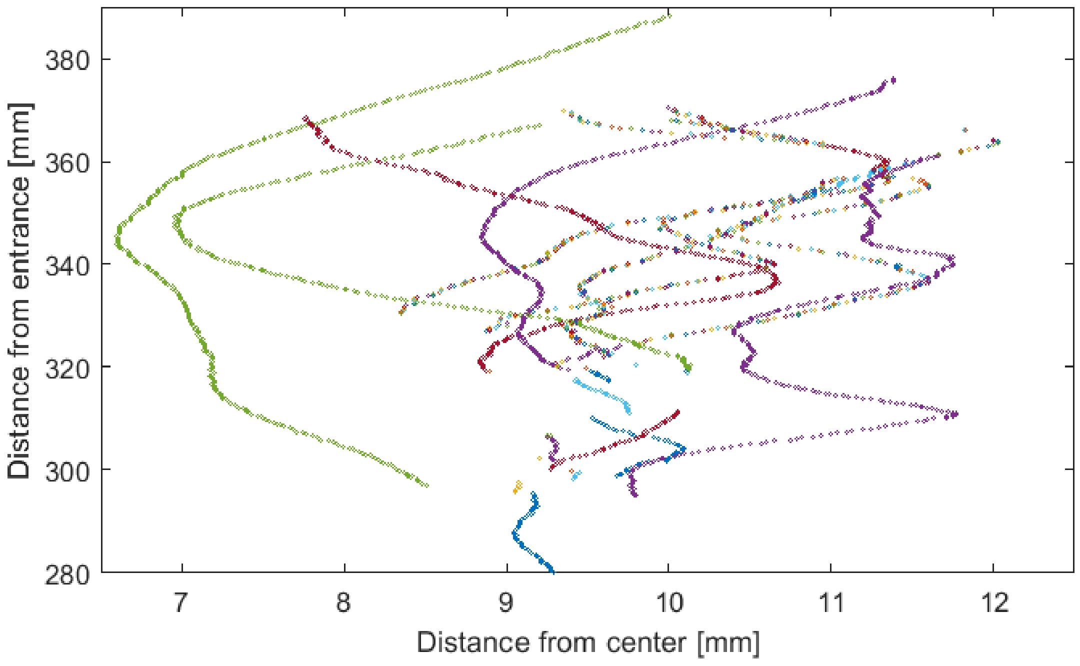


Figure 7.3: Case 2: Tracking with old tracking logic.

The old tracking logic does not do a good job in finding which droplets are the same as can be seen in Fig. 7.3. This resulted in 664 different droplets, which are many more than in the real experiment and this also increases the computational time. As the software compares droplets in the current frame to droplets in the previous frame, having more droplets will increase the computational time drastically. By inspecting the different droplets in the real images, the tracking logic often fails when trying to predict the smaller droplets.

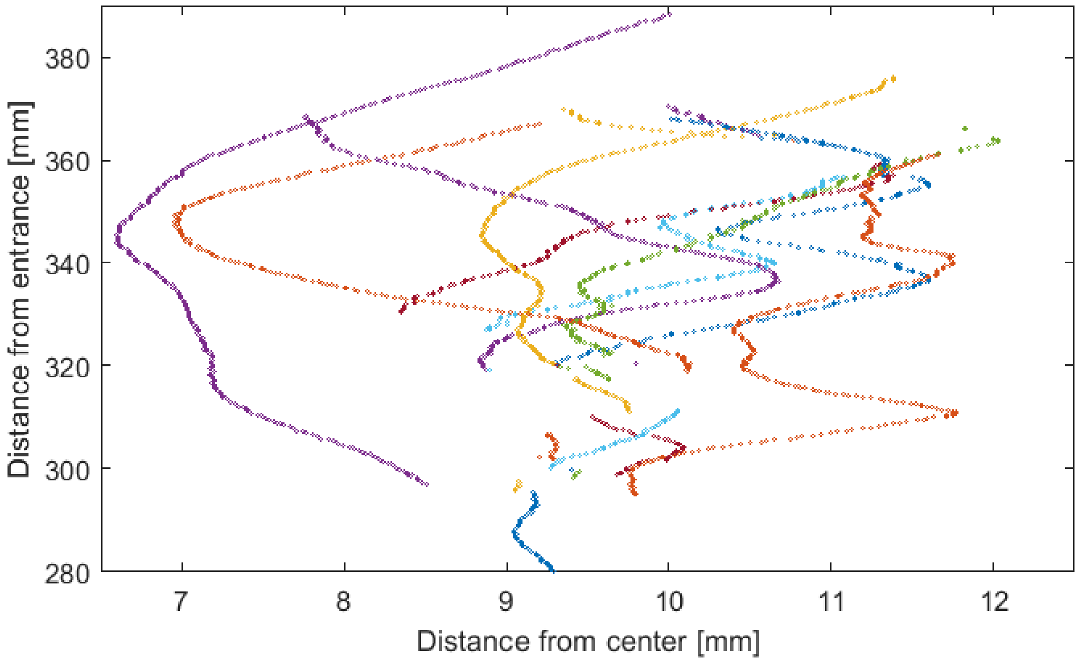


Figure 7.4: Case 2: Tracking with the use of machine learning.

The tracking with the implemented machine learning models is shown in Fig. 7.4. As seen from the plot, almost all droplets that are the same droplet are classified correctly. A total of 27 droplets were classified by the image analysis software. This is much lower than for the old tracking and is much closer to reality. There are still a few droplets that have to be corrected manually, but very few.

Table 7.2: Case 2: Results from tracking with both the new and old tracking logic.

Logic	Breakage precision	True positive rate of breakages
Old	0.18	1.00
New	1.00	1.00

A summary of the breakage precision and true positive rate is given in Table 7.2. In this case, the old tracking logic manages to find all the breakage instances, but it also misclassifies some instances that do not show a breakage. These false positives contribute negatively to the precision, which is why it is low. On the other hand, the tracking logic classifies no false negatives, meaning that the true positive rate is 1. Both the precision and the true positive rate with the new tracking logic as given in Table 7.2 is 1. This means that the new tracking logic predicts all the breakage instances correctly, without any false negatives or false positives.

In this case, the new tracking logic by the use of the machine learning models performs better. Only a few of the droplets that in reality are the same needs to be corrected, but the tracking logic predicted all the breakages correctly.

7.3 Case Study 3

The individual experiments can be very different. Some break into two droplets, some may break into several droplets and some do not break at all. Therefore it is important that the models have been tested on a different case than shown in Section 7.1 and Section 7.2, where the droplet did not break into a lot of smaller droplets. With both the droplet and breakage model, there will be some error. This means, the more droplets there are in an experiment, the more misclassifications there will be. Although a misclassification rate was presented earlier, this was calculated on a large number of experiments with a large variety of different experiments. For more complicated experiments where the mother droplet breaks into several smaller droplets that may eventually break again, the rate of misclassifications may be higher than expected. Droplets that are very different are easier for the models to classify properly, but closely related droplets are more difficult. In the frames before a breakage and after a breakage, the droplets will be very similar, making it harder for the models to classify. The following case is at the extreme end of the number of breakages and complexity of the experiment and would most likely not be processed with the old tracking logic as the extreme number of manual corrections needed. The case has been included to test the limits of the machine learning models.

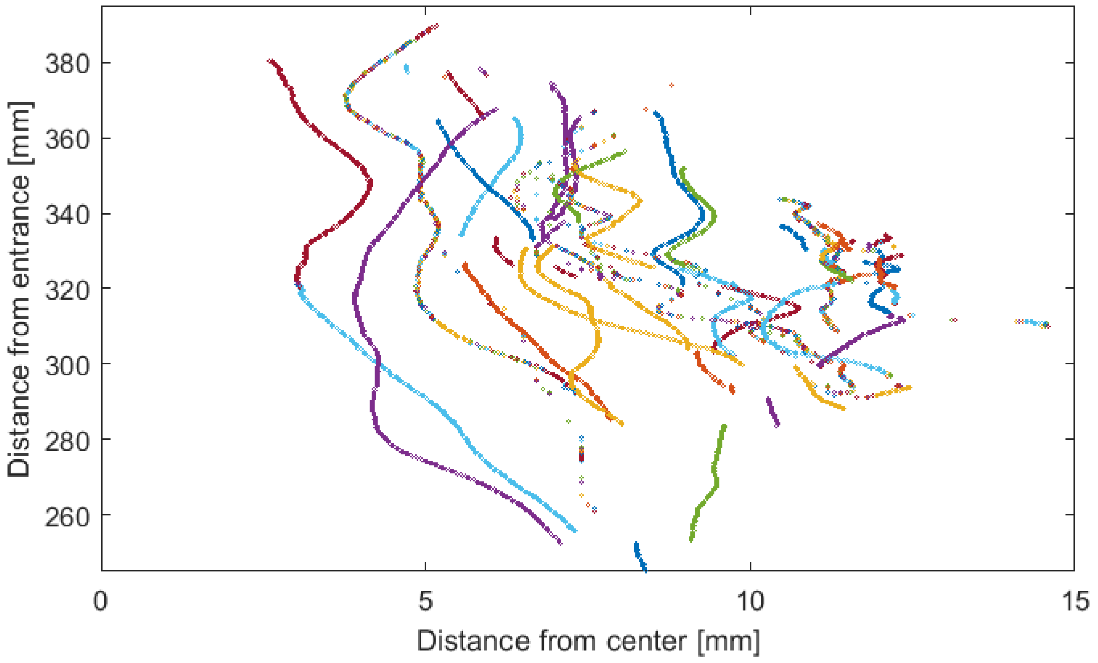


Figure 7.5: Case 3: Tracking with old tracking logic.

Looking at the plot for the tracking with the old tracking logic given in Fig. 7.5, it is clear that the old tracking logic makes a lot of mistakes. Some of the droplets have many colours, even though many of these droplets look like the same droplet. This was also rechecked in the actual images. By inspecting the plot, there are some types of mistakes that the old tracking logic usually does. The first one is that if droplets disappear and reappear, the software struggles to see that it is the same droplet. Secondly, it struggles a lot when there are many small droplets. And lastly, it struggles when there are several droplets very close to each other. When the old logic does not manage to classify a droplet as either the same droplet or a breakage, it will define it as a new droplet. A new droplet does not mean there has appeared a new droplet, but it works as a last resort when the software does not manage to classify the droplet.

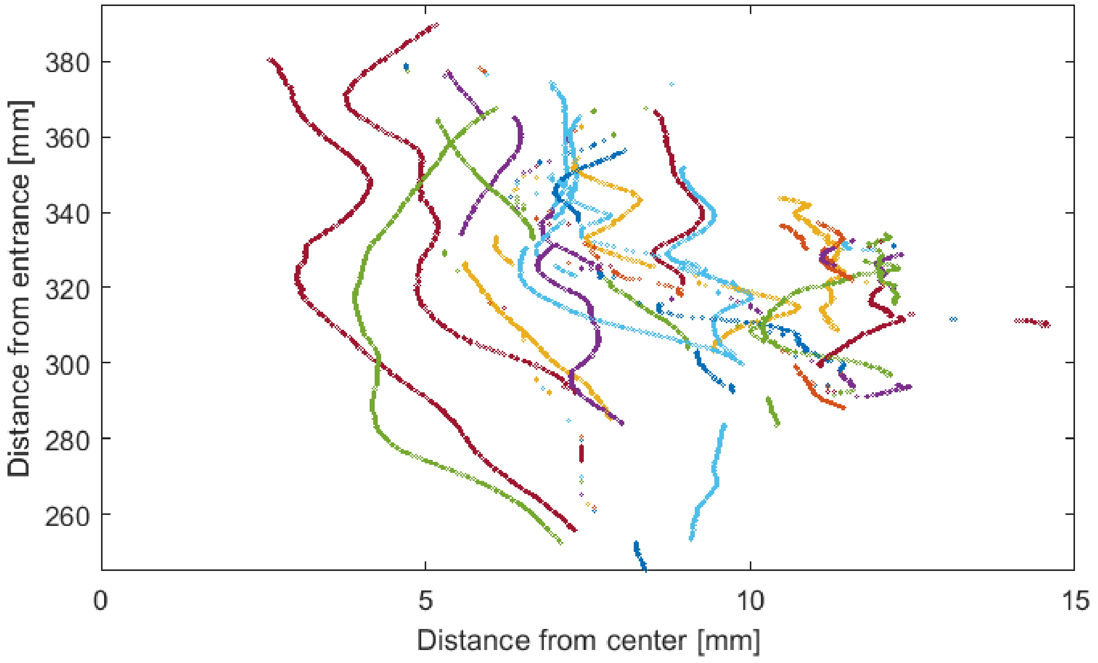


Figure 7.6: Case 3: Tracking with the use of machine learning.

By comparing the plot for the old tracking logic given in Fig. 7.5 and the new tracking logic given in Fig. 7.6, it is not immediately clear that the new tracking logic has done better than the old tracking logic at classifying the which droplets are the same. In a perfect case, there would not be any new droplets other than the ones that come from breakage. The total number of droplets by using the old tracking logic was 1016, while the number of new droplets using the new tracking logic was 131. By inspecting the two plots closely, it is possible to see that the new tracking logic is better at classifying the same droplets. Although in cases where there are several frames in between each time the software finds the droplet, both the old and the new tracking logic often fail in classifying it as the same droplet. Because if there are many frames in between, the droplet itself will have changed too much for the tracking logic to realize it is, in fact, the same droplet as the one, for instance, ten frames ago. Even though the new tracking logic is much better, there is still a need for some manual input to merge some of the droplets, although the needed manual corrections are reduced drastically.

Table 7.3: Case 3: Results from tracking with both the new and old tracking logic.

Logic	Breakage precision	True positive rate of breakages
Old	0.05	0.18
New	1.00	0.70

Looking at the breakage precision and true positive rate which is given in Table 7.3, the old tracking logic classifies too many observations as breakages, meaning the precision is low because of many false positives. The software also failed in capturing the true breakages, meaning that the true positive rate is also very low due to a high number of false negatives. In addition, it struggles more to capture the true breakages, meaning a lot of false negatives and a low true positive rate. On the other hand, the new tracking logic predicts the breakage instances with a precision of 1, meaning that all of the predicted breakages are true positives. Although, not all of the true breakages are captured, meaning that there are some false negatives, leading to a lower true positive rate. Having a limited of observations with breakage to train on may affect the result, and may be improved in the future when more data is available. For both the precision and the true positive rate, the machine learning models perform better.

For this case, the requirement for manual corrections has been decreased, although there is a need for manual corrections. The range of validity of the new tracking logic is higher, as the models have been trained on many different experiments while the old logic was based on a basic tracking principle where changes in position and size were used, thus not being able to cover the whole range of the variability in the experiments. The accuracy of the new tracking logic should also increase over time as more data is processed and can be used in addition to the already existing data set to create more accurate models.

The computational time of the new tracking logic is faster than for the old tracking logic which is mainly due to the higher number of misclassifications of the old tracking logic makes the experiment much more complex than it is. Both the new and the old tracking logic look some frames back in time and if there are many new droplets, it will take a longer time to compare these observations.

Chapter 8

Conclusion and Further Work

In this thesis, two data sets were created. The first one being the breakage data set containing observations of a droplet in the current and the previous frame and labeling whether this is a breakage or not. The second data set, the droplet data set, contains observations of a droplet in the current and previous frame and labeling whether it is the same droplet or not. The following machine learning models were tried: logistic regression, discriminant analysis, tree ensembles, K-nearest neighbors, and support vector machines. The hyperparameters of these models were optimized using Bayesian optimization. The objective function for the tuning of the hyperparameters for breakage data set was to maximize the AUC based on PRC. For the droplet data set, the AUC was maximized using ROC. The best models were for both data sets decided to be tree ensembles with the boosting algorithm GentleBoost for the breakage data set and AdaBoost for the droplet data set.

The models were implemented in the image analysis software, replacing the old image analysis software. The new and improved image analysis software gave a significant lower 1-AUC and a lower classification error than for the old image analysis software. All three case studies show that the new and improved image analysis software performs better than the old image analysis software in detecting the breakages and tracking the droplets. With the machine learning models, the precision for detecting breakages is much higher, meaning that there are fewer false positives. In addition, the software also manages to detect almost all the breakages, meaning few false negatives. The software now also manages to better track which droplets in the current frame are the same as the one in the previous frame. With the two machine learning models, almost all observations are classified as a breakage or the same droplet, meaning very few misclassifications and few new droplets are created. By adding a function to update the models when more data is available the accuracy of the new software is believed to be improved over time.

The machine learning models greatly improve the image analysis software as the machine learning models do few misclassifications. In this context, very little manual input is needed to correct the mistakes of the software. Thanks to the machine learning models, the need for manual corrections, was greatly reduced.

Since the experiments are based on using images, a possible continuation of the project could be to try to use convolutional neural networks to detect the breakages and to track the droplets. Convolutional neural networks are known for being very good when dealing with images, and instead of finding certain properties within an image, such as diameter or area, it uses all of the pixels in the image as variables. As the cameras used for these experiments are monochrome, a pixel is only a certain amount of the colour black. Another area of improvement could also be to try other machine learning models, such as normal neural networks which were not used during this thesis. Exploration of machine learning with imbalanced data sets could also be a topic to further investigate. Very many data sets contain very imbalanced data, most often only a few positives and very many negatives, just as the breakage data set. Most machine learning algorithms minimize some kind of classification error in the construction of the model itself as the objective function, but there may be other objectives that give a more satisfactory outcome. Since machine learning models does a good job in tracking droplets, a similar approach of making new machine learning models for tracking bubbles could also be explored.

Bibliography

[1] Eirik H. Herø, Nicolas La Forgia, Jannike Solsvik, and Hugo A. Jakobsen. Determination of breakage parameters in turbulent fluid-fluid breakage. *Chemical Engineering & Technology*, 42(4):903–909, 2019.

[2] Jannike Solsvik and Hugo A. Jakobsen. On the constitutive equations for fluid particle breakage. *Reviews in Chemical Engineering*, pages 241–356, 2013.

[3] Nicolas La Forgia, Eirik Helno Herø, Jannike Solsvik, and Hugo Atle Jakobsen. Dissipation rate estimation in a rectangular shaped test section with periodic structure at the walls. *Chemical Engineering Science*, 195:159 – 178, 2019.

[4] Christopher K. I Williams. Learning with kernels: Support vector machines, regularization, optimization, and beyond. *Journal of the American Statistical Association*, 98(462):489–489, 2003.

[5] Ensemble machine learning : Methods and applications, 2012.

[6] Trevor Hastie Jerome Friedman, Robert Tibshirani. *The Elements of Statistical Learning*. Springer, Dordrecht, 2009.

[7] Gareth James. *An Introduction to statistical learning : with applications in R*. Springer texts in statistics. Springer, New York, 2013.

[8] David M. J Tax, Herman M. J Sontrop, Marcel J. T Reinders, and Perry D Moerland. The effect of aggregating subtype performances depends strongly on the performance measure used. In *2014 22nd International Conference on Pattern Recognition*, pages 3720–3725. IEEE, 2014.

[9] Charles X. Ling, Jin Huang, and Harry Zhang. Auc: A better measure than accuracy in comparing learning algorithms. *Advances In Artificial Intelligence, Proceedings*, 2671:329–341, 2003.

[10] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets.(report). *PLoS ONE*, 10(3), 2015.

[11] Eugenia San Segundo, Athanasios Tsanas, and Pedro Gómez-Vilda. Euclidean distances as measures of speaker similarity including identical twin pairs: A forensic investigation using

source and filter voice characteristics. *Forensic Science International (Online)*, 270:25–38, 2017.

[12] Visual search ranking, 2014.

[13] Yevhenii Udovychenko, Anton Popov, and Illya Chaikovsky. Binary classification of heart failures using k-nn with various distance metrics. *International Journal of Electronics and Telecommunications*, 61(4):339–344, 2015.

[14] Torleiv Kløve. Lower bounds on the size of spheres of permutations under the chebychev distance. *Designs, Codes and Cryptography*, 59(1):183–191, 2011.

[15] Richard Connor and Robert Moss. A multivariate correlation distance for vector spaces. volume 7404, pages 209–225, 2012.

[16] Jinn-Min Yang. Exploring effectiveness of interval type-2 fuzzy k-nearest neighbor classifier in different distance metric spaces. *International Journal of Intelligent Technologies and Applied Statistics*, 10(4):241–255, 2017.

[17] Dan C Marinescu. Classical and quantum information, 2012.

[18] Xianlong Wang, Yonggang Lu, Dachuan Wang, Li Liu, and Huiyu Zhou. Using jaccard distance measure for unsupervised activity recognition with smartphone accelerometers. volume 10612, pages 74–83. Springer Verlag, 2017.

[19] Yaqin Xie, Yan Wang, Arumugam Nallanathan, and Lina Wang. An improved k-nearest-neighbor indoor localization method based on spearman distance. *IEEE Signal Processing Letters*, 23(3):351–355, 2016.

[20] Laura Raileanu and Kilian Stoffel. Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1):77–93, 2004.

[21] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3(4):319–342, 1989.

[22] Leszek Rutkowski, Maciej Jaworski, Lena Pietruczuk, and Piotr Duda. The cart decision tree for mining data streams. *Information Sciences*, 266(C):1–15, 2014.

[23] Niels Landwehr, Mark Hall, and Eibe Frank. Logistic model trees. *Machine Learning*, 59(1):161–205, 2005.

[24] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals Of Statistics*, 28(2):337–374, 2000.

[25] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: Improving classification performance when training data is skewed. 2008.

[26] Inc. The MathWorks. *Statistics and Machine Learning Toolbox*. Natick, Massachusetts, United State, 2020.

[27] Kenneth Lange. *Optimization*, volume 95 of *Springer Texts in Statistics*. Springer New York, New York, NY, 2013.

[28] Jorge Nocedal. Numerical optimization, 2006.

[29] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on machine learning*, volume 69 of *ICML '04*, page 116. ACM, 2004.

[30] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. volume 4, pages 2951–2959, 2012.

[31] Adam D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12:2879–2904, 2011.

Appendix A

A.1 Standardization of data

To improve some of the machine learning algorithms, it is possible to standardize the data before making a machine learning model. Scaling can be important if the scales of the data are different. Even though the values of the variables may be very different, it is not clear that the one with a higher value is more or less important than the other variables. For instance, the KNN algorithm compares distances between the observations. Thus, the variables that has a high value will have a greater impact than other variables that may be on a lower scale, but this may in reality not be true. To prevent this effect, it is possible to standardize all of the observations before making a machine learning model. Standardizing the data is done by making all the variables have a mean of zero. In addition the standard deviation is set to one [7]. All of the observations can be scaled in the following way for all of the variables j:

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}} \tag{A.1}$$

Where $\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}$ is an estimation of the standard deviation [7].

A.2 Derivation of the LDA Classifier

Substituting Eq. (5.15) into Eq. (5.16) gives:

$$p_k(x) = \frac{\pi_k}{\sum_{l=1}^{K}\pi_l f_l(x)} \cdot \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right) \tag{A.2}$$

Merging $\frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}} \cdot \Sigma_{l=1}^{K}\pi_l f_l(x)}$ into the discriminant score δ_k since these terms are equal for all classes. This gives the following discriminant score:

$$\delta_k(x) = \pi_k exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right) \tag{A.3}$$

Taking the logarithm on both sides and realizing that $x^T\Sigma^{-1}\mu = \mu^T\Sigma^{-1}x$ since Σ^{-1} is a symmetric matrix. Merging the logarithm and $x^T\Sigma^{-1}x$ into $\delta_k(x)$ gives:

$$\delta_k(x) = ln(\pi_k) - \frac{1}{2}\mu^T\Sigma^{-1}\mu + x^T\Sigma^{-1}\mu_k \tag{A.4}$$

A.3 Mean Squared Error

The mean squared error (MSE) is often used as a measurement of performance for regression methods and is given by:

$$\min \sum_{x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{A.5}$$

A.4 Deivation of the QDA Classifier

Starting at Eq. (A.2), but this time keeping in mind that Σ_k is not constant for each class, $\frac{1}{(2\pi)^{\frac{p}{2}} \cdot \sum_{l=1}^{K} \pi_l f_l(x)}$ can be merged into the discriminant score. The resulting equation is thus:

$$\delta_k(x) = \frac{\pi_k}{|\Sigma_k|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right) \tag{A.6}$$

After taking the logarithm, this is equal to:

$$ln(\delta_k(x)) = ln(\pi_k) - \frac{1}{2}ln(|\Sigma_k|) - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) \tag{A.7}$$

Again, merging all terms that are not dependent on the class, k, into the discriminant score gives:

$$\delta_k(x) = -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k + ln(\pi_k) - \frac{1}{2}ln(|\Sigma_k|) \tag{A.8}$$

A.5 Calculation of the average variance

The expected value, $E(x)$, can be estimated by:

$$E(x) = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{A.9}$$

All of the observations in the data set are assumed to have the same variance, namely: $var(x) = \sigma^2$. Thus, the variance of the mean, $\hat{x}$, becomes:

$$var(\bar{x}) = var(\frac{1}{n}\sum_{i=1}^{n} x_i) \tag{A.10}$$

$$= \frac{1}{n^2} \cdot \sum_{i=1}^{n} var(x_i) \tag{A.11}$$

$$= \frac{\sigma^2}{n} \tag{A.12}$$

A.6 Variance captured by PCA

Table A.1: Percentage of variance captured in each principle component.

Principle component	Variance captured [%]
PCA 1	99.0324%
PCA 2	0.3662%
PCA 3	0.3150%
PCA 4	0.1517%
PCA 5	0.0677%
PCA 6	0.0574%
PCA 7	0.0052%
PCA 8	0.0035%
PCA 9	0.0005%
PCA 10	0.0002%
PCA 11	0.0001%
PCA 12-25	0.0000%

A.7 Singular Value Decomposition (SVD)

A matrix X with the dimensions n x p with $n > p$ can be written as its singular value decomposition [27]:

$$X = USV^T \tag{A.13}$$

Where U is orthogonal and has the dimension n x n and V^T also othogonal and has the dimension n x p [28]. The matrix S has the dimensions p x p and is a diagonal matrix where the entries are called singular values in descending order, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_p \geq 0$.

Since both U and V are orthogonal matrices, it means that $U^TU = I$ and $V^TV = 0$. In addition to this, since S is diagonal, S^T = S. Thus the following extension to Eq. (A.13) can be done:

$$X^T X = (XSV^T)^T)(XSV^T) = VSX^T XSV^T = VSSV^T \tag{A.14}$$

Where S will contain the eigenvalues of $X^T X$ in descending order while V will contain the corresponding eigenvectors. Thus, these can be found by $det(X^T X - \lambda I) = 0$. Having both Σ and V, U can be found by using the fact that orthogonal matrices has the following property: $V^T V = I$. Putting this into Eq. (A.13), gives:

$$U = XVS^{-1} \tag{A.15}$$

In Eq. (A.13), each column of US will represent the loadings, ϕ_i, for one principle component, Z_i, while V describes the principle directions. The loadings of the first principle component, Z_1 will thus be the first column of US.

More information on the SVD algorithm can be found in [27] and [28].

A.8 Pseudo-Inverse

The Moore-Penrose pseudo-inverse is denoted A^+.

$$AA^+A = A \tag{A.16a}$$
$$A^+AA^+ = A^+ \tag{A.16b}$$
$$(AA^+)^T = AA^+ \tag{A.16c}$$
$$(A^+A)^T = A^+A \tag{A.16d}$$

All equations in Eq. (A.16a) must be satisfied. By using SVD as explained in Appendix A.7, the matrix A can be defined as $A = USV^T$. Putting this into Eq. (A.16a) and using the properties of the SVD that $S^T S = I$, $U^T U = I$ and that S is invertible, gives:

$$(USV^T)A^+USV^T = USV^T \tag{A.17a}$$
$$(USV^T)^T(USV^T)A^+USV^T = (USV^T)^T(USV^T) \tag{A.17b}$$
$$VSU^T USV^T A^+USV^T = VSU^T USV^T \tag{A.17c}$$
$$VSSV^T A^+USV^T = VSSV^T \tag{A.17d}$$
$$A^+USV^T = V(SS)^{-1}SSV^T \tag{A.17e}$$
$$A^+ = VIV^T VS^{-1}U^T \tag{A.17f}$$
$$A^+ = VS^{-1}U^T \tag{A.17g}$$

A.9 Coefficient Estimates for Logistic Regression

Table A.2: Coefficient estimates for logistic regression using the both data sets.

Coefficient	Breakage data set	Droplet data set
β_0	-76.9440	-17.3289
β_1	-118.3567	-6.0463
β_2	-16.1610	-21.4601
β_3	-38.0150	-33.9808
β_4	-3.9725	0.3859
β_5	2.2358	1.6217
β_6	-3.0813	-1.0297
β_7	-8.5429	-0.5074
β_8	-1.2037	-1.2164
β_9	3.3062	-0.1387
β_{10}	-0.1134	-0.0841
β_{11}	-2.7271	3.7421
β_{12}	8.0113	-2.0517
β_{13}	0.2182	-0.0411
β_{14}	-4.2742	0.1594
β_{15}	-1.2777	0.7046
β_{16}	1.0327	-0.5562
β_{17}	3.0465	-2.2013
β_{18}	-3.8928	1.9377
β_{19}	-0.1489	0.0660
β_{20}	3.4480	1.2237
β_{21}	9.0318	0.8689
β_{22}	5.0506	0.7195
β_{23}	4.3325	0.4037
β_{24}	-4.1278	2.1926

A.10 PRC and ROC for the Old Image Analysis Software

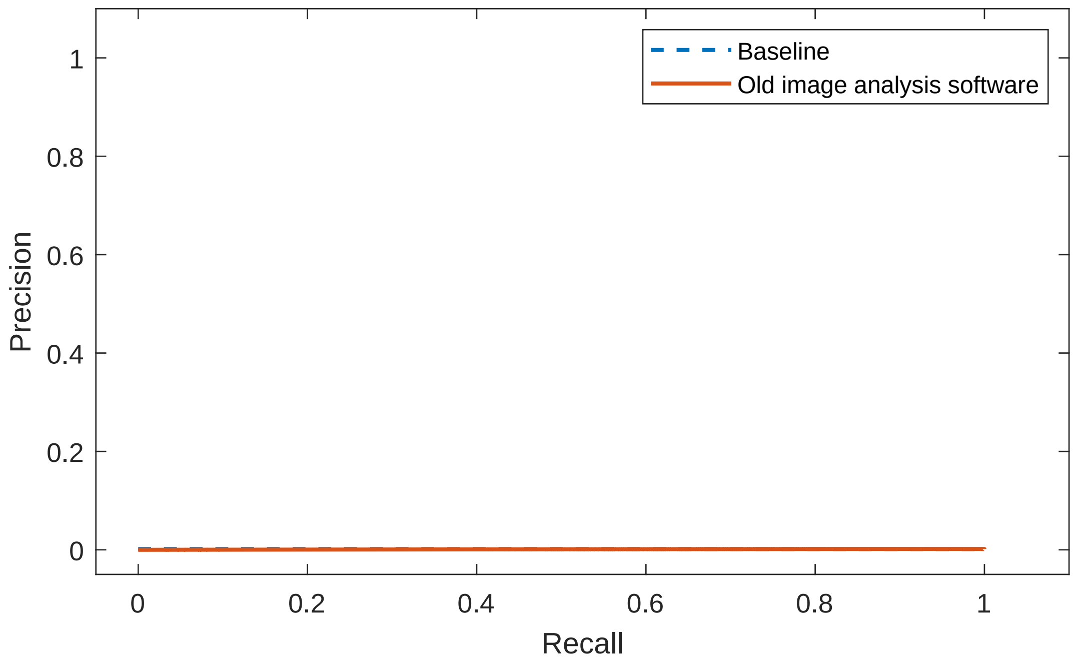


Figure A.1: PRC curve for the old breakage model.

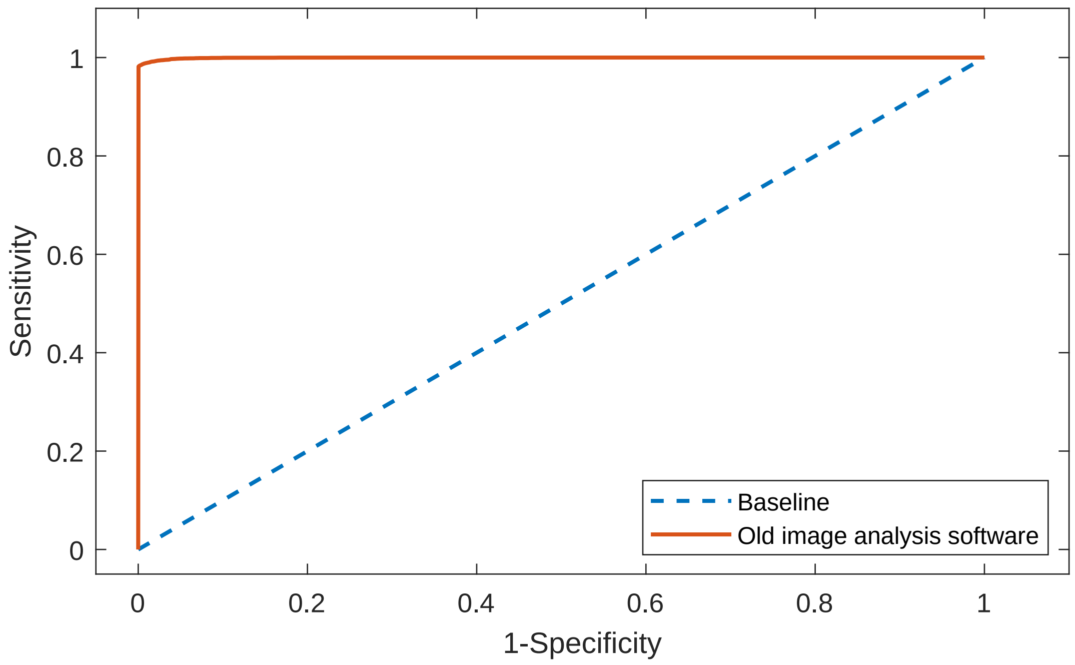


Figure A.2: ROC curve for the old droplet model.

A.11 Stochastic Gradient Descent

The stochastic gradient descent (SGD) is an optimization method that utilizes the gradient of the objective function to move towards a maximum or minimum. The loss function used for logistic regression is given by [29]:

$$\phi(\beta, y) = ln(1 + exp(-\beta y)) \tag{A.18}$$

The SGD algorithm is given by:

Algorithm 9: Stochastic gradient descent algorithm [29].

```
The variable matrix of the observations are denoted $X$, while the responses are denoted $Y$. Set
  an initial guess for $\hat{\beta}_0$+; for $m = 1, 2, ...$ do
    Randomly pick one observation $X_m$ with the corresponding response $Y_m$;
    Updating the coefficients: $\hat{\beta}_m = \hat{\beta}_{m-1} - \eta_t(\lambda \hat{\beta}_{m-1} + \frac{\partial \phi(\beta_{m-1}, Y)}{\partial \beta_{m-1}}$;
end
```

Where η_m is the learn rate and λ is the amount of regularization. The condition for convergence is:

$$\left\| \frac{\beta_m - \beta_{m-1}}{\beta_m} \right\| \leq \gamma \tag{A.19}$$

Where γ is a set tolerence.

A.12 Bayesian Optimization

Although it is out of the scope of this thesis to explore the field of optimization, the optimization algorithm used to optimize the hyperparameters will be discussed briefly. All of the methods discussed earlier have some hyperparameters that need to be chosen. This can be done in many ways, but as the number of hyperparameters increases, the search space increases exponentially. Using optimization can be done, although it is time-consuming. By using Bayesian optimization, it is possible to optimize the hyperparameters to reduce the classification error, although it should be noted that there is no guarantee to find a global minimum. Bayesian optimization is a bounded method, meaning that there need to be bounds on the constraints, which in this case are the hyperparameters. It is an optimization tool for solving non-convex optimization problems.

The objective function of the optimization is the AUC based on PRC for the breakage data set and AUC based on ROC for the droplet data set Although it is very computationally expensive to evaluate this objective function as the machine learning model would have to be trained at every iteration of the optimization. In general, Bayesian optimization assumes that the objective function is sampled from a Gaussian process [30]. It creates a probability model based on the Gaussian process, namely a Gaussian process regression, to find the next point of evaluation of the objective function. This is different from other optimization methods, as many of them require an evaluation of the objective function in addition to the Hessian at every iteration to find the next point of evaluation.

The Gaussian process assumes that any set of n observations belongs to the multivariate Gaussian (normal) distribution in which the density function is given by [7]:

$$f(x) = \frac{1}{(2\pi)^{\frac{1}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{A.20}$$

Where μ is the mean and Σ is the covariance matrix.

The acquisition function is the function the determines the next point of evaluation. There are many popular choices, but in this case, expected improvement per second plus has been used. The improvement is given by:

$$\gamma(x) = \frac{f(x_{best}) - \mu(x;\{x_n, y_m\}, \theta)}{\sigma(x;\{x_n, y_m\}, \theta)} \tag{A.21}$$

Where $\mu(x;\{x_n, y_m\}, \theta)$ is given as the predicted mean and $\sigma(x;\{x_n, y_m\}, \theta)$ is given as the predicted variance [30].

The expected improvement is given by:

$$a_{EI}(x;\{x_n, y_m\}, \phi) = \sigma(x;\{x_n, y_m\}, \phi)(\gamma(x)\Phi\gamma(x) + \mathcal{N}(\gamma(x);0,1)) \tag{A.22}$$

Where Φ is the cumulative distribution function for the multivariate normal distribution given in Eq. (A.20).

For the Gaussian process regression, there is a need to calculate the covariance matrix, and for this, the ARD Matérn 5/2-kernel is used [30]:

$$K(x,x') = \theta_0\left(1 + \sqrt{5r^2(x,x')} + \frac{5}{3}r^2(x,x')\right)\exp\left(-\sqrt{5r^2(x,x')}\right) \tag{A.23}$$

With:

$$r^2(x,x') = \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\theta_d^2} \tag{A.24}$$

For further reading, the interested reader is referred to [30, 31].