XAI$_{\text{Text}}$: A Domain-Specific Language for Developing an AI Pipeline

Master thesis

Author: Håvard Brynjulfsen
Supervisor: Fazle Rabbi

University of Bergen
The Department of Information Science and Media Studies

June 15, 2021

Acknowledgements

I want to extend gratitude to all my coworkers at Dips Front in Bergen. Everything I have learned at the job provided me with the tools to create the software presented in this thesis from the ground up. I would also like to extend my gratitude to Fazle Rabbi, for his feedback. For our discussion of ideas and his guidance through this process from start to finish. Fazle was believed to be one of the best supervisors by the students, and he did not disappoint. Finally, I would also like to thank the participants of the INTROMAT project for providing data.

Håvard Brynjulfsen
Bergen Norway, 15.06.2021

Abstract

AI has a wide range of applicability and uses in the healthcare sector spanning everything from drug-drug interaction (*Zhou et al.*, 2018), depression(*Garcia-Ceja et al.*, 2018a) and prostate cancer (*Pantanowitz et al.*, 2020). Explainable artificial intelligence - XAI has seen growth in the past years. With an increasing number of articles coming out every year (*Barredo Arrieta et al.*, 2020), since the GDPR (General Data Protection Regulation) puts stringent demands on the algorithm's ability to explain its decisions to participants(*Goodman and Flaxman*, 2017). In addition, multiple fields use XAI, presenting an explanation for participants to legally use machine learning algorithm's (*Barredo Arrieta et al.*, 2020). Thus the area of AI research has become more tailored towards XAI. The field of XAI already has many good explanations and visualization techniques. With everything from sensitivity analysis of what factors produces what kind of output to bar charts that explain the weights in a regressor. These techniques can visually present the algorithm's decisions made to a user group. The field still lacks an easy and flexible way to customize a domain-specific language (DSL) for problems related to XAI.

The thesis suggests creating an entire pipeline by weaving time-series, preprocessing, feature extraction, machine learning algorithms, user explanations, and algorithm tweaking inside a DSL. Then, applying this to the XAI field to extract features, train an algorithm, and explain the algorithm's decision to a broader audience. The XAI_{Text} software was created through multiple iterations and testing on two datasets from the INTROMAT project, the depression dataset containing data from a depression study and the ADHD dataset containing data from an ADHD intervention study. XAI_{Text} can deploy this pipeline through a domain-specific language based on a model-driven engineering approach.

Through this easily deployable domain-specific language with state of the art time-series feature extraction, the pipeline achieved a weighted F1 score of 0,83 and an MCC of 0.68 on the depression dataset. Compared to the best previous scores of 0.73 and 0.44. The software demonstrated visually how a simplified version of the SVM algorithm worked by allowing real-time tweaking of two input factors.

It proved pragmatically viable and time-saving to deploy the DSL on both the ADHD and depression datasets. Thus, demonstrating the prospect and potential of a rapidly deployable pipeline for solving XAI problems.

Contents

List of Figures

Chapter 1

Introduction

Artificial intelligence has a wide range of healthcare domain applications. Significant areas include cancer, neurology and cardiology (*McCarthy*, 2007). With the demand for decision support systems in healthcare (*McCarthy*, 2007) and the stringent demands of the GDPR(*Goodman and Flaxman*, 2017), there has been an increase in the demand for explainable artificial intelligence - XAI(*Barredo Arrieta et al.*, 2020). The purpose of XAI is to increase the capacity of explainability for AI (artificial intelligence) outcomes. XAI is crucial for the healthcare domain as the decision-makers need to make decisions that involve patients' lives. Given these factors, XAI has seen an increase in recent years(*Barredo Arrieta et al.*, 2020). With articles in the XAI spanning everything from explaining artificial intelligence, visualising artificial intelligence and how multiple machine learning algorithms can be made explainable to a broader audience.

1.1 Motivation

The main problem concerning XAI is how to make the AI algorithms more explainable. Under certain circumstances, an AI technique called machine learning performs well. It can generate new insight, gives us new knowledge, and help us solve problems in the healthcare domain. Moreover, more and more data is now time-series based, and we can gather data on a scale never seen before (*Sharma*, 2015): through watches, social media, and smartphones. We can help patients solve health problems like irregular heart rhythm, depression, and diabetes with machine learning from all this data. The most pressing issue is that if we are to use these machine learning algorithms. We will have to explain to a broader audience how these algorithms work and how they made their decision, transparency being the key.
Domain-specific languages (DSL) simplify programming complexity as it raises the level of abstraction. They are specialised computer languages designed for a domain. Like any other general-purpose programming language, domain-specific languages use grammar for the construction of their syntax rules. Many programming projects usually deploy many DSLs to achieve their goals. (*Fowler*, 2010). The key reason for doing so is the pro-

ductivity gains from deploying them. (*Fowler*, 2010). They make it easier to communicate with experts and make things easier for domain experts who want to contribute (*Fowler*, 2010).
Searching for scientific articles in scholar.google.com keyword "XAI + DSL" returns 654 results. However, in the hundred first hits, there are no relevant articles. The only relevant, up-to-date article is "The next evolution of MDE: a seamless integration of machine learning into domain modelling" (*Hartmann et al.*, 2017). Found by searching for "MDE + machine learning". This article illustrates experiments with machine learning in domain modelling. Given the lacking research on the topic, utilising DSL in XAI seems like an untapped area of novelty for the XAI field.

1.2 Contribution

The contribution to the XAI field is a novel pipeline built using a model-driven engineering approach (MDE). It contains what is needed to train a machine learning algorithm and explain it: tweak two factors on it, visualise it, handle the input data, present explanations to users, and allow researchers to visualise the machine learning algorithm and explore user inputs. The pipeline is helpful because it is a tool a researcher can deploy relatively efficiently over a dataset with user inputs. It solves a machine learning-related problem while at the same time being able to explain the algorithm's decisions to participants. The pipeline has two logically distinguished parts. The first part will consist of a Domain-Specific grammar containing an entire machine learning pipeline that is deployable onto an XAI problem. The second part is the deployed pipeline. The evaluation of the second part is what kind of score this pipeline can produce on a dataset and what utility this pipeline will create for researchers and users.
Due to time constraints, the domain-specific language (DSL) had to narrow itself into a small DSL grammar containing the essentials for a deployable XAI analysis and user explanation tool. It is not very comprehensive but has enough features for a proof of concept stage or minimal usage stage with the options in the DSL grammar. The pipeline is deployable with two machine learning algorithms, SVM and LogisticRegression. It can present an explanatory machine learning visualisation to users. It contains many tweak-able parts that are also customisable in the grammar.
For the analysis stage, there is the possibility to view every time-series plot from users. A code segment not fully implemented in the pipeline showed that it could export this time-series plot into GraphML format. This time-series was then importable into popular graph visualisation tools such as Gephi. The scalar in the machine learning algorithm is customisable in the DSL. There is also the option to choose from time-series or non time-series data. The pipeline was deployed and tested on two datasets from the INTROMAT project, the depression dataset containing data from a depression study and the ADHD dataset containing data from an ADHD intervention study.

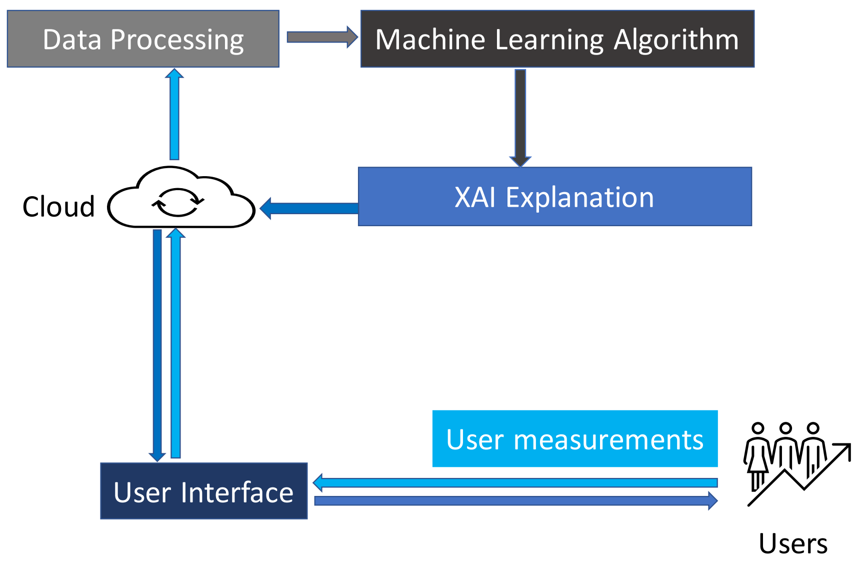


Figure 1.1: Overview of the programming artefacts (Generated from the deployment of MDE components)

Figure 1.1 shows the entire program once deployed through the DSL. The bottom left begins with the users, which uploads time-series user data that the program takes in. From there, the program organises the data and uploads it. Next, the program takes in this data and trains a machine learning algorithm on it. Finally, the program creates explainable information that both the users and researchers can use.

1.3 Availability of the project

The contribution of the master's thesis project has been made available as an open-source project which the researcher community can use. The project is available at the author's GitHub repository; see appendix D for the full web address. Contributors can improve it, with endless possibilities regarding machine learning models, tweak ability, and DSL features. New features can be easily implemented in the DSL grammar. The pipeline can easily be deployed for any project within certain constraints. As it stands now, researchers interested in deploying the pipeline can do so with the ability to utilise an SVM analysis tool. This tool can inspect every time-series point and produce an explanation to end-users that contributed with their data. Many libraries and tools have proved to be very good at visualisation, machine learning, and data handling. The DSL brings together many of these tools, which can have a sizeable pragmatic value for researchers and programmers.

1.4 Research questions

Research question 1:
"How can we increase the application of explainable AI utilizing a model-driven engineering approach?"

Research question 2:
"How can the area of XAI be improved by interpretation and visualization techniques?"

1.5 Thesis outline

Chapter 2 Theory
This section outlines the theoretical foundation of the thesis. It discusses XAI, machine learning, model-driven engineering, time-series, visualisations, similarity graphs and graph theory.

Chapter 3 Related work
This chapter outlines and shows articles which contain work that is related to or relevant to the thesis.

Chapter 4 Data
The data used for the thesis consists of 2 datasets. One is the publicly available depression dataset, while the other is the ADHD dataset. This section introduces both datasets. In addition, the section also introduces the Infromat project and the thesis's contribution to it.

Chapter 5 Software development
This chapter outlines the technologies and their use in the creation of the program. It outlines the development process from start to finish is. It illustrates the significant problems encountered and their resolution.

Chapter 6 Analysis
The chapter first presents an example of creating a model with the XAI_{Text} DSL grammar. Furthermore, it illustrates how the graphical user interface looks and behaves with a deployed pipeline on the depression dataset.

Chapter 7 Results
This chapter presents the results from both datasets as cross-validation scores, illustrates the GUI, usability, a user explanation and a Gephy graph.

Chapter 8 Discussion and Conclusion
This section outlines and deliberates on the results achieved, discussing the problems and issues that arose. It contains ideas for improvements and the potential of the pipeline. This chapter also concludes the project.

Chapter 2

Theory

2.1 AI

2.1.1 Overview - AI in healthcare

Artificial intelligence research started after WW2, thought of as the science and engineering behind intelligent machines (*McCarthy*, 2007). Its thought of as being methods that pertain to understanding human intelligence (*McCarthy*, 2007).

AI applications are, among others, game playing, speech recognition, understanding natural language, computer vision, expert systems, heuristic classification (*McCarthy*, 2007) and more. Various branches of AI include logical AI, search AI, pattern recognition, representation, inference, common sense knowledge and reasoning, learning from experience, planning, epistemology, heuristics, and genetic programming.

As outlined in *McCarthy* (2007) intelligence is the goal-directed computational part responsible for achieving goals. It varies between individuals, humans and machines. The problem with the construct of intelligence is that we do not know what kind of computational mechanisms we can classify as intelligent or not. Machines as we know them today cannot necessarily classify as either intelligent or not intelligent. Machines can perform well on specific tasks but cannot function the same way humans do. So it would be correct to call them somewhat intelligent. The Turing test is a test to verify if we have created an intelligent machine. The test says that if a person manages to fool a human being into believing a machine is human, That person should consider the machine intelligent. Still, this is an open philosophical question since a machine could be intelligent but fail to imitate humans.

The healthcare systems have been using AI to diagnose and cure diseases since the 1970's (*Davenport and Kalakota*, 2019). As outlined in *Amisha et al.* (2019) AI is growing and becoming a more extensive part of the healthcare industry every year. AI in healthcare divides into two aspects, Virtual and physical. Examples of the virtual being a patient journal and examples of the physical being robots assisting a surgeon. Numerous smartwatches and health care trackers like Fitbit and apple allow continuous monitoring of various patient inputs, uploading information to doctors for treatment and diagnosis. AI is used to diagnose healthcare workflow and system analysis for more efficient therapies and workflow. AI is a tool for demanding things in doctors work, like giving a prognosis of the adverse effects of multidrug prescriptions. Even the traditional role of the doctors, sitting down and asking multiple questions about the state and symptoms of a patient's health, can be given to an AI. Which, based on a whole database of potential illnesses, concludes a diagnosis. The only problem here is cues that only the doctor can observe; AI also challenges this aspect by improving image recognition technology. As outlined in *Davenport and Kalakota* (2019) the future potential of AI in healthcare is multifold. This future consists of having AI perform phone calls and conversations with patients, improved robots helping with surgeries and stitching. Better early warning detection and predictions of disease based on big data and AI. More personalised and contextual care. Care based on smartwatches, biosensors, and smartwatches to guide patients into better coping methods.

2.1.2 Machine learning

Outlined in *Andreas C. Müller* (2016) machine learning is the study of algorithms that learns or improve through input and testing. The key point behind machine learning is that machine learning is a tool for extracting knowledge from data. DNA sequence analysis, recommender systems, and personalised cancer treatments, among others, all use machine learning techniques. (*Andreas C. Müller*, 2016). Facebook uses techniques from machine learning together with Amazon and Netflix, which is why people probably interact with machine learning models daily. The field is said to be a field where computer science, statistics, and artificial intelligence meet. (*Andreas C. Müller*, 2016)

2.1.3 Support Vector Machine

The pipeline presented in this thesis will primarily use the Support Vector Machine (SVM) machine learning technique, an excellent machine learning technique visualised in a 2d space with two input factors. It takes in various inputs, which are used for classification. The mathematical basis behind SVM is a technique that plots different points from different classes in a graph. From there, we draw a boundary between the two clusters of points. By making one line the mainline with two symmetrical margins on opposite sides of the mainline, these two margins go through the nearest points of each class. The points are called support vectors. Hence the name "support vector machines." The whole point with the mathematics behind the SVM approach is to maximise the size of these two margins. The mathematical tools utilised for this approach are algebra, vector calculus, and Lagrange optimisation. (*Fletcher*, 2009) Figure 2.1 shows an SVM model with a boundary drawn between two classes.

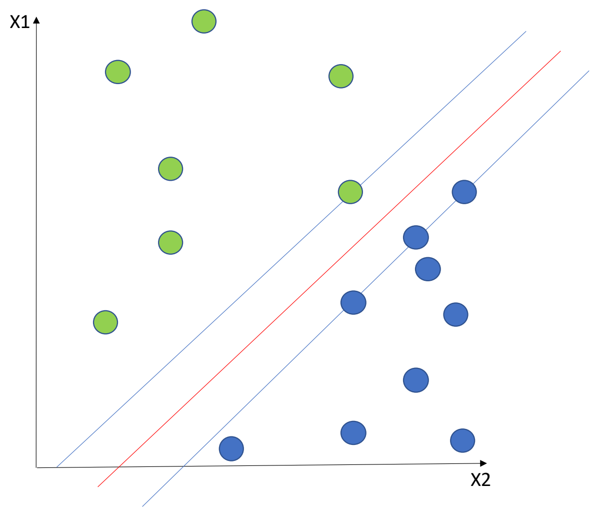


Figure 2.1: SVM algorithm

2.1.4 Feature

A feature means an attribute that is selected. Feature engineering means combining features into new ones.(*Andreas C. Müller*, 2016, p. 213) An example of a feature can be height, weight, and age. An engineered feature could be BMI derived from height and weight.

2.1.5 Scaling

Scaling is when a feature, which might otherwise vary significantly, is standardised on a standard scale.(*Andreas C. Müller*, 2016, p.134-142) An example being a feature with minimum and maximum values of 4 and 1000000, getting its place on a scale from 0-1. Making everything much easier for extracting features and training machine learning algorithms as extreme number differences can create unnecessary problems and confusion.

2.1.6 Classification

Classification means classifying something as either "a" or "b", giving it a class label. It means putting a definite and non floating value on something. We can use binary classification, or we can use multi-class classification. For example, Binary being two possible values and multi-class being more than two.(*Andreas C. Müller*, 2016, p.27-28)

2.1.7 Supervised/Unsupervised learning

Supervised learning consists of a machine learning algorithm learning from inputs with a given output classification. Algorithms use these input/output pairs to predict the value of new inputs. Unsupervised learning is the phenomenon where the algorithms try to find new patterns in unclassified data.(*Andreas C. Müller*, 2016, p.2-3)

2.1.8 Generalization

As outlined in (*Andreas C. Müller*, 2016, p.17-18) generalisation is the phenomenon where the trained algorithm is usable on other data. Poor generalisation means that the algorithm cannot utilise predictions on anything other than its predictive value on the domain trained on successfully. Generalisation divides itself into two sub phenomena.

Underfitting

Underfitting is weakly tailoring the model to the trained data. The model will be very generalisable to new datasets but will not be very accurate.

Overfitting

Overfitting is heavily tailoring the model to the dataset. Meaning that the model cannot accurately predict a range of different datasets since it is not very generalisable.

2.1.9 Regularization

Regularisation is when we usually put a penalty on a machine-learning algorithm to reduce the chance of it overfitting. The critical insight of regularisation is that we reduce the complexity of a model to make it less likely to overfit.(*Andreas C. Müller*, 2016, p.51)

2.1.10 Cross-Validation

Cross-validation is a technique used for evaluating the generalised performance of a machine learning model on a dataset.(*Andreas C. Müller*, 2016, p.258-259) It functions by training and testing the model several times. Each time shuffling the test and training data around, we use more parts of the data for testing and more parts of the data for training through the whole process. The total score obtained is an average of all the models trained. By shuffling the training and test data around, we make sure that we do not end up with a non-generalisable result. Because we got lucky or unlucky when selecting the test data.

2.1.11 AI Limitations

The limitations of AI is that implementing, using and explaining them to a broader audience can be difficult. Using machine learning takes technical knowledge and skill. Understanding how machine learning algorithms work and make their decisions also takes much knowledge.

2.2 XAI

2.2.1 Introduction

Explainable artificial intelligence is a strand of artificial intelligence that explains how the algorithm came to its conclusion and classification. The field has had an upsurge in recent years due to society's stringent demands on utilising AI systems in society (*Barredo Arrieta et al.*, 2020). The critical key here is the emergence of a need for trustworthy, understandable, and manageable AI systems for society. (*Barredo Arrieta et al.*, 2020) XAI alleviates the limitations of traditional AI by aiming to increase the degree of explainability of an AI model. The limitations of XAI is the added work needed to create XAI systems since we have to design in explanations and transparency.

2.2.2 Definitions

Explainability in this context is outlined in (*Barredo Arrieta et al.*, 2020, p.5) and consists of the following XAI terms:

Understandability

Understandability means how easy it is for human beings to understand how the algorithm functions and decide without illustrating and explaining its inner workings.

Comprehensibility

The ability of the model to present the knowledge is has attained in a fashion that is understandable to humans.

Interpretability

Ability to provide the meaning in an understandable human way. With terms and explanations being comprehensible for humans.

Explainability

In this context, explainability denotes the link or interface between humans and a decision-maker. Meaning the interface acts as a proxy for the decision-maker (the algorithm), in a way that is understandable to humans.

Transparency

The model is transparent if, in and of itself, it is understandable. *Lipton* (2016) categorise models into "simulatable models, decomposable models, and algorithmically transparent models."

Of the five above mentioned facets of XAI, the first one, understandability is the most important one(*Barredo Arrieta et al.*, 2020, p.5)

2.2.3 Ways of explaining a model

Particular models are more straightforward to explain than others but may provide lower accuracy for the task at hand. Deep learning is the most difficult to explain due to its complexity, and rule-based learning is the easiest due to its simplicity. (*Barredo Arrieta et al.*, 2020, p.12) outlines six ways the field of XAI uses for explaining an AI model. These techniques used standalone or in conjunction provides explainability towards the model. Outlining the techniques below:

Text explanations

Text explanations mean that we are spelling out in text how the algorithm worked. Included in these text explanations are symbols for method generation that illustrates how the algorithm worked.

Visual explanation

Visual explanations present the algorithm's function visually. They usually include simplifications and dimensionality reduction. They are often coupled with other techniques to explain a machine learning model. Considered the best way to introduce machine learning to people who have not been introduced to machine learning algorithms.

Local explanations

Local explanations take the entire machine learning problem and divide them into multiple problem spaces that together make up the whole. Furthermore, From there, explaining each problem space locally.

Explanations by example

Explanations by example mean that the explanation takes the correlations and relationships within the model and represents them as examples towards an audience. Much like humans do when trying to explain unknown knowledge or information towards one another.

Explanations by simplification

These are all the techniques where we build a simplified model of the original model to have a close resemblance and score of the original model. In its simplicity, it is less difficult to explain. With this goal achieved, the unintended side effect is that this new model is easier to implement due to its lower complexity.

Feature relevance explanation

Feature relevance is explanations thought of as indirect post-hoc techniques for explaining models. The techniques take each input and perform a sensitivity analysis or feature relevance technique to map out how important each input was for the model's output.

2.2.4 Visualizations

Visualisation techniques mean a visual explanation of how an AI algorithm made its decision. In XAI, visualisation techniques centres around feature relevance techniques. (*Barredo Arrieta et al.*, 2020) The most common techniques for drawing out and explaining the most relevant factors in a machine learning model are sensitivity analysis techniques. They measure the output when varying the input and has proven its value as a technique for explaining an ML model. (*Cortez and Embrechts*, 2013) Illustrates Many different sensitivity analyses outlined as Data-based SA, Cluster-based SA, Monte Carlo SA. (*Goldstein et al.*, 2013) Outlines Individual Conditional Expectation.

List of common visualisation techniques used for black-box models in XAI:

- Data-based SA
- Cluster-based SA
- Monte Carlo SA
- Individual Conditional Expectation
- Bar charts
- Vectors
- Correlation map
- Vectors
- Contour maps

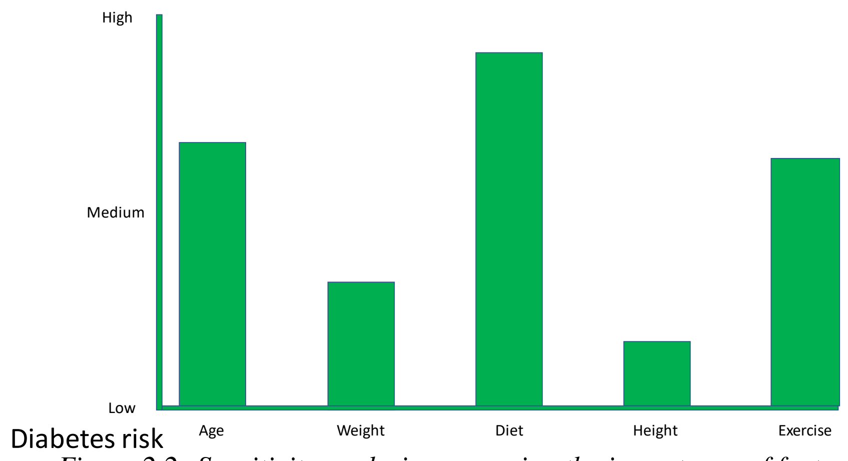


Figure 2.2: Sensitivity analysis, measuring the importance of factors

Figure 2.2 shows an example of sensitivity analysis. The result achieved shows us how important the various factors in the figure are regarding diabetes risk.

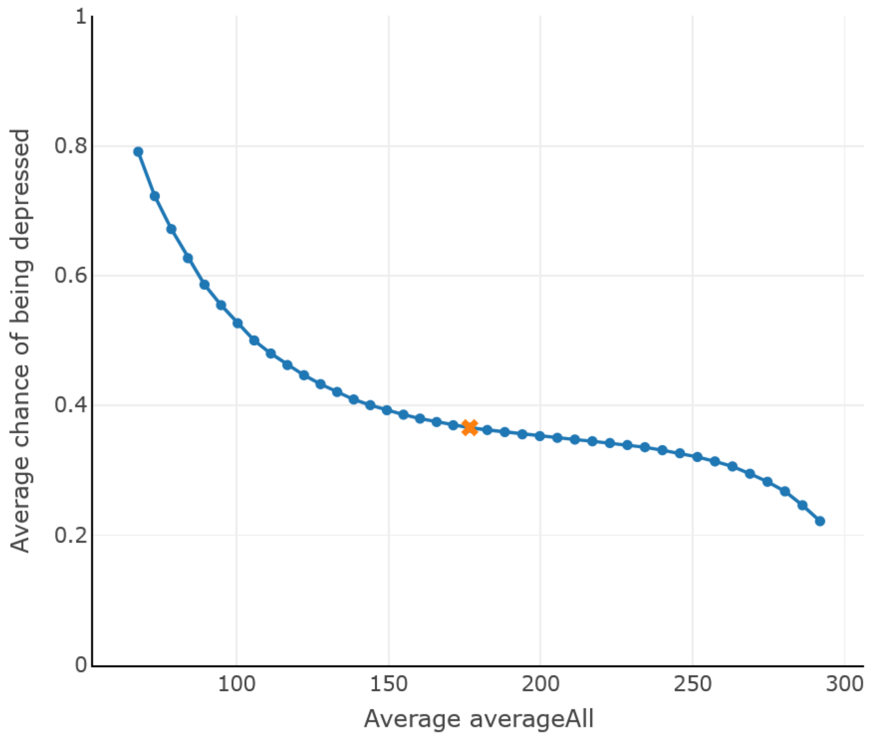


Figure 2.3: Sensitivity analysis over the range of a factor

Figure 2.3 shows us an example of what happens when we run a sensitivity analysis on a particular factor and measure the change to the classification chance.

2.3 Model-driven engineering

2.3.1 Definitions

The MDE approach, defined as model-driven engineering, utilises domain models in the design process. *Gurunule and Nashipudimath* (2015) describe domain models as conceptualisations of all the topics, features, or concepts related to a problem sphere. The approach attempts to conceptualise the problem abstractly rather than in a programmed concrete instance of the necessary algorithms and implementations. Described in *Fowler* (2010) It is an approach where we can auto-generate code and implement the same concepts over many domains. Domain-specific languages (DSL) has been around for a long time. They are languages tailored to the specific domain, with a grammar that functions as pre-assembled building blocks the programmer can use to buildup his code. A perfect example would be CSS, where CSS is just the styling implemented over the domain of HTML. The lines between a DSL and a general programming language are often blurry. An example is Ruby on rails, usually considered a domain-specific language, even though it is easy to think of it as a generalised programming language. The critical point is that a DSL is not Turing complete. Meaning it cannot express everything the same way an entire generalized Turing complete language like Python can.

2.3.2 Advantages

With a DSL, it is easy to read and understand the whole concept we are trying to illustrate by modelling abstract concepts orderly and organised. Once modelled in an MDE approach, using the same model in different domains is possible by simply changing the relevant domain concepts(*Fowler*, 2010). In the book *Fowler* (2010), the author outlines the critical point about DSLs: it can potentially increase the productivity of programmers by implementing them over a domain.
The second key point outlined in (*Fowler*, 2010) is the ease of understanding a DSL. Easy-to-understand DSLs give domain-specific experts the ability to improve and contribute code instead of just pure programmers. It provides benefits for both the domain experts and the programmer working towards solving a problem.
The third point in (*Fowler*, 2010) is that with DSLs, presenting the work of the programmers to potential clients can make the presentation more understandable. This presentation is beneficial for someone that otherwise might not understand anything of the code without a programming background. The strength of the DSL is a function of its presentation value to outsiders.

2.3.3 Disadvantages

The first problem with DSLs is the cost accosted with building, maintaining, and deploying a DSL. If the cost outweighs the gain by making a DSL, it has not served its intended function. A problem might occur if the developers of the DSL accidentally make it into a general-purpose programming language, thus undermining their effort at creating a pragmatically viable DSL. Further problems can arise when applying DSLs to a context where it is unfeasible. (*Fowler*, 2010). There is also the pitfall of building things that you may get from the outside (*Fowler*, 2010) or simply adding too much syntactic sugar towards your DSL (*Fowler*, 2010).

2.3.4 Model-driven engineering in XAI

The reasons for combining Model-driven engineering(MDE) and XAI is that MDE can fix the problems of implementing XAI. This thesis created a DSL pipeline that is easily deployable compared to typing out every code when making an XAI application. Combining MDE with XAI is because the MDE approach can enhance the explaining power of XAI. Multiple ways of explaining and showcasing a model are possible as pre-defined DSL parts with the MDE approach. These deployable parts give programmers and researcher a tool to increase the level of abstraction of the XAI domain for a gain in efficiency when creating XAI applications. Creating an interface between humans, machine learning algorithms and the tools needed to explain them.

2.4 Graph Theory

2.4.1 Definitions

Outlined in *tutorialspoint* (2021) Graph theory is a popular subject used in both computer science and mathematics, defined as "the study of graphs that concerns the relationship among edges and vertices" (*tutorialspoint*, 2021). A graph consists of vertices and edges. A point is a particular position on the graph, and edges are the lines that connect the vertices. A line means a connection between two points. The point where multiple lines meet is a vertex. The line connecting two vertices is an edge. A graph is a set of edges and vertices. A loop is just an edge drawn from a vertex to itself. Graph theory can illustrate the connectives of computers in a computer network. It illustrates different algorithms in computer science, including Kruskal's Algorithm, Prim's Algorithm, and Dijkstra's Algorithm. Figure 2.4 shows a node, 2.5 an edge and 2.6 a graph.

Figure 2.4: Node

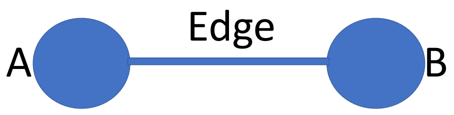


Figure 2.5: Line between two nodes

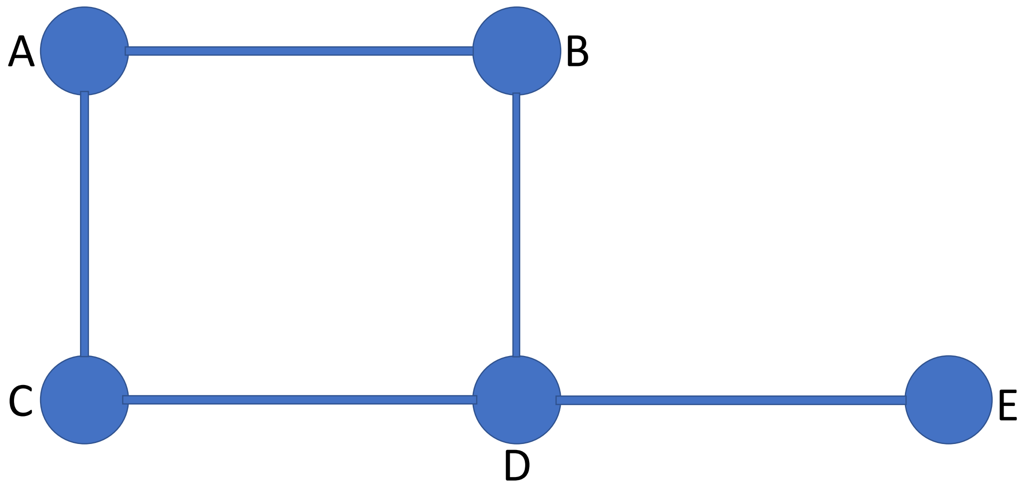


Figure 2.6: Graph with edges and nodes

2.4.2 Visibility algorithm

Núñez et al. (2012) Outlines the idea behind the visibility algorithm. It is a technique for mapping time-series into graph theory. Bridging time-series into graph theory allows us to use graph theory techniques on time-series data.

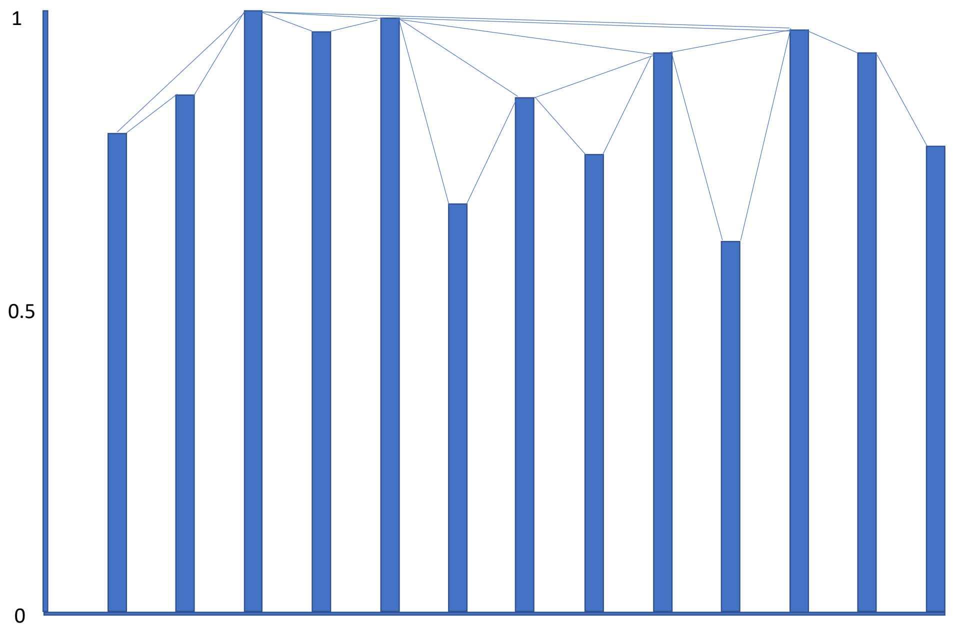


Figure 2.7: Visibility algorithm

The visibility algorithm is an algorithm that functions by creating a line between two points for every point in a time-series as shown in figure 2.7. If the points are visible for each other, There must be a visible line between both points. If one time-series point gets in the way of this visibility, The line between the two points not visible to each other gets omitted. *Núñez et al.* (2012) draws out four premises for the visibility algorithm "(i) connected: each node sees at least its nearest neighbours (left-hand side and right-hand side).
(ii) undirected: the way the algorithm is built up, there is no direction defined in the links.
(iii) invariant under affine transformations of the series data: the visibility criterium is invariant under rescaling of both horizontal and vertical axis, as well as under horizontal and vertical translations.
(iv) "lossy ": some information regarding the time-series is inevitably lost in the mapping from the fact that the network structure is completely determined in the (binary) adjacency matrix. For instance, two periodic series with the same period as T1=..., 3, 1, 3, 1, ... and T2=..., 3, 2, 3, 2, ... would have the same visibility graph, albeit being quantitatively different."

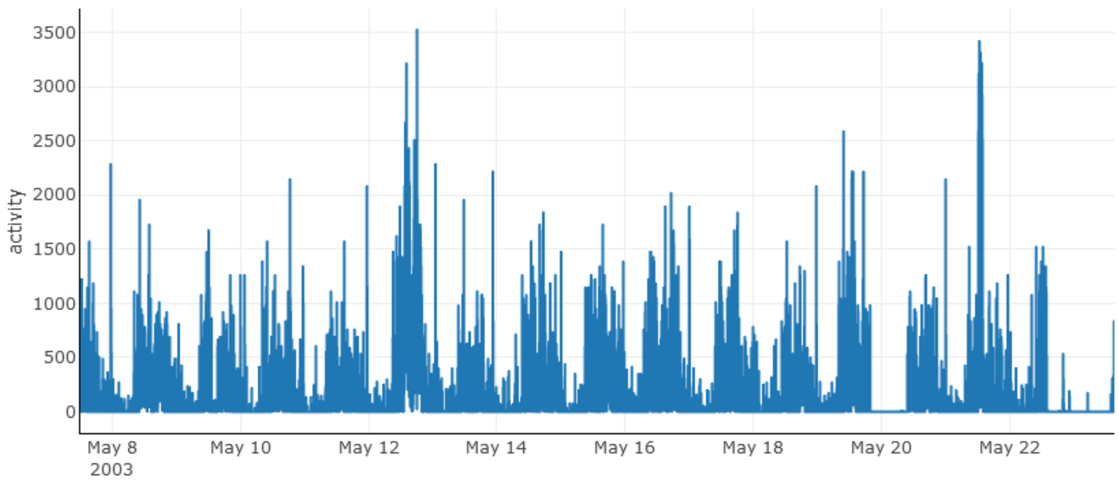


Figure 2.8: Time-series

2.5 Time-series

With the advent of big data, time-series has become more relevant. Because time-series data in real-time data can be mined and exploited for AI purposes, spanning everything from heart rates to stock exchanges. This time-series data can be presented and analysed for various purposes.

Time-series means that the data is in a format where the entire time-series with timestamps presents itself from beginning to end as shown in figure 2.8.

Researchers in (*Núñez et al.*, 2012) illustrated that it is possible to take a time-series and apply the visibility algorithm upon this time-series. It is possible to utilise the features generated from the time-series graph.

Chapter 3

Related Work

3.1 MDE in Machine learning

purpose of the study

In *Hartmann et al.* (2017), the researchers illustrated that it was possible to weave machine learning into a DSL seamlessly. With a novel approach centred around microlearning units and use it for a potential boost in prediction accuracy. There stated goal in the research report was to predict individual and global power consumption.

Theory of the study

Hartmann et al. (2017) makes a distinction between fine-grained and coarse-grained learning. Fine-grained learning is when we extract and infer knowledge on a detailed small scale level. Coarse-grained learning aggregates knowledge from multiple sources. Combining coarse-grained and fine-grained learning compared to only using coarse-grained learning can be a better result. This type of learning is especially relevant in IoT (the internet of things) when multiple small units work individually and together.

Hartmann et al. (2017) define microlearning units as "reusable, chainable, and independently computable elements". The purpose of the microlearning units is to infer and derive the necessary attributes for machine learning algorithms deployed into the meta-model. The microlearning units are the core element of the approach presented in *Hartmann et al.* (2017).

The meta-model is the model representing all necessary elements for implementing the approach. A code generator can generate code in Java, Scala, or Go with the meta-model.

The meta-meta model represents what goes into the Meta-model. The DSL's function is to be a tool for creating this meta-meta model.

Execution and results

What *Hartmann et al.* (2017) did in their study was to deploy a DSL over the domain of power consumption. Once deployed, this DSL would use microlearning units, inferring and gathering all the necessary data to gather attributes and parameters for machine learning algorithms. Afterwards, creating a meta-model from the DSL and the inferred information from the microlearning units, using this Meta-model as a model for generating code in Java, Scala or C++. The Domain-specific language of the research report is an extension of a domain-specific language defined in *Fouquet et al.* (2014)
Hartmann et al. (2017) deduced that their microlearning approach achieved an accuracy of 85 per cent versus accuracy of 72 per cent for coarse-grained learning. The method proved its utility, especially over an IoT type domain.

3.2 The visibility graph

Lacasa et al. (2008) presents the visibility algorithm. The article is from 2008, and *Lacasa et al.* (2008) outline how to make the visibility algorithm from time-series. *Lacasa et al.* (2008) show how to take the visibility algorithm and apply analysis tools on the created graph. The method is computationally inexpensive. The whole point of the article is that with the creation of the visibility graph, a whole new dimension of analysis of time-series is possible. *Lacasa et al.* (2008) also suggest many more avenues of research based on their method.

3.3 Visibility Algorithms: A Short Review

Núñez et al. (2012) is a state of the art article from 2012 making a summary of how time-series features is used through the visibility algorithm. It shows all the work built around the authors from *Lacasa et al.* (2008). It takes, as its basis, the technique of the visibility algorithm outlined in the theory section. The whole point of the article is to present an overview as to how to use the visibility algorithm on time-series and apply graph theory for analysis. The article discusses different ways of applying graph theory in this domain and the challenges faced. The article lists several different methods for mapping time-series into graphs. Below is a list from *Núñez et al.* (2012) of some of the most important ones:

- "mapping each cycle of a pseudo periodic time-series into a node in a graph."
- "relative frequencies of appearance of four-node motifs inside a particular graph."
- "technique based on the properties of recurrence in the phase space of a dynamical system."

- "a surjective mapping which admits an inverse operation."
- "Horizontal visibility algorithm."
- "Directed horizontal visibility graph."

Núñez et al. (2012) give many mathematical backgrounds and in-depth explanations of applying graph theory can to time-series. To extract information and explore the abilities of the various techniques and graphs in different problem spaces. *Núñez et al.* (2012) gives suggestions towards more research into these problem spaces and how the focus of research might be in order to solve them.

3.4 Graph theory applied to schizophrenia and depression

purpose of the study

In *Fasmer et al.* (2018), a group consisting of 24 psychotic patients and 23 patients with mood disorders, all currently depressed, were used. In the study, *Fasmer et al.* (2018) had a control group consisting of 18 women and 11 men.
What *Fasmer et al.* (2018) had available of data was actigraph registrations of each participants motor activity spanning over 12 days, obtained from watches that registered activity. This data was converted into graphs, applying graph theory as an analysis tool on the extracted graphs. The study hoped that their new and novel similarity graph would display different characteristics for depressed and schizophrenic patients compared to the control group. Conducting various experiments and methods for creating and analysing the graph to find any differences between the participants in the study.

Results and conclusions

Fasmer et al. (2018) results showed a marked difference between the control group versus the schizophrenic and depressed patients. There was also a significant difference between schizophrenic and depressed patients. Another discovery was that the depressed patients had more complexity in their time-series.

3.4.1 Machine learning applied to depression

purpose of the study

The participants in *Garcia-Ceja et al.* (2018b) were 23 bipolar and unipolar depressed patients with 32 healthy controls. Five hospitalised subjects were part of the depressed group. The purpose of the study was to have patients and the control group wear actigraph watches and measure their activity during an average of 12.6 days. From these measurements, *Garcia-Ceja et al.* (2018b) used statistical techniques to extract features and train machine learning models to classify subjects as either depressed or non-depressed. The research in *Garcia-Ceja et al.* (2018b) aims to classify depression based on activity levels from the actigraph watches. The study built upon similar research with a lot of the digital appliances used as measurement devices. Below follows a list of previous findings from *Garcia-Ceja et al.* (2018b) that the study builds upon:

- "They found that physical activity was reduced in adults with Late-life depression compared to healthy controls and showed slower fine motor movements."
- "They found that in patients with bipolar disorder the more severe the depressive symptoms, the less answered incoming calls, fewer outgoing calls and patients moved less."
- "Used smartphone data to classify depressed and manic states in bipolar patients achieving a 76 per cent recognition accuracy."
- "Used audio, motor activity and questionnaires to classify mood in bipolar patients with an accuracy of 85 per cent."
- "Correctly identified 70 per cent of all depressed cases with Random Forest from uploaded photos to Instagram."
- "This systematic review identified that a depressive state is associated with reduced daytime motor-activity, as well as an increased nighttime activity when comparing to healthy controls."
- "This systematic review identified that reduced motor activity is associated with bipolar depressions, besides increased variability in activity levels compared to healthy controls."

Garcia-Ceja et al. (2018b) shows the potential of monitoring technology for the healthcare sector. With all the previous research attesting to the ability of digital technology in gathering data and performing a diagnosis related classification. *Garcia-Ceja et al.* (2018b) also elaborates on depression as it is increasing in society. It is a diagnosis that hampers the willingness to engage in activities, socialise, and have a positive outlook on life. It breeds anxiety, feelings of emptiness and low self-worth. In its worst cases, it can create suicidal feelings. Previous research has shown that depressed people have increased nighttime activity and decreased daytime activity. Depression correlates with alcohol and drug abuse. *Garcia-Ceja et al.* (2018b) shows how it is possible to extract statistical features used as training data for a machine-learning algorithm. Conducting machine learning classification with both the Random Forest algorithm and the Deep neural network model.

Results and conclusions

The authors of *Garcia-Ceja et al.* (2018b) achieved a weighted F1 score of 0.73 and an MCC of 0.44. In terms of classification, the Random forest outperformed the deep neural network model. DNN was better at detecting non-depression than random forest.

3.5 Sensitivity analysis on black box models

Cortez and Embrechts (2011) explains how particular models, such as neural networks, support vector machine and ensembles, help make accurate predictions. However, they are challenging to explain to an audience. *Cortez and Embrechts* (2011) explain that the explanation power of sensitivity analysis can be thought of by how easy it is for humans to understand. *Cortez and Embrechts* (2011) suggests a novel visualisation approach based on sensitivity analysis. The visualisation technique functions by varying the input of a factor used in the machine learning model and measuring the corresponding output. With this technique, it is possible to rank the input according to its importance. Primarily SA has been used for feature selection. However, it also has its value in being a valuable tool for explaining black-box models. By showcasing their global sensitivity analysis algorithm and the visualisation technique, the Variable Effect Characteristic (VEC) curve can open up any black-box model for explanation purposes. With this technique, *Cortez and Embrechts* (2011) manage to explain datasets by showing what factors were the most important and showcasing how these various factors were crucial during the whole input spectrum of that factor.

3.6 Further sensitivity analysis on black box models

Cortez and Embrechts (2013) continue their work from (*Cortez and Embrechts*, 2011). They reiterate that some models like SVM and deep neural networks, ensembles, random forests, and kernel-based methods are complicated to explain to an audience. For explaining these kinds of models, rule extraction or visualisation are the preferred techniques. The problem with rule extraction is that they risk simplifying the model to where crucial information about its inner workings is lost. *Cortez and Embrechts* (2013) suggests sensitivity techniques as a way to explain these black-box models. *Cortez and Embrechts* (2013) presents a multitude of these techniques for handling both regression and classification tasks. The SA, methods and techniques outlined by *Cortez and Embrechts* (2013) are in the list below.

- "novel and computationally efficient DSA."
- "novel and computationally efficient MSA."
- "novel and computationally efficient CSA."
- "comparing DSA, MSA and CSA with 1D-SA and GSA."

- "a new SA measure of input importance (AAD)."
- "AAD is tested against three other measures."
- "adapting the SA methods and measures for handling discrete variables and classification tasks."
- "novel functions for aggregating multiple sensitivity responses."
- "3-metric aggregation for 1D regression analysis."
- "fast aggregation strategy for input pair (2D) analysis."
- "new synthetic datasets."
- "input importance bars."
- "colour matrix."
- "variable effect characteristic curve."
- "surface and contour."
- "explore the black box model NN."
- "explore the black box model SVM."
- "explore the black box model RF."
- "white box model decision tree."
- "Show how SA can open up black box models on four real-world tasks."

The authors of *Cortez and Embrechts* (2013) shows a plethora of SA techniques and methods for opening up black-box models. *Cortez and Embrechts* (2013) also point out the importance of computational expense when using SA models. *Cortez and Embrechts* (2013) suggests that future research uses more real-world data and presents the SA analysis within a graphical user interface.

Chapter 4

Data

4.1 Introduction

To show the effectiveness of the proposed method in this thesis, we have performed experiments with healthcare datasets. The datasets are from a project called INTROMAT. The INTROMAT project abbreviates "Introducing personalised Treatment Of Mental health problems using Adaptive Technology project". The project intends to use technologies and psychological interventions to handle the burden of psychological illnesses. The collaborators for the project include domain experts, project managers, postdoctoral researchers, PhD fellows and other project members.
During the thesis development, we held meetings and showcases with the stakeholders who contributed to datasets. Both datasets are from previous and current work on the INTROMAT project, which contained data from individual participants in a research setting. The ADHD dataset needed usage permission.
The depression dataset (*Garcia-Ceja et al.*, 2018b) is publicly available, used in many research reports from the Infomat project and contain time-series data. Reasons for choosing the datasets are that there were possibilities to try out different machine learning techniques, improve feature extraction, and achieve a better score than previously achieved. It was also a good case for experimenting with deploying the pipeline through the DSL and presenting a user explanation to the participants.
The ADHD dataset contains warnings given to users, how often they logged on and if they completed their training modules or not. Using this dataset was because the pipeline was an analysis tool deployable over the data. Therefore, it could show the efficacy of the pipeline as an analysis tool.
The thesis and its attachment to the project showcased the DSL pipeline as a tool for current and potential research. Furthermore, to obtain better results than previously achieved and doing novel experiments relevant to the INTROMAT project.

4.1.1 INTROMAT project owners

INTROMAT is one of three projects chosen by the Norwegian research council through their "IKTPLUSS Lighthouse call"(*Haukeland-University-Hospital*, 2021a). Haukeland University hospital owns the project. It builds on the premise that the prevalence of anxiety and depression in the population stands at 20-25 per cent lifetime occurrence of depression and 8 per cent current anxiety or depression. The project develops interactive and adaptable technology with flexible treatment modules for mental health issues. The project also develops scalable infrastructure to support its effort. The project originates from eMeistring (*Haukeland-University-Hospital*, 2021b).

4.1.2 INTROMAT participants

Participants in the project include domain experts, project managers, postdoctoral researchers, PhD fellows and other project members. Partners of the project include Attensi, Bryggen research, CheckWare, Explorable, Helse Bergen, Helse vest ikt, Høgskolen på vestlandet, IBM, Imatis, Modumbad, Telenor, UiB, UiO, and Youwell.

4.1.3 Goals of INTROMAT

The goals of INTROMAT is to use Information and communications technologies (ICT) to improve the mental health of the public. Therefore, they want to create relevant technologies and infrastructure for this task. Furthermore, the goal is for the developed technologies to be novel, interactive and adaptive.
Hindrances and obstacles these goals face are many. There is low recognition of ICT techniques for treating and assessing mental health patients. There is a lack of knowledge around ICT's efficacy in the healthcare domain. The systems in the healthcare sector also need to be integrated with these new technologies and solutions. User-friendly interfaces that engage users and clever new ways of procuring data are still lacking.

4.1.4 What has INTROMAT achieved

The INTROMAT project has published numerous reports since 2016. The published reports span everything from ethics, data visualisation patterns, applying machine learning on time-series data and artificial intelligence. In addition, publishing multiple master degrees with subjects and work relative to the project. The media has also mentioned the project in numerous articles.(*Haukeland-University-Hospital*, 2021a)

4.2 Project participation

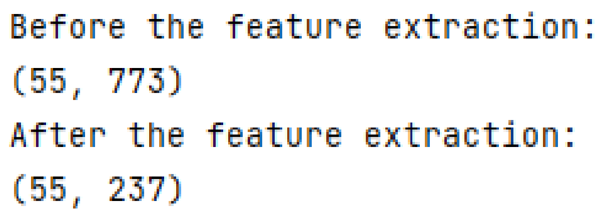
The participation and contribution towards the INTROMAT project are numerous. The most obvious contribution was towards gaining a better score than previously achieved on the depression dataset. In addition, trying on a new machine learning algorithm and extracting time-series features for the dataset. For the ADHD dataset, novelty included creating an artificial time-series that the feature extractor could utilise. Furthermore, the thesis showed how the SVM algorithm could present data for users and researchers contributing to a research project. Moreover, demonstrating the user, handler, researcher as a DSL framework for the XAI field. Furthermore, it created suggestions for interactive dashboards that researchers and users can use. Moreover, collaboration and presentation towards project contributors was an ongoing process. Finally, through a professional experience in DIPS, the Norwegian ICT healthcare firm, I achieved synergy between my work and master's thesis research. As a result, the $\mathrm{XAI}_{\mathrm{Text}}$ DSL is usable on both future and current projects. Used alone to deploy a pipeline, be expanded or used as a scaffolding framework.

4.3 Depression dataset

The depression dataset is a publicly available dataset containing the activity measurements of 55 patients (*Garcia-Ceja et al.*, 2018b). Twenty-three are depressed, and 32 control subjects are not depressed; these measures, taken during several weeks, with the measurements presented as time-series data. In addition, varying factors of the subject are offered, such as gender, bipolar, and age. The idea behind the dataset is that depressed people have different activity measurements of activity, especially daytime rhythms caused by depression, than non-depressed people. (*Garcia-Ceja et al.*, 2018b) The best results achieved from the depression dataset are a weighted F1 score of 0,73 and an MCC of 0.44. (*Garcia-Ceja et al.*, 2018b) Appendix B presents technical details of data handling for the dataset.

4.4 Variables and their usage

The classification variable in the depression dataset is if the contributor is either depressed or not depressed. Measurements from the participants are in time-series data. The feature extracting software Tsfresh uses this time-series data to generate features. Meaning we start with a user, classified as either depressed or non-depressed. When Tsfresh has done its job, we have a user classified as either depressed or non-depressed with several associated time-series features. Adjusting the number and complexity of these time-series features in Tsfresh is efficiently done according to our needs. The training variable is all these extracted time-series features associated with a user. Using these time-series features on a machine learning algorithm lets the algorithm decide which features are the most important and discard unimportant ones. This approach is the crucial strength of Tsfresh. Extract time-series features in the hundreds, with every kind of mathematical feature extracted. From there, it filters down the most important ones, using them as the basis for training a good machine learning algorithm.



```
Before the feature extraction:
(55, 773)
After the feature extraction:
(55, 237)
```

Figure 4.1: After extraction features for all 55 participants, we narrow them down from 773 to 237 per user.

Figure 4.1 shows the feature filtering process using a linear support vector classifier. The classifier decides what features are the most important and discard the least important ones.

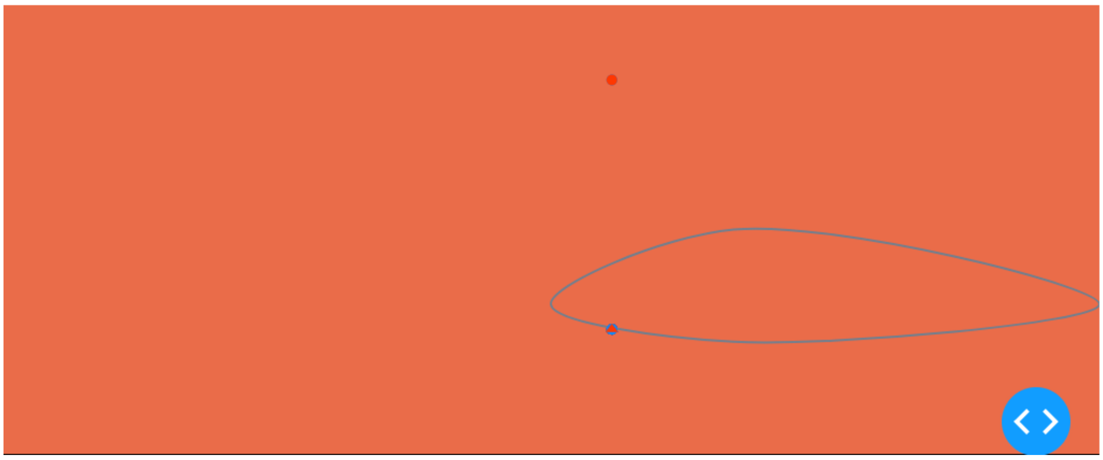


Figure 4.2: The SVM, when we use two meaningless variables to train it.

The SVM in the Dash Plotly graphical user interface is only trainable on two features simultaneously. Because of how the SVM functions. Meaning in 2d space, we can only visualise two factors at the same time. We could potentially use three factors if we had a 3d space to picture them in, but any more than that would be impossible to visualise in this way. We can create an SVM with more than three factors, but we cant visualise all of them at once. In figure 4.2, we have used two features, "dim0varianceLargerThan-StandardDeviation" and "dim0HasDuplicateMax" extracted from Tsfresh. Since none of these two features has any predictive value, we end up with an SVM with all the points dotted on top of each other and a meaningless boundary. Throughout the rest of the thesis, we will present how we can fine-tune the performance of classification algorithms through a DSL.

4.5 ADHD dataset and study

4.5.1 ADHD dataset

The dataset consisted of willing participants over the age of 18. Further requirements were that the participants had access to a smartphone, computer and the internet. The participants needed to speak, write and read Norwegian. Exclusion criteria consisted of current standing psychiatric disorders, like substance abuse, psychosis or suicidal ideation. The number of participants available for analysis after the researcher had run the study was 125. Appendix B presents technical details of data handling for the dataset.

4.5.2 Purpose of study

The idea of the study relevant to the dataset was to figure out if there were any effects of internet-based interventions to help ADHD patients cope with their symptoms. In addition, the idea was to give the participants warnings to see if this had any influence on their completion and participation rates in the study.

4.5.3 Measurements of the study

From the research protocol, the study methods aimed to measure included in the dataset are as follows.

Primary measure:

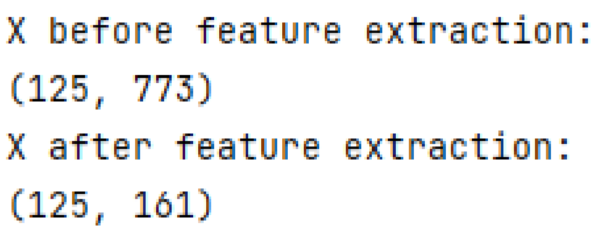
- Adherence (completed modules);
- Participant feedback regarding self-reported engagement;

Secondary clinical outcomes:

- Inattention and hyperactivity/impulsivity measured by two - Subscales from the Adult ADHD Self-Rating Scale (ASRS);
- quality of life measured by Adult ADHD Quality of Life Measure (AAQol);
- Stress measured by the Perceived Stress Scale (PSS)
- Executive functioning measured with Behavior Rating Inventory of Executive Function (BRIEF), and self-compassion measured by the Self compassion-Scale (SCS)

4.6 Variables and their usage

In the ADHD dataset, the data is in a non time-series format. Therefore, deploying the timeline on the dataset was predicated on transforming the data into usable time-series data. Creating this time-series data was done by measuring the completion time per week for each user when doing a training module. The classification variable in this context is if the participant completed the last week of the training modules. The training variables are the features extracted from the feature extractor associated with a user. Figure 4.3 shows the feature filtering process using a linear support vector classifier the same way as figure 4.1.



```
X before feature extraction:
(125, 773)
X after feature extraction:
(125, 161)
```

Figure 4.3: After extraction features for all 125 participants, we narrow them down from 773 to 161 per user.

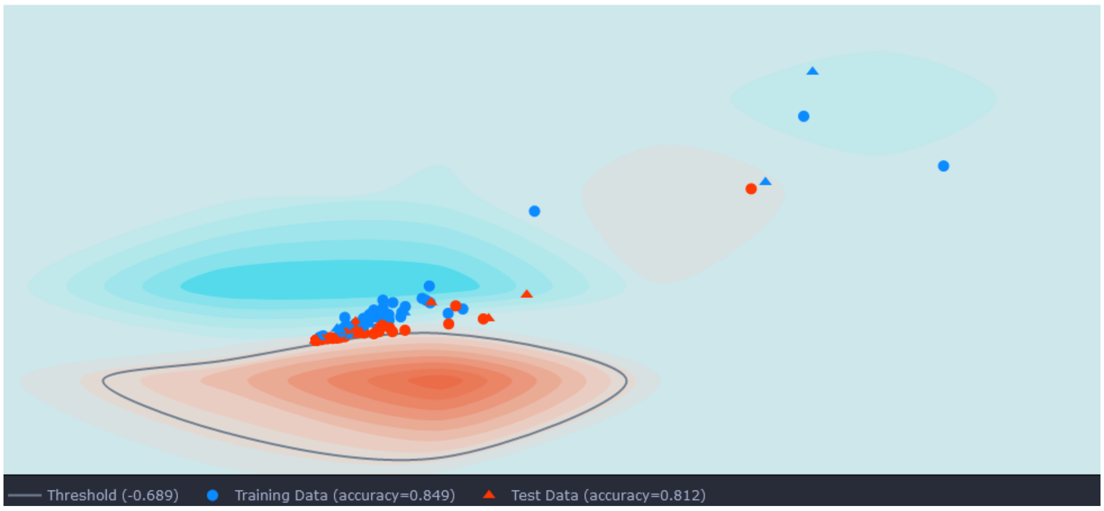


Figure 4.4: The SVM, when we use two meaningful variables to train on it.

Figure 4.4 shows us what happens when we use the two significant variables "Dim0maximum" and "dim0SumValues" to train an SVM algorithm. Here we can see that the two clusters spread out in an orderly pattern, allowing good classification to happen.

Chapter 5

Software Development

5.1 Proposed solution architecture

The creation of the entire program happened with Python in Pycharm community edition. By using Sklearn for deploying the SVM machine learning algorithm. Time-series was created in a readable manner using Pandas to splice and combine the time-series measurements. Tsfresh was used to extract time-series features, which the SVM machine learning algorithm from the Sklearn library could train a model on. The XAI_{Text} DSL language, created using TextX with Jinja2 for generating code from the templates. Finally, presenting all this in Plotly Dash, where the user or researcher could tweak and interact with the result.

5.1.1 Python

Python is a complete object-based programming language that is both high level and general purpose (*Python software foundation*, 2021). It is well suited for machine learning and data manipulation, given its extensive support for libraries such as pandas and Sklearn. The Python programming language ranks as one of the most used and popular.

5.1.2 Sklearn

Sklearn is a python library consisting of a wide plethora of machine learning models (*Pedregosa et al.*, 2011). It functions well and designed to be compatible with both NumPy and Schipy. Sklearn can integrate with well-known python modules such as Plotly, Matplotlib, pandas, and Tsfresh.

5.1.3 Tsfresh

Tsfresh is a module used for extracting time-series features(*Christ et al.*, 2016). It is suitable for extracting time-series features on a static dataset. It takes a time- series and extracts features like the mean, median, max value, min value, and the number of peaks. Then, Sklearn can train these features on a machine learning algorithm. Tsfresh also has transformer modules for Sklearn; Tsfresh can extract and filter features through a Sklearn pipeline. Tsfresh is not suitable for streaming data but can use a roll function for splicing more time-series data into the existing one.

5.2 Ethik

Ethik is a plugin for Python designed to aid in creating fair and explainable AI and designed to perform a sensitivity analysis. By taking a variable and change its input, measuring how this affects a machine learning algorithm (*Max Halford*, 2019). Ethik is usable for investigating a single variable and measure how influential this variable is.

5.2.1 Pycharm

Pycharm is an IDE for both professional and non-professional developers(Jetbrains, 2021). It includes several tools for python developers, git up-loading and the standard help functions from JetBrains when used as a plugin in other languages. It is a popular IDE featuring an intelligent code editor, smart code navigation, fast and safe refactorings, debugging, remote development, scientific tools, interactive python console, Conda integration, IPython Notebook integration and database tools.

5.2.2 Jinja 2

It is a templating engine used for creating and writing templates. It is designed for Python and is text-based (*Ronacher*, 2021). It throws exceptions to make debugging easier, allows inheritance, inclusion and has many more features. It is one of the most used templating engines compatible with Python. Combined with TextX models and a code generator, it can generate python code.

5.2.3 TextX

TextX is a domain-specific modelling language (*Dejanovi et al.*, 2017). It is inspired by but should not be confused with the similarly sounding Xtext, a domain-specific language designed for compatibility with the Java programming language. TextX is a domain-specific language designed for Python. What is possible to do in TextX is to create a DSL grammar. From this grammar created, it is possible to create models. These models can be used together with templating engines and code generators to create entire programs. Projects to help TextX with syntax highlighting, code outline and Language Server Protocol support for IDE's and editors are underway.

5.2.4 Plotly dash

With Plotly Dash, it is easy to create interactable and live updating graphs and illustrations (*Plotly*, 2021). The dashboards can build with Plotly Dash can be opened and illustrated in a browser.
The three pillars of Plotly Dash, which the whole concept builds upon, is as follows:
Dash components
These include everything from buttons to sliders to input boxes. They control and interact with the apps created. Plotly Dash has a whole library with these components.
Plotly graphs
These are the illustrative screens or graphs where the interactivity of Plotly Dash. It is possible to show 3d graphs, financial graphs, and medical graphs. Only the imagination limits what the illustrative capabilities of interactive graphs are.
Callback function
The callback function is the most central feature of Dash Plotly. It binds the buttons with the output of the graph, triggering all the underlying programming code the button was supposed to trigger.

5.2.5 Pandas

It is a python based library designed for manipulating and analysing data (*pandas development team*, 2020). It is very flexible and can change and manage data by a data frame object with integrated indexing. It is possible to join, discard and merge data with pandas. It is considered one of the best tools for data manipulation in Python.

5.3 Why the technologies were chosen

5.3.1 XAI technology needs

The needs of XAI are primarily good ways of explaining how machine learning made its decision to all participants. Therefore, the explanations should be interactable and presented in a way that is the easiest for each particular user to understand.
The other significant concern is having good support for AI and machine learning and, for example, accessing good machine learning libraries and manipulating data. Other concerns for developers in the XAI field are speedy and easily deployable apps. Libraries and technologies with high in between compatibility, preferably with everything under one programming language is a major advantage.

5.3.2 Technology choice

The initial plan was to develop the XAI capable DSL using the Eclipse-based toolset Xtext. Because it has a large user base, online tutorials, books, and a live creation editor. This editor provides feedback for DSL syntax errors while creating a model based on DSL grammar. Xtext is not Python-compatible but compatible with Java. Java does not have the same support for machine learning and data manipulation that Python has based on the libraries Pandas and Sklearn. A front-end technology to display the machine learning algorithm and explanations needed to fulfil the project's needs. There were alternatives for Java, and there are many technologies and tools available for this task, but Dash Plotly has several advantages. It is in Python, it is browser-based and works well with machine learning libraries and tools. The support, flexibility and interactability in Dash Plotly is a significant advantage. The need and practicality of having everything in one programming language was the principal decision behind choosing the Python-based technology stack. The technology stack selected helps create an XAI application. Most of these tools are compatible with each other as a design feature. There are many tools available to create an XAI application and solve problems in the domain. The solution architecture chosen is a solid example of a highly compatible technology stack capable of solving these problems. It is not perfect in all its single parts, but it is a reliable compromise and alternative.

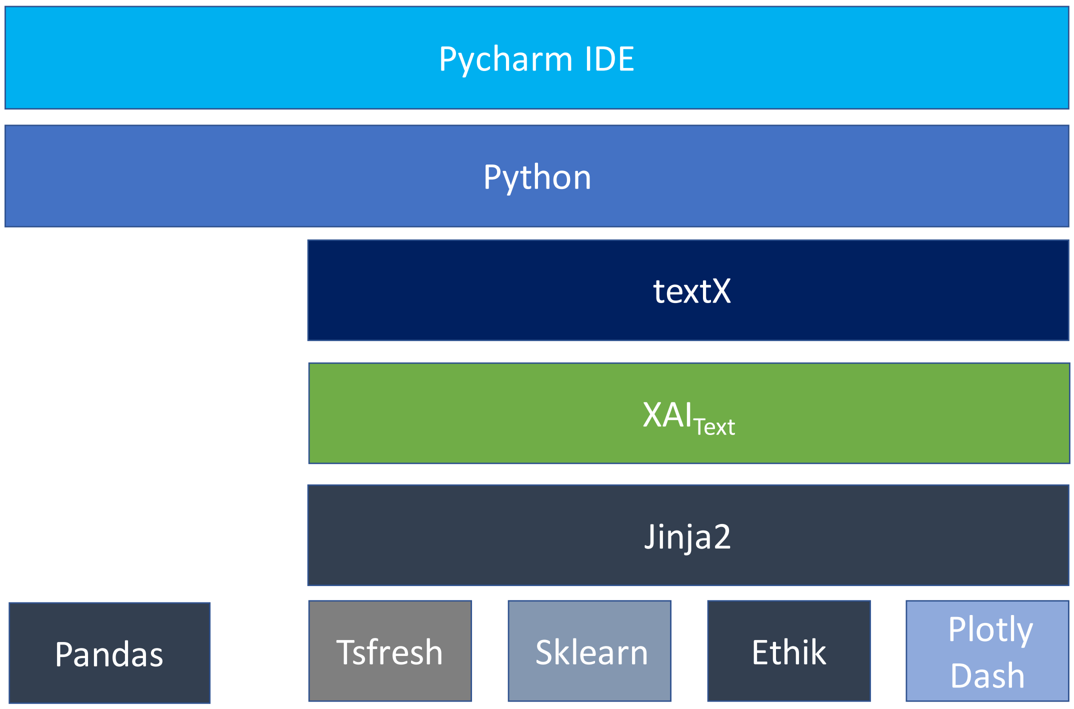


Figure 5.1: The solution architecture and how the technologies in it overlap

Figure 5.1 shows how everything is usable within Pycharm IDE. Furthermore, everything is in Python. For example, using Pandas for data manipulation, Tsfresh for time-series extraction, Sklearn for machine learning and Plotly Dash for outputting interactive dashboards. Tsfresh, Sklearn, and Plotly Dash are within the XAI$_{Text}$ DSL of TextX. Using Ethik for sensitivity analysis of the features extracted by Tsfresh. XAI$_{Text}$ creates the pipeline by writing out the model with Python and generating the code with a code generator and Jinja2 templates.

5.3.3 Bringing the technology under a DSL

The limitations of Sklearn, Dash Plotly and Tsfresh is that they are useless alone in this context. With a DSL, it is pretty easy to bring them all in under one umbrella. Python is a Turing complete expressive programming language that can express everything that needs programmatic expression. Within the confines of the XAI domain, particular chunks of code repeat themselves throughout applications, and there are bound to be certain parts that need modification the most. A DSL can bring these chunks of code together for an efficiency gain.
With the TextX DSL creation tool, it is possible to generate many lines with very few lines of code. For example, creating a tiny model in the XAI$_{Text}$ DSL with 20+ lines can produce over 1000+ lines of runnable python code.

5.4 Development

5.4.1 The goal of the final product

The goal of the final product was to create a pipeline consisting of three main Python classes: A handler class for reading and handling all the data, a user class for presenting an explanation for each user, and a research class for training and investigating the machine learning model in real-time.

5.4.2 Methods

Python 3.8 was the main interpreter used with Pycharm as the primary development tool. By using a TextX file within Pycharm to create the DSL grammar. The primary development methodology was to develop the program with informal sprints, going from 1 version to the next, in an ever-improving product result cycle. The supervisor and intromat participants gave feedback during development.

5.4.3 Constraints

Initially, there were development problems because the entire XAI field is incredibly vast with all sorts of machine learning algorithms and visualisation techniques. The restricted development time put limits on the project's scope to a proof of concept phase. This limit was for a DSL deployable as an XAI pipeline on the datasets. There were prioritising issues on the pipeline development, given the limited amount of time for the project. The solution was to prioritise the research class over the user and handler class. Because this class probably had the most central position in the program, with the most power to illustrate the pipeline's potential.

5.4.4 The program visualized

The deployment of the entire pipeline and different classes visualise as follows. First, in figure 5.2, a programmer uses the XAI$_{\text{Text}}$ DSL grammar to create a model. Then, with this model and templates, a code generator generates code. The three generated classes in this code are a user class, a handler class and a research class. The user class takes the specific user and uploads that users measurements as data the handler can understand. Next, the handler class takes in and transforms this user data into readable data for Tsfresh contained in the research class. Finally, the research class takes the data and trains a machine learning algorithm uploaded into Plotly Dash, creating explainable data through an interactive dashboard. This interactive dashboard can be used by the user class or the research class, fulfilling the explainability criterium for XAI.

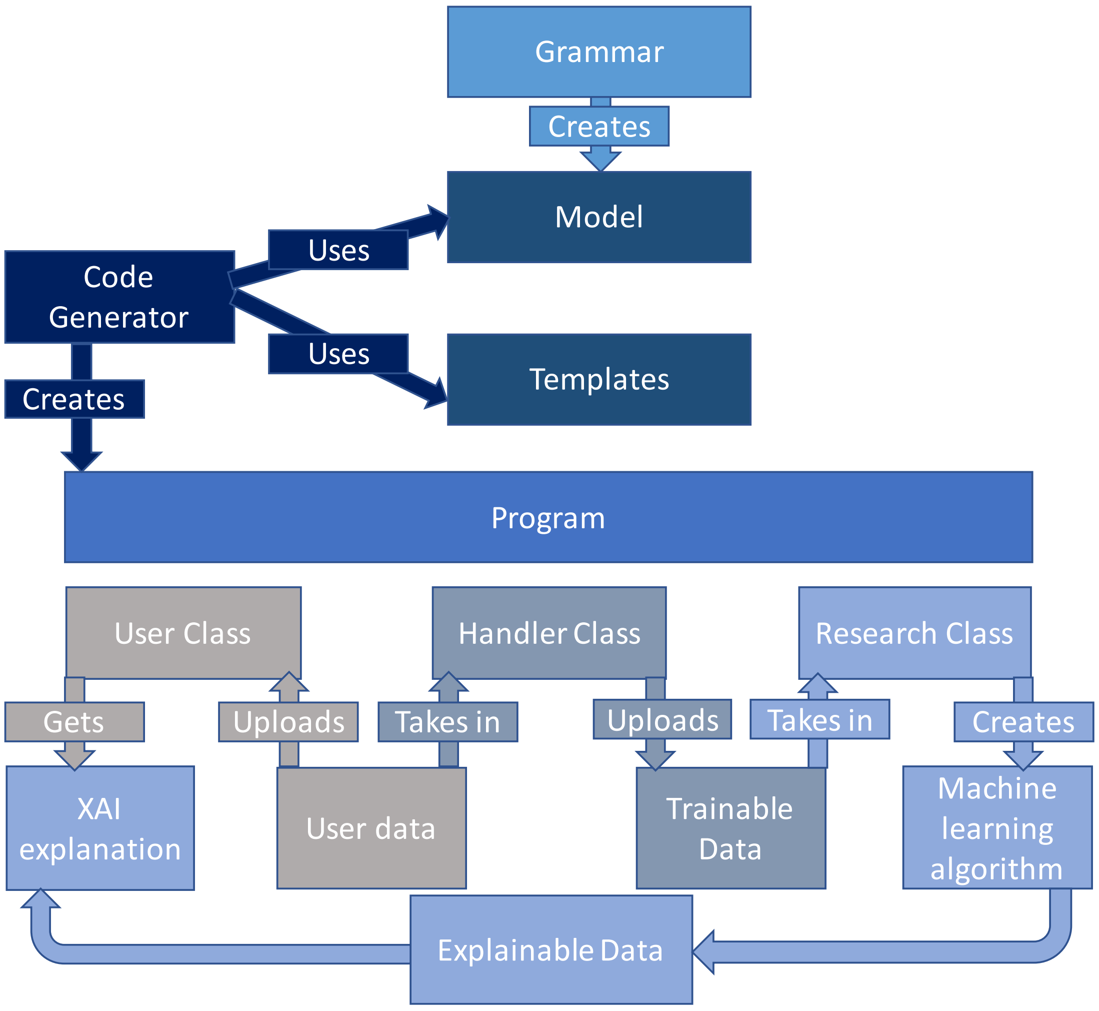


Figure 5.2: The grammar, code generation, and program in the same figure

5.4.5 Problems that needed to be solved

The first problem that needed to be solved was how to handle time-series data, exactly how to handle and parse the data. For this purpose, Pandas was

used to create time-series data that a potential handler could use. The next step was that the feature extractor Tsfresh could only use the data if all the time-series data chunks were of equal length. In the depression dataset, this is not the case. As the measurements between every user are different, some span a much longer time than others. The solution to this was to give each user a predefined time-series chunk given to the handler. Meaning that all time-series measurements from each user in the depression dataset would be of the same length.
For code generation and the DSL grammar, Xtext for Java was touted as the best alternative online for this task. Unfortunately, Python is more the gold standard for machine learning (Sklearn) and data splicing (pandas). Plotly Dash was also only compatible with Python. Fortunately, an equivalent to XText tool named TextX was found, which is compatible with Python. TextX has several features of the same features as Xtext.
Presenting the data to users and researchers, Plotly Dash was the preferred tool for building interactive graphs and dashboards. With Plotly Dash, it is possible to use callbacks, which means that we use input buttons that run a certain amount of code, which changes the output. The problem faced with Plotly Dash was around global variables. An example was when two callbacks got called simultaneously, where one changed a global variable that the other needed. Unfortunately, the one callback that needed this global variable did not pick up this change. The solution to this was to use async programming were putting a delay timer event on top of the callback that needed to get the global variable that the other callback had changed.

The software stands as a good analysis and productivity tool for researchers to deploy a wanting to deploy and tweak an SVM machine learning algorithm. The researcher can deploy the pipeline within the customisable parameters of the XAI_{Text} DSL grammar, giving him a tweakable tool to deploy over a dataset to study it and solve problems.

Chapter 6

Analysis

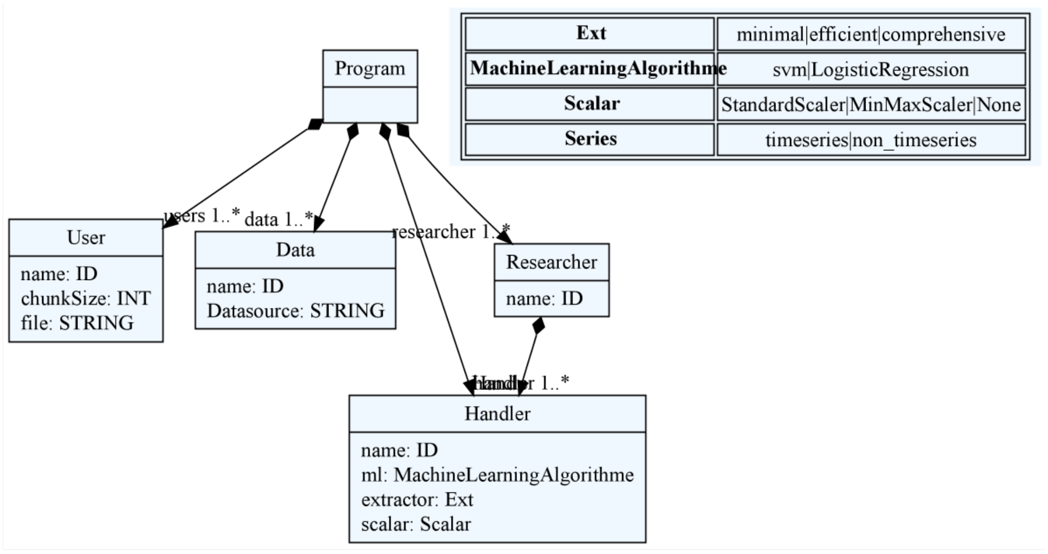
The following chapter will show an example of the deployment of the pipeline. By illustrating the XAI_{Text} DSL grammar available. The model created from the XAI_{Text} DSL grammar is described together with the generated classes and their functionality.

6.1 Grammar

A Domain-specific grammar is the building blocks, rules and syntax a domain-specific language uses to create a domain-specific model. The grammar allows us to develop a model of the domain within a set specified scope and boundary. The XAI_{Text} DSL grammar is in Appendix A. Anyone with a knowledge of TextX can use, modify or expand the grammar.

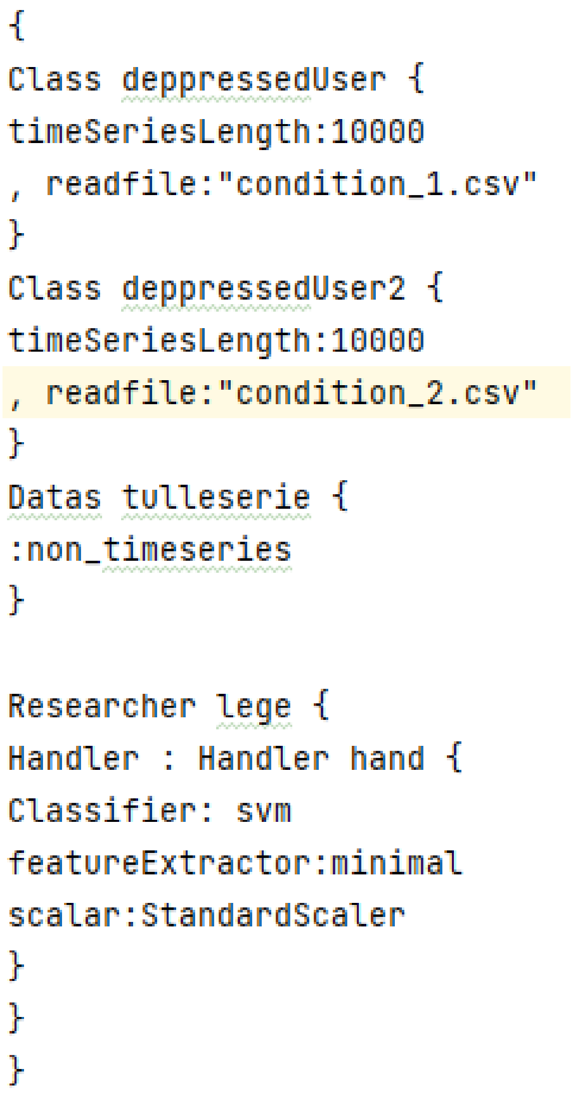
6.2 Model created from the grammar

Below in figure 6.1 is a simple .dot figure presented as a UML class diagram of the grammar, generated through the Pycharm command line with TextX, showcasing the XAI_{Text} DSL grammar. Figure 6.2 is a model we typed out with the syntax rules of our grammar. Finally, in figure 6.3, TextX generated a .dot file through the command line of Pycharm.



Ext	minimal|efficient|comprehensive
MachineLearningAlgorithme	svm|LogisticRegression
Scalar	StandardScaler|MinMaxScaler|None
Series	timeseries|non_timeseries

Figure 6.1: Dot modell showing the entire grammar of the XAI$_{Text}$ DSL



```
{
Class deppressedUser {
timeSeriesLength:10000
, readfile:"condition_1.csv"
}
Class deppressedUser2 {
timeSeriesLength:10000
, readfile:"condition_2.csv"
}
Datas tulleserie {
:non_timeseries
}

Researcher lege {
Handler : Handler hand {
Classifier: svm
featureExtractor:minimal
scalar:StandardScaler
}
}
}
```

Figure 6.2: How the model was written from the XAI$_{Text}$ DSL

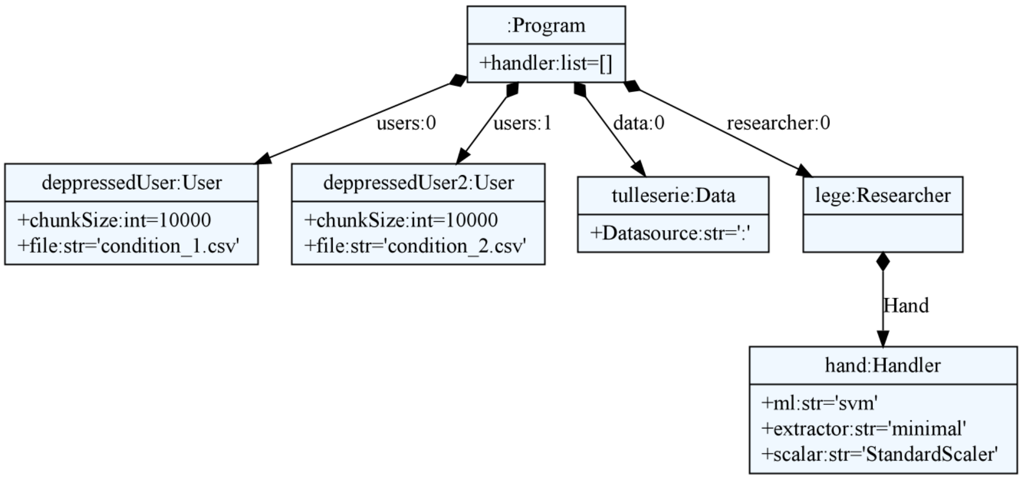


Figure 6.3: Dot file created from the written model

6.3 User class

We have generated two "User" classes from the model in figures 6.2 and 6.3 through a code generator based on the templates of our choosing. The main functionality of the user classes is to read a .csv file and upload the data from this .csv file as a text format file. This format makes all the text files sewable together to form a single .TS file by the handler. Ideally, every User class should have had many visual explanations to choose from in the XAI$_{\text{Text}}$ DSL grammar. Still, due to time constraints, only a single graphic description of an SVM model, the weight of its features and sensitivity analysis of input factors are available for the two users. Note that the user classes are not extracting the entire time-series but have to make due using only 10000 time-series points; the time-series has to be of equal length for Tsfresh to work.

6.4 Handler class

The handler class in figures 6.2 and 6.3 takes all the text files outputted by the User classes, combines them, and creates a .TS file. Because this is what the time-series features extractor Tsfresh needs to extract time-series features.

6.5 Research class

The research class in figures 6.2 and 6.3, generated through a code generator with a template, takes the .TS file and extracts features from it through Tsfresh's feature extractor. In this version of the model from the XAI$_{\text{Text}}$ DSL, we have decided that the research class contains a features extractor set to minimal feature extraction. The features are scaled to a standard scalar and trained through an SVM machine learning algorithm.

From there, the researcher can choose from and view the following GUI alternatives: choosing dataset, viewing individual time-series points, ROC curve, confusion matrix, threshold, cost(C), choosing feature and a retrain button.

6.6 Running example from the depression dataset

For readability, the following section will show a running example of the depression dataset. Figure 6.4 illustrates the ability to choose data sets. Note that this method remains an unimplemented method in the pipeline but implements itself quickly if needed.

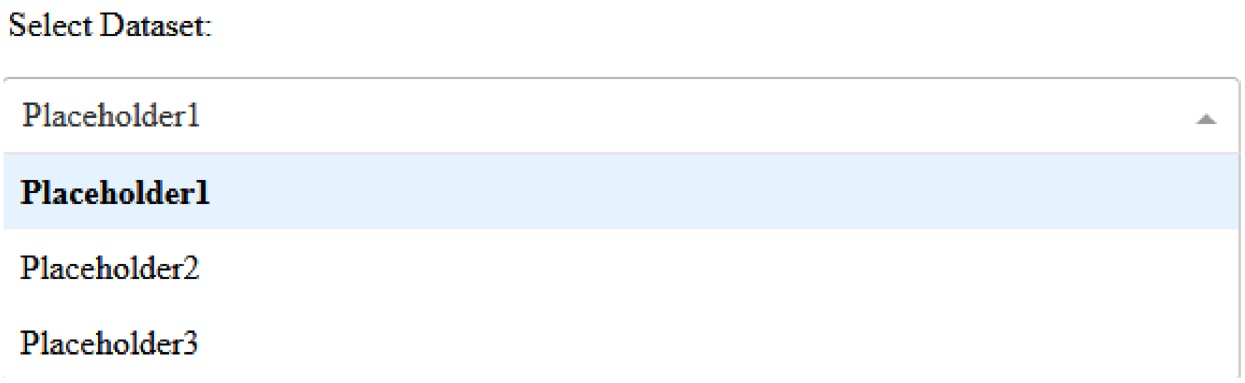


Figure 6.4: Selection of dataset

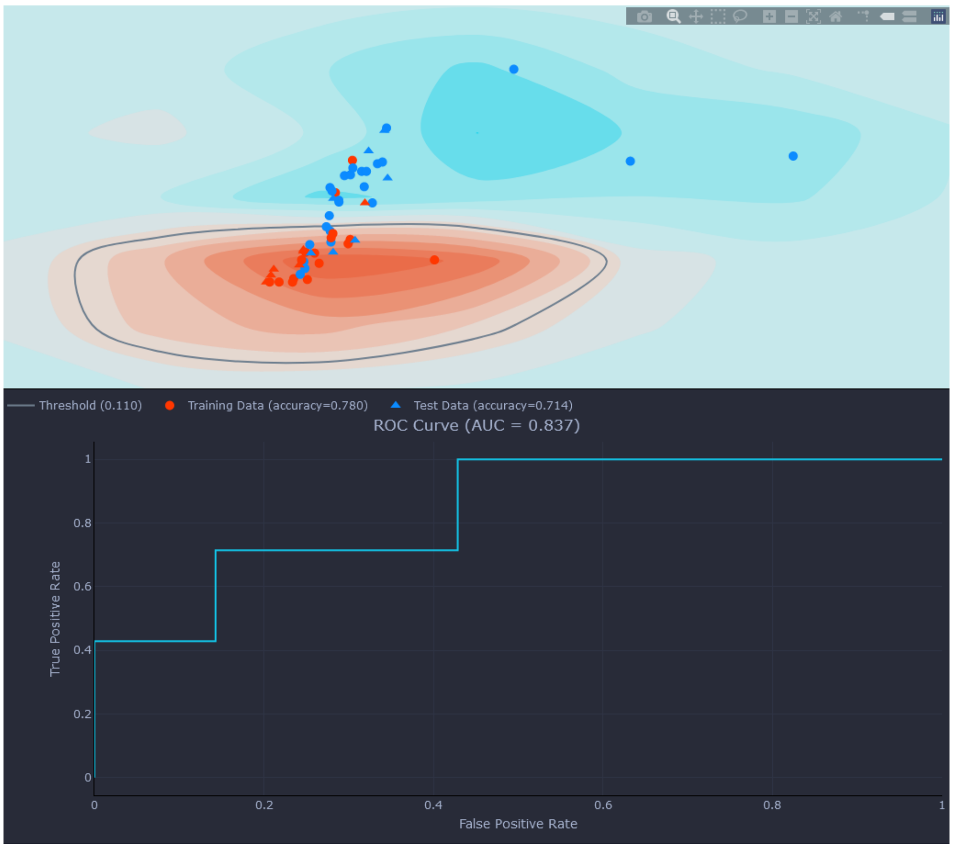


Figure 6.5: SVM machine learning algorithm with blue and red dots

Figure 6.5, illustration of the SVM algorithm. We see each point of the participants as dots, with circles denoting training data and triangles denoting test data.

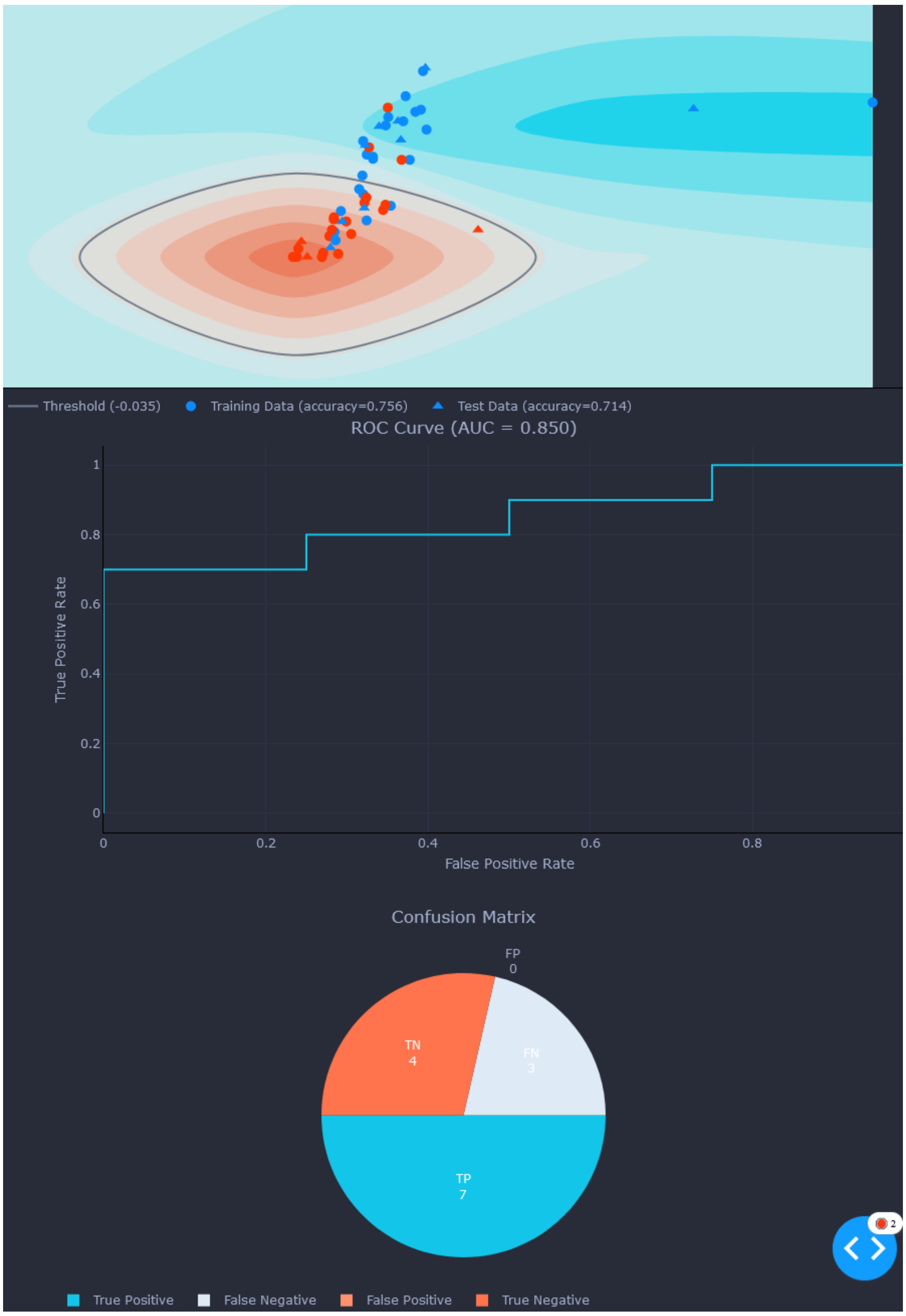


Figure 6.6: the SVM deployed on the deppression dataset top to bottom

Figure 6.6 showing the entire output dashboard of the SVM when trained on the depression dataset. Not shown are the trainability options, selecting datasets and time-series for a specified user.

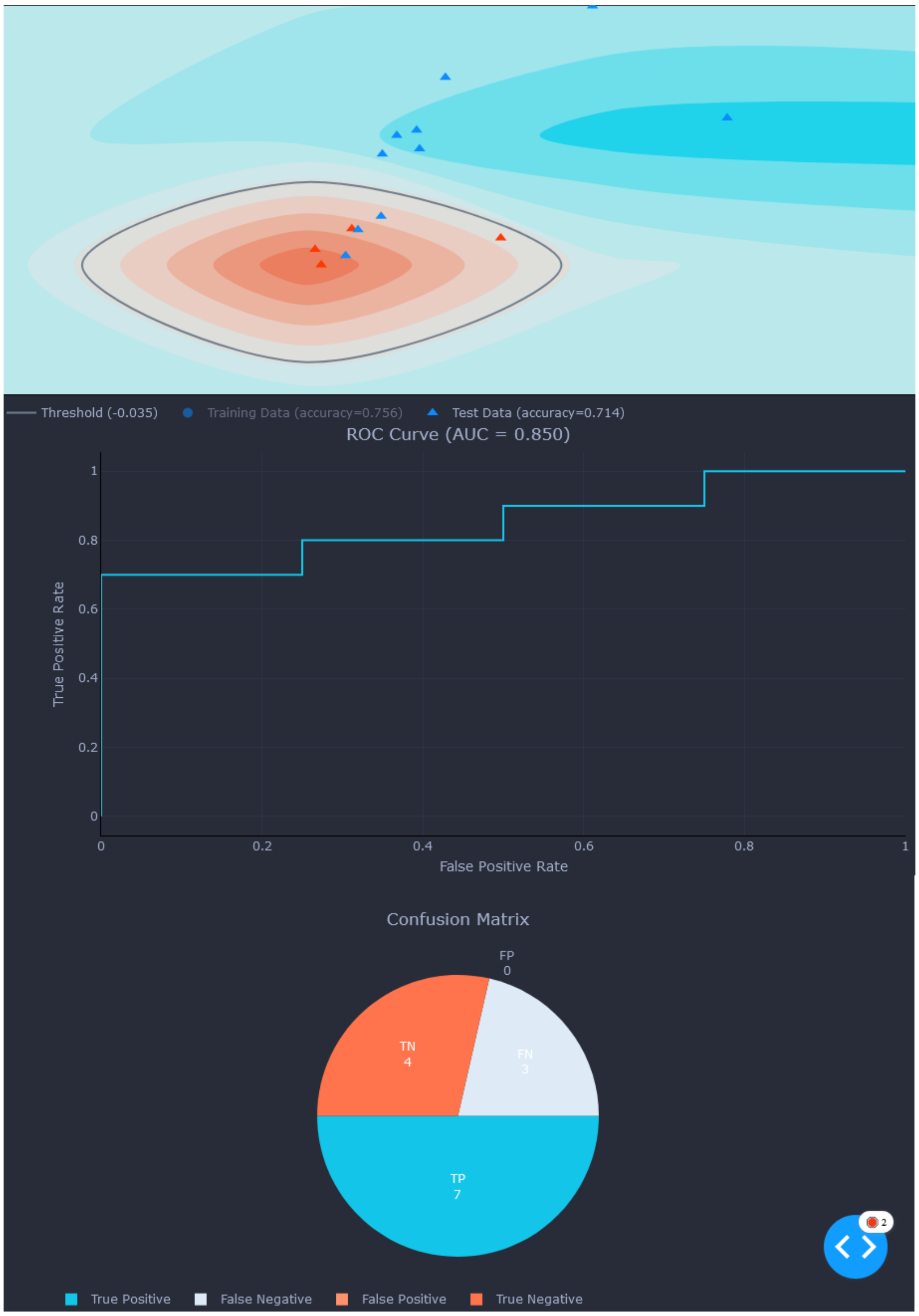


Figure 6.7: the SVM deployed on the deppression dataset only showing the test samples

Figure 6.7 shows us that by clicking on training data, we can see all the test data points alone. From there, it is possible to see the misclassification of three depressed (blue) samples.

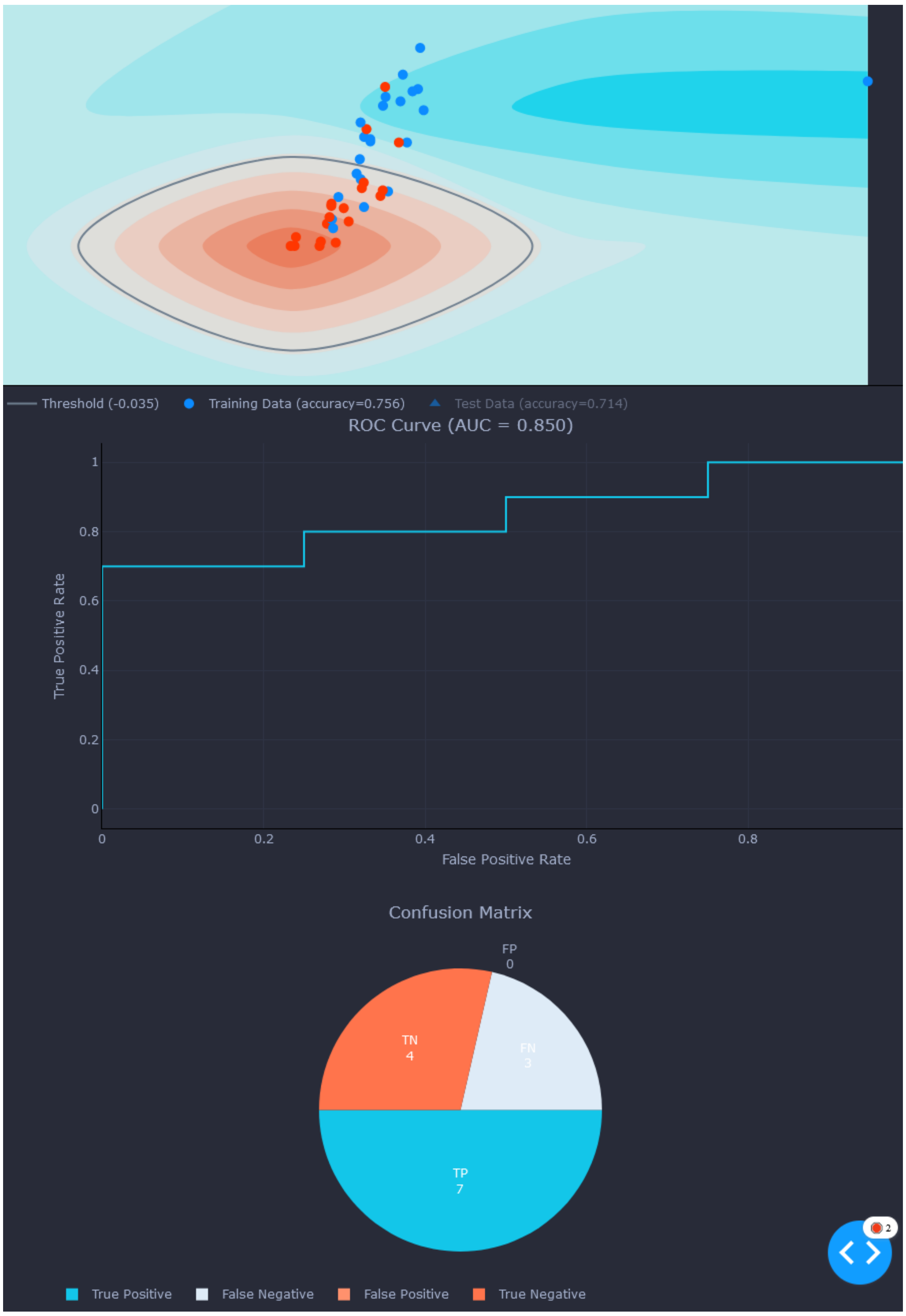


Figure 6.8: the SVM deployed on the deppression dataset only showing the training samples

Figure 6.8 shows us that we can see all the training points alone by clicking on "Test Data" and get a picture of how the SVM drew the boundary.

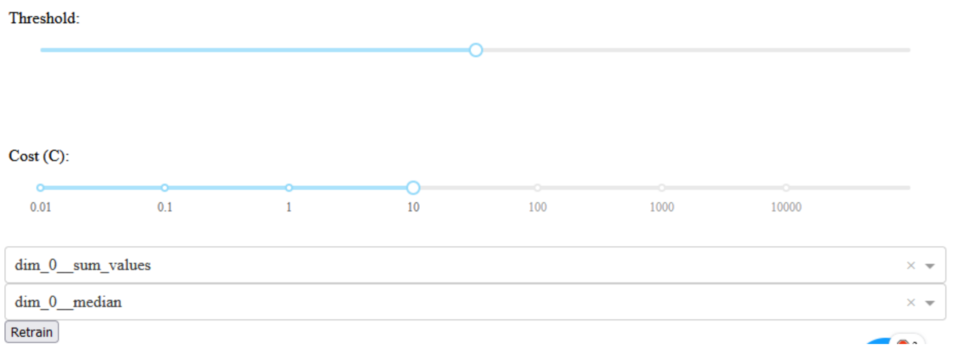


Figure 6.9: Retrainability menu

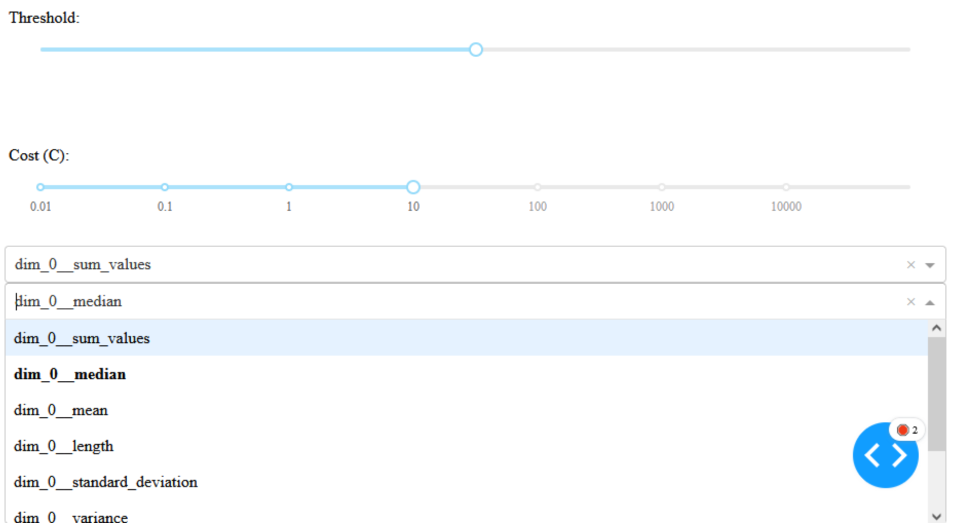
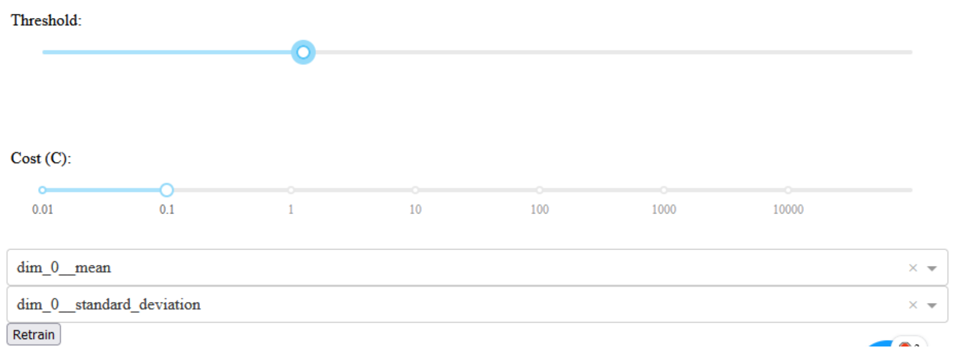


Figure 6.10: Choosing factors to train on

Figure 6.9 and 6.10 shows the trainability options; changing the threshold, meaning where the SVM draws its boundary between points. In addition, the tuning regularisation parameter C is changeable. Also, we can change which two factors from Tsfresh we use as training variables in the SVM algorithm.


Threshold:

Cost (C):

dim_0__mean

dim_0__standard_deviation

Figure 6.11: changing all the training parameters and factors to train on

In figure 6.11, we have changed the threshold, cost(C), and training features.

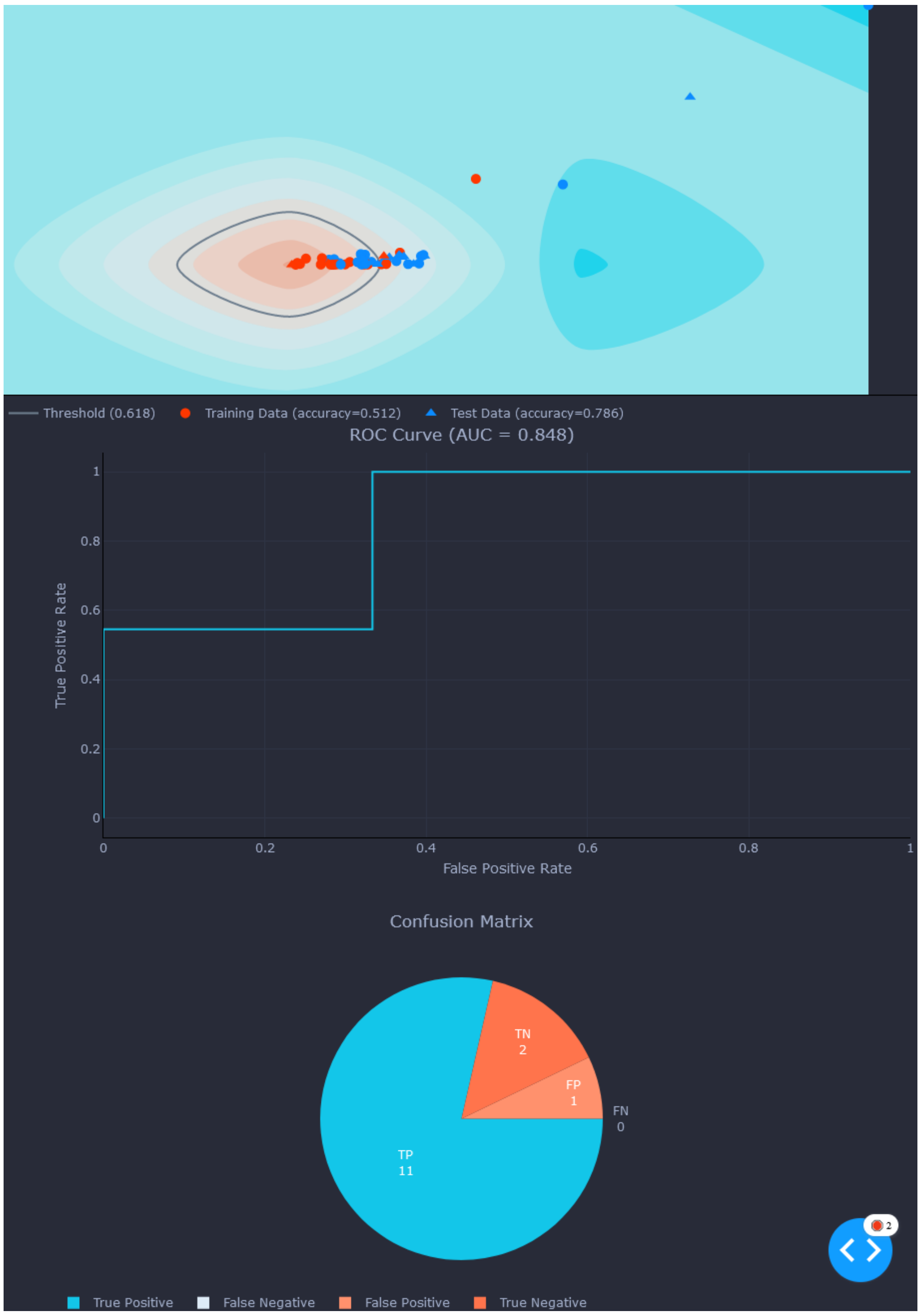


Figure 6.12: The newly trained model after we changed the training parameters

Figure 6.12 shows the newly trained SVM model after changing the training parameters and hitting the retrain button.

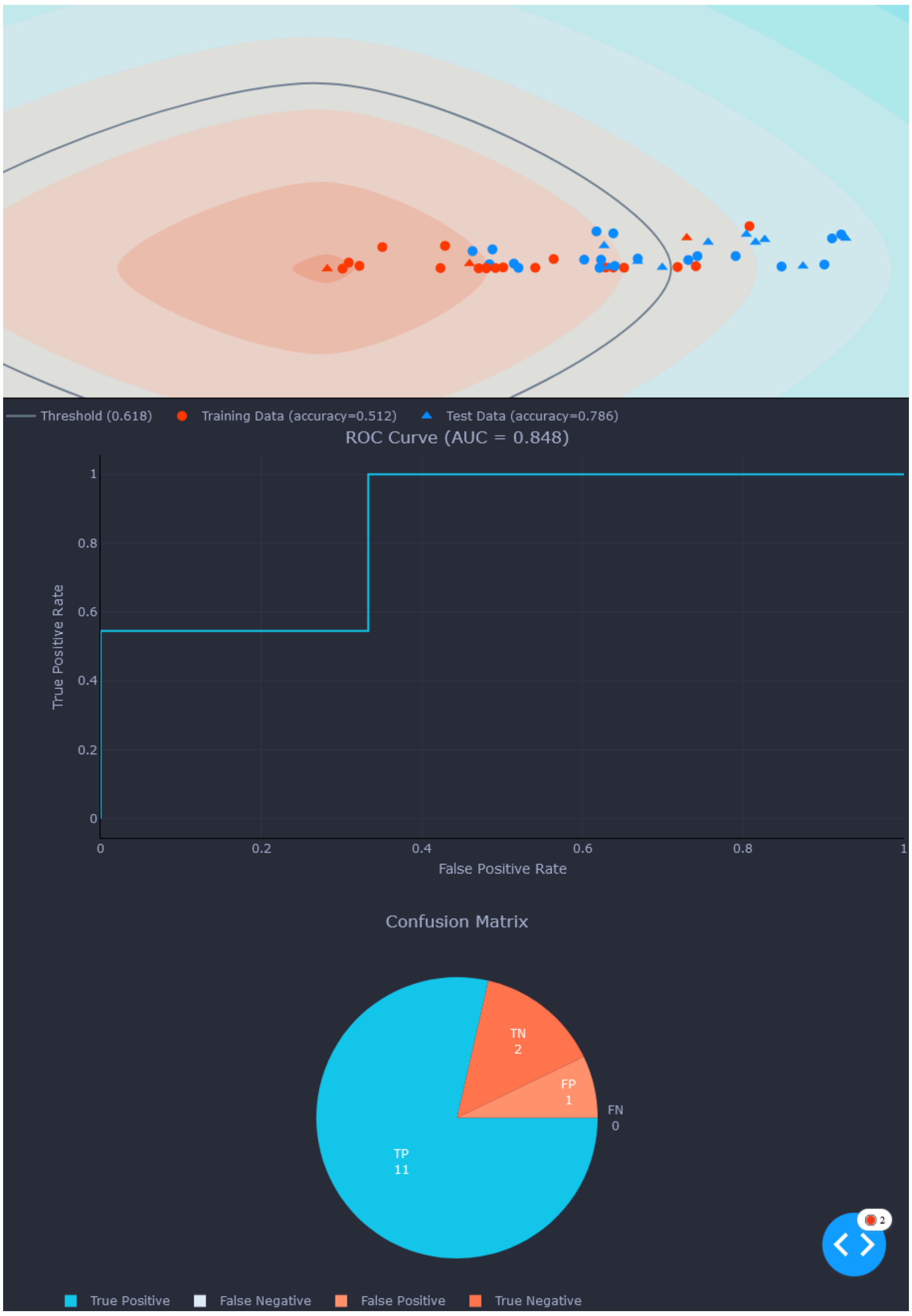
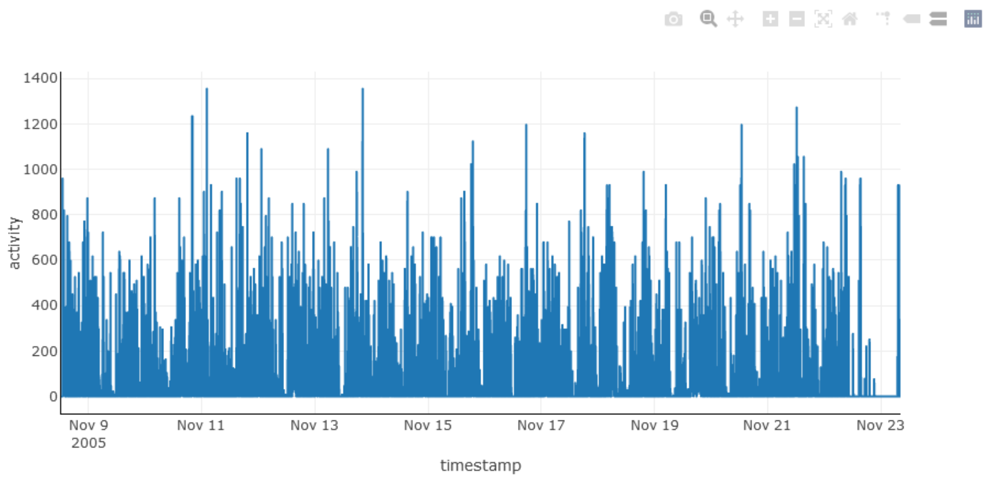


Figure 6.13: The newly trained model after we changed the training parameters and zoomed in

Figure 6.13 shows the newly trained SVM model after we drew a square with the mouse around the SVM screen and by doing so zoomed in.



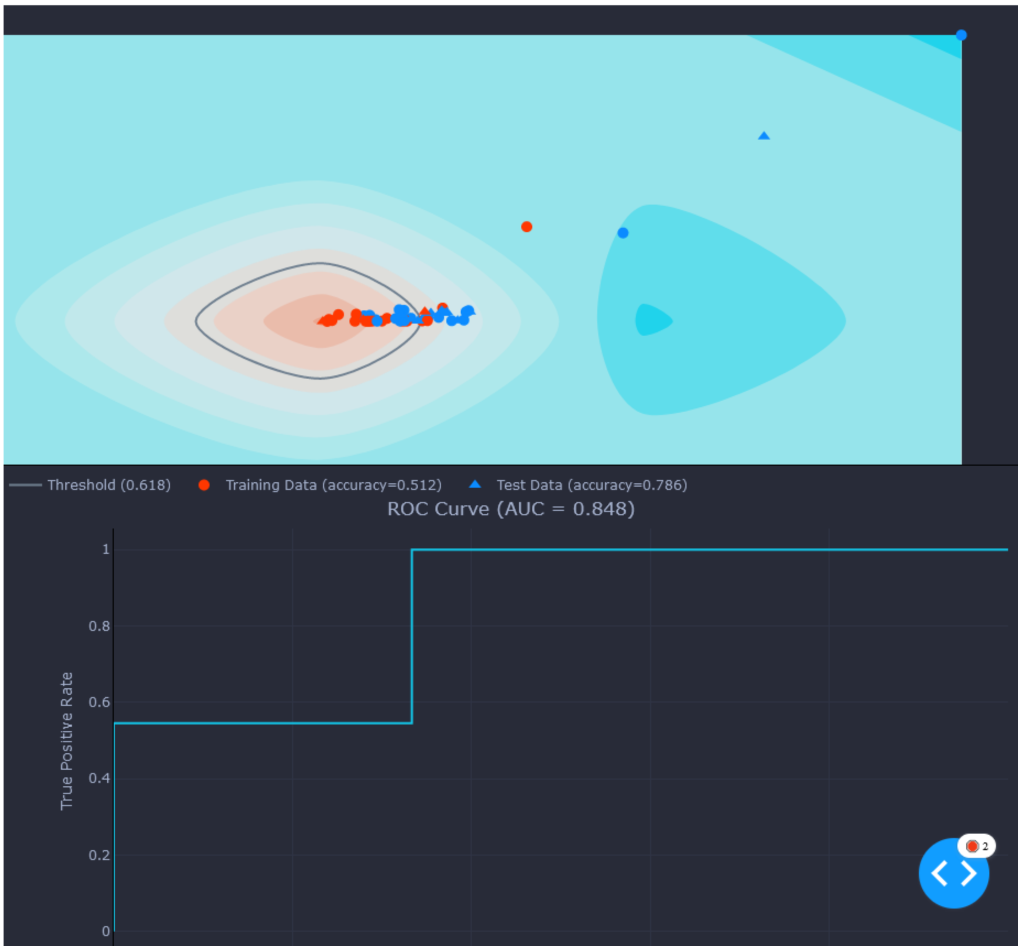


Figure 6.14: time-series from user

Figure 6.14 shows the time-series from a user represented as a dot in the SVM.

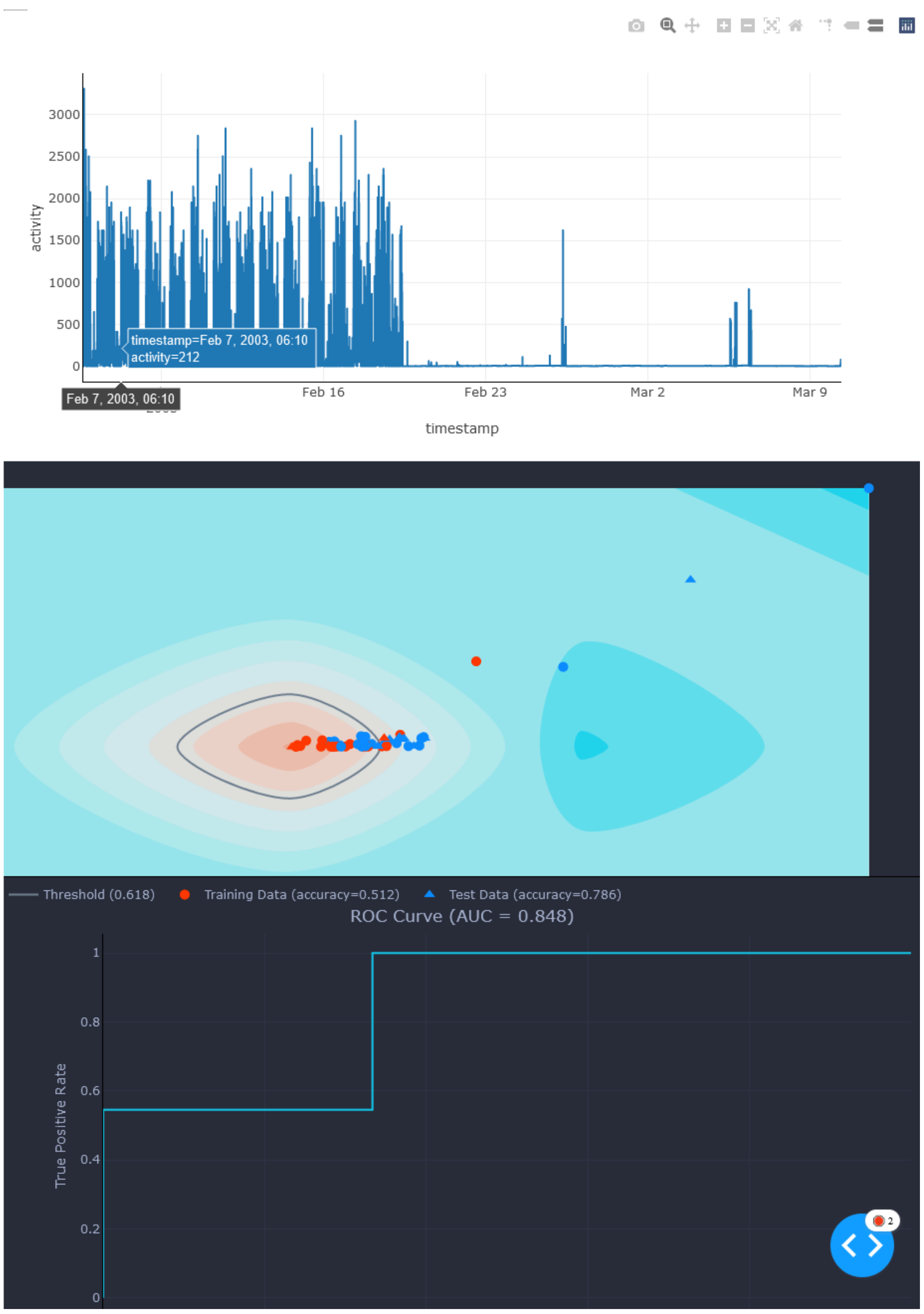


Figure 6.15: time-series from another user

Figure 6.15 shows the time-series when we click on the dot representing another user.

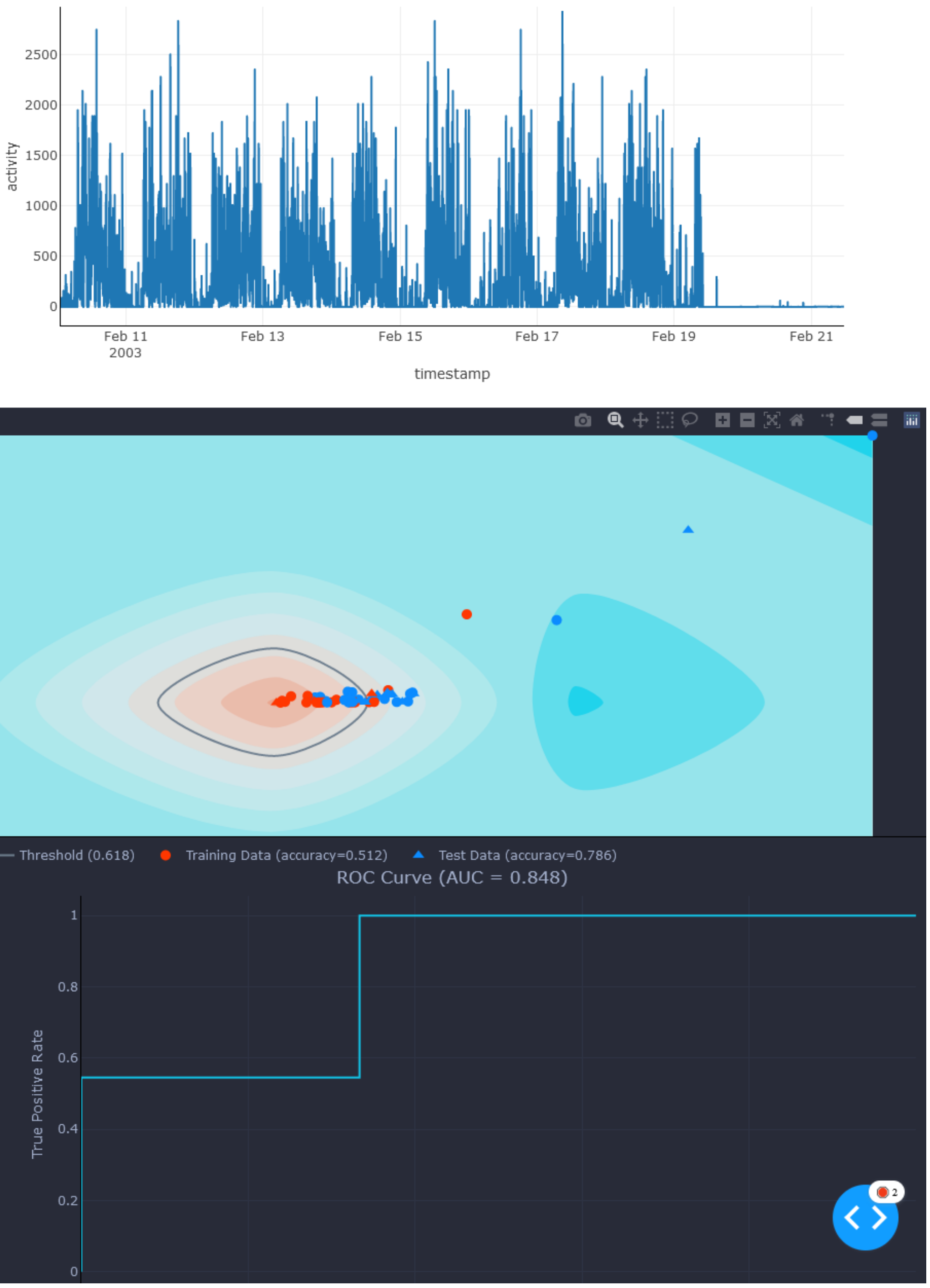


Figure 6.16: Zoomed in on the time-series

Figure 6.16 shows what happens when we draw a box around the time-series screen and zoom in.

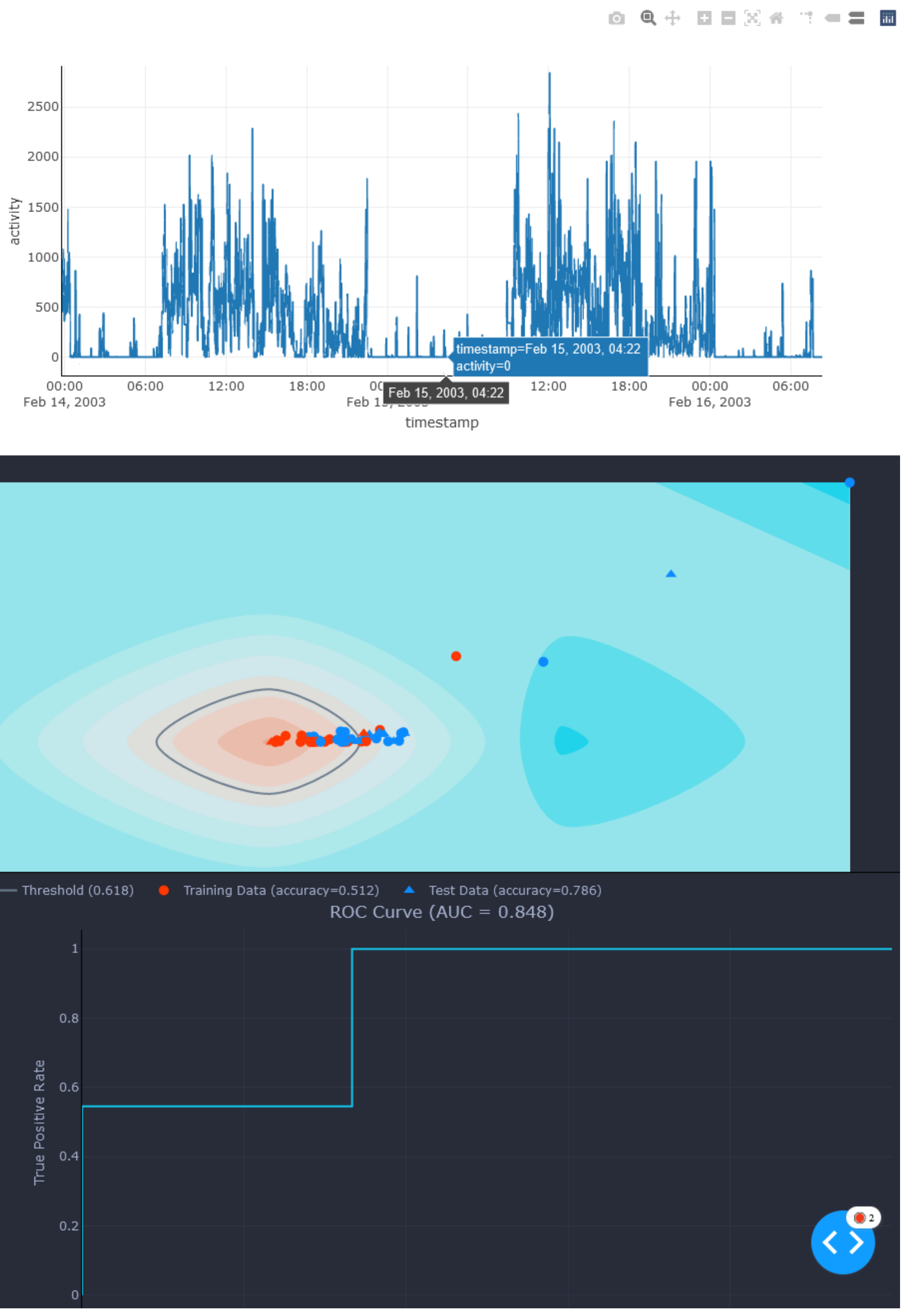


Figure 6.17: Zooming in again

Figure 6.17 shows what happens when we again draw a box around the time-series screen and zoom in even further.

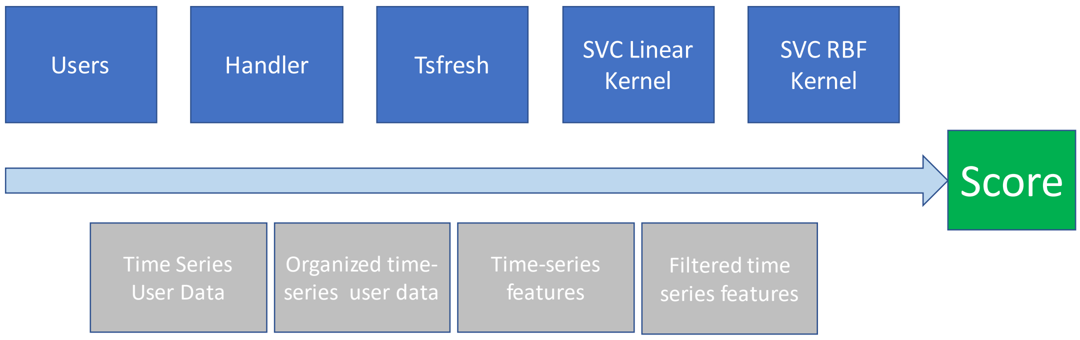


Figure 6.18: Pipeline achieving its score

Figure 6.18 explains how the pipeline achieved its score on the depression dataset. First, we have users who upload their time-series data to a handler. This handler takes these time-series features and organises them. Tsfresh uses these organised time-series to extract time-series features. From there, we put these time-series features into an SVC Linear Kernel that determines which of these features are the most important. Afterwards, putting these filtered time-series features into an SVC with an RBF kernel. This SVC trains a model and gets the final score. The final score is not the score from the SVM GUI since this GUI is from a model that only takes two factors to train a model.

Chapter 7

Results

7.1 Results from the depression datasets

The following section shows the results obtained with the pipeline deployed onto both the ADHD and Depression dataset. The section also illustrates the deployment of the Gephy graph and the user explanation in the user class.

7.1.1 Graphical user interface Depression dataset

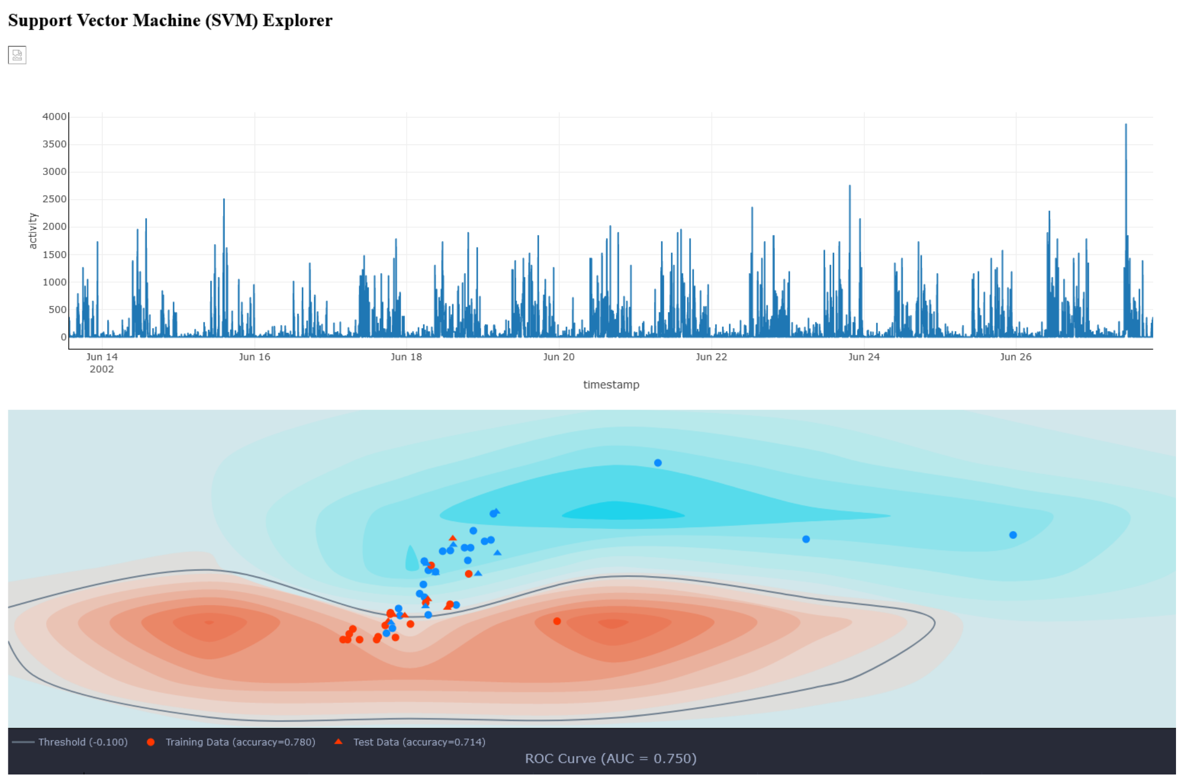


Figure 7.1: Deppression dataset with timeseries and SVM

From the depression dataset in figure 7.1, the researcher could see all the participants and their place in the SVM algorithm. By clicking on each point illustrated in the figure, the researcher could see each individual's activity level on each timeline. The functioning of the timeline let the researcher zoom in and investigate it in more detail, allowing the researcher to zoom in on a more exact time point of the timeline.

There was also a ROC and confusion matrix available to illustrate the efficiency of the currently trained model in figure 7.2.

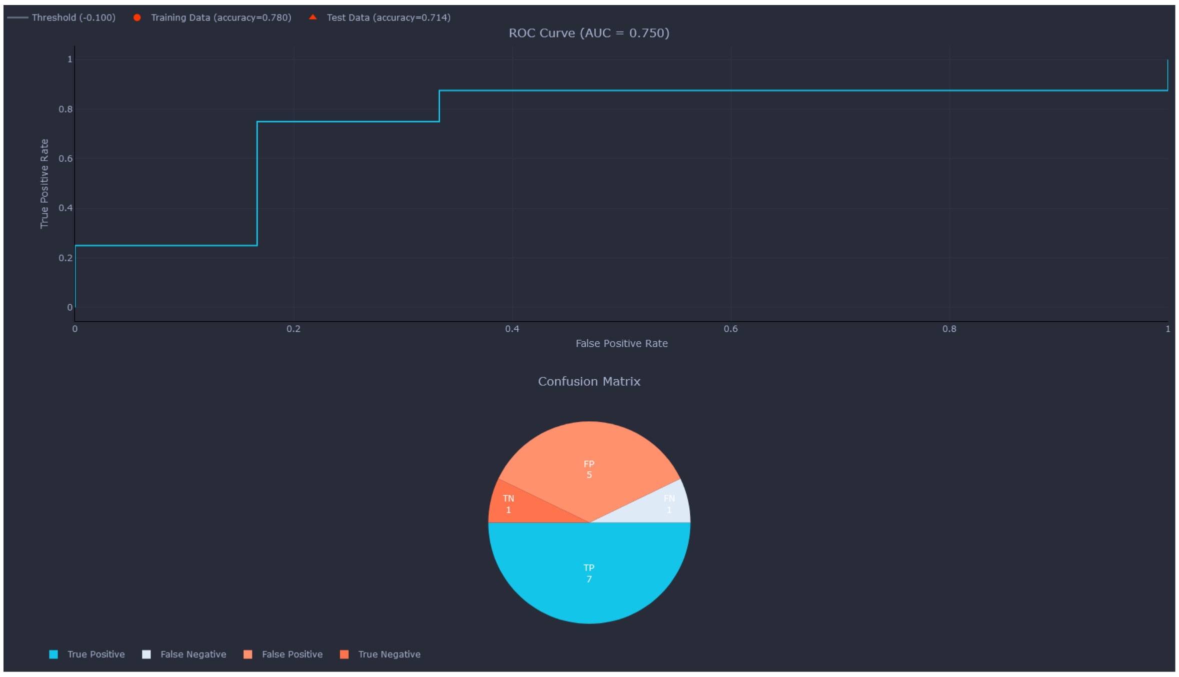
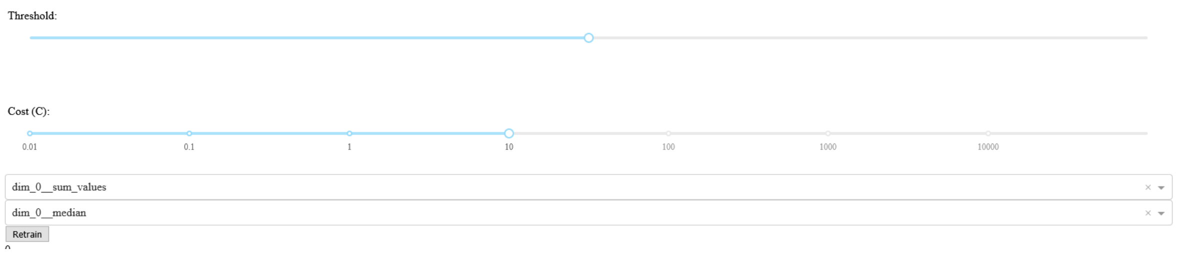


Figure 7.2: ROC and Confusion Matrix

The research class in figure 7.3 made it possible to retrain the model based on the time-series features.



Threshold:

Cost (C):

0.01 0.1 1 10 100 1000 10000

dim_0__sum_values

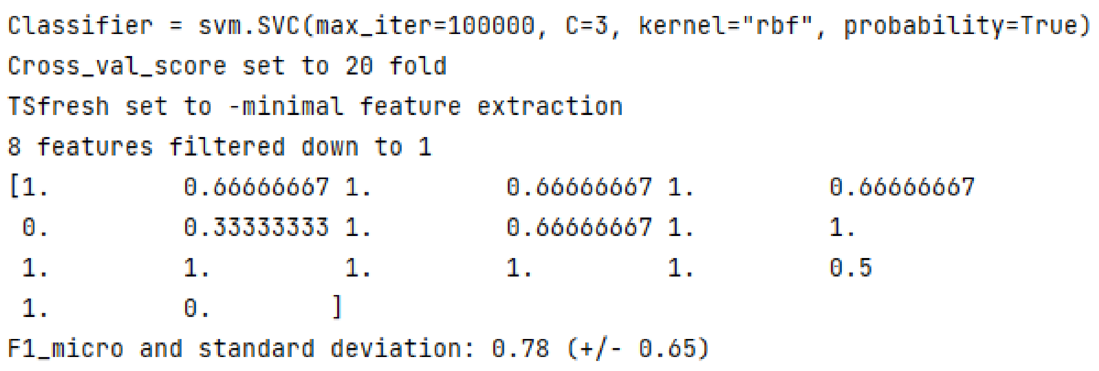
dim_0__median

Figure 7.3: tweak menu

7.1.2 Depression Data-set Score results

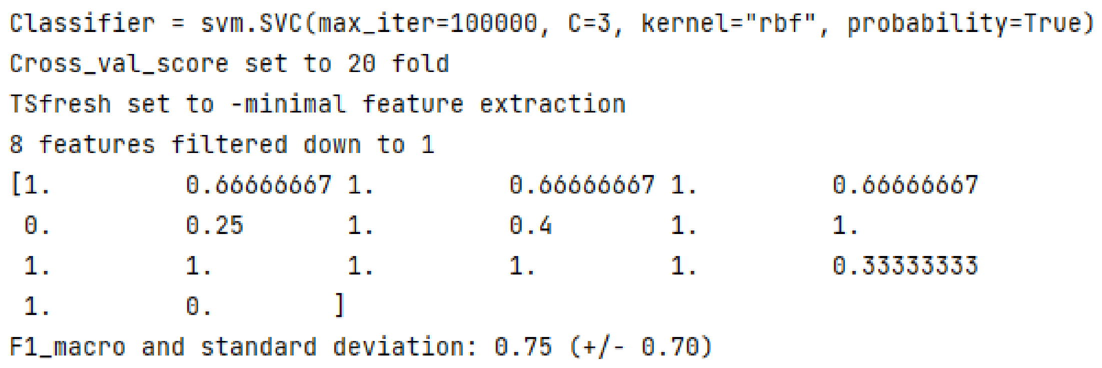
Result from the six runs of cross-validation on the Depression dataset with time- series extracted features, organised into minimal, efficient and comprehensive feature extraction from Tsfresh. Showing the number of features and how much they were filtered down. The classifier is illustrated but is the same across every cross-validation run: F1 micro and F1 macro scores displayed on the results with the standard deviation.

Run 1 with minimal feature extraction from Tsfresh



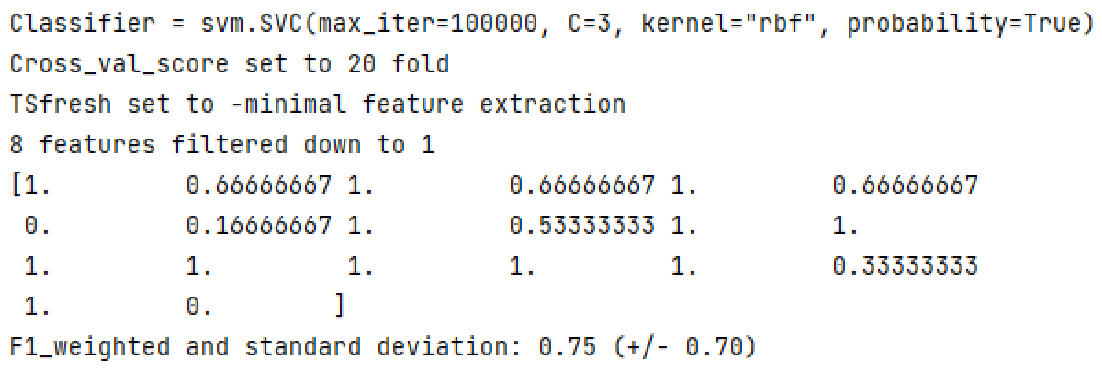
```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -minimal feature extraction
8 features filtered down to 1
[1.         0.66666667 1.         0.66666667 1.         0.66666667
 0.         0.33333333 1.         0.66666667 1.         1.
 1.         1.         1.         1.         1.         0.5
 1.         0.        ]
F1_micro and standard deviation: 0.78 (+/- 0.65)
```

Figure 7.4: F1 micro with minimal feature extraction



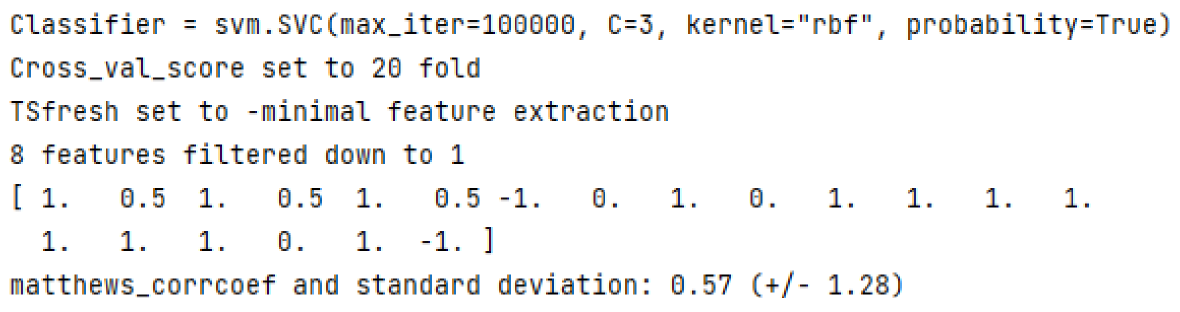
```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -minimal feature extraction
8 features filtered down to 1
[1.         0.66666667 1.         0.66666667 1.         0.66666667
 0.         0.25       1.         0.4        1.         1.
 1.         1.         1.         1.         1.         0.33333333
 1.         0.        ]
F1_macro and standard deviation: 0.75 (+/- 0.70)
```

Figure 7.5: F1 macro with minimal feature extraction



```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -minimal feature extraction
8 features filtered down to 1
[1.         0.66666667 1.         0.66666667 1.         0.66666667
 0.         0.16666667 1.         0.53333333 1.         1.
 1.         1.         1.         1.         1.         0.33333333
 1.         0.        ]
F1_weighted and standard deviation: 0.75 (+/- 0.70)
```

Figure 7.6: F1 weighted with minimal feature extraction

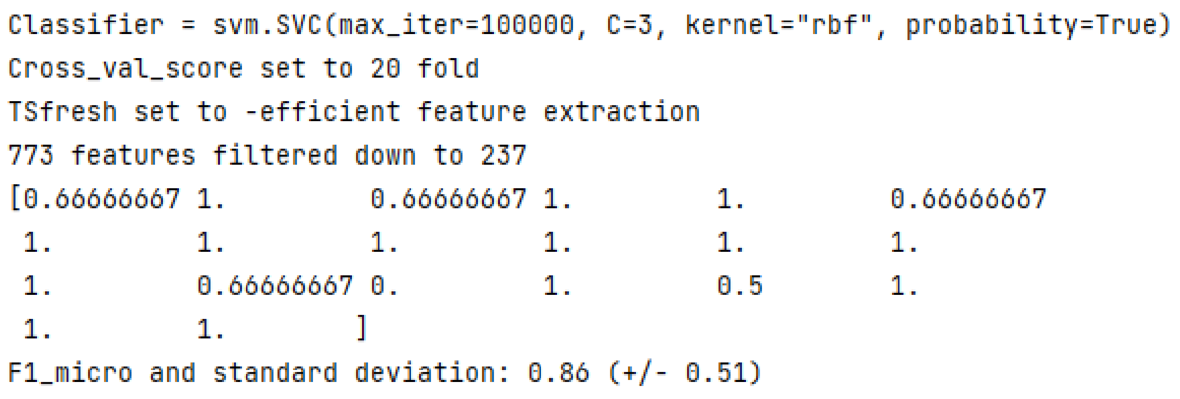


```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -minimal feature extraction
8 features filtered down to 1
[ 1.   0.5  1.   0.5  1.   0.5 -1.   0.   1.   0.   1.   1.   1.   1.
  1.   1.   1.   0.   1.  -1. ]
matthews_corrcoef and standard deviation: 0.57 (+/- 1.28)
```

Figure 7.7: Matthews Corrcoef with minimal feature extraction

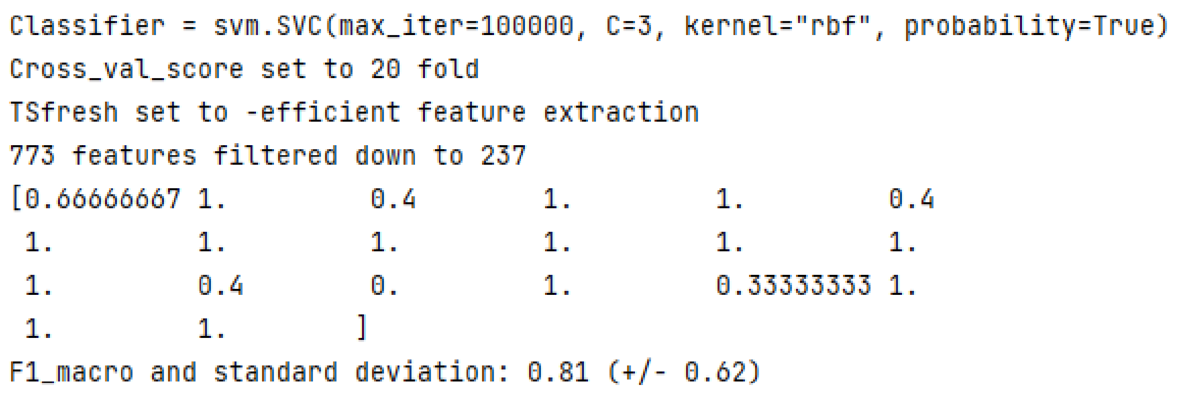
Figure 7.4, 7.5, 7.6 and 7.7 shows the F1 micro, macro, weighted and MCC score from the pipeline when using minimal feature extraction from Tsfresh.

Run 2 with efficient feature extraction from Tsfresh



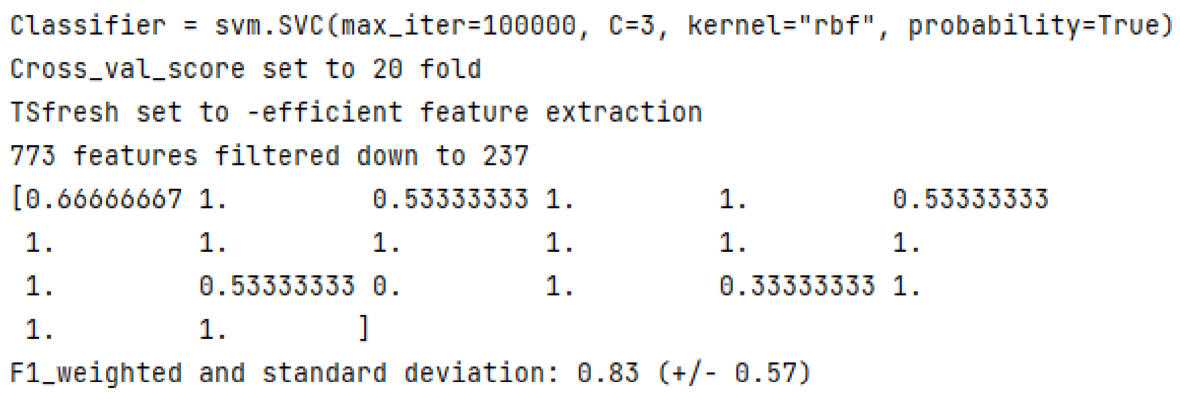
```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -efficient feature extraction
773 features filtered down to 237
[0.66666667 1.         0.66666667 1.         1.         0.66666667
 1.         1.         1.         1.         1.         1.
 1.         0.66666667 0.         1.         0.5        1.
 1.         1.        ]
F1_micro and standard deviation: 0.86 (+/- 0.51)
```

Figure 7.8: F1 micro score with efficient feature extraction



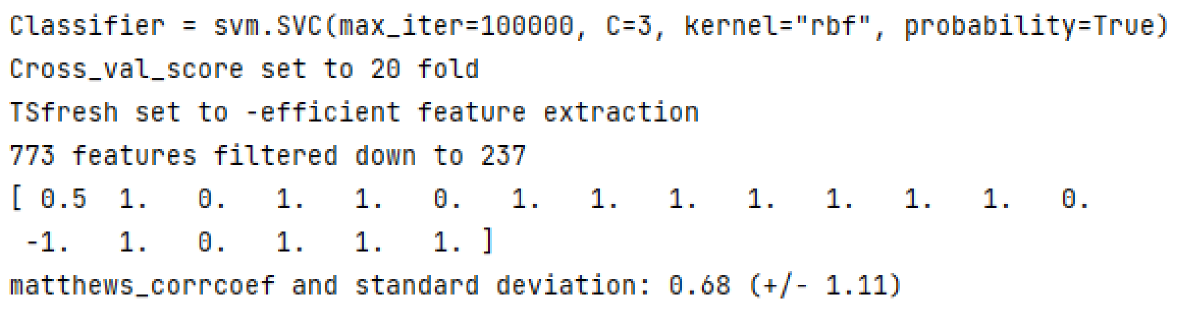
```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -efficient feature extraction
773 features filtered down to 237
[0.66666667 1.         0.4        1.         1.         0.4
 1.         1.         1.         1.         1.         1.
 1.         0.4        0.         1.         0.33333333 1.
 1.         1.        ]
F1_macro and standard deviation: 0.81 (+/- 0.62)
```

Figure 7.9: F1 macro score with efficient feature extraction



```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -efficient feature extraction
773 features filtered down to 237
[0.66666667 1.         0.53333333 1.         1.         0.53333333
 1.         1.         1.         1.         1.         1.
 1.         0.53333333 0.         1.         0.33333333 1.
 1.         1.        ]
F1_weighted and standard deviation: 0.83 (+/- 0.57)
```

Figure 7.10: F1 weighted score with efficient feature extraction

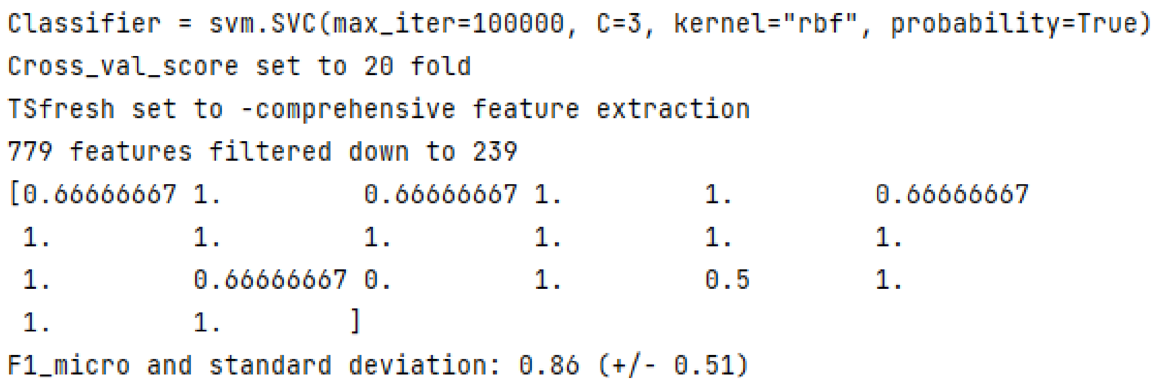


```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -efficient feature extraction
773 features filtered down to 237
[ 0.5  1.   0.   1.   1.   0.   1.   1.   1.   1.   1.   1.   1.   0.
 -1.   1.   0.   1.   1.   1. ]
matthews_corrcoef and standard deviation: 0.68 (+/- 1.11)
```

Figure 7.11: Matthews Corrcoef with efficient feature extraction

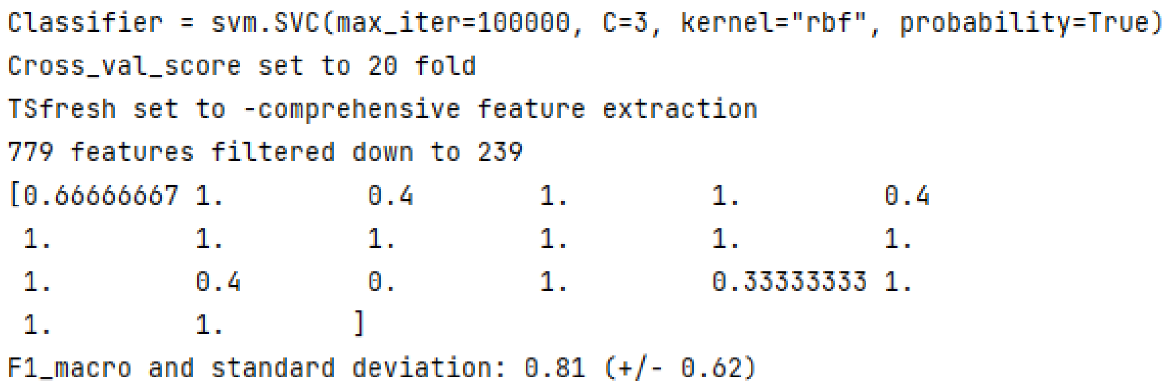
Figure 7.8, 7.9, 7.10 and 7.11 shows the F1 micro, macro, weighted and MCC score from the pipeline when using efficient feature extraction from Tsfresh.

Run 3 with comprehensive feature extraction from Tsfresh



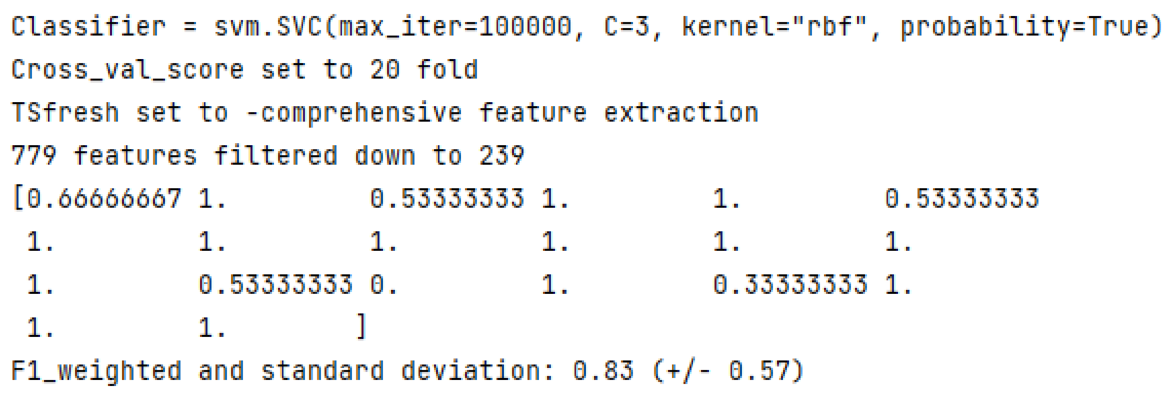
```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -comprehensive feature extraction
779 features filtered down to 239
[0.66666667 1.         0.66666667 1.         1.         0.66666667
 1.         1.         1.         1.         1.         1.
 1.         0.66666667 0.         1.         0.5        1.
 1.         1.        ]
F1_micro and standard deviation: 0.86 (+/- 0.51)
```

Figure 7.12: F1 micro score with Comprehensive feature extraction



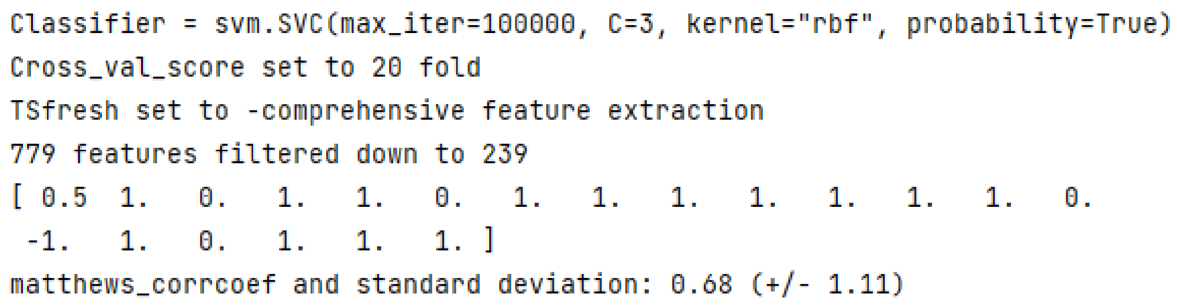
```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -comprehensive feature extraction
779 features filtered down to 239
[0.66666667 1.         0.4        1.         1.         0.4
 1.         1.         1.         1.         1.         1.
 1.         0.4        0.         1.         0.33333333 1.
 1.         1.        ]
F1_macro and standard deviation: 0.81 (+/- 0.62)
```

Figure 7.13: F1 macro score with Comprehensive feature extraction



```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -comprehensive feature extraction
779 features filtered down to 239
[0.66666667 1.         0.53333333 1.         1.         0.53333333
 1.         1.         1.         1.         1.         1.
 1.         0.53333333 0.         1.         0.33333333 1.
 1.         1.        ]
F1_weighted and standard deviation: 0.83 (+/- 0.57)
```

Figure 7.14: F1 weighted score with Comprehensive feature extraction



```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -comprehensive feature extraction
779 features filtered down to 239
[ 0.5  1.   0.   1.   1.   0.   1.   1.   1.   1.   1.   1.   1.   0.
 -1.   1.   0.   1.   1.   1. ]
matthews_corrcoef and standard deviation: 0.68 (+/- 1.11)
```

Figure 7.15: Matthews Corrcoef with Comprehensive feature extraction

Figure 7.12, 7.13, 7.14 and 7.15 shows the F1 micro, macro, weighted and MCC score from the pipeline when using comprehensive feature extraction from Tsfresh.

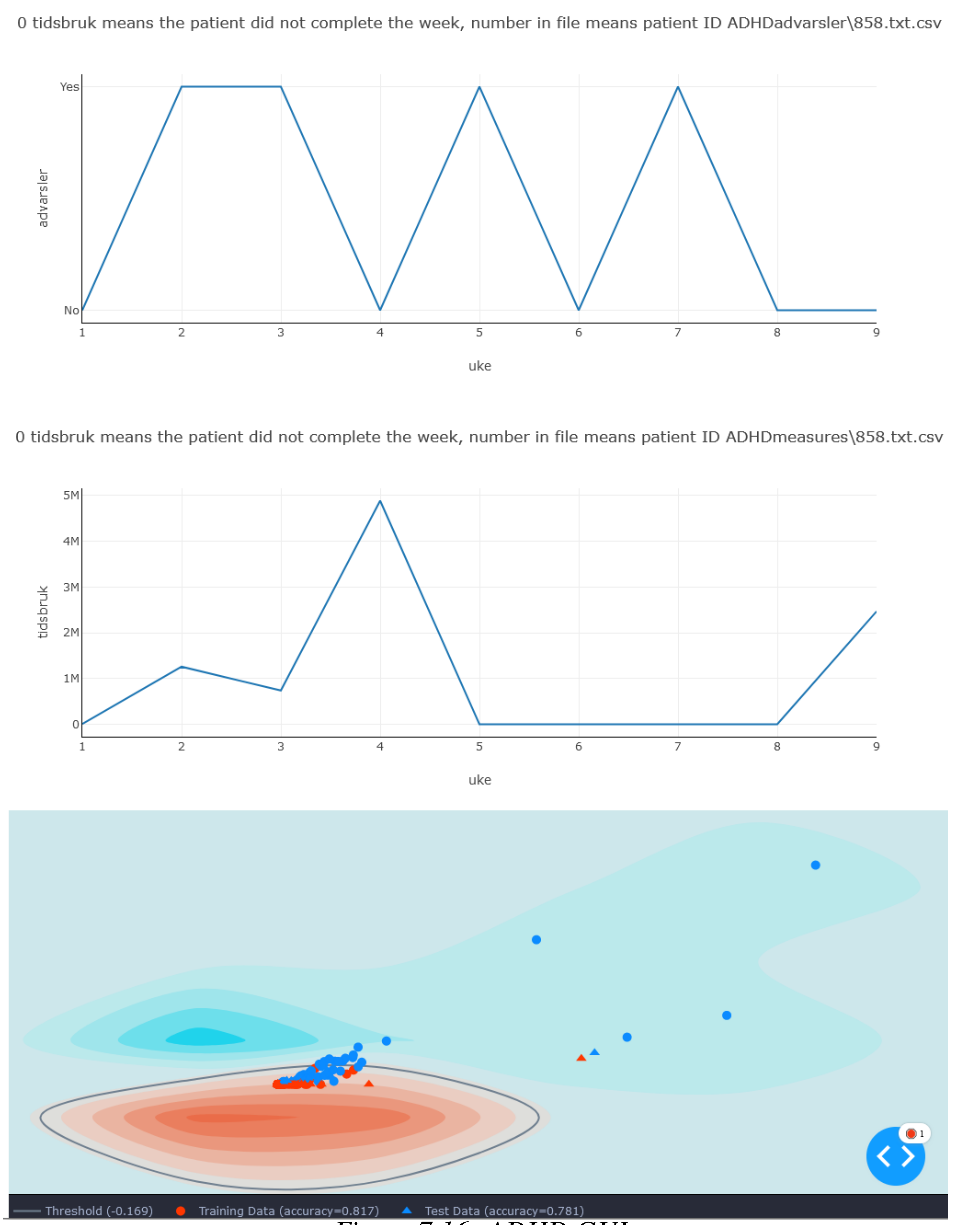
7.2 Results from the ADHD datasets

7.2.1 ADHD dataset graphical user interface

From the ADHD dataset, it is possible to click on every point and get the corresponding completion rate of every user in seconds. 0 means in this context that the user has not completed that week's module. It is also possible to view if that particular user received any warnings or not during a specific week.

7.2.2 ADHD dataset Classification scores

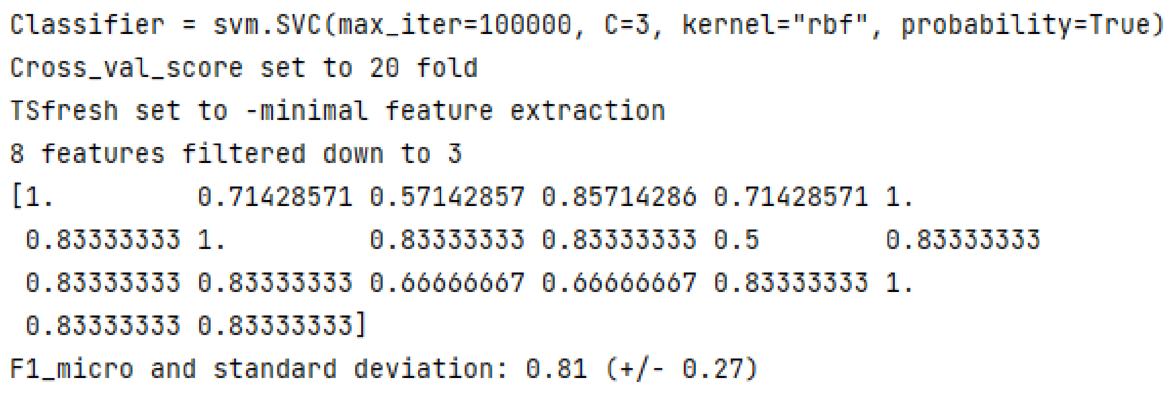
In the ADHD dataset, the last week in the module, number seven, was set as the classification criteria, classifying those who completed it with one and those who did not complete it with 0. Based on the completion rates, meaning the time it took to complete a module for each week, time-series features extracted through Tsfresh. These features are then used together with the classification to create a machine learning model. Below in figure 7.16 are the results from this model. This model shows if the user completed a module and warnings given.


0 tidsbruk means the patient did not complete the week, number in file means patient ID ADHDadvarsler\858.txt.csv

0 tidsbruk means the patient did not complete the week, number in file means patient ID ADHDmeasures\858.txt.csv

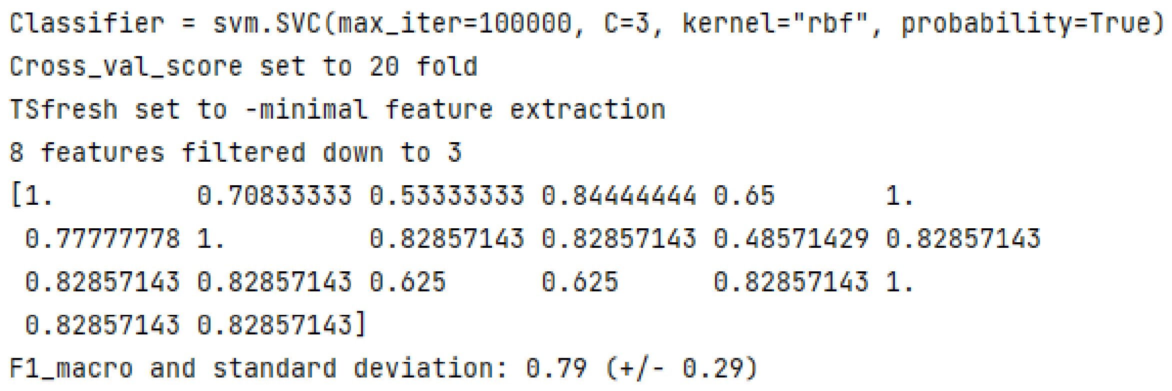
Figure 7.16: ADHD GUI

Run 1 with minimal feature extraction from Tsfresh



```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -minimal feature extraction
8 features filtered down to 3
[1.         0.71428571 0.57142857 0.85714286 0.71428571 1.
 0.83333333 1.         0.83333333 0.83333333 0.5        0.83333333
 0.83333333 0.83333333 0.66666667 0.66666667 0.83333333 1.
 0.83333333 0.83333333]
F1_micro and standard deviation: 0.81 (+/- 0.27)
```

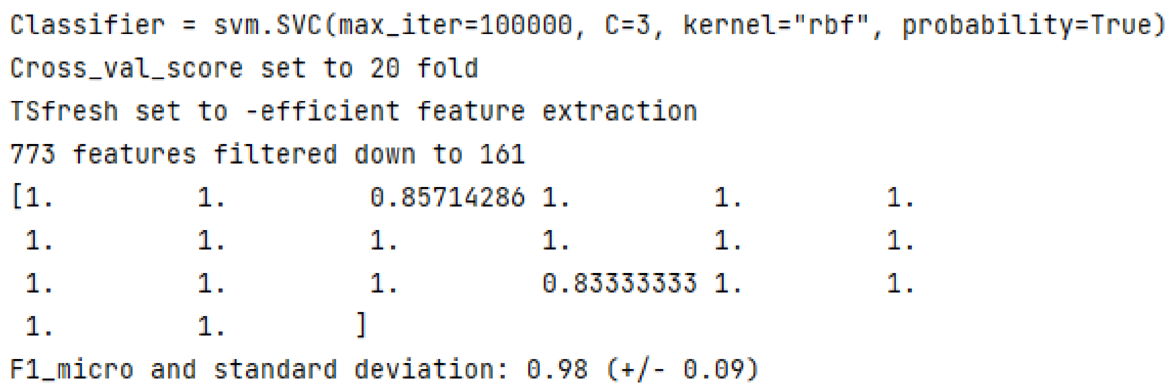
Figure 7.17: F1 micro score with minimal feature extraction



```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -minimal feature extraction
8 features filtered down to 3
[1.         0.70833333 0.53333333 0.84444444 0.65       1.
 0.77777778 1.         0.82857143 0.82857143 0.48571429 0.82857143
 0.82857143 0.82857143 0.625      0.625      0.82857143 1.
 0.82857143 0.82857143]
F1_macro and standard deviation: 0.79 (+/- 0.29)
```

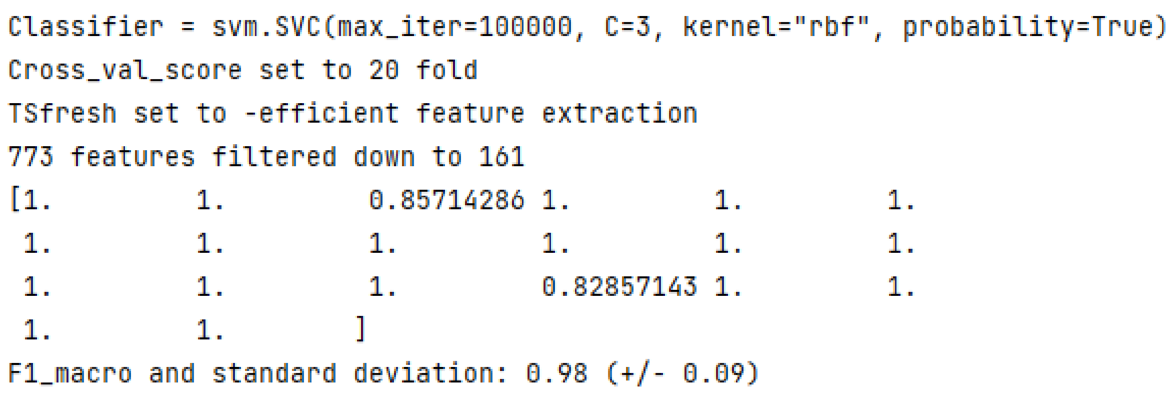
Figure 7.18: F1 macro score with minimal feature extraction

Run 2 with efficient feature extraction from Tsfresh



```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -efficient feature extraction
773 features filtered down to 161
[1.         1.         0.85714286 1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         0.83333333 1.         1.
 1.         1.        ]
F1_micro and standard deviation: 0.98 (+/- 0.09)
```

Figure 7.19: F1 micro score with efficient feature extraction



```
Classifier = svm.SVC(max_iter=100000, C=3, kernel="rbf", probability=True)
Cross_val_score set to 20 fold
TSfresh set to -efficient feature extraction
773 features filtered down to 161
[1.         1.         0.85714286 1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         0.82857143 1.         1.
 1.         1.        ]
F1_macro and standard deviation: 0.98 (+/- 0.09)
```

Figure 7.20: F1 macro score with efficient feature extraction

Figure 7.17 and 7.18 shows the F1 micro and macro with minimal feature extraction. Figure 7.19 and 7.20 shows the F1 micro and macro with efficient feature extraction.

7.3 User Explanation Graphical user interface

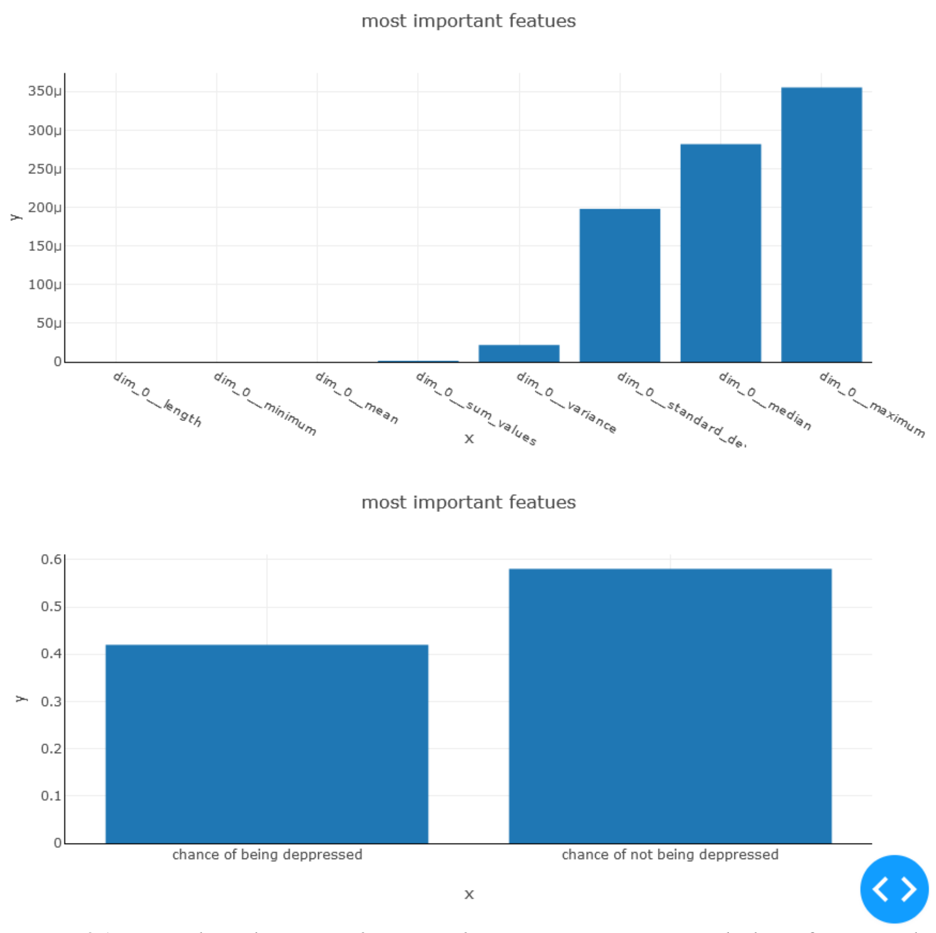


Figure 7.21: Visual explanation for user, feature importance and classification chance

Figure 7.21 shows the presentation of the user explanation, for example, showing the relative importance of the features drawn out from Tsfresh and showing how certain the classifier is in this particular case.

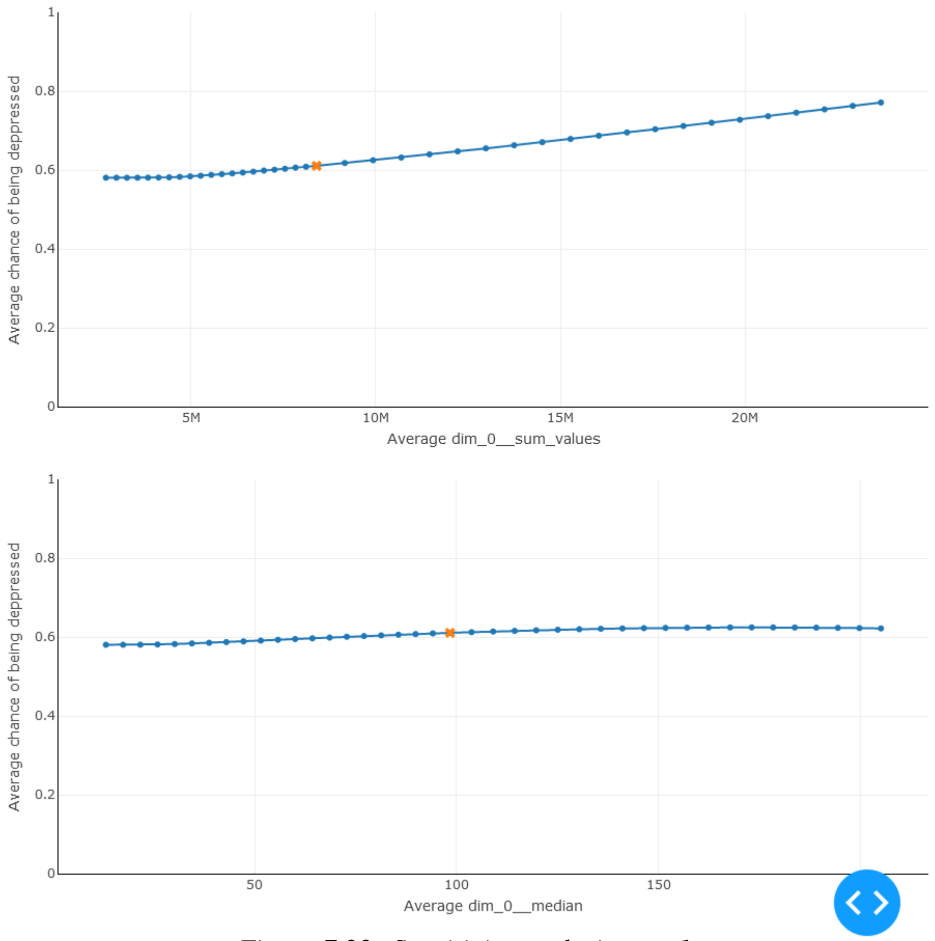


Figure 7.22: Sensitivity analysis part 1

In figure 7.22, we can see that user's with overall high average activity levels aggregated across the entire time-series have a higher chance of being depressed. In addition, the average median also slightly increase the chance of being depressed.

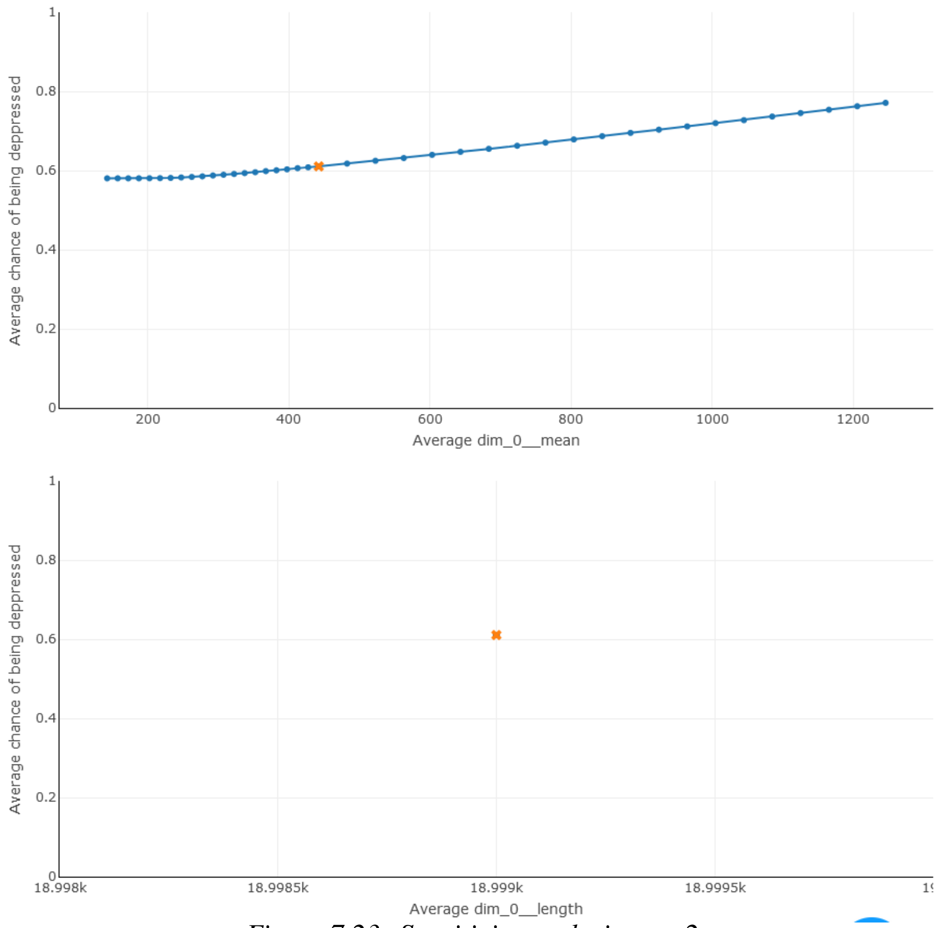


Figure 7.23: Sensitivity analysis part 2

In figure 7.23, the mean value is important, increasing the chance of depression if increased. Unfortunately, the average length variable is not shown due to a bug in Ethik software.

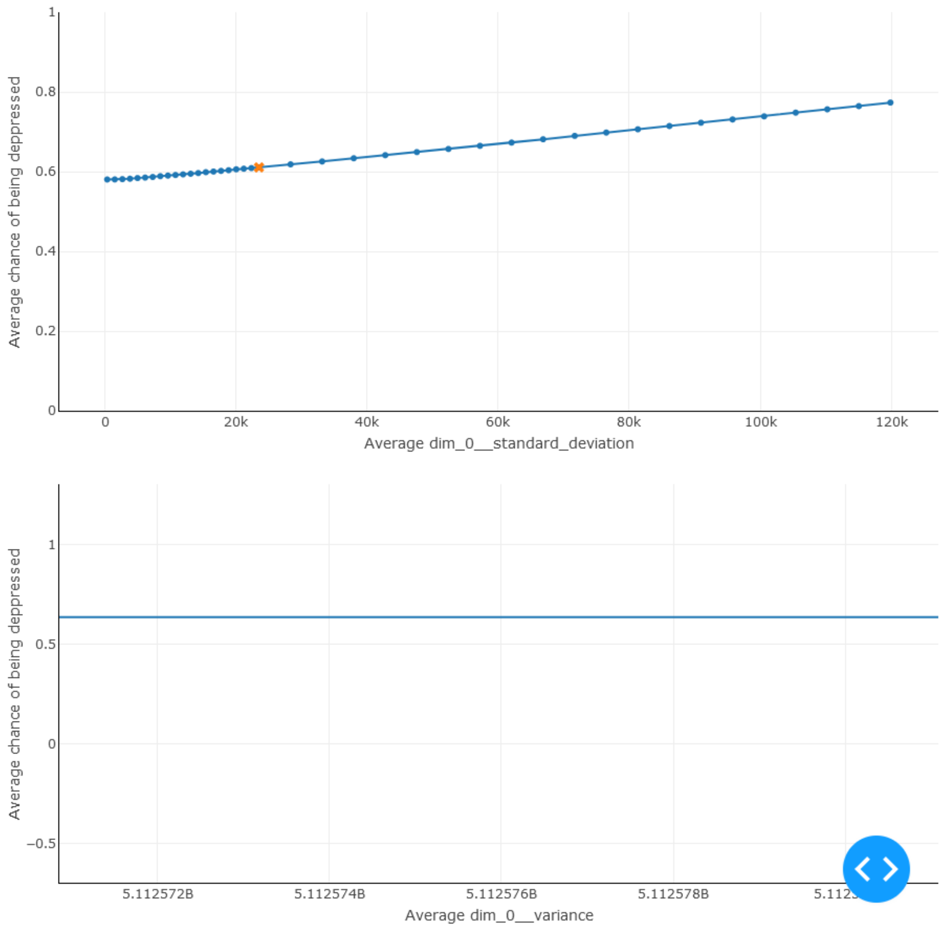


Figure 7.24: Sensitivity analysis part 3

In figure 7.24, standard deviation had a large impacts on the chances of being depressed, and the variance had a negligible impact on the result. The standard deviation result is comparable to the gephy graphs showed later in the thesis where a depressed person having a larger deviation in his time-series.

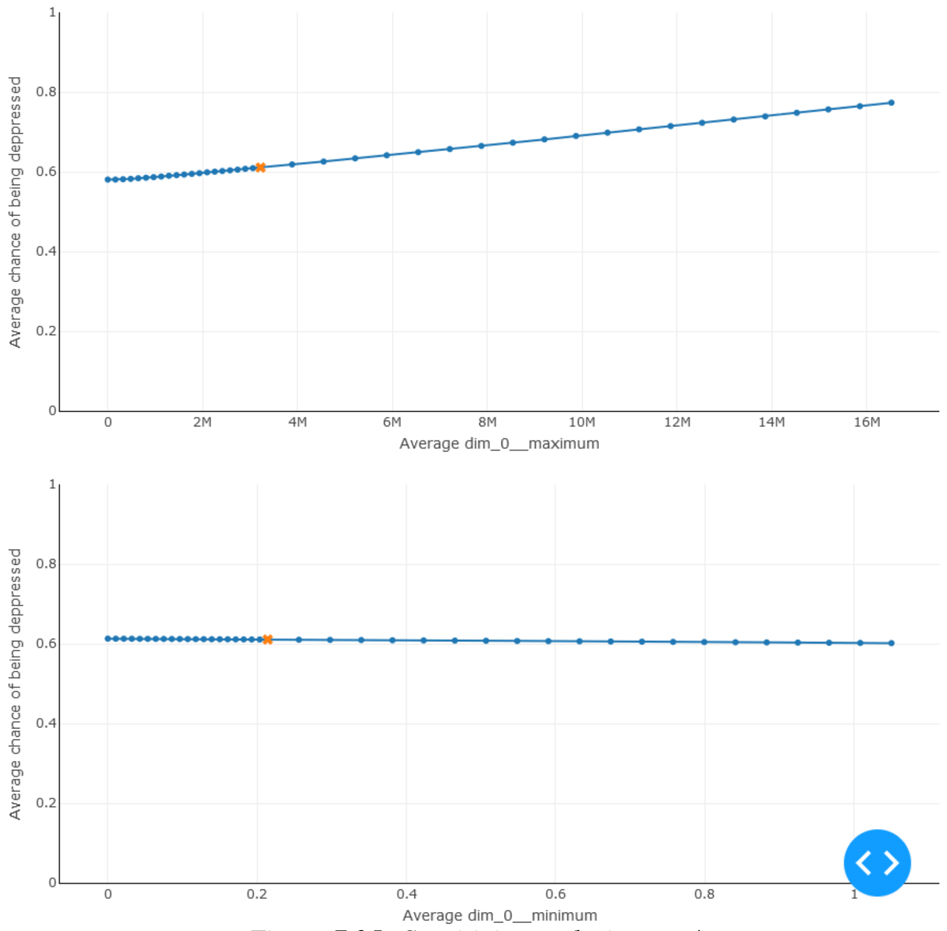


Figure 7.25: Sensitivity analysis part 4

Figure 7.25 shows that the maximum value increased the chance of being depressed, and the minimum value had no relevance.

7.4 Deploying a Gephy graph

There were also experiments conducted on running a Gephy graph of the activity section of a depressed and non depressed users CSV file. The 1000 first activity entries in the users time-series illustrated with the nodes standing for each activity score in each point of a time-series in figure 7.26 and 7.27. Edges between the nodes are drawn based on the rules of the visibility algorithm. Note, the Gephy graph is not deployable through the research class in the pipeline but is deployable through a self-contained code segment. We can see that the depressed user has a different activity level than the non-depressed user. All the edges going into one node means that the user at that particular point in time had a very high activity level. Due to time constraints, extracting graph features to use on the pipeline was not done.

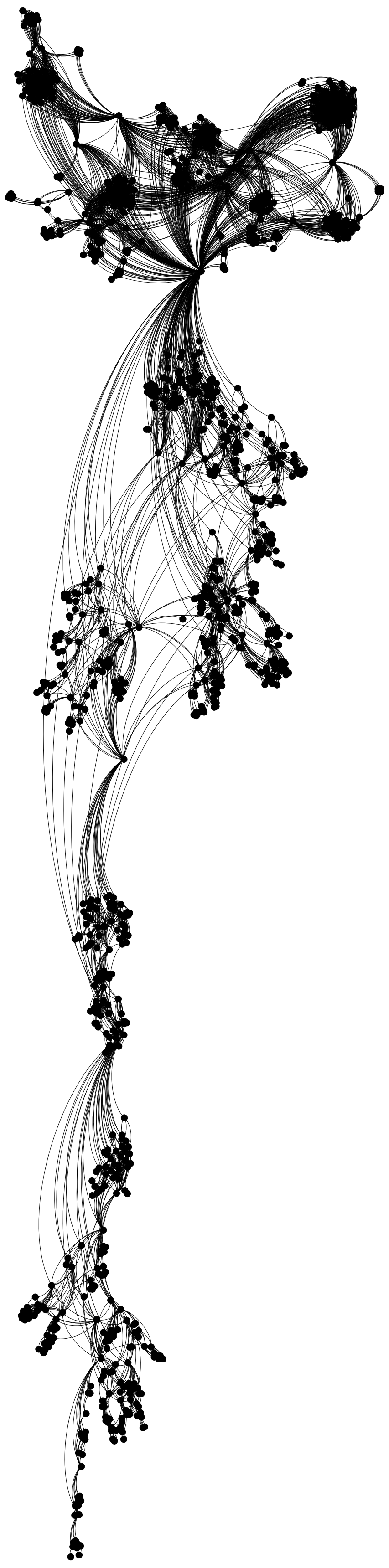


Figure 7.26: Visualization of activity data using visibility algorithm for a patient with depression symptom

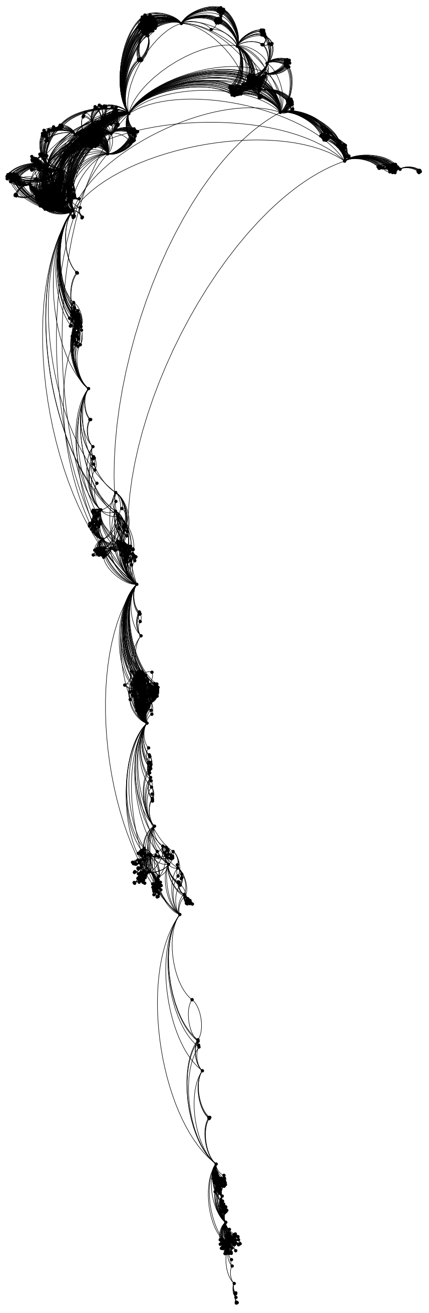


Figure 7.27: Visualization of activity data using visibility algorithm for a patient without depression symptom

Chapter 8

Discussion and Conclusion

8.1 Challenges and limitations

The apparent restrictions were time constraints forcing limitations on the scope of development. There was not enough time to integrate the XAI_{Text} domain model grammar and the whole Plotly Dash logical structure. Some parts of the program needed to receive the most resources, and a decision to prioritise the research class as it is best at illustrating the potential of the developed tool was needed.

8.2 Pipeline score

8.2.1 depression dataset

In the depression dataset, the highest score achieved is a weighted F1 score of 0,73. With minimal feature extraction, the pipeline reached an F1 micro score of 0,78 and an F1 macro score of 0,75, better than the best score on the dataset. With efficient feature extraction, achieving an F1 score of 0.81 macro and 0,86 micro, this is far better than the best score achieved on the dataset. Extracting even more features did not improve the overall score.
The high degree of standard deviation on the dataset is interesting, resulting from the low number of participants - 55. Still, an F1 micro score of 0,78 by extracting and selecting features through Tsfresh and training them on a dataset seems like an outstanding achievement. From the results, it appears that monitoring patients at risk for depression with activity monitors has its use. Machine learning techniques are also something that is best on large datasets(*Zhang and Ling*, 2018). Predicting how the scores would have changed if the dataset contained more participants is difficult. Still, it is not unlikely that an improvement would occur, or most definitely, the standard deviations would have gone down - presenting a more stable result. The depression dataset has many factors like age, gender, work, marriage, education, MADRS scores, and daycare days. The authors of *Garcia-Ceja et al.* (2018a) tried utilising these features and receiving little success in using these factors in a predictive way. Perhaps Tsfresh somehow extracts

time- series features that correlate with these unused inputs. Still, it would be interesting to see if we had a larger dataset, how training algorithms could improve the overall result when taking these factors into account. Even with this small dataset, a natural next step would be to factor in these variables to investigate if an even better score is achievable. There is also some good work on extracting graph theory features from time-series. Suppose the feature extractor of Tsfresh does not cover this. Strictly for pragmatical reasons to achieve a better score, combining as many features as possible, from time-series extraction, graph theory, or static variables, seems like a good strategy for future research.

8.2.2 ADHD dataset

The first run of the pipeline on the ADHD dataset provided an F1 micro score of 0,81 and an f1 macro score of 0,79 with minimal features extraction. It achieved an F1 score of 0.98 for both micro and macro F1 with a standard deviation of 0.09 with efficient feature extraction, an excellent score. However, the pipeline tried to solve a problem where the SVM algorithm might not have been the ideal algorithm. The algorithm only tried to solve how we categorise through time-series features which would finish week seven versus who would not. The intention behind the research of the ADHD dataset was to see if retention rates, meaning people who finish the modules, could be improved by adding warnings or reminders in various forms. What the pipeline did accomplish was to show its ability as an analysis tool. It plots all the participants into an SVM graph, their relative position being their mathematical relations towards the SVM model based on the time-series extracted features of Tsfresh. It did prove to be an analysis tool that at least could show deviating scores. Letting any researcher click on the point and investigate it or see where the different classified scores tended to cluster together.

8.3 Research questions

Research question1
"How can we increase the application of explainable AI utilising a model-driven engineering approach?".

Domain model engineering approaches showed great potential at creating and scaffolding out AI applications. It is possible to take a pre-made AI application project and modify it for other purposes. However, having this all in a deployable pipeline through a domain model grammar could significantly shorten the work required to create and deploy an XAI application. Having a logical connection between research, user and handler class was very useful. The user class could upload data, which the handler aggregated into a TS file. The research class then brought this all into a machine learning algorithm. TextX proved that it could generate large amounts of code from small models, potentially producing large amounts of productivity gains.

Research question 2
"How can the area of XAI be improved by interpretation and visualisation techniques?

The thesis did not make any better visualisation techniques. However, it demonstrated an approach for efficiently enabling visualisation techniques for user and researchers to be deployed by the pipeline deployed through the MDE approach. Referencing the definitions in the theory section for XAI. The suggested explanation for the SVM, visualised with two selected factors, makes it easy to understand for humans. It is "understandable" for human beings without revealing its inner workings. When presenting the sensitivity analysis in the user explanation, the model has "comprehensibility". Because the knowledge and insight attained through the sensitivity analysis are presentable to humans. It is a "feature relevance explanation". It is also "interpretable" since the meaning from the model presents itself in a way that has meaning to humans. The pipeline also has "explainability" since there is an interface between the humans and the algorithm. The algorithm is also "transparent" since so much of its inner working reveals itself to the user. A "explanations by simplification", "explanation by example", and "visual explanation" is also demonstrated for users by showing the SVM when it takes in 2 factors extracted from Tsfresh.

8.4 Improvements and future work

8.4.1 Code cleaning

The code itself could also be more pragmatically valid for developers if refactored to meet the most stringent clean code principles. It would also be helpful if the code were also simplified as much as possible, meaning one chunk of code could replace three if possible. The code is also more complicated than it needs to be at certain places; simplifications could make the whole tool more useful.

8.4.2 User Class

Expanding the user class since what it was now was only bare bones. Future work should have a "commit data" or something similar style button. The user can upload his data towards a project. Also, a "get explanation button", where Plotly Dash graphs present the workings of how the machine learning algorithm made its decision. With a wast library of various explanation techniques in the DSL, this button could have a tailored explanation to the individual user. For example, someone with aphasia, learning disabilities, visual problems, or similar could have their explanation custom-fitted.

8.4.3 Handler Class

What needed doing while handling both datasets was cleaning and fixing the data beforehand. Working knowledge of pandas and IT would be a prerequisite for such a job. What might be an option would be to create a user interface around the handler class. Making it possible to take in the data in various formats. From there, it might be helpful to clean the dataset from within a GUI. It was possible to retrain and tweak a machine learning algorithm through an interactive GUI. It was also possible to research every data point within the algorithm. Doing something similar with the handler class, where it takes in a CSV file or other input should be possible. This step illustrating the CSV file in a viewable and browsable format. From there, pandas style cleaning and data organisation tasks are available. This operation could improve the whole job of cleaning and organising data easier for people with less IT or experience with pandas or similar tools. A drop-down menu with different datasets, like demonstrated in the research class, could also be helpful as a way to organise a pandas GUI workbench.

8.4.4 Server data uploading and download

What could also be helpful is if it was possible to create a class or function through the user class where the data the user inputs somehow get uploaded to a server. If this server could send data down to the user, it would be an overall beneficial system. There should also be cross-platform compatibility and usability for different operating systems. Given that most smartphones and smartwatches do not use windows as their operating system.

8.4.5 Research Class

The research class has gotten the most work done on it in this thesis since it is the best for illustrative purposes and overall utility. Still, some more work should have gotten done on it if there were time. An expansion of the research capability of the class would also be beneficial. It can see all the users' points and visualise what kind of time-series data the machine learning algorithm received as its input. In this class, researchers should get more information about each user when clicking on a user point in the SVM screen. The researcher should have the ability to dig deep into each user seeing all the data he wants about each user. As shown, this can be done and would not be a problematic expansion programmatically. All this added information could be optional in the XAI_{Text} grammar.
Furthermore, it might be helpful if the researcher could do some tweaking towards an explanation aimed at each user. This explanation is saved for each user on a server. When the user logs on and requests an explanation, the stored explanation from the researcher gets illustrated. This explanation being useful if the researcher, for example, browses through the users and finds someone with a diagnosis of autism, aphasia, learning disabilities or

lack of IT knowledge. From there, he could use whatever is available of empirical knowledge and custom-make an explanation for every user based on their peculiar situation.

8.4.6 Grammar expansion

The most apparent improvement towards the code and the tool would be if the XAI_{Text} grammar, expanded to include many machine learning algorithms. These machine learning algorithms could span everything from the easily explainable regression-based models to the more complicated to explain deep learning neural net models. There should also be more of an emphasis on creating complete callbacks functions customisable in the XAI_{Text} DSL.

8.4.7 Dash Plotly improvements

There is also a vast potential to improve the process concerning Plotly Dash with domain modelling for creating fast deployable visualisation techniques. For example, by experiments with more flexible templates. Even on the website of Plotly Dash, there are many machine learning apps displayed. Having similar apps like those deployed through customisable grammar and very flexible templates can make creating apps like this a lot easier.

8.4.8 Creating an executable file

Something that would also have some practical value is if the code generation, XAI_{Text} DSL grammar and the templating system is organised into an executable program containing an interface. This interface could ease the app creation process for people unfamiliar with programming and IT. It could help deploy XAI applications and curb the comprehensive learning curve by writing a model within TextX with a graphical user interface.

8.4.9 Conclusions

Domain-specific languages and domain engineering does have their use in the area of XAI. XAI needs efficient ways to deploy applications with explainability. This thesis shows that the link between a handler, a user class presenting an explanation and a researcher class training and tweaking an algorithm has its use. Many parts of the XAI_{Text} DSL and the templates are grounded in Plotly Dash as the output. Suppose something proves to be more applicable than Plotly Dash. In that case, it should not be too difficult to only change out the front end part of the pipeline.
Something which much literature about the usefulness of DSL focuses on is whether the given case the developer has in front of them is the ideal for a domain-specific language. Since we are essentially creating the building blocks of something that we are to mass-produce through a factory-like code

generator. The question is if the domain chosen is ideal for a DSL. Or, more specifically, is the creation of a domain-specific language with a grammar and a code generator, with its usage of time, adequately compensated for by all the applications and code potentially turned out from it?
In this case, it seems to be worth it. Having one excellent and flexible handler, a solid user class and a research class, all being very flexible in their usage. The possibility of churning out many suitable applications based on this foundation and grammar shortens the creation time for apps. Any new invention in either Pandas, Plotly, or Sklearn can be added to the existing grammar with ease. With a critical mass of grammar and templates, this tool could be highly effective and helpful for churning out XAI applications with the latest and best tools.
Reaching a critical mass and expanding the $\mathrm{XAI}_{\mathrm{Text}}$ grammar beyond a proof of concept face is possible if the project gets expanded, as mentioned previously. It is possible if some master students would like to continue the project, for example, in a group of 2-3 students next year. They would then have a foundation and the time resources to expand the $\mathrm{XAI}_{\mathrm{Text}}$ grammar and functionality to reach this critical mass by having a reliable and helpful tool to be deployed over multiple datasets and creating XAI applications with solid individual user explainability.

Appendix A

XAI$_{\text{Text}}$ - DSL grammar

```
Program: ”
users+=User?
data+=Data?
handler+=Handler?
researcher+=Researcher?
”
;


User:
’Class’ name=ID ”
’timeSeriesLength’ ’:’ chunkSize=INT ’,’ ’readfile’ ’:’ file=STRING
”
;

Data:
’Datas’ name=ID ”
Datasource= ’:’ Series
”
;

Handler:
’Handler’ name=ID ”
’Classifier’ ’:’ ml=MachineLearningAlgorithme
’featureExtractor’ ’:’ extractor= Ext
’scalar’ ’:’ scalar= Scalar
”
;

Researcher:
’Researcher’ name=ID ”
’Handler’ ’:’ Hand=Handler
”
;
```

```
Ext:
"minimal" | "efficient" | "comprehensive"
;

Series:
"timeseries" | "non-timeseries"
;

Scalar:
"StandardScaler" | "MinMaxScaler" | "None"
;

MachineLearningAlgorithme:
"svm" | "LogisticRegression"
;
```

Appendix B

Technical details of data handling

B.1 Data exploration deppression dataset

The depression dataset is a collection of .csv files. The format changes, and the operations performed on the dataset was multifold. Firstly, what was needed was to gather everything into a .TS file. Because the feature extraction from Tsfresh requires that the format for feature extraction is in a .TS format, to achieve this, the most pragmatic solution was to read the relevant column, which in the case of the depression dataset was the "activity" column of every user .csv file. After reading the column and converting it into a text file, writing the text file in a long string with commas separating every value, adding annotations at the end of the text file with either a :1 or :0 for a depressed or non-depressed person. All these text files were written as text file and put into a user class folder. The handler could take all these files and write them into a single .TS file from this folder. From there, the research class could read this .TS file, extract features, train a machine learning algorithm and display all the points of each user.

B.2 Data exploration ADHD dataset

The ADHD dataset presents itself through an XLSX file. From this XLSX file, we have access to all the relevant data, viewing each dataset's page as we please. From the relevant pages, choosing the modules page and the alarm page. The modules page showed when the participants started their exercises and when they finished their activities. Firstly the dataset was cleaned and fixed in pandas. Afterwards, saving all the data from each user in the modules page saved as a .csv file. From there, loading the .csv files into Pandas. After that, Pandas could perform operations on the data. With pandas, calculating for every user the time it took to complete a module for a week by taking the completion date and subtracting it from the starting date. From there, converting the result to seconds. Saving the converted work as a text file for each user.
The second page chosen was the alarm section, which showed the number of alarms sent out to each user per week. Handling this data was relatively

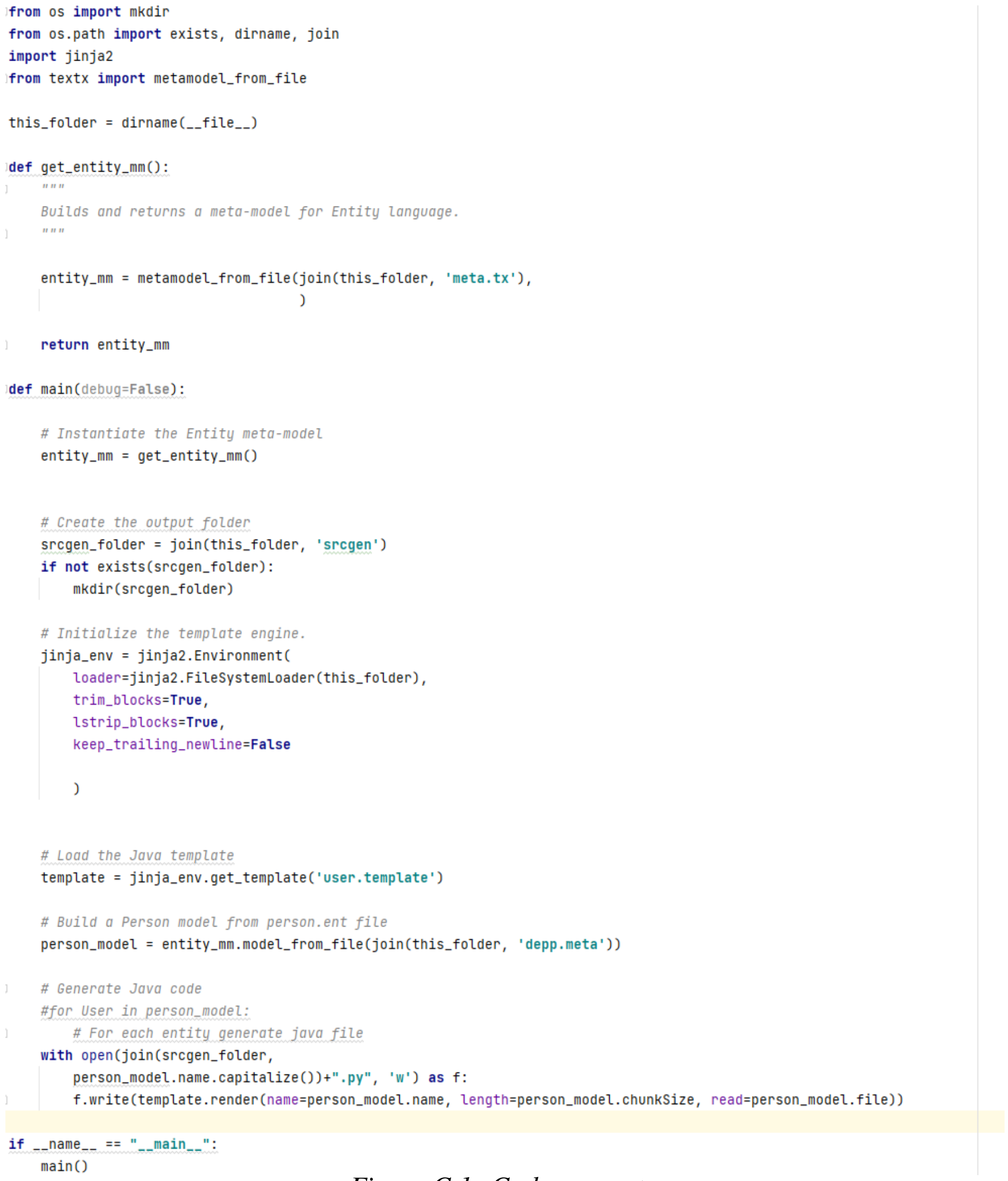
straightforward. Converting it to a .csv file. From there, the number of alarm's sent out to each user per week was saved into a text file the same way as on the modules page.
When processing these pages and the user data in them had been into a .TS file by the handler, the research class displayed all the user's data as points in the SVM. And showing both the time it took for each user to complete a task per week and how many warnings they received for that week as independent graphs.

Appendix C

Code generator

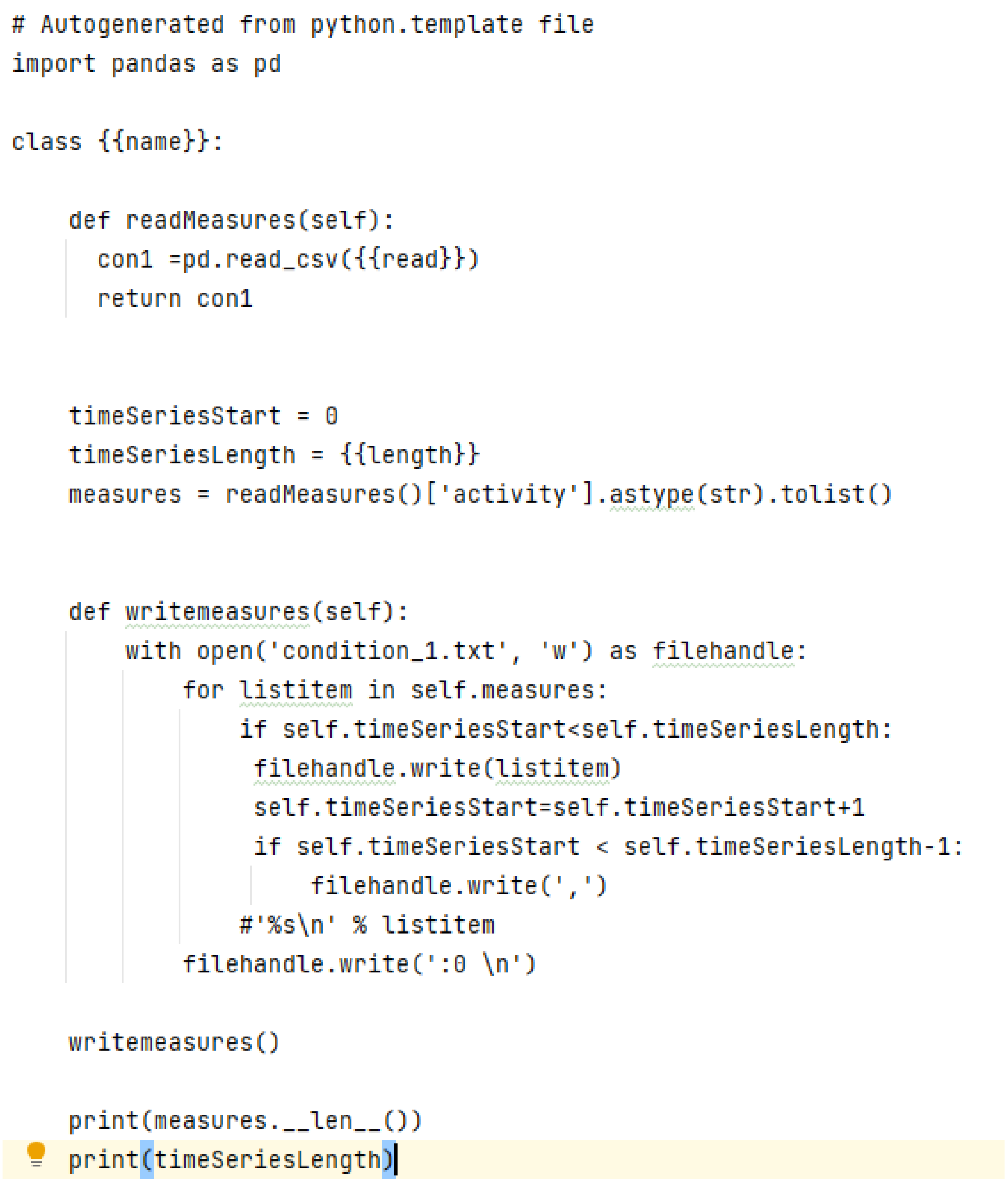
Generating code from TextX with the grammar was relatively straightforward. All that was needed was to create a model from the grammar in textX. From this model, it was possible to use jinja2 templates to generate code through a code generator. It would be possible to create an entire application through this code generation process.

Figure C.1 shows a code generator, and figure C.2 shows a template. By using the templates, code generator and model generated from the XAI_{Text} it is possible to create the entire pipeline.



```python
from os import mkdir
from os.path import exists, dirname, join
import jinja2
from textx import metamodel_from_file

this_folder = dirname(__file__)

def get_entity_mm():
    """
    Builds and returns a meta-model for Entity language.
    """

    entity_mm = metamodel_from_file(join(this_folder, 'meta.tx'),
                                    )

    return entity_mm

def main(debug=False):

    # Instantiate the Entity meta-model
    entity_mm = get_entity_mm()


    # Create the output folder
    srcgen_folder = join(this_folder, 'srcgen')
    if not exists(srcgen_folder):
        mkdir(srcgen_folder)

    # Initialize the template engine.
    jinja_env = jinja2.Environment(
        loader=jinja2.FileSystemLoader(this_folder),
        trim_blocks=True,
        lstrip_blocks=True,
        keep_trailing_newline=False

        )


    # Load the Java template
    template = jinja_env.get_template('user.template')

    # Build a Person model from person.ent file
    person_model = entity_mm.model_from_file(join(this_folder, 'depp.meta'))

    # Generate Java code
    #for User in person_model:
        # For each entity generate java file
    with open(join(srcgen_folder,
        person_model.name.capitalize())+".py", 'w') as f:
        f.write(template.render(name=person_model.name, length=person_model.chunkSize, read=person_model.file))

if __name__ == "__main__":
    main()
```

Figure C.1: Code generator



```
# Autogenerated from python.template file
import pandas as pd

class {{name}}:

    def readMeasures(self):
      con1 =pd.read_csv({{read}})
      return con1


    timeSeriesStart = 0
    timeSeriesLength = {{length}}
    measures = readMeasures()['activity'].astype(str).tolist()


    def writemeasures(self):
        with open('condition_1.txt', 'w') as filehandle:
            for listitem in self.measures:
                if self.timeSeriesStart<self.timeSeriesLength:
                 filehandle.write(listitem)
                 self.timeSeriesStart=self.timeSeriesStart+1
                 if self.timeSeriesStart < self.timeSeriesLength-1:
                     filehandle.write(',')
                #'%s\n' % listitem
            filehandle.write(':0 \n')

    writemeasures()

    print(measures.__len__())
    print(timeSeriesLength)
```

Figure C.2: Code generator template

Appendix D

Github repository

https://github.com/kingofthenorth871/deppresionDatasetGUI

Bibliography

Amisha, P. Malik, M. Pathania, and V. Rathaur (2019), Overview of artificial intelligence in medicine, *Journal of Family Medicine and Primary Care*, *8*, 2328, doi:10.4103/jfmpc.jfmpc_440_19. 2.1.1

Andreas C. Müller, S. G. (2016), *Introduction to Machine Learning with Python: A Guide for Data Scientists*, OReilly Media. 2.1.2, 2.1.4, 2.1.5, 2.1.6, 2.1.7, 2.1.8, 2.1.9, 2.1.10

Barredo Arrieta, A., N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera (2020), Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, *Information Fusion*, *58*, 82–115, doi:https://doi.org/10.1016/j.inffus.2019.12.012. (document), 1, 2.2.1, 2.2.2, 2.2.2, 2.2.3, 2.2.4

Christ, M., A. W. Kempa-Liehr, and M. Feindt (2016), Distributed and parallel time series feature extraction for industrial big data applications, *arXiv e-prints*, arXiv:1610.07717. 5.1.3

Cortez, P., and M. Embrechts (2011), Opening black box data mining models using sensitivity analysis, pp. 341–348, doi:10.1109/CIDM.2011.5949423. 3.5, 3.6

Cortez, P., and M. Embrechts (2013), Using sensitivity analysis and visualization techniques to open black box data mining models, *Information Sciences*, *225*, doi:10.1016/j.ins.2012.10.039. 2.2.4, 3.6

Davenport, T., and R. Kalakota (2019), The potential for artificial intelligence in healthcare, *Future Hospital Journal*, *6*, 94–98, doi:10.7861/futurehosp.6-2-94. 2.1.1

Dejanovi, I., R. Vaderna, G. Milosavljevi, and Z. Vukovi (2017), Textx: A python tool for domain-specific languages implementation, *Knowledge-Based Systems*, *115*, 1–4, doi:https://doi.org/10.1016/j.knosys.2016.10.023. 5.2.3

Fasmer, E., O. B. Fasmer, J. Berle, K. Oedegaard, and E. Hauge (2018), Graph theory applied to the analysis of motor activity in patients with schizophrenia and depression, *PloS one*, *13*, e0194,791, doi:10.1371/journal.pone.0194791. 3.4, 3.4

Fletcher, T. (2009), Support vector machines explained. 2.1.3

Fouquet, F., G. Nain, B. Morin, E. Daubert, O. Barais, N. Plouzeau, and J.-M. Jézéquel (2014), Kevoree modeling framework (kmf): Efficient modeling techniques for runtime use. 3.1

Fowler, M. (2010), *Domain-Specific Languages*, Addison-Wesley Professional, London. 1.1, 2.3.1, 2.3.2, 2.3.3

Garcia-Ceja, E., M. Riegler, P. Jakobsen, J. Torresen, T. Nordgreen, K. J. Oedegaard, and O. B. Fasmer (2018a), Motor activity based classification of depression in unipolar and bipolar patients, in *2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS)*, pp. 316–321, doi:10.1109/CBMS.2018.00062. (document), 8.2.1

Garcia-Ceja, E., M. Riegler, P. Jakobsen, J. T. rresen, T. Nordgreen, K. J. Oedegaard, and O. B. Fasmer (2018b), Depresjon: A motor activity database of depression episodes in unipolar and bipolar patients, in *Proceedings of the 9th ACM on Multimedia Systems Conference*, MMSys'18, ACM, New York, NY, USA, doi:10.1145/3204949.3208125. 3.4.1, 3.4.1, 4.1, 4.3

Goldstein, A., A. Kapelner, J. Bleich, and E. Pitkin (2013), Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation, *Journal of Computational and Graphical Statistics*, *24*, doi:10.1080/10618600.2014.907095. 2.2.4

Goodman, B., and S. Flaxman (2017), European union regulations on algorithmic decision-making and a right to explanation, *AI Magazine*, *38*(3), 50–57, doi:10.1609/aimag.v38i3.2741. (document), 1

Gurunule, D., and M. Nashipudimath (2015), A review: Analysis of aspect orientation and model driven engineering for code generation, *Procedia Computer Science*, *45*, 852–861, doi:10.1016/j.procs.2015.03.171. 2.3.1

Hartmann, T., A. Moawad, F. Fouquet, and Y. Le Traon (2017), The next evolution of mde: A seamless integration of machine learning into domain modeling, pp. 180–180, doi:10.1109/MODELS.2017.32. 1.1, 3.1, 3.1, 3.1

Haukeland-University-Hospital (2021a), Intromat, `https://intromat.no/`, accessed: 2021-08-06. 4.1.1, 4.1.4

Haukeland-University-Hospital (2021b), Emeistring, `https://helse-bergen.no/emeistring/`, accessed: 2021-08-06. 4.1.1

Lacasa, L., B. Luque, F. Ballesteros, J. Luque, and J. C. Nuño (2008), From time series to complex networks: The visibility graph, *Proceedings of the National Academy of Sciences*, *105*, 4972, doi:10.1073/pnas.0709247105. 3.2, 3.3

Lipton, Z. (2016), The mythos of model interpretability, *Communications of the ACM*, *61*, doi:10.1145/3233231. 2.2.2

Max Halford, V. L. (2019), Ethik, https://pypi.org/project/ethik/, accessed: 2021-08-06. 5.2

McCarthy, J. (2007), What is artificial intelligence. 1, 2.1.1

Núñez, A., L. Lacasa, J. Gomez, and B. Luque (2012), *Visibility Algorithms: A Short Review*, doi:10.5772/34810. 2.4.2, 2.4.2, 2.5, 3.3

pandas development team, T. (2020), pandas-dev/pandas: Pandas, doi:10.5281/zenodo.3509134. 5.2.5

Pantanowitz, L., G. M. Quiroga-Garza, L. Bien, R. Heled, D. Laifenfeld, C. Linhart, J. Sandbank, A. Albrecht Shach, V. Shalev, M. Vecsler, P. Michelow, S. Hazelhurst, and R. Dhir (2020), An artificial intelligence algorithm for prostate cancer diagnosis in whole slide images of core needle biopsies: a blinded clinical validation and deployment study, *The Lancet Digital Health*, *2*(8), e407–e416, doi:https://doi.org/10.1016/S2589-7500(20)30159-X. (document)

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011), Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research*, *12*, 2825–2830. 5.1.2

Plotly (2021), Plotly dash, https://plotly.com/dash/, accessed: 2021-08-06. 5.2.4

Python software foundation (2021), Python, https://www.python.org/community-landing/, accessed: 2021-08-06. 5.1.1

Ronacher, A. (2021), Jinja2, https://pypi.org/project/Jinja2/, accessed: 2021-08-06. 5.2.2

Sharma, S. (2015), Rise of big data and related issues, in *2015 Annual IEEE India Conference (INDICON)*, pp. 1–6, doi:10.1109/INDICON.2015.7443346. 1.1

tutorialspoint (2021), Graphtheory, https://www.tutorialspoint.com/graph_theory/index.htm, accessed: 2021-08-06. 2.4.1

Zhang, Y., and C. Ling (2018), A strategy to apply machine learning to small datasets in materials science, *npj Computational Materials*, *4*, doi:10.1038/s41524-018-0081-z. 8.2.1

Zhou, D., L. Miao, and Y. He (2018), Position-aware deep multi-task learning for drugdrug interaction extraction, *Artificial Intelligence in Medicine*, *87*, 1–8, doi:https://doi.org/10.1016/j.artmed.2018.03.001. (document)