# Classifying Collected Sensor Data for Mental Health Prediction using General, Personalized and Hybrid Machine Learning Models

Anders Kvalvik Kvernberg

Thesis submitted for the degree of
Master in Informatics: Programming and Networks
60 credits

Institutt for informatikk
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2018

# Classifying Collected Sensor Data for Mental Health Prediction using General, Personalized and Hybrid Machine Learning Models

Anders Kvalvik Kvernberg

Classifying Collected Sensor Data for Mental Health Prediction using
General, Personalized and Hybrid Machine Learning Models

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

## Abstract

Mental health is a growing concern in today's society. Mental disorders such as bipolar disorder and depression affects one in four adults at some point in their life, and is one of the leading causes of disability. This greatly affects the individual quality of life as well as the world economy. There is a need for new treatment options, and with the rise of machine learning and wearable electronics over the last years it gives us a new approach. While we have new means of data collection, the data will still have to be labeled by specialists which is a resource-heavy task. The use of general models are less reliable on data collection before it can be taken into use, but suffers in terms of performance. Personalized models require a lot of data, but has better performance. An alternative to this is proposed through hybrid and user adaptive models which tries to get the performance of the personalized model with a reduced amount of data.

This thesis explores the classification of mental health states based on motor activity and speech data by using different machine learning classifiers, techniques and models. We create general, personalized, hybrid and user adaptive models to classify migraine attacks based on motor activity from subjects with bipolar disorder and emotional states based on speech activity from actors. From this it is concluded that by using hybrid and user adaptive models it is possible to get close to the performance of a personalized model but with significantly less data.

# Contents

# List of Figures

# List of Tables

# Acknowledgements

First and foremost I would like to thank postdoctoral fellow, and supervisor, Enrique Garcia Ceja for his motivation, inspiration and help. Our weekly meetings kept me on track even when motivation was low. Your supervision has been nothing but amazing and I greatly appreciate it. Thank you.

I would also like to thank professor, and supervisor, Jim Tørresen for his input and comments. When I was lost in my own world they pulled me out again and made me look at the thesis from another angle.

Last, but not least, I thank my friends and family for their support and words of encouragement. Especially to my parents who also helped with feedback and proofreading.

# Chapter 1

# Introduction

## 1.1 Motivation

Mental health is a big topic in today's society. According to numbers from the world health organization a total of 27% of the adult population (aged 18-65) has experienced at least one of a series of mental disorders in the last year [13] and another study shows that more than 50% of the general population in middle- and high-income countries will suffer from at least one mental disorder at some point in their life [52]. The same study estimates that the global direct and indirect costs of mental disorders are US$2.5 trillion.

In other words; mental disorders are a major public health problem with implications for both the individual and the society.
In a world where we wear more and more smart devices, such as smartphones and smartwatches, there are more opportunities for treatments to be explored. Predictions of mental states in mental health based on data collection from these devices are something that is being researched, but has not yet established itself as a common tool in treatment and prevention of mental disorders. Reasons for this include that it is hard to train classifiers that perform at an acceptable rate and that collecting labeled data is difficult as well. Even though the collection of data is easier with the use of smart devices, it would still have to be labeled by a professional, at least for the use of supervised learning. Still the rise of new research, new libraries like scikit-learn and tensorflow and a growing focus on mental health make this technology more and more accessible.

The project is part of the research project INtroducing personalized TReatment Of Mental health problems using Adaptive Technology (INTROMAT), which is financed by the Research Council of Norway as one of the three IKTPLUSS lighthouse projects. The goal of the INTROMAT projects is to increase access to mental health services, with the help of innovative ICT. The ROBIN group at UIO will be contributing to the project by developing patient monitoring and support systems running as smartphone apps. This will provide the patients and clinicians with a support system for use in treatment.

## 1.2 Problem statement

Earlier work has concluded that one can achieve good classification results based on analysis of motor and voice activity for mental health state prediction [21, 29, 36]. These studies have resorted to using general models, which are trained on a dataset consisting of multiple subjects before being tested on a previously not seen target subject, and personalized models which are trained and tested on the same subject. Personalized models are clearly superior to the general models in terms of performance, but the downside is that there would have to be a lot of available data collected from each individual when the model is created in order to personalize it as best as possible. This is less than ideal as one would like to start using the classifier as soon as possible and not after several weeks, if not months, of data collection which is very difficult in the domain of mental health. The use of generalized models did provide some favorable results as well, but not as good as the personalized models. The gap between these and the personalized ones are so big that personalized models are easily favored in terms of performance, but not in terms of practicality.

The solution to this could be to make use of hybrid models. A hybrid model could be made from creating a generalized model and then adapt it to each individual user on the go. This has proved(section 2.4.3) to provide almost as good results as personalized models, and could be created a lot faster as you would not have to gather as much data for each user.

### 1.2.1 Research questions

To research the problem stated above the following research questions are proposed:

1. How do general and personalized models perform in the mental health domain?

2. Is it possible to create a hybrid model that harbors the advantages of both the general and personalized model?

3. Can such a hybrid model be built by adding data from the target user to the training data?

4. Can such a hybrid model be built using transfer learning techniques?

To properly research these questions a number of different classifiers will be tested and compared. The classifiers will be trained and tested once for each of the following model types:

**General model** The general model is made with the mindset "one-size-fits-all". It is trained on a dataset with different subjects before it is applied on a new target subject. It has the advantage that no data collection is required before applied to a new subject, but suffers in terms of performance.

**Personalized model** The personalized model is made for each individual subject and trained solely on data from the subject. It is superior to the general model in terms of performance. The downside is as stated earlier that a lot of data would have to be collected before the model could be trained and applied.

**Hybrid model** The hybrid model is created from training the classifier on both other subjects and the target subject. The model can be trained by adding more and more training data from the target subject as it becomes available. It has to be balanced enough that it requires a small amount of samples from the target user for the improvement to be worth not using a personalized model.

**User adaptive model** The user adaptive model is created from using transfer learning techniques [41]. This method adapts the classifier to the target subject after learning the base features from the source dataset instead of just adding the target data to the training data. It is strictly speaking a hybrid model, but for the sake of differentiating between them it will be called a user adaptive model from here on.

## 1.3 Contributions

This thesis provides research on whether it is possible to predict migraine attacks in subjects with bipolar disorder based on motor activity, which to the best of the author's knowledge has not been done before.
The main contribution lies in the use of hybrid and user adaptive models in the mental health domain. The research on the use of hybrid models in this domain is very limited, and the use of transfer learning techniques to create a user adaptive model for use in the mental health domain is also untested to the best of the author's knowledge. Furthermore this thesis shows that both the hybrid and user adaptive model can reach almost the same performance as the personalized model, but with significantly less samples.

## 1.4 Limitations

There are certain obstacles to keep in mind during this project.
First of all the process of collecting data in the domain of mental health is long and tedious. One would have to find subjects that are willing to participate in the study, using the data collection tool and seeing a specialist on a regular basis. Even when these conditions are met it is not given that the subject will experience the conditions that is being researched in the given time period. For example a subject with bipolar disorder does not necessarily experience a manic period while data is being collected.

These obstacles tend to provide very limited datasets that can be hard to work with and provide poor results. To get more results a public dataset on emotions prediction from speech activity was utilized. This dataset was however collected in a controlled environment from actors and are not necessarily completely representable.

Furthermore it is important to mention that the thesis will not focus on the performance of the different classifiers but rather on the relative performance of the models. While several different classifiers will be tested, they will mostly be trained with default parameters and will not be fine-tuned.

## 1.5 Overview

**Chapter 2 - Background**

This chapter provides background on the domain of mental health, machine learning and the classifiers that are used. Related works are then explored and accounted for before the different metrics that are used when reviewing model performance are described.

**Chapter 3 - Building and testing**

Here the libraries that were used in the implementation process are covered before the general, personalized and hybrid models are explained in detail.

**Chapter 4 - Experiments and results: Migraine attack detection**

Chapter 4 covers the experiments that were conducted on the migraine attack dataset and the corresponding results. It also provides details on the dataset itself and the feature extraction process.

**Chapter 5 - Experiments and results: Speech mood recognition**

This chapter covers much of the same information as chapter 4, but for the emotions prediction dataset. It is not as detailed as chapter 4 as it was only meant to provide additional results to aid reaching a conclusion.

**Chapter 6 - Conclusion**

Finally the results are summarized and a conclusion is set. Future work is also discussed.

# Chapter 2

# Background

## 2.1 Mental health

Mental health defines our emotional and psychological well-being and is a growing subject in today's society. A myriad of factors affect a person's mental health, such as social life or how much stress one experience in the everyday life. When a deteriorating mental health starts to affect one's life it can be classified as a mental illness or mental disorder. Some of the most common types of mental disorders are depression and bipolar disorder [37]. While bipolar disorder is not a main area of research in this specific project it is relevant in one of the datasets and is a key factor in future work.

### 2.1.1 Emotions

Emotions is a complex subject that is hard to define. While most people know what emotions are, they might be hard to describe and explain. There is seemingly no consensus on a concrete definition of emotions, but it can be described as any experience with a degree of pleasure or displeasure [10]. Typical emotions are anger, sadness, happiness, and anxiety. Emotions such as these, impact our life everyday and is thus a quite important feature. When one displays emotions other people react accordingly, but it can be used in other situations as well. Emotions can be a symptom of a mental or physical illness and learning how to predict them can be quite useful.

Since emotions play such an instrumental part in our everyday life, they can be extracted in several different ways. People often get an idea of how another person is feeling based on facial expressions, conversation, and body language, and similar features can be of use in machine learning. Training a classifier that can predict emotions based on data like speech and motor activity would be quite useful in predicting more complex things like bipolar disorder.

### 2.1.2 Bipolar disorder

Bipolar disorder is a serious mental illness that causes periods of depression and elevated mood, also known as mania [3]. People affected by bipolar disorder, face difficulties on daily basis, even between episodes, as they are still bothered by mood swings that are below the criteria of what one would call an episode. It can be divided into roughly two categories; Bipolar type I and type II. Bipolar I disorder is defined by manic episodes usually followed by a depressive episode, while bipolar II disorder is defined by a series of depressive and hypomanic episodes. Hypomanic episodes are not as intense as a manic episode would be.

The periods of depression tend to be quite severe, and the risk of suicide is quite high, more than 6% over 20 years [3]. Self-harm occurs in 30-40 percent of the cases [3]. The manic and hypomanic episodes are spans of time where the mood is quite elevated. A manic person may experience, among other things, several of the following behaviors: rapid speach, short attention span, hypersexuality and increased goal-oriented activites. Needless to say a good form of treatment is desired, not only because it is a terrible illness to live with, but also because it is quite expensive for the society. Bipolar disorder is the 6th leading cause of disability in the world [15].

Today's treatment relies on medicine and therapy, but for it to have effect it has to be administered at the right times. If a manic or depressive episode progresses too far the medicine will not have the same effect. Therefore a method to predict such episodes is something of interest, and with the rise of smartphones and wearables such methods are very close to reality.

### 2.1.3 Migraine

Migraine can be described as a moderate to severe headache experienced as a throbbing pain on one side of the head [38]. Other symptoms such as nausea, vomiting and increased sensitivity to light or sound may also occur. Migraine is a common condition affecting approximately one in 5 women and one in 15 men [38], but the frequency of it varies. Some experience migraine attacks several times a week while some can go multiple years without an attack.

As with a lot of common conditions there are several different types of migraine where the following are some of the most common:

- Migraine with aura
  Symptoms such as flashing lights are experienced before the migraine attack occurs.

- Migraine without aura
  The migraine attack occurs without any preluding symptoms.

- Migraine aura without headache
  The headache itself does not occur, but one may experience the aura or other migraine related symptoms.

As migraine attacks has such uncomfortable symptoms as headaches and nausea, it may affect the quality of life of the ones suffering from it. There is no known treatment, but some people find that sleeping or relaxing in a darkened room helps. Detecting an oncoming or ongoing migraine attack as early as possible may help the patient. By recording the motor activity of subjects with migraine it might be possible to classify such attacks.

## 2.2 Machine learning

Machine learning is the use of algorithms that can learn and make predictions based on data and is a field that has been experiencing a major increase in popularity over the last years. A more formal definition was proposed by Tom M. Mitchell as "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." [39].
The domain of machine learning covers a myriad of different fields, such as supervised and unsupervised learning, and is often used for classification problems [35].
A classification problem is the task of identifying which class, or classes, a new observation belongs to given a set of known classes based on training data. How the classification is made and what the training data is depends on the type of learning. Supervised learning learns based on a set of observations with a matching set of labels, much in the same way a human would learn from examples, while unsupervised learning learns from just a set of observations with no labels.

Over the course of this project several different state-of-the-art machine learning algorithms using supervised learning will be used to correctly classify mental health states.

### 2.2.1 Supervised learning

Supervised learning is a machine learning task that uses labeled data to help the learning process. The classifier receives input which it maps to a label based on labeled data defined as training data. The training data can be viewed as examples that the classifier learns from so that it can recognize similar input and correctly label it later on, much like the learning process a person goes through.

When using supervised learning the data collection process is very important. The training data has to contain enough data representing each possible class so that the classifier has enough data to learn from. The amount of data depends on the domain and number of classes. An optimal dataset contains enough data with enough variance so that the classifier can even recognize previously unseen samples and correctly label them.
Furthermore the data has to undergo a feature extraction-process. Feature extraction is done to improve the performance of the classifier, but also to reduce computation time and storage space [26]. Raw data is not ideal when working with most machine learning algorithms and thus a number of features are extracted from it and put into what is referred to as a feature vector which is then given to the algorithm as input.

### 2.2.2 Deep learning

Deep learning is an area within machine learning that focuses on recognizing data representations [14]. The area has gained massive popularity over the last years, some of which can be attributed to the steady increase in computing power. Deep learning can often be computationally heavy as it can involve using massive, multilayered neural networks to discover structure and patterns in large datasets that are impossible to detect by humans. Deep learning has contributed to several important breakthroughs in areas like image, speech and text classification which again can be used in several different fields. Part of the reason why deep learning is so suitable for such tasks is that contrary to other machine learning techniques it can be fed raw data without any feature extraction. It then extracts features on its own, which is used for classification.
When using machine learning in the domain of mental health a lot of the data that is used comes from speech and motor activity, but text data is also somewhat used. As deep learning also supports techniques such as transfer learning, which makes it easy to adapt models to new users, it is a viable approach in this project.

### 2.2.3 Transfer learning

Transfer learning is the process of learning a new task based on knowledge from an allready known task [49]. When the problem you are trying to solve has a small amount of data transfer learning is a good option to use in order to train a well performing network [55]. This makes it a suitable approach when creating hybrid models as we wish to have the best performance possible from as litle data as possible. A real world example of this is networks trained on ImageNet [28]. Training a modern convolutional network on ImageNet is time consuming, so people have released their trained convolutional networks so that others can use them and finetune them to their specific tasks and domains [51].

The most common way of doing transfer learning is to train a base network and then copy the first n layers to the first n layers of a target network [55]. The last layers of the network are then initialized with random weights and trained towards the target task. It is common to not alter the weights copied from the base network.

There are however a few different ways of doing transfer learning and the technique that is being used for this thesis is closely related to inductive transfer learning [41].

**Definition 1** (Inductive transfer learning). *Given a source domain $D_S$ and a learning task $T_S$ , a target domain $D_T$ and a learning task $T_T$, inductive transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $D_T$ using the knowledge in $D_S$ and $T_S$, where $T_S \neq T_T$.*

When a deep neural network has been trained on a source domain $D_S$ for a task $T_S$ the layers can be split into two; the feature layers and the

classification layers.
The feature layers extract and learn the features of $D_S$ and can be reused for a new learning task $T_T$.
The classification layer is specific to learning task $T_S$ but the weights can be retrained, or tuned, to fit the new learning task $T_T$. By using the feature layers that is achieved by training a deep neural network on $D_S$ we can train just the classification layer on $D_T$ for $T_T$ and reduce the number of samples needed in $D_T$.

## 2.3 Supervised machine learning algorithms

In the following sections I will look at some of the algorithms used in earlier studies [20, 21, 29, 36] in the mental health domain and the ones that will be used in this project.

### 2.3.1 Support Vector Machines

Support Vector Machines (SVMs) is one of the most popular algorithms used for data classification in modern machine learning [35]. It was developed by Vapnik [53] and is quite popular due to very impressive performance on reasonably sized datasets, though not as good on very large datasets. The goal of the SVM is to find a linear classifier that separates the data correctly while it maximises the margin. This linear classifier is called the optimal seperating hyperplane [25].

### 2.3.2 Decision Tree

Decision trees are tree-like structures where classification is broken down into a set of choices about each feature, starting at the root of the tree and then evaluating one feature at a time as you progress down the tree to the leaves where the classification is decided [35]. A previous study [36] noticed that algorithms based on decision trees, such as C4.5 [45] which was used in that particular study, achieved the best results on average.

### 2.3.3 Random Forest

The Random Forests, or random decision forests, algorithm was first developed by Tin Kam Ho [27] in 1995. An extension to the algorithm was later developed by Leo Breiman [8] that combined Breiman's idea of "bagging" with Ho's random selection of features. Random Forest is an example of ensemble learning, a form of machine learning that generate several classifiers and combine them to get better results than one would by using only one of them.

The algorithm combines several decision tree classifiers, from the idea that many trees(a forest) is better than one, and is excellent for handling datasets with many features, as tree induction methods automatically choose the most discriminating features in the data [8], which makes it excellent for this problem.

Random forest is able to quite fast. It is quite easy to parallelize as the trees do not depend upon each other and because it searches over a reduced number of features [35]. Contrary to Support Vector Machines they handle big datasets very well [35] which may be quite useful in this project.

### 2.3.4 K-nn

The k-nearest neighbor algorithm(k-nn) is one of the most straightforward machine learning algorithms. The principle behind it is finding the $k$ number of data points from the training set that are closest to the point you want to classify, and then assign a label to it based on a majority vote. The distance can be calculated in a number of ways, but using Euclidean distance is the most common. k-nn suffers from the curse of dimensionality [35] as the computational costs get higher as the number of dimensions grow. The distance to other datapoints also increase which might have an impact on the performance of the algorithm.

### 2.3.5 Gaussian Naive Bayes

The naive bayes classifiers are a set of machine learning algorithms that are based on applying Bayes' theorem with the naive assumption of independent features, thus naive bayes [35]. In this way it kind of avoids the curse of dimensionality. Given the naive approach of independent features and the simplicity of the algorithm it is surprising that the classifier works so well on real world problems but the classifiers has been shown to be comparable to other machine learning algorithms. The Gaussian Naive Bayes assumes the feature distribution to be gaussian and classify the input data based on the mean and variance of each feature in each class.

### 2.3.6 AdaBoost

AdaBoost(Adaptive boosting) is a meta-estimator developed by Yoav Freund and Robert E. Schapire [22] in 1996. AdaBoost fit a sequence of weak learners on repeatedly modified versions of data [2]. The predictions from all of the classifiers are then combined through a weighted sum to produce the final prediction. The use of several weak learners that are each trained on data sets where the weights of wrongly classified instances are adjusted makes it easier for the AdaBoost classifier to correctly classify outliers and makes it a valid choice in classification. A popular choice for the base-estimator is a decision tree.

### 2.3.7 Bagging

Bagging(bootstrap aggregating) is another meta-estimator. It was proposed by Leo Breiman in 1994 [7]. The algorithm fits several instances of an estimator on random subsets of the training set and then aggregates their predictions, usually by a majority vote, to compute the final prediction. Bagging is a popular way to improve the results of a model by making it more robust and accurate. It works best with more complex models such as decision trees.

### 2.3.8 Multilayer perceptron

The multilayer perceptron is a feedforward neural network that consists of at least three layers of nodes, one input-layer, one or more hidden layers and one output-layer. The nodes are connected to the nodes in the previous and next layer by weights, which is what is adjusted when training the network. When training the input is fed through the network by multiplying each weight by the input. At each node the sum of the result from each connected node in the previous layer is calculated and fed on to the next layer if there is one. An error is calculated by using a loss function which is fed bakwards through the network adjusting the weights by a process called backpropagation.

Multilayer perceptrons can be used to classify and approximate extremely complex problems and are widely used. It is viewed as one of the simpler deep learning algorithms.

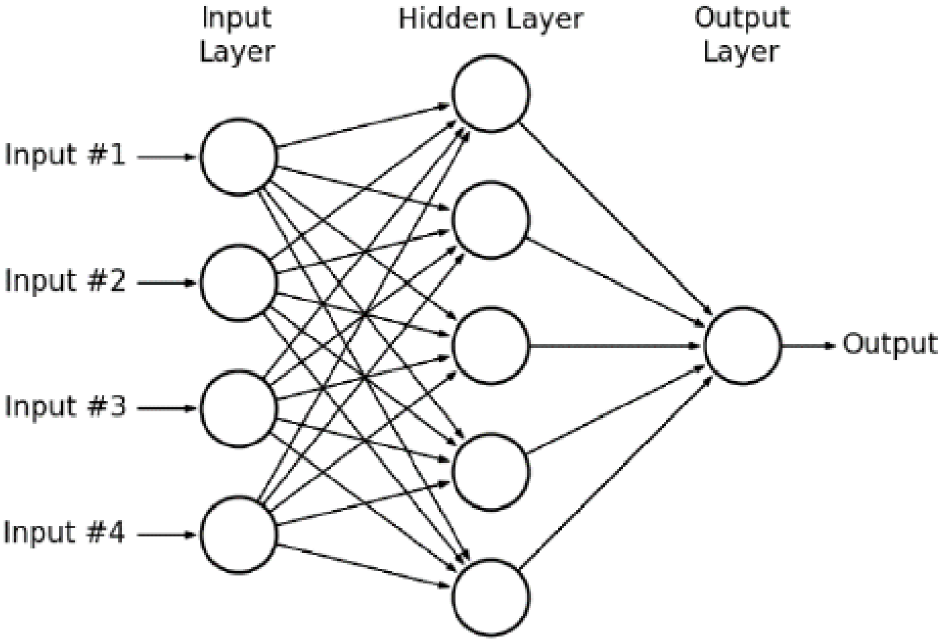

Figure 2.1: Illustration of a multilayer perceptron from [4]

## 2.4 Relevant work

There have been several studies that have collected sensor data from wearables and smartphones to predict mental states, but to the best of my knowlegde there are not many studies that research the use of hybrid models in mental state prediction. There have been several studies (section 2.4.3) done on Human Activity Recognition and some on stress recognition, which is closely related to mental health, using hybrid models. In the following sections we will take a look at the different studies done on mental state prediction and the use of hybrid models. As this project will use two different data sets; one of speech activity and one of motor activity there is one section for each of those two fields.

### 2.4.1 Analysis of voice activity for the mental health domain

The nature of the symptoms in bipolar disorder, and similar mental disorders, strongly suggests that doing voice analysis will provide some results in terms of mood state predictions. It is not hard to believe that the data collected from analysis of voice activity during a depressed episode will differ quite a bit from data collected during a period of mania or euthymia. In fact both the Hamilton Depression Rating Scale(HAMD) and the Young Mania Rating Scale(YMRS) have items related to changes in speech Several studies [21, 29] demonstrates that one can increase the accuracy of classification by extracting features from recorded speech data, such as information on the pitch of the voice. Software for extraction of such data has been developed, for example the pyAudioAnalysis [44] tool. One can also look at the frequency and duration of phone calls to get an idea of the social activity of the patient.

An earlier study [29] done at the university of Michigan made a point of gathering data from six patients over the course of six months to a year using cell phone-based recording software to gather data in an unstructured and uncontrolled environment. The software recorded all outgoing speech during phone calls.

The study used a support vector machine with linear and radial-basis-function kernels trained on a data set consisting of statistics of low-level features such as pitch and RMS energy. The classifier is trained using participant independent modeling to understand how speech is modulated as a function of mood state. The findings show that hypomania and depression can be differentiated from euthymia using speech-based classifiers trained on both structured data collected during the weekly clinical assessments, and unstructured cell phone recordings. However the classifier is most accurate when modeling structured data, and it is hypothesised that this is because the structured data is the only set that has associated labels. It's also more effective at classifying hypomanic states than depressive states outside of controlled environments.

A later study [21] by Faurholt-Jepsen also used voice features collected during phone calls to classify a patient's state, but also tried to see if combining voice features with automatically generated objective smartphone data on behavorial activities and electronic self-monitored data on illness activity would increase the accuracy. The authors of the study developed an electronic monitorin system for smartphones in 2010 called the MONARCA system. The system would collect automatically generated objective smartphone data and was later extended to collect and extract voice features from phone calls. The automatically generated objective data was described as level of social activity, mobility and phone usage. Data used was number of incoming and outgoing text messages and phone calls, duration of phone calls, changes in cell tower IDs and number of times and how long the smartphones's screens were turned on.

The study made use of the Random Forest classification algorithm because of the reasons stated earlier regarding the algorithm. The study showed the voice features were more accurate in classifying manic states than depressive states, as the previous study did, and that combining voice features, electronic self-monitored data and automatically generated objective data increased the accuracy slightly. The accuracy based on voice features alone was in the range of 61%-74% and increased slightly when combined with automatically generated objective smartphone data and electronic self-monitored data.

### 2.4.2 Analysis of motor activity for the mental health domain

A study [36] released in 2016 tried using motor activity as well as voice activity and self-assessment data to classify with high confidence the course of mood episodes in bipolar patients. The study monitored 10 patients affected by bipolar disorder over a 12 week period with clinical assessments every 3. weeks. Evaluating the motor activity of patients with bipolar disorder has always been an essential part of psychiatric evaluations, but the clinical measurements are largely subjective and comes from caregivers' observations. The use of smartphones to collect the data and using machine learning classifiers could make for more objective measurements and classifiers which again might improve the classification.

The study used and compared several different classifiers. They noticed that the use of decission tree algorithms performed better on average and stuck to reporting results from the C4.5 algorithm. This study experienced that the information obtained from the frequence domain features of the accelorometer, lead to better performance than the information extracted from audio and that when combined there was only a slight improvement.

In the end the study reported an average classification accuracy of 85.56% and over 80% for all precision and recall values for each mental state of the different patients.

There have been other studies that have not used machine learning, but studied the motor activity in subjects with mental disorders. One study [5] studied motor activity recorded from an actigraph in both schizophrenic and depressed patient. Motor activity was significantly

reduced in both groups, but schizophrenic patients had motor activity that was less complex and more structured than both the depressed subject and the control group.
Another study [40] used an actigraph to record motor activity from 29 adult with Late-life depression and 30 healthy controls. They found that patients with late-life depression has a significant reduction in general physical activity compared with the controls.

### 2.4.3 Hybrid and User Adaptive Models for the mental health domain

There have been several studies done on hybrid or user adaptive models using various approaches.
The most common approach seems to be creating a hybrid model using cluster based methods, as can be seen in [1, 23, 24, 54]. In [23] cluster based methods are used to follow a similar users-approach to classify stress levels. The idea is that a model created based on data from users similar to the user the model is created for will result in a better suited model with less noise than a general model. A related approach is used in [24] for human activity recognition and in [54] for stress evaluation. [1] uses a completely different approach where it integrates supervised, unsupervised and active learning in a technique for activity recognition they have called StreamAR. The technique creates clusters for each activity using supervised learning that is continously updated using unsupervised and active learning.
While these cluster based approaches have yielded good results, cluster based methods does not necessarily perform well on data with a large amount of features, and they make the assumption that the clusters found in the target data are also found in the source data, which is not necessarily the case.
Another approach was used in [42] where an adaptive approach was tried. They used a Bayesian maximum a posteriori classifier with user feedback for adaptation to recognize activities and environments. The user was simply asked if the given classification was correct and the user input was used to adapt the distribution parameters. Hong Lu et al. uses a similar method in [34], but in addition to the user labeling new samples they also try unsupervised learning. While this method of user feedback works well on objective classification domains, such as activity and environment recognition, it would probably not work as well in other domains such as mental state prediction.
Fallahzadeh and Ghasemzadeh use a combination of transfer learning and unsupervised learning in [19] by training a general model on a source dataset and then using cross-subject transfer learning to label an unlabeled target dataset, thus making it easy to create a personalized model for the target user. They report achieving over 87% accuracy on average with this method when testing it in activity recognition on the real-world UCI "Daily andSports Activities Data Set" [16].

In [33, 47] we can see examples of the approaches to be taken in this project. Lockhart and Weiss explore hybrid models for activity recognition in [33] by taking the same approach that will be taken in this thesis; transferring data from the target subject to the training data. They conclude that personalized and hybrid models will consistently outperform general models, but that hybrid models generally perform worse than personalized ones.

Transfer learning is explored by Rokni et al. in [47] by conducting human activity recognition using convolutional neural networks. This is similar to what will be done in this project, though multilayer perceptrons will be used instead of convolutional neural networks. They show that by randomly acquiring 3 labeled instances for each activity and retraining the classification layer the results are improved significantly from hybrid models.

## 2.5 Preprocessing

When working with data for machine learning purposes it most often requires some preprocessing before it can be used. The data might have some missing fields which needs to be handled, it might need to be scaled down to the same numeric range or it might need to be sampled so as to create an either smaller or larger dataset.

### 2.5.1 Normalization

Feature normalization was used in order to standardize the range of values. This is done in order to prevent some features to completely overrule the others in cases where their value is a lot greater than the others. The values were scaled to the range [0, 1] by the following formula:
$x' = \frac{x - min(x)}{max(x) - min(x)}$.

The features were all normalized individually, but all the data could not be normalized together in one batch. The different model types used different strategies when normalizing the data as they are all trained in different ways.

- The general model normalized the training data and used the max- and min-parameters to normalize the test data. If the data had all been normalized together, the data sets would have affected each other which could have had impact on the results. This was done within each fold.

- The personalized model normalized the features of each subject individually. This was done only once for each user and not within each fold as both the training data and test data came from the same subject.

- The hybrid model normalized all the subjects together. As the hybrid model was trained on data from all the subjects, it was not deemed necessary to normalize it in the same way as the in general model.

### 2.5.2 Random Oversampling

Random oversampling [32] is the process of picking samples with replacement at random from one or more minority classes resulting in a balanced dataset with several duplicates. It is a very naive, but computationally cheap way of balancing a dataset. While this will balance the class distribution, it will probably result in a less robust model. A visualization can be seen in figure 2.2.

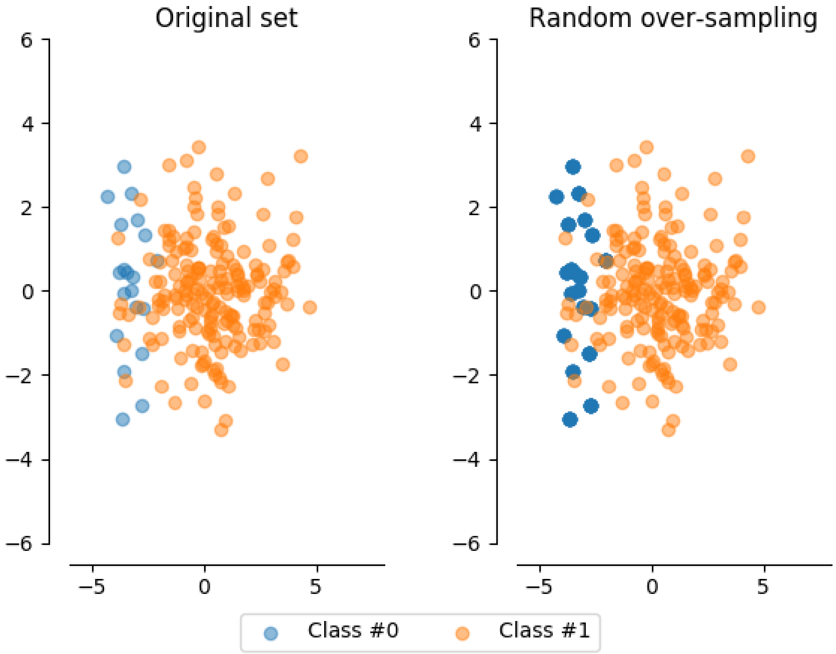

Figure 2.2: Illustration of random oversampling from [30]

### 2.5.3 SMOTE

SMOTE, or Synthetic Minority Over-sampling Technique is a technique for oversampling an imbalanced dataset that was first proposed in the Journal of Artificial Intelligence Research [11]. The minority class is oversampled by generating synthetic samples from the line segments to some, or all, of the *k nearest neighbors* of any minority class sample. An example from [11] that shows calculation of synthetic samples can be seen below.

```
Consider a sample (6,4 )and let (4,3 )be its nearest neighbor.
(6,4 )is the sample for which k-nearest neighbors are being identified.
(4,3 )is one of its k-nearest neighbors.
Let:f1_1=6 f2_1=4 f2_1-f1_1=-2
f1_2=4 f2_2=3 f2_2-f1_2=-1
The new samples will be generated as
(f1',f2')= (6,4)+ rand(0-1)* (-2,-1)
rand(0-1)generates a random number between 0 and 1.
```

This example shows the original algorithm for generating synthetic samples, but a number of different such algorithms have later been developed. In figure 2.3 we can see the dataset visualized both before SMOTE is applied and after the different algorithms have generated new samples. It is worth comparing this visualization to figure 2.2 to get a clear view of what affect the different methods have on a dataset.

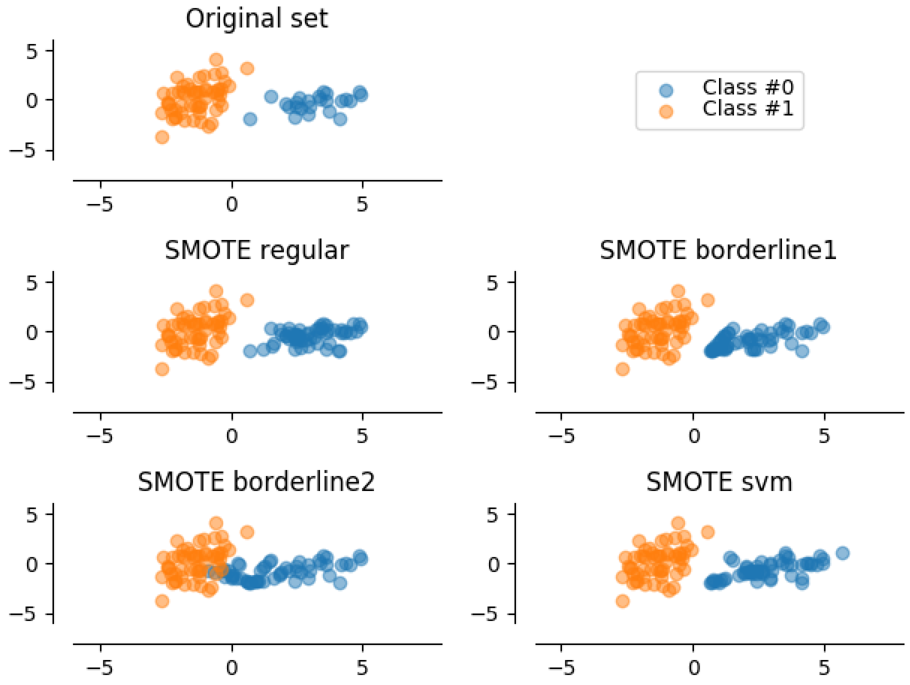

Figure 2.3: Illustration of SMOTE from [31]

## 2.6 Metrics

When measuring the performance of a model it's important to look at the correct classification metrics. Knowing which ones to look at might be difficult, but it is easier if you look at the problem domain when choosing. There's a myriad of different metrics one can look at, but for this project 7 different ones have been chosen; accuracy, precision, recall, specificity, F1-score, Matthew's correlation coefficient and AUC. In the following sections these metrics will be explained in more detail and why they are useful.

### 2.6.1 Accuracy

Accuracy is the share of correct predictions out of the total amount of predictions said to be right. It is often a good metric to have in the mix when measuring the performance of a model, but it's often not a good idea to solely look at that as it does not tell much of what might be wrong. In an imbalanced class problem you might get high accuracy, but the performance might still be bad. Given 300 samples where 290 of them are True and 10 of them are False the classifier could label all the samples as True and still achieve an accuracy of $\frac{290}{300} \approx 0.96$. This goes to show that to fully measure the performance of a classifier, it is important to look at more metrics than the accuracy alone.

### 2.6.2 Confusion matrix

To fully understand precision and recall it is important to understand what a confusion matrix is and what data it holds.
The Confusion matrix contains instances of the *true* class labels vs the *predicted* class labels. This can be visualized below using the example from section 2.6.1 adjusted for explanatory purposes.

| | Predicted: False | Predicted: True |
|---|---|---|
| Real: False | 30 | 5 |
| Real: True | 20 | 245 |

Table 2.1: Confusion Matrix

From table 2.1 we can derive that the classifier predicted True 250 times, but it was only True in 245 of the cases. The other 5 times it was actually False, though it predicted False only 50 times. The times the classifier predicted True and it actually was True is called a True Positive. When it predicted True and it actually was False it is called a False positive. If it had predicted False and it actually is False it is called a True negative, and when it predicts False and it actually is True it is called a False Negative.
These terms can be used to calculate metrics that provide a good information basis for evaluating a model.

### 2.6.3 Recall

Recall, otherwise known as sensitivity or the true positive rate, is the share of a class that was actually labeled as that class. It's calculated by $\frac{tp}{tp+fn}$ where tp is True Positive and fn is False negative.
Using table 2.1 we can measure the recall to be $\frac{245}{245+20} \approx 0.925$.
Recall is great for measuring how good a classifier is at recognizing a class and a high recall, such as in the example above, which means that it is great at maximizing the true positive rate, but that it still can provide false positives. Using both the precision and recall is a good method for ensuring high true positive rate and a low false positive rate.

### 2.6.4 Specificity

The specificity, or the true negative rate, is the share of negatives that were actually labeled as such. It can be calculated by $\frac{tn}{tn+fp}$ where tn is true negative and fp is false positive. In the example above the specificity can be calculated to be $\frac{30}{30+5} \approx 0.857$.
When the specificity is high, such as in the given example, the classifier is not likely to register a positive as anything else.

A visualization of the recall and specificity can be seen in 2.4.

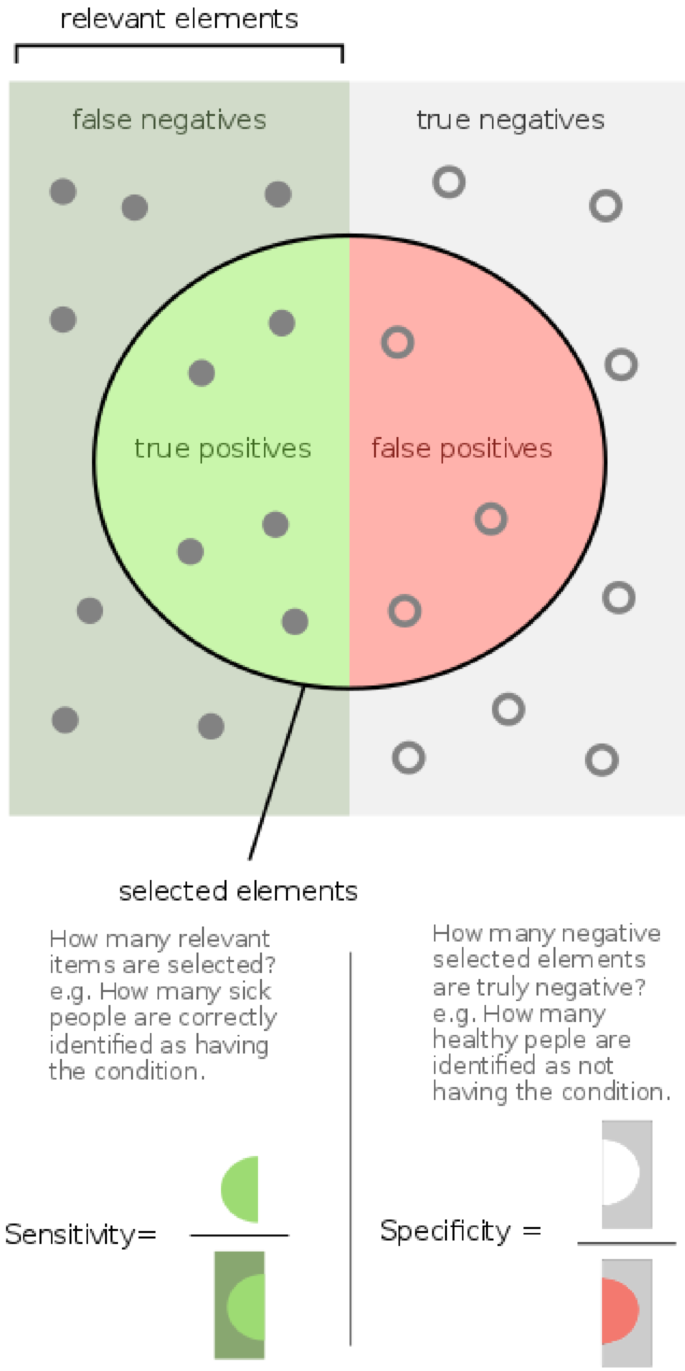
relevant elements
false negatives
true negatives
true positives
false positives
selected elements
How many relevant items are selected? e.g. How many sick people are correctly identified as having the condition.
How many negative selected elements are truly negative? e.g. How many healthy peple are identified as not having the condition.
Sensitivity=
Specificity =

Figure 2.4: Illustration of recall and specificity from [48]

### 2.6.5 Precision

Precision represents how many of the classified instances that actually are relevant. It can be calculated by the following formula: $\frac{tp}{tp+fp}$ where tp is True Positive and fp is False Positive.

Using table 2.1 we can calculate the precision to be $\frac{245}{245+5} = 0.98$ which is very good and means that the classifier did few mistakes when classifying the instances.
Precision is a good metric to use to get an impression of how well a classifier is correctly labeling a class and while a high precision shows that the classifier does not provide many false positives, it does not necessarily give a high true positive rate.

### 2.6.6 F1-score

The F1-score is a score computed from both the precision and the recall. An F1-score of 1 signifies a perfect precision and recall, while a score of 0 signifies the opposite. As stated in the previous section it's good to use both the precision and recall to get the best classifier possible and the F1-score is a nice way to get a view on the performance by looking at just one number. It can be computed by the following equation: $2 \cdot \frac{precision \cdot recall}{precision + recall}$.

### 2.6.7 Matthew's correlation coefficient

The Matthew's correlation coefficient is used to measure the performance of binary classifications. It makes use of all the content of the confusion matrix and takes class balance into account. It can thus be used as a reliable metric in unbalanced datasets, as opposed to for example accuracy. The formula to calculate the MCC from a confusion matrix is given by $MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$.
The formula returns a value between -1 and 1 where -1 indicates that the predictions are completely wrong, 0 indicates random predictions and 1 is perfect. In order to extend it to a multinomial classification problem this project made use of the one-vs-rest strategy.

### 2.6.8 AUC

Area under the curve(AUC), or AUROC(Area Under the Receiver Operating Characteristic curve), is a metric used to estimate the probability that a classifier will correctly predict a random sample. It is computed from calculating the area under the receiver operating characteristic curve, which is created by combining the false positive rate and true positive rate into one single metric and plotting it on a single graph as can be seen in figure 2.5. An AUC score of 1 indicates perfect predictions while 0.5 represents a random predictor.

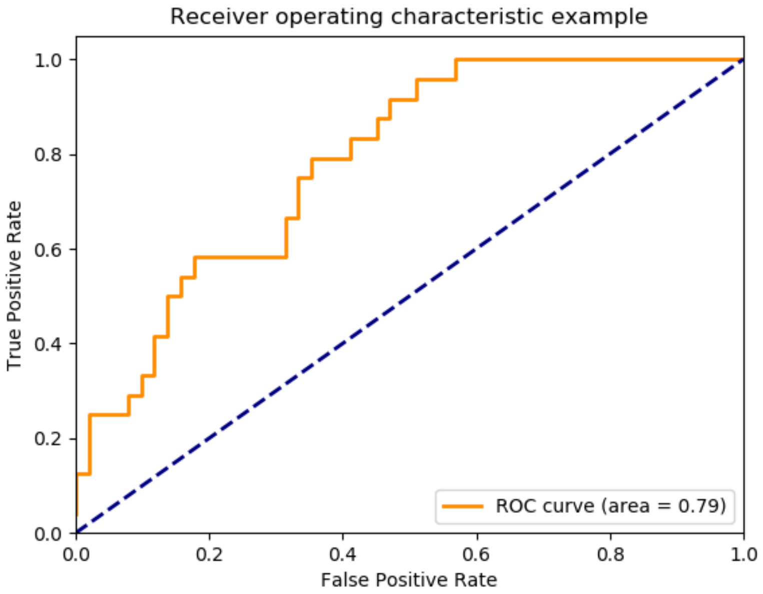

Figure 2.5: Illustration of a plotted ROC curve with the computed AUC from [46].

# Chapter 3

# Building and testing

This chapter describes how the experiments were implemented and tested, and which tools were implemented in the process.

## 3.1 Libraries and software

To ensure the least amount of bugs and best results, different software and libraries have been used throughout the project. The subsections in this section describes frameworks used for model selection, classifiers, feature extraction and so on.

### 3.1.1 scikit-learn

Scikit-learn [43] is a free, open source machine learning library for use in Python built on NumPy, SciPy and matplotlib. The library features methods for classification, regression, clustering, dimensionality reduction, model selection and preprocessing.

Scikit-learn was used for all the classifiers but the multilayer perceptron. It was also used for all cross validation, sampling and computation of metrics.

All classifiers were used with the scikit-learn default parameters except for the Support Vector Machine and the Random Forest classifier. Below you can find a summary of the most important parameters for each classifiers, both the ones that were kept using the default values and the ones that were changed.

- Support Vector Machine
  The SVM was implemented with a linear kernel, as opposed to the default rbf-kernel as it is thought to be faster.

- Random Forest
  The Random Forest classifier was trained with 100 trees instead of the default 10 as more trees is generally seen as better.
  Default parameters included using the Gini impurity to measure the quality of a split and the square root of the original number of features as the number of features to consider when looking for the best split.

- Decision Tree
  The Decision Tree classifier use the Gini impurity aswell to measure the quality of a split, but use the original number of features as the number of features to consider when looking for the best split.

- K Nearest Neighbors
  The K Nearest Neighbor classifier uses a default of 5 neighbors and minkowski with a power parameter of 2 as distance metric, which makes it equivalent of using eucledian distance.

- Gaussian Naive Bayes
  The Gaussian Naive Bayes classifier can be created with prior probabilities of the classes, but is not in this case.

- AdaBoost
  The AdaBoost classifier use the Decision Tree classifier as the base estimator. It use a default learning rate of 1 and stop at a maximum of 50 estimators.

- Bagging
  The Bagging classifier also uses the Decision Tree classifier as the base estimator with a default value of 10 base estimators. The classifier uses all the samples with all the features to train each base estimator.

- Dummy Classifier
  The dummy classifier is used as a simple baseline to compare with the other classifiers. It uses a "stratified" strategy when generating predictions, which means it makes predictions based on the training sets class distribution.

### 3.1.2 Keras

Keras [12] is a high-level deep learning library written in python. It is capable of running on top of TensorFlow, CNTK or Theano. In this project it was run on top of TensorFlow.
A Multilayer Perceptron was implemented using the sequential model seen in figure 3.1. This architecture was used throughout the project.
Dropout was used to prevent overfitting. Overfitting is a problem in deep neural networks [50], and by randomly dropping nodes and their connections from the network during training it may prevent the nodes from co-adapting too much.

The network was trained over 50 epochs with a batch size of 128. The loss function was binary crossentropy loss when doing binary classification. Sparse categorical crossentropy loss was used when doing multinomial classification. It was optimized using the adagrad optimizer [17].

### 3.1.3 Imbalanced learn

Imbalanced learn [32] is a python package built on scipy, numpy and scikit-learn that offers several sampling techniques used to balance out

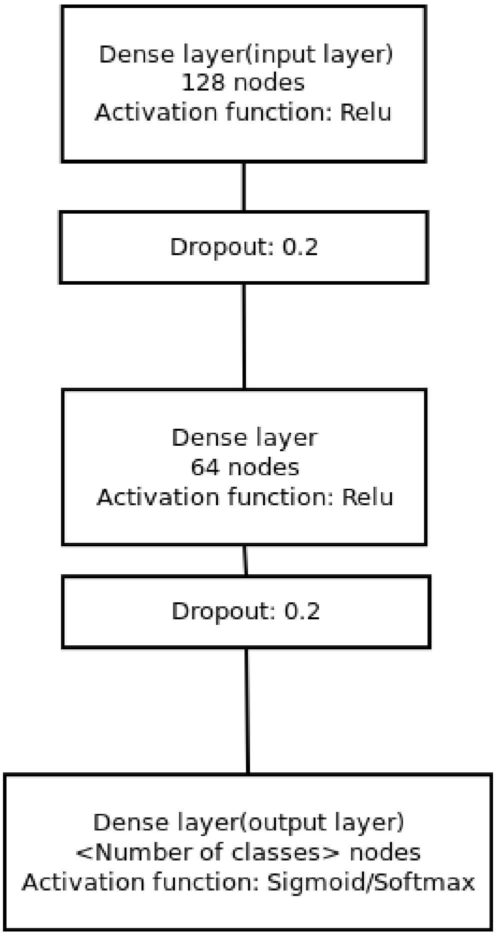
Dense layer(input layer)
128 nodes
Activation function: Relu
Dropout: 0.2
Dense layer
64 nodes
Activation function: Relu
Dropout: 0.2
Dense layer(output layer)
<Number of classes> nodes
Activation function: Sigmoid/Softmax

Figure 3.1: Illustration of the MLP network architecture. With a binary classification problem the number of output nodes are 1 and activation function is Sigmoid. With multinomial, the number of output nodes are equal to the number of classes and activation function is Softmax.

imbalanced data sets.
This project only made use of the SMOTE and RandomOversampling methods which respectively performed SMOTE and oversampling at random with replacement on the training data.

## 3.2 Model types

### 3.2.1 General model

The idea of the general model is that one can use data from all, or several, of the subjects that is available to create a general purpose model, that is it would work for all subjects. This model obviously has both advantages and disadvantages associated with it, and it might not be clear if the advantages beat the disadvantages or the other way around.

The most obvious problem with this model type is that some problems are very hard to generalize, for example the problem of mental health prediction. People are different and while one can say mental disorders affect people in similar fashion, people does not necessarily exhibit the same symptoms. Two people suffering from the same mental disorder, such as bipolar disorder, might react different to it and show different patterns both in speech and motor activity. However; given enough data from enough subjects the classifiers might still be able to generalize.

An advantage of this model that might justify the problem described above, is that there would not be any need for data collection before applying the model to a new subject. Especially within the domain of mental health prediction, where labeling the data would require the subject to meet regularly with a psychologist or psychiatrist, this is beneficial. A new subject could start using a classifier right away.

In this project the general model was created by training a classifier on all the subjects available. The model was evaluated using leave-one-subject-out cross validation as can be seen in figure 3.2.

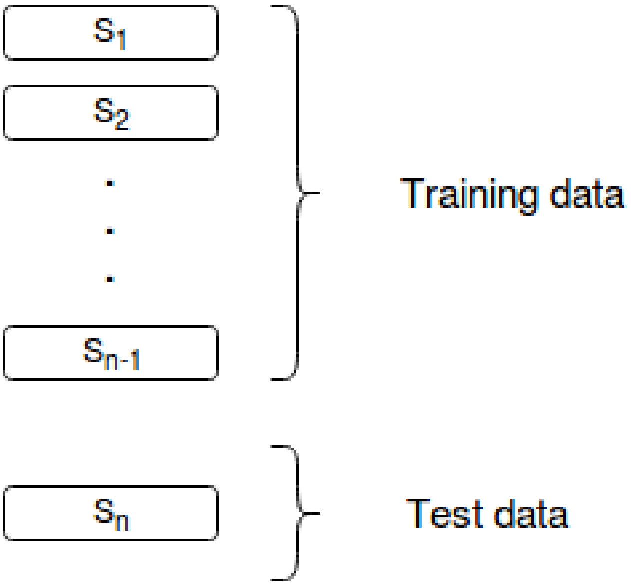

Figure 3.2: Illustration of the general model

### 3.2.2 Personalized model

The personalized model is expected to perform best of all the models as it is trained on data from one, and only one, subject. When trained on data from only one subject it does not have to generalize over several subjects and can focus on what defines the subject in question. Given enough data of a certain quality, it should be able to correctly classify samples with close to perfect accuracy.

This model is obviously better than the general when looking at performance alone, but there is one critical disadvantage. When a classifier is going to be trained on a new subject there would have to be quite a long period of data collection before the classifier could be trained. In the domain of mental health prediction it would mean a lot of meetings with professionals to help label the data collected, which is very resource consuming. This combined with the amount of time it takes before the subject could start taking advantage of the classifier makes it less of an obvious choice.

The personalized models in this project were evaluated using 10-fold cross validation.

### 3.2.3 Hybrid model

The general model and the personalized model both have their own advantages and disadvantages, and chosing the right one is not always easy. A possible solution is the hybrid model described in this section or the similar user adaptive model described in the next section.

The hybrid model is created very much in the same way as the general model with leave-one-subject-out cross validation, but a number of samples from the target, or test, subject is transferred to the training data. The goal is that the classifier learns the general features from the training dataset, but is slightly personalized by the data from the target subject.

To get a proper view of the performance of this model, compared to the generalized and personalized model, a number of different sample sizes are to be tested. As the amount of samples per subject is very different, a maximum sample size is set. The max is set to half the number of samples from the subject with the least amount of samples. A step size is set individual to each dataset. The experiment will train each classifier for each sample size from 0 to the max limit with the set step size.

A visualization of the hybrid model can be seen in figure 3.3. It is very similar to that of the general model, but subject n is split between the training data and test data.

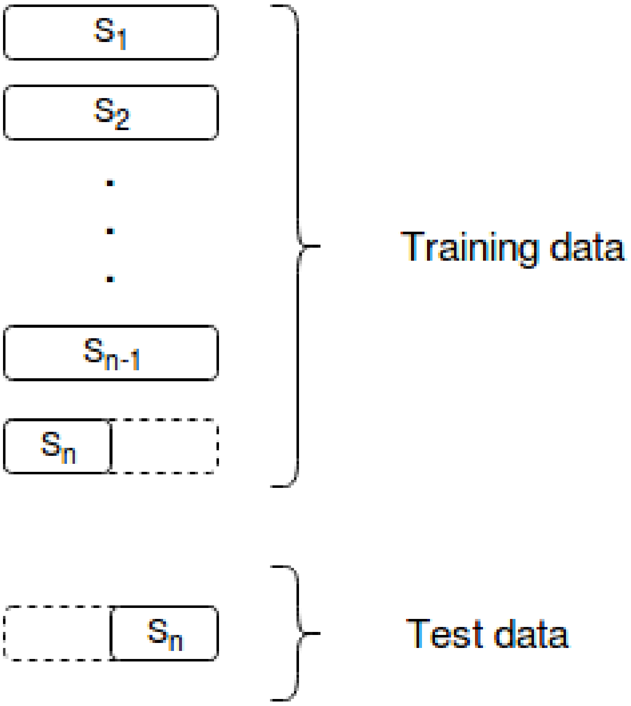

Figure 3.3: Illustration of the hybrid model

### 3.2.4 User adaptive model

A model very similar to the hybrid model described in the previous section, will also be tested. The basic idea of the model is very much alike, but the technique is different. Because of this, and for the sake of differentiating between the two hybrid models, the model will be referred to as a user adaptive model.

The user adaptive model makes use of transfer learning in a multilayer perceptron to adapt the general model to the target user. This way really utilizes the general model to extract the general features required for classification, and then tunes the classifier using samples from the target subject. This model is expected to boost the performance from *fewer* samples than the hybrid model. The model uses the same testing scheme for several sample sizes.

The model is built in the following way:

1. Create a network structure containing a feature layer and a classification layer.
2. Train a general model on the network
3. Freeze the feature layer
4. Retrain the classification layer on a set number of samples from the target user

A visualization can be seen in figure 3.4

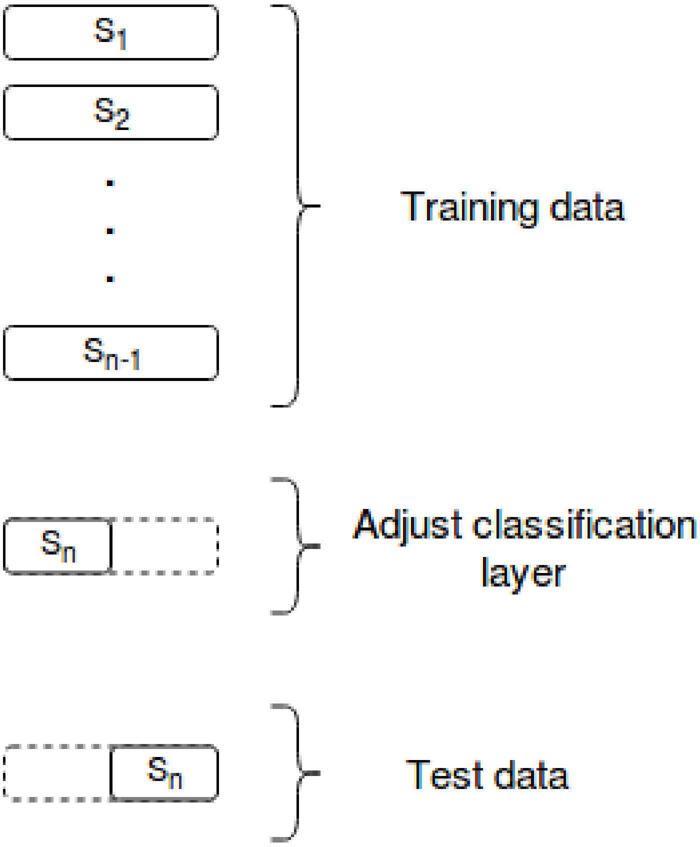

Figure 3.4: Illustration of the user adaptive model

# Chapter 4

# Experiments and results: Migraine attack detection

## 4.1 Subjects

This dataset consists of motor activity samples from 4 patients during a depressive period. The group of subjects consisted of 1 woman and 3 men where 2 of them were diagnosed with unipolar depression and the other two were diagnosed with type 2 bipolar disorder. The mean age of the subjects was 37.5 ± 2.5 years.

## 4.2 Recording of motor activity

Motor activity was collected during a continuous 2 week depressive period with migraine attacks during the period by an actigraph worn at the right arm. The actigraph was an AW4 from Cambridge Neurotechnology Ltd, England and uses a piezoelectric accelerometer to record the integration of intensity, amount and duration of movement in all directions. It records all movements over 0.05g at a sampling frequency of 32Hz. The recorded activity counts was summed up over 1 minute intervals and stored as such. The patients were asked to record all migraine attacks, when it started and how long it lasted, over the time they wore the actiwatch.

## 4.3 Feature Extraction

A total of 102787 samples were collected from all patients. Labels representing either an attack or non-attack were assigned to each sample. The samples were divided into segments where one segment represents a continuous series of samples from one subject of one class type. Every time the class label or subject changed the current segment ended and a new one started. This results in segments filled with samples from one subject with all samples labeled the same.

From these segments a total of 15 features were extracted from a sliding windows of size 100 with a step size of 20. The sliding window never overlapped between two segments to ensure no confusion as to what the label should be. These features are listed in table 4.1. A description of how some of the features, more specifically those the reader is not expected to know, are calculated follows.

| | |
|---|---|
| RMS | Mean squared difference |
| Curve length | Q25 |
| Mean | Q50 |
| Standard deviation | Q75 |
| Variance | Minute |
| Max | Hour |
| Min | Zero percentage |
| Sum | |

Table 4.1: Features extracted

**RMS** $\sqrt{\frac{1}{n}(x_1^2 + x_2^2 + ... + x_n^2)}$

**Curve Length** $\sum_{i=1}^{n} |x_{i-1} - x_i|$

**Mean Squared Difference** $\frac{1}{n}\sum_{i=1}^{n}(x_i - x_{i-1})^2$

**Zero percentage** The percentage of values in a segment that is equal to 0.

## 4.4 Class imbalance

Studying migraine attacks over a continuous 2 week period will result in a very imbalanced class distribution due to the nature of migraine attacks and the sampling. Migraine attacks do not necessarily occur very often or last very long resulting in more samples of no migraine attacks. When data is recorded continuously it will also result in measurements from when the patient is sleeping and there are no recorded migraine attacks and the motor activity often is 0. This increases the class imbalance even further and because there is such a large amount of samples with a motor activity of 0 it might reduce precision. The class distribution ended with 4728 occurences of no attacks and 349 occurences of attacks distributed across all subjects, as can be seen in figure 4.1. This is not a good distribution considering that this is gathered from 4 different patients that will have to be treated individually. These 4 patients might have different patterns of motor activity during migraine attacks and such a small and imbalanced dataset makes it hard to detect these patterns.

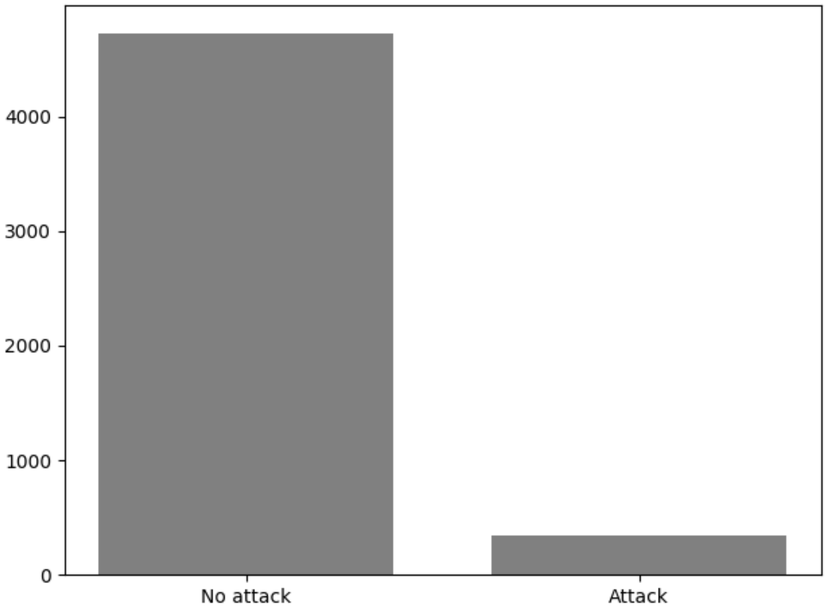

Figure 4.1: Class distribution

Just from looking at these numbers you can't really tell how evenly distributed across the different subjects the classes are, which will have an impact on the performance of the created models, especially for the hybrid and personalized model. The class distribution over the different subjects can be seen in table 4.2 and it's clear that subject sb has a better class distribution than the others. The very limited number of samples labeled as "Attack" might make it difficult to train classifiers that perform at an acceptable rate.

| Class | aasane_01_03 | aasane16 | aasane18 | sb |
|---|---|---|---|---|
| No attack | 1108 | 1070 | 1227 | 1323 |
| Attack | 43 | 23 | 59 | 224 |

Table 4.2: Class count in each user

### 4.4.1 Oversampling

To try to handle the class imbalance oversampling has been explored through Random Oversampling and SMOTE as described in sections 2.5.2 and 2.5.3.

This balanced the class distribution by increasing the number of attack occurences to 4728, and is visualized in figure 4.2. The oversampling is done in each fold during the cross validation. This way it will oversample according to the class distribution with each user, and it will only oversample the training data and not affect the test set. Doing it this way ensures as accurate metrics as possible when testing the performance. A classifier will be trained using no sampling, random oversampling and SMOTE so that the results can be compared. One classifier might react poorly to duplicate and similar data, while others might benefit from it. Which kind of sampling that has been used to provide the different results will be specified.

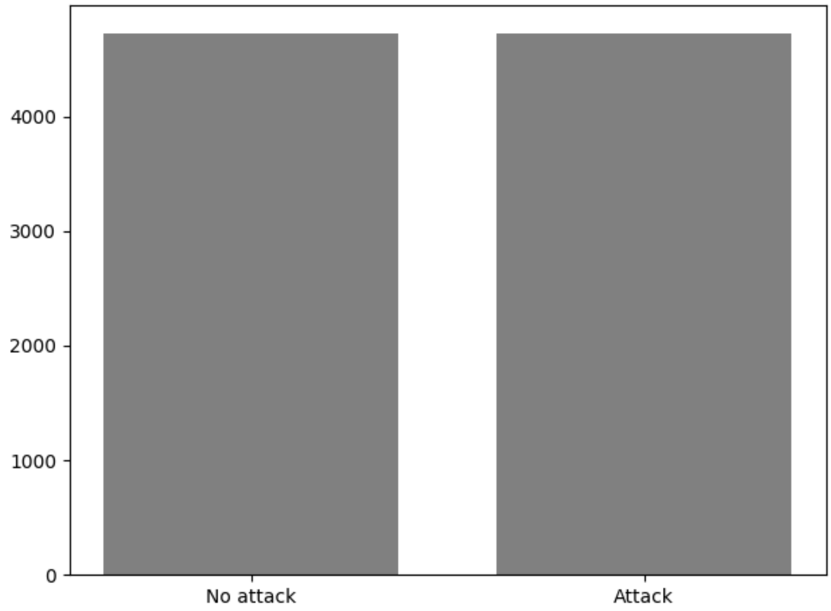

Figure 4.2: Class distribution after oversampling

## 4.5 Experiments description

**General model with no oversampling** Training and testing the classifiers using a general model approach. There is one fold for each subject, where each classifier is trained on all the other subjects and tested on the one left out. No oversampling is applied.

**Testing sample sizes for oversampling** When using oversampling it is useful to try out different goal sample sizes. This experiment trains classifiers using oversampling with the size of the attack class in the interval 30% of the size of the non-attack class to 100% of the size of the non-attack class. The experiment is conducted using a general model.

**General model with oversampling** The results from the classifiers trained using the general model with the best performing oversampling technique with the best goal sample size.

**Personalized model** Training and testing the classifiers using a personalized model approach. Classifiers are trained and tested using 10-fold cross validation for each subject. Trained both with and without oversampling. Oversampling uses the optimized parameters found above. There will also be trained a model using only 525 samples per subject.

**Hybrid model using chronologically sampled data** Training and testing classifiers using a hybrid model with a chronological sampling approach. Leave-one-user-out cross validation is applied for each subject. For each subject the first n samples are transferred to the training set in chronological order. This is done for n in the interval 0, which will result in a general model, to a max size of half of the size of the subject with the least amount of samples. Trained both with and without oversampling

**Hybrid model using stratified sampled data** Training a hybrid model by transferring stratified sampled data, which means that data samples were not necessarily picked in chronological order, but were rather picked at random in a way that maintains the class balance. Trained both with and without oversampling

**User adaptive model** Training a user adaptive model by using transfer learning techniques. One model was trained for each sample size in the same interval that was used in the other hybrid models. The samples used for the retraining of the classification layers were sampled using stratified sampling.

## 4.6 Results

Results have been collected by training classifiers using the algorithms described in section 2.3 and the model types described in section 3.2. To get an idea of how well the classifier performs compared to just classifying samples at random the classifiers will be compared to a "Dummy"-classifier [18]. The Dummy-classifiers use simple strategies for classifying samples and is designed for the purpose of comparing classifiers to it. It is not expected that the results will be state of the art because of the low number of samples and the imbalanced class set, especially without any type of oversampling. It is however expected that they will follow the expected pattern that the personalized model will outperform the general model and that the hybrid and user adaptive model will be close to the personalized one in terms of performance. Classifiers trained on specific subjects will probably perform better than others, for example will classifiers trained on subject *sb* and maybe *aasane18*, probably perform a lot better than the ones trained on *aasane16*.

### 4.6.1 General model

The general model performed very poor without any kind of oversampling, as can be seen in table 4.3. The SVM performed as bad as possible with a precision, recall and thus F1-score of 0, but that is not really surprising considering the use of a linear kernel. It has a MCC of 0.0 and an AUC of 0.5 which represents completely random predictions. The results really goes to show how important it is to look at several metrics when evaluating the performance of the classifier. The classifiers all have an exceptionally high accuracy, but both the precision and recall is poor and with MCC and AUC close to 0.0 and 0.5 respectively it indicates that the classifier performs not much better than random predictions.

The precision of the AdaBoost and Bagging classifiers is better than the other classifiers, but the recall is low resulting in a bad F1-score. This shows that it's will probably not label a sample as an attack when it is not, but it will probably not label an attack as one neither meaning that it's too strict. This is probably because of the low concentration of "Attack"-samples. The best F1 score belongs to the Gaussian Naive Bayes classifier which has a recall of 47.2% and a precision of 12.2% resulting in a F1-score of 18.1%, which again is seen in the MCC and AUC which indicates that it is slightly better than random.

| Algorithm | Accuracy(std) | Precision(std) | Recall(std) | F1(std) |
|---|---|---|---|---|
| SVM | 0.938(0.056) | 0.000(0.000) | 0.000(0.000) | 0.000(0.000) |
| Random forest | 0.929(0.057) | 0.213(0.184) | 0.074(0.079) | 0.105(0.111) |
| Decision tree | 0.888(0.042) | 0.125(0.135) | 0.141(0.097) | 0.115(0.089) |
| K-nn | 0.915(0.045) | 0.148(0.157) | 0.072(0.077) | 0.076(0.068) |
| **Gaussian NB** | **0.740(0.094)** | **0.122(0.134)** | **0.472(0.201)** | **0.181(0.173)** |
| AdaBoost | 0.933(0.059) | 0.273(0.189) | 0.055(0.044) | 0.091(0.071) |
| Bagging | 0.920(0.048) | 0.201(0.211) | 0.098(0.090) | 0.115(0.092) |
| MLP | 0.933(0.058) | 0.107(0.124) | 0.026(0.040) | 0.039(0.058) |
| Dummy | 0.878(0.031) | 0.060(0.076) | 0.045(0.005) | 0.038(0.020) |

| Algorithm | Specificity(std) | MCC(std) | AUC(std) |
|---|---|---|---|
| SVM | 1.000(0.000) | 0.000(0.000) | 0.500(0.000) |
| Random forest | 0.987(0.008) | 0.095(0.126) | 0.531(0.040) |
| Decision tree | 0.937(0.006) | 0.068(0.081) | 0.539(0.048) |
| K-nn | 0.972(0.014) | 0.054(0.063) | 0.522(0.033) |
| **Gaussian NB** | **0.751(0.102)** | **0.132(0.137)** | **0.611(0.076)** |
| AdaBoost | 0.992(0.006) | 0.100(0.097) | 0.523(0.023) |
| Bagging | 0.976(0.010) | 0.095(0.106) | 0.537(0.047) |
| MLP | 0.993(0.007) | 0.030(0.055) | 0.509(0.017) |
| Dummy | 0.934(0.022) | -0.011(0.017) | 0.490(0.010) |

Table 4.3: Mean performance across users of the general model trained on data from the migraine data without any oversampling

The standard deviation is very high in all cases, except for the SVM, at almost the same level, or higher, as the metric it belongs too. This clearly shows that classifiers trained on some subjects perform better than others.

To get a better understanding of which classifiers perform well and which does not it could be smart to look at the results for the different classifiers trained. Looking at the precision and recall for each classifier in tables 4.4 and 4.5 respectively we can see that classifiers tested on especially subject *aasane01_03* perform a lot worse than classifiers tested on other subjects It was not expected that this would perform that much worse than the other classifiers, especially when you keep in mind that it has almost twice as many examples of attacks as *aasane16*. The reason for this might be that subject *aasane01_03* has other motor activity patterns than the rest of the subjects when suffering from a migraine attack. Looking at the Confusion Matrix for the AdaBoost classifier tested on subject *aasane01_03* in table 4.6 we can see that while it predicted a total of 10 attacks, only 4 of them were actually attacks resulting in the given precision of 40%. It is also important to keep in mind that there were actually a total of 43 attacks to be recognized and that it failed to recognize 39 out of the 43. It does not really matter that it has a good precision when the recall is so low.

The Gaussian Naive Bayes classifier, which had the best F1-score of the general models, gave the best results when tested on subject *sb*. This may be accredited to the amount of attack samples it has available for testing. It might also have a motor activity pattern that is easier to recognize and correctly classify. Looking at its confusion matrix in table 4.7 we can see that the performance from this classifier is a lot better than what we have seen previously, but it is still far from optimal. The amount of False positives is more than twice the number of True positives which is far from ideal. On the other hand its recall is quite good at 68.8%.

Most of the classifiers perform better than the dummy-classifier, which is good. Whether the SVM did better than the dummy classifier is not clear. While the SVM had a precision and recall of 0, and the Dummy classifier had both at $\approx$ 5%, it had a lower specificity and both MCC and AUC indicated that it is worse than random. This could mean that the tradeoff between an increase in precision and recall and a decrease in specificity might not be worth in, but that is subjective. This helps to show that this is quite a difficult dataset to make predictions on.

| Algorithm | aasane01_03 | aasane16 | aasane18 | sb |
|---|---|---|---|---|
| SVM | 0.000 | 0.000 | 0.000 | 0.000 |
| Random forest | 0.000 | 0.444 | 0.171 | 0.238 |
| Decision tree | 0.000 | 0.071 | 0.112 | 0.316 |
| K-nn | 0.029 | 0.034 | 0.164 | 0.364 |
| Gaussian NB | 0.064 | 0.034 | 0.068 | 0.322 |
| AdaBoost | 0.400 | 0.400 | 0.000 | 0.290 |
| Bagging | 0.000 | 0.217 | 0.098 | 0.488 |
| MLP | 0.000 | 0.000 | 0.217 | 0.211 |
| Dummy | 0.024 | 0.012 | 0.030 | 0.173 |

Table 4.4: Precision from testing the general model on each subject trained on data from the other subjects from the migraine data without any oversampling

| Algorithm | aasane01_03 | aasane16 | aasane18 | sb |
|---|---|---|---|---|
| SVM | 0.000 | 0.000 | 0.000 | 0.000 |
| Random forest | 0.000 | 0.174 | 0.102 | 0.022 |
| Decision tree | 0.000 | 0.217 | 0.186 | 0.161 |
| K-nn | 0.023 | 0.043 | 0.186 | 0.036 |
| Gaussian NB | 0.233 | 0.391 | 0.576 | 0.688 |
| AdaBoost | 0.093 | 0.087 | 0.000 | 0.040 |
| Bagging | 0.000 | 0.217 | 0.085 | 0.089 |
| MLP | 0.000 | 0.000 | 0.085 | 0.018 |
| Dummy | 0.047 | 0.043 | 0.051 | 0.040 |

Table 4.5: Recall from testing the general model on each subject trained on data from the other subjects from the migraine data without any oversampling

| | Predicted negative | Predicted positive |
|---|---|---|
| True negative | 1102 | 6 |
| True positive | 39 | 4 |

Table 4.6: Confusion Matrix from testing the general model AdaBoost classifier on subject *aasane01_03* trained on data from the other subjects from the migraine data without any oversampling

| | Predicted negative | Predicted positive |
|---|---|---|
| True negative | 998 | 325 |
| True positive | 70 | 154 |

Table 4.7: Confusion Matrix from testing the general model Gaussian NB classifier on subject *sb* trained on data from the other subjects from the migraine data without any oversampling

When dealing with scores as low as the above it might be an idea to look at the scores gained from the training data. Looking at table 4.8 we can see that the classifiers seem to overfit in many cases. For example both the Random Forest and Decision Tree have perfect performance when tested on the training data, but quite poor performance on the test data. This might come from two things: the low amount of samples or that the classifiers are trained on default parameters without any tuning, but the latter should not matter as much as it does.

| Algorithm | Accuracy(std) | Precision(std) | Recall(std) | F1(std) |
|---|---|---|---|---|
| SVM | 0.932(0.022) | 0.000(0.000) | 0.000(0.000) | 0.000(0.000) |
| **Random forest** | **1.000(0.000)** | **1.000(0.000)** | **1.000(0.000)** | **1.000(0.000)** |
| **Decision tree** | **1.000(0.000)** | **1.000(0.000)** | **1.000(0.000)** | **1.000(0.000)** |
| K-nn | 0.965(0.010) | 0.903(0.022) | 0.538(0.060) | 0.672(0.043) |
| Gaussian NB | 0.741(0.031) | 0.146(0.059) | 0.572(0.121) | 0.231(0.086) |
| AdaBoost | 0.940(0.022) | 0.762(0.082) | 0.187(0.061) | 0.299(0.084) |
| Bagging | 0.995(0.002) | 0.999(0.002) | 0.922(0.016) | 0.959(0.009) |
| MLP | 0.934(0.021) | 0.613(0.033) | 0.081(0.066) | 0.136(0.097) |
| Dummy | 0.877(0.036) | 0.075(0.028) | 0.072(0.025) | 0.073(0.026) |

| Algorithm | Specificity(std) | MCC(std) | AUC(std) |
|---|---|---|---|
| SVM | 1.000(0.000) | 0.000(0.000) | 0.500(0.000) |
| **Random forest** | **1.000(0.000)** | **1.000(0.000)** | **1.000(0.000)** |
| **Decision tree** | **1.000(0.000)** | **1.000(0.000)** | **1.000(0.000)** |
| K-nn | 0.995(0.002) | 0.680(0.028) | 0.767(0.029) |
| Gaussian NB | 0.750(0.036) | 0.186(0.077) | 0.661(0.053) |
| AdaBoost | 0.996(0.002) | 0.358(0.085) | 0.592(0.031) |
| Bagging | 1.000(0.000) | 0.957(0.008) | 0.961(0.008) |
| MLP | 0.996(0.004) | 0.194(0.077) | 0.538(0.031) |
| Dummy | 0.935(0.019) | 0.007(0.015) | 0.504(0.007) |

Table 4.8: Mean performance across users of the general model when the classifiers are trained and tested on the training data from the migraine dataset without any oversampling

**General model with oversampling**

To try to mitigate class imbalance a new approach was explored. The same classifiers were trained using the same leave-one-user-out cross validation, but this time using random oversampling or SMOTE on the training data within each fold. To get the best results, several different strategies were tried in terms of how many samples one should aim for when oversampling. The maximum number of attack samples resulted in the same amount of attack samples as non-attack samples after oversampling. The minimum number of attack samples was 30% of the number of non-attack samples.
This can be depicted in figure 4.3 and 4.4 showing Random Oversampling and SMOTE respectively. The figures show that there is not much difference in most classifiers, except for with the SVM which shows a significant improvement. Sampling to the maximum size seems to be the best choice and will be used for the remainder of the project. SMOTE seems to be performing a bit better than Random Oversampling, possibly due to the way it collects new samples, and will thus be the designated choice for the rest of the project. A general model will be trained once using no oversampling and once using SMOTE for each classifier.

Looking at table 4.9 we can see that the oversampling slightly increased performance in a lot of cases and for the SVM the performance increased a lot. It has a precision that is well within the average precision of the

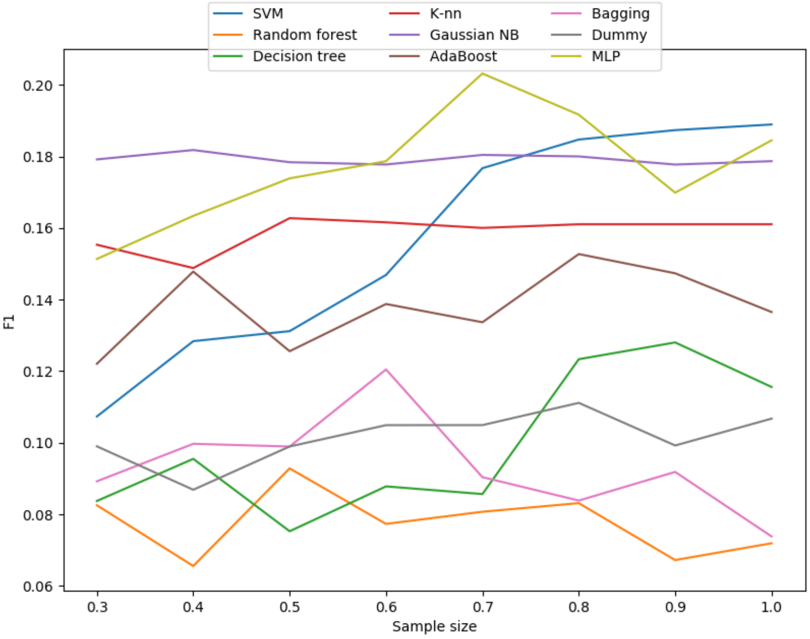

Figure 4.3: F1 score from general models trained on migraine dataset using Random Oversampling with various sample sizes. Sample size represents the amount of attack-samples as a percentage of non-attack samples.

classifiers and has by far the best recall giving it the best F1-score of the classifiers.

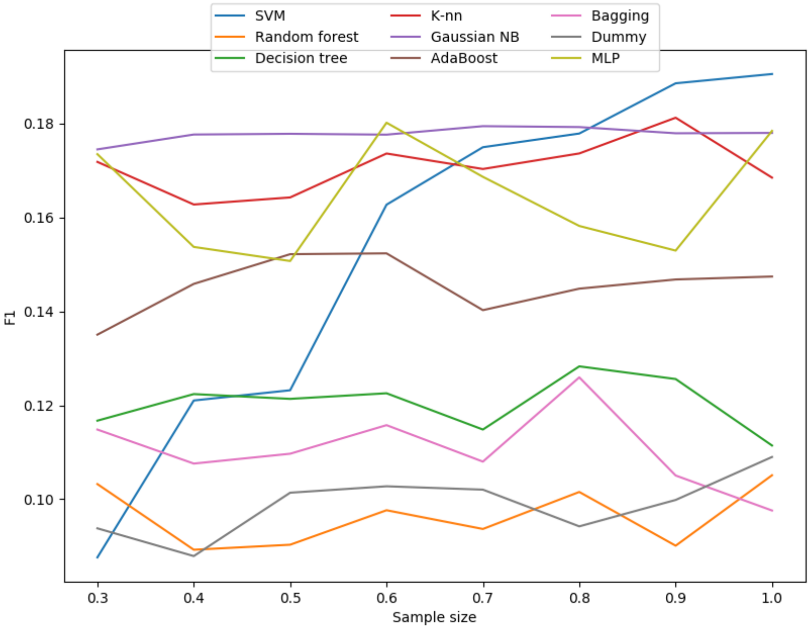

Figure 4.4: F1 score from general models trained on migraine dataset using SMOTE with various sample sizes. Sample size represents the amount of attack-samples as a percentage of non-attack samples.

### 4.6.2 Personalized model

The personalized model will probably perform somewhat better than the generalized model, but it's important to keep in mind that there is one classifier for each subject that is only trained and tested on data from that subject alone. In this case that might reduce the performance quite a lot because of the limited number of samples, especially with the subjects that have very few examples of attacks such as *aasane16*.

The results in table 4.10 shows that while it definitely did increase performance in some cases, a lot of the classifiers have the same performance.Especially the Decision Tree, but also the Bagging classifier seem to have benefited from this model type. It is still important to keep in mind that these classifiers were trained on roughly less than a third of the data that the general model was trained on and that when trained on more data it probably would have better performance.

Again we see that the standard deviation is quite high, which again makes sense considering the class distribution seen earlier in table 4.2. To get a better view of what it would look like in a slightly better dataset it might be beneficial to look at the classifiers trained on subject *sb* as that subject has the most data of the group. Table 4.11 shows that the personalized model does better on a more balanced dataset. Again the Gaussian NB stands out with a F1-score of 0,492. It does not have the best precision of the lot, but it has by far the best recall with a recall of 76%.

| Algorithm | Accuracy(std) | Precision(std) | Recall(std) | F1(std) |
|---|---|---|---|---|
| **SVM** | **0.573(0.112)** | **0.123(0.130)** | **0.815(0.107)** | **0.190(0.166)** |
| Random forest | 0.879(0.039) | 0.110(0.118) | 0.140(0.131) | 0.093(0.070) |
| Decision tree | 0.866(0.036) | 0.127(0.120) | 0.237(0.205) | 0.138(0.108) |
| K-nn | 0.820(0.015) | 0.146(0.153) | 0.341(0.127) | 0.172(0.121) |
| Gaussian NB | 0.662(0.078) | 0.112(0.122) | 0.597(0.256) | 0.178(0.175) |
| AdaBoost | 0.727(0.047) | 0.101(0.095) | 0.394(0.271) | 0.132(0.092) |
| Bagging | 0.884(0.044) | 0.112(0.117) | 0.156(0.151) | 0.102(0.085) |
| MLP | 0.730(0.080) | 0.125(0.136) | 0.418(0.061) | 0.158(0.122) |
| Dummy | 0.497(0.005) | 0.062(0.054) | 0.502(0.025) | 0.104(0.079) |

| Algorithm | Specificity(std) | MCC(std) | AUC(std) |
|---|---|---|---|
| **SVM** | 0.565(0.122) | 0.181(0.106) | 0.690(0.041) |
| Random forest | 0.930(0.034) | 0.053(0.072) | 0.535(0.053) |
| Decision tree | 0.910(0.029) | 0.096(0.111) | 0.573(0.090) |
| K-nn | 0.853(0.042) | 0.127(0.094) | 0.597(0.057) |
| Gaussian NB | 0.657(0.088) | 0.141(0.152) | 0.627(0.099) |
| AdaBoost | 0.757(0.088) | 0.081(0.102) | 0.575(0.116) |
| Bagging | 0.934(0.034) | 0.064(0.081) | 0.545(0.059) |
| MLP | 0.757(0.108) | 0.107(0.081) | 0.588(0.031) |
| Dummy | 0.497(0.006) | -0.001(0.009) | 0.499(0.010) |

Table 4.9: Mean performance across users of the general model trained on data from the migraine data with SMOTE

We can see that a lot of the classifiers have a recall around 30%-40% with a standard deviation of 10%-20% and a precision in the range 40%-50%, but with a much higher standard deviation in the 25%-30% range. This might be due to the two issues briefly discussed earlier: the low number of samples, especially attack samples, or that the motor activity pattern is hard to classify.

| Algorithm | Accuracy(std) | Precision(std) | Recall(std) | F1(std) |
|---|---:|---:|---:|---:|
| SVM | 0.938(0.056) | 0.000(0.000) | 0.000(0.000) | 0.000(0.000) |
| Random forest | 0.930(0.064) | 0.270(0.305) | 0.166(0.180) | 0.189(0.210) |
| **Decision tree** | **0.908(0.062)** | **0.382(0.217)** | **0.372(0.287)** | **0.330(0.211)** |
| K-nn | 0.922(0.060) | 0.234(0.252) | 0.169(0.184) | 0.181(0.196) |
| Gaussian NB | 0.712(0.110) | 0.230(0.126) | 0.592(0.339) | 0.237(0.171) |
| AdaBoost | 0.915(0.068) | 0.352(0.317) | 0.303(0.263) | 0.289(0.267) |
| Bagging | 0.922(0.057) | 0.321(0.301) | 0.270(0.248) | 0.258(0.235) |
| MLP | 0.930(0.068) | 0.162(0.253) | 0.096(0.121) | 0.109(0.149) |
| Dummy | 0.887(0.091) | 0.042(0.064) | 0.044(0.061) | 0.042(0.062) |

| Algorithm | Specificity(std) | MCC(std) | AUC(std) |
|---|---:|---:|---:|
| SVM | 1.000(0.000) | 0.000(0.000) | 0.500(0.000) |
| Random forest | 0.975(0.034) | 0.175(0.208) | 0.570(0.082) |
| **Decision tree** | 0.943(0.038) | 0.306(0.236) | 0.657(0.146) |
| K-nn | 0.965(0.034) | 0.155(0.186) | 0.567(0.083) |
| Gaussian NB | 0.717(0.124) | 0.211(0.137) | 0.654(0.113) |
| AdaBoost | 0.956(0.036) | 0.267(0.285) | 0.629(0.137) |
| Bagging | 0.965(0.027) | 0.243(0.246) | 0.617(0.121) |
| MLP | 0.979(0.035) | 0.094(0.127) | 0.538(0.045) |
| Dummy | 0.938(0.053) | -0.019(0.019) | 0.491(0.010) |

Table 4.10: Mean performance across users of the personalized model trained on data from the migraine data without any oversampling

**Personalized model with oversampling**

Oversampling had seemingly better effect on the personalized model than on the general. The F1-score have gone up significantly for all classifiers and especially the MLP and SVM benefited a lot from it, as can be seen in table 4.12. There has been a nice increase in both precision and recall without too much of an increase in standard deviation. The MCC and AUC also indicates significantly better performance with the AUC now being in the 0.65-0.75 range for most classifiers, as opposed to 0.55-0.65 as it was without any oversampling. The reason why the personalized model benefited better from the oversampling than the general model might be the low amount of training data compared to the general model. It might also be because similar data is not too much of a big deal when it comes from the same subject. The similar data might help highlight patterns in the data, which makes classification easier.

| Algorithm | Accuracy(std) | Precision(std) | Recall(std) | F1(std) |
|---|---|---|---|---|
| SVM | 0.855(0.002) | 0.000(0.000) | 0.000(0.000) | 0.000(0.000) |
| Random forest | 0.837(0.064) | 0.548(0.306) | 0.326(0.123) | 0.379(0.146) |
| Decision tree | 0.821(0.064) | 0.473(0.245) | 0.424(0.136) | 0.414(0.128) |
| K-nn | 0.836(0.062) | 0.531(0.273) | 0.366(0.103) | 0.405(0.128) |
| **Gaussian NB** | **0.715(0.159)** | **0.417(0.302)** | **0.760(0.166)** | **0.492(0.227)** |
| AdaBoost | 0.818(0.075) | 0.409(0.313) | 0.299(0.185) | 0.326(0.210) |
| Bagging | 0.839(0.054) | 0.492(0.283) | 0.326(0.161) | 0.369(0.171) |
| MLP | 0.830(0.077) | 0.535(0.322) | 0.265(0.147) | 0.322(0.177) |
| Dummy | 0.754(0.027) | 0.136(0.076) | 0.129(0.076) | 0.131(0.072) |

| Algorithm | Specificity(std) | MCC(std) | AUC(std) |
|---|---|---|---|
| SVM | 1.000(0.000) | 0.000(0.000) | 0.500(0.000) |
| Random forest | 0.924(0.071) | 0.321(0.199) | 0.625(0.072) |
| Decision tree | 0.887(0.078) | 0.332(0.168) | 0.656(0.067) |
| K-nn | 0.915(0.070) | 0.337(0.173) | 0.641(0.061) |
| **Gaussian NB** | **0.708(0.181)** | **0.402(0.283)** | **0.734(0.129)** |
| AdaBoost | 0.906(0.077) | 0.241(0.254) | 0.603(0.106) |
| Bagging | 0.925(0.058) | 0.303(0.209) | 0.626(0.084) |
| MLP | 0.926(0.082) | 0.271(0.223) | 0.596(0.088) |
| Dummy | 0.859(0.029) | -0.011(0.084) | 0.494(0.041) |

Table 4.11: Mean across folds of classifiers trained on subject *sb* of the personalized model trained on data from the migraine data without any oversampling

**Personalized model from 525 samples**

When comparing personalized and hybrid models it is important to keep in mind what would happen if you trained a personalized model using the same amount of data that you take from the target subject when training the hybrid model. If such a personalized model perform better than the hybrid model there will be no point in creating a hybrid model as one could just create a personalized model instead. Table 4.13 shows the results from creating such a model. It only shows the models where SMOTE is used as they are deemed to be the most relevant as they have the best results. We can see that it performs worse than the personalized model trained on all the data, but it is still better than the general better.

### 4.6.3 Hybrid model

In section 3.2.3 we specified that the test set would be a fixed size of 50% of the size of the subject with the least amount of samples and that the number of samples transferred would be in the range of 0 to test size with a fixed step size. This means that in the case of this dataset the size of the test data will have a fixed size of 546 samples and the transferred data wil have a max size of 525. The step size is set to 25. This means that we will train a total of 1584 classifiers. There are 9 different classifiers that is trained once with oversampling and once without. This has to be done once for each

| Algorithm | Accuracy(std) | Precision(std) | Recall(std) | F1(std) | Specificity(std) | MCC(std) | AUC(std) |
|---|---|---|---|---|---|---|---|
| SVM | 0.761(0.097) | 0.275(0.098) | 0.790(0.096) | 0.326(0.134) | 0.753(0.106) | 0.337(0.107) | 0.772(0.080) |
| Random forest | 0.914(0.068) | 0.396(0.244) | 0.394(0.249) | 0.360(0.225) | 0.945(0.051) | 0.334(0.238) | 0.669(0.126) |
| Decision tree | 0.898(0.072) | 0.329(0.183) | 0.414(0.220) | 0.332(0.178) | 0.928(0.050) | 0.299(0.197) | 0.671(0.121) |
| K-nn | 0.882(0.070) | 0.328(0.163) | 0.603(0.172) | 0.397(0.173) | 0.897(0.063) | 0.376(0.171) | 0.750(0.098) |
| Gaussian NB | 0.645(0.075) | 0.198(0.150) | 0.735(0.190) | 0.229(0.177) | 0.640(0.079) | 0.218(0.133) | 0.687(0.069) |
| AdaBoost | 0.880(0.076) | 0.378(0.340) | 0.559(0.250) | 0.400(0.290) | 0.896(0.072) | 0.378(0.303) | 0.727(0.135) |
| Bagging | 0.903(0.067) | 0.373(0.239) | 0.411(0.238) | 0.352(0.208) | 0.932(0.050) | 0.322(0.228) | 0.672(0.123) |
| **MLP** | **0.872(0.085)** | **0.379(0.195)** | **0.663(0.141)** | **0.429(0.184)** | **0.881(0.085)** | **0.418(0.186)** | **0.772(0.090)** |
| Dummy | 0.503(0.017) | 0.066(0.061) | 0.509(0.040) | 0.110(0.089) | 0.503(0.018) | 0.010(0.020) | 0.506(0.022) |

Table 4.12: Mean performance across users of the personalized model trained on data from the migraine data with SMOTE

subject and then all of this has to be done once for each step in the interval 0 to 525 which is 22 steps bringing it to a total of $9 * 2 * 4 * 22 = 1584$.

As mentioned in section 3.2.3 the hybrid model will follow two different approaches; adding testing data to training data and transfer learning. When transfering data from the test set to the training set, there are several ways of doing it, and we will herein explore the results from implementing two of them.

### Hybrid model with chronological sample data

The first method is the simplest, most intuitive way of doing it. We simply slice a piece of the testing data and add it to the training data in chronological order. This can be viewed in pseudocode below.

```
#Size is the number of samples to be transferred
training_data = testing_data[0:size]
testing_data = testing_data[max_size:]
```

To get a better view of how the performance of each classifier changesdas the sample size increased, the results have been visualized in figure 4.5. This shows that the score fluctuates a lot and some classifiers like the Gaussian NB never experiences any improvement. For a lot of the classifiers the best results occur when the sample size is 425. This does however not seem to be a reliable method of creating a hybrid model, at

| Algorithm | Accuracy(std) | Precision(std) | Recall(std) | F1(std) |
|---|---|---|---|---|
| SVM | 0.779(0.112) | 0.245(0.053) | 0.690(0.136) | 0.307(0.095) |
| Random forest | 0.887(0.107) | 0.264(0.158) | 0.342(0.182) | 0.266(0.137) |
| Decision tree | 0.880(0.107) | 0.270(0.096) | 0.361(0.198) | 0.280(0.120) |
| K-nn | 0.854(0.117) | 0.297(0.060) | 0.570(0.209) | 0.349(0.083) |
| Gaussian NB | 0.760(0.066) | 0.214(0.165) | 0.542(0.272) | 0.244(0.193) |
| AdaBoost | 0.879(0.099) | 0.279(0.175) | 0.436(0.144) | 0.306(0.143) |
| Bagging | 0.877(0.111) | 0.257(0.114) | 0.354(0.166) | 0.267(0.104) |
| **MLP** | **0.863(0.110)** | **0.311(0.063)** | **0.603(0.146)** | **0.362(0.057)** |
| Dummy | 0.494(0.013) | 0.061(0.054) | 0.493(0.072) | 0.103(0.079) |

| Algorithm | Specificity(std) | MCC(std) | AUC(std) |
|---|---|---|---|
| SVM | 0.776(0.125) | 0.299(0.054) | 0.733(0.017) |
| Random forest | 0.912(0.106) | 0.228(0.123) | 0.627(0.052) |
| Decision tree | 0.904(0.103) | 0.236(0.114) | 0.633(0.074) |
| K-nn | 0.865(0.117) | 0.321(0.112) | 0.718(0.101) |
| Gaussian NB | 0.765(0.070) | 0.213(0.185) | 0.653(0.125) |
| AdaBoost | 0.901(0.093) | 0.272(0.143) | 0.669(0.061) |
| Bagging | 0.901(0.107) | 0.223(0.094) | 0.628(0.053) |
| **MLP** | **0.871(0.120)** | **0.347(0.039)** | **0.737(0.038)** |
| Dummy | 0.495(0.014) | -0.004(0.030) | 0.494(0.039) |

Table 4.13: Mean performance across users of the personalized model trained on 525 samples per subject from the migraine data with SMOTE

least for this dataset. We can see that many classifiers follow the same pattern for their increase and decrease in performance and this might be due to the method for tansferring data. When data is transferred in a chronological order, and especially in such an imbalanced dataset as this, it will gain more and more samples of one class before it suddenly gets a burst of another class and then goes back to the first class again. This might lead to what we see in figure 4.5 where the score fluctuates a lot. When the score suddenly rises, such as what we see at $x = 425$ it has probably been exposed to more examples of attacks. It might also contain data that is more specific to the subject so that it helps the classifier correctly separate between the two classes.

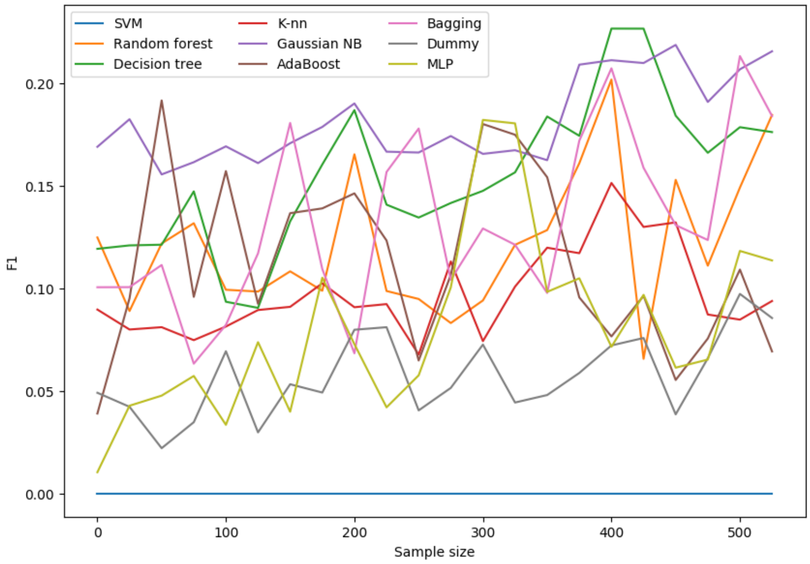

Figure 4.5: F1 score from the hybrid model using chronological sampling trained on migraine dataset with no oversampling

The results from using the same approach but applying SMOTE to the training data fluctuates even more than the previous approach, but over a smaller interval as can be seen in figure 4.6. Here we can see that after only 25 samples there is a significant increase in performance with almost all classifiers. What is interesting here is that towards the end, after approximately $x = 350$, all the classifiers seem to follow the same pattern and converge towards the same score.

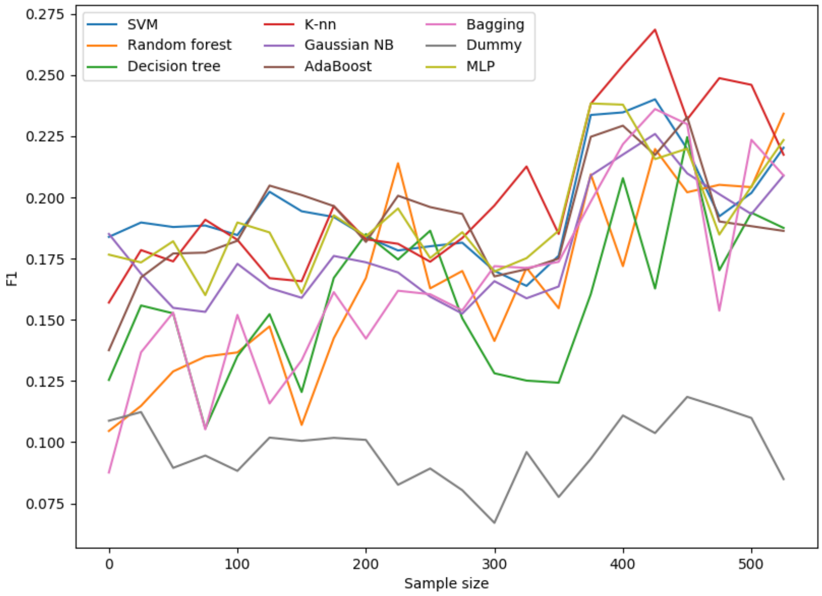

Figure 4.6: F1 score from the hybrid model using chronological sampling trained on migraine dataset with SMOTE

**Hybrid model with stratified sample data.**

The other approach when adding testing data to the training data is to do stratified sampling instead of sampling the data in chronological order. This aims to preserve the class balance and should effectively remove the issue with chronological sampling, where some sample sizes have a better class rate than others. It will also ensure that there will always be examples of both classes in the test set, an issue that was quite frequent when sampling data in a chronological order on this dataset. Because of this it is better to use stratified sampling for this research.

It is quite clear from figure 4.7 that using stratified sampling provided better results. While the results still fluctuates, they fluctuate over better scores. Especially classifiers like Random Forest, Decision Tree, Bagging and K-nn have improved performance. They seem to reach their peak when 400-500 samples have been transferred, but as this dataset is probably not representative this number might increase or decrease for better datasets. It is still clear that most classifiers benefit more from stratified sampling than chronological sampling.

When adding oversampling by SMOTE(figure 4.8) the trend become significantly clearer. All classifiers except for the SVM and Gaussian NB classifier have a clear boost in performance. Especially the random forest benefit great from this use of a hybrid model with a F1-score of 0.1 in the general model in this case and a peak performance with a F1-score of approximately 0.4 when new samples are added.

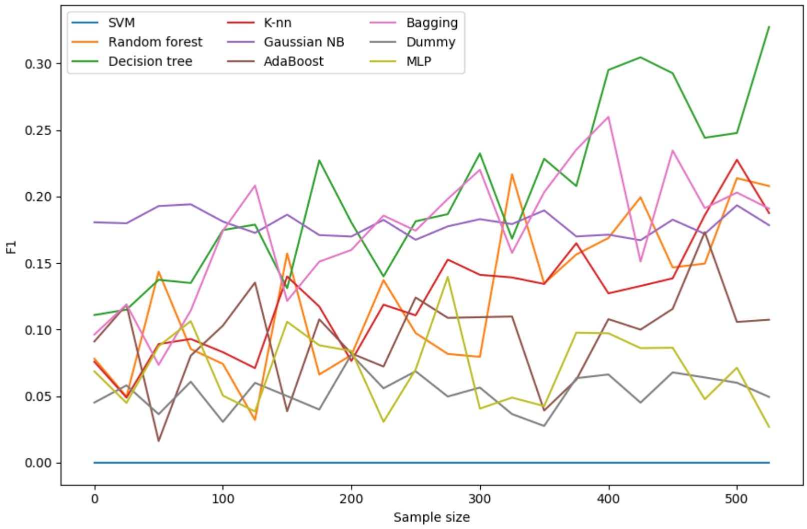

Figure 4.7: F1 score from the hybrid model using stratified sampling trained on migraine dataset with no oversampling

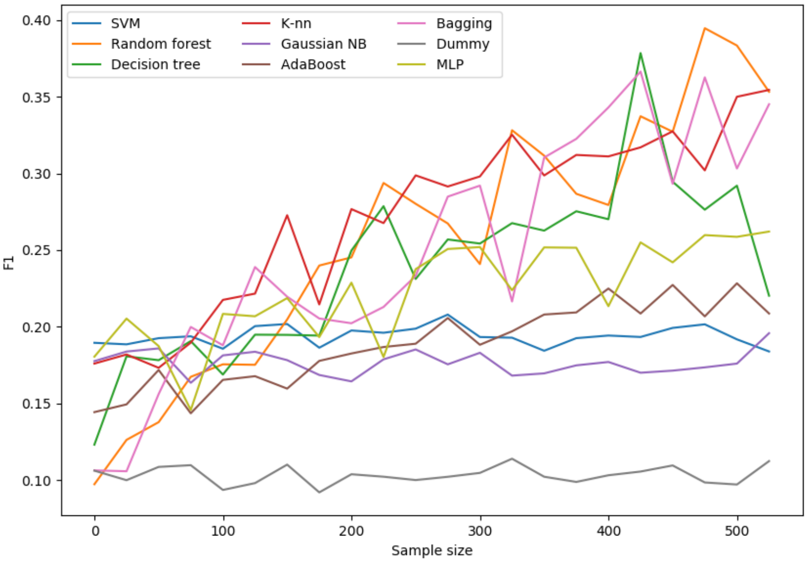

Figure 4.8: F1 score from the hybrid model using stratified sampling trained on migraine dataset with SMOTE

### 4.6.4 User adaptive model

Transfer learning might not require large amounts of data from the subject it is adapting to as it already has trained both feature extraction and classification on other subjects when training the general model. We are simply adjusting the classification layer(s) using data from just the one subject we are adapting the model for and might not need as much data as one would need for the other methods of hybrid learning.

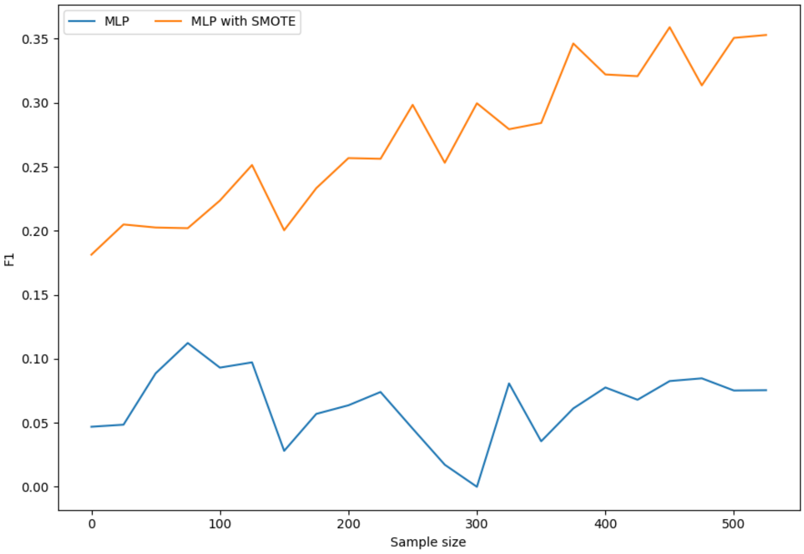

Figure 4.9: F1 score from the hybrid model using transfer learning trained on migraine dataset

In figure 4.9 we can see that transfer learning did not significantly improve the neural network without SMOTE. At its best there was only an improvement of approximately 0.05 to the F1 scores from $\approx 0.05$ to $\approx 0.1$. However when SMOTE was applied you can see improvement that was even better than the neural network using the hybrid model with stratified sampling instead of transfer learning and it's even competing with the best classifier at times which is the Random Forest classifier at $\approx 0.4$.

Figure 4.10 visualizes the relationship between the F1 scores of the best classifier in each model. The SVM had the best performance in the general model, the MLP in both the personalized models and the random forest was the best when using the hybrid model.
The hybrid model was slightly better than the user adaptive model, but both were worse than the personalized model. The user adaptive model was even a bit worse than the personalized model created from the same amount of data from the target subject.

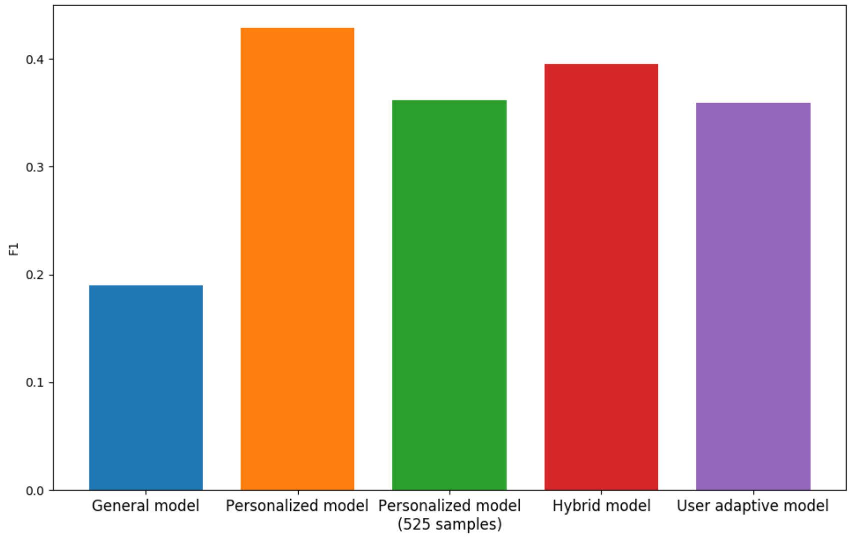

Figure 4.10: The best F1-score of the different models on the migraine dataset

## 4.7 Discussion

This is a tough dataset to work with, not only is it small and severely unbalanced, but it is also seemingly tough to predict migraine attacks from motor activity alone. People act very different during migraine attacks, and it is not granted that people will act different than normal during a migraine attack. Nevertheless, we were able to train models that in most cases outperformed the dummy classifier and that could predict a migraine attack to some extent.

The general model did very badly, but that came as no surprise. When no oversampling was applied most classifiers did not show much better performance than the dummy classifier, which picks out predictions almost by random. With oversampling it did improve somewhat, but not by a considerable amount. The best classifier, which was the SVM, had a F1-score that was only 0.08 higher than the dummy classifier.

The personalized model performed better than the general model, which was to be expected, but did not do as well as expected. Again this can be attributed to the small dataset, which was even smaller for the personalized model since it only used data from each subject. Still it should perform better as it only had to focus on the pattern for **that** subject and not be disturbed by patterns from other users that might behave completely different both during and outside of a migraine attack. The standard deviation of the personalized model was quite high, in one case twice the deviation of the general model, which might indicate what was stated earlier; that some people show less symptoms that indicate migraine attacks in their motor activity than others. Still it did outperform the general model and when applied with SMOTE it benefited from it, boosting the F1 score by a significant amount. The neural network went from an average F1-score of 0.109 to 0.429.

The hybrid model was expected to perform better than the general model, almost as good as the personalized model and while its performance does not quite reach that of the personalized model in most cases when using the approach with chronological sampling, it even outperforms the personalized model at times when using stratified sampling. The user adaptive model has a clear boost in performance when applied with SMOTE, but does not have quite the same improvement as some of the classifiers when using the traditional hybrid model approach, and it is no better than creating a personalized model on the same amount of data. This bides good news for the user of hybrid models prediction of mental health states and even though the dataset is not optimal and provides poor results it is still an indicator of the relative performance of a general model, personalized model, hybrid model and user adaptive model.

# Chapter 5

# Experiments and results: Speech mood recognition

## 5.1 Dataset

The data [6, 9] has been collected from a total 10 subjects, all of which are actors. The subjects were between 21 and 35 years old at the time of recording. Each subject was recorded reading a series of short texts where each text was supposed to be read displaying a specific emotions. The emotions, with their label representation, were as follows:

- anger("W")
- boredom("L")
- disgust("E")
- anxiety/fear("A")
- happiness("F")
- sadness("T")
- neutral("N")

This gives a total of 7 different classes to predict, a tougher challenge than the motor activity dataset which was a binary classification problem. From figure 5.1 we can see that while this dataset is not perfectly balanced, it is a lot more balanced than the migraine dataset. Also because there is several more classes the inbalance will not have as much impact. Because of this it was decided that oversampling was not necessary for these experiments.

## 5.2 Feature Extraction

The features were extracted using the pyAudioAnalysis [44] tool, which is an open source python library that provides audio-related functionality like feature extraction, classification, segmentation and visualization.

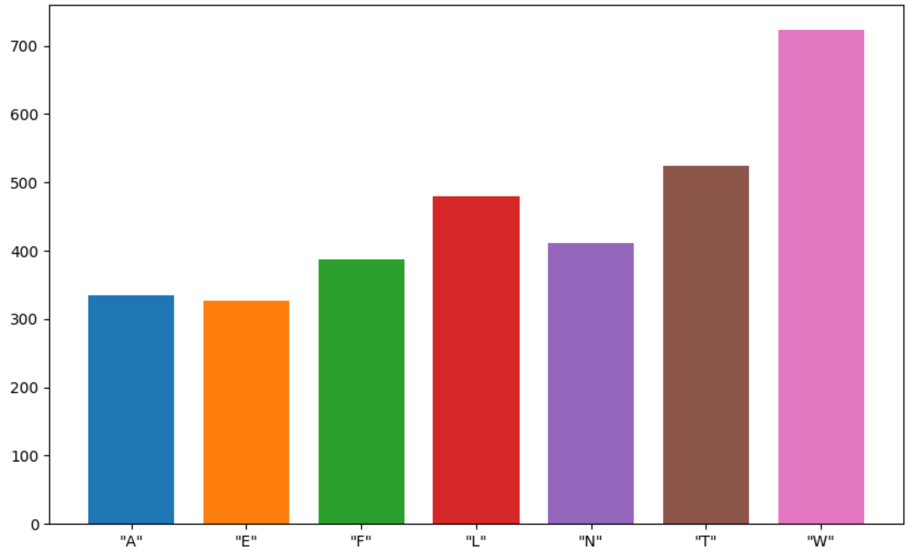

Figure 5.1: The distribution of samples across classes for the emotions dataset.

| | |
|---|---|
| Zero Crossing Rate | Energy |
| Entropy of Energy | Spectral Centroid |
| Spectral Spread | Spectral Entropy |
| Spectral Flux | Spectral Rolloff |
| MFCCs | Chroma Vector |
| Chrome Deviation | |

Table 5.1: Short-term features

The feature extraction consists of two steps; short-term and mid-term feature extraction. The short-term feature extraction splits the input into short-term windows and computes the features seem in table 5.1 for each window-frame.

The mid-term feature extraction calculates a number of statistics, like mean and standard deviation, on the short-term feature sequences. A total of 68 features were extracted containing 3189 samples.

## 5.3 Experiments description

The experiments for this dataset closely resemble the previous experiments, but as there is no need for oversampling those are not conducted and as stratified sampling proved superior to chronological sampling we will use stratified sampling exclusively. It is also worth noting that the metrics will not include AUC for these experiments because it proved to cumbersome to implement for a multinomial classification problem.

**General model** Training and testing the classifiers using a general model approach. There is one fold for each subjects where each classifier is trained on all the other subjects and tested on the one left out. No oversampling is applied.

**Personalized model** Training and testing the classifiers using a personalized model approach. Classifiers are trained and tested using 10-fold cross validation for each subject. There will also be trained a model using only 90 samples per subject

**Hybrid model using stratified sampled data** Training a hybrid model by sampling stratified sampled data from the target user.

**User adaptive model** Training a user adaptive model by using transfer learning techniques. One model was trained for each sample size in the same interval that was used in the other hybrid models. The samples used for the retraining of the classification layers were sampled using stratified sampling.

## 5.4 Results

The experiments have been conducted very much in the same way as in section 4.6, but since this dataset is multiclass and not binary the classifiers had to be trained slightly different. The multilayer perceptron used the softmax activation function in the classifier layer instead of the sigmoid function used in the previous experiments. The other classifiers followed the one-vs-rest technique which means that there was created one classifier for every class.
Since this dataset contains seven classes distributed over very few samples collected from 10 different subjects the experiments conducted is not expected to provide very good results, though they are expected to perform better than that of the motor activity dataset. The reasoning behind this is that the domain is not as difficult and that the classes are better balanced.

### 5.4.1 General model

The general model created from this voice activity dataset is expected to perform better than the motor activity dataset as it has data from more subjects that might help generalize the model. As the data has been collected in a controlled environment from actors it is a fair assumption to make that the emotions displayed in the speech samples follows the stereotypical representation of that emotion which makes it more plausible that the subjects display the emotions in the same way. This might not be the case in reality.

Looking at table 5.2, we can see that the overall performance of the classifiers is quite a lot better when trained and tested on this dataset. The accuracy also seems to be a better performance metric in this case, which can be attributed to the class balance and number of classes. Still it is important to look at all the metrics, but there seems to be a strong correlation between the metrics, at least between the precision and recall, and thus the F1-score. Looking at the classifiers we can see that the precision and recall seem to be almost equal while the accuracy lies right above. The specificity is quite high here as well, which indicates that the problem is still accepting classes and not rejecting them. The Random Forest, AdaBoost and Multilayer Perceptron stands out in terms of performance and while the two first comes as no surprise with the former experiments in mind, it is a bit surprising that the MLP performs as well as it does. |

| Algorithm | Accuracy(std) | Precision(std) | Recall(std) | F1(std) | Specificity(std) | MCC(std) |
|---|---|---|---|---|---|---|
| SVM | 0.490(0.083) | 0.462(0.085) | 0.431(0.075) | 0.385(0.078) | 0.857(0.034) | 0.407(0.092) |
| Random forest | 0.501(0.055) | 0.465(0.090) | 0.436(0.083) | 0.404(0.080) | 0.860(0.023) | 0.413(0.060) |
| Decision tree | 0.375(0.041) | 0.328(0.056) | 0.307(0.047) | 0.278(0.044) | 0.796(0.022) | 0.248(0.041) |
| K-nn | 0.445(0.054) | 0.388(0.050) | 0.372(0.042) | 0.347(0.044) | 0.829(0.026) | 0.342(0.055) |
| Gaussian NB | 0.430(0.062) | 0.358(0.093) | 0.365(0.072) | 0.322(0.077) | 0.823(0.030) | 0.330(0.071) |
| AdaBoost | 0.471(0.057) | 0.461(0.062) | 0.437(0.087) | 0.407(0.072) | 0.847(0.029) | 0.379(0.067) |
| Bagging | 0.466(0.066) | 0.408(0.089) | 0.407(0.080) | 0.376(0.084) | 0.840(0.033) | 0.367(0.075) |
| **MLP** | **0.497(0.080)** | **0.477(0.071)** | **0.446(0.089)** | **0.404(0.080)** | **0.860(0.034)** | **0.416(0.089)** |
| Dummy | 0.197(0.028) | 0.160(0.023) | 0.148(0.009) | 0.127(0.012) | 0.670(0.027) | 0.006(0.023) |

Table 5.2: Mean across users of the general model trained on data from the voice activity dataset

### 5.4.2 Personalized model

The personalized model has significantly better performance as can be seen in 5.3. The SVM has the best performance closely followed by the Random forest, AdaBoost and MLP.

The standard deviation is quite low, meaning that the classifiers handle data from all the different subjects with quite similar performance. The same, low standard deviation was seen in the general model in table 5.2. This is a good indication that one can create robust models for emotions detection, but as the number of samples is quite low and collected in a controlled environment it would be wrong to conclude anything from this.

#### Personalized model from 90 samples

The previous experiments showed that it was not necessarily better to create a hybrid model on the available data instead of just creating a personalized model. Again the performance of the model created from fewer samples is performing considerably worse, but this is to be expected. The SVM is the best classifier on both personalized models.

| Algorithm | Accuracy(std) | Precision(std) | Recall(std) | F1(std) |
|---|---|---|---|---|
| **SVM** | **0.702(0.058)** | **0.662(0.063)** | **0.650(0.063)** | **0.630(0.062)** |
| Random forest | 0.669(0.046) | 0.608(0.069) | 0.588(0.044) | 0.569(0.049) |
| Decision tree | 0.456(0.040) | 0.427(0.049) | 0.384(0.034) | 0.367(0.033) |
| K-nn | 0.598(0.052) | 0.544(0.067) | 0.532(0.056) | 0.509(0.057) |
| Gaussian NB | 0.600(0.050) | 0.547(0.040) | 0.544(0.037) | 0.521(0.038) |
| AdaBoost | 0.633(0.056) | 0.582(0.056) | 0.575(0.050) | 0.558(0.050) |
| Bagging | 0.602(0.046) | 0.535(0.060) | 0.533(0.043) | 0.511(0.050) |
| MLP | 0.660(0.053) | 0.581(0.069) | 0.583(0.049) | 0.554(0.050) |
| Dummy | 0.219(0.044) | 0.154(0.032) | 0.158(0.021) | 0.134(0.021) |

| Algorithm | Specificity(std) | MCC(std) |
|---|---|---|
| **SVM** | 0.929(0.017) | 0.648(0.066) |
| Random forest | 0.918(0.015) | 0.605(0.052) |
| Decision tree | 0.844(0.018) | 0.347(0.051) |
| K-nn | 0.893(0.020) | 0.522(0.058) |
| Gaussian NB | 0.894(0.019) | 0.522(0.056) |
| AdaBoost | 0.907(0.020) | 0.561(0.065) |
| Bagging | 0.895(0.016) | 0.523(0.053) |
| MLP | 0.915(0.016) | 0.596(0.058) |
| Dummy | 0.684(0.044) | 0.021(0.032) |

Table 5.3: Mean across users of the personalized model trained on data from the speech activity data

### 5.4.3 Hybrid model

The hybrid model was created using stratified sampling as that method provided the best results in the previous experiments. Unfortunately this dataset does not have as many samples per user as the motor activity dataset, meaning that we will only be able to see the change over a quite small interval of samples.

Figures 5.2 and 5.3 show the accuracy and F1 score from the hybrid model respectively. We can see that the number of samples that was tried out was 0 to 90 samples, which is quite low. We can however see that some classifiers, such as the random forest and SVM experience a boost in performance. It does not really compete with the performance of the personalized model, but it is still an improvement.

| Algorithm | Accuracy(std) | Precision(std) | Recall(std) | F1(std) |
|---|---|---|---|---|
| **SVM** | **0.590(0.087)** | **0.524(0.092)** | **0.530(0.082)** | **0.500(0.089)** |
| Random forest | 0.559(0.090) | 0.475(0.098) | 0.490(0.080) | 0.456(0.090) |
| Decision tree | 0.385(0.066) | 0.319(0.051) | 0.324(0.052) | 0.289(0.047) |
| K-nn | 0.516(0.077) | 0.438(0.084) | 0.457(0.069) | 0.416(0.073) |
| Gaussian NB | 0.517(0.065) | 0.456(0.053) | 0.453(0.041) | 0.425(0.049) |
| AdaBoost | 0.499(0.086) | 0.463(0.074) | 0.448(0.073) | 0.416(0.076) |
| Bagging | 0.478(0.079) | 0.419(0.067) | 0.428(0.069) | 0.397(0.067) |
| MLP | 0.547(0.087) | 0.465(0.082) | 0.486(0.068) | 0.446(0.074) |
| Dummy | 0.209(0.040) | 0.142(0.031) | 0.154(0.017) | 0.129(0.020) |

| Algorithm | Specificity(std) | MCC(std) |
|---|---|---|
| **SVM** | **0.890(0.031)** | **0.516(0.096)** |
| Random forest | 0.876(0.036) | 0.477(0.098) |
| Decision tree | 0.794(0.043) | 0.258(0.075) |
| K-nn | 0.859(0.033) | 0.430(0.087) |
| Gaussian NB | 0.862(0.025) | 0.426(0.066) |
| AdaBoost | 0.856(0.035) | 0.410(0.096) |
| Bagging | 0.839(0.039) | 0.380(0.087) |
| MLP | 0.874(0.033) | 0.467(0.095) |
| Dummy | 0.665(0.059) | 0.010(0.029) |

Table 5.4: Mean performance across users of the personalized model trained on 90 samples per subject from the voice activity dataset

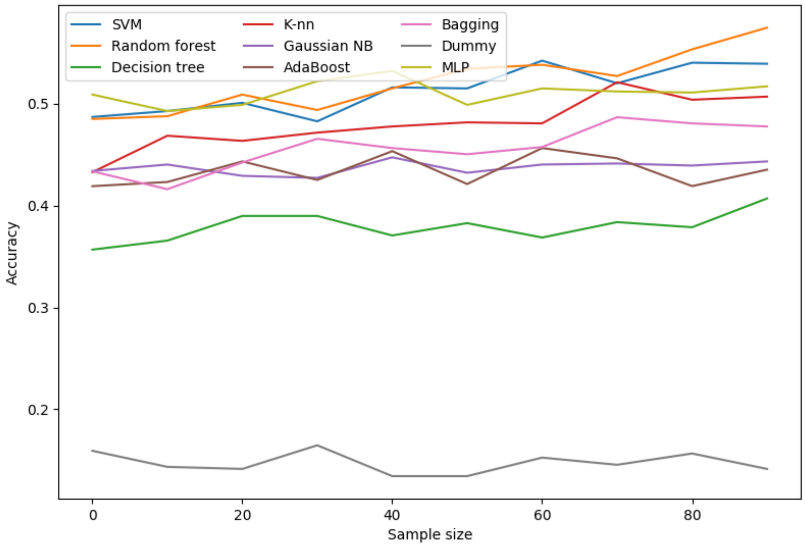

Figure 5.2: Accuracy from the hybrid model trained on the emotions dataset

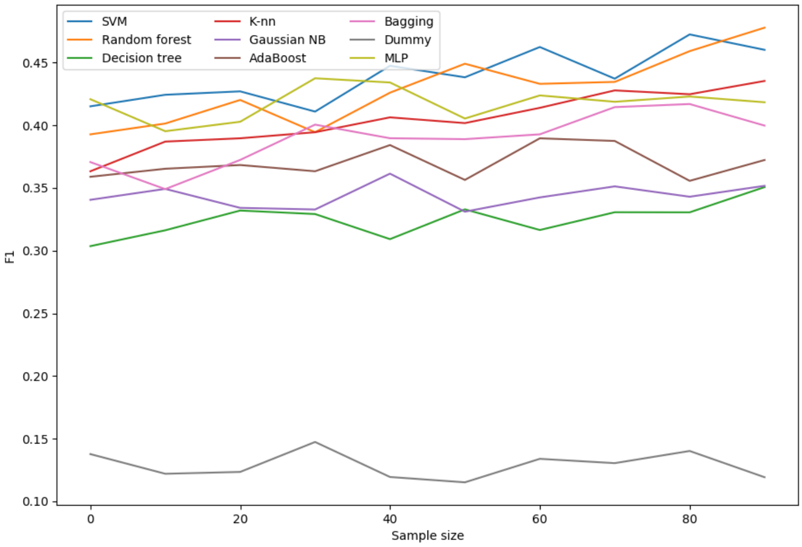

Figure 5.3: F1 score from the hybrid model trained on the emotions dataset

### 5.4.4 User Adaptive Model

The user adaptive model, which is trained using transfer learning, shows an incredible boost in performance after only a few samples, as can be seen in figures 5.4 and 5.5. The performance mirrors that of the personalized model and is significantly better than the hybrid model.

It seems that the performance is best at 70 samples, but it has a major performance increase after only 20 samples, which is notably different from the hybrid model which has a less drastic change.

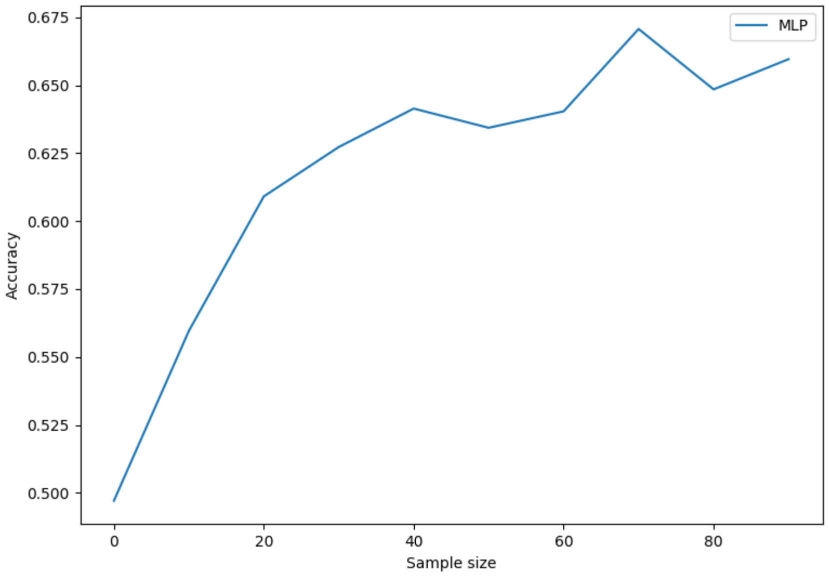

Figure 5.4: Accuracy from the user adaptive model trained on the emotions dataset

Figure 5.6 visualizes the relation between the accuracy of the best classifier in each model. The random forest had the best performance in both the general and hybrid model, but the SVM had the best performance in both the personalized models.

It shows that while the hybrid model was an improvement on the general model, it is no better than creating a personalized model, as was the case with the user adaptive model in the previous experiments. The user adaptive model is however clearly superior to all models but the personalized model created from all available data.

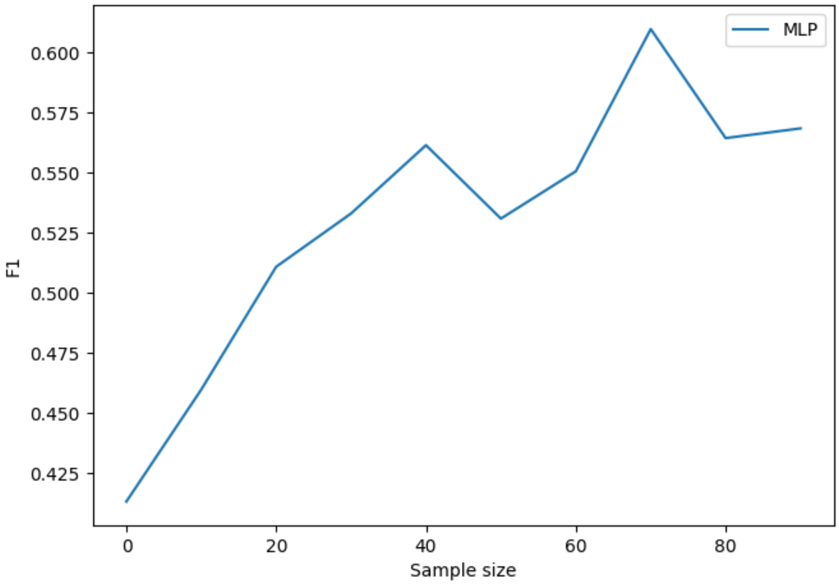

Figure 5.5: F1 score from the user adaptive model trained on the emotions dataset

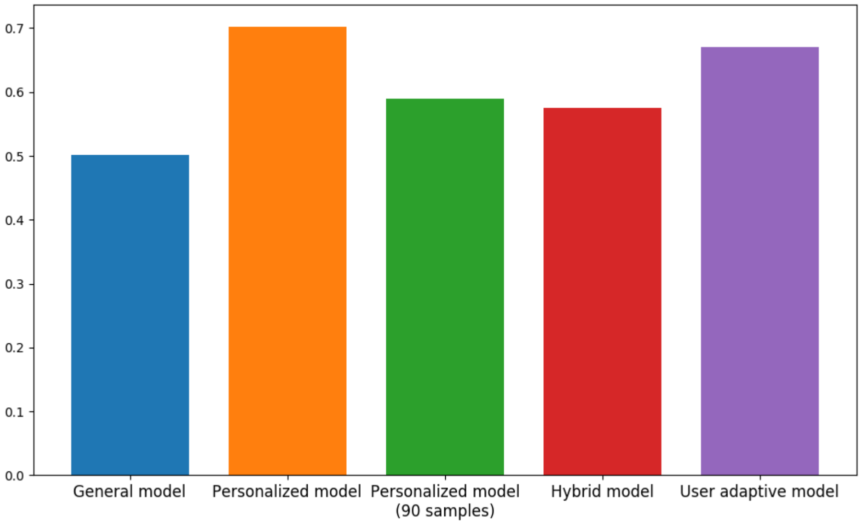

Figure 5.6: The best accuracy of the different models on the emotions dataset

## 5.5 Discussion

This dataset was better balanced than the migraine dataset which showed in the results. The downside is that there were not as many samples per user and class. The total number of samples was considerably less than that of the migraine dataset, but the number of both the subjects and classes was considerably higher.

The gap between the general and personalized model was as expected, but the performance of the personalized model was a bit disappointing. Given how the data was collected it was expected that it would perform a bit better. The fact that the dataset was better balanced did however show in the considerably lower standard deviation which shows the robustness of the models.

The hybrid models suffered from the low number of samples per subject, seeing that the number of samples that could be transferred was quite low at a maximum of 90 samples. The hybrid model did not show much of an improvement, and it was no better than creating a personalized model on the same amount of data. At most the accuracy improved by roughly 9 percentage points in the random forest classifier. The user adaptive model did however experience a great boost in performance after training on very few samples which is promising.

# Chapter 6

# Conclusion

The thesis has trained and compared different classifiers using general, personalized and hybrid models for mental health prediction. The work was split into two parts; one dataset predicting migraine attacks in subjects with bipolar disorder based on motor activity and one dataset classifying emotions based on speech analysis.

Both datasets had advantages and disadvantages, but looking at the combination of them the results might be more reliable and significant. The migraine prediction dataset was very unbalanced, but it contained a lot of samples compared to the emotions prediction dataset which was more balanced, but very small. The results from the migraine dataset does not indicate that it is possible to predict such migraine attacks based on motor activity alone, but they might be unreliable because of the poor data available.
The performance of the general and personalized model in the emotions dataset has a stronger indication of the possibility of using machine learning as a viable method for emotions prediction, but that dataset also has the disadvantage of being collected in a controlled environment from actors and might not be representative for samples collected in the real world. The results are however promising as the dataset was very small and could probably perform even better on a larger dataset.

To make the final conclusion we take a look at the research questions introduced in chapter 1:

1. **How do general and personalized models perform in the mental health domain?**
   The general and personalized model did not perform well when applied to the two datasets available. This might stem from the small amount of data or the difficulty of the task in question. The personalized model did significantly better than the general model in both cases as was expected.

2. **Is it possible to create a hybrid model that harbors the advantages of both the general and personalized model?**

Yes, figures 4.10 and 5.6 show that the hybrid and user adaptive models can almost compete with the personalized model in terms of performance with significantly less data from the target subject.

3. **Can such a hybrid model be built by adding data from the target user to the training data?**
   Yes, we have shown that by sampling data in a balanced manner from the target subject and adding it to the training data we can create a hybrid model that performs better than the general model and almost as well as the personalized model. However, the hybrid model was no better than just using the same amount of data to create a personalized model, when it was trained and tested on the emotions dataset.

4. **Can such a hybrid model be built using transfer learning techniques?**
   Yes, the user adaptive model showed the best results when taking into account both performance and amount of data required. The user adaptive model improved significantly based on a tiny amount of data from the target subject. As with the hybrid model it should however be compared to a personalized model trained on the same amount of samples, as that might be more beneficial.

## 6.1 Further work

While this thesis has made it clear that hybrid models are a viable approach in the mental health domain, it would be interesting to see it researched on bigger and better datasets. To properly research whether it is possible to predict migraine attacks based on motor activity this would be a necessity.

Furthermore this thesis did not focus on tuning any of the classifiers used and applied a very basic form of deep learning. Researching the use of deep learning in the mental health domain might prove beneficial as the domain relies heavily on analysis of speech and motor activity which deep learning has excelled in over the last years. It might also be interesting to do predictions on text analysis from text messages and such, using the same techniques.

This thesis did only explore the use of labeled data and supervised learning, but it would be interesting to try out unsupervised learning to mitigate the process of labeling data altogether.

# Bibliography

[1] Z.S. Abdallah et al. 'StreamAR: Incremental and Active Learning with Evolving Sensory Data for Activity Recognition'. In: *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*. Vol. 1. Nov. 2012, pp. 1163–1170. DOI: 10.1109/ICTAI.2012.169.

[2] *AdaBoost*. Jan. 2018. URL: http://scikit-learn.org/stable/modules/ensemble.html#adaboost.

[3] Ian M Anderson, Peter M Haddad and Jan Scott. 'Bipolar disorder'. In: *BMJ* 345 (2012). DOI: 10.1136/bmj.e8508. eprint: http://www.bmj.com/content/345/bmj.e8508.full.pdf. URL: http://www.bmj.com/content/345/bmj.e8508.

[4] *Assessment of Artificial Neural Network for bathymetry estimation using High Resolution Satellite imagery in Shallow Lakes: Case Study El Burullus Lake*. Accessed january 31. 2018. Apr. 2015. URL: https://www.researchgate.net/A-hypothetical-example-of-Multilayer-Perceptron-Network_273768094.

[5] Jan O. Berle et al. 'Actigraphic registration of motor activity reveals a more structured behavioural pattern in schizophrenia than in major depression'. In: *BMC Research Notes* 3.1 (May 2010), p. 149. DOI: 10.1186/1756-0500-3-149. URL: https://doi.org/10.1186/1756-0500-3-149.

[6] *Berlin Database of Emotional Speech*. http://emodb.bilderbar.info/docu/. Accessed: 28 January 2018. 1999.

[7] Leo Breiman. 'Bagging predictors'. In: *Machine Learning* 24.2 (Aug. 1996), pp. 123–140. ISSN: 1573-0565. DOI: 10.1007/BF00058655. URL: https://doi.org/10.1007/BF00058655.

[8] Leo Breiman. 'Random Forests'. In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[9] Felix Burkhardt et al. 'A database of German emotional speech'. In: *Ninth European Conference on Speech Communication and Technology*. 2005.

[10] Michel Cabanac. 'What is emotion?' In: *Behavioural Processes* 60.2 (2002), pp. 69–83. ISSN: 0376-6357. DOI: https://doi.org/10.1016/S0376-6357(02)00078-5. URL: http://www.sciencedirect.com/science/article/pii/S0376635702000785.

[11] Nitesh V. Chawla et al. 'SMOTE: Synthetic Minority Over-sampling Technique'. In: *Journal of Artificial Intelligence Research* (June 2002).

[12] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[13] *Data and statistics*. Jan. 2018. URL: `http://www.euro.who.int/en/health-topics/noncommunicable-diseases/mental-health/data-and-statistics`.

[14] *Deep learning*. URL: `https://www.nature.com/articles/nature14539`.

[15] Depression and Bipolar Support Alliance. *Bipolar Disorder Statistics*. URL: `http://www.dbsalliance.org/site/PageServer?pagename=education_statistics_bipolar_disorder` (visited on 30/08/2017).

[16] Dua Dheeru and Efi Karra Taniskidou. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml`.

[17] John Duchi, Elad Hazan and Yoram Singer. 'Adaptive Subgradient Methods for Online Learning and Stochastiz Optimization'. In: *Journal of Machine Learning Research* (2011).

[18] *Dummy estimators*. Jan. 2001. URL: `http://scikit-learn.org/stable/modules/model_evaluation.html#dummy-estimators`.

[19] Ramin Fallahzadeh and Hassan Ghasemzadeh. 'Personalization without user interruption: boosting activity recognition in new subjects using unlabeled data'. In: *Proceedings of the 8th International Conference on Cyber-Physical Systems*. ACM. 2017, pp. 293–302.

[20] M. Faurholt-Jepsen et al. 'Smartphone data as objective measures of bipolar disorder symptoms'. In: *Psychiatry Res* 217.1-2 (June 2014), pp. 124–127.

[21] M. Faurholt-Jepsen et al. 'Voice analysis as an objective state marker in bipolar disorder'. In: *Transl Psychiatry* 6 (July 2016), e856.

[22] Yoav Freund and Robert E. Schapire. 'A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting'. In: *Journal of Computer and System Sciences* (1996).

[23] E. Garcia-Ceja, V. Osmani and O. Mayora. 'Automatic Stress Detection in Working Environments From Smartphones' Accelerometer Data: A First Step'. In: *IEEE Journal of Biomedical and Health Informatics* 20.4 (July 2016), pp. 1053–1060. ISSN: 2168-2194. DOI: `10.1109/JBHI.2015.2446195`.

[24] Enrique Garcia-Ceja and Ramon Brena. 'Building Personalized Activity Recognition Models with Scarce Labeled Data Based on Class Similarities'. In: *Ubiquitous Computing and Ambient Intelligence. Sensing, Processing, and Using Environmental Information*. Ed. by Juan M. García-Chamizo, Giancarlo Fortino and Sergio F. Ochoa. Cham: Springer International Publishing, 2015, pp. 265–276. ISBN: 978-3-319-26401-1.

[25] Steve R. Gunn. *Support Vector Machines for Classification and Regression*. 1998.

[26] Isabelle Guyon and André Elisseeff. 'An Introduction to Feature Extraction'. In: *Feature Extraction: Foundations and Applications*. Ed. by Isabelle Guyon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–25. ISBN: 978-3-540-35488-8. DOI: 10.1007/978-3-540-35488-8_1. URL: https://doi.org/10.1007/978-3-540-35488-8_1.

[27] Tin Kam Ho. 'Random Decision Forests'. In: *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*. ICDAR '95. IEEE Computer Society, 1995, pp. 278–. ISBN: 0-8186-7128-9. URL: http://dl.acm.org/citation.cfm?id=844379.844681.

[28] *ImageNet*. URL: http://www.image-net.org/.

[29] Z. N. Karam et al. 'ECOLOGICALLY VALID LONG-TERM MOOD MONITORING OF INDIVIDUALS WITH BIPOLAR DISORDER USING SPEECH'. In: *Proc IEEE Int Conf Acoust Speech Signal Process* 2014 (May 2014), pp. 4858–4862.

[30] Guillaume Lemaître, Fernando Nogueira and Christos K. Aridas. *Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning*. [Online; accessed January 29, 2018]. URL: http://contrib.scikit-learn.org/imbalanced-learn/stable/_images/sphx_glr_plot_random_over_sampling_001.png.

[31] Guillaume Lemaître, Fernando Nogueira and Christos K. Aridas. *Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning*. [Online; accessed January 29, 2018]. URL: http://contrib.scikit-learn.org/imbalanced-learn/stable/_images/sphx_glr_plot_smote_001.png.

[32] Guillaume Lemaître, Fernando Nogueira and Christos K. Aridas. 'Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning'. In: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5. URL: http://jmlr.org/papers/v18/16-365.html.

[33] Jeffrey W. Lockhart and Gary M. Weiss. 'The Benefits of Personalized Smartphone-Based Activity Recognition Models'. In: *Proceedings of the 2014 SIAM International Conference on Data Mining*, pp. 614–622. DOI: 10.1137/1.9781611973440.71. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611973440.71. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611973440.71.

[34] Hong Lu et al. 'StressSense: Detecting Stress in Unconstrained Acoustic Environments Using Smartphones'. In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. UbiComp '12. New York, NY, USA: ACM, 2012, pp. 351–360. ISBN: 978-1-4503-1224-0. DOI: 10.1145/2370216.2370270. URL: http://doi.acm.org/10.1145/2370216.2370270.

[35] Stephen Marsland. *Machine Learning: An Algorithmic Perspective, Second Edition*. 2nd. Chapman & Hall/CRC, 2014. ISBN: 1466583282, 9781466583283.

[36] Alban Maxhuni et al. 'Classification of bipolar disorder episodes based on analysis of voice and motor activity of patients'. In: *Pervasive and Mobile Computing* 31 (2016), pp. 50–66. ISSN: 1574-1192. DOI: `http://dx.doi.org/10.1016/j.pmcj.2016.01.008`. URL: `http://www.sciencedirect.com/science/article/pii/S1574119216000109`.

[37] *Mental disorders*. URL: `http://www.who.int/mediacentre/factsheets/fs396/en/`.

[38] *Migraine*. URL: `https://www.nhs.uk/conditions/migraine/`.

[39] Tom M. Mitchell. *Machine Learning*.

[40] J. T. O'Brien et al. 'A study of wrist-worn activity measurement as a potential real-world biomarker for late-life depression'. In: *Psychological Medicine* 47.1 (2017), pp. 93–102. DOI: `10.1017/S0033291716002166`.

[41] S. J. Pan and Q. Yang. 'A Survey on Transfer Learning'. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. ISSN: 1041-4347. DOI: `10.1109/TKDE.2009.191`.

[42] Jussi Parviainen et al. 'Adaptive Activity and Environment Recognition for Mobile Phones'. In: *Sensors* 14.11 (2014), pp. 20753–20778. ISSN: 1424-8220. DOI: `10.3390/s141120753`. URL: `http://www.mdpi.com/1424-8220/14/11/20753`.

[43] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[44] *Python Audio Analysis Library*. https://github.com/tyiannak/pyAudioAnalysis. Accessed: 28 January 2018. 2016.

[45] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN: 1-55860-238-0.

[46] *Receiver Operating Characteristic (ROC)*. URL: `http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html`.

[47] Seyed-Ali Rokni, Marjan Nourollahi and Hassan Ghasemzadeh. 'Personalized Human Activity Recognition Using Convolutional Neural Networks'. In: *32nd AAAI Conference on Aritificial Intelligence (AAAI-18)*. 2018, p. 00.

[48] *Sensitivity and specificity*. URL: `https://commons.wikimedia.org/wiki/File:Sensitivity_and_specificity.svg`.

[49] E. Soria et al. *Handbook of Research on Machine Learning Applications*. 2009.

[50] Nitish Srivastava et al. 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting'. In: *Journal of Machine Learning Research* (2014).

[51] *Transfer Learning*. URL: `http://cs231n.github.io/transfer-learning/`.

[52] Sebastian Trautmann, Jürgen Rehm and Hans-Ulrich Wittchen. 'The economic costs of mental disorders: Do our societies react appropriately to the burden of mental disorders?' In: *EMBO Reports* (Sept. 2016). DOI: 10.15252/embr.201642951..

[53] Vladimir Vapnik. *The nature of statistical learning theory*. Springer, 1995. ISBN: 0-387-94559-8.

[54] Qianli Xu, Tin Lay Nwe and Cuntai Guan. 'Cluster-Based Analysis for Personalized Stress Evaluation Using Physiological Signals'. In: *Biomedical and Health Informatics, IEEE Journal of* 19.1 (Jan. 2015), pp. 275–281. ISSN: 2168-2194. DOI: 10.1109/JBHI.2014.2311044.

[55] Jason Yosinski et al. 'How transferable are features in deep neural networks?' In: *CoRR* abs/1411.1792 (2014). arXiv: 1411.1792. URL: http://arxiv.org/abs/1411.1792.