TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Software Science

Henry Laur, 204155 IAPM

AUTOMATED SEGMENTATION AND SEMANTIC ANALYSIS OF WRITING AND DRAWING TESTS FOR PARKINSON'S DISEASE DIAGNOSTICS

Master's thesis

Supervisors: Sven Nõmm, PhD
Elli Valla, MSc
Aaro Toomela, PhD

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Henry Laur, 204155 IAPM

JOONISTUSTESTIDE AUTOMATISEERITUD SEGMENTATSIOON JA SEMANTILINE ANALÜÜS PARKINSONI TÕVE DIAGNOOSIMISEKS

Magistritöö

Juhendajad: Sven Nõmm, PhD

Elli Valla, MSc

Aaro Toomela, PhD

Tallinn 2022

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Henry Laur

Date: 10.05.2022

Automated Segmentation and Semantic Analysis of Writing and Drawing Tests for Parkinson's Disease Diagnostics

Abstract

This thesis will analyze digitized drawing tests used to diagnose Parkinson's disease. It will examine Luria's alternating series drawing test with a focus on segmentation and corner detection. The goal of this thesis is to determine whether different segments of the Pi Lambda drawing test have different predictive powers. There are two main reasons for this. Firstly, currently physician still largely rely on analyzing the drawing tests by eye. However, using more advanced methods to examine the drawing test can give more information to the physician than solely relying on their eyes. The second reason is academic. Finding out which segments contain the best predictive power can help with further research into this area.

In order to automatically segment the drawing tests, the deep learning model YOLO is used in a novel way to automatically detect corners in the Pi Lambda test. In this thesis, the DraWritePD data set is used, which contains drawing tests for control and PD patients.

Full segmentation based classification reveals that the best segment had an accuracy of 96.7%, a precision of 95%, a sensitivity of a 100% and a specificity of 93%. The segment mentioned was the vertical line at the end of the test on the continuation task. This is a state-of-the-art result that beats the accuracy of a previous model by 5%. Reviewing the results for the other segments shows that different segments do have different predictive powers.

In terms of segmentation YOLO achieved an recall of 89% in detecting corners and a precision of 97%. After fully segmenting the PL tests, 70% were segmented correctly, 20% had a slight mistake and 10% had a major mistake. With transition segmentation, 85% were segmented correctly, 10% had a slight mistake and 5% had a major mistake.

Other classification methods besides full segmentation based classification were used. Picture based corner classification, transition segmentation based classification and

micrography based classification. However, they did not offer as good of a result as full segmentation based classification.

Present thesis is written in English and is 62 pages long, including 8 chapters, 19 tables and 24 figures.

Joonistustestide automatiseeritud segmentatsioon ja semantiline analüüs parkinsoni tõve diagnoosimiseks

Annotatsioon

Käesolevas töös analüüsitakse digitaliseeritud joonistusteste, mida kasutatakse Parkinsoni tõve diagnoosilisel. Selleks vaadatakse Luria vahelduva seeria joonistusteste. Fookus tuuakse nende segmenteerimisele ja nurkade leidmisele. Töö eesmärk on leida, kas erinevatel Pi Lambda joonistustesti segmentidel on erinev eraldusvõime. Selles on kaks peamist põhjust. Esiteks praegu arstid analüüsivad joonistusteste peamiselt enda silmadega. Kuid, kui kasutada täpsemaid meetodeid, et uurida joonistusteste, on võimalik saada neist palju rohkem infot, mis võib abistada arsti diagnoosimisel. Teine põhjus on akadeemiline. Kui leida, millised segmendid omavad suuremat eraldusvõimet võimaldaks see kiiremat edasiarengut antud valdkonnas.
Automatiseeritud segmenteerimiseks on kasutatud süvaõppe mudelit YOLO uuel viisil, et automaatselt leida Pi Lambda joonistustesti nurkasid. Töös on kasutusel DraWritePD andmekogum, kus on olemas joonistustestid tervetelt patsientidelt ja Parkinsonitõve patisentidelt.
Täis segmenteerimisel põhinev klassifitseerimisel tuli välja, et kõige parema segmendi täpsus on 96.7%, kordustäpsus on 95%, tundlikus on 100% ja spetsiifilisus on 93%. See segment oli vertikaalselt sirge joon PL joonistustesti lõpus. Antud tulemus parandab eelnevat 5% võrra. Kui vaadata teiste segmentide tulemusi on selge, et erinevatel segmentidel on erinev eraldusvõime.
Segmentatsiooni juures suutis YOLO saavutada 89% saagise ja 97% kordustäpsuse. Pärast täis Pi Lambda joonistustesti segmenteerimist oli 70% neid õigesti segmenteeritud, 20% neist olid väikese veaga segmenteeritud ja 10% oli suure veaga segmenteeritud. Ülemineku segmentatsioonil oli 85% segmentidest õigesti määratud, 10% oli väikese veaga ja 5% oli suure veaga.
Prooviti teisi klassifitseerimise meetodeid peale täis segmenteerimise. Pildi põhine nurkade klassifitseerimine, ülemineku alusel klassifitseerimine ja micrograafia alusel klassifitseerimine, kuid ükski neist ei andnud nii häid tulemusi kui täis segmentat-

siooni põhjal klassifitseerimine.

Töö on kirjutatud Inglise keeles ning sisaldab teksti 62 leheküljel, 8 peatükki, 19 tabelit ja 24 diagrammi.

Contents

List of Figures **3**

List of Tables **4**

1 Introduction **5**

1.1 Motivation 8

2 Literature **9**

2.1 Object detection 9

2.2 Drawing test analysis 12

3 Background **13**

3.1 Luria's Alternating series tests 13

3.2 Data 16

3.3 Segmentation 17

4 Problem statement **19**

5 Methods **21**

5.1 YOLO corner detection 23

5.2 Straight line segmentation 25

5.3 Segment types 26

5.4 Segment position 28

5.5 Indicators 29

6 Results **30**

6.1 YOLO corner detection 30

6.1.1 Training YOLO 30

6.1.2 Validation 33
6.1.3 Picture based corner classification 35
6.1.4 Full segmentation 38
6.1.5 Full segmentation based classification 39
6.1.6 Transition segmentation 46
6.1.7 Transition segmentation based classification 47
6.2 Indicators 51
6.2.1 Micrography based classification 51

7 Discussion **54**

8 Conclusion **56**

Bibliography **59**

A Non-exclusive licence **63**

B Results of the fully segmented PL trace task **64**

C Results of the fully segmented PL copy task **73**

D Results of the fully segmented PL continue task **82**

E Results of the transition corners segmented PL trace task **91**

F Results of the transition corners segmented PL copy task **96**

G Results of the transition corners segmented PL continue task **101**

H Results of the micrography angles PL trace task **106**

I Results of the micrography angles PL copy task **108**

J Results of the micrography angles PL continue task **110**

List of Figures

3.1 PL Test 13
3.2 Segmented PL Test 17

5.1 Workflow of the methods 22
5.2 PL Test With Highlighted Corner 23
5.3 Usage of YOLO 24
5.4 Only corners image 24
5.5 PL Test With Highlighted Corner 25
5.6 PL Test Positions 28
5.7 Micrography example 29

6.1 mAP of yolo with 300 epochs 31
6.2 Recall and precision of yolo with 300 epochs 31
6.3 Box loss of yolo with 300 epochs 32
6.4 Class loss of yolo with 300 epochs 32
6.5 Object loss of yolo with 300 epochs 33
6.6 Example of low confidence threshold 33
6.7 Example of high confidence threshold 34
6.8 Example of good confidence threshold 34
6.9 PL Trace Heatmap 43
6.10 PL Copy Heatmap 45
6.11 PL Continue Heatmap 46
6.12 Example of transition corners 47
6.13 PL Trace Corners Heatmap 48
6.14 PL Copy Corners Heatmap 49
6.15 PL Continue Corners Heatmap 50

List of Tables

6.1 Confidence threshold with recall and precision 35
6.2 New classification layers 36
6.3 PL Trace Picture Corner Classification 37
6.4 PL Copy Picture Corner Classification 37
6.5 PL Continue Picture Corner Classification 38
6.6 Fully segmented PL test results 39
6.7 Sample subset of vector features.[31] 41
6.8 Sample subset of single-value features.[31] 42
6.9 PL Trace Kinematic Classification 43
6.10 Fully segmented PL Copy Kinematic Classification 44
6.11 Fully segmented PL Continue Kinematic Classification 45
6.12 Transition corners PL test results 47
6.13 Corner transition PL Trace Kinematic Classification 48
6.14 Corner transition PL Copy Kinematic Classification 49
6.15 Corner transition PL Continue Kinematic Classification 50
6.16 Micrography angle standard deviation 51
6.17 Corner transition PL Trace Micrography Angles 52
6.18 PL Copy Micrography Angles 52
6.19 PL Continue Micrography Angles 53

Chapter 1

Introduction

This thesis will analyze Luria's alternating series drawing test with a focus on segmentation and corner detection. These tests are used to given an indication of whether a subject has Parkinson's disease(PD). More specifically the Pi Lambda(PL) variation of the drawing test will be examined. However, the workflow developed in this thesis could also be applied to various other drawing tests.

Physicians still largely rely on examining the drawing tests by eye. However, when analyzing digitized drawing tests, it is possible to give additional information to the physicians that would be invisible to the naked eye.

Analyzing segments separately will give a good idea which segments are the most important when examining the PL drawing test. This can have significant impact because when analyzing solely based on segments that have the most predictive power it will be possible to get better results than analyzing the whole test. Furthermore, it would be possible to conduct further research into why exactly certain segment have better predictive power.

Research shows [7, 11] that the prevalence of Parkinson's disease increases with age and people older than 65 have almost a 2% chance of having Parkinson's. Even diagnosing Parkinson's is hard for physicians. When autopsies were performed to confirm the diagnosis in patients up to 15% of the patients diagnosed with PD did not meet the clinical or pathological criteria for the disease. The misdiagnosis of Parkinson's disease ranges from 10% to 20%.

Parkinson's disease is clinically characterized by an asymmetric onset of bradykinesia, rigidity, and resting tremors. The main cause of it is the death of dopaminergic

neurons in the midbrain. Usually by the time Parkinson's is diagnosed in patients about 70-80% of the dopaminergic neurons have died. The earlier treatment is provided to save the at risk neurons, the better.[25]

One of the ways of diagnosing long-term neurodegenerative diseases like Parkinson's is drawing and writing tests. Research shows [19, 18] that Luria's drawing test has the ability to distinguish PD patients from controls.

In this thesis, the main problem to answer is whether different segments of Luria's alternating series drawing tests have different discriminative powers. In order to achieve this goal, computer vision, machine learning, and object detection techniques will be used. Machine learning algorithms will use object detection to find specific patterns in the drawing tests. Afterwards the patters will be segmented further and each segment will be marked by what type of segment it is. Finally, it is shown whether the test contains micrography or perseverance. This has been verified on one type of Luria's alternating series drawing test; however, the process can be adapted to many different types of drawing tests.

The main novelty of the thesis is the automated segmentation of Luria's drawing tests. Being able to analyze which parts of the tests gives the best predictive power means that it is possible to find out which parts of the tests are the most important. Furthermore, it would be possible for further research as to why these parts are more important in terms of diagnosis than others.

In this thesis many different ways of classifying Parkinson's disease will be used. Kinematic features for the whole segmented drawing test, kinematic features from where patterns change, micrography angles and pictures of segmented drawing tests will all be used to classify patients. All the classification will be done on the DraWritePD data set which contains data from control and patients with Parkinson's drawing tests.

With the results of this thesis, it is possible to analyze the drawing tests and obtain the segmented results and determine whether the tests contain any indicators.

The thesis is organized the following. Chapter 1 will introduce the topic and the motivations behind it. Chapter 2 will give an overview of the relevant literature. Chapter 3 will describe the background of the subject. Chapter 4 will lay out the problem statement. Chapter 5 will describe the methods used in the thesis. Chapter

6 will go over the obtained results. Chapters 7 and 8 will give the closing remarks on the topic.

1.1 Motivation

Knowing that millions of people are suffering and many millions more will develop Parkinson's in their lifetimes makes diagnosing this disease extremely important. Moreover, Parkinson's has no known cure, and currently the only way to help people with this disease is to slow it down and relieve the symptoms. The earlier Parkinson's is diagnosed the more neurons might be saved meaning the persons can live a longer live with higher quality.

The goal of this thesis is not to single-handedly give the patient a diagnosis of whether they have Parkinson's or not. Rather, it would be a tool in the physician's arsenal to alleviate the amount of work they have to do. It would highlight common indicators for the tests. Furthermore, using the segmented data on classifiers will give the physician more information about the drawing test they might not be able to see just with their eyes.

One of the goals of this thesis is to introduce a robust workflow that is able to segment and analyze drawing tests that current methods are not able to segment. The traditional use of heuristical algorithms specifically for corner detection like Shi-Tomasi[26] have not been robust enough to find the correct corners on disorganized tests. Having knowledge that deep neural networks might offer more robust results gives an interesting opportunity to explore ways of segmenting the drawing tests. Having had previous experience with neural networks specifically and CNNs, might give an advantage in obtaining better results with the new methods.

Drawing tests have been used extensively to diagnose different neurological diseases. Specifically, Luria's drawing tests have been used to diagnose many different diseases such as Alzheimer's, frontotemporal dementia [32] and Parkinson's disease. This thesis will focus only on one of Luria's drawing tests; however, the workflow has been created to minimize the amount of changes required to work with other drawing tests.

Analyzing whether different segments have different predictive power can also help future research in determine exactly why patients might give more information in particular segments of the tests. Giving opportunities to more precisely focus on the segments that give the best predictive power in order to achieve better results.

Chapter 2

Literature

2.1 Object detection

In object detection, different metrics are used to show how good a model is performing. The most common are the following.

IoU - Intersection over union and represents how accurately the bounding box is over the ground truth. The bigger the IoU, the better. It is calculated by

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{2.1}$$

Precision - It shows how many of the detected objects were correct. For example, if the model detects 50 corners, but only 40 are correct, the precision will be 80%. It is calculated by

$$Precision = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{2.2}$$

Recall - It is the ratio of detected object and the number of total objects in the image. For example, the model detects 30 corners, but there are a total of 50, so the recall will be 60%. It is calculated by

$$Recall = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \tag{2.3}$$

mAP - Mean average precision, and it compares the predicted bounding box with the ground truth. mAP 0.5 and mAP 0.5: 0.95 denote different IoU thresholds.

The object detection model used in this thesis is YOLO however, first an overview of the first CNN used in object detection and its evolution will be given.

R-CNN - was introduced in [9]. It consists of three modules. The first generates category-independent region proposals, essentially meaning that it groups similar things together. The R in R-CNN means regions. These proposals will be candidates for the objects that the detector will detect. The second module is a large CNN that gives a feature vector for each region. After computing the features for each of the regions, the vectors will go to the last module a class-specific linear SVM. The SVM will classify whether an object is present in the region and what that object is. R-CNN was one of the first CNN type of object detection algorithms that achieved excellent results compared to at the time state of the art algorithms. This algorithm was however quite slow as each image takes around 47 seconds to compute as shown in [8].

Fast R-CNN - it was introduced in [8] and expands on the original concept of R-CNN. It uses several new innovations to improve speed and detection accuracy. Fast R-CNN introduces the concept of Spatial pyramid pooling networks(SPPnets) that speed up the network by sharing computations. The network processes an image with several convolutional layers and produces a feature map of it. Then each region of interest(RoI) pooling player takes a fixed length feature vector from the feature map. Then each feature vector is fed into a series of fully connected layers that have two output layers. One uses softmax to give the class of the object or background, and the other layer outputs the bounding box of the object. Fast R-CNN is 10x to 100x times faster during testing and training time is reduced by up to 3 times. The reasons why Fast R-CNN is faster than the original R-CNN is that it is single stage and it shares computations across layers.

Faster R-CNN - it was introduces in [24]. The main difference between Faster R-CNN and Fast R-CNN is the way region proposals are generated. Faster R-CNN consists of two modules, the first is the new deep fully convolutional network that proposes regions and the second is the Fast R-CNN detector that uses the proposed regions. The region proposal network is a fully convolutional network takes in an image and output a set of rectangular object proposals, each with an objectness score. Faster R-CNN runs nearly at real time and the new region proposal network improved the regions accuracy meaning the overall accuracy also improved.

YOLO - it was introduced in [21]. YOLO does not use region proposal to detect

object from the image. It uses only a single convolutional network to predict the bounding boxes and the classes of the boxes. YOLO splits the image into several grids. Each grid will try to detect an object and if the center of an object falls within a certain gird that grid will be responsible for detecting it. The bounding boxes set by the grid can be positioned outside it. Each bounding box consists of five predictions: x coordinate, y coordinate, width, height, and confidence. In object detection, in order to measure how good a bounding box is, IoU is used.

YOLOv2 - it was introduced in [22]. YOLOv2 has a host of improvements on YOLO. Batch normalization, high resolution classifier, introduction of five predefined anchor boxes, anchor box sizes are chosen based on the clustered data with the highest IoU, direct location prediction where the bounding box offsets are relative to the location of the grid cell and fall between 0 and 1, fine-grained features and multi-scale training. With all these improvements YOLOv2 is about 15% more accurate than YOLO.

YOLOv3 - it was introduced in [23]. YOLOv3 uses a more advanced backbone of Darknet-53 instead of YOLOv2's Darknet-19. Furthermore, YOLOv3 predicts bounding boxes on 3 different scales. Instead of 5 anchor boxes, YOLOv3 has 3 for each scale totaling to 9 anchor boxes.

YOLOv4 - it was introduced in [2]. YOLOv4 uses a new backbone called CSP-Darknet53. It splits the layer into two parts with one part passing through the convolution layers and the other not passing through them. The result of the layers is aggregated. There are also many small improvements with YOLOv4 and with all of improvements and the new backbone YOLOv4 achieved a 10% average precision increase and 12% more fps.

YOLOv5 - it was introduced 2 months after YOLOv4 but no official paper has been published. The main difference between YOLOv4 and YOLOv5 is that YOLOv5 uses the PyTorch framework. Furthermore, YOLOv5 introduces the Focus layer, it replaces the first three layers of the networks. This improves memory management, reduces the number of layers and increases forward propagation and back propagation [17]

The version used in this thesis is YOLOv5 and it will be referred to as YOLO.

2.2 Drawing test analysis

Previous research has been done on analyzing drawing tests in order to classify patients with Parkinson's. More specifically the most state of the art research into Luria's alternating series test is [18]. It is very similarly to this thesis in that it analyzed and segmented PL tests. The main difference will be the segmentation methods used. This thesis will focus on more advanced segmentation methods. In classification six different machine learning classifiers were used AdaBoost, DecisionTree, KNN, RandomForest, SVM with linear kernel and SVM with radial basis function kernel. In the end, the best accuracy was achieved by the RandomForest classifier at 0.91.

More research [6] has been done to analyze handwriting kinematic and pressure features in order to classify patients. Three different classifiers were analyzed, KNN, Adaboost, and SVMs. The best accuracy was obtained with SVM at 0.813. When evaluated separately, pressure features were more important than kinematic features.

Drawing tests can be used to classify different neurological diseases. When analyzing Luria's alternating Series Tests for supranuclear palsy [29] showed a 0.852 accuracy. However, the study did not have automatic segmentation of the tests, making the segmentation process time consuming and a potential bottleneck.

Previously done research into drawing tests [16, 14] laid much of the groundwork for the current research. Previous research mainly focused on the statistical analysis of kinematic features. However, no deep learning or machine learning algorithms were used to classify patients.

The meander and spiral drawing tests have been analyzed and tested on a CNN [20]. The CNN achieved a 87% accuracy in classifying PD patients. In [4] neural networks achieved a classification score of 92.9%. This shows that deep neural networks can be used to classify patients with high accuracy.

Chapter 3

Background

3.1 Luria's Alternating series tests

Luria's alternating series tests are used to help diagnose whether a patient has Parkinson's disease. The subject is asked to draw a pattern, and depending on how well they are able to accomplish this task, will be an indicator of whether they have PD. There are many patterns in the Luria's alternating series however in this thesis only the PL pattern is being analyzed. The name of the pattern is PL because the pattern reminds of Greek letters Pi (Π) and Lambda (Λ). An example of a PL test can be seen in Figure 3.1. The number values on the axis refer to the x and y coordinates of the data point.

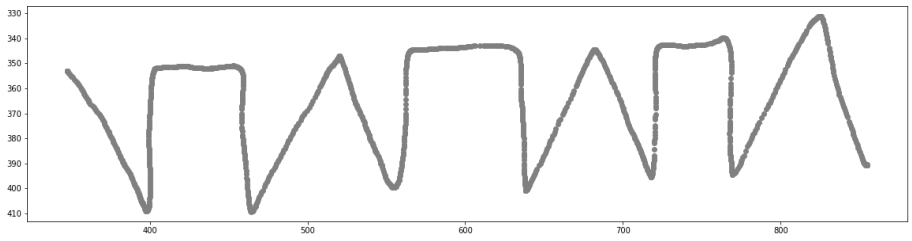

Figure 3.1: PL Test

According to [15] once decision is made to perform certain motor activity a two step process is initiated. Firstly, the planning of the motions, human brain generates the sequence of impulses referred by [15] as *motion melodies*. Secondly, the motion execution, motion melodies are transferred to the spinal cord, which controls the muscles contractions necessary to perform the action. Motion melodies

may be seen as the programs controlling muscles contractions and in turn motor action. Progressing PD may affect both steps. In the planning step, PD will lead melodies that are not optimal, meaning they require more movements to achieve the goal of the action. In turn, these may affect the execution step; nonoptimal programs may require corrections during the execution, causing the necessity to slow the movement, change its directions, and make some other adjustments. PD may also directly affect the execution phase by causing rigidity, freezings, tremor and unpurposeful movements. In turn these result in increasing amount of motion and decreasing smoothness of the motions reflected by greater acceleration and more frequent changes of directions.

In addition to the purely kinematic descriptions *micrographia* and *perseveraton* are the common symptoms of PD and used by practitioners as indicators in diagnostics. Micrographia [13] is the disorder that is reflected by the inability to keep the same size of the written letters and the repeated patterns. Persevearion is the disorder reflected by the inability to switch between the repeating patterns. During the test digitization greater attention was payed to the kinematic parameters. This accounts for the gap between the experiences of human practitioners and digital systems meant to support PD diagnostics. Namely having kinematic parameters invisible for the naked eye on the one side and semantic properties on the other side.

In this thesis, the three tasks that are being used are trace, copy, and continue. Each of them has a different level of difficulty, and they test different parts of the process.

Trace task - Patient is asked to draw the pattern with the finished pattern below where they draw their own pattern. This task does not require any significant planning phase making it the easiest task.

Copy task - Patient is asked to draw the pattern with the finished pattern being right above where they are drawing. This means that they can copy it. This task requires both the planning and implementation phases. This can cause problems even for healthy patients making it the medium task in term of difficulty.

Continue task - The patient is asked to finish the pattern when only a few segments of the pattern are shown. This requires extensive planning and implementation

phases. This task can be problematic even for healthy patients making it the hardest task of the three.

3.2 Data

A total of 24 patients with PD and 34 healthy control subjects participated in the creation of the data set[31]. The data was obtained with a strict adherence to privacy law and the data does not contain any identifiable information to keep the patients identities private.

Data is collected from patients who complete the three tasks in the Luria series mentioned above on a tablet with a stylus. The tablet saves 240 records per second, and each on-surface movement is treated like a separate stroke, where the stroke number is also saved. Meaning the data consists of an array of strokes and strokes contain an array of records [1].

Each record is an array of [x, y, t, p, a, l] where:

- x, y – coordinates
- t – timestamp
- p – vertical pressure
- a, l – altitude and latitude

3.3 Segmentation

The main way to have a deeper analysis of the PL pattern is to segment it into smaller parts. The goal of this is to have many smaller parts that are similar to each other and turn them into segments. Segment types are the following:

- Right angle corners
- Upper sharp angle corners
- Lower sharp angle corners
- Horizontal straight line
- Vertical straight line
- Diagonal line

For example, all points that form the right angle corners would be of one type. There are many steps in this segmentation, the first step is to find all the corners and their types. The second step is to segment the remaining points into their own separate segments. Third step is to find the types for these remaining separate segments. Finally, each segment will also be given a position of start, middle, or end. A fully segmented PL test is shown in Figure 3.2 with each matching segment having the same color.

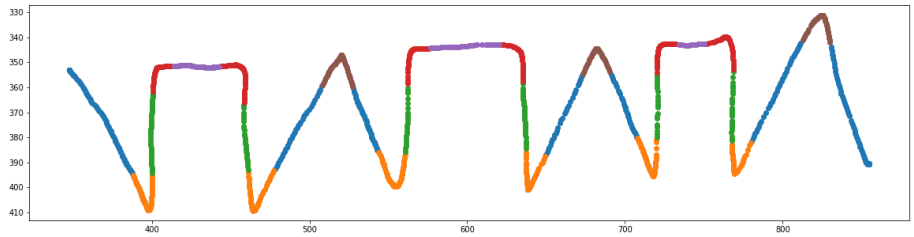

Figure 3.2: Segmented PL Test

The corners will also have some amount of data point preceding and proceeding the exact center of the corner. This will give an opportunity to classify the drawing tests based on each segment. Furthermore, if the drawing test is fully segmented it

will give opportunities to create new segment types. In this thesis transition corners will also be tested meaning corners where the pattern transitions from one shape to another.

Chapter 4

Problem statement

The main hypothesis and novel part in the thesis is that different segments of the PL pattern test contain different predictive powers. That is, they are able to better predict whether a patient has Parkinson's.
The research into Parkinson's classification and Luria's alternating series has been done previously in Tallinn University of Technology, in cooperation with Tartu university hospital (neurology clinic). This thesis relies on work done previously in [1]. The main difference will be the segmentation methods used. In previous work, in order to detect where the corners of a PL test are located, the Shi-Tomasi algorithm was used. However, this algorithm does not preform well with many drawing tests because it finds many false positive corners. In this thesis a more robust way of finding corners will be developed. Furthermore, this thesis will segment the PL test into different segments and test whether they have differing predictive powers. Also, three other classification results are obtained as a part of this thesis and they are the following: transition based classification, micrography based classification and picture based corner classification.
In order to obtain the required results, the problem will be broken into its main components:

- YOLO corner detection
- Straight line segmentation
- Segment types
- Segment position

- Indicators

Furthermore, this thesis will also cover how accurately this workflow will segment the PL tests. To verify the hypothesis, the results of the segmentation will be used to classify real patients from the DraWritePD data set. This will give a good overview of how each segment and its position affect the predictions.

Chapter 5

Methods

In this chapter, all the methods used to extract the different segments from the PL test are introduced. The extraction process consists of five steps:

- YOLO corner detection
- Straight line segmentation
- Segment types
- Segment position
- Indicators

After all extraction steps have been completed, there will be four sets of extracted results.
Pictures of corners - Picture of only the predicted corners.
Fully segmented PL test - Every data point will have a position and type of segment.
Transition segmentation - Only lower sharp corners are used. Every lower sharp corner data point will have a type from Pi to Lambda or from Lambda to Pi.
Micrography angles - PL test is divided into 3 parts according to position, and the micrography angles are taken from each set.
In Figure 5.1 it is possible to view the entire workflow of the methods in order to obtain the four sets of results. Each step is also annotated in which method it takes place. The results will later be verified by using them to detect whether a patient has Parkinson's.

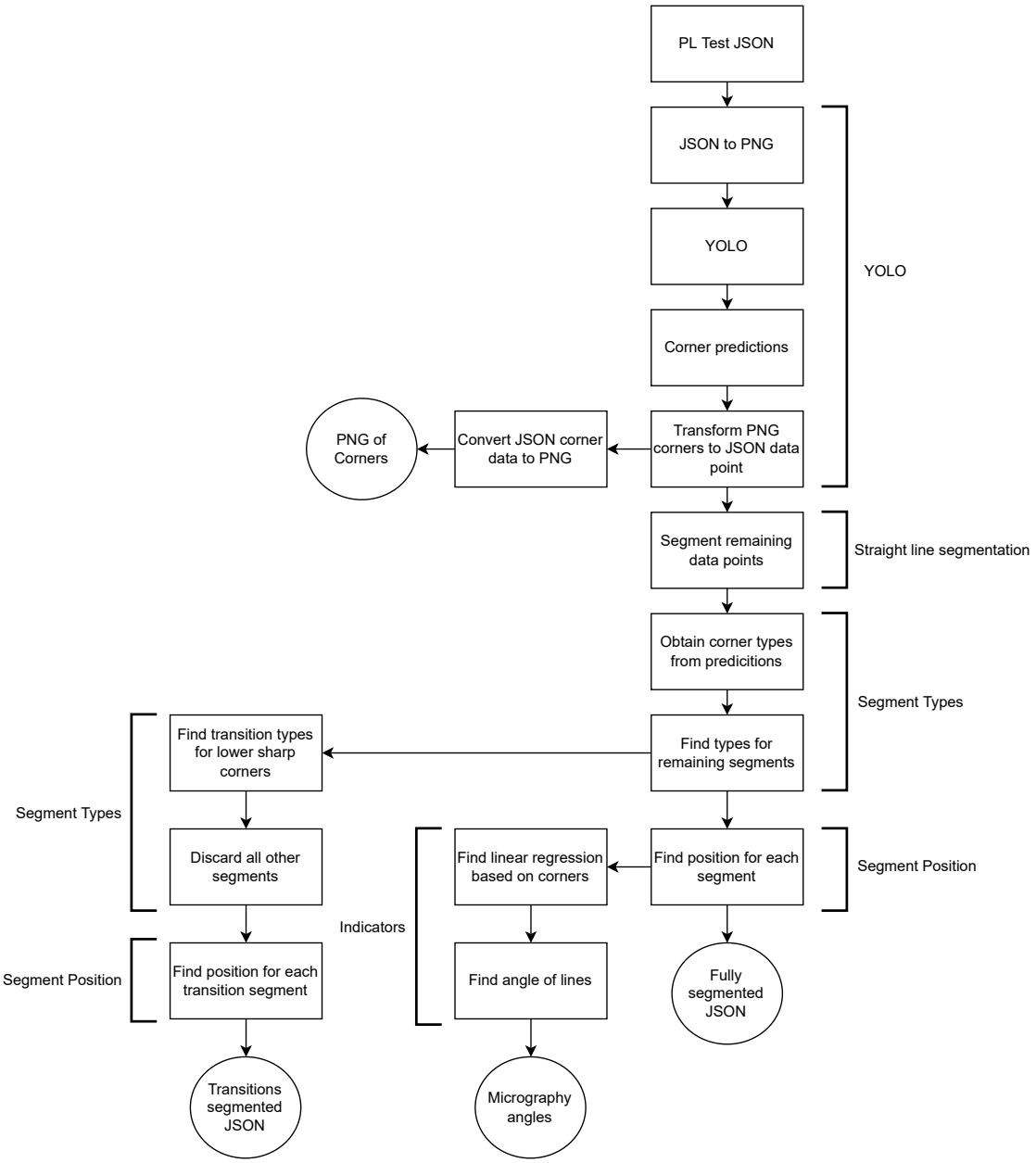

Figure 5.1: Workflow of the methods

5.1 YOLO corner detection

The first step in segmenting the PL tests is finding all the corners, and, in order to achieve this goal YOLO is used. The data provided by the tablet is in a JSON format however YOLO uses pictures to detect objects. The JSON data needs to be converted into a picture representation of the PL test. When converting the images, they are made in an aspect ratio of 4 to 1. The y axis is also inverted to get the original drawing in the correct orientation. After all the JSON data is converted into the PNG picture format, it needs to be labeled. All the labeling of the corners is done manually.
The corners will be divided into the following categories:

- Right angle corners
- Upper sharp angle corners
- Lower sharp angle corners

When labeling the corners, an area around the corner is selected in order to get a sufficient number of data points to be able to analyze the corners separately from the straight and diagonal lines. Figure 5.2 shows a single corner segment in red. It is not just the exact center point of the corner, rather it encompasses an area around it.

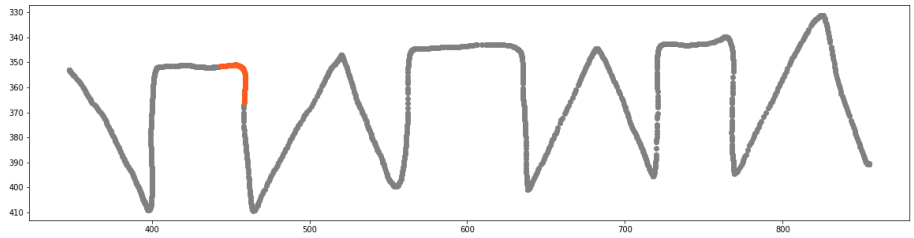

Figure 5.2: PL Test With Highlighted Corner

After all the data is labeled, it is converted to the Darknet data format because YOLO needs it in a certain format to be able to train. After conversion, the data is ready to be trained on YOLO.

The amount of real patient data is limited and in order to train and verify YOLO's results the following has been done. YOLO will be trained on artificially created PL tests, meaning that they have been purpose-created to increase the amount of data on which YOLO can be trained on. Furthermore, YOLO will not be trained on any real patient data; rather, this data will be used to verify the results from YOLO. Figure 5.3 shows how YOLO is used in this thesis. YOLO will be trained on synthetic corners and afterwards it will be used to predict corners on real patient drawing tests.

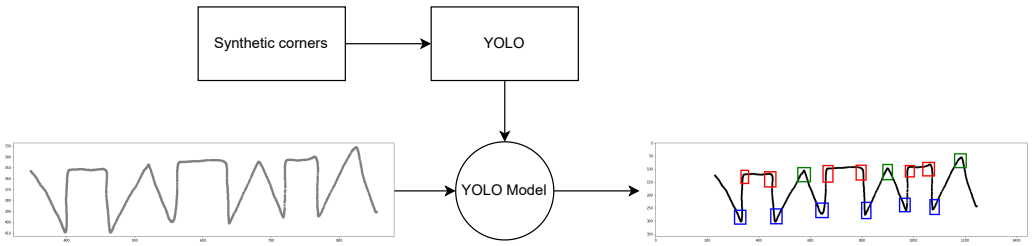

Figure 5.3: Usage of YOLO

After obtaining the corner predictions from YOLO they will be in PNG coordinates which will be converted to the original JSON coordinates. Then in order to obtain the first set of results, the JSON of only the corners is converted back into the PNG format. This gives the first set of results called pictures of corners.
Figure 5.4 shows an example of the first set of results. These images will later be used to classify whether a patient has PD by pretrained convolutional neural networks.

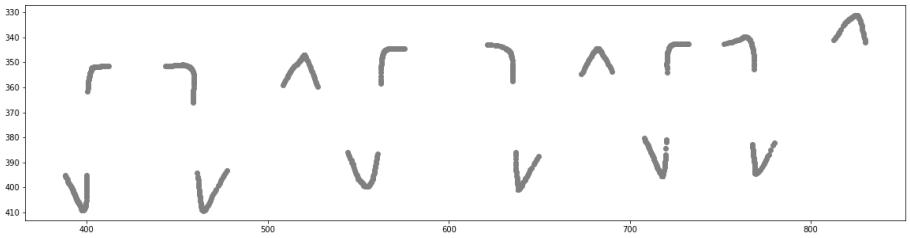

Figure 5.4: Only corners image

5.2 Straight line segmentation

The next step in segmenting the PL test is to obtain individual segments. YOLO gives the corners of the PL test; however, the rest of the remaining data also needs to be segmented. This represents a problem of how to tell when one segment begins and when another ends. It is not possible to rely solely on the time when the data point was placed because it is conceivable that the patient later tried to fix a mistake, they made earlier in the drawing process. Since all the corners are known, it is possible to ignore them and when representing the drawing without the corners, it is clear that there are caps between each of the separate segments. This can be seen clearly in Figure 5.2 where all the detected corners have been left out. It is clear that each segment is separate from the rest. This enables the segmentation to be carried out successfully.

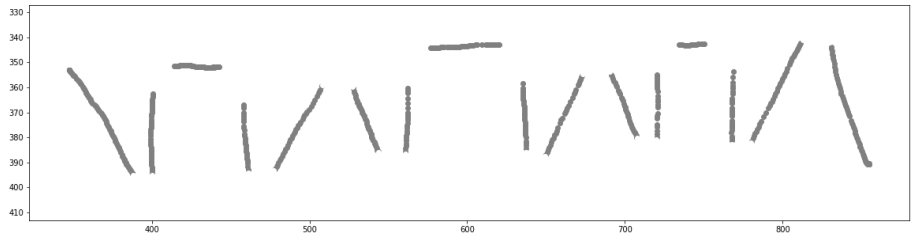

Figure 5.5: PL Test With Highlighted Corner

The approach used to cluster all data points within a certain distance of themselves as a single segment. First, select a point and designate it as a new segment. Next take the closest next point and if the distance to it is under the maximum allowed distance, then add it to the segment. Repeat until no points that are under the maximum allowed distance are left. Then select a new data point and designate it as a new segment and repeat. This approach simulates how the patient drew the drawing originally going from one point to another. Clustering solves the biggest problem in the segmentation process in that it relies solely on the x and y data to separate the segments. This method also does not need to rely on a known number of clusters, meaning that it is more robust to anomalies.

5.3 Segment types

Giving each segment a type falls largely into two main categories. The types from YOLO and the types from segmentation.
YOLO gives the following types:

- Right angle corners
- Upper sharp angle corners
- Lower sharp angle corners

Segmentation gives the following types:

- Horizontal straight line
- Vertical straight line
- Diagonal line

YOLO can already give the needed types because when it makes an object detection it also gives its class. However, the remaining lines need to be classified and in order to achieve this the known YOLO corner types are used. The lines will be classified according to the following logic:

- A line between two right angle corners is a horizontal line
- A line between two sharp angle corners is a diagonal line
- A line between one sharp angle and one right angle corner is a vertical line

The next problem is finding out which corners are next to the segment. What is needed is the data point right next to the corner. This makes it possible to check what the closest corner type is for that point. Since the remaining types are all lines the closest point to the corners will be the tips of the line. Meaning two points that have the maximum distance from each other. Finding two points that have a maximum distance by checking each point's distance with each other point is not cost-effective. The more efficient way is to find the convex hull of the data points and comparing the distances of the convex hull instead. This will give the two points

that have the maximum distance between each other. Then checking which corner is the closest to that particular point will give the two surrounding corners of the line.
After this step, all individual segments will be segmented and will have their associated type with them. This means that the types for the fully segmented PL test are found.
The next step is to find the types for the transitions from Pi to Lambda and Lambda to Pi. Each lower sharp corner will receive a type.

- Pi to Lambda
- Lambda to Pi

To determine whether the lower sharp corner is going from Pi to Lambda or Lambda to Pi, the previous segments are analyzed. If the previous segment to the lower sharp corner is a diagonal straight line or an upper sharp angle corner, the type is Lambda to Pi. If the previous segment is a horizontal, a vertical line or a right angle corner, then the type is Pi to Lambda. After all the types for the lower sharp corners are found, then all other segment data point are discarded.

5.4 Segment position

The next step is to find the position of each segment. The segments will be divided into three positions.

- Start
- Middle
- End

To give each data point its corresponding position, the minimum and maximum x values from the data are found. Then the test is divided into 3 same length parts based on the x-axis. If a segment's mean x value is in the first part, it is a start segment. If it is in the second part, it is a middle segment and if it is in the third part it is a end segment. Figure 5.6 shows how a PL test will be split on positions. The blues are the segments with position start, green are segments with position middle and red are segments with position end.

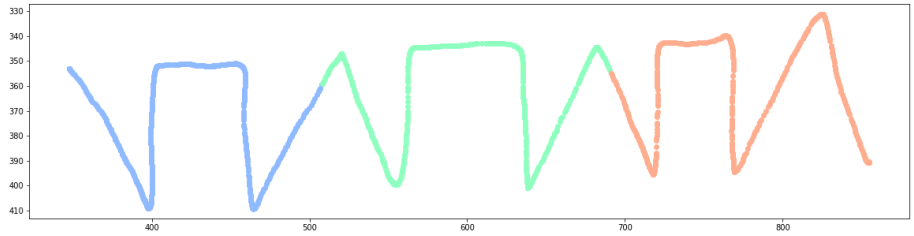

Figure 5.6: PL Test Positions

After this step segments will have their corresponding types and positions. This means that the results for the fully segmented PL test and transition segmentation are completed

5.5 Indicators

The two main indicators that are analyzed for each PL test are the following.

- Micrography
- Perseveration

For micrography, a linear regression is used to find the upper and lower lines of all the corners. With the lines, it is possible to find the angle of the upper and lower lines and their difference. Figure 5.7 shows how the micrography is found for each position separately. Micrography will be calculated by finding the angle between the regression line and the x-axis. Finding the delta between the angles can tell whether there is any micrography in PL test. A positive delta angle means that the lines are diverging and the pattern is getting larger with distance. The reverse is true with a negative delta angle. An indicator will be defined as a two-time deviation from the standard distribution of the angle. Each test type will have its own standard deviation because as the tests become harder, the standard distribution will most likely increase.

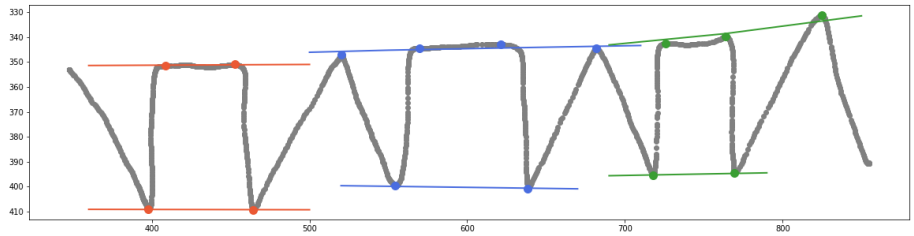

Figure 5.7: Micrography example

To detect whether the pattern is drawn correctly, the lower sharp angles are analyzed. Since the pattern is supposed to continuously change if any repetition is found, which means that the pattern goes from Lambda to Lambda or Pi to Pi, an indicator has been found.

Chapter 6

Results

6.1 YOLO corner detection

There are a total of 90 images in the artificially created data set and it will have a split of 80/20 in the training and validation images. This means that there are a total of 18 images for validation. In order to increase the amount of training data even further the training images will also be augmented. There will be 3 augmentation methods used, horizontal flip, rotation and shear. After augmenting the images there were a total of 288 training images.

6.1.1 Training YOLO

YOLO was run for 300 epochs, and the best epoch was 237 which achieved a mAP 0.5: 0.95 of 0.2952, a mAP 0.5 of 0.8331, a precision of 0.8403, and a recall of 0.8202. The overall improvement of mAP over 300 epochs can be seen in Figure 6.1. In the first 50 epochs, a very large improvement can be seen, and mAP 0.5 is near 0.75. After 150 epochs, the mAP scores started to plateau and only very slight improvements can be seen.

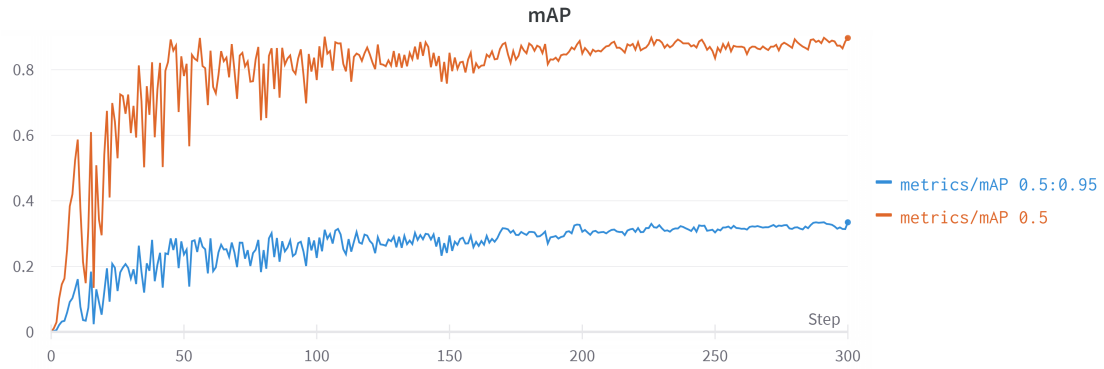

Figure 6.1: mAP of yolo with 300 epochs

The precision and recall are similar in characteristics to mAP and that can be seen in Figure 6.2. After about 100 epochs they start to plateau, and only minor improvements can be seen. Furthermore, recall and precision share almost the same value meaning the model is accurate in terms of both detecting most of the object in the picture and having few false positives. Precision, recall and mAP are calculated on only validation images.

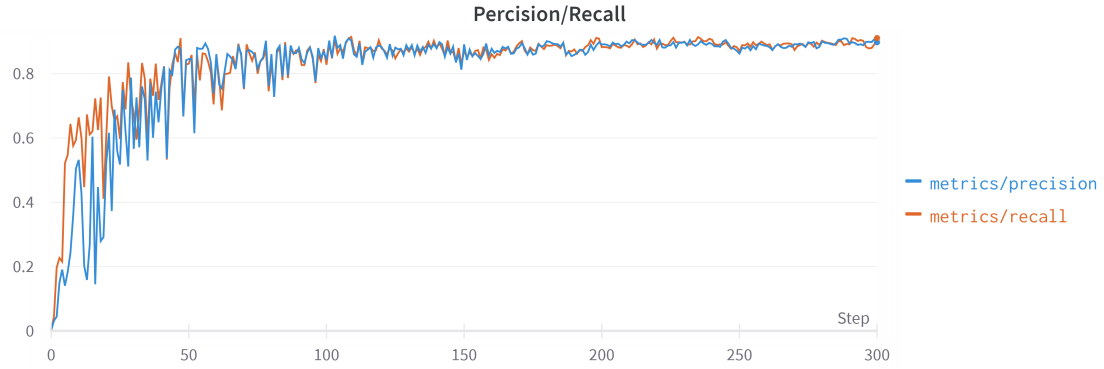

Figure 6.2: Recall and precision of yolo with 300 epochs

In Figure 6.3 the box loss is shown. The box loss represents the loss function for how accurately the bounding box has been placed. Over the course of the 300 epochs, the training loss was slowly decreasing; however, that is not the same for the validation. The validation loss does not change significantly after 150 epochs; however, notably, no real overfitting can be seen on the validation loss.

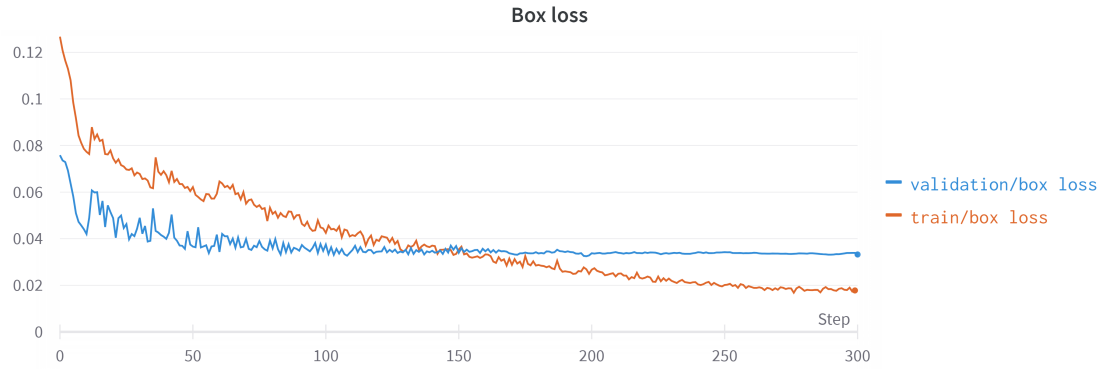

Figure 6.3: Box loss of yolo with 300 epochs

Class loss represents how well the model classifies the three corner types from each other. The class loss for YOLO can be seen in Figure 6.4. After a significant drop in the beginning epochs both the training and validation losses start to plateau with only the training loss having slight improvements. Notably again there are no signs of overfitting for the class loss.

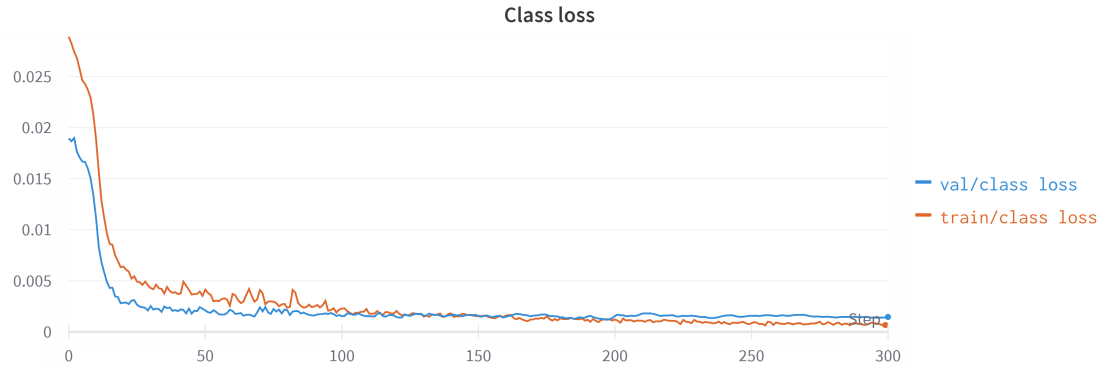

Figure 6.4: Class loss of yolo with 300 epochs

The object loss represents how the model deals with whether or not there is an object in the bounding box. This can be seen in Figure 6.5 and this loss graph differs quite significantly from the previous two. The training loss is slowly decreasing over time as expected however significant overfitting can be seen on the validation loss. After about 150 epochs the validation loss starts to increase and at the end of the 300 epochs it has increased significantly. However, since YOLO consists of multiple parts and loss functions, the best overall accuracy was achieved on epoch 237.

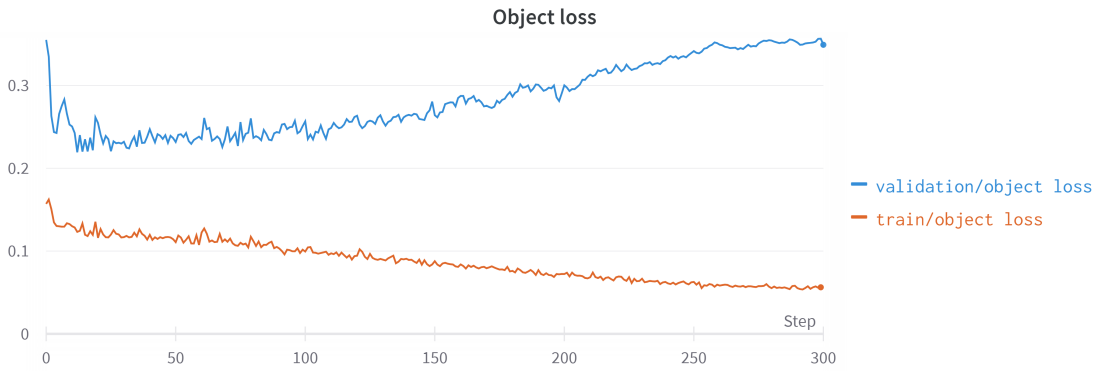

Figure 6.5: Object loss of yolo with 300 epochs

6.1.2 Validation

Since YOLO was only trained on artificially created and augmented images, it is important to verify the results on real patient data. Furthermore, it is necessary to pick a confidence threshold where any bounding boxes below are discarded and only bounding boxes with a confidence level higher than the threshold are used. This is necessary because otherwise it will increase the number of false positives and make multiple bounding boxes overlap on the same corner.

Figure 6.6 demonstrates what happens when the confidence threshold is too low. The corners will start to overlap and each bounding box might have a different class.

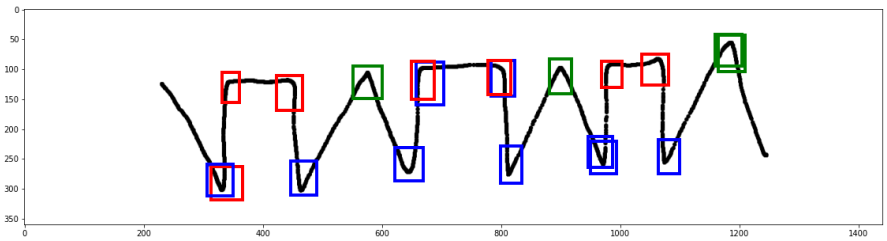

Figure 6.6: Example of low confidence threshold

Figure 6.7 demonstrates what happens when the confidence threshold is too high. YOLO will start miss some corners that it otherwise might correctly predict.

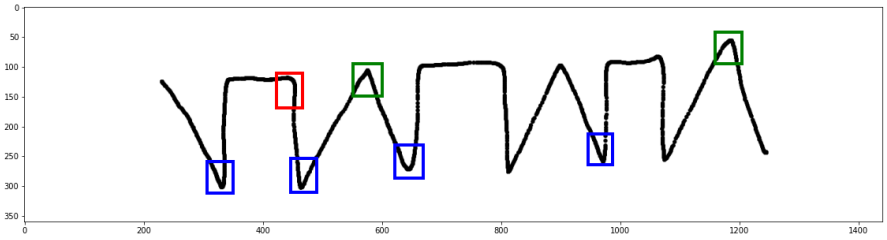

Figure 6.7: Example of high confidence threshold

Figure 6.8 shows what happens when the confidence threshold is good. All the predictions are correct, no false positives and no overlapping bounding boxes appear. YOLO generally does a good job of making sure bounding boxes do not overlap.

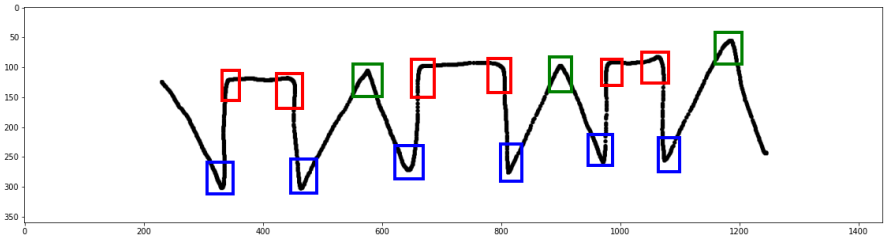

Figure 6.8: Example of good confidence threshold

In order to validate the model a number of real patient tests were used and all centers of the corners on the tests were marked. If the mark is inside the bounding box it will be counted as correct and if a bounding box does not have a mark inside of it, it will be counted as a false positive. If multiple bounding boxes are placed over a single mark, they will not be counted. This is done in order to see which confidence threshold needs the minimal amount of bounding boxes. In Table 6.1 the results for all tested confidence thresholds can be seen. The highest recall is 0.890 and highest precision is 0.978. The higher the confidence threshold the higher the precision is. This is expected, since the number of false positives should decrease as the confidence increases. The precision is high on all thresholds, which means that the number of false positives is generally quite low. However, recall does vary quite a bit with thresholds. This is because if multiple bounding boxes have the same mark, they will not be counted. The difference between the lowest and highest recall is over 10% and the difference between the precision is quite low with only 2%.

Since all the thresholds have very high precision, the threshold with best recall will be selected as the best option. This means that the best combination of precision and recall is with the 0.5 confidence threshold. This will be the confidence threshold used in future tests.

Confidence threshold	Recall	Precision
0.2	0.797	0.959
0.3	0.806	0.960
0.4	0.833	0.963
0.5	**0.890**	**0.970**
0.6	0.855	0.971
0.7	0.835	0.974
0.8	0.787	0.978

Table 6.1: Confidence threshold with recall and precision

6.1.3 Picture based corner classification

The first way to classify the patients will be with the pictures of the corners. The corners will be classified by five deep neural networks for image classification.

- Resnet50 [10]
- Xception [3]
- EfficientNetB7 [30]
- VGG19 [27]
- NASNetLarge [33]

The networks will be trained with transfer learning because the number of trainable images even with image augmentation is low. Transfer learning means pretraining the model on a bigger data set and then using it on a smaller data set. Freezing the core of the model and only training the last few layers means that the amount of overfitting is reduced. As found in [12] transfer learning will give much better results

than training a new CNN solely on the new smaller data set. All the networks are pretrained on the Imagenet [5] dataset. In order to make transfer learning possible the final classification layers are removed from the networks and are replaced with new layers. The new layers can be seen in Table 6.2. For Resnet50, Xception and VGG19 0.5, 0.5 and 0.4 dropout was used. In EfficientNetB7 and NASNetLarge the dropout values were dropped to 0.2, 0.2 and 0.1 respectively. The amount of dropout is large, however with lowered dropout some of the models started to overfit very quickly and as shown in [28] dropout decreases the amount of overfitting. Since both EfficientNetB7 and NASNetLarge are very deep networks the amount of overfitting was lower and a higher score was received with the lowered dropout values. The hidden fully connected layers use ReLu, and the output layer uses softmax. The learning rate was set at 2e−5 and the networks were trained for 50 epochs.

Layer
Batch normalization
Fully connected layer(2048)
Dropout(0.5|0.2)
Batch normalization
Fully connected layer(512)
Dropout(0.5|0.2)
Batch normalization
Fully connected layer(128)
Dropout(0.4|0.1)
Fully connected Layer(2)

Table 6.2: New classification layers

In Table 6.3 the results of the trace PL test corner classification are shown. Although EfficientNetB7 has very high accuracy, its sensitivity is very low, meaning it had a lot of false positives. The only networks that had 0.5 or above on all metrics were NASNetLarge and Resnet50 and even they had only 0.5 sensitivity. The worst performing network is VGG19. It is also the smallest tested network.

Network	Accuracy	Precision	Sensitivity	Specificity
Resnet50	0.666	0.750	0.500	0.875
Xception	0.400	0.500	**1.000**	0.250
EfficientNetB7	**0.750**	**1.000**	0.250	**1.000**
VGG19	0.250	0.000	0.000	0.375
NASNetLarge	0.666	0.750	0.500	0.875

Table 6.3: PL Trace Picture Corner Classification

The results of the PL Copy PNG corner classification are shown in Table 6.4. The results are similar to those from the trace task. The only network that had more than 0.5 on all metrics was NASNetLarge. This time it had higher sensitivity but lower accuracy, precision, and specificity. Resnet50 did not achieve as good results as the last time with a sensitivity of only 0.25. VGG19 was the worst with only 0.363 accuracy and 0 specificity.

Network	Accuracy	Precision	Sensitivity	Specificity
Resnet50	**0.630**	0.500	0.250	**0.857**
Xception	0.545	0.333	0.250	0.714
EfficientNetB7	0.400	0.454	**1.000**	0.142
VGG19	0.363	0.363	**1.000**	0.000
NASNetLarge	0.600	**0.727**	0.750	0.714

Table 6.4: PL Copy Picture Corner Classification

The results of the PL Continue picture corner classification are shown in Table 6.5. The networks that achieved over 0.5 on all metrics were NASNetLarge and EfficientNetB7. The metrics were very similar to the last two tests. VGG19 had very similar results to last time with an accuracy of 0.33 and a specificity of 0.

Network	Accuracy	Precision	Sensitivity	Specificity
Resnet50	0.666	**1.000**	0.200	**1.000**
Xception	**0.750**	**1.000**	0.400	**1.000**
EfficientNetB7	**0.750**	0.666	**0.800**	0.714
VGG19	0.333	0.363	**0.800**	0.000
NASNetLarge	0.666	0.750	**0.800**	0.714

Table 6.5: PL Continue Picture Corner Classification

In general, the only network that preformed relatively well over all three tests was NASNetLarge. Every other network had at least on a single occasion a sensitivity or specificity below 0.5. Since the number of training samples was limited, a high dropout rate was introduced in order to mitigate the effect of overfitting that was happening. The relatively small network size of VGG19 meant it tried to overfit the data very quickly. The large size of NASNetLarge meant that it was not as prone to overfitting making it able to train better on the data set. EfficientNetB7 and Resnet50 also had more than 0.5 in all metrics in at least one of the tests. However, when looking at the results of the other two tests, they are not as reliable as NASNetLarge.

6.1.4 Full segmentation

In order to fully segment a PL test, the methods described previously are used. Since every steps accuracy relies on how well the previous step preformed the overall accuracy for segmentation will be lower with each step. To gauge how well the segmentation process works, it will be used in real patient PL tests. There will be three categories for a test.

Correct - Each segment of the PL test is correctly segmented and classified.

Slight mistakes - A single mistake in segmentation. A corner not predicted correctly, misclassified, or a mistake in segmentation.

Major mistakes - More than one mistake in segmentation or more than one corner classified incorrectly

The results can be seen in Figure 6.6. Slightly more than 70% of the tests are

segmented correctly. About 20% have a single mistake in them and about 10% have more than 1 mistake. This will give an idea how well this process works on PL tests. Most tests are perfectly segmented, and some have only slight mistakes. The most common errors in the slight mistakes category were YOLO model misclassifying the corner type and two straight lines touching each other causing them to be classified as a single segment. In most cases where a major mistake occurred, the PL itself had a few anomalies making the segmentation process hard. Overall, the segmentation process works well and even though a direct comparison between YOLO's results and these results is not possible, the accuracy of the segmentation means not much accuracy is lost in between the steps.

Type	Test distribution
Correct	0.701
Slight mistakes	0.197
Major mistakes	0.102

Table 6.6: Fully segmented PL test results

6.1.5 Full segmentation based classification

In order to identify whether different segment types have different predictive power they were used to classify healthy and Parkinson's patients. Six machine learning classifiers were used, Logistic Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF) and AdaBoost (AB). A total of 202 features were extracted from the raw signals of the fully segmented PL test. A SVM recursive feature elimination wrapper method was used to decrease the dimensionality of the data. This works by training the SVM to predict the importance a set of features [31].

The workflow of [31] was applied to each type of the segments. Each segmentation will be split by test types, continue, copy, and trace. Each test will be further divided by position and segment type. The best results for each segment and position and the full results for each classifier and shown in the appendix.

In Tables 6.7 and 6.8 an example of kinematic and pressure features are shown.

These features are calculated from the fully segmented PL test for each segment. These features describe changes in the trajectory, pressure and hand position. Each segment will be tested with one to four features. The exact features used for the results are shown in the appendix.

Feature set	Feature	Description
Spatial-temporal features	displacement	$d_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$
Kinematic features	velocity	Rate of change in displacement with respect to time. First time derivative of the displacement.
	acceleration	Rate of change in velocity with respect to time. Second time derivative of the displacement.
	jerk	Rate of change in acceleration with respect to time. Third time derivative of the displacement.
	snap	Rate of change in jerk with respect to time. Fourth time derivative of the displacement.
	crackle	Rate of change in snap with respect to time. Fifth time derivative of the displacement.
	pop	Rate of change in crackle with respect to time. Sixth time derivative of the displacement.
Pressure features	pressure_diff	Change in pressure between points $[p_i, p_{i+1}]$
	yank	Rate of change in pressure. First time derivative of the force applied on the surface.
	tug	Rate of change in yank. Second time derivative of the force applied on the surface.
	snatch	Rate of change in tug. Third time derivative of the force applied on the surface.
	shake	Rate of change in snatch. Fourth time derivative of the force applied on the surface.
Geometric features	altitude_diff	Change in altitude angle between points $[p_i, p_{i+1}]$
	azimuth_diff	Change in the azimuth angle between points $[p_i, p_{i+1}]$
	alphas_diff	Change in alpha angle between points $[p_i, p_{i+1}]$
	phi_angle_diff	Change in phi angle between points $[p_i, p_{i+1}]$
	yaw_diff	Change in yaw angle between points $[p_i, p_{i+1}]$

Table 6.7: Sample subset of vector features.[31]

Feature set	Feature	Description
Spatial-temporal features	duration	Time interval between first and last time stamp signal
Kinematic features	velocity_mass	Velocity mass of the point vector $[p_1, p_2, \ldots, p_k, \ldots, p_N]$
	acceleration_x_mass	Mass of the x-directional rate of change in velocity
	jerk_median	Median value of the rate of change in acceleration
	snap_mass	Mass of the fifth time derivative of displacement.
Pressure features	shake_median	Median value of the fifth time derivative of pressure (force applied on the surface).
	pressure_diff_min	Minimum difference of pressure between points $[p_i, p_j]$
	tug_mass	Mass of the second time derivative of pressure (force applied on the surface)
Geometric features	ϕ_mass	Mass of the angle ϕ (in radians)
	α_accel_min	Minimum acceleration of the angle α
	yaw_std	Standard deviation of the yaw

Table 6.8: Sample subset of single-value features.[31]

In the appendix, the results have a type from 1 to 5 and a position. The types are the following, one is vertical, two is horizontal, three is sharp angle, four is right angle and five is diagonal. Furthermore, each segment is classified four times by different features. The features can be seen in the feature column. Each feature was classified by the six above mentioned classifiers.

In Table 6.9 the best results of the fully segmented PL trace test can be seen. The best segment to classify the test was the vertical lines at the end of the test. They achieved an accuracy of 0.938, a precision of 0.960, a sensitivity of 0.950 and a specificity of 0.900. All the metrics are quite similar in terms of accuracy, meaning that the number of false positives and false negatives is also very low. In terms of Parkinson's disease, the main thing to avoid are false negatives, which means that false positives, although not good, are more acceptable. The lowest scoring segment was the sharp angles at the start of the test. It had an accuracy of 0.619, a precision of 0.6533, a sensitivity of 0.783, and a specificity of 0.467. Overall, the vertical lines gave the very good predictive power in each position and the rest of the segments hovered around 0.75 to 0.90.

Position	Type	Accuracy	Precision	Sensitivity	Specificity
Start	Vertical	0.905	0.860	**1.000**	0.800
	Horizontal	0.776	0.850	0.717	0.867
	Sharp angle	0.619	0.653	0.783	0.467
	Right angle	0.776	0.900	0.667	**0.933**
	Diagonal	0.871	0.910	0.867	0.867
Middle	Vertical	0.819	0.867	0.783	0.867
	Horizontal	0.743	0.883	0.683	0.800
	Sharp angle	0.714	0.713	0.833	0.567
	Right angle	0.848	0.820	0.950	0.733
	Diagonal	0.724	0.780	0.717	0.733
End	**Vertical**	**0.938**	**0.960**	0.950	0.900
	Horizontal	0.657	0.763	0.650	0.633
	Sharp angle	0.748	0.740	0.900	0.567
	Right angle	0.843	0.933	0.783	**0.933**
	Diagonal	0.776	0.753	0.867	0.667

Table 6.9: PL Trace Kinematic Classification

In Figure 6.9 the same results are shown on a training PL example. A visual approach gives a better idea as to where exactly on the drawing test these segments are. Exactly like in the table the brightest green segments are the vertical lines at the end of the test.

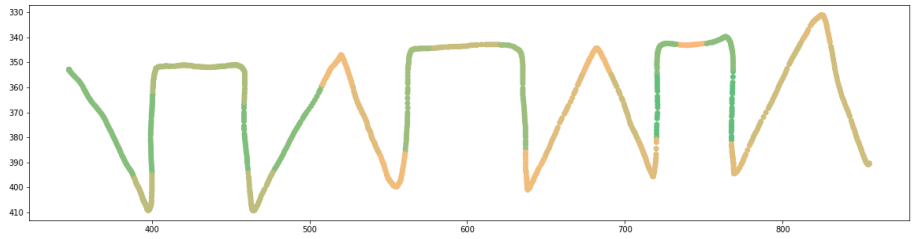

Figure 6.9: PL Trace Heatmap

The results of the PL copy task can be seen in Table 6.10. In the copy task, the best performing segment is the sharp angles at the start. This is somewhat surprising

since the sharp angles performed poorly in the trace task. However, the verticals at the end of the test are also good at distinguishing between the classes, just like in the trace task. Furthermore, the specificity of the sharp angles is higher than that of the verticals in the trace task. There is no clear segment that is worse than the rest. Most of the segment's hover around 0.75 to 0.90, which is similar to the trace task.

Position	Type	Accuracy	Precision	Sensitivity	Specificity
Start	Vertical	0.800	0.790	**0.950**	0.667
	Horizontal	0.810	0.900	0.783	0.833
	Sharp angle	**0.938**	**1.000**	0.883	**1.000**
	Right angle	0.838	0.933	0.817	0.800
	Diagonal	0.876	0.910	0.900	0.833
Middle	Vertical	0.771	0.813	0.817	0.700
	Horizontal	0.805	0.883	0.783	0.867
	Sharp angle	0.910	0.910	**0.950**	0.833
	Right angle	0.848	0.950	0.783	0.933
	Diagonal	0.843	0.850	0.900	0.767
End	Vertical	**0.938**	0.950	0.950	0.933
	Horizontal	0.820	0.920	0.833	0.800
	Sharp angle	0.686	0.753	0.733	0.667
	Right angle	0.781	0.883	0.717	0.867
	Diagonal	0.867	0.860	0.933	0.767

Table 6.10: Fully segmented PL Copy Kinematic Classification

In Figure 6.10 the visualization of the results is shown. On average there is more green on this figure compared to the last one. Visually it is easy to see that as position goes from start to end the predictive power of the sharp corners also worsens. End vertical lines still have good predictive power.

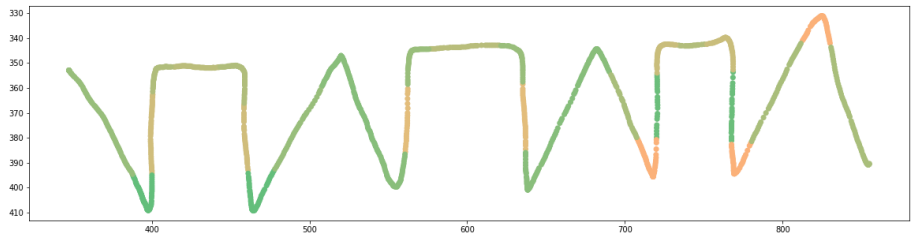

Figure 6.10: PL Copy Heatmap

The results of the continuing task are given in Table 6.11. The best performing segments were the vertical lines in the middle of the PL test. They have similar metrics to the best of the copy and trace tasks. The only notable difference is the sensitivity being 100% in this task. The worst segment is the right angle corners in the middle of the test. They have the worst metrics of all the segments in the tests.

Position	Type	Accuracy	Precision	Sensitivity	Specificity
Start	Vertical	0.629	0.670	0.783	0.467
	Horizontal	0.867	0.900	0.883	0.867
	Sharp angle	0.840	0.900	0.867	0.800
	Right angle	0.700	0.740	0.833	0.600
	Diagonal	0.820	0.850	0.883	0.767
Middle	**Vertical**	**0.967**	**0.950**	**1.000**	**0.933**
	Horizontal	0.780	0.767	0.867	0.700
	Sharp angle	0.780	0.817	0.800	0.767
	Right angle	0.560	0.550	0.733	0.367
	Diagonal	0.827	0.900	0.867	0.800
End	Vertical	0.805	0.763	**1.000**	0.533
	Horizontal	0.787	0.920	0.767	0.867
	Sharp angle	0.667	0.700	0.733	0.567
	Right angle	0.867	**0.950**	0.817	**0.933**
	Diagonal	0.781	0.933	0.683	**0.933**

Table 6.11: Fully segmented PL Continue Kinematic Classification

In Figure 6.11 the visualization of the results is shown. This time the brightest

green is the middle vertical lines. The right angles in the end are also greener than the rest however, the rest of the segments are similar in terms of predictive power.

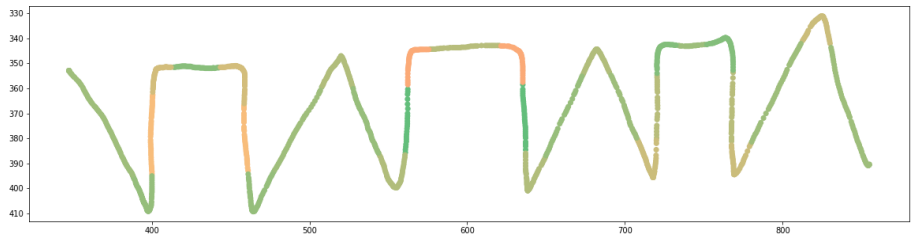

Figure 6.11: PL Continue Heatmap

Overall, the segments with the best predictive power were the vertical lines and sharp corners. The sharp corners performed better than the right angle corners; this might be because at the sharp corners more complex movement is taking place. The vertical lines provided the biggest predictive power of all the lines, however the diagonal lines were also very close in term of accuracy. The horizontal lines although their metrics were not bad did not preform as well as the other two lines. This might be because horizontal lines are usually shorter than the rest meaning the number of data points is also smaller. The results also align with the results from [18], which achieved an accuracy of 0.91. Interestingly the position of the segment did not seem to have a clear effect on the results. Each test had the best segment from different positions. With these results it is clear which segments hold the best predictive power.

6.1.6 Transition segmentation

In this section the classification by transition corners will be reviewed. The transitions will be from Pi to Lambda and from Lambda to Pi. Only data points that are in the lower sharp corner segments will be used.

Figure 6.12 shows an example of transition corners. All the transition corners are shown in red. They are the corners that are in between different pattern types. Analyzing when the subject switches from one pattern to another can give additional information that can help the classification process.

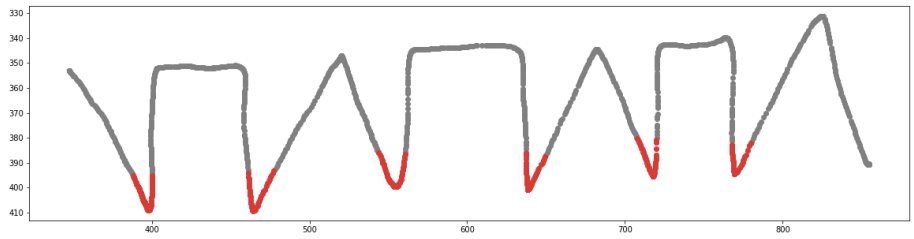

Figure 6.12: Example of transition corners

Table 6.12 shows the percentages of tests with no mistakes, slight mistakes, and major mistakes. The mistakes are the same as previously defined. It is clear that only segmenting the lower sharp corners and their types has higher accuracy than fully segmenting the PL test which is expected because the overall number of segments is smaller. The number of slight mistakes and major mistake is halved.

Type	Test distribution
Correct	0.856
Slight mistakes	0.106
Major mistakes	0.038

Table 6.12: Transition corners PL test results

6.1.7 Transition segmentation based classification

The same classification methods and classifiers as previously mentioned are used. In Table 6.13 the results of the trace task can be seen. The segment with the best discriminative power was the Lambda to Pi transition at the end of the test. It had an accuracy of 0.811, a precision of 0.883, a sensitivity of 0.617, and a specificity of 0.927. As stated above, having a low number of false negatives is important in classifying diseases. Furthermore the accuracy and precision are about 10% lower than in the fully segmented tests. The sensitivity for all of the segments is very poor at around 65%. Overall, the transition corners had worse results than the full segmentation based classification.

Position	Type	Accuracy	Precision	Sensitivity	Specificity
Start	Pi to Lambda	0.756	0.680	0.690	0.810
	Lambda to Pi	0.729	0.650	**0.700**	0.753
Middle	Pi to Lambda	0.765	0.773	0.650	0.843
	Lambda to Pi	0.740	0.733	0.550	0.867
End	Pi to Lambda	0.740	0.798	0.630	0.833
	Lambda to Pi	**0.811**	**0.883**	0.617	**0.927**

Table 6.13: Corner transition PL Trace Kinematic Classification

The visualization of the results can be seen in Figure 6.13. Only one of the segments is bright green the rest do not have good predictive power. The segments at the start of the test are performing worse than the segments at the end.

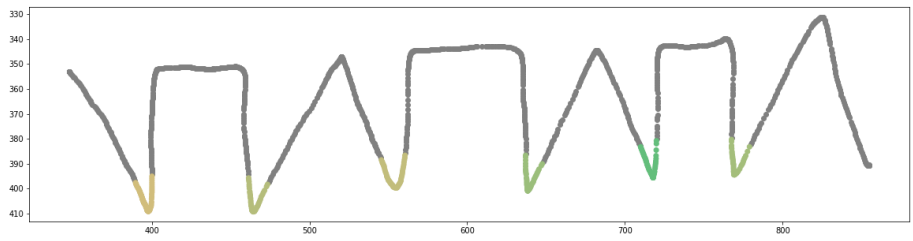

Figure 6.13: PL Trace Corners Heatmap

The results of the copy task can be seen in Table 6.14. This time the best segment was the Pi to Lambda in the middle of the test. Each metric is better than in the trace task. In particular, the sensitivity has improved quite a bit. However, the rest of the segments are similar to those from the trace task. They have poor sensitivity and lower accuracy. Although all of them for the exception of Pi to Lambda at the start have good specificity. The best segments still do not give as good a result as the fully segmented tests did. However, it did perform better than the trace task's segment.

Position	Type	Accuracy	Precision	Sensitivity	Specificity
Start	Pi to Lambda	0.680	0.656	**0.800**	0.600
	Lambda to Pi	0.671	0.633	0.450	0.814
Middle	**Pi to Lambda**	**0.860**	**0.920**	0.750	**0.933**
	Lambda to Pi	0.727	0.633	0.650	0.786
End	Pi to Lambda	0.780	0.840	0.650	0.867
	Lambda to Pi	0.747	0.780	0.600	0.843

Table 6.14: Corner transition PL Copy Kinematic Classification

The visualization of the results is shown in Figure 6.14. This visualization is very similar to the last test. Only one of the segments is bright green with the rest being yellow. The start position has the worst segments in term of predictive power.

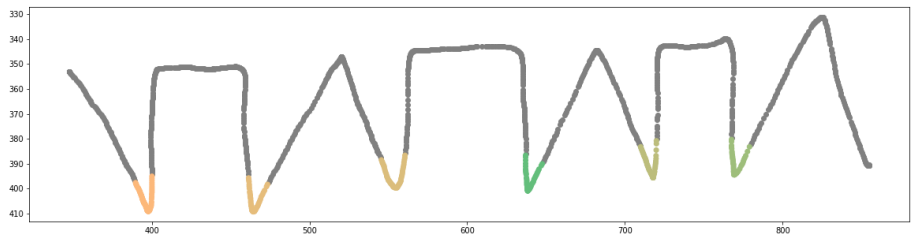

Figure 6.14: PL Copy Corners Heatmap

In Table 6.15 the results of the continuation task are shown. The best segment is the Pi to Lambda at the end of the test. Overall, the results are very similar to those of the trace task. There is no certain worst performing segment since most of them have very low scores. Again the metric with the highest score is specificity. Overall the results are worse than the results achieved in full segmentation based classification.

Position	Type	Accuracy	Precision	Sensitivity	Specificity
Start	Pi to Lambda	0.698	0.667	0.417	0.867
	Lambda to Pi	0.718	0.684	**0.667**	0.753
Middle	Pi to Lambda	0.744	0.763	0.633	0.813
	Lambda to Pi	0.706	0.533	0.367	0.887
End	**Pi to Lambda**	**0.847**	**0.933**	**0.667**	**0.960**
	Lambda to Pi	0.711	0.727	0.600	0.760

Table 6.15: Corner transition PL Continue Kinematic Classification

The visualization of the results is shown in Figure 6.15. It is a very similar picture with only one good segment. This time, however, start and middle both have terrible segments.

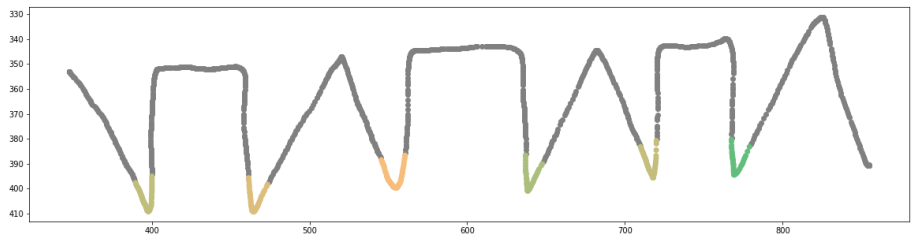

Figure 6.15: PL Continue Corners Heatmap

The transition segments did not perform as well as full segmentation based classification. It especially had a poor sensitivity that is not suitable for helping to accurately diagnose patients. The start position is the worst in terms of predictive power. This was very well illustrated on the visualized results. Overall, these results fall behind in terms of metrics to the full segmentation based classification and other state of the art solutions.

6.2 Indicators

After fully segmenting the PL test, the indicators can also be analyzed. The application will highlight whether the PL test has any of the following:

- Micrography
- Perseveration

Since the transition segmentation requires each lower corner to have a type of Pi to Lambda or Lambda to Pi, the only thing needed to find if the pattern is repeating correctly is to arrange the corners by the x-axis. Then if any corner is the same as the last, it means that the pattern is repeating incorrectly, and it will be highlighted when the segmentation is run. A perseveration indicator was discovered in only 1.7% of the tests.

Any micrographcy angle greater than 2 standard deviations from the norm will be counted as a micrography indication. In Table 6.16 the standard deviations of the test types are shown. As predicted, as the tests become more difficult the standard deviation of the tests increases. If the application discovers that the micrography angle of the test is more than two times the standard deviation it will be highlighted

Type	Degrees
Trace	1.5
Copy	2.4
Continue	4.2

Table 6.16: Micrography angle standard deviation

6.2.1 Micrography based classification

The classification is done the same as before. The micrography angles will be split into the three position same as with the other tests. Each PL test task will be examined separately.

In Table 6.17 the first results can be seen. The best position is the start, however the results are not good when compared to the full segmentation based classification

results. Specificity is high; however, sensitivity is very low at 0.650 for the best position and only 0.200 for the worst. The precision also drops from 0.680 to 0.460 along the positions. Precision hovers around 45% to 70% which is not impressive when compared to previous results.

Position	Accuracy	Precision	Sensitivity	Specificity
Start	**0.680**	0.600	**0.650**	0.700
Middle	0.660	**0.700**	0.200	**0.967**
End	0.460	0.469	0.250	0.600

Table 6.17: Corner transition PL Trace Micrography Angles

The results of the copy task can be seen in Table 6.18. Here the best results is the middle position with an accuracy of 0.685 and generally the results are very similar to the trace task. Overall, the results were not impressive, especially when compared to earlier fully segmented results.

Position	Accuracy	Precision	Sensitivity	Specificity
Start	0.607	0.200	0.117	0.905
Middle	**0.685**	**0.800**	0.217	**0.967**
End	0.585	0.413	**0.350**	0.714

Table 6.18: PL Copy Micrography Angles

In Table 6.18 the results of the continue task are shown. Overall, it is a very similar story to the two previous results. The only metric that has reasonable values is specificity. In general, the results are poor and it would be impossible to classify the patients with any degree of certainty.

Position	Accuracy	Precision	Sensitivity	Specificity
Start	0.567	0.417	0.350	0.700
Middle	**0.711**	**0.733**	**0.467**	**0.867**
End	0.691	0.633	0.450	0.833

Table 6.19: PL Continue Micrography Angles

The results of the micrography angles were poor. Position does not play a significant role in the predictive power of the angles. Angles have extremely poor predictive power when compared to full segmentation based classification results. This further cements the fact that the results gotten from the them were very good. Accuracy, precision, and sensitivity are about 20% to 30% higher in the fully segmented tests. However, the specificity remained relatively good even compared to that of full segmentation based classification results. This result can also be seen in the transition segments. The micrography angles performed even worse than the transition corners which had about 10% to 15% higher values. Overall, the micrography angles had similar scores to the pictures of corners classification results.

Chapter 7

Discussion

The goal of this thesis was to answer the following problem, whether different segments have differing predictive power. Different segments do have different predictive powers and the vertical lines in full segmentation based classification had the best results. YOLO was used to find all the corners on a PL drawing test. YOLO had a recall of 89% and a precision of 97% in fining corners. Overall, four segmentation methods were used.

Picture based corner classification with pretrained image classification networks had a best accuracy of 66.6%, a precision of 75%, a sensitivity of a 80% and a specificity of 71.4%.

Full segmentation based classification had a best results of 96.7% accuracy, 95% precision, 100% sensitivity and 93.3% specificity.

Transition segmentation classification had a best accuracy of 86%, precision of 92%, sensitivity of a 75% and a specificity of 93.3%.

Micrography based classification had a best accuracy of 68%, precision of 60%, sensitivity of a 65% and a specificity of 70%.

The best preforming segmentation method used was full segmentation based classification that had an accuracy of 96.7% on the PL continue test with the vertical middle lines. The second best was transition segmentation classification that had an accuracy of 86%. Third and fourth place were a tie between picture based corner classification and micrography based classification with an accuracy of 66.6% and 68% respectively.

One drawback to using YOLO to detect corners is that each YOLO model will

have slightly different bounding boxes for the corners meaning every segment will be slightly different with each model. This poses a problem when wanting to check whether the test is segmented correctly or not. Currently they were checked by hand however, an automated validation method would be preferred. Another drawback to using object detection models is the fact that the bounding boxes are rectangular. Segmentation with rectangular boxes might not work for every drawing test, for example the spiral pattern.

The segmentation algorithm used after YOLO corner detection worked well. In full segmentation it got 70% correct and on transition corners it got 85% correct. This means that if the test is performed without any anomalies present in the data, it would most likely get segmented correctly. However, a drawback to the segmentation is the fact that if the straight line segments are too close together they will be counted as a single segment. However, in those scenarios, often times YOLO also struggles to find the correct corners. To increase the number of correctly segmented tests, two things have to be improved. First, the corner prediction model has to be made more accurate, and second, a more advanced segmentation algorithm has to be developed.

Overall, it does not seem like any particular test type like trace, copy, or continue makes it easier to classify patient. Each classification method had different test types as the best. Furthermore, the position of the segments did not seem to have be a factor in the predictive power of the segment. Except in transition segmentation classification where start had the worst performing segments.

The main advantage besides the improved accuracy of the current process is the robustness of it. Since YOLO is a deep neural network it can more easily adapt to unexpected circumstances than a traditional corner detection algorithm.

Perseveration was also analyzed, however perseveration only occurred for 1.7% of PL drawing tests. This is not enough data to draw any certain conclusions. A bigger data set would be needed for a deeper analysis of perseveration.

Chapter 8

Conclusion

The goal of this thesis was to determine whether different segments have different predictive powers. A novel automatic corner detection deep learning model was used to achieve segmentation of the PL test. The segment with the most predictive power were the vertical lines in the middle of the PL continue test. Full segmentation based classification reached a best accuracy of 96.7%, a precision of 95%, a sensitivity of 100% and a specificity of 93% with the above mentioned segment.

Three other classification methods were used.

Picture based corner classification with pretrained image classification networks had a best accuracy of 66.6%, precision of 75%, sensitivity of a 80% and a specificity of 71.4%.

Transition segmentation classification had a best accuracy of 86%, precision of 92%, sensitivity of a 75% and a specificity of 93.3%.

Micrography based classification had a best accuracy of 68%, precision of 60%, sensitivity of a 65% and a specificity of 70%.

However, they did not offer as good of a results as full segmentation based classification.

Overall, 70% of the PL tests were segmented correctly, 20% with only slight mistakes and 10% with major mistakes. As long as the PL test does not contain any anomalies, there is a very good chance that it would be segmented correctly.

For future research there are many different ways to advance the research. Improving the corner detection models and segmentation algorithms would be an obvious choice. Expanding the workflow to function on different drawing patterns like the

meander drawing test would also be an excellent direction to take.

Acknowledgments

I would like to thank my supervisors Sven Nõmm for his insight on the topic, Elli Valla for the continued guidance throughout this research process and Aaro Toomela for coming up with new ideas. Lastly, I would like to thank my mother for her continued support and faith in me.

Bibliography

[1] Konstantin Bardoš. Analysis of interpretable anomalies and kinematic parameters in lurias alternating series tests for parkinsons disease modeling. Master's thesis, Tallinn University of Technology, 2018.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

[4] Resul Das. A comparison of multiple classification methods for diagnosis of parkinson disease. *Expert Systems with Applications*, 37(2):1568–1572, 2010.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[6] Peter Drotar, Jiri Mekyska, Irena Rektorova, Lucia Masarova, Zdenek Smekal, and Marcos Faundez-Zanuy. Evaluation of handwriting kinematics and pressure for differential diagnosis of parkinson's disease. *Artificial intelligence in medicine*, 67, 01 2016.

[7] Rossiter JP Frank C, Pari G. Approach to diagnosis of parkinson disease. *Canadian family physician Medecin de famille canadien*, 52(7):862–868, 2006.

[8] Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[11] Christopher W Hess and Michael S Okun. Diagnosing parkinson disease. *CONTINUUM: Lifelong Learning in Neurology*, 22(4):1080–2371, 2016.

[12] Mahbub Hussain, Jordan J. Bird, and Diego Resende Faria. A study on cnn transfer learning for image classification. In *UKCI*, 2018.

[13] Rivka Inzelberg, Meir Plotnik, Naama Kadmon Harpaz, and Tamar Flash. Micrographia, much beyond the writer's hand. *Parkinsonism & Related Disorders*, 26:1–9, 2016.

[14] Julia Kozhenkina. Quantitative analysis of the kinematic features for the luria's alternating series test. Master thesis, Tallinn University of Technology, 2016.

[15] Aleksandr Romanovich Luria. *Higher Cortical Functions in Man*. Springer, 1995.

[16] Ilja Mašarov. Digital clock drawing test implementation and analysis. Master thesis, Tallinn University of Technology, 2017.

[17] U Nepal and H Eslamiat. Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs. *Sensors*, 22(2):464, 2022.

[18] Sven Nõmm, Konstantin Bardõš, Aaro Toomela, Kadri Medijainen, and Pille Taba. Detailed analysis of the luria's alternating seriestests for parkinson's disease diagnostics. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1347–1352. IEEE, 2018.

[19] Sven Nomm, Aaro Toomela, Julia Kozhenkina, and Toomas Toomsoo. Quantitative analysis in the digital luria's alternating series tests. In *2016 14th Inter-*

national Conference on Control, Automation, Robotics and Vision (ICARCV), pages 1–6, 2016.

[20] Clayton R Pereira, Silke AT Weber, Christian Hook, Gustavo H Rosa, and João P Papa. Deep learning-aided parkinson's disease diagnosis from handwritten dynamics. In *Graphics, Patterns and Images (SIBGRAPI), 2016 29th SIBGRAPI Conference on*, pages 340–346. IEEE, 2016.

[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[22] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[23] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[25] A H V Schapira. Parkinson's disease. *BMJ*, 318(7179):311–314, 1999.

[26] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[29] P. Stepien, J. Kawa, D. Wieczorek, M. Dabrowska, J. Slawek, and Sitek E.J. Computer aided feature extraction in the paper version of luria's alternating series test in progressive supranuclear palsy. In E. Pietka, P. Badura, J. Kawa,

and W. Wieclawek, editors, *Information Technology in Biomedicine*, pages 561–570. Springer, 2019.

[30] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

[31] Elli Valla, Sven Nõmm, Kadri Medijainen, Pille Taba, and Aaro Toomela. Tremor-related feature engineering for machine learning based parkinson's disease diagnostics. *Biomedical Signal Processing and Control*, 75:103551, 2022.

[32] Myron F Weiner, Linda S Hynan, Heidi Rossetti, and Jed Falkowski. Luria's three-step test: what is it and what does it tell us? *Journal of consulting and clinical psychology*, 23(10):1602–1606, 2011.

[33] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.

Appendix A

Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Henry Laur

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Automated Segmentation and Semantic Analysis of Writing and Drawing Tests for Parkinson's Disease Diagnostics", supervised by Sven Nõmm, Elli Valla, Aaro Toomela

   1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

   1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2. I am aware that the author also retains the rights specified in clause 1 of the non exclusive licence

3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

11.05.2022

[1] The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix B

Results of the fully segmented PL trace task

settings	features	model	accuracy	precision	sensitivity	specificity
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'start')]	2 Index(['acceleration_y_mean', 'azimuth_median'], dtype='object')	LogReg	0.904762	0.86	1	0.8
		RF	0.814286	0.89333333	0.76666667	0.86666667
		KNN	0.814286	0.81666667	0.88333333	0.73333333
		SVM	0.871429	0.84333333	0.93333333	0.8
		DT	0.757143	0.74333333	0.83333333	0.66666667
		AdaBoost	0.757143	0.74333333	0.83333333	0.66666667
	3 Index(['acceleration_y_mean', 'acceleration_y_min', 'azimuth_median'], dtype='object')	LogReg	0.814286	0.79333333	0.88333333	0.73333333
		RF	0.752381	0.75	0.81666667	0.66666667
		KNN	0.752381	0.76666667	0.83333333	0.63333333
		SVM	0.847619	0.86666667	0.88333333	0.8
		DT	0.72381	0.72666667	0.76666667	0.66666667
		AdaBoost	0.72381	0.70333333	0.83333333	0.56666667
	4 Index(['acceleration_y_mean', 'acceleration_y_min', 'yaw_mean', 'azimuth_median'],	LogReg	0.780952	0.78333333	0.88333333	0.66666667
		RF	0.785714	0.81666667	0.83333333	0.73333333
		KNN	0.785714	0.81666667	0.83333333	0.73333333
		SVM	0.785714	0.81666667	0.83333333	0.73333333
		DT	0.728571	0.73333333	0.78333333	0.66666667
		AdaBoost	0.757143	0.74333333	0.83333333	0.66666667
	5 Index(['acceleration_y_mean', 'acceleration_y_min', 'yaw_mean', 'azimuth_median',	LogReg	0.819048	0.83333333	0.9	0.73333333
		RF	0.842857	0.84333333	0.88333333	0.8
		KNN	0.847619	0.83333333	0.95	0.73333333
		SVM	0.819048	0.83333333	0.9	0.73333333
		DT	0.728571	0.73333333	0.78333333	0.66666667
		AdaBoost	0.757143	0.74333333	0.83333333	0.66666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'start')]	2 Index(['alphas_acceleration_max', 'altitude_median'], dtype='object')	LogReg	0.77619	0.85	0.71666667	0.86666667
		RF	0.752381	0.8	0.71666667	0.8
		KNN	0.780952	0.83333333	0.78333333	0.8
		SVM	0.719048	0.73333333	0.76666667	0.66666667
		DT	0.695238	0.7	0.61666667	0.8
		AdaBoost	0.728571	0.73333333	0.68333333	0.8
	3 Index(['alphas_acceleration_max', 'altitude_median', 'azimuth_median'], dtype='object')	LogReg	0.657143	0.74	0.66666667	0.66666667
		RF	0.752381	0.75	0.76666667	0.73333333
		KNN	0.690476	0.74	0.71666667	0.66666667
		SVM	0.685714	0.7	0.76666667	0.6
		DT	0.719048	0.82	0.71666667	0.73333333
		AdaBoost	0.657143	0.8	0.66666667	0.66666667
	4 Index(['alphas_acceleration_max', 'snap_mass', 'altitude_median', 'azimuth_median'], dtype='object')	LogReg	0.628571	0.72	0.61666667	0.66666667
		RF	0.728571	0.72666667	0.71666667	0.73333333
		KNN	0.72381	0.75	0.71666667	0.73333333
		SVM	0.628571	0.72	0.56666667	0.73333333
		DT	0.757143	0.81	0.76666667	0.73333333
		AdaBoost	0.747619	0.75333333	0.88333333	0.6

	5 Index(['phi_angle_velocity_min', 'alphas_acceleration_max', 'snap_mass', 'altitude_median',	LogReg	0.752381	0.78333333	0.71666667	0.8
		RF	0.728571	0.73333333	0.73333333	0.73333333
		KNN	0.752381	0.75333333	0.83333333	0.66666667
		SVM	0.719048	0.78333333	0.66666667	0.8
		DT	0.728571	0.73333333	0.73333333	0.73333333
		AdaBoost	0.657143	0.74	0.65	0.66666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 3), ('position: ', 'start')]	2 Index(['jerk_x_min', 'altitude_jerk_mean'], dtype='object')	LogReg	0.619048	0.63666667	0.73333333	0.5
		RF	0.619048	0.70333333	0.68333333	0.6
		KNN	0.619048	0.63666667	0.73333333	0.5
		SVM	0.585714	0.61666667	0.66666667	0.5
		DT	0.585714	0.60666667	0.73333333	0.43333333
		AdaBoost	0.552381	0.57333333	0.68333333	0.43333333
	3 Index(['velocity_std', 'jerk_x_min', 'altitude_jerk_mean'], dtype='object')	LogReg	0.619048	0.63666667	0.73333333	0.5
		RF	0.647619	0.73666667	0.68333333	0.66666667
		KNN	0.585714	0.60333333	0.68333333	0.5
		SVM	0.614286	0.63666667	0.68333333	0.56666667
		DT	0.519048	0.55	0.55	0.5
		AdaBoost	0.642857	0.68333333	0.65	0.63333333
	4 Index(['velocity_std', 'jerk_x_min', 'pressure_diff_median', 'altitude_jerk_mean'],	LogReg	0.685714	0.70333333	0.78333333	0.56666667
		RF	0.652381	0.66666667	0.73333333	0.56666667
		KNN	0.552381	0.58666667	0.73333333	0.36666667
		SVM	0.709524	0.76666667	0.73333333	0.7
		DT	0.585714	0.60333333	0.68333333	0.5
		AdaBoost	0.67619	0.71666667	0.71666667	0.63333333
	5 Index(['velocity_std', 'jerk_x_min', 'pressure_diff_median', 'azimuth_mean',	LogReg	0.680952	0.73666667	0.71666667	0.63333333
		RF	0.652381	0.73333333	0.68333333	0.66666667
		KNN	0.619048	0.65333333	0.78333333	0.46666667
		SVM	0.77619	0.83333333	0.78333333	0.76666667
		DT	0.552381	0.57	0.61666667	0.5
		AdaBoost	0.709524	0.75	0.71666667	0.7
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 4), ('position: ', 'start')]	2 Index(['velocity_x_mass', 'phi_angle_velocity_mass'], dtype='object')	LogReg	0.72381	0.74333333	0.78333333	0.63333333
		RF	0.719048	0.74333333	0.71666667	0.7
		KNN	0.661905	0.59333333	0.7	0.56666667
		SVM	0.690476	0.74333333	0.71666667	0.63333333
		DT	0.690476	0.71666667	0.71666667	0.63333333
		AdaBoost	0.661905	0.68333333	0.71666667	0.56666667
	3 Index(['velocity_x_mass', 'phi_angle_velocity_mass', 'yaw_velocity_mass'], dtype='object')	LogReg	0.728571	0.72666667	0.83333333	0.56666667
		RF	0.752381	0.81666667	0.78333333	0.7
		KNN	0.695238	0.69333333	0.83333333	0.5
		SVM	0.690476	0.72	0.76666667	0.56666667
		DT	0.652381	0.73333333	0.66666667	0.6
		AdaBoost	0.595238	0.60333333	0.71666667	0.4
	4 Index(['velocity_x_mass', 'phi_angle_velocity_mass', 'yaw_velocity_mass', 'azimuth_max'],	LogReg	0.7	0.70666667	0.83333333	0.5
		RF	0.661905	0.70333333	0.66666667	0.63333333
		KNN	0.666667	0.65666667	0.83333333	0.43333333
		SVM	0.633333	0.69	0.65	0.56666667
		DT	0.633333	0.72	0.61666667	0.66666667

	dtype='object')	AdaBoost	0.666667	0.68	0.71666667	0.56666667
	5	LogReg	0.757143	0.75333333	0.83333333	0.63333333
	Index(['velocity_std',	RF	0.747619	0.79333333	0.78333333	0.7
	'velocity_x_mass',	KNN	0.728571	0.72666667	0.83333333	0.56666667
	'phi_angle_velocity_mass	SVM	0.666667	0.68	0.71666667	0.56666667
	',	DT	0.77619	0.9	0.66666667	0.93333333
	'yaw_velocity_mass',	AdaBoost	0.72381	0.74333333	0.71666667	0.7
[('scaler: ',	2	LogReg	0.742857	0.75	0.8	0.66666667
StandardScaler()),	Index(['jerk_y_max',	RF	0.809524	0.83333333	0.8	0.8
('folds: ', 5), ('type:	'jerk_y_min'],	KNN	0.77619	0.78333333	0.8	0.73333333
', 5), ('position: ',	dtype='object')	SVM	0.77619	0.78333333	0.8	0.73333333
'start')]		DT	0.771429	0.86	0.68333333	0.86666667
		AdaBoost	0.838095	0.86	0.81666667	0.86666667
	3	LogReg	0.77619	0.76666667	0.8	0.73333333
	Index(['acceleration_y_m	RF	0.809524	0.78333333	0.86666667	0.73333333
	ax', 'jerk_y_max',	KNN	0.742857	0.71666667	0.8	0.66666667
	'jerk_y_min'],	SVM	0.742857	0.71666667	0.8	0.66666667
	dtype='object')	DT	0.714286	0.77428571	0.73333333	0.63333333
		AdaBoost	0.871429	0.91	0.86666667	0.86666667
	4	LogReg	0.680952	0.69761905	0.75	0.6
	Index(['acceleration_y_m	RF	0.714286	0.70761905	0.8	0.56666667
	ax', 'jerk_y_max',	KNN	0.747619	0.73095238	0.86666667	0.6
	'jerk_y_min',	SVM	0.77619	0.76666667	0.8	0.73333333
	'altitude_diff_mass'],	DT	0.714286	0.79761905	0.75	0.63333333
	dtype='object')	AdaBoost	0.614286	0.61428571	0.75	0.43333333
	5	LogReg	0.77619	0.76666667	0.8	0.73333333
	Index(['acceleration_y_m	RF	0.747619	0.74761905	0.8	0.66666667
	ax', 'jerk_y_max',	KNN	0.814286	0.84761905	0.86666667	0.73333333
	'jerk_y_min',	SVM	0.77619	0.76666667	0.8	0.73333333
	'altitude_diff_mass',	DT	0.714286	0.74761905	0.75	0.66666667
		AdaBoost	0.804762	0.79333333	0.8	0.8
[('scaler: ',	2	LogReg	0.819048	0.83333333	0.83333333	0.8
StandardScaler()),	Index(['acceleration_y_m	RF	0.785714	0.79333333	0.83333333	0.7
('folds: ', 5), ('type:	edian',	KNN	0.819048	0.86666667	0.78333333	0.86666667
', 1), ('position: ',	'yaw_velocity_mass'],	SVM	0.790476	0.81666667	0.78333333	0.8
'middle')]	dtype='object')	DT	0.785714	0.79333333	0.83333333	0.7
		AdaBoost	0.785714	0.79333333	0.83333333	0.7
	3	LogReg	0.819048	0.83333333	0.83333333	0.8
	Index(['acceleration_y_m	RF	0.785714	0.83333333	0.76666667	0.8
	edian', 'jerk_x_mean',	KNN	0.790476	0.79761905	0.88333333	0.66666667
	'yaw_velocity_mass'],	SVM	0.785714	0.79333333	0.76666667	0.8
	dtype='object')	DT	0.752381	0.76	0.76666667	0.7
		AdaBoost	0.690476	0.72666667	0.7	0.63333333
	4	LogReg	0.790476	0.78666667	0.88333333	0.66666667
	Index(['acceleration_y_m	RF	0.819048	0.81666667	0.88333333	0.73333333
	edian', 'jerk_x_mean',	KNN	0.819048	0.84761905	0.88333333	0.73333333
	'yaw_velocity_mass',	SVM	0.728571	0.73428571	0.81666667	0.6

	'azimuth_jerk_mean'],	DT	0.72381	0.73	0.76666667	0.63333333
		AdaBoost	0.72381	0.71333333	0.81666667	0.56666667
	5 Index(['acceleration_median', 'acceleration_y_median', 'jerk_x_mean', 'yaw_velocity_mass',	LogReg	0.790476	0.78666667	0.88333333	0.66666667
		RF	0.847619	0.86666667	0.88333333	0.8
		KNN	0.819048	0.84761905	0.88333333	0.73333333
		SVM	0.757143	0.81428571	0.76666667	0.73333333
		DT	0.661905	0.64428571	0.81666667	0.43333333
		AdaBoost	0.72381	0.71333333	0.81666667	0.56666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'middle')]	2 Index(['jerk_median', 'jerk_x_median'], dtype='object')	LogReg	0.57619	0.66666667	0.51666667	0.66666667
		RF	0.614286	0.72	0.58333333	0.66666667
		KNN	0.642857	0.76666667	0.58333333	0.73333333
		SVM	0.57619	0.66666667	0.58333333	0.6
		DT	0.52381	0.55	0.53333333	0.5
		AdaBoost	0.580952	0.65333333	0.51666667	0.66666667
	3 Index(['jerk_median', 'jerk_x_median', 'shake_max'], dtype='object')	LogReg	0.57619	0.6	0.56666667	0.56666667
		RF	0.709524	0.78333333	0.68333333	0.73333333
		KNN	0.609524	0.73333333	0.51666667	0.73333333
		SVM	0.547619	0.58095238	0.56666667	0.5
		DT	0.67619	0.73333333	0.68333333	0.63333333
		AdaBoost	0.542857	0.56666667	0.51666667	0.56666667
	4 Index(['jerk_median', 'jerk_x_median', 'tug_max', 'shake_max'], dtype='object')	LogReg	0.614286	0.60761905	0.66666667	0.5
		RF	0.742857	0.88333333	0.68333333	0.8
		KNN	0.580952	0.61428571	0.61666667	0.53333333
		SVM	0.614286	0.69761905	0.61666667	0.56666667
		DT	0.614286	0.70333333	0.6	0.66666667
		AdaBoost	0.614286	0.68666667	0.58333333	0.63333333
	5 Index(['jerk_median', 'jerk_x_median', 'tug_max', 'shake_max', 'altitude_mean'], dtype='object')	LogReg	0.647619	0.62428571	0.73333333	0.5
		RF	0.742857	0.75333333	0.81666667	0.66666667
		KNN	0.647619	0.62428571	0.73333333	0.5
		SVM	0.742857	0.86	0.68333333	0.76666667
		DT	0.714286	0.81666667	0.66666667	0.76666667
		AdaBoost	0.714286	0.84	0.7	0.73333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 3), ('position: ', 'middle')]	2 Index(['velocity_max', 'pressure_diff_mean'], dtype='object')	LogReg	0.652381	0.68333333	0.71666667	0.56666667
		RF	0.685714	0.69333333	0.76666667	0.56666667
		KNN	0.585714	0.61	0.71666667	0.4
		SVM	0.685714	0.69333333	0.76666667	0.56666667
		DT	0.652381	0.69333333	0.71666667	0.56666667
		AdaBoost	0.652381	0.69333333	0.71666667	0.56666667
	3 Index(['velocity_max', 'pressure_diff_mean', 'tug_max'], dtype='object')	LogReg	0.62381	0.65	0.66666667	0.56666667
		RF	0.590476	0.64	0.66666667	0.5
		KNN	0.652381	0.67	0.78333333	0.5
		SVM	0.690476	0.67666667	0.78333333	0.56666667
		DT	0.652381	0.69333333	0.71666667	0.56666667
		AdaBoost	0.628571	0.64333333	0.71666667	0.5
	4 Index(['velocity_max', 'pressure_diff_mean',	LogReg	0.657143	0.66666667	0.73333333	0.56666667
		RF	0.62381	0.66	0.66666667	0.56666667
		KNN	0.714286	0.71333333	0.83333333	0.56666667

	'tug_max', 'azimuth_mean'], dtype='object')	SVM	0.719048	0.71	0.78333333	0.63333333
		DT	0.590476	0.65	0.61666667	0.56666667
		AdaBoost	0.690476	0.7	0.71666667	0.66666667
	5 Index(['velocity_max', 'jerk_y_min', 'pressure_diff_mean', 'tug_max', 'azimuth_mean'],	LogReg	0.657143	0.66	0.71666667	0.56666667
		RF	0.62381	0.66	0.66666667	0.56666667
		KNN	0.657143	0.64666667	0.78333333	0.5
		SVM	0.652381	0.69333333	0.66666667	0.63333333
		DT	0.528571	0.63333333	0.56666667	0.5
		AdaBoost	0.628571	0.68333333	0.66666667	0.6
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 4), ('position: ', 'middle')]	2 Index(['acceleration_mean', 'altitude_acceleration_max'], dtype='object')	LogReg	0.780952	0.78666667	0.81666667	0.73333333
		RF	0.780952	0.77	0.81666667	0.73333333
		KNN	0.847619	0.82	0.95	0.73333333
		SVM	0.847619	0.82	0.95	0.73333333
		DT	0.719048	0.87	0.66666667	0.8
		AdaBoost	0.719048	0.71333333	0.83333333	0.56666667
	3 Index(['acceleration_mean', 'altitude_diff_std', 'altitude_acceleration_max'], dtype='object')	LogReg	0.747619	0.74666667	0.81666667	0.63333333
		RF	0.780952	0.82	0.75	0.8
		KNN	0.780952	0.76333333	0.88333333	0.63333333
		SVM	0.780952	0.76333333	0.88333333	0.63333333
		DT	0.719048	0.68	0.68333333	0.73333333
		AdaBoost	0.685714	0.69666667	0.76666667	0.56666667
	4 Index(['acceleration_mean', 'pressure_median', 'altitude_diff_std', 'altitude_acceleration_m	LogReg	0.77619	0.77666667	0.81666667	0.7
		RF	0.77619	0.81	0.75	0.76666667
		KNN	0.77619	0.77666667	0.81666667	0.7
		SVM	0.77619	0.77666667	0.81666667	0.7
		DT	0.719048	0.68	0.68333333	0.73333333
		AdaBoost	0.752381	0.77333333	0.81666667	0.63333333
	5 Index(['acceleration_mean', 'pressure_median', 'altitude_diff_std', 'azimuth_median', 'altitude_acceleration_m	LogReg	0.809524	0.84333333	0.81666667	0.76666667
		RF	0.780952	0.81333333	0.81666667	0.7
		KNN	0.747619	0.73333333	0.88333333	0.56666667
		SVM	0.780952	0.77666667	0.83333333	0.7
		DT	0.685714	0.64	0.68333333	0.63333333
		AdaBoost	0.719048	0.78666667	0.76666667	0.6
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 5), ('position: ', 'middle')]	2 Index(['jerk_min', 'jerk_x_min'], dtype='object')	LogReg	0.72381	0.74	0.76666667	0.63333333
		RF	0.72381	0.74	0.76666667	0.63333333
		KNN	0.690476	0.69	0.76666667	0.56666667
		SVM	0.72381	0.74	0.76666667	0.63333333
		DT	0.657143	0.68	0.71666667	0.56666667
		AdaBoost	0.62381	0.68	0.65	0.56666667
	3 Index(['jerk_min', 'jerk_x_min', 'yaw_velocity_mean'], dtype='object')	LogReg	0.690476	0.67666667	0.71666667	0.63333333
		RF	0.62381	0.65	0.71666667	0.46666667
		KNN	0.62381	0.7	0.65	0.53333333
		SVM	0.590476	0.63666667	0.6	0.53333333
		DT	0.657143	0.66333333	0.76666667	0.46666667
		AdaBoost	0.657143	0.68	0.71666667	0.56666667
	4 Index(['jerk_min',	LogReg	0.62381	0.66666667	0.6	0.63333333
		RF	0.690476	0.68	0.76666667	0.6

	'jerk_x_min', 'yaw_median', 'yaw_velocity_mean'], dtype='object')	KNN	0.62381	0.7	0.65	0.53333333
		SVM	0.595238	0.62	0.66666667	0.46666667
		DT	0.72381	0.78	0.71666667	0.73333333
		AdaBoost	0.690476	0.73	0.71666667	0.63333333
	5 Index(['jerk_min', 'jerk_x_min', 'yaw_median', 'yaw_velocity_mean', 'tug_median'],	LogReg	0.619048	0.68333333	0.63333333	0.56666667
		RF	0.690476	0.73	0.71666667	0.66666667
		KNN	0.495238	0.53666667	0.53333333	0.4
		SVM	0.490476	0.53333333	0.51666667	0.4
		DT	0.657143	0.7	0.71666667	0.6
		AdaBoost	0.72381	0.78	0.7	0.73333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'end')]	2 Index(['acceleration_y_max', 'alphas_velocity_max'], dtype='object')	LogReg	0.880952	0.87	0.95	0.76666667
		RF	0.842857	0.84333333	0.88333333	0.76666667
		KNN	0.880952	0.87	0.95	0.76666667
		SVM	0.880952	0.87	0.95	0.76666667
		DT	0.842857	0.84333333	0.88333333	0.76666667
		AdaBoost	0.814286	0.80333333	0.88333333	0.7
	3 Index(['acceleration_y_max', 'yaw_mass', 'alphas_velocity_max'], dtype='object')	LogReg	0.938095	0.96	0.95	0.9
		RF	0.87619	0.92	0.88333333	0.83333333
		KNN	0.909524	0.92	0.95	0.83333333
		SVM	0.880952	0.89333333	0.95	0.76666667
		DT	0.842857	0.84333333	0.88333333	0.76666667
		AdaBoost	0.757143	0.77666667	0.78333333	0.7
	4 Index(['acceleration_y_mean', 'acceleration_y_max', 'yaw_mass',	LogReg	0.87619	0.86	0.95	0.76666667
		RF	0.842857	0.84333333	0.88333333	0.76666667
		KNN	0.871429	0.82	1	0.7
		SVM	0.847619	0.82	0.95	0.7
		DT	0.814286	0.80333333	0.88333333	0.7
		AdaBoost	0.842857	0.84333333	0.88333333	0.76666667
	5 Index(['acceleration_y_mean', 'acceleration_y_max', 'phi_angle_jerk_std', 'yaw_mass',	LogReg	0.814286	0.78	0.95	0.63333333
		RF	0.842857	0.81	0.95	0.7
		KNN	0.809524	0.76333333	1	0.56666667
		SVM	0.819048	0.79	0.95	0.63333333
		DT	0.814286	0.77	0.95	0.63333333
		AdaBoost	0.87619	0.86	0.95	0.76666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'end')]	2 Index(['pressure_std', 'tug_mass'], dtype='object')	LogReg	0.657143	0.69666667	0.71666667	0.56666667
		RF	0.528571	0.55333333	0.6	0.43333333
		KNN	0.561905	0.58666667	0.66666667	0.43333333
		SVM	0.528571	0.55333333	0.66666667	0.36666667
		DT	0.52381	0.55333333	0.65	0.33333333
		AdaBoost	0.457143	0.5	0.58333333	0.26666667
	3 Index(['velocity_mass', 'pressure_std', 'tug_mass'], dtype='object')	LogReg	0.62381	0.73333333	0.65	0.53333333
		RF	0.652381	0.75333333	0.7	0.53333333
		KNN	0.62381	0.66666667	0.71666667	0.46666667
		SVM	0.62381	0.73333333	0.65	0.53333333
		DT	0.657143	0.71333333	0.71666667	0.56666667
		AdaBoost	0.62381	0.64666667	0.73333333	0.5
	4	LogReg	0.657143	0.73333333	0.71666667	0.53333333

	Index(['velocity_mass', 'pressure_std', 'tug_mass', 'azimuth_std'], dtype='object')	RF	0.652381	0.58333333	0.63333333	0.63333333
		KNN	0.590476	0.73333333	0.58333333	0.53333333
		SVM	0.557143	0.53	0.51666667	0.56666667
		DT	0.657143	0.76333333	0.65	0.63333333
		AdaBoost	0.590476	0.56333333	0.6	0.56666667
	5 Index(['velocity_mass', 'pressure_std', 'pressure_deriv_mass', 'tug_mass', 'azimuth_std'],	LogReg	0.62381	0.73333333	0.65	0.53333333
		RF	0.652381	0.58333333	0.63333333	0.63333333
		KNN	0.62381	0.73333333	0.65	0.53333333
		SVM	0.590476	0.53333333	0.58333333	0.53333333
		DT	0.657143	0.76333333	0.65	0.63333333
		AdaBoost	0.590476	0.56333333	0.6	0.56666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 3), ('position: ', 'end')]	2 Index(['velocity_std', 'velocity_x_std'], dtype='object')	LogReg	0.714286	0.72333333	0.85	0.56666667
		RF	0.614286	0.70666667	0.65	0.56666667
		KNN	0.747619	0.74	0.9	0.56666667
		SVM	0.714286	0.72333333	0.83333333	0.56666667
		DT	0.580952	0.60666667	0.65	0.5
		AdaBoost	0.647619	0.73666667	0.65	0.63333333
	3 Index(['velocity_std', 'velocity_x_std', 'snatch_std'], dtype='object')	LogReg	0.619048	0.68666667	0.71666667	0.5
		RF	0.614286	0.68666667	0.65	0.56666667
		KNN	0.647619	0.70666667	0.71666667	0.56666667
		SVM	0.614286	0.67333333	0.66666667	0.56666667
		DT	0.580952	0.65333333	0.6	0.56666667
		AdaBoost	0.514286	0.55333333	0.6	0.43333333
	4 Index(['velocity_std', 'velocity_x_std', 'alphas_velocity_mean', 'snatch_std'], dtype='object')	LogReg	0.680952	0.72333333	0.76666667	0.56666667
		RF	0.647619	0.70666667	0.71666667	0.56666667
		KNN	0.647619	0.72	0.66666667	0.63333333
		SVM	0.614286	0.67333333	0.66666667	0.56666667
		DT	0.580952	0.65333333	0.6	0.56666667
		AdaBoost	0.580952	0.60333333	0.66666667	0.5
	5 Index(['velocity_std', 'velocity_x_std', 'alphas_velocity_mean', 'snatch_std', 'azimuth_diff_min'],	LogReg	0.680952	0.72333333	0.76666667	0.56666667
		RF	0.647619	0.70666667	0.71666667	0.56666667
		KNN	0.67619	0.72333333	0.76666667	0.53333333
		SVM	0.614286	0.67333333	0.66666667	0.56666667
		DT	0.614286	0.67333333	0.66666667	0.56666667
		AdaBoost	0.580952	0.60333333	0.66666667	0.5
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 4), ('position: ', 'end')]	2 Index(['velocity_x_min', 'acceleration_y_max'], dtype='object')	LogReg	0.785714	0.78333333	0.83333333	0.73333333
		RF	0.685714	0.71	0.71666667	0.63333333
		KNN	0.780952	0.75333333	0.88333333	0.63333333
		SVM	0.747619	0.72333333	0.88333333	0.56666667
		DT	0.752381	0.82666667	0.7	0.8
		AdaBoost	0.747619	0.78666667	0.76666667	0.7
	3 Index(['velocity_x_min', 'acceleration_y_max', 'jerk_y_mass'], dtype='object')	LogReg	0.72381	0.78333333	0.73333333	0.73333333
		RF	0.680952	0.63333333	0.65	0.7
		KNN	0.752381	0.76666667	0.83333333	0.66666667
		SVM	0.690476	0.71333333	0.78333333	0.6
		DT	0.62381	0.68333333	0.46666667	0.8
		AdaBoost	0.680952	0.75	0.53333333	0.83333333

	4 Index(['velocity_x_min', 'acceleration_y_max', 'jerk_y_mass', 'jerk_y_max'], dtype='object')	LogReg	0.842857	0.93333333	0.78333333	0.93333333
		RF	0.714286	0.68333333	0.65	0.76666667
		KNN	0.747619	0.81666667	0.76666667	0.7
		SVM	0.685714	0.70333333	0.78333333	0.56666667
		DT	0.590476	0.58333333	0.46666667	0.73333333
		AdaBoost	0.67619	0.73333333	0.53333333	0.83333333
	5 Index(['velocity_x_min', 'acceleration_y_max', 'jerk_y_mass', 'jerk_y_max', 'pressure_diff_max'],	LogReg	0.747619	0.88333333	0.68333333	0.86666667
		RF	0.657143	0.7	0.68333333	0.63333333
		KNN	0.680952	0.74	0.78333333	0.6
		SVM	0.685714	0.74666667	0.73333333	0.66666667
		DT	0.590476	0.58333333	0.46666667	0.73333333
		AdaBoost	0.67619	0.64333333	0.63333333	0.7
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 5), ('position: ', 'end')]	2 Index(['jerk_x_std', 'shake_std'], dtype='object')	LogReg	0.671429	0.69333333	0.66666667	0.6
		RF	0.709524	0.66333333	0.86666667	0.5
		KNN	0.609524	0.59333333	0.66666667	0.5
		SVM	0.704762	0.72	0.8	0.56666667
		DT	0.709524	0.66333333	0.86666667	0.5
		AdaBoost	0.771429	0.73	0.86666667	0.66666667
	3 Index(['jerk_x_std', 'yaw_mean', 'shake_std'], dtype='object')	LogReg	0.709524	0.73333333	0.68333333	0.73333333
		RF	0.709524	0.70333333	0.81666667	0.6
		KNN	0.647619	0.71428571	0.61666667	0.66666667
		SVM	0.709524	0.73333333	0.81666667	0.6
		DT	0.77619	0.75333333	0.86666667	0.66666667
		AdaBoost	0.680952	0.69	0.76666667	0.6
	4 Index(['jerk_x_std', 'yaw_mean', 'shake_std', 'azimuth_diff_max'], dtype='object')	LogReg	0.580952	0.64761905	0.56666667	0.6
		RF	0.709524	0.70333333	0.81666667	0.6
		KNN	0.547619	0.61428571	0.56666667	0.53333333
		SVM	0.580952	0.64761905	0.7	0.46666667
		DT	0.709524	0.75333333	0.73333333	0.66666667
		AdaBoost	0.680952	0.69	0.76666667	0.6
	5 Index(['jerk_x_std', 'yaw_mean', 'shake_std', 'altitude_diff_std', 'azimuth_diff_max'], dtype='object')	LogReg	0.709524	0.76666667	0.7	0.73333333
		RF	0.742857	0.75333333	0.81666667	0.66666667
		KNN	0.67619	0.73333333	0.7	0.66666667
		SVM	0.709524	0.73333333	0.81666667	0.6
		DT	0.742857	0.70333333	0.86666667	0.6
		AdaBoost	0.714286	0.74	0.76666667	0.66666667

Appendix C

Results of the fully segmented PL copy task

settings	features	model	accuracy	precision	sensitivity	specificity
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'start')]	2 Index(['acceleration_std', 'pressure_diff_mean'], dtype='object')	LogReg	0.771429	0.76333333	0.88333333	0.66666667
		RF	0.742857	0.74	0.9	0.6
		KNN	0.8	0.79	0.95	0.66666667
		SVM	0.771429	0.75	0.95	0.6
		DT	0.742857	0.75333333	0.83333333	0.66666667
		AdaBoost	0.742857	0.75333333	0.83333333	0.66666667
	3 Index(['acceleration_std', 'pressure_diff_mean', 'altitude_diff_max'], dtype='object')	LogReg	0.77619	0.82	0.83333333	0.73333333
		RF	0.77619	0.82	0.83333333	0.73333333
		KNN	0.77619	0.82	0.83333333	0.73333333
		SVM	0.742857	0.82	0.78333333	0.73333333
		DT	0.747619	0.80333333	0.78333333	0.73333333
		AdaBoost	0.747619	0.80333333	0.78333333	0.73333333
	4 Index(['velocity_x_max', 'acceleration_std', 'pressure_diff_mean', 'altitude_diff_max'], dtype='object')	LogReg	0.742857	0.75333333	0.83333333	0.66666667
		RF	0.742857	0.75333333	0.83333333	0.66666667
		KNN	0.709524	0.72	0.9	0.53333333
		SVM	0.742857	0.77	0.85	0.66666667
		DT	0.77619	0.82	0.83333333	0.73333333
		AdaBoost	0.77619	0.82	0.83333333	0.73333333
	5 Index(['velocity_x_max', 'acceleration_std', 'pressure_diff_mean', 'altitude_diff_max', 'azimuth_velocity_max'],	LogReg	0.742857	0.75333333	0.83333333	0.66666667
		RF	0.77619	0.77	0.9	0.66666667
		KNN	0.714286	0.71	0.9	0.53333333
		SVM	0.77619	0.82	0.85	0.73333333
		DT	0.780952	0.80333333	0.85	0.73333333
		AdaBoost	0.77619	0.77	0.9	0.66666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'start')]	2 Index(['jerk_x_std', 'tug_mass'], dtype='object')	LogReg	0.709524	0.75333333	0.78333333	0.63333333
		RF	0.714286	0.75333333	0.78333333	0.63333333
		KNN	0.747619	0.80333333	0.78333333	0.7
		SVM	0.747619	0.80333333	0.78333333	0.7
		DT	0.77619	0.83	0.83333333	0.7
		AdaBoost	0.714286	0.75333333	0.78333333	0.63333333
	3 Index(['jerk_std', 'jerk_x_std', 'tug_mass'], dtype='object')	LogReg	0.652381	0.65333333	0.73333333	0.56666667
		RF	0.685714	0.66	0.83333333	0.5
		KNN	0.747619	0.83333333	0.73333333	0.76666667
		SVM	0.747619	0.83333333	0.73333333	0.76666667
		DT	0.742857	0.75	0.9	0.56666667
		AdaBoost	0.67619	0.72333333	0.8	0.56666667
	4 Index(['jerk_std', 'jerk_x_std', 'tug_mass', 'altitude_median'], dtype='object')	LogReg	0.652381	0.65333333	0.73333333	0.56666667
		RF	0.809524	0.84333333	0.83333333	0.76666667
		KNN	0.747619	0.79333333	0.78333333	0.7
		SVM	0.747619	0.83333333	0.73333333	0.76666667
		DT	0.77619	0.8	0.9	0.63333333
		AdaBoost	0.742857	0.79	0.85	0.63333333

	5 Index(['jerk_std', 'velocity_x_max', 'jerk_x_std', 'tug_mass', 'altitude_median'], dtype='object')	LogReg	0.742857	0.78333333	0.78333333	0.7
		RF	0.77619	0.83333333	0.78333333	0.76666667
		KNN	0.809524	0.9	0.78333333	0.83333333
		SVM	0.747619	0.83333333	0.73333333	0.76666667
		DT	0.809524	0.88	0.83333333	0.76666667
		AdaBoost	0.77619	0.82	0.85	0.7
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 3), ('position: ', 'start')]	2 Index(['velocity_std', 'acceleration_y_mean'], dtype='object')	LogReg	0.904762	0.93333333	0.88333333	0.93333333
		RF	0.938095	1	0.88333333	1
		KNN	0.904762	0.93333333	0.88333333	0.93333333
		SVM	0.904762	0.93333333	0.88333333	0.93333333
		DT	0.904762	0.9	0.95	0.86666667
		AdaBoost	0.904762	0.9	0.95	0.86666667
	3 Index(['velocity_std', 'acceleration_mean', 'acceleration_y_mean'], dtype='object')	LogReg	0.87619	0.88333333	0.88333333	0.86666667
		RF	0.87619	0.9	0.88333333	0.86666667
		KNN	0.909524	0.95	0.88333333	0.93333333
		SVM	0.87619	0.88333333	0.88333333	0.86666667
		DT	0.909524	0.9	0.95	0.86666667
		AdaBoost	0.909524	0.9	0.95	0.86666667
	4 Index(['velocity_std', 'acceleration_mean', 'acceleration_y_mean', 'jerk_std'], dtype='object')	LogReg	0.838095	0.85	0.88333333	0.8
		RF	0.938095	0.95	0.95	0.93333333
		KNN	0.871429	0.88333333	0.88333333	0.86666667
		SVM	0.871429	0.88333333	0.88333333	0.86666667
		DT	0.904762	0.9	0.95	0.86666667
		AdaBoost	0.909524	0.9	0.95	0.86666667
	5 Index(['velocity_std', 'acceleration_mean', 'acceleration_y_mean', 'jerk_std', 'jerk_y_std'],	LogReg	0.871429	0.88333333	0.88333333	0.86666667
		RF	0.904762	0.95	0.88333333	0.93333333
		KNN	0.838095	0.85	0.88333333	0.8
		SVM	0.871429	0.88333333	0.88333333	0.86666667
		DT	0.909524	0.9	0.95	0.86666667
		AdaBoost	0.909524	0.9	0.95	0.86666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 4), ('position: ', 'start')]	2 Index(['acceleration_mean', 'acceleration_std'], dtype='object')	LogReg	0.742857	0.87	0.71666667	0.73333333
		RF	0.77619	0.88333333	0.76666667	0.73333333
		KNN	0.742857	0.87	0.71666667	0.73333333
		SVM	0.709524	0.84	0.71666667	0.66666667
		DT	0.742857	0.81666667	0.76666667	0.66666667
		AdaBoost	0.709524	0.68333333	0.63333333	0.73333333
	3 Index(['acceleration_mean', 'acceleration_std', 'slopes_mass'], dtype='object')	LogReg	0.742857	0.87	0.71666667	0.73333333
		RF	0.809524	0.88333333	0.83333333	0.73333333
		KNN	0.742857	0.87	0.71666667	0.73333333
		SVM	0.742857	0.87	0.71666667	0.73333333
		DT	0.842857	0.88333333	0.9	0.73333333
		AdaBoost	0.742857	0.88333333	0.7	0.73333333
	4 Index(['acceleration_mean', 'acceleration_std', 'slopes_mass', 'phi_angle_jerk_std'],	LogReg	0.742857	0.87	0.71666667	0.73333333
		RF	0.809524	0.88333333	0.83333333	0.73333333
		KNN	0.838095	0.93333333	0.81666667	0.8
		SVM	0.771429	0.87	0.76666667	0.73333333
		DT	0.742857	0.78666667	0.83333333	0.6

	dtype='object')	AdaBoost	0.77619	0.88333333	0.76666667	0.73333333
	5	LogReg	0.742857	0.87	0.71666667	0.73333333
	Index(['acceleration_mea	RF	0.809524	0.88333333	0.83333333	0.73333333
	n', 'acceleration_std',	KNN	0.77619	0.82	0.83333333	0.66666667
	'jerk_y_min',	SVM	0.804762	0.88333333	0.81666667	0.73333333
	'slopes_mass',	DT	0.742857	0.78666667	0.83333333	0.6
	'phi_angle_jerk_std'],	AdaBoost	0.742857	0.8	0.76666667	0.66666667
[('scaler: ',	2	LogReg	0.838095	0.83	0.95	0.7
StandardScaler()),	Index(['acceleration_y_m	RF	0.809524	0.83	0.9	0.7
('folds: ', 5), ('type:	ean', 'phi_angle_mean'],	KNN	0.838095	0.83	0.95	0.7
', 5), ('position: ',	dtype='object')	SVM	0.87619	0.91	0.9	0.83333333
'start')]		DT	0.842857	0.86	0.9	0.76666667
		AdaBoost	0.742857	0.83	0.76666667	0.7
	3	LogReg	0.814286	0.83333333	0.85	0.76666667
	Index(['acceleration_y_m	RF	0.809524	0.83	0.9	0.7
	ean', 'phi_angle_mean',	KNN	0.842857	0.85	0.9	0.76666667
	'pop_mass'],	SVM	0.809524	0.83333333	0.83333333	0.76666667
	dtype='object')	DT	0.809524	0.81	0.9	0.7
		AdaBoost	0.742857	0.77	0.85	0.63333333
	4	LogReg	0.87619	0.91	0.9	0.83333333
	Index(['velocity_x_std',	RF	0.809524	0.83	0.9	0.7
	'acceleration_y_mean',	KNN	0.842857	0.85	0.9	0.76666667
	'phi_angle_mean',	SVM	0.809524	0.83333333	0.83333333	0.76666667
	'pop_mass'],	DT	0.742857	0.75	0.9	0.56666667
	dtype='object')	AdaBoost	0.742857	0.77	0.85	0.63333333
	5	LogReg	0.809524	0.83	0.9	0.7
	Index(['velocity_x_std',	RF	0.809524	0.83	0.9	0.7
	'velocity_y_mean',	KNN	0.809524	0.78	0.95	0.63333333
	'acceleration_y_mean',	SVM	0.809524	0.88333333	0.78333333	0.83333333
	'phi_angle_mean',	DT	0.77619	0.78	0.9	0.63333333
	'pop_mass'],	AdaBoost	0.742857	0.77	0.85	0.63333333
[('scaler: ',	2	LogReg	0.642857	0.63	0.75	0.5
StandardScaler()),	Index(['acceleration_mea	RF	0.671429	0.67666667	0.81666667	0.5
('folds: ', 5), ('type:	n',	KNN	0.671429	0.70333333	0.76666667	0.56666667
', 1), ('position: ',	'acceleration_y_mean'],	SVM	0.704762	0.73333333	0.76666667	0.63333333
'middle')]	dtype='object')	DT	0.638095	0.66666667	0.7	0.56666667
		AdaBoost	0.704762	0.72333333	0.81666667	0.56666667
	3	LogReg	0.704762	0.71333333	0.81666667	0.56666667
	Index(['acceleration_mea	RF	0.671429	0.69333333	0.81666667	0.5
	n',	KNN	0.771429	0.81333333	0.81666667	0.7
	'acceleration_x_mean',	SVM	0.771429	0.81333333	0.81666667	0.7
	'acceleration_y_mean'],	DT	0.738095	0.74	0.88333333	0.56666667
	dtype='object')	AdaBoost	0.671429	0.71	0.68333333	0.63333333
	4	LogReg	0.671429	0.70333333	0.7	0.63333333
	Index(['acceleration_mea	RF	0.704762	0.71333333	0.88333333	0.5
	n',	KNN	0.609524	0.68333333	0.71666667	0.5
	'acceleration_x_mean',	SVM	0.609524	0.67	0.65	0.56666667

	'acceleration_y_mean', 'pressure_diff_min'],	DT	0.542857	0.56666667	0.7	0.36666667
		AdaBoost	0.704762	0.71333333	0.81666667	0.56666667
	5 Index(['acceleration_mean', 'acceleration_x_mean', 'acceleration_y_mean', 'pressure_diff_min',	LogReg	0.680952	0.68	0.76666667	0.56666667
		RF	0.738095	0.73333333	0.88333333	0.56666667
		KNN	0.614286	0.63666667	0.71666667	0.5
		SVM	0.642857	0.70333333	0.65	0.63333333
		DT	0.609524	0.61666667	0.7	0.5
		AdaBoost	0.709524	0.68	0.88333333	0.5
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'middle')]	2 Index(['velocity_x_std', 'pressure_median'], dtype='object')	LogReg	0.671429	0.76	0.7	0.66666667
		RF	0.804762	0.88333333	0.78333333	0.86666667
		KNN	0.647619	0.76666667	0.6	0.73333333
		SVM	0.738095	0.85	0.71666667	0.8
		DT	0.738095	0.85	0.71666667	0.8
		AdaBoost	0.704762	0.82	0.73333333	0.73333333
	3 Index(['velocity_x_mean', 'velocity_x_std', 'pressure_median'], dtype='object')	LogReg	0.771429	0.79333333	0.8	0.73333333
		RF	0.771429	0.85	0.78333333	0.8
		KNN	0.704762	0.78333333	0.7	0.7
		SVM	0.804762	0.88333333	0.76666667	0.86666667
		DT	0.67619	0.75333333	0.66666667	0.7
		AdaBoost	0.642857	0.82	0.53333333	0.8
	4 Index(['velocity_x_mean', 'velocity_x_std', 'pressure_median', 'azimuth_velocity_min'],	LogReg	0.704762	0.69333333	0.81666667	0.56666667
		RF	0.77619	0.88333333	0.73333333	0.86666667
		KNN	0.671429	0.71666667	0.7	0.63333333
		SVM	0.738095	0.76666667	0.76666667	0.7
		DT	0.742857	0.81666667	0.73333333	0.76666667
		AdaBoost	0.67619	0.85333333	0.6	0.8
	5 Index(['velocity_x_mean', 'velocity_x_std', 'pressure_median', 'snatch_mean',	LogReg	0.738095	0.72666667	0.81666667	0.63333333
		RF	0.77619	0.88333333	0.73333333	0.86666667
		KNN	0.709524	0.76666667	0.71666667	0.7
		SVM	0.742857	0.77666667	0.76666667	0.7
		DT	0.742857	0.81666667	0.73333333	0.76666667
		AdaBoost	0.67619	0.85333333	0.6	0.8
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 3), ('position: ', 'middle')]	2 Index(['velocity_y_min', 'jerk_mean'], dtype='object')	LogReg	0.809524	0.8	0.9	0.7
		RF	0.809524	0.8	0.9	0.7
		KNN	0.809524	0.78333333	0.95	0.63333333
		SVM	0.809524	0.8	0.9	0.7
		DT	0.747619	0.72	0.9	0.56666667
		AdaBoost	0.747619	0.75333333	0.85	0.63333333
	3 Index(['velocity_y_min', 'acceleration_x_max', 'jerk_mean'], dtype='object')	LogReg	0.842857	0.85	0.9	0.76666667
		RF	0.842857	0.85	0.9	0.76666667
		KNN	0.842857	0.85	0.9	0.76666667
		SVM	0.842857	0.85	0.9	0.76666667
		DT	0.752381	0.75	0.9	0.56666667
		AdaBoost	0.814286	0.82	0.9	0.7
	4 Index(['velocity_y_min', 'acceleration_x_max',	LogReg	0.842857	0.85	0.9	0.76666667
		RF	0.842857	0.85	0.9	0.76666667
		KNN	0.842857	0.85	0.9	0.76666667

	'jerk_mean', 'yaw_mean'], dtype='object')	SVM	0.77619	0.81666667	0.78333333	0.76666667
		DT	0.680952	0.70333333	0.85	0.5
		AdaBoost	0.714286	0.73666667	0.78333333	0.63333333
	5 Index(['velocity_y_min', 'acceleration_max', 'acceleration_x_max', 'jerk_mean', 'yaw_mean'],	LogReg	0.87619	0.86	0.95	0.76666667
		RF	0.842857	0.85	0.9	0.76666667
		KNN	0.909524	0.91	0.95	0.83333333
		SVM	0.809524	0.83333333	0.83333333	0.76666667
		DT	0.714286	0.72	0.9	0.5
		AdaBoost	0.714286	0.73666667	0.78333333	0.63333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 4), ('position: ', 'middle')]	2 Index(['acceleration_median', 'jerk_mean'], dtype='object')	LogReg	0.814286	0.83333333	0.83333333	0.8
		RF	0.780952	0.81333333	0.83333333	0.7
		KNN	0.77619	0.75333333	0.88333333	0.66666667
		SVM	0.747619	0.81333333	0.76666667	0.7
		DT	0.747619	0.83	0.78333333	0.7
		AdaBoost	0.809524	0.85	0.81666667	0.8
	3 Index(['acceleration_median', 'acceleration_min', 'jerk_mean'], dtype='object')	LogReg	0.742857	0.71333333	0.88333333	0.56666667
		RF	0.77619	0.81	0.81666667	0.7
		KNN	0.709524	0.72333333	0.83333333	0.6
		SVM	0.709524	0.69666667	0.81666667	0.56666667
		DT	0.780952	0.79333333	0.83333333	0.7
		AdaBoost	0.814286	0.87	0.83333333	0.76666667
	4 Index(['acceleration_median', 'acceleration_min', 'jerk_mean', 'shake_max'], dtype='object')	LogReg	0.709524	0.71666667	0.78333333	0.63333333
		RF	0.809524	0.83	0.88333333	0.7
		KNN	0.642857	0.66333333	0.76666667	0.5
		SVM	0.642857	0.65	0.7	0.56666667
		DT	0.814286	0.91	0.78333333	0.83333333
		AdaBoost	0.814286	0.87	0.83333333	0.76666667
	5 Index(['acceleration_median', 'acceleration_min', 'jerk_mean', 'jerk_y_std', 'shake_max'], dtype='object')	LogReg	0.709524	0.70333333	0.83333333	0.56666667
		RF	0.842857	0.91	0.83333333	0.83333333
		KNN	0.709524	0.70333333	0.81666667	0.6
		SVM	0.67619	0.68333333	0.7	0.63333333
		DT	0.847619	0.95	0.78333333	0.93333333
		AdaBoost	0.747619	0.80333333	0.76666667	0.7
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 5), ('position: ', 'middle')]	2 Index(['velocity_x_mean', 'acceleration_x_mean'], dtype='object')	LogReg	0.809524	0.82	0.9	0.73333333
		RF	0.780952	0.85333333	0.78333333	0.8
		KNN	0.780952	0.78666667	0.83333333	0.73333333
		SVM	0.747619	0.73666667	0.83333333	0.66666667
		DT	0.680952	0.68666667	0.78333333	0.56666667
		AdaBoost	0.747619	0.83333333	0.71666667	0.8
	3 Index(['velocity_x_mean', 'acceleration_x_mean', 'jerk_x_std'], dtype='object')	LogReg	0.77619	0.80333333	0.83333333	0.73333333
		RF	0.780952	0.85333333	0.78333333	0.8
		KNN	0.780952	0.78666667	0.83333333	0.73333333
		SVM	0.814286	0.85333333	0.83333333	0.8
		DT	0.714286	0.73666667	0.78333333	0.66666667
		AdaBoost	0.680952	0.75333333	0.73333333	0.66666667
	4 Index(['velocity_x_mean'	LogReg	0.87619	0.9	0.88333333	0.86666667
		RF	0.780952	0.85333333	0.78333333	0.8

	, 'acceleration_x_mean', 'jerk_x_std', 'pressure_mean'], dtype='object')	KNN	0.87619	0.85	0.95	0.8
		SVM	0.87619	0.9	0.88333333	0.86666667
		DT	0.714286	0.75333333	0.78333333	0.63333333
		AdaBoost	0.680952	0.76666667	0.66666667	0.7
	5 Index(['velocity_x_mean', 'acceleration_x_mean', 'jerk_x_std', 'pressure_mean', 'pressure_diff_max'],	LogReg	0.842857	0.85	0.9	0.76666667
		RF	0.814286	0.85333333	0.83333333	0.8
		KNN	0.87619	0.85	0.95	0.8
		SVM	0.909524	0.9	0.95	0.86666667
		DT	0.719048	0.77	0.78333333	0.63333333
		AdaBoost	0.619048	0.65333333	0.73333333	0.5
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'end')]	2 Index(['acceleration_y_std', 'snatch_mean'], dtype='object')	LogReg	0.933333	0.92	1	0.86666667
		RF	0.77619	0.92	0.7	0.86666667
		KNN	0.933333	0.95	0.93333333	0.93333333
		SVM	0.904762	1	0.81666667	1
		DT	0.742857	0.77333333	0.83333333	0.66666667
		AdaBoost	0.77619	0.92	0.7	0.86666667
	3 Index(['acceleration_y_std', 'jerk_mean', 'snatch_mean'], dtype='object')	LogReg	0.938095	0.95	0.95	0.93333333
		RF	0.838095	0.95	0.75	0.93333333
		KNN	0.871429	0.9	0.88333333	0.86666667
		SVM	0.904762	0.95	0.88333333	0.93333333
		DT	0.809524	0.92	0.76666667	0.86666667
		AdaBoost	0.809524	0.92	0.76666667	0.86666667
	4 Index(['acceleration_y_std', 'jerk_mean', 'pressure_mean', 'snatch_mean'], dtype='object')	LogReg	0.87619	0.85	0.95	0.8
		RF	0.804762	0.92	0.75	0.86666667
		KNN	0.842857	0.82	0.95	0.73333333
		SVM	0.909524	0.9	0.95	0.86666667
		DT	0.809524	0.92	0.76666667	0.86666667
		AdaBoost	0.77619	0.92	0.7	0.86666667
	5 Index(['acceleration_y_std', 'jerk_mean', 'pressure_mean', 'snatch_mean', 'shake_mean'],	LogReg	0.938095	0.91	1	0.86666667
		RF	0.804762	0.92	0.75	0.86666667
		KNN	0.938095	0.91	1	0.86666667
		SVM	0.909524	0.95	0.88333333	0.93333333
		DT	0.77619	0.85333333	0.76666667	0.8
		AdaBoost	0.77619	0.92	0.7	0.86666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'end')]	2 Index(['velocity_x_std', 'azimuth_jerk_mean'], dtype='object')	LogReg	0.72	0.80333333	0.76666667	0.66666667
		RF	0.653333	0.77333333	0.7	0.6
		KNN	0.686667	0.78666667	0.7	0.66666667
		SVM	0.653333	0.78666667	0.63333333	0.66666667
		DT	0.72	0.77333333	0.83333333	0.6
		AdaBoost	0.653333	0.82	0.63333333	0.66666667
	3 Index(['velocity_x_mean', 'velocity_x_std', 'azimuth_jerk_mean'], dtype='object')	LogReg	0.82	0.92	0.83333333	0.8
		RF	0.786667	0.87	0.83333333	0.73333333
		KNN	0.786667	0.87	0.83333333	0.73333333
		SVM	0.753333	0.87	0.76666667	0.73333333
		DT	0.753333	0.80333333	0.83333333	0.66666667
		AdaBoost	0.786667	0.87	0.83333333	0.73333333
	4	LogReg	0.753333	0.85333333	0.76666667	0.73333333

	Index(['velocity_x_mean', 'velocity_x_std', 'acceleration_max', 'azimuth_jerk_mean'],	RF	0.753333	0.87	0.76666667	0.73333333
		KNN	0.753333	0.80333333	0.83333333	0.66666667
		SVM	0.686667	0.80333333	0.7	0.66666667
		DT	0.72	0.80333333	0.78333333	0.66666667
		AdaBoost	0.753333	0.87	0.76666667	0.73333333
	5 Index(['velocity_x_mean', 'velocity_x_std', 'acceleration_max', 'acceleration_x_max', 'azimuth_jerk_mean'],	LogReg	0.686667	0.80333333	0.7	0.66666667
		RF	0.72	0.87	0.71666667	0.73333333
		KNN	0.753333	0.87	0.76666667	0.73333333
		SVM	0.686667	0.80333333	0.7	0.66666667
		DT	0.753333	0.80333333	0.83333333	0.66666667
		AdaBoost	0.72	0.87	0.7	0.73333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 3), ('position: ', 'end')]	2 Index(['acceleration_x_mean', 'jerk_x_std'], dtype='object')	LogReg	0.614286	0.73333333	0.65	0.6
		RF	0.685714	0.76666667	0.65	0.73333333
		KNN	0.614286	0.68666667	0.65	0.6
		SVM	0.647619	0.8	0.65	0.66666667
		DT	0.685714	0.76666667	0.65	0.73333333
		AdaBoost	0.652381	0.75	0.73333333	0.6
	3 Index(['acceleration_x_mean', 'jerk_x_std', 'phi_angle_mass'], dtype='object')	LogReg	0.647619	0.77333333	0.65	0.66666667
		RF	0.680952	0.70666667	0.76666667	0.6
		KNN	0.552381	0.62	0.66666667	0.46666667
		SVM	0.647619	0.77333333	0.65	0.66666667
		DT	0.685714	0.7	0.71666667	0.66666667
		AdaBoost	0.585714	0.63333333	0.65	0.53333333
	4 Index(['velocity_x_mean', 'acceleration_x_mean', 'jerk_x_std', 'phi_angle_mass'], dtype='object')	LogReg	0.647619	0.77333333	0.65	0.66666667
		RF	0.685714	0.75333333	0.73333333	0.66666667
		KNN	0.547619	0.64	0.66666667	0.46666667
		SVM	0.547619	0.64	0.6	0.53333333
		DT	0.585714	0.61666667	0.6	0.56666667
		AdaBoost	0.519048	0.63333333	0.55	0.53333333
	5 Index(['velocity_x_mean', 'acceleration_x_mean', 'jerk_x_std', 'phi_angle_mass', 'yaw_mean'],	LogReg	0.614286	0.67333333	0.65	0.6
		RF	0.647619	0.70666667	0.71666667	0.6
		KNN	0.580952	0.68666667	0.61666667	0.6
		SVM	0.614286	0.71666667	0.61666667	0.66666667
		DT	0.552381	0.63333333	0.53333333	0.6
		AdaBoost	0.485714	0.6	0.41666667	0.6
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 4), ('position: ', 'end')]	2 Index(['velocity_y_std', 'acceleration_y_std'], dtype='object')	LogReg	0.714286	0.75	0.7	0.73333333
		RF	0.714286	0.78333333	0.65	0.8
		KNN	0.714286	0.75	0.7	0.73333333
		SVM	0.714286	0.78333333	0.65	0.8
		DT	0.647619	0.66761905	0.75	0.53333333
		AdaBoost	0.680952	0.71666667	0.65	0.73333333
	3 Index(['velocity_y_std', 'acceleration_y_std', 'azimuth_velocity_mean'], dtype='object')	LogReg	0.647619	0.66666667	0.75	0.5
		RF	0.747619	0.77	0.75	0.73333333
		KNN	0.680952	0.69666667	0.75	0.56666667
		SVM	0.685714	0.75	0.65	0.73333333
		DT	0.780952	0.88333333	0.71666667	0.86666667
		AdaBoost	0.714286	0.72666667	0.7	0.7

	4	LogReg	0.652381	0.67666667	0.7	0.56666667
	Index(['velocity_y_std',	RF	0.714286	0.75	0.7	0.73333333
	'acceleration_median',	KNN	0.652381	0.67666667	0.7	0.56666667
	'acceleration_y_std',	SVM	0.719048	0.74333333	0.76666667	0.63333333
		DT	0.747619	0.78333333	0.78333333	0.73333333
	'azimuth_velocity_mean'	AdaBoost	0.680952	0.76666667	0.58333333	0.8
	5	LogReg	0.680952	0.69666667	0.75	0.56666667
	Index(['velocity_y_std',	RF	0.714286	0.75	0.7	0.73333333
	'acceleration_median',	KNN	0.680952	0.69666667	0.75	0.56666667
	'acceleration_y_std',	SVM	0.719048	0.74333333	0.76666667	0.63333333
	'jerk_y_std',	DT	0.714286	0.76666667	0.71666667	0.7
	'azimuth_velocity_mean'	AdaBoost	0.647619	0.66666667	0.63333333	0.66666667
[('scaler: ',	2	LogReg	0.766667	0.80333333	0.81666667	0.7
StandardScaler()),	Index(['jerk_x_median',	RF	0.666667	0.7	0.68333333	0.63333333
('folds: ', 5), ('type:	'tug_max'],	KNN	0.7	0.63	0.73333333	0.63333333
', 5), ('position: ',	dtype='object')	SVM	0.7	0.63	0.73333333	0.63333333
'end')]		DT	0.7	0.73333333	0.75	0.63333333
		AdaBoost	0.7	0.73333333	0.75	0.63333333
	3	LogReg	0.766667	0.81666667	0.75	0.76666667
	Index(['jerk_x_median',	RF	0.733333	0.83333333	0.68333333	0.76666667
	'alphas_median',	KNN	0.766667	0.77666667	0.8	0.7
	'tug_max'],	SVM	0.8	0.79333333	0.86666667	0.7
	dtype='object')	DT	0.666667	0.63333333	0.61666667	0.7
		AdaBoost	0.566667	0.55	0.55	0.56666667
	4	LogReg	0.766667	0.76666667	0.81666667	0.7
	Index(['jerk_x_median',	RF	0.733333	0.73666667	0.81666667	0.63333333
	'alphas_median',	KNN	0.866667	0.86	0.93333333	0.76666667
	'tug_max',	SVM	0.8	0.81666667	0.81666667	0.76666667
	'altitude_diff_mean'],	DT	0.7	0.68666667	0.81666667	0.56666667
	dtype='object')	AdaBoost	0.7	0.68666667	0.81666667	0.56666667
	5	LogReg	0.833333	0.83333333	0.88333333	0.76666667
	Index(['duration',	RF	0.766667	0.83333333	0.75	0.76666667
	'jerk_x_median',	KNN	0.666667	0.65333333	0.88333333	0.43333333
	'alphas_median',	SVM	0.866667	0.84333333	0.93333333	0.76666667
	'tug_max',	DT	0.766667	0.77	0.88333333	0.63333333
	'altitude_diff_mean'],	AdaBoost	0.8	0.76333333	0.93333333	0.63333333

Appendix D

Results of the fully segmented PL continue task

settings	features	model	accuracy	precision	sensitivity	specificity
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'start')]	2 Index(['velocity_y_std', 'alphas_velocity_std'], dtype='object')	LogReg	0.561905	0.62	0.66666667	0.46666667
		RF	0.595238	0.71333333	0.61666667	0.6
		KNN	0.628571	0.67	0.78333333	0.46666667
		SVM	0.595238	0.66666667	0.66666667	0.53333333
		DT	0.595238	0.66333333	0.66666667	0.53333333
		AdaBoost	0.595238	0.66333333	0.66666667	0.53333333
	3 Index(['velocity_y_std', 'alphas_velocity_std', 'altitude_diff_min'], dtype='object')	LogReg	0.533333	0.55	0.6	0.46666667
		RF	0.6	0.65	0.61666667	0.6
		KNN	0.561905	0.58666667	0.71666667	0.36666667
		SVM	0.566667	0.55	0.6	0.53333333
		DT	0.533333	0.56666667	0.48333333	0.6
		AdaBoost	0.566667	0.63333333	0.61666667	0.53333333
	4 Index(['velocity_y_std', 'alphas_velocity_std', 'altitude_diff_min', 'altitude_velocity_mass'],	LogReg	0.590476	0.60333333	0.6	0.6
		RF	0.590476	0.68666667	0.61666667	0.6
		KNN	0.595238	0.62666667	0.71666667	0.46666667
		SVM	0.566667	0.55	0.6	0.53333333
		DT	0.566667	0.58333333	0.55	0.6
		AdaBoost	0.557143	0.63333333	0.6	0.53333333
	5 Index(['velocity_mean', 'velocity_y_std', 'alphas_velocity_std', 'altitude_diff_min', 'altitude_velocity_mass'],	LogReg	0.528571	0.57	0.55	0.53333333
		RF	0.590476	0.68666667	0.61666667	0.6
		KNN	0.628571	0.66	0.78333333	0.46666667
		SVM	0.533333	0.55	0.55	0.53333333
		DT	0.561905	0.62	0.55	0.6
		AdaBoost	0.561905	0.61666667	0.6	0.53333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'start')]	2 Index(['velocity_mean', 'acceleration_median'], dtype='object')	LogReg	0.733333	0.76	0.88333333	0.6
		RF	0.733333	0.80333333	0.76666667	0.73333333
		KNN	0.8	0.84	0.88333333	0.73333333
		SVM	0.866667	0.9	0.88333333	0.86666667
		DT	0.666667	0.70333333	0.7	0.66666667
		AdaBoost	0.7	0.80333333	0.7	0.73333333
	3 Index(['velocity_mean', 'acceleration_median', 'pressure_max'], dtype='object')	LogReg	0.766667	0.76	0.95	0.6
		RF	0.766667	0.82	0.83333333	0.73333333
		KNN	0.733333	0.76	0.88333333	0.6
		SVM	0.833333	0.85	0.88333333	0.8
		DT	0.766667	0.82	0.83333333	0.73333333
		AdaBoost	0.766667	0.82	0.83333333	0.73333333
	4 Index(['velocity_mean', 'velocity_x_std', 'acceleration_median', 'pressure_max'], dtype='object')	LogReg	0.733333	0.76	0.88333333	0.6
		RF	0.766667	0.82	0.83333333	0.73333333
		KNN	0.733333	0.76	0.88333333	0.6
		SVM	0.8	0.82	0.88333333	0.73333333
		DT	0.766667	0.82	0.83333333	0.73333333
		AdaBoost	0.766667	0.82	0.83333333	0.73333333

	5 Index(['velocity_mean', 'velocity_x_std', 'acceleration_median', 'acceleration_x_median',	LogReg	0.733333	0.76	0.88333333	0.6
		RF	0.8	0.87	0.83333333	0.8
		KNN	0.766667	0.79	0.88333333	0.66666667
		SVM	0.8	0.84	0.88333333	0.73333333
		DT	0.833333	0.9	0.83333333	0.86666667
		AdaBoost	0.766667	0.82	0.83333333	0.73333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 3), ('position: ', 'start')]	2 Index(['velocity_x_std', 'pressure_mean'], dtype='object')	LogReg	0.74	0.79	0.86666667	0.6
		RF	0.673333	0.76	0.81666667	0.53333333
		KNN	0.6	0.54	0.75	0.46666667
		SVM	0.633333	0.56	0.75	0.53333333
		DT	0.673333	0.77	0.75	0.6
		AdaBoost	0.673333	0.77333333	0.75	0.6
	3 Index(['velocity_x_mean', 'velocity_x_std', 'pressure_mean'], dtype='object')	LogReg	0.84	0.9	0.86666667	0.8
		RF	0.666667	0.8	0.7	0.66666667
		KNN	0.706667	0.77	0.81666667	0.6
		SVM	0.74	0.9	0.7	0.8
		DT	0.706667	0.82	0.76666667	0.66666667
		AdaBoost	0.74	0.87	0.76666667	0.73333333
	4 Index(['velocity_x_mean', 'velocity_x_std', 'pressure_mean', 'azimuth_jerk_min'], dtype='object')	LogReg	0.766667	0.86	0.8	0.7
		RF	0.633333	0.77	0.7	0.6
		KNN	0.606667	0.67	0.81666667	0.36666667
		SVM	0.666667	0.85	0.63333333	0.73333333
		DT	0.706667	0.82	0.76666667	0.66666667
		AdaBoost	0.64	0.78666667	0.63333333	0.66666667
	5 Index(['velocity_x_mean', 'velocity_x_max', 'velocity_x_std', 'pressure_mean', 'azimuth_jerk_min'],	LogReg	0.7	0.76	0.8	0.56666667
		RF	0.6	0.74	0.7	0.53333333
		KNN	0.566667	0.67	0.75	0.36666667
		SVM	0.666667	0.78333333	0.68333333	0.66666667
		DT	0.606667	0.73666667	0.63333333	0.6
		AdaBoost	0.68	0.76666667	0.68333333	0.7
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 4), ('position: ', 'start')]	2 Index(['velocity_x_std', 'acceleration_x_std'], dtype='object')	LogReg	0.7	0.74	0.83333333	0.6
		RF	0.666667	0.70666667	0.76666667	0.6
		KNN	0.7	0.74	0.83333333	0.6
		SVM	0.7	0.74	0.83333333	0.6
		DT	0.633333	0.65666667	0.76666667	0.53333333
		AdaBoost	0.633333	0.65666667	0.76666667	0.53333333
	3 Index(['velocity_x_std', 'acceleration_x_std', 'alphas_velocity_max'], dtype='object')	LogReg	0.633333	0.64	0.7	0.6
		RF	0.633333	0.67	0.76666667	0.53333333
		KNN	0.633333	0.64	0.7	0.6
		SVM	0.666667	0.67	0.7	0.66666667
		DT	0.633333	0.67	0.7	0.6
		AdaBoost	0.633333	0.67	0.7	0.6
	4 Index(['velocity_x_std', 'acceleration_x_max', 'acceleration_x_std',	LogReg	0.666667	0.67	0.7	0.66666667
		RF	0.666667	0.72	0.7	0.66666667
		KNN	0.633333	0.65333333	0.63333333	0.66666667
		SVM	0.6	0.60333333	0.63333333	0.6
		DT	0.6	0.65333333	0.63333333	0.6

	'alphas_velocity_max'],	AdaBoost	0.6	0.65333333	0.63333333	0.6
	5	LogReg	0.566667	0.57333333	0.63333333	0.53333333
	Index(['velocity_x_std',	RF	0.666667	0.73666667	0.7	0.66666667
	'acceleration_x_max',	KNN	0.666667	0.69	0.83333333	0.53333333
	'acceleration_x_std',	SVM	0.566667	0.57	0.56666667	0.6
		DT	0.6	0.65333333	0.63333333	0.6
	'alphas_velocity_max',	AdaBoost	0.633333	0.66666667	0.7	0.6
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 5), ('position: ', 'start')]	2	LogReg	0.753333	0.74	0.95	0.5
	Index(['velocity_x_mean'	RF	0.673333	0.77	0.76666667	0.6
	, 'velocity_x_median'],	KNN	0.746667	0.77	0.88333333	0.56666667
	dtype='object')	SVM	0.82	0.82	0.95	0.66666667
		DT	0.633333	0.73666667	0.7	0.6
		AdaBoost	0.7	0.85	0.63333333	0.83333333
	3	LogReg	0.786667	0.77	0.95	0.56666667
	Index(['velocity_x_mean'	RF	0.633333	0.73333333	0.7	0.56666667
	, 'velocity_x_median',	KNN	0.786667	0.77	0.95	0.56666667
	'phi_angle_jerk_mean'],	SVM	0.72	0.72333333	0.9	0.5
	dtype='object')	DT	0.633333	0.73666667	0.7	0.6
		AdaBoost	0.6	0.63666667	0.7	0.5
	4	LogReg	0.82	0.82	0.95	0.63333333
	Index(['velocity_x_mean'	RF	0.82	0.85	0.88333333	0.76666667
	, 'velocity_x_median',	KNN	0.786667	0.77	0.95	0.56666667
	'phi_angle_jerk_mean',	SVM	0.786667	0.77	0.95	0.56666667
	'alphas_mass'],	DT	0.68	0.73666667	0.76666667	0.6
	dtype='object')	AdaBoost	0.753333	0.82	0.83333333	0.7
	5	LogReg	0.746667	0.80333333	0.83333333	0.63333333
	Index(['velocity_x_mean'	RF	0.82	0.85	0.88333333	0.76666667
	, 'velocity_x_median',	KNN	0.713333	0.75333333	0.83333333	0.56666667
	'phi_angle_jerk_mean',	SVM	0.746667	0.80333333	0.83333333	0.63333333
	'alphas_mass',	DT	0.786667	0.82	0.88333333	0.7
	'pressure_median'],	AdaBoost	0.64	0.72333333	0.78333333	0.5
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'middle')]	2	LogReg	0.933333	0.95	0.95	0.93333333
	Index(['velocity_y_std',	RF	0.9	0.95	0.88333333	0.93333333
	'yaw_std'],	KNN	0.966667	0.95	1	0.93333333
	dtype='object')	SVM	0.9	0.95	0.88333333	0.93333333
		DT	0.933333	0.95	0.95	0.93333333
		AdaBoost	0.86	0.85	0.95	0.76666667
	3	LogReg	0.933333	0.95	0.95	0.93333333
	Index(['velocity_std',	RF	0.933333	0.95	0.95	0.93333333
	'velocity_y_std',	KNN	0.933333	0.95	0.95	0.93333333
	'yaw_std'],	SVM	0.9	0.95	0.88333333	0.93333333
	dtype='object')	DT	0.833333	0.9	0.83333333	0.86666667
		AdaBoost	0.86	0.85	0.95	0.76666667
	4	LogReg	0.933333	0.95	0.95	0.93333333
	Index(['velocity_std',	RF	0.893333	0.9	0.95	0.83333333
	'velocity_y_std',	KNN	0.893333	0.9	0.95	0.83333333
	'phi_angle_std',	SVM	0.826667	0.9	0.81666667	0.83333333

	'yaw_std'], dtype='object')	DT	0.9	0.95	0.88333333	0.93333333
		AdaBoost	0.866667	0.9	0.88333333	0.86666667
	5 Index(['velocity_std', 'velocity_y_std', 'phi_angle_std', 'yaw_std', 'crackle_mass'],	LogReg	0.893333	0.9	0.95	0.83333333
		RF	0.86	0.9	0.9	0.83333333
		KNN	0.886667	0.87	1	0.73333333
		SVM	0.926667	0.9	1	0.83333333
		DT	0.826667	0.9	0.83333333	0.83333333
		AdaBoost	0.793333	0.85	0.83333333	0.76666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'middle')]	2 Index(['velocity_std', 'altitude_acceleration_max'], dtype='object')	LogReg	0.753333	0.75	0.8	0.7
		RF	0.753333	0.8	0.76666667	0.7
		KNN	0.68	0.65	0.8	0.56666667
		SVM	0.76	0.73333333	0.86666667	0.63333333
		DT	0.68	0.7	0.7	0.63333333
		AdaBoost	0.606667	0.58333333	0.53333333	0.63333333
	3 Index(['velocity_std', 'pressure_median', 'altitude_acceleration_max'], dtype='object')	LogReg	0.746667	0.73666667	0.86666667	0.63333333
		RF	0.673333	0.75	0.63333333	0.7
		KNN	0.753333	0.73333333	0.86666667	0.63333333
		SVM	0.686667	0.68333333	0.73333333	0.63333333
		DT	0.64	0.68333333	0.63333333	0.63333333
		AdaBoost	0.6	0.61666667	0.53333333	0.63333333
	4 Index(['velocity_std', 'phi_angle_jerk_std', 'pressure_median', 'altitude_acceleration_m	LogReg	0.78	0.76666667	0.86666667	0.7
		RF	0.6	0.66666667	0.56666667	0.63333333
		KNN	0.746667	0.73666667	0.86666667	0.63333333
		SVM	0.713333	0.72	0.8	0.63333333
		DT	0.673333	0.71666667	0.7	0.63333333
		AdaBoost	0.486667	0.5	0.46666667	0.46666667
	5 Index(['phi_angle_acceleration_max', 'velocity_std', 'phi_angle_jerk_std', 'pressure_median',	LogReg	0.713333	0.78666667	0.7	0.7
		RF	0.6	0.7	0.56666667	0.63333333
		KNN	0.753333	0.73333333	0.86666667	0.63333333
		SVM	0.673333	0.68666667	0.7	0.63333333
		DT	0.606667	0.51666667	0.56666667	0.63333333
		AdaBoost	0.613333	0.66666667	0.6	0.6
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 3), ('position: ', 'middle')]	2 Index(['acceleration_y_min', 'jerk_max'], dtype='object')	LogReg	0.62	0.63666667	0.66666667	0.56666667
		RF	0.573333	0.61666667	0.53333333	0.63333333
		KNN	0.646667	0.68333333	0.66666667	0.63333333
		SVM	0.613333	0.65	0.6	0.63333333
		DT	0.573333	0.73333333	0.53333333	0.63333333
		AdaBoost	0.54	0.58666667	0.53333333	0.56666667
	3 Index(['acceleration_y_min', 'jerk_max', 'shake_median'], dtype='object')	LogReg	0.613333	0.65	0.6	0.63333333
		RF	0.573333	0.66666667	0.6	0.56666667
		KNN	0.606667	0.68333333	0.66666667	0.53333333
		SVM	0.613333	0.68333333	0.6	0.63333333
		DT	0.646667	0.76666667	0.6	0.7
		AdaBoost	0.54	0.61666667	0.53333333	0.56666667
	4 Index(['acceleration_y_min', 'jerk_max', 'tug_min',	LogReg	0.673333	0.71666667	0.73333333	0.6
		RF	0.606667	0.63333333	0.6	0.63333333
		KNN	0.74	0.77	0.86666667	0.6

	'shake_median'], dtype='object')	SVM	0.713333	0.75	0.73333333	0.7
		DT	0.606667	0.73333333	0.6	0.63333333
		AdaBoost	0.64	0.71666667	0.73333333	0.56666667
	5 Index(['acceleration_min', 'acceleration_y_min', 'jerk_max', 'tug_min', 'shake_median'], dtype='object')	LogReg	0.746667	0.78333333	0.8	0.7
		RF	0.64	0.72	0.66666667	0.63333333
		KNN	0.78	0.77	0.93333333	0.6
		SVM	0.78	0.81666667	0.8	0.76666667
		DT	0.673333	0.78333333	0.6	0.76666667
		AdaBoost	0.64	0.71666667	0.66666667	0.63333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 4), ('position: ', 'middle')]	2 Index(['acceleration_mean', 'azimuth_jerk_mass'], dtype='object')	LogReg	0.42	0.38666667	0.53333333	0.26666667
		RF	0.346667	0.33333333	0.46666667	0.2
		KNN	0.486667	0.48666667	0.66666667	0.26666667
		SVM	0.48	0.5	0.66666667	0.26666667
		DT	0.38	0.26666667	0.4	0.33333333
		AdaBoost	0.38	0.33333333	0.46666667	0.26666667
	3 Index(['acceleration_mean', 'acceleration_std', 'azimuth_jerk_mass'], dtype='object')	LogReg	0.38	0.4	0.53333333	0.2
		RF	0.413333	0.38666667	0.6	0.2
		KNN	0.486667	0.47333333	0.73333333	0.2
		SVM	0.446667	0.46666667	0.66666667	0.2
		DT	0.313333	0.26666667	0.4	0.2
		AdaBoost	0.42	0.4	0.53333333	0.3
	4 Index(['acceleration_mean', 'acceleration_std', 'yaw_max', 'azimuth_jerk_mass'],	LogReg	0.42	0.4	0.53333333	0.3
		RF	0.386667	0.36666667	0.46666667	0.3
		KNN	0.56	0.53666667	0.8	0.3
		SVM	0.553333	0.54	0.73333333	0.36666667
		DT	0.346667	0.26666667	0.4	0.26666667
		AdaBoost	0.38	0.36666667	0.53333333	0.2
	5 Index(['acceleration_mean', 'acceleration_std', 'yaw_max', 'tug_mean', 'azimuth_jerk_mass'],	LogReg	0.553333	0.48666667	0.66666667	0.43333333
		RF	0.48	0.45333333	0.66666667	0.26666667
		KNN	0.56	0.55	0.73333333	0.36666667
		SVM	0.52	0.45333333	0.6	0.43333333
		DT	0.413333	0.4	0.53333333	0.26666667
		AdaBoost	0.38	0.36666667	0.53333333	0.2
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 5), ('position: ', 'middle')]	2 Index(['velocity_std', 'acceleration_y_std'], dtype='object')	LogReg	0.753333	0.83333333	0.8	0.73333333
		RF	0.68	0.83	0.66666667	0.7
		KNN	0.713333	0.78333333	0.8	0.63333333
		SVM	0.713333	0.78333333	0.8	0.63333333
		DT	0.68	0.83	0.66666667	0.7
		AdaBoost	0.713333	0.85	0.73333333	0.7
	3 Index(['velocity_std', 'acceleration_y_std', 'yank_median'], dtype='object')	LogReg	0.753333	0.83333333	0.8	0.73333333
		RF	0.726667	0.78	0.73333333	0.73333333
		KNN	0.753333	0.83333333	0.8	0.73333333
		SVM	0.68	0.6	0.66666667	0.73333333
		DT	0.646667	0.8	0.6	0.7
		AdaBoost	0.646667	0.8	0.6	0.7
	4 Index(['velocity_std',	LogReg	0.826667	0.9	0.86666667	0.8
		RF	0.76	0.8	0.8	0.73333333

	'acceleration_y_std', 'jerk_y_std', 'yank_median'], dtype='object')	KNN	0.713333	0.85	0.73333333	0.7
		SVM	0.753333	0.9	0.73333333	0.8
		DT	0.713333	0.85	0.73333333	0.7
		AdaBoost	0.646667	0.85	0.53333333	0.8
	5 Index(['velocity_std', 'acceleration_x_median', 'acceleration_y_std', 'jerk_y_std', 'yank_median'],	LogReg	0.826667	0.9	0.86666667	0.8
		RF	0.76	0.8	0.8	0.73333333
		KNN	0.753333	0.9	0.73333333	0.8
		SVM	0.786667	0.92	0.73333333	0.86666667
		DT	0.753333	0.9	0.73333333	0.8
		AdaBoost	0.686667	0.85	0.6	0.8
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'end')]	2 Index(['velocity_std', 'acceleration_std'], dtype='object')	LogReg	0.614286	0.59333333	0.88333333	0.26666667
		RF	0.557143	0.48	0.75	0.33333333
		KNN	0.642857	0.63	0.9	0.33333333
		SVM	0.609524	0.61	0.83333333	0.33333333
		DT	0.457143	0.48666667	0.6	0.26666667
		AdaBoost	0.490476	0.44	0.61666667	0.33333333
	3 Index(['velocity_std', 'acceleration_std', 'azimuth_velocity_mean'], dtype='object')	LogReg	0.704762	0.66333333	0.93333333	0.4
		RF	0.585714	0.57333333	0.78333333	0.33333333
		KNN	0.647619	0.62666667	0.9	0.33333333
		SVM	0.709524	0.67333333	0.95	0.4
		DT	0.62381	0.55333333	0.73333333	0.46666667
		AdaBoost	0.685714	0.67333333	0.85	0.46666667
	4 Index(['velocity_std', 'acceleration_std', 'jerk_std', 'azimuth_velocity_mean'	LogReg	0.804762	0.76333333	1	0.53333333
		RF	0.62381	0.58333333	0.66666667	0.53333333
		KNN	0.709524	0.67333333	0.95	0.4
		SVM	0.77619	0.75333333	0.95	0.53333333
		DT	0.657143	0.63333333	0.66666667	0.6
		AdaBoost	0.719048	0.75333333	0.78333333	0.6
	5 Index(['velocity_std', 'velocity_x_std', 'acceleration_std', 'jerk_std',	LogReg	0.771429	0.74333333	0.93333333	0.53333333
		RF	0.714286	0.73666667	0.83333333	0.53333333
		KNN	0.738095	0.68333333	1	0.4
		SVM	0.738095	0.69333333	0.93333333	0.46666667
		DT	0.657143	0.63333333	0.66666667	0.6
		AdaBoost	0.685714	0.67333333	0.85	0.46666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'end')]	2 Index(['jerk_min', 'jerk_x_min'], dtype='object')	LogReg	0.686667	0.85	0.65	0.8
		RF	0.653333	0.75333333	0.71666667	0.63333333
		KNN	0.686667	0.85	0.65	0.8
		SVM	0.686667	0.85	0.65	0.8
		DT	0.68	0.74	0.83333333	0.53333333
		AdaBoost	0.68	0.75333333	0.76666667	0.6
	3 Index(['jerk_min', 'jerk_x_min', 'shake_mass'], dtype='object')	LogReg	0.72	0.79	0.85	0.6
		RF	0.62	0.72	0.78333333	0.46666667
		KNN	0.646667	0.75	0.65	0.7
		SVM	0.686667	0.74	0.85	0.53333333
		DT	0.653333	0.73333333	0.76666667	0.53333333
		AdaBoost	0.653333	0.75333333	0.78333333	0.53333333
	4	LogReg	0.753333	0.84	0.83333333	0.66666667

	Index(['jerk_min', 'jerk_x_min', 'pressure_mean', 'shake_mass'], dtype='object')	RF	0.72	0.84	0.78333333	0.66666667
		KNN	0.786667	0.92	0.76666667	0.86666667
		SVM	0.786667	0.85333333	0.83333333	0.76666667
		DT	0.686667	0.8	0.76666667	0.6
		AdaBoost	0.62	0.75333333	0.71666667	0.53333333
	5 Index(['jerk_min', 'jerk_x_min', 'pressure_mean', 'shake_mass', 'shake_max'],	LogReg	0.753333	0.84	0.83333333	0.66666667
		RF	0.686667	0.78666667	0.71666667	0.7
		KNN	0.82	0.87	0.9	0.73333333
		SVM	0.786667	0.87	0.83333333	0.73333333
		DT	0.686667	0.8	0.76666667	0.6
		AdaBoost	0.653333	0.75333333	0.78333333	0.53333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 3), ('position: ', 'end')]	2 Index(['velocity_x_mean', 'jerk_y_mean'], dtype='object')	LogReg	0.571429	0.65	0.56666667	0.56666667
		RF	0.47619	0.59761905	0.51666667	0.43333333
		KNN	0.571429	0.66428571	0.61666667	0.5
		SVM	0.514286	0.63095238	0.51666667	0.5
		DT	0.419048	0.48095238	0.41666667	0.43333333
		AdaBoost	0.480952	0.61428571	0.46666667	0.5
	3 Index(['velocity_x_mean', 'jerk_y_mean', 'azimuth_diff_max'], dtype='object')	LogReg	0.57619	0.61666667	0.63333333	0.5
		RF	0.480952	0.54761905	0.56666667	0.36666667
		KNN	0.638095	0.7	0.68333333	0.6
		SVM	0.638095	0.76666667	0.58333333	0.7
		DT	0.571429	0.6	0.61666667	0.5
		AdaBoost	0.47619	0.55	0.51666667	0.43333333
	4 Index(['velocity_x_mean', 'jerk_y_mean', 'altitude_diff_mean', 'azimuth_diff_max'], dtype='object')	LogReg	0.633333	0.66	0.61666667	0.63333333
		RF	0.57619	0.66428571	0.56666667	0.56666667
		KNN	0.666667	0.7	0.73333333	0.56666667
		SVM	0.633333	0.68333333	0.61666667	0.63333333
		DT	0.604762	0.71666667	0.56666667	0.63333333
		AdaBoost	0.571429	0.63333333	0.56666667	0.56666667
	5 Index(['velocity_x_mean', 'velocity_x_std', 'jerk_y_mean', 'altitude_diff_mean', 'azimuth_diff_max'],	LogReg	0.733333	0.77	0.73333333	0.7
		RF	0.57619	0.66428571	0.56666667	0.56666667
		KNN	0.738095	0.74666667	0.85	0.56666667
		SVM	0.671429	0.74333333	0.66666667	0.63333333
		DT	0.542857	0.57	0.61666667	0.43333333
		AdaBoost	0.633333	0.74666667	0.61666667	0.63333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 4), ('position: ', 'end')]	2 Index(['velocity_x_mass', 'velocity_x_std'], dtype='object')	LogReg	0.766667	0.81666667	0.76666667	0.76666667
		RF	0.8	0.88333333	0.75	0.83333333
		KNN	0.833333	0.85	0.88333333	0.76666667
		SVM	0.766667	0.8	0.78333333	0.76666667
		DT	0.766667	0.8	0.81666667	0.7
		AdaBoost	0.833333	0.88333333	0.81666667	0.83333333
	3 Index(['velocity_x_mass', 'velocity_x_std', 'phi_angle_mean'], dtype='object')	LogReg	0.733333	0.80333333	0.75	0.7
		RF	0.8	0.83333333	0.81666667	0.76666667
		KNN	0.766667	0.77	0.88333333	0.63333333
		SVM	0.833333	0.85	0.88333333	0.76666667
		DT	0.833333	0.85	0.88333333	0.76666667
		AdaBoost	0.8	0.88333333	0.76666667	0.86666667

	4 Index(['velocity_x_mass', 'velocity_x_std', 'phi_angle_mean', 'pressure_median'], dtype='object')	LogReg	0.866667	0.95	0.81666667	0.93333333
		RF	0.766667	0.78333333	0.75	0.8
		KNN	0.833333	0.85	0.88333333	0.8
		SVM	0.833333	0.9	0.81666667	0.86666667
		DT	0.8	0.8	0.81666667	0.8
		AdaBoost	0.7	0.70333333	0.75	0.66666667
	5 Index(['velocity_x_mass', 'velocity_x_std', 'acceleration_y_max', 'phi_angle_mean', 'pressure_median'],	LogReg	0.866667	0.9	0.88333333	0.86666667
		RF	0.766667	0.75	0.81666667	0.73333333
		KNN	0.833333	0.87	0.88333333	0.8
		SVM	0.833333	0.9	0.81666667	0.86666667
		DT	0.8	0.8	0.81666667	0.8
		AdaBoost	0.7	0.69	0.81666667	0.6
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 5), ('position: ', 'end')]	2 Index(['acceleration_x_std', 'phi_angle_mean'], dtype='object')	LogReg	0.747619	0.86	0.66666667	0.86666667
		RF	0.752381	0.89333333	0.68333333	0.86666667
		KNN	0.747619	0.9	0.61666667	0.93333333
		SVM	0.780952	0.93333333	0.68333333	0.93333333
		DT	0.719048	0.88333333	0.63333333	0.83333333
		AdaBoost	0.685714	0.85333333	0.63333333	0.73333333
	3 Index(['acceleration_x_std', 'phi_angle_mean', 'yaw_acceleration_mean'], dtype='object')	LogReg	0.714286	0.82666667	0.66666667	0.8
		RF	0.77619	0.88333333	0.73333333	0.83333333
		KNN	0.714286	0.86666667	0.61666667	0.86666667
		SVM	0.685714	0.86666667	0.56666667	0.86666667
		DT	0.719048	0.84333333	0.68333333	0.76666667
		AdaBoost	0.714286	0.85333333	0.68333333	0.73333333
	4 Index(['acceleration_x_mean', 'acceleration_x_std', 'phi_angle_mean',	LogReg	0.752381	0.87	0.75	0.8
		RF	0.780952	0.93333333	0.68333333	0.93333333
		KNN	0.780952	0.92	0.75	0.86666667
		SVM	0.714286	0.9	0.63333333	0.86666667
		DT	0.685714	0.79333333	0.68333333	0.66666667
		AdaBoost	0.742857	0.85333333	0.73333333	0.73333333
	5 Index(['acceleration_x_mean', 'acceleration_x_std', 'phi_angle_mean',	LogReg	0.752381	0.85333333	0.75	0.8
		RF	0.780952	0.93333333	0.68333333	0.93333333
		KNN	0.752381	0.85333333	0.75	0.8
		SVM	0.780952	0.92	0.75	0.86666667
		DT	0.690476	0.88333333	0.58333333	0.86666667
		AdaBoost	0.719048	0.83333333	0.63333333	0.86666667

Appendix E

Results of the transition corners segmented PL trace task

settings	features	model	accuracy	precision	sensitivity	specificity
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'start')]	2 Index(['yaw_mass', 'alphas_mean'], dtype='object')	LogReg	0.698182	0.63	0.63	0.747619
		RF	0.643636	0.576667	0.53	0.719048
		KNN	0.663636	0.596667	0.58	0.719048
		SVM	0.718182	0.68	0.63	0.780952
		DT	0.572727	0.503333	0.49	0.633333
		AdaBoost	0.623636	0.516667	0.63	0.619048
	3 Index(['yaw_mass', 'alphas_mean', 'pressure_median'], dtype='object')	LogReg	0.698182	0.629048	0.63	0.757143
		RF	0.643636	0.556667	0.54	0.719048
		KNN	0.681818	0.619048	0.59	0.757143
		SVM	0.68	0.576667	0.64	0.719048
		DT	0.589091	0.536667	0.49	0.661905
		AdaBoost	0.683636	0.6	0.68	0.690476
	4 Index(['slopes_min', 'yaw_mass', 'alphas_mean', 'pressure_median'], dtype='object')	LogReg	0.734545	0.663333	0.63	0.814286
		RF	0.721818	0.746667	0.53	0.842857
		KNN	0.718182	0.626667	0.59	0.809524
		SVM	0.716364	0.61	0.5	0.87619
		DT	0.7	0.65	0.52	0.809524
		AdaBoost	0.701818	0.62	0.58	0.785714
	5 Index(['slopes_min', 'yaw_mass', 'yaw_velocity_median', 'alphas_mean', 'pressure_median'], dtype='object')	LogReg	0.716364	0.62	0.63	0.780952
		RF	0.701818	0.683333	0.49	0.838095
		KNN	0.756364	0.68	0.69	0.809524
		SVM	0.736364	0.62	0.55	0.87619
		DT	0.701818	0.616667	0.62	0.747619
		AdaBoost	0.701818	0.63	0.58	0.785714
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'start')]	2 Index(['velocity_x_max', 'acceleration_y_mass'], dtype='object')	LogReg	0.688889	0.64	0.583333	0.76
		RF	0.691111	0.65	0.533333	0.793333
		KNN	0.668889	0.606667	0.533333	0.76
		SVM	0.624444	0.506667	0.416667	0.76
		DT	0.711111	0.633333	0.7	0.726667
		AdaBoost	0.664444	0.49	0.6	0.726667
	3 Index(['velocity_x_max', 'acceleration_y_mass', 'acceleration_y_min'], dtype='object')	LogReg	0.648889	0.613333	0.583333	0.693333
		RF	0.653333	0.602381	0.583333	0.693333
		KNN	0.608889	0.546667	0.533333	0.66
		SVM	0.648889	0.594286	0.583333	0.693333
		DT	0.671111	0.593333	0.65	0.693333
		AdaBoost	0.606667	0.505714	0.65	0.58
	4 Index(['velocity_x_max', 'velocity_x_std', 'acceleration_y_mass', 'acceleration_y_min'],	LogReg	0.631111	0.605714	0.633333	0.62
		RF	0.628889	0.546667	0.583333	0.66
		KNN	0.631111	0.58	0.583333	0.66
		SVM	0.586667	0.546667	0.466667	0.66
		DT	0.606667	0.513333	0.533333	0.653333
		AdaBoost	0.626667	0.552381	0.533333	0.693333

	5 Index(['velocity_x_max', 'velocity_x_std', 'acceleration_y_mass', 'acceleration_y_min', 'phi_angle_std'],	LogReg	0.611111	0.552381	0.583333	0.62
		RF	0.608889	0.502381	0.55	0.66
		KNN	0.606667	0.5	0.583333	0.626667
		SVM	0.628889	0.6	0.516667	0.693333
		DT	0.728889	0.65	0.7	0.753333
		AdaBoost	0.586667	0.495	0.6	0.593333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'middle')]	2 Index(['jerk_median', 'phi_angle_mean'], dtype='object')	LogReg	0.727273	0.753333	0.6	0.814286
		RF	0.745455	0.75	0.6	0.838095
		KNN	0.765455	0.773333	0.65	0.842857
		SVM	0.747273	0.783333	0.6	0.847619
		DT	0.725455	0.553333	0.55	0.838095
		AdaBoost	0.745455	0.703333	0.6	0.838095
	3 Index(['jerk_median', 'phi_angle_mean', 'pressure_diff_min'], dtype='object')	LogReg	0.709091	0.689048	0.65	0.752381
		RF	0.723636	0.703333	0.55	0.833333
		KNN	0.689091	0.659048	0.65	0.719048
		SVM	0.729091	0.719048	0.65	0.785714
		DT	0.745455	0.773333	0.6	0.838095
		AdaBoost	0.683636	0.59	0.55	0.766667
	4 Index(['Unnamed: 0', 'jerk_median', 'phi_angle_mean', 'pressure_diff_min'], dtype='object')	LogReg	0.629091	0.539048	0.6	0.652381
		RF	0.705455	0.703333	0.55	0.804762
		KNN	0.667273	0.62	0.65	0.685714
		SVM	0.689091	0.669048	0.6	0.752381
		DT	0.665455	0.586667	0.55	0.738095
		AdaBoost	0.667273	0.566667	0.6	0.709524
	5 Index(['Unnamed: 0', 'jerk_median', 'phi_angle_mean', 'pressure_diff_min', 'tug_max'],	LogReg	0.629091	0.552381	0.65	0.619048
		RF	0.687273	0.67	0.55	0.77619
		KNN	0.667273	0.62	0.7	0.652381
		SVM	0.649091	0.635714	0.6	0.685714
		DT	0.647273	0.602381	0.55	0.709524
		AdaBoost	0.605455	0.52	0.5	0.671429
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'middle')]	2 Index(['Unnamed: 0', 'phi_angle_std'], dtype='object')	LogReg	0.7	0.651667	0.7	0.7
		RF	0.7	0.7	0.5	0.833333
		KNN	0.56	0.461667	0.5	0.6
		SVM	0.66	0.635714	0.55	0.733333
		DT	0.72	0.713333	0.6	0.8
		AdaBoost	0.68	0.725	0.6	0.733333
	3 Index(['Unnamed: 0', 'acceleration_x_median', 'phi_angle_std'], dtype='object')	LogReg	0.7	0.700476	0.65	0.733333
		RF	0.7	0.75	0.45	0.866667
		KNN	0.62	0.608333	0.5	0.7
		SVM	0.6	0.590476	0.45	0.7
		DT	0.74	0.733333	0.55	0.866667
		AdaBoost	0.64	0.513333	0.4	0.8
	4 Index(['Unnamed: 0', 'acceleration_x_median', 'phi_angle_std',	LogReg	0.64	0.59381	0.6	0.666667
		RF	0.72	0.72	0.55	0.833333
		KNN	0.64	0.591667	0.6	0.666667
		SVM	0.64	0.617143	0.55	0.7
		DT	0.68	0.616667	0.5	0.8

	'yaw_acceleration_media	AdaBoost	0.66	0.62	0.5	0.766667
	5	LogReg	0.68	0.608333	0.75	0.633333
	Index(['Unnamed: 0',	RF	0.68	0.633333	0.6	0.733333
	'acceleration_x_median',	KNN	0.68	0.59	0.7	0.666667
	'phi_angle_std',	SVM	0.66	0.617143	0.6	0.7
		DT	0.6	0.505714	0.5	0.666667
	'phi_angle_jerk_mean',	AdaBoost	0.66	0.483333	0.5	0.766667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'end')]	2	LogReg	0.68	0.708333	0.63	0.733333
	Index(['acceleration_x_median',	RF	0.6	0.53	0.54	0.66
	'yaw_velocity_mass'],	KNN	0.72	0.783333	0.53	0.866667
	dtype='object')	SVM	0.7	0.769048	0.58	0.8
		DT	0.56	0.5	0.53	0.593333
		AdaBoost	0.56	0.503333	0.48	0.626667
	3	LogReg	0.68	0.67	0.58	0.766667
	Index(['acceleration_x_median',	RF	0.64	0.6	0.53	0.726667
	'phi_angle_velocity_mass',	KNN	0.72	0.74	0.63	0.8
		SVM	0.7	0.739048	0.63	0.766667
		DT	0.56	0.45	0.48	0.626667
		AdaBoost	0.6	0.536667	0.58	0.62
	4	LogReg	0.64	0.633333	0.48	0.766667
	Index(['acceleration_x_median',	RF	0.68	0.65	0.53	0.793333
	'phi_angle_velocity_mass', 'yaw_velocity_mass',	KNN	0.72	0.766667	0.63	0.8
		SVM	0.74	0.797619	0.63	0.833333
		DT	0.6	0.590476	0.48	0.693333
		AdaBoost	0.64	0.57	0.58	0.686667
	5	LogReg	0.68	0.733333	0.53	0.8
	Index(['acceleration_x_median',	RF	0.72	0.7	0.63	0.793333
	'phi_angle_velocity_mass', 'yaw_velocity_mass',	KNN	0.66	0.672381	0.58	0.726667
		SVM	0.68	0.769048	0.53	0.8
		DT	0.58	0.493333	0.48	0.653333
	'alphas_median',	AdaBoost	0.58	0.477619	0.54	0.62
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'end')]	2	LogReg	0.726667	0.733333	0.516667	0.86
	Index(['duration',	RF	0.726667	0.666667	0.566667	0.826667
	'phi_angle_mass'],	KNN	0.748889	0.783333	0.566667	0.86
	dtype='object')	SVM	0.748889	0.816667	0.516667	0.893333
		DT	0.706667	0.633333	0.616667	0.76
		AdaBoost	0.686667	0.593333	0.616667	0.726667
	3	LogReg	0.771111	0.85	0.566667	0.893333
	Index(['duration',	RF	0.706667	0.65	0.516667	0.826667
	'phi_angle_mass',	KNN	0.748889	0.8	0.516667	0.893333
	'altitude_diff_std'],	SVM	0.748889	0.833333	0.516667	0.893333
	dtype='object')	DT	0.686667	0.62	0.516667	0.793333
		AdaBoost	0.686667	0.62	0.516667	0.793333
	4	LogReg	0.768889	0.75	0.616667	0.86
	Index(['duration',	RF	0.726667	0.733333	0.516667	0.86
	'phi_angle_mass',	KNN	0.748889	0.813333	0.566667	0.86
	'pressure_max',	SVM	0.726667	0.733333	0.516667	0.86

	'altitude_diff_std'],	DT	0.706667	0.65	0.516667	0.826667
	dtype='object')	AdaBoost	0.726667	0.78	0.516667	0.86
	5	LogReg	0.811111	0.883333	0.616667	0.926667
	Index(['duration',	RF	0.726667	0.833333	0.416667	0.926667
	'phi_angle_mass',	KNN	0.708889	0.813333	0.466667	0.86
	'alphas_mass',	SVM	0.728889	0.813333	0.516667	0.86
	'pressure_max',	DT	0.662222	0.666667	0.416667	0.82
	'altitude_diff_std'],	AdaBoost	0.728889	0.82	0.516667	0.853333

Appendix F

Results of the transition corners segmented PL copy task

settings	features	model	accuracy	precision	sensitivity	specificity
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'start')]	2 Index(['jerk_y_std', 'altitude_diff_std'], dtype='object')	LogReg	0.62	0.525	0.65	0.6
		RF	0.58	0.45888889	0.65	0.53333333
		KNN	0.64	0.55555556	0.65	0.63333333
		SVM	0.68	0.62222222	0.7	0.66666667
		DT	0.54	0.44404762	0.6	0.5
		AdaBoost	0.6	0.50833333	0.55	0.63333333
	3 Index(['jerk_y_std', 'altitude_diff_std', 'azimuth_velocity_mass'], dtype='object')	LogReg	0.64	0.575	0.6	0.66666667
		RF	0.56	0.42833333	0.55	0.56666667
		KNN	0.64	0.535	0.65	0.63333333
		SVM	0.66	0.59166667	0.65	0.66666667
		DT	0.54	0.38928571	0.6	0.5
		AdaBoost	0.56	0.40833333	0.55	0.56666667
	4 Index(['jerk_y_std', 'altitude_max', 'altitude_diff_std', 'azimuth_velocity_mass']	LogReg	0.66	0.58888889	0.65	0.66666667
		RF	0.52	0.36984127	0.55	0.5
		KNN	0.56	0.45595238	0.65	0.5
		SVM	0.66	0.57222222	0.65	0.66666667
		DT	0.52	0.37555556	0.6	0.46666667
		AdaBoost	0.56	0.375	0.5	0.6
	5 Index(['Unnamed: 0', 'jerk_y_std', 'altitude_max', 'altitude_diff_std',	LogReg	0.66	0.48222222	0.65	0.66666667
		RF	0.64	0.53095238	0.6	0.66666667
		KNN	0.68	0.65555556	0.8	0.6
		SVM	0.68	0.67222222	0.65	0.7
		DT	0.64	0.49761905	0.6	0.66666667
		AdaBoost	0.64	0.46666667	0.5	0.73333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'start')]	2 Index(['acceleration_std', 'yaw_jerk_max'], dtype='object')	LogReg	0.649091	0.55333333	0.35	0.83809524
		RF	0.607273	0.56666667	0.45	0.70952381
		KNN	0.609091	0.5	0.45	0.71428571
		SVM	0.630909	0.60333333	0.4	0.78095238
		DT	0.592727	0.48666667	0.5	0.65238095
		AdaBoost	0.550909	0.43	0.45	0.61428571
	3 Index(['acceleration_std', 'yaw_jerk_max', 'alphas_std'], dtype='object')	LogReg	0.590909	0.42	0.35	0.74285714
		RF	0.629091	0.58333333	0.4	0.77619048
		KNN	0.670909	0.63333333	0.45	0.81428571
		SVM	0.650909	0.58333333	0.35	0.84285714
		DT	0.550909	0.43	0.4	0.64761905
		AdaBoost	0.589091	0.47666667	0.45	0.68095238
	4 Index(['acceleration_std', 'phi_angle_mean', 'yaw_jerk_max', 'alphas_std'], dtype='object')	LogReg	0.610909	0.52	0.35	0.77619048
		RF	0.669091	0.61666667	0.45	0.80952381
		KNN	0.632727	0.55	0.4	0.78095238
		SVM	0.670909	0.65	0.35	0.87619048
		DT	0.550909	0.43333333	0.45	0.61904762
		AdaBoost	0.569091	0.36333333	0.35	0.70952381

	5 Index(['acceleration_std', 'phi_angle_mean', 'yaw_jerk_max', 'alphas_std',	LogReg	0.549091	0.41333333	0.35	0.67619048
		RF	0.629091	0.58333333	0.4	0.77619048
		KNN	0.570909	0.45	0.3	0.74761905
		SVM	0.609091	0.61666667	0.3	0.80952381
		DT	0.549091	0.49904762	0.4	0.65238095
		AdaBoost	0.649091	0.53	0.6	0.68095238
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'middle')]	2 Index(['velocity_std', 'phi_angle_mean'], dtype='object')	LogReg	0.857778	0.78	0.9	0.83333333
		RF	0.775556	0.82	0.7	0.83333333
		KNN	0.84	0.86	0.85	0.83333333
		SVM	0.84	0.9	0.75	0.9
		DT	0.755556	0.68904762	0.8	0.73333333
		AdaBoost	0.775556	0.82	0.7	0.83333333
	3 Index(['velocity_std', 'phi_angle_mean', 'tug_min'], dtype='object')	LogReg	0.837778	0.83	0.8	0.86666667
		RF	0.837778	0.87	0.75	0.9
		KNN	0.817778	0.81	0.8	0.83333333
		SVM	0.86	0.92	0.75	0.93333333
		DT	0.777778	0.71666667	0.75	0.8
		AdaBoost	0.777778	0.75	0.7	0.83333333
	4 Index(['velocity_std', 'phi_angle_mean', 'tug_min', 'altitude_max'], dtype='object')	LogReg	0.817778	0.81	0.8	0.83333333
		RF	0.817778	0.85	0.75	0.86666667
		KNN	0.797778	0.8	0.75	0.83333333
		SVM	0.84	0.9	0.75	0.9
		DT	0.717778	0.63571429	0.75	0.7
		AdaBoost	0.837778	0.85	0.8	0.86666667
	5 Index(['velocity_std', 'phi_angle_mean', 'tug_min', 'altitude_max', 'azimuth_acceleration_m	LogReg	0.82	0.86	0.75	0.86666667
		RF	0.817778	0.83571429	0.8	0.83333333
		KNN	0.775556	0.77	0.75	0.8
		SVM	0.82	0.9	0.7	0.9
		DT	0.677778	0.56238095	0.65	0.7
		AdaBoost	0.817778	0.65	0.75	0.86666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'middle')]	2 Index(['velocity_x_max', 'alphas_velocity_min'], dtype='object')	LogReg	0.689091	0.61333333	0.55	0.78095238
		RF	0.589091	0.44666667	0.43333333	0.69047619
		KNN	0.707273	0.66666667	0.53333333	0.80952381
		SVM	0.689091	0.64666667	0.53333333	0.78095238
		DT	0.549091	0.39904762	0.41666667	0.62857143
		AdaBoost	0.627273	0.5	0.48333333	0.72380952
	3 Index(['velocity_x_max', 'alphas_velocity_min', 'pressure_deriv_mass'], dtype='object')	LogReg	0.629091	0.46333333	0.5	0.71904762
		RF	0.510909	0.31666667	0.33333333	0.62380952
		KNN	0.649091	0.58	0.48333333	0.74761905
		SVM	0.649091	0.58	0.48333333	0.74761905
		DT	0.489091	0.33238095	0.41666667	0.52857143
		AdaBoost	0.529091	0.31904762	0.43333333	0.5952381
	4 Index(['velocity_x_max', 'alphas_velocity_min', 'pressure_deriv_mass', 'altitude_median'],	LogReg	0.687273	0.63333333	0.55	0.78095238
		RF	0.669091	0.58571429	0.5	0.78571429
		KNN	0.667273	0.6	0.5	0.78095238
		SVM	0.687273	0.56666667	0.55	0.78095238
		DT	0.509091	0.35238095	0.43333333	0.55714286

	dtype='object')	AdaBoost	0.629091	0.42666667	0.45	0.74761905
	5	LogReg	0.727273	0.63333333	0.65	0.78571429
	Index(['velocity_x_max',	RF	0.670909	0.60833333	0.5	0.79047619
	'yaw_velocity_median',	KNN	0.727273	0.63333333	0.55	0.84761905
	'alphas_velocity_min',	SVM	0.707273	0.63333333	0.55	0.81428571
		DT	0.549091	0.40238095	0.55	0.55714286
	'pressure_deriv_mass',	AdaBoost	0.589091	0.41	0.45	0.68095238
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'end')]	2	LogReg	0.7	0.57	0.6	0.76666667
	Index(['acceleration_y_mean', 'phi_angle_std'],	RF	0.76	0.8	0.65	0.83333333
	dtype='object')	KNN	0.68	0.55	0.55	0.76666667
		SVM	0.7	0.55	0.55	0.8
		DT	0.68	0.68904762	0.6	0.73333333
		AdaBoost	0.72	0.75	0.65	0.76666667
	3	LogReg	0.72	0.77	0.65	0.76666667
	Index(['acceleration_y_mean', 'phi_angle_std',	RF	0.7	0.745	0.65	0.73333333
	'alphas_acceleration_min'], dtype='object')	KNN	0.74	0.76428571	0.7	0.76666667
		SVM	0.76	0.79	0.65	0.83333333
		DT	0.76	0.8	0.7	0.8
		AdaBoost	0.76	0.79761905	0.65	0.83333333
	4	LogReg	0.78	0.84	0.65	0.86666667
	Index(['velocity_y_std',	RF	0.7	0.72571429	0.65	0.73333333
	'acceleration_y_mean',	KNN	0.7	0.72	0.6	0.76666667
	'phi_angle_std',	SVM	0.72	0.72	0.6	0.8
		DT	0.68	0.665	0.7	0.66666667
	'alphas_acceleration_min	AdaBoost	0.76	0.76	0.7	0.8
	5	LogReg	0.66	0.68666667	0.55	0.73333333
	Index(['velocity_y_std',	RF	0.68	0.72833333	0.6	0.73333333
	'acceleration_y_mean',	KNN	0.66	0.68571429	0.6	0.7
	'phi_angle_std',	SVM	0.68	0.68	0.55	0.76666667
	'yaw_median',	DT	0.74	0.72888889	0.8	0.7
		AdaBoost	0.7	0.70666667	0.65	0.73333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'end')]	2	LogReg	0.590909	0.46666667	0.5	0.65714286
	Index(['alphas_mean',	RF	0.630909	0.5	0.5	0.71904762
	'pressure_diff_mean'],	KNN	0.550909	0.45333333	0.5	0.58571429
	dtype='object')	SVM	0.572727	0.45666667	0.4	0.68571429
		DT	0.550909	0.41333333	0.55	0.56190476
		AdaBoost	0.530909	0.43380952	0.6	0.49047619
	3	LogReg	0.629091	0.56	0.6	0.64761905
	Index(['velocity_std',	RF	0.650909	0.56333333	0.55	0.71428571
	'alphas_mean',	KNN	0.629091	0.62666667	0.53333333	0.68095238
	'pressure_diff_mean'],	SVM	0.610909	0.53	0.43333333	0.71428571
	dtype='object')	DT	0.630909	0.58333333	0.5	0.72380952
		AdaBoost	0.612727	0.50333333	0.55	0.65714286
	4	LogReg	0.630909	0.56333333	0.48333333	0.71904762
	Index(['velocity_std',	RF	0.629091	0.49666667	0.5	0.71428571
	'alphas_mean',	KNN	0.590909	0.59666667	0.41666667	0.68095238
	'pressure_diff_mean',	SVM	0.690909	0.73	0.5	0.81428571

	'yank_min'], dtype='object')	DT	0.572727	0.45714286	0.55	0.6
		AdaBoost	0.610909	0.5	0.6	0.62857143
	5 Index(['velocity_std', 'alphas_mean', 'pressure_mean', 'pressure_diff_mean', 'yank_min'],	LogReg	0.685455	0.64666667	0.53333333	0.77142857
		RF	0.709091	0.68	0.55	0.80952381
		KNN	0.705455	0.68	0.65	0.73809524
		SVM	0.747273	0.78	0.6	0.84285714
		DT	0.670909	0.58	0.65	0.69047619
		AdaBoost	0.689091	0.6	0.6	0.74285714

Appendix G

Results of the transition corners segmented PL continue task

settings	features	model	accuracy	precision	sensitivity	specificity
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'start')]	2 Index(['phi_angle_std', 'altitude_velocity_std'], dtype='object')	LogReg	0.571111	0.4	0.23333333	0.77142857
		RF	0.655556	0.6	0.36666667	0.83333333
		KNN	0.631111	0.43333333	0.38333333	0.77619048
		SVM	0.631111	0.46666667	0.33333333	0.8047619
		DT	0.653333	0.52	0.5	0.73809524
		AdaBoost	0.653333	0.57	0.58333333	0.7
	3 Index(['phi_angle_std', 'pressure_mean', 'altitude_velocity_std'], dtype='object')	LogReg	0.591111	0.4	0.28333333	0.76666667
		RF	0.697778	0.66666667	0.41666667	0.86666667
		KNN	0.671111	0.46666667	0.38333333	0.83809524
		SVM	0.693333	0.5	0.33333333	0.9
		DT	0.651111	0.53333333	0.5	0.73809524
		AdaBoost	0.595556	0.51333333	0.35	0.73809524
	4 Index(['phi_angle_std', 'pressure_mean', 'altitude_max', 'altitude_velocity_std'],	LogReg	0.593333	0.46666667	0.35	0.73333333
		RF	0.613333	0.35	0.23333333	0.83333333
		KNN	0.631111	0.43333333	0.33333333	0.8
		SVM	0.653333	0.43333333	0.33333333	0.83333333
		DT	0.673333	0.46666667	0.48333333	0.77142857
		AdaBoost	0.633333	0.35333333	0.38333333	0.77142857
	5 Index(['acceleration_y_median', 'phi_angle_std', 'pressure_mean', 'altitude_max', 'altitude_velocity_std'],	LogReg	0.593333	0.35	0.28333333	0.76666667
		RF	0.613333	0.38333333	0.28333333	0.8
		KNN	0.611111	0.41666667	0.33333333	0.76666667
		SVM	0.633333	0.41666667	0.33333333	0.8
		DT	0.633333	0.40333333	0.43333333	0.74285714
		AdaBoost	0.573333	0.26666667	0.28333333	0.74285714
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'start')]	2 Index(['velocity_y_max', 'alphas_mean'], dtype='object')	LogReg	0.504444	0.37333333	0.3	0.64
		RF	0.462222	0.25	0.25	0.60666667
		KNN	0.564444	0.38333333	0.45	0.64
		SVM	0.588889	0.53	0.4	0.70666667
		DT	0.462222	0.2	0.2	0.64666667
		AdaBoost	0.546667	0.33	0.28333333	0.71333333
	3 Index(['Unnamed: 0', 'velocity_y_max', 'alphas_mean'], dtype='object')	LogReg	0.631111	0.51666667	0.51666667	0.71333333
		RF	0.588889	0.43	0.41666667	0.71333333
		KNN	0.606667	0.36428571	0.46666667	0.71333333
		SVM	0.717778	0.68428571	0.66666667	0.75333333
		DT	0.546667	0.28333333	0.25	0.75333333
		AdaBoost	0.588889	0.43333333	0.3	0.78
	4 Index(['Unnamed: 0', 'velocity_y_max', 'phi_angle_mean', 'alphas_mean'], dtype='object')	LogReg	0.586667	0.38333333	0.46666667	0.68
		RF	0.586667	0.41666667	0.46666667	0.67333333
		KNN	0.626667	0.38333333	0.46666667	0.74666667
		SVM	0.671111	0.51333333	0.56666667	0.74
		DT	0.604444	0.45	0.4	0.74
		AdaBoost	0.5	0.28	0.3	0.64

	5 Index(['Unnamed: 0', 'velocity_y_max', 'phi_angle_mean', 'alphas_mean', 'azimuth_std'],	LogReg	0.586667	0.41666667	0.46666667	0.67333333
		RF	0.626667	0.46666667	0.46666667	0.74
		KNN	0.626667	0.45	0.51666667	0.70666667
		SVM	0.693333	0.58333333	0.61666667	0.74666667
		DT	0.582222	0.38333333	0.35	0.74
		AdaBoost	0.564444	0.4	0.4	0.68
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 1), ('position: ', 'middle')]	2 Index(['acceleration_x_max', 'yank_std'], dtype='object')	LogReg	0.577778	0.4	0.4	0.68
		RF	0.533333	0.45	0.41666667	0.61333333
		KNN	0.62	0.41666667	0.45	0.70666667
		SVM	0.555556	0.38	0.4	0.64
		DT	0.555556	0.38	0.31666667	0.72
		AdaBoost	0.58	0.46666667	0.36666667	0.72
	3 Index(['acceleration_x_max', 'yaw_max', 'yank_std'], dtype='object')	LogReg	0.662222	0.63333333	0.48333333	0.78666667
		RF	0.682222	0.76666667	0.41666667	0.86
		KNN	0.66	0.61333333	0.51666667	0.74666667
		SVM	0.64	0.48	0.4	0.78
		DT	0.595556	0.48666667	0.41666667	0.71333333
		AdaBoost	0.533333	0.41666667	0.36666667	0.64666667
	4 Index(['acceleration_x_max', 'yaw_max', 'pressure_diff_std', 'yank_std'], dtype='object')	LogReg	0.662222	0.68	0.48333333	0.78
		RF	0.635556	0.61333333	0.46666667	0.74666667
		KNN	0.64	0.58	0.46666667	0.74666667
		SVM	0.744444	0.76333333	0.63333333	0.81333333
		DT	0.615556	0.58333333	0.46666667	0.71333333
		AdaBoost	0.615556	0.6	0.36666667	0.78666667
	5 Index(['velocity_median', 'acceleration_x_max', 'yaw_max', 'pressure_diff_std', 'yank_std'],	LogReg	0.662222	0.68	0.48333333	0.78
		RF	0.635556	0.61333333	0.46666667	0.74666667
		KNN	0.704444	0.63333333	0.68333333	0.71333333
		SVM	0.724444	0.74666667	0.56666667	0.81333333
		DT	0.637778	0.58333333	0.51666667	0.71333333
		AdaBoost	0.637778	0.62	0.46666667	0.75333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('type: ', 2), ('position: ', 'middle')]	2 Index(['Unnamed: 0', 'azimuth_max'], dtype='object')	LogReg	0.680556	0.5	0.36666667	0.84666667
		RF	0.633333	0.36666667	0.33333333	0.80666667
		KNN	0.683333	0.5	0.36666667	0.85333333
		SVM	0.705556	0.53333333	0.36666667	0.88666667
		DT	0.658333	0.4	0.46666667	0.77333333
		AdaBoost	0.611111	0.33333333	0.4	0.74
	3 Index(['Unnamed: 0', 'velocity_x_median', 'azimuth_max'], dtype='object')	LogReg	0.608333	0.26666667	0.3	0.78
		RF	0.658333	0.4	0.36666667	0.81333333
		KNN	0.611111	0.34666667	0.36666667	0.74666667
		SVM	0.630556	0.3	0.3	0.81333333
		DT	0.611111	0.38	0.36666667	0.74
		AdaBoost	0.611111	0.33333333	0.33333333	0.77333333
	4 Index(['Unnamed: 0', 'velocity_x_median', 'yaw_mean', 'azimuth_max'],	LogReg	0.583333	0.26666667	0.3	0.74
		RF	0.586111	0.24666667	0.3	0.74666667
		KNN	0.611111	0.34666667	0.36666667	0.74666667
		SVM	0.633333	0.36666667	0.36666667	0.78
		DT	0.611111	0.31333333	0.36666667	0.74666667

	dtype='object')	AdaBoost	0.588889	0.36666667	0.36666667	0.70666667
	5	LogReg	0.508333	0.21666667	0.3	0.63333333
	Index(['Unnamed: 0',	RF	0.561111	0.24666667	0.3	0.70666667
	'velocity_x_median',	KNN	0.586111	0.31333333	0.36666667	0.70666667
	'yaw_mean',	SVM	0.683333	0.56666667	0.36666667	0.85333333
	'pressure_diff_median',	DT	0.583333	0.26666667	0.43333333	0.67333333
	'azimuth_max'],	AdaBoost	0.658333	0.4	0.53333333	0.74
[('scaler: ',	2	LogReg	0.782222	0.78333333	0.6	0.89333333
StandardScaler()),	Index(['alphas_jerk_min',	RF	0.735556	0.73333333	0.48333333	0.9
('folds: ', 5), ('type:	'altitude_diff_mean'],	KNN	0.671111	0.45	0.35	0.85333333
', 1), ('position: ',	dtype='object')	SVM	0.737778	0.68333333	0.46666667	0.89333333
'end')]		DT	0.713333	0.77333333	0.55	0.83333333
		AdaBoost	0.691111	0.65	0.53333333	0.78666667
	3	LogReg	0.693333	0.61666667	0.61666667	0.75333333
	Index(['alphas_velocity_	RF	0.737778	0.70333333	0.61666667	0.82666667
	median',	KNN	0.713333	0.66666667	0.41666667	0.88666667
	'alphas_jerk_min',	SVM	0.693333	0.73333333	0.48333333	0.82666667
	'altitude_diff_mean'],	DT	0.757778	0.81666667	0.56666667	0.9
	dtype='object')	AdaBoost	0.713333	0.76	0.61666667	0.8
	4	LogReg	0.76	0.73333333	0.66666667	0.82666667
	Index(['alphas_velocity_	RF	0.846667	0.9	0.66666667	0.96666667
	median',	KNN	0.691111	0.66666667	0.41666667	0.85333333
	'alphas_jerk_min',	SVM	0.735556	0.78333333	0.55	0.86
	'pressure_median',	DT	0.802222	0.78666667	0.71666667	0.86
	'altitude_diff_mean'],	AdaBoost	0.8	0.74333333	0.76666667	0.82
	5	LogReg	0.78	0.8	0.61666667	0.89333333
	Index(['alphas_velocity_	RF	0.846667	0.93333333	0.66666667	0.96
	median',	KNN	0.78	0.73333333	0.55	0.92666667
	'alphas_acceleration_min	SVM	0.735556	0.78333333	0.55	0.86
	', 'alphas_jerk_min',	DT	0.802222	0.78666667	0.71666667	0.86
	'pressure_median',	AdaBoost	0.824444	0.79333333	0.71666667	0.89333333
[('scaler: ',	2	LogReg	0.704444	0.68095238	0.58333333	0.76666667
StandardScaler()),	Index(['yaw_max',	RF	0.642222	0.43	0.31666667	0.82666667
('folds: ', 5), ('type:	'altitude_diff_mean'],	KNN	0.62	0.58571429	0.48333333	0.69333333
', 2), ('position: ',	dtype='object')	SVM	0.662222	0.61666667	0.53333333	0.72666667
'end')]		DT	0.62	0.4	0.31666667	0.78666667
		AdaBoost	0.64	0.52	0.41666667	0.75333333
	3	LogReg	0.662222	0.61095238	0.58333333	0.69333333
	Index(['Unnamed: 0',	RF	0.706667	0.67	0.43333333	0.86
	'yaw_max',	KNN	0.662222	0.59761905	0.53333333	0.72666667
	'altitude_diff_mean'],	SVM	0.662222	0.53095238	0.48333333	0.76
	dtype='object')	DT	0.684444	0.46	0.36666667	0.86
		AdaBoost	0.744444	0.63	0.56666667	0.82666667
	4	LogReg	0.702222	0.65095238	0.63333333	0.72666667
	Index(['Unnamed: 0',	RF	0.702222	0.65	0.53333333	0.79333333
	'jerk_x_min', 'yaw_max',	KNN	0.702222	0.66428571	0.58333333	0.76
	'altitude_diff_mean'],	SVM	0.702222	0.73095238	0.53333333	0.79333333

	dtype='object')	DT	0.711111	0.72666667	0.6	0.76
		AdaBoost	0.702222	0.65666667	0.58333333	0.76
	5	LogReg	0.704444	0.68095238	0.58333333	0.76666667
	Index(['Unnamed: 0',	RF	0.706667	0.7	0.43333333	0.86666667
	'jerk_x_min', 'yaw_max',	KNN	0.682222	0.63095238	0.53333333	0.76
	'alphas_velocity_max',	SVM	0.704444	0.68095238	0.53333333	0.8
	'altitude_diff_mean'],	DT	0.708889	0.67333333	0.6	0.76
	dtype='object')	AdaBoost	0.702222	0.65095238	0.63333333	0.72666667

Appendix H

Results of the micrography angles PL trace task

settings	features	model	accuracy	precision	sensitivity	specificity
[('scaler: ', StandardScaler()), ('folds: ', 5), ('position: ', 'start')]	2 Index(['Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.64	0.5	0.2	0.93333333
		RF	0.6	0.48333333	0.35	0.76666667
		KNN	0.68	0.63333333	0.35	0.9
		SVM	0.6	0.2	0.05	0.96666667
		DT	0.68	0.6	0.65	0.7
		AdaBoost	0.66	0.59	0.6	0.7
	3 Index(['Upper line angle', 'Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.66	0.53333333	0.25	0.93333333
		RF	0.68	0.61666667	0.4	0.86666667
		KNN	0.6	0.38333333	0.3	0.8
		SVM	0.62	0.2	0.1	0.96666667
		DT	0.48	0.31904762	0.3	0.6
		AdaBoost	0.6	0.46333333	0.4	0.73333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('position: ', 'middle')]	2 Index(['Upper line angle', 'Lower line angle'], dtype='object')	LogReg	0.64	0.5	0.15	0.96666667
		RF	0.46	0.3	0.25	0.6
		KNN	0.52	0.45	0.2	0.73333333
		SVM	0.64	0.5	0.15	0.96666667
		DT	0.6	0.5	0.55	0.63333333
		AdaBoost	0.56	0.38666667	0.55	0.56666667
	3 Index(['Upper line angle', 'Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.66	0.7	0.2	0.96666667
		RF	0.44	0.26666667	0.15	0.63333333
		KNN	0.58	0.60666667	0.3	0.76666667
		SVM	0.64	0.5	0.15	0.96666667
		DT	0.52	0.43333333	0.45	0.56666667
		AdaBoost	0.44	0.28333333	0.3	0.53333333
[('scaler: ', StandardScaler()), ('folds: ', 5), ('position: ', 'end')]	2 Index(['Upper line angle', 'Delta angle'], dtype='object')	LogReg	0.52	0.05	0.05	0.83333333
		RF	0.46	0.33333333	0.2	0.63333333
		KNN	0.46	0.28333333	0.25	0.6
		SVM	0.54	0	0	0.9
		DT	0.44	0.34	0.25	0.56666667
		AdaBoost	0.48	0.35	0.35	0.56666667
	3 Index(['Upper line angle', 'Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.52	0.05	0.05	0.83333333
		RF	0.46	0.36380952	0.3	0.56666667
		KNN	0.4	0.13333333	0.1	0.6
		SVM	0.54	0	0	0.9
		DT	0.36	0.30714286	0.35	0.36666667
		AdaBoost	0.46	0.46857143	0.25	0.6

Appendix I

Results of the micrography angles PL copy task

settings	features	model	accuracy	precision	sensitivity	specificity
[('scaler: ', StandardScaler()), ('folds: ', 5), ('position: ', 'start')]	2 Index(['Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.607273	0.2	0.11666667	0.9047619
		RF	0.469091	0.2	0.15	0.65714286
		KNN	0.465455	0.28333333	0.2	0.62857143
		SVM	0.607273	0	0	0.97142857
		DT	0.430909	0.16666667	0.2	0.56190476
		AdaBoost	0.467273	0.25	0.25	0.5952381
	3 Index(['Upper line angle', 'Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.607273	0.2	0.11666667	0.9047619
		RF	0.489091	0.27333333	0.26666667	0.62857143
		KNN	0.525455	0.3	0.1	0.78095238
		SVM	0.607273	0	0	0.97142857
		DT	0.410909	0.2	0.2	0.53333333
		AdaBoost	0.414545	0.2	0.3	0.46666667
[('scaler: ', StandardScaler()), ('folds: ', 5), ('position: ', 'middle')]	2 Index(['Upper line angle', 'Delta angle'], dtype='object')	LogReg	0.667273	0.6	0.16666667	0.96666667
		RF	0.629091	0.64666667	0.31666667	0.81904762
		KNN	0.585455	0.43333333	0.21666667	0.80952381
		SVM	0.570909	0	0	0.90952381
		DT	0.590909	0.53333333	0.38333333	0.72857143
		AdaBoost	0.590909	0.5	0.41666667	0.6952381
	3 Index(['Upper line angle', 'Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.685455	0.8	0.21666667	0.96666667
		RF	0.607273	0.48333333	0.36666667	0.74761905
		KNN	0.549091	0.3	0.21666667	0.75238095
		SVM	0.570909	0	0	0.90952381
		DT	0.570909	0.46666667	0.43333333	0.65714286
		AdaBoost	0.549091	0.42	0.53333333	0.55714286
[('scaler: ', StandardScaler()), ('folds: ', 5), ('position: ', 'end')]	2 Index(['Upper line angle', 'Lower line angle'], dtype='object')	LogReg	0.567273	0	0	0.9
		RF	0.585455	0.41333333	0.35	0.71428571
		KNN	0.549091	0.42	0.36666667	0.66190476
		SVM	0.587273	0.15	0.11666667	0.88095238
		DT	0.469091	0.25	0.26666667	0.58571429
		AdaBoost	0.565455	0.36666667	0.25	0.75238095
	3 Index(['Upper line angle', 'Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.567273	0	0	0.9
		RF	0.529091	0.26666667	0.25	0.68571429
		KNN	0.449091	0.18333333	0.21666667	0.58095238
		SVM	0.569091	0.1	0.05	0.88095238
		DT	0.550909	0.47666667	0.36666667	0.64761905
		AdaBoost	0.469091	0.3	0.35	0.52857143

Appendix J

Results of the micrography angles PL continue task

settings	features	model	accuracy	precision	sensitivity	specificity
[('scaler: ', StandardScaler()), ('folds: ', 5), ('position: ', 'start')]	2 Index(['Upper line angle', 'Delta angle'], dtype='object')	LogReg	0.544444	0.05	0.05	0.83333333
		RF	0.437778	0.3	0.18333333	0.6
		KNN	0.457778	0.28666667	0.33333333	0.53333333
		SVM	0.564444	0.06666667	0.06666667	0.86666667
		DT	0.413333	0.18	0.23333333	0.53333333
		AdaBoost	0.52	0.45666667	0.28333333	0.66666667
	3 Index(['Upper line angle', 'Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.524444	0.04	0.05	0.8
		RF	0.56	0.38333333	0.28333333	0.73333333
		KNN	0.417778	0.18333333	0.18333333	0.56666667
		SVM	0.564444	0.06666667	0.06666667	0.86666667
		DT	0.54	0.44666667	0.4	0.63333333
		AdaBoost	0.566667	0.41666667	0.35	0.7
[('scaler: ', StandardScaler()), ('folds: ', 5), ('position: ', 'middle')]	2 Index(['Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.604444	0.1	0.1	0.9
		RF	0.711111	0.73333333	0.46666667	0.86666667
		KNN	0.542222	0.43	0.5	0.56666667
		SVM	0.626667	0.3	0.18333333	0.9
		DT	0.642222	0.6	0.31666667	0.83333333
		AdaBoost	0.584444	0.35	0.28333333	0.76666667
	3 Index(['Upper line angle', 'Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.604444	0.1	0.1	0.9
		RF	0.606667	0.53333333	0.4	0.73333333
		KNN	0.54	0.37333333	0.31666667	0.66666667
		SVM	0.62	0.53333333	0.2	0.86666667
		DT	0.602222	0.56	0.45	0.7
		AdaBoost	0.542222	0.47333333	0.28333333	0.7
[('scaler: ', StandardScaler()), ('folds: ', 5), ('position: ', 'end')]	2 Index(['Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.646667	0.26666667	0.15	0.93333333
		RF	0.586667	0.3	0.26666667	0.76666667
		KNN	0.646667	0.52	0.38333333	0.8
		SVM	0.58	0.2	0.05	0.9
		DT	0.586667	0.48	0.45	0.66666667
		AdaBoost	0.526667	0.48	0.38333333	0.6
	3 Index(['Upper line angle', 'Lower line angle', 'Delta angle'], dtype='object')	LogReg	0.646667	0.26666667	0.15	0.93333333
		RF	0.624444	0.58333333	0.43333333	0.73333333
		KNN	0.691111	0.63333333	0.45	0.83333333
		SVM	0.537778	0.1	0.05	0.83333333
		DT	0.54	0.40666667	0.55	0.53333333
		AdaBoost	0.497778	0.28666667	0.31666667	0.6