Thesis

Machine Based Learning of Multidimensional Data in Bipolar Disorder - Choice of Methods and Pilot Results

submitted by

Clemens ELST, BSc

in fulfilment of the requirements for the degree of

Doctor of Medicine
(Dr. med. univ.)

at the

Medical University of Graz

executed at the
Department of Psychiatry and Psychotherapeutic Medicine

under the guidance of

Univ. FA Priv.-Doz. Dr.med.univ. Dr.scient.med. Armin Birner
and Ass.-Prof. Dipl.-Ing. Dr.scient.med. Markus Kreuzthaler

Graz, 17th March 2022

Statutory Declaration

I hereby confirm that the present diploma thesis is the result of my own independent scholarly work. I also confirm that in all cases, where material from the work of others (in books, articles, essays, dissertations, and on the internet) is acknowledged, quotations and paraphrases are clearly indicated. No material other than that cited in the reference list has been used. I have read and understood the Medical University's regulations and procedures concerning plagiarism.

Graz, 17th March 2022 Clemens Elst m.p.

Acknowledgements

This thesis was written in 2022 at the Medical University of Graz's Department of Psychiatry and Psychotherapeutic Medicine.

First and foremost, I would like to thank both my supervisors, Univ. FA Priv.-Doz. Dr.med.univ. Dr.scient.med. Armin Birner and Ass.-Prof. Dipl.-Ing. Dr.scient.med. Markus Kreuzthaler, for developing the idea of this thesis, providing the data for my research and especially for their guidance and support during writing.

Further, I want to thank my cousin Stephanie for excellently proofreading my work.

Special thanks are due to my parents for always providing their support, emotionally and financially, and enabling me to change my career path by moving from computer science to medicine.

Graz, 17th March 2022 Clemens Elst m.p.

Contents

List of Abbreviations

AMWF	Association of Scientific Medical Societies in Germany
AUROC	Area Under the Receiver Operating Characteristic
BMI	Body Mass Index
CANTAB	Cambridge Neuropsychological Test Automated Battery
CV	Cross-Validation
CVLT	California Verbal Learning Test
DSM-5	Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition
GRADE	Grading of Recommendations Assessment, Development and Evaluation
ICD-11	ICD-11 International Classification of Diseases for Mortality and Morbidity Statistics - Eleventh Revision
LTU	Linear Threshold Unit
MLP	Multi-Layer Perceptron
PCA	Principal Component Analysis

RFECV	Recursive Feature Elimination with Cross-Validation
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbor Embedding
TMT	Trail Making Test
WAIS	Wechsler Adult Intelligence Scale
WHO	World Health Organization

List of Figures

List of Tables

Zusammenfassung

Bipolare Störung ist eine psychiatrische Erkrankung mit grossen Auswirkungen auf das gesellschaftliche Leben der betroffenen PatientInnen, geht mit einem erhöhten Risiko für wiederholte Krankenhausaufenthalte und einem erhöhten Lebenszeitrisiko durch Suizidalität einher. Verzögerungen beim Stellen der korrekten Diagnose und beim Beginn einer wirksamen Behandlung resultieren in schlechteren Therapieergebnissen, sind aber aufgrund des sehr heterogenen Krankheitsbildes immer noch häufig. Techniken des maschinellen Lernens können dazu beitragen, die Diagnose "Bipolare Störung" früher zu stellen und so die Zeit vom Auftreten der ersten Symptome bis zum Beginn der Behandlung zu verkürzen.

Um diese Hypothese zu testen, wurden demografische Informationen und Ergebnisse aus kognitiven Leistungstests von 196 PatientInnen mit Bipolarer Störung und 145 gesunden Kontrollpersonen verwendet, um damit fünf verschiedene Algorithmen des maschinellen Lernens zu trainieren. Die Leistung dieser Algorithmen wurde verglichen, um den für diese Problemstellung am besten geeigneten Algorithmus zu finden. Der Algorithmus mit der besten Leistung war Logistische Regression mit einem macro-average F1 Score von 0,69 [95% CI 0,66 - 0,73]. Nach weiterer Optimierung konnte ein Modell mit einem macro-average F1 Score von 0,75, einem micro-average F1 Score von 0,77 und einer AUROC von 0,84 trainiert werden. Auf der Grundlage dieses Modells wurde analysiert, inwieweit einzelne Variablen zur Klassifizierung beitragen. Das Ergebnis war, dass der BMI eines/r PatientIn und die Ergebnisse des Stroop Tests und des d2/d2-R Tests allein eine Klassifizierung mit annähernd gleicher Genauigkeit ermöglichen.

Die Verwendung dieser Daten für eine klinische Anwendung bringt eine akzeptable Leistung, ist aber noch nicht geeignet, um die Diagnosefindung durch eine/n erfahrene/n ÄrztIn effektiv zu unterstützen. Der Schwerpunkt der weiteren Forschung sollte auf der Identifizierung von Variablen liegen, die einen hohen Beitrag zur Klassifizierung leisten, um so die Leistung von Modellen des maschinellen Lernens weiter zu verbessern und dadurch in weiterer Folge die Diagnosestellung der Bipolaren Störung zu vereinfachen.

Abstract

Bipolar disorder is a psychiatric disorder with a high impact on a patient's normal societal function and is associated with an increased risk of repeated hospitalizations and an elevated lifetime risk for suicide. Delays in the correct diagnosis and to the start of an effective treatment are associated with poorer outcomes, but are common due to a very heterogeneous progression of the disorder. Techniques of machine learning can be used to aid in the diagnosis of bipolar disorder and shorten the time from the onset of symptoms to the beginning of treatment.

To test this hypothesis, a de-identified data set of demographic information and the results of cognitive tests of 196 patients with bipolar disorder and 145 healthy controls was used to train and compare five different machine learning algorithms. The best performing algorithm was Logistic Regression, with a macro-average F1 score of 0.69 [95% CI 0.66 - 0.73]. After further optimization, a model with an improved macro-average F1 score of 0.75, a micro-average F1 score of 0.77 and an AUROC of 0.84 could be built. Based on this model, it was analyzed how much each single variable contributes to the classification, which resulted in the finding that a patient's BMI and results of the Stroop test and the d2/d2-R test alone allow for a classification with equal performance.

Using this data for clinical application results in an acceptable performance, but has not yet reached a state where it can sufficiently augment a diagnosis made by an experienced clinician. The focus of further research should be to identify variables with a high contribution to classification to further improve the performance of machine learning models in this context and to subsequently simplify the diagnosis of bipolar disorder.

Chapter 1

Introduction

1.1 Motivation

According to the results of the 2019 "Global Burden of Disease study", bipolar disorder has a global prevalence of 39.5 (33.0 to 46.8) cases per one million [1]. Estimates of lifetime prevalence are approximately 3 - 4% measured in study populations from the United States [2, 3]. A literature review in 2005 found a cumulative lifetime prevalence of 1.5 - 2% for European countries, when including bipolar spectrum disorders, the prevalence increased up to an estimated 6% [4]. Due to a heterogeneous clinical picture and a lack of validated tools for early recognition and diagnosis, there is a significant time span from the onset of symptoms to the correct diagnosis and the beginning of treatment. One study found an average delay of 8.8 years [5]. As bipolar disorder represents an enormous burden on society and an even higher one on patients, with considerable social, professional and health implications, early recognition and initiation of treatment of bipolar disorder should be a major goal of any public health system. To date, there are no widely used tools for early recognition. The recent increase in acceptance and use of machine learning in various clinical contexts presents an opportunity to use these applications to learn more about patterns in patients with bipolar disorder and to possibly develop diagnostic tools.

1.2 Objectives

The objective of this thesis is to develop a machine learning model to predict whether a patient has bipolar disorder or not. The data set used for prediction is a subset of the data set generated by the "BIPFAT" study carried out by the Department of Psychiatry and Psychotherapeutic Medicine at the Medical University of Graz. The data used deliberately only contains basic de-identified patient information and results of cognitive tests, which are not specific to bipolar disorder, and does not contain any specific diagnostic scores or results of imaging studies. The main question to be answered is, if it is possible to accurately predict bipolar disorder based on this data. The results of this thesis should be regarded as pilot results for this topic and as a basis for further study.

1.3 Outline

The work is structured as follows:

- Chapter 2 contains general information on bipolar disorder, consisting of definition, pathogenesis and treatment options, followed by a section illustrating problems in the diagnosis of bipolar disorder and presenting studies related to the research question of this thesis.
- Chapter 3 consists of the terminology and theoretical knowledge of machine learning that is required for understanding this thesis. The theoretical part is followed by a description of the data set used and, finally, details the exact methodology applied for answering the research question.
- Chapter 4 follows the methodology described in Chapter 3 and the results of the performed research are presented and discussed.
- Chapter 5 concludes the thesis with a general discussion of the findings and tries to contextualize them with other research. After a brief mention of possible limitations to this study and things to consider for a possible clinical real-world application, proposals for further research are made.

Chapter 2

Background and Related Work

2.1 Bipolar disorder

2.1.1 Classification of bipolar disorder

Diagnosis and classification of bipolar disorder is mostly done by using either the classification defined in the WHO's International Statistical Classification of Diseases and Related Health Problems (ICD), currently at the 10^{th} revision (ICD-10) with the 11^{th} revision (ICD-11) being released and slowly adapted into clinical practice [6], or as defined in the Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition (DSM-5) [7]. Both systems define three main categories: Bipolar I, Bipolar II and Cyclothymia, as well as some other specialized types like substance-induced bipolar disorder [6, 7]. Both systems define episodes of mood disorders which have to be present in a patient's history or at the time of examination to meet the criteria of one of these categories. In the following section, a brief overview of the types of bipolar disorder is given based on the WHO's definition in the released, but not yet widely implemented ICD-11.

Bipolar type I disorder - 6A60

Type I of bipolar disorder is characterized by "*the occurrence of one or more manic or mixed episodes*". The criteria for a manic episode include an extreme mood state defined

by "*euphoria, irritability, or expansiveness*" and an increased level of activity or perceived level of energy, alongside other symptoms like "*rapid or pressured speech, flight of ideas, increased self-esteem or grandiosity, decreased need for sleep, distractibility, impulsive or reckless behaviour, and rapid changes among different mood states*". A duration of at least one week has to be met to fulfill the criteria for a manic episode. A mixed episode is defined by rapidly alternating states of manic and depressive symptoms during a period of two weeks. A singular manic or mixed episode is sufficient for the diagnosis of type I bipolar disorder, regardless of whether a depressive episode occurs or not [8, ICD-11 Code 6A60].

Bipolar type II disorder - 6A61

For a diagnosis of type II bipolar disorder, the occurrence of one or more hypomanic episodes and a minimum of one depressive episode is required, without the occurrence of a manic or mixed episode. A hypomanic episode is defined as a mood state with a duration of several days or more with "*increased irritability, increased activity or a subjective experience of increased energy*" and symptoms like "*increased talkativeness, rapid or racing thoughts, increased self-esteem, decreased need for sleep, distractability, and impulsive or reckless behavior*". These symptoms constitute a change in a patient's baseline behaviour, but are not severe enough to result in an impaired function. A depressive episode is characterized as a "*period of almost daily depressed mood or diminished interest in activities occurring most of the day*" for at least two weeks, accompanied by "*changes in appetite or sleep, psychomotor agitation or retardation, fatigue, feelings of worthlessness or excessive or inappropriate guilt, feelings of hopelessness, difficulty concentrating, and suicidality*" [8, ICD-11 Code 6A61].

Cyclothymic disorder - 6A62

The criteria for cyclothymic disorder are met when multiple episodes with symptoms of hypomania or depression occur for most of the time over a period of at least two years, but the criteria for classification as either depressive or manic episode are not met. If criteria for a manic or a depressive episode are met, the requirements for the diagnosis of Bipolar

I or II are met, and a patient will be classified accordingly. Cyclothymic disorder impairs a patient's normal functioning in various aspects of everyday life [8, ICD-11 Code 6A62].

2.1.2 Etiology and pathogenesis

The exact etiology of bipolar disorder is unknown. Currently, a model of multifactorial genetic, neurobiological and psychosocial risk factors is discussed which, combined, influence the manifestation and progression of the disorder.

Genetics play a major role in bipolar disorder and are one of the most studied risk factors. First-degree relatives of patients with bipolar disorder have a lifetime risk of approximately 5 to 10 percent for developing bipolar depression, but there is also an association with higher rates of unipolar depression [9]. For comparison, the estimated lifetime prevalence for European countries is between 1.5 - 2 % according to Pini et al. [4]. Molecular genetic studies have identified multiple genetic loci associated with a higher risk for bipolar disorder. The impact of these gene polymorphisms each on its own is very small, but in combination they may result in a higher risk of disease. It is suggested, that certain genetic variations may interact with environmental factors, like substance abuse or stress, and therefore increase the susceptibility to developing bipolar disorder [9].

Differences in levels of neurotransmitters and their receptors are also discussed as potential factors for developing bipolar disorder. Associated neurotransmitters are serotonin, norepinephrine and dopamine and their corresponding receptors. Changes in receptor expression and affinity were found to be present in patients with bipolar disorder, but these differences are not specific and no coherent model of parthenogenesis based on these findings has been developed [10].

Other associated factors for the development of bipolar disorder have been found. Childhood trauma or maltreatment are general risk factors for psychological diseases, including bipolar disorder. The occurrence of stressful life events correlates with the onset of bipolar disorder or relapses into manic or depressive episodes, although these findings are also not specific for bipolar disorder only. Due to various confounding effects, a direct causality between major upsetting life events and episodes in bipolar disorder has not been shown [9].

Substance abuse has also been linked to the development and progression of bipolar disorder, with indications that abuse in adolescence and the consumption of higher doses increase the risk even further. In general, Cannabis is well studied in the context of bipolar disorder and psychiatric disorders, as well as other substances like alcohol or cocaine, but there is an ongoing discussion whether substance abuse triggers psychiatric disorders or vice versa. Apart from drugs, several medical comorbidities are also associated with an increased risk for developing bipolar disorder, with unclear causality [9].

Recent research focusing on neuroanatomical features has identified structural changes in the brains of patients with bipolar disorder. Affected regions are the prefrontal cortex, the subcortical area and the third and lateral ventricles, with varying specific findings in imaging and neuropathological studies. As with other associated factors, these findings are not specific for bipolar disorder, but can also be identified in unipolar depression [10].

The further development of a conclusive model of this disorder remains a topic for further research.

2.1.3 Progression and prognosis

Typically, patients with bipolar disorder suffer from multiple episodes of varying intensity and duration. The length of an episode can range from weeks to months. Exposure to the risk factors mentioned in chapter 2.1.2 can further worsen the progression of bipolar disorder, which leads up to chronic impairment in severe cases [11]. Some patients show residual symptoms even after successful treatment, which increases the probability of a relapse and can also have an impact on the patient's everyday life and their ability to function in social situations and to carry out cognitive tasks [12]. In addition to the impairment of social function, suicide risk is increased in patients with bipolar disorder. Two recent reviews found twenty- to thirty-fold increased suicide rates when compared to the general population [13, 14]. The lifetime risk of suicide for patients with confirmed diagnosis is estimated to be up to 19 percent [15].

2.1.4 Therapy

The *S3-Leitlinie zur Diagnostik und Therapie Bipolarer Störungen* (S3 Guideline on diagnosis and therapy of bipolar disorders) [16] is used in the following sections for an overview of therapeutic principles and to outline the pharmacological and non-pharmacological treatment of bipolar disorder. The main focus for therapy is to restore and preserve a patient's psychosocial function and the ability to participate in their daily life. A second important principle is that when the acute therapy for an episode has begun, it should be already carried out with the prevention of future episodes in mind. To achieve this, a patient's individual history, number of episodes, comorbidities, previous response to treatment and social setting need to be considered.

The guideline differentiates between the therapy for acute mania or hypomania, depressive episodes and recurrence prevention. The guideline lists medications and non-pharmacological treatments for each one of these settings and the grade of recommendation found by the guideline committee. The Association of Scientific Medical Societies in Germany (AMWF) as publisher of the guideline uses the Grading of Recommendations Assessment, Development and Evaluation (GRADE) approach [17] to systematically review and evaluate evidence. According to the AMWF's manual for guideline development, the levels of recommendation correlate with the GRADE's strength of evidence, as shown in table 2.1 [18].

Level	Recommendation	Quality of evidence (GRADE)
A	Strong recommendation	High
B	Weak recommendation	Moderate
0	No recommendation	Low and Very Low
CC	Clinical consensus	No available research, consensus only

Table 2.1: Levels of evidence and corresponding GRADE rating. Level CC when no applicable data is available and recommendations are based on consensus of the guideline committee.

Therapy of manic episodes

Pharmacological treatment plays a central role during these episodes, as other, non-pharmacological interventions require a high level of compliance from patients to be effective. For the therapy of manic episodes, various substances from different groups are available. This broad selection allows for an individualized approach based on the patient's needs, the medication's desired and potentially unwanted effects and the prior experience and effectiveness of this substance. Pharmaceutical monotherapy is well researched for treatment of manic episodes, with also the possibility of combinations of various substances. Table 2.2 shows the suggested pathway for treatment of manic episodes [16].

<table>
<tr><th>Substance</th><th>Recommendation</th><th></th><th></th></tr>
<tr><td colspan="2">First-line Monotherapy</td><td rowspan="16">Benzodiazepines</td><td rowspan="16">Psychotherapy</td></tr>
<tr><td>Carbamazepine</td><td>B</td></tr>
<tr><td>Lithium</td><td>B</td></tr>
<tr><td>Valproate</td><td>B</td></tr>
<tr><td>Aripiprazole</td><td>B</td></tr>
<tr><td>Asenapine</td><td>B</td></tr>
<tr><td>Olanzapine</td><td>B</td></tr>
<tr><td>Quetiapine</td><td>B</td></tr>
<tr><td>Risperidone</td><td>B</td></tr>
<tr><td>Ziprasidone</td><td>B</td></tr>
<tr><td>Haloperidol</td><td>B</td></tr>
<tr><td>Paliperidone</td><td>0</td></tr>
<tr><td colspan="2">Second-line Combination therapy
(if first-line is not sufficient)</td></tr>
<tr><td>Valproate or Lithium with
Olanzapine or Risperidone</td><td>B</td></tr>
<tr><td>Valproate or Lithium with
Aripiprazole or Quetiapine</td><td>0</td></tr>
<tr><td>Valproate or Lithium with
Allopurinol</td><td>0</td></tr>
<tr><td colspan="4">Additional treatment for episodes resistant to pharmacological treatment</td></tr>
<tr><td colspan="4">Electroconvulsive therapy (B)</td></tr>
</table>

Table 2.2: Treatment algorithm for a manic episode as suggested by the S3-Guideline [16, Algorithm 3].

Therapy of depressive episodes

Due to differences in the manifestation and parthenogenesis of unipolar and bipolar depression and a lack of evidence specific to bipolar depression, the knowledge of effects of antidepressants in bipolar disorder is limited. As there are differences between these two diseases, therapeutic principles of unipolar depression cannot be easily transferred to bipolar depression. Using antidepressants bears the risk of inducing manic episodes, mixed affective states, and of increasing the frequency of episodes. As most patients have experienced previous episodes, pharmacological treatment should have been established already to prevent the recurrence of episodes, which further complicates therapy. The guideline suggests the usage of antidepressants based on the severity of depression and reevaluating the medication once the depressive symptoms subside to reach the goal of remission of symptoms and regaining social function, but also to reduce the risk of the aforementioned induction of manic episodes. Besides medication for the acute treatment of a depressive episode, an evaluation of the medication for recurrence prophylaxis is suggested by measuring blood levels and optimising the dosage. Apart from antidepressants, mood stabilizers and atypical antipsychotics are available for treatment of bipolar depression.

Optimization or beginning of recurrence prophylaxis			
Pharmacological treatment		**Non-pharmacological treatments** (additional to pharmacotherapy)	
Substance	Evidence	Treatment	Evidence
Quetiapine	A	Evidence-based Psychotherapy	A
Lurasidone	B	Sleep deprivation	B
Lurasidone with Lithium or Valproate	B	Repetitive high-frequency Transcranial magnetic stimulation	0
Carbamazepine	0	Light therapy	0
Lamotrigine	0		
Olanzapine	0		
If unsuccessful, change of substance or additional substance			
Additional treatment for episodes resistant to pharmacological treatment			
Electroconvulsive therapy (B)			

Table 2.3: Algorithm for treatment of bipolar depression suggested by the S3-Guideline [16, ALgorithm 4].

As evidence is limited, the guideline suggests a combination of pharmacological and non-pharmacological treatments. Table 2.3 shows the pathway that is based on treatments for which evidence is available. With regard to non-pharmacological options for treatment, different forms of evidence-based psychotherapy (for example cognitive behavioral therapy and family-focused therapy) and other treatments, like electroconvulsive therapy or repetitive high-frequency transcranial magnetic stimulation are recommended [16].

Prophylaxis of recurrence

<table>
<tr><td colspan="2">Pharmacological treatment
(first-line monotherapy)</td><td colspan="2">Additional psychotherapy
(specific types of therapy)</td></tr>
<tr><td>Substance</td><td>Evidence</td><td>Treatment</td><td>Evidence</td></tr>
<tr><td>Lithium</td><td>A</td><td>Psychoeducation</td><td>B</td></tr>
<tr><td>Quetiapine</td><td>B</td><td>Cognitive behavioral therapy</td><td>B</td></tr>
<tr><td>Lamotrigine</td><td>B</td><td>Family-focused therapy</td><td>0</td></tr>
<tr><td>Carbamazepine</td><td>0</td><td rowspan="2">Interpersonal and social rhythm therapy</td><td rowspan="2">0</td></tr>
<tr><td>Valproate</td><td>0</td></tr>
<tr><td>Aripiprazole</td><td>0</td><td rowspan="2">Functional and cognitive remediation therapy</td><td rowspan="2">0</td></tr>
<tr><td>Olanzapine</td><td>0</td></tr>
<tr><td>Paliperidone</td><td>0</td><td></td><td></td></tr>
<tr><td>Risperidone</td><td>0</td><td></td><td></td></tr>
<tr><td colspan="4">If unsuccessful:
selection of another substance OR second-line combination therapy</td></tr>
<tr><td colspan="4">Combinations for second-line therapy</td></tr>
<tr><td colspan="3">Substances</td><td>Evidence</td></tr>
<tr><td colspan="3">Valproate with Quetiapine or Ziprasidone or Lithium</td><td>0</td></tr>
<tr><td colspan="3">Lithium with Quetiapine or Ziprasidone</td><td>0</td></tr>
<tr><td colspan="3">Treatment as usual with Risperidone</td><td>0</td></tr>
<tr><td colspan="4">Additional treatment for episodes resistant to pharmacological treatment</td></tr>
<tr><td colspan="4">Electroconvulsive therapy (B)</td></tr>
</table>

Table 2.4: Algorithm for recurrence prophylaxis as suggested by the S3-Guideline [16, Algorithm 5].

The goal of recurrence prevention is to suppress all episodes of mania or depression, to reduce residual symptoms, and to maintain normal social function. The challenges during

this phase of treatment are to find the optimal drug or combination of drugs to stabilize a patient, and subsequently to ensure a patient's compliance and adherence. Therapeutic drug monitoring should be used to ensure a patient is taking his medication and to guide the dosing for optimal effects. During phases of remission, continuous reevaluation of the treatment should be made. Like the treatment of episodes, recurrence prophylaxis also combines pharmacological and non-pharmacological treatments [16].

2.2 Difficulties in diagnosing bipolar disorder

Several studies found associations of delayed diagnosis with worse outcomes with regard to social function, episodes and hospitalizations and with an increased risk of self-harm [19]. Altamura et al. compared a delay in correct treatment of less than six years with a delay of more than six years and found a higher number of hospitalizations and suicide attempts in the group with the longer delay [20]. Besides suicidality and symptomatic episodes, McCraw et al. found correlations between the delay to treatment with higher rates of unemployment and impaired social function [21].

Both classifications used for the diagnosis of bipolar disorder, ICD-11 and DSM-5, require episodes of depressive mood and episodes of mania or hypomania to fulfil the diagnostic criteria. Especially, hypomania can be difficult to diagnose due to varying interpretations of symptoms by the patient or the clinician. The absence of mania or hypomania can lead to a misdiagnosis of unipolar depression, due to the same definition of a depressive episode used for unipolar depression and bipolar disorder [8, 22]. An analysis of surveys of patients with bipolar disorder by Hirschfeld et al. in 2000 reports a rate of 69% of initial misdiagnosis, with unipolar depression being the most frequent incorrect diagnosis [23]. In a study including 764 patients, Baldessarini et al. found an average delay of 8.8 years from the onset of symptoms to the beginning of pharmacological treatment. When assuming that the correct diagnosis of bipolar disorder is required to start an effective pharmacological treatment, this could be seen as a surrogate for the delay in correct diagnosis.[5]. Using a French population, Drancourt et al. reported a similar finding of a mean duration of untreated bipolar disorder of 9.6 years. When split into subgroups based on the polarity of the first-ever episode, they found an even longer delay of 14.5 and 13 years respectively

for patients with hypomania and depression as their first manifestation, compared with a shorter-than-average time of eight years for patients with mania as the first episode [24]. This highlights possible substantial differences in the diagnostic accuracy caused only by different clinical features presented at the time of assessment, and stresses the further need for reliable and objective diagnostic tools independent of a patient's current affective state.

2.3 Related work

To search for related work, PubMed was searched with the MeSH term "Bipolar and Related Disorders"[Mesh] AND "Artificial Intelligence"[Mesh]. This search on 16th of November 2021 resulted in 139 studies matching these MeSH terms. Of these results, 43 studies tried to classify patients based on data from imaging studies. 8 studies researched if genetic mutations could be identified and used to classify. Using machine learning to classify based on these two areas seems to be a more researched approach than using cognitive variables.

Three studies tried to answer the same question as this thesis, whether it is possible to classify patients with bipolar disorder using machine learning only based on results from cognitive assessments.

Wu et al. [25] used results of 37 scores from Cambridge Neuropsychological Test Automated Battery (CANTAB) [26] to train and evaluate a model. Included were 21 patients with Bipolar disorder I or II which were currently in an euthymic state and 21 matched healthy controls. When using leave-one-out cross-validation, an accuracy of 71%, sensitivity of 76% and specificity of 67% were achieved [25].

Sawalha et al. [27] used results of the CANTAB of patients with bipolar disorder and healthy controls to build a model to differentiate both groups. Two cohorts of patients were defined, one with chronic bipolar disorder, defined as more than two mixed or manic episodes, and a cohort of first-episode bipolar disorder with one manic or mixed episode. A dataset of 129 variables was used, containing results from the CANTAB. They trained a model using the cohort of 74 patients with chronic bipolar disorder and 53 healthy controls with no history of psychiatric or severe somatic disease. Performance metrics for

this model were calculated using 5-fold nested cross-validation with an accuracy of 77%, precision of 82% and resulting sensitivity of 76% and specificity of 77%. This model was evaluated using the second cohort of 37 patients with first-episode bipolar disorder and 18 healthy controls. Results for the second cohort were an accuracy of 76%, precision of 87%, sensitivity of 75% and specificity of 79%. The approach to use chronic and first-episode bipolar patients as the train and test data set was used to determine whether a model can differentiate between patients and healthy controls and if it is possible to identify patients who have their first episode as reliably as patients with a longer history of the disease. The authors concluded that this approach could help in early recognition of bipolar disorder [27].

A study by Sonkurt et al. used a subset of six cognitive evaluation tests from CANTAB to classify Bipolar I patients and healthy controls. A sample of 17 patients and 19 controls was used. The group performed 10-fold cross-validation and reported an F1 score of 0.8, accuracy of 0.78 and AUROC of 0.78 for the correct classification as Bipolar I with a sensitivity of 80% and specificity of 76.2% [28].

These three studies are the most closely related to the research question of this thesis. The results indicate that it is possible to classify patients as either bipolar or healthy with an accuracy between 0.71 and 0.78 when using only results from cognitive assessments.

A study by Fernandes et al. used immunological and inflammatory biomarkers in combination with cognitive data to classify patients with bipolar disorder, schizophrenia and healthy controls. Three classifiers were built for each combination of groups and trained, once with only the biomarkers and once with biomarkers and cognitive data. The relevant model of bipolar against control with cognitive data achieved an accuracy of 79.73%, sensitivity of 88.29% and specificity of 71.11%. Results from the Wechsler Adult Intelligence Scale (WAIS) [30], National Adult Reading Test [31] and the California Verbal Learning Test (CVLT) were used. The CVLT is also used as a test for cognitive performance in the "BIPFAT" study [29].

Walsh-Messinger et al. [32] tried to identify which variables are of importance when trying to distinguish between patients with schizophrenia or schizoaffective disorder and patients with affective disorders, including bipolar disorder, or healthy controls. They used Global Assessment of Function [33], the Positive and Negative Syndrome Scale [34] and four

scores from the WAIS as assessments of cognitive function and the Diagnostic Interview for Genetic Studies [35] for assessment of psychiatric symptoms as well as parental age at birth. Their experiments yielded similar results when separating patients from healthy controls and schizophrenia from affective disorders. When analyzing their models, they found that variables like the Global Assessment of Function or positive or negative symptoms had a tendency to be more influential than cognitive assessment [32].

In a Chinese study by Ma et al. [36], feature selection was used in an attempt to simplify the correct diagnosis of bipolar disorder and the distinction from major depressive disorder. They retrospectively used a cohort of patients with bipolar or affective disorders and healthy controls to train different machine learning models. Using feature selection, 17 questions were selected from a structured questionnaire containing originally 113 clinically relevant questions. Using this subset of questions, the new "Bipolar Diagnosis Checklist in Chinese" was created, which has an equal performance when diagnosing bipolar disorder as the original. The authors concluded that machine learning and feature selection can be used to shorten complex diagnostic interviews, which can result in significant time savings when used in clinical practice [36].

Chapter 3

Methods

3.1 Methods

3.1.1 Overview of machine learning

Machine learning, a term popularized by Arthur Samuel in 1959 [37], covers various algorithms and techniques to learn from and predict on data. There are three basic types of machine learning approaches, differentiated by their way of learning from data.

Supervised learning
Supervised learning is used on data where each sample has a respective label which is known at the time of training. An algorithm is trained using training data and afterwards predicts the labels of previously unseen test data.

Unsupervised learning
In unsupervised learning, labels of the data are unknown. This approach is used to infer new information about the data by using techniques such as clustering or dimensionality reduction.

Reinforcement learning
In reinforcement learning, an iterative approach is used in which an algorithm learns by

either receiving or not receiving a reward for correct predictions and optimizing itself to get the reward more often [38].

Depending on the available data and the questions posed, different methods from these three groups can be used. In this thesis, mainly supervised learning is used to evaluate and predict on the data. Unsupervised learning is used to a certain extent for visualizations.

3.1.2 Terminology

The following section provides a brief overview of the terms used in this thesis.

Samples, Features, Labels
A sample represents one element of the data set and consists of multiple features (also called attributes or variables) and a respective label (also target). In this thesis, a patient represents a sample with the demographic data and test results as the features and the group, either "bipolar" or "control", as the label [38].

Algorithm
An algorithm describes the underlying method of how a model is implemented and trained. Algorithms are, for example, "Logistic Regression" or "Support Vector Machine". The settings of an algorithm, e.g. which kernel to use or a learning rate, are called hyperparameters [38].

Model
A model represents an algorithm which was constructed with a set of specific hyperparameters after training. During training, the model's parameters are learned from the training data; therefore, a models consists of an algorithm, the predefined hyperparameters and the model's parameters. A model can also be called classifier or estimator [38].

3.1.3 Performance metrics

To evaluate and compare the performance of machine learning models, various performance metrics can be calculated. As the data set only includes two groups of patients, "bipolar" and "healthy control", all classifications are made for this binary case and therefore only

metrics for binary classifications are used. When unspecified, these metrics are calculated for the positive label. For this thesis, the positive label is always "bipolar". The following metrics are used in this thesis:

Accuracy
Ratio of correct predictions to the total number of samples [38].

$$Accuracy(ACC) = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative}$$

Precision
Ratio of correct positive predictions to the total predicted positives. Precision is also called "positive predictive value" or short "PPV" [38].

$$Precision\ (P) = \frac{TruePositive}{TruePositve + FalsePositive}$$

Recall
Ratio of correct positive predictions to the total positive samples. In a binary classification task, this is also called "sensitivity" [38].

$$Recall\ (R) = \frac{TruePositive}{TruePositive + FalseNegative}$$

F1 score
Harmonic mean of precision and recall [38].

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

Receiver operating characteristic curve
The receiver operating characteristic (ROC) curve is a plot of the true positive rate, which is the recall or sensitivity, against the false positive rate, which equals $1 -$ specificity using different thresholds for the decision function. The area under this curve (AUC or AUROC) can be used for easy comparison of different models. A perfect model would have an AUROC of 1, whereas a purely random model would have 0.5 [39].

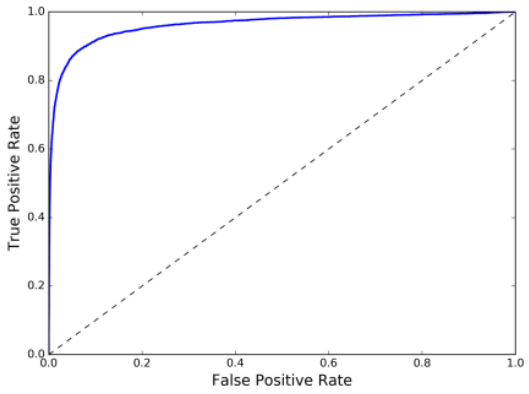


Figure 3.1: Example of a ROC curve. AUROC [39, Figure 3-6].

Averaging

Averaging is typically used in multi-class or multi-label classifications to better assess the overall performance of a model and not only the classification performance for a single class or label. In the binary classifications made in this thesis, using only the metrics calculated for the positive label "biploar" could result in biased results due to better or worse performance for one of the two classes. Therefore, it was specified prior to analysis to evaluate algorithms and models using averaging strategies to assess the performance for both classes at once and to mitigate possible bias [38].

Micro-average

A micro-average is calculated by adding each numerator and each denominator for all scores and dividing them. A micro-average weights each prediction equally [38].

$$score_{micro} = \frac{numerator_{score_1} + numerator_{score_2} + \cdots + numerator_{score_k}}{denominator_{score_1} + denominator_{score_2} + \cdots + denominator_{score_k}}$$

Macro-average

The arithmetic mean of all scores. A macro-average weights each class equally [38].

$$score_{macro} = \frac{score_1 + score_2 + \cdots + score_k}{k}$$

3.1.4 Techniques for training and optimizing models

The performance of a machine learning model is always measured when predicting on previously unseen data. This necessitates the usage of distinct subsets of the data set for training and testing the model. The performance of a model depends on which samples are part of the training and test split, as small differences of these subsets of the complete data set can influence how a model learns to predict. Therefore, performance measured when using such a strategy is always only valid for this distinct split. To estimate how a model performs when learning on the complete data set, multiple runs of training and measuring performance with different splits of data are required, and the average values can be used as estimates for the true performance of the model. This true performance, which cannot be measured directly, is called the generalization performance of a model. The same applies for the error rate of a model; the estimated true error rate is called the generalization error. Strategies for measuring performance are outlined in the following section.

Holdout validation

The easiest method is holdout validation. The data set is split in a subset of training and test data; in the most common cases, an 80/20 train-test-split is used. If optimization strategies, like grid search, are performed, the data set can be split into three subsets: a training set, a validation set used for optimizing the model, and a test set to measure the performance of the optimized model. This prevents information leakage during training, but on the downside, it reduces the overall amount of data available for training [39, p. 29f]. Usually, these splits are generated using random subsets of the data. A model's performance can vary significantly depending on the randomness of the selected data in the respective splits [38].

Cross-validation

Cross-validation means splitting the data set into k subsets or folds, and using k-1 of these folds as training data and the remaining fold as test data. Typical numbers for k are 5 or 10. This can be used for evaluating a model's performance by calculating selected metrics

k-times and averaging the results for a better estimate of a model's generalization error. [38] Cross-validation can also be used in combination with grid search.

Grid search

To optimize a model's performance, its hyperparameters can be tuned for better results. Grid search is a technique for this process with which different settings for hyperparameters are defined prior to training and a model using all possible combinations of these hyperparameters is trained and evaluated. The models are either trained and tested using holdout validation or cross-validation. After every combination has been tested, the best performing model and its hyperparameters are used for further experiments. [38]

Nested cross-validation

To compare and evaluate different algorithms and models, their generalization error must be calculated as precisely as possible. Usually measuring a model's performance is done by using holdout validation, optimizing a model for the training data using grid search and then evaluating the model using the test data. Due to the randomness in the training/test splits, the measured generalization error can vary significantly. A way to mitigate this problem is by combining grid search for optimization of the model with cross-validation for evaluation of the generalization error. This is called $k \times n$ nested cross-validation. The complete data set is split into k-folds of $k-1$ parts to be used as training data and the rest as test data. For each of these folds, a n-fold cross-validated grid search is performed to select an optimized model for which the performance metrics are calculated. The average of all k calculated metrics is used as the final score of the algorithm. This results in an almost unbiased estimation of generalization performance [40]. In this thesis, 10×5 nested cross-validation is used to evaluate and compare different algorithms. Figure 3.2 shows the process of nested cross-validation.

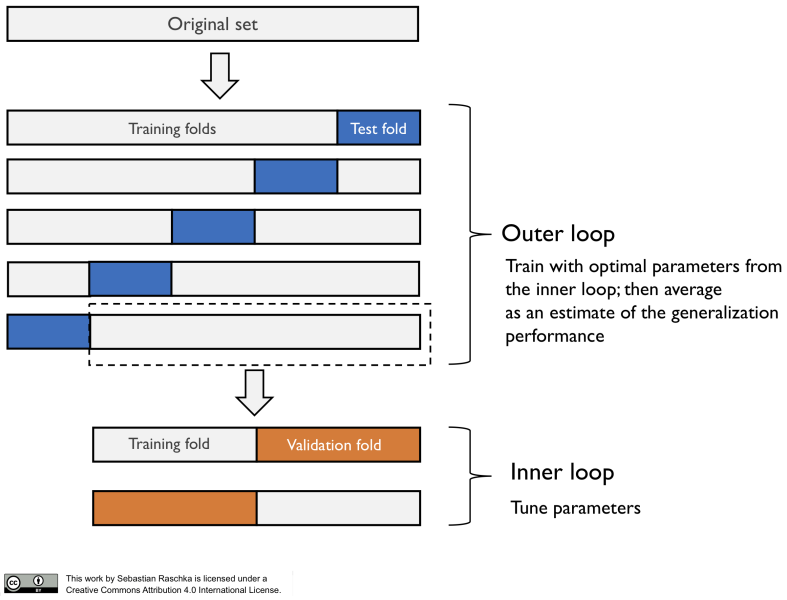


Figure 3.2: Nested cross-validation using 5×2 nested CV [41]

3.1.5 Machine learning algorithms

There are different algorithms available for building and training machine learning models. The following algorithms are used in this thesis as they are commonly used algorithms and are implemented in scikit-learn [42]. Other algorithms could be used as well, but testing additional algorithms was not included in the scope of this thesis, as the goal is to focus on well-accepted and tested methods to gain an idea of how these methods perform when applied to the "BIPFAT" data set.

Logistic Regression

Using a logistic function for binary classification was first proposed by Cox [43] in 1958. Logistic Regression estimates a probability by assigning weights to every feature and then calculates the weighted sum of all features. The classification is done by applying this probability to a sigmoid function, also called the logistic or logit function, and deciding if a sample is positive or negative based on a cut-off. Training is done by finding the ideal cut-off where the rate of correct predictions is the highest [39].

Support Vector Machine

Support vector machines were invented in 1974 by Vapnik and Chervonenkis [44]. When assuming all n features of a data set are located in a higher space with n dimensions, a support vector machine tries to construct an imaginary plane called hyperplane, which, in the case of binary classification, separates these two classes. This plane is optimized to have the maximum distance from the closest samples on both of its sides, which are called the support vectors. In figure 3.3, an example is displayed with maximum margins on both sides of the hyperplane. The encircled samples are the support vectors which define the hyperplane. More complex shapes of hyperplanes can be used for more complex problems [38].

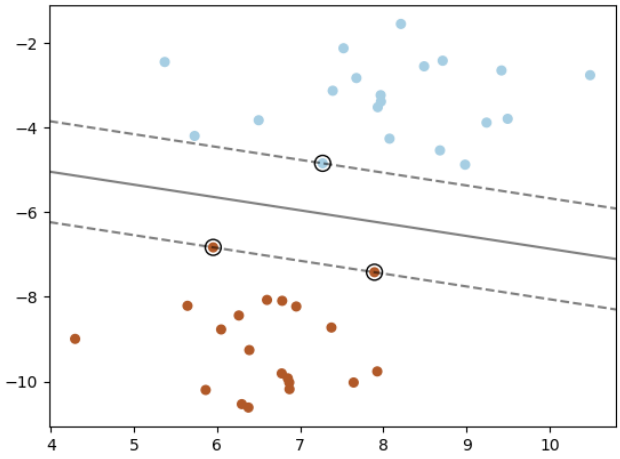


Figure 3.3: Visual representation of a support vector machine for a linearly separable problem from the official scikit-learn API documentation [45]

AdaBoost

Adaptive Boosting, in short AdaBoost, is an ensemble method which means that it combines multiple models with poorer performance into a single model which performs better. The technique of boosting was introduced by Freund and Schapire [46] in 1997. AdaBoost trains a model, in the case of this thesis a decision tree, and afterwards the algorithm tries to improve the model's performance by adjusting the weights of incorrect classifica-

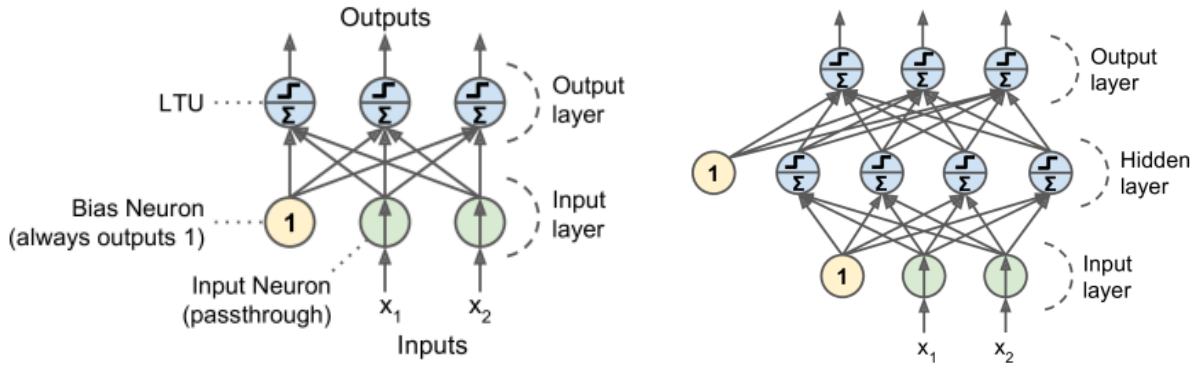
tions to predict them correctly in the future. Using the updated weights, a new model is trained, and the process is repeated a set amount of times. This approach should optimize the predictive performance for difficult samples and therefore increase the overall model's performance [39].

Random Forest

Random Forest is another ensemble method, first described by Breiman [47] in 2001. A random forest consists of a set amount of decision trees which are trained with random subsets of the data set. To make a final decision, all the classifications made by each tree are combined, either by majority vote or, as in the case of scikit-learn, by combining their predicted probabilities [39].

Artificial Neural Network

Nowadays, a variety of different neural networks exists, but the general idea of an artificial neural network goes back to a paper by Mccullock and Pitts [48] from 1943. The form of neural network used in this thesis is the so-called Multi-Layer Perceptron (MLP). This network is based on the architecture for artificial neural networks called Perceptron, which was invented by Rosenblatt [49] in 1957. A perceptron consists of a single layer of Linear Threshold Units (LTUs), also called neurons, which calculate an output as the sum of all inputs, where every input is multiplied with a weight factor, and applying it to a decision function. An LTU basically works like Logistic Regression. The perceptron consists of as many LTU's as there are a variable amount of outputs required, for example two in the case of binary classification, and a variable amount of inputs, which are all connected to every one of the output layers. An Multi-Layer Perceptron adds a variable number of so-called hidden layers between the input and output layer. During training, the weight of each input is adjusted to increase the performance of the network. In figure 3.4, visualizations of a perceptron and an MLP are shown [39]. Scikit-learn only provides an MLP as neural net. Other forms of more complex neural nets could also be used for the task of this thesis and might deliver better results.



(a) Perceptron

(b) Multi-Layer Perceptron

Figure 3.4: Visualizations of a Perceptron (a) with three output LTU's/neurons and as Multi-Layer Perceptron (b) with a single hidden layer of four LTU's/neurons [39, Figures 10-5 and 10-7].

3.1.6 Development environment

scikit-learn, which is a free, open source library for developing machine learning applications, was used to develop all models presented in this thesis. scikit-learn is implemented in the programming language Python and mainly used for scientific projects [42]. The version used was 0.23.2.

All calculations were performed using JupyterLab 3.0.5, a free open source, browser-based environment for data science which provides a graphical interface for a Python interpreter and functions resembling an Integrated Development Environment [50]. This environment is run under macOS 10.14.6. The Python version used was 3.8.2. Descriptive statistics were calculated, and all data was handled with Pandas 1.0.3 and NumPy 1.18.2. For diagrams and plots, Seaborn 0.10.0 and Matplotlib 3.2.1.

3.1.7 BIPFAT study and data set

BIPFAT study

The single-center "BIPFAT" study was conducted by the Medical University of Graz's Department of Psychiatry and Psychotherapeutic Medicine and assessed demographic parameters, complete current and lifetime psychiatric history using the Structured Clinical Interview according to DSM-IV (SCID I), the psychiatric rating scales Hamilton-Depression (HAM-D), Young Mania Rating Scale (YMRS) and Beck's Depression Inventory (BDI), history of medication, anthropometric measure, blood pressure, fasting blood, cognitive testing, EEG, stool sample, different lifestyle questionnaires and magnetic resonance imaging (MRI) of the brain.

All patients included were former in- or outpatients of the Medical University of Graz and had a diagnosis of BD I or BD II according to the DSM-IV criteria. Patients needed to be in the state of euthymia (HAM-D score <11 and YMRS <9) and had given written informed consent prior to participating in the study. The study has been approved by the local ethics committee (Medical University of Graz, Austria) in compliance with the current revision of the Declaration of Helsinki, ICH guideline for Good Clinical Practice and current regulations (EK-number: 24-123 ex 11/12).

Exclusion criteria were the presence of chronic obstructive pulmonary disease, rheumatoid arthritis, systemic lupus erythematosus, inflammatory bowel disease, neurodegenerative and neuroinflammatory disorders (i.e. Alzheimer's, Huntington's and Parkinson's disorder, multiple sclerosis), hemodialysis and interferon-α-based immunotherapy. Further exclusion criteria for controls were the presence of lifetime psychiatric diagnoses (verified by SCID I) and first- and second-grade relationship to relatives with psychiatric disorders [51].

BIPFAT data set

The data were selected prior to analysis to have a near equal distribution of both sexes from the complete "'BIPFAT" data set. The used data set contains 341 samples of two groups, "'biploar" and "control". Each sample consists of 18 features, where one is the

group ID and one is a de-identified patient ID from the original data set. The patient ID is not used for analysis, as it is only an index. The data set contains basic patient information and scores from various cognitive tests performed during the study.

Trail Making Test (TMT)
Trail Making Test (TMT) is a cognitive test done in two parts. In the first part, TMT-A, participants must connect circles containing numbers in their ascending order. During the second part, TMT-B, the numbers are mixed with letters and also have to be connected in ascending order by alternating between numbers and letters. For both parts, the time to completion is measured [52].

Stroop test
The Stroop effect was first described by Stroop [53] and is implemented in the Stroop test. It measures a patient's ability to inhibit cognitive interference while simultaneously processing incongruent stimuli by measuring reaction time [54].

California Verbal Learning Test (CVLT)
The California Verbal Learning Test (CVLT) is a test to estimate a patient's verbal learning and memory capability by having the test subject remember a list of 16 words and measuring how many of these can be recalled after some time and after completing an interference task in between [55].

d2 and d2-R test
The d2 test and its revised version d2-R were developed by Rolf Brickenkamp in 1962. The test evaluates a participant's cognitive performance by measuring the errors made when crossing out certain combinations of letters and symbols and the number of processed symbols in a given time [56].

Feature	Description
Pat_ID	Continuous number, omitted before analysis
age	Patient age
sex	Sex, encoded as 1 = female, 2 = male
group	Group, encoded as 1 = bipolar, 2 = control

BMI	Body Mass Index
TMT_A	TMT Part A – time to completion
TMT_B	TMT Part B – time to completion
STROOP_FWL	Stroop Test Reading Color Names
STROOP_FSB	Stroop Test Naming Colors
STROOP_INT	Stroop Test Interference
CVLT_A_LS_COR	CVLT Learning sum of round 1 to 5
CVLT_DG1	CVLT No. of correct words round 1
CVLT_DG5	CVLT No. of correct words round 5
CVLT_VFW1	CVLT Delayed free recall 1
CVLT_WA1	CVLT Cued recall 1
CVLT_VFW2	CVLT Delayed free recall 2
CVLT_WA2	CVLT Cued recall 2
vereint_KL	combined d2 and d2-R "Konzentrationsleistung"

Table 3.1: Features of the data set and their description.

3.2 Approach

3.2.1 Data preprocessing

Before machine learning models can be trained in a meaningful way, the data used for training and evaluating the models must be checked and, if necessary, modified to avoid invalid results. The first step performed is imputation of missing data by filling missing values of features with the mean value of all other samples. An exception is the feature "sex", missing values are imputed with random 1 or 2, as imputation with the mean value would result in a third group with a float value between 1 and 2.

The second step is feature scaling. Some algorithms perform worse if the features have varying orders of magnitude. The technique used is standard scaling, meaning all features are standardized in a way so that their mean value approximates to 0 and their standard

deviation to 1. Both steps of preprocessing were performed using functions provided by scikit-learn.

3.2.2 Dimensionality reduction and cluster analysis

Prior to training machine learning models by using approaches of supervised learning, different dimensionality reduction techniques are used in this thesis to visualize the data. During this process, data is projected into a 2-dimensional space to find possible clusters of samples or to identify other structures in the data. Principal Component Analysis (PCA), a technique described in 1901 by Pearson [57], is used to visualize the data on a 2-dimensional plot and to calculate the data's principal components, which contain information about the data set and its features. t-Distributed Stochastic Neighbor Embedding (t-SNE), a newer technique described by Maaten and Hinton in 2008 [58], is used to search for clusters in the data. Both methods give visual hints of structures in the data set.

3.2.3 Algorithm evaluation and selection

Machine learning models can be built using different algorithms, for example Support Vector Machine (SVM) or Random Forest. Therefore, comparing and selecting the best performing algorithm to be used for the final model and further experimentation is a necessary step when building these models.

To determine which algorithm has the best performance, the F1 score is used as an evaluation metric for comparison. Micro- and macro-average F1 scores of both labels are presented as results during this thesis. 10×5 nested cross-validation is used to calculate the performance metrics for all selected algorithms. The results are compared and the algorithm with the best micro-average and macro-average F1 scores is selected as the best performing algorithm and used for further analysis.

For all algorithms and for the function used to generate the splits of the data set, a seed can be set to control the randomness in these processes. This seed can be set to an integer value or to an unknown, random value. For this thesis, the seed is set to 0 for each model and for the splits. Using a fixed seed leads to reproducible results, but comes at the cost

of reduced information gained about the generalization performance of these models due to omitting the model's inherent randomness.

3.2.4 Model optimization

Nested cross-validation can be used for estimating an algorithm's generalization performance, but due to multiple runs of optimization using grid search, no final best performing model can be generated this way. After algorithm selection, this step has to be done again using the best performing algorithm. To further demonstrate the increase in performance by hyperparameter tuning, the F1 score of a model only using the default hyperparameters is compared to an optimized model. The comparison is done by using an 80/20 train-test-split for scoring. The tuned model is optimized by performing a grid search with cross-validation using the same parameter grid as during algorithm selection. Due to the usage of holdout validation for comparing these models, the comparability of these results to results when using other splits of the data set is limited.

A second, experimental approach to inspect a model's performance is to reduce the number of features used for training by selecting features based on their importance. This can be done by manually setting a threshold feature importance to include a feature or not, or by using an algorithm like Recursive Feature Elimination with Cross-Validation (RFECV). RFECV is implemented by scikit-learn to gradually select fewer features and scoring the model's performance using cross-validation. The previously optimized model is used to perform this task. The features are reduced one by one and the micro-average F1 score is used for scoring. A ranking of the features is computed by the algorithm, and a plot of the number of features and their scores is presented. The algorithm also returns the optimal number of features, resulting in the highest score. In the presented feature ranking, all the features selected to obtain the best score are given the rank 1 and the remaining features are ranked in descending order of their importance [59]. An analysis of the selected features is not done when using RFECV, as this is performed only as an optimization task.

3.2.5 Feature analysis

Feature analysis is done by using the algorithm with the best F1 score selected earlier. A model of this algorithm is trained and optimized by performing a grid search for its best hyperparameters. The macro-average F1 score is used as the target for this optimization. For this model, the internal ranking of features is displayed. This can provide information which features of the data set are the most important and which can be possibly ignored when training the model.

A second step is the usage of the library "ELI5" to gain insight into the model. This library is used to "explain" a machine learning model by displaying a prediction as a table of each feature's contribution, where the sum of all contributions is the result. The contribution of a feature is calculated by multiplying the feature's value with its coefficient [60]. For this step, the optimized model is trained using an 80/20 train-test-split. The first two samples from the test split are selected, and their predictions displayed using "ELI5".

3.2.6 Final model

For possible future applications or to predict on data which is not included in the used data set, a final model needs to be generated. The final model is generated by selecting the algorithm with the best performance, searching for the best hyperparameters using grid search with cross-validation and, as a final step, training the model with the optimized hyperparameters using the complete data set as training data. As this is not within the scope of this thesis, the generation of a final model is not included in the notebook appended in Appendix A.

Chapter 4

Results

4.1 Properties of the data set

The data set includes data from 341 patients of the observational "BIPFAT" study, conducted by the Medical University of Graz's Department of Psychiatry and Psychotherapeutic Medicine, as detailed in Section 3.1.7.

The data set is made up of two groups, 196 patients with a confirmed diagnosis of bipolar disorder and 145 healthy patients as the control group. Both groups were selected to contain an almost equal number of male and female samples. The data set contains 16 features for each of the samples and their respective group as target variable. The 16 features consist of basic patient data (age, sex, BMI) and the results of different cognitive tests performed during the study. The following table gives an overview of all features and their values, stratified by their group.

Feature	All (N = 341)	Bipolar (N = 196)	Control (N = 145)	Missing values
Age – yr	41.2 ± 14.8	44.2 ± 13.7	37.1 ± 15.3	5
Sex				5
Male – no. (%)	158 (47.0)	107 (54.9)	90 (63.8)	

Female – no. (%)	178 (53.0)	88 (55.1)	51 (36.2)	
BMI	26.4 ± 5.8	27.9 ± 6.1	24.4 ± 4.5	5
TMT_A	32.0 ± 13.8	36.0 ± 14.4	26.5 ± 10.6	0
TMT_B	73.5 ± 39.7	84.2 ± 45.6	59.3 ± 23.5	3
STROOP_FWL	30.7 ± 5.9	32.0 ± 6.1	28.9 ± 5.2	7
STROOP_FSB	47.2 ± 8.9	49.3 ± 8.8	44.4 ± 8.2	7
STROOP_INT	75.9 ± 20.1	82.4 ± 22.4	67.4 ± 12.0	7
CVLT_A_LS_COR	54.8 ± 12.3	51.3 ± 12.3	59.0 ± 11.0	102
CVLT_DG1	7.8 ± 5.5	6.9 ± 2.5	8.9 ± 7.8	4
CVLT_DG5	13.2 ± 2.7	12.4 ± 2.9	14.2 ± 2.1	4
CVLT_VFW1	11.5 ± 3.4	10.4 ± 3.4	12.9 ± 2.8	7
CVLT_WA1	12.4 ± 5.1	11.2 ± 3.2	13.9 ± 6.5	7
CVLT_VFW2	12.1 ± 3.5	11.2 ± 3.5	13.3 ± 3.0	8
CVLT_WA2	12.5 ± 3.2	11.6 ± 3.3	13.6 ± 2.7	8
vereint_KL	160.2 ± 52.7	139.8 ± 44.8	188.1 ± 50.0	19

Table 4.1: Characteristics of the "BIPFAT" data set. Values are means ± SD.

4.2 Dimensionality reduction

PCA

A 2 component PCA was performed, and both resulting principal components are visualized on a 2-D plot. No clusters in the data or a clear separation of both groups can be observed.

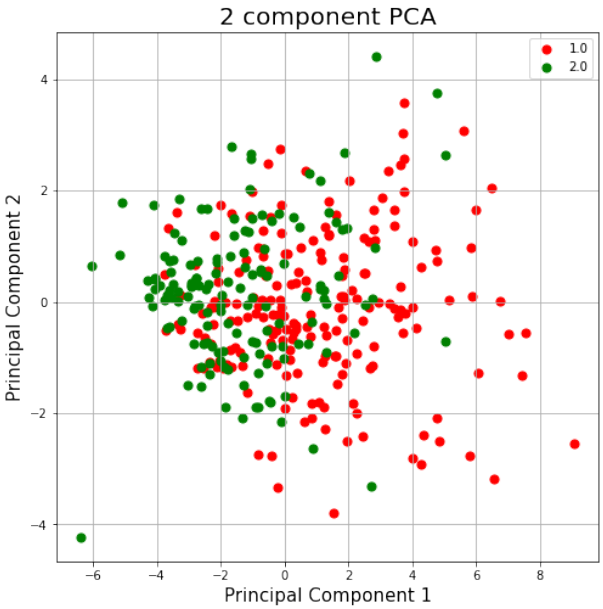


Figure 4.1: 2-D plot of both principal components of a 2 component PCA.

A 16 component PCA was performed to generate a feature ranking based on the most important feature for each of the 16 principal components. Table 4.2 shows the most important feature for each principal component and the principal component's explained variance ratio. Some features are ranked more than once.

PC	Feature	Variance ratio
0	CVLT_VFW1	0.436010
1	STROOP_FSB	0.108661
2	sex	0.071194
3	BMI	0.064334
4	CVLT_DG1	0.055407
5	BMI	0.048681
6	CVLT_WA1	0.037827
7	CVLT_WA1	0.035407
8	vereint_KL	0.032666
9	STROOP_INT	0.025062
10	TMT_B	0.024625

11	CVLT_A_LS_COR	0.020803
12	STROOP_FSB	0.014341
13	CVLT_DG5	0.010996
14	CVLT_VFW1	0.008884
15	CVLT_VFW2	0.005102

Table 4.2: Principal components (PC) and their most important features with corresponding ratio of explained variance.

t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) was done for dimensionality reduction as described in section 3.2.2. As settings for the algorithm, the number of iterations was set to 1000 and the random state to 0. Three different values for the parameter "perplexity", which influences the shape of the resulting plot, were used to identify clusters in the data. No clusters can be identified in the resulting plots, and no clear separation between both groups can be seen.

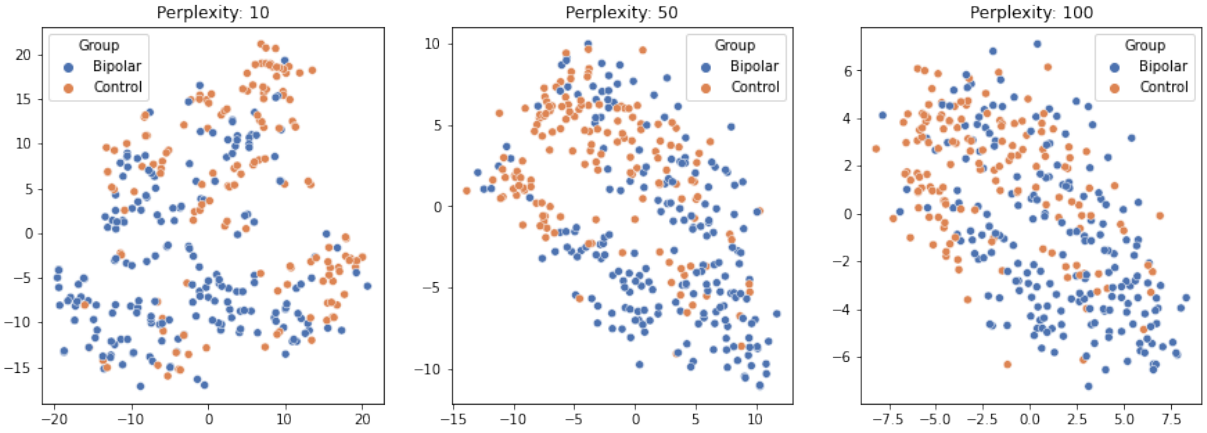


Figure 4.2: t-SNE plots using different values of perplexity.

Neither the PCA nor the t-SNE plots show visible clusters or a clear separation of both groups. Both plots show areas where one group is dominant, but also areas where both groups overlap and outliers can be found for both groups in all areas. Based purely on these observations, classification of the data set using machine learning might not yield high scores.

4.3 Evaluation of machine learning algorithms

10 × 5 nested cross-validation was performed to calculate the F1 scores as described in Section 3.2.3 for the five selected algorithms.

Algorithm	F1 macro-average	F1 micro-average
Logistic Regression	0.69 ± 0.06 [0.66-0.73]	0.70 ± 0.06 [0.67-0.74]
AdaBoost	0.67 ± 0.05 [0.64-0.69]	0.69 ± 0.05 [0.66-0.72]
Support Vector Machine	0.68 ± 0.06 [0.64-0.72]	0.69 ± 0.07 [0.65-0.73]
Random Forest	0.68 ± 0.06 [0.64-0.71]	0.69 ± 0.06 [0.65-0.73]
Neural Network	0.68 ± 0.07 [0.64-0.73]	0.70 ± 0.08 [0.65-0.74]

Table 4.3: Results of 10 × 5 nested cross-validation, sorted by best F1 macro-average score. F1 score ± STD [95% CI].

The scores are the average results for 10 runs of cross-validation. Typically, performance metrics are calculated with a precision of two decimal places. Logistic Regression has the highest score using macro-average as averaging strategy with a score of 0.69. Using micro-averaging, Logistic Regression and the Neural Network perform equally with a score of 0.70, with Logistic Regression having the smaller standard deviation of 0.06 and the narrowest 95% confidence interval of [0.67-0.74]. Due to this marginal difference, Logistic Regression can be selected as the best performing algorithm. The differences in scores between the best and the worst algorithm are 0.02 for macro-averaging and 0.01 for micro-averaging, which indicates only marginal differences in performance with unclear implications for practical usage. The 95% confidence intervals for all results overlap, which

indicates that there are no statistically significant differences in the performances of the algorithms; therefore, no statistical hypothesis tests were performed.

A limitation of these results is that random seeds used for models and splits influence model performance; therefore, these results and the ranking of algorithms are only valid for the used seed 0. As nested cross-validation combines model optimization by grid search and cross-validation, it generates almost unbiased results and for this reason, these results are a good approximation of an algorithm's true generalization performance [40]. When assuming that the average results of the performed nested cross-validation are a close representation of the true generalization error, it can be further assumed that using different seeds is not going to generate better results when using the same data. To examine if different seeds have an impact on these results, multiple runs of the program would have been necessary. Due to the long runtime required to perform a complete cycle of nested cross-validation and only minimal expected differences, this has not been done in this thesis.

These five different algorithms are tested for their classification performance, and all of them yield almost equal results. Considering these are well accepted algorithms and their implementations in scikit-learn are well optimized, it can be assumed that further testing of different algorithms is not going to result in significantly different performances. Dimensionality reduction using PCA and t-SNE showed no visible clustering and in large parts overlapping distributions of samples, which is consistent with the finding of a mediocre macro-average F1 score of 0.69 for the top algorithm. These observations lead to the conclusion that the available data does not contain enough information or clearly distinguishable data to obtain better results when making predictions using machine learning.

4.3.1 Consideration of training runtime

The runtime of training a machine learning model is mainly dependent on the size of the data set and the complexity of the algorithm used. For a small data set, like the "BIPFAT" data set used for this thesis, the difference in runtime when training a single instance of an algorithm can be neglected. When using grid search or nested cross-validation, the time needed for training is multiplied by the size of the hyperparameter grid and the number of cross-validation folds.

The runtimes of a grid search with cross-validation and $k \times n$ nested cross-validation can be calculated using the following formulas, where $Time_{model}$ is the runtime of training a single instance and $Grid\ Size$ the number of possible combinations of a parameter grid.

$$Runtime_{\text{Grid Search with k-fold CV}} = Time_{model} \times Grid\ Size \times k{-}folds \tag{4.1}$$

$$Runtime_{k \times n \text{ Nested CV}} = Time_{model} \times Grid\ Size \times n{-}inner\ folds \times k{-}outer\ folds \tag{4.2}$$

Algorithm	Grid Size	Nr. of models	Runtime
Logistic Regression	450	22500	30.1 seconds
Support Vector Machine	42135	2106750	102.2 minutes
AdaBoost	40	2000	35.7 minutes
Random Forest	3168	158400	163.8 minutes
Neuronal Network	160	8000	24.4 minutes

Table 4.4: No. of models trained during algorithm evaluation and total runtimes per algorithm. Runtimes were measured while doing a test during development and are not equal to the times shown in the notebook in Appendix A.

The hyperparameter grids used can be found in Jupyter Notebook. Grid sizes vary depending on the complexity of the algorithm and their possible settings. The runtimes were measured on a MacBook Pro Retina Mid 2012 running macOS 10.14.6 using 8 logical threads on an i7-3615QM CPU. scikit-learn has no native GPU support. The values for the runtimes presented in table 4.4 were measured during a test run of the nested cross-validation. They might vary between runs of the same program, and are therefore to be seen as approximations to get a general idea of computing requirements.

As the results of the algorithm selection has not shown a clear algorithm with significantly better performance than the others, the computing power needed to achieve these results

can be considered as an additional factor when evaluating performances, especially in resource-limited settings.

Logistic Regression has a drastically shorter runtime when performing 10×5 nested cross-validation, and also the best macro-average and second-best micro-average F1 score. Due to no significant differences in performance when training more complex algorithms and requiring way less computing power, Logistic Regression can be regarded as the best performing algorithm on this data set.

4.4 Model optimization

Logistic Regression is used as a model for further optimization, as it showed high performance metrics at a low computational cost.

4.4.1 Grid search

Due to the k-fold repetition of cross-validation when using nested cross-validation, k best models are created in this process. For this reason, it is necessary to build the best model from the beginning after selecting an algorithm by performing a grid search with cross-validation. To compare the base model and the optimized model, an 80/20 train-test-split is done prior to grid search to evaluate the model on data unseen during training. Micro- and macro-average F1 scores are calculated for both models. As a target score for the grid search, the macro-average F1 score is used to select the model with the best macro-average F1 score as best model. Both models and the function for the train-test-split are initialized with seed=0 for reproducible results. No hyperparameters are set for the base model, so only the default settings are used. For the grid search, the same hyperparameter grid is used as during nested cross-validation. The results are shown in Table 4.5 and for visual comparison, the ROC curves for both models are shown with their respective AUROC values.

Model	macro-average F1	micro-average F1

Base model	0.73	0.75
Optimized model	0.75	0.77

Table 4.5: Performances of models before and after hyperparameter tuning for the used train-test-split.

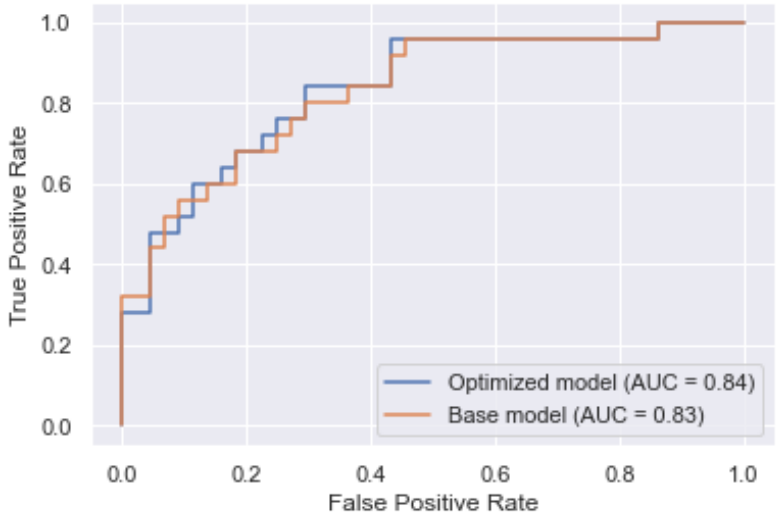


Figure 4.3: ROC curves of the base and optimized model.

Only a minimal increase in performance can be achieved by hyperparameter tuning using grid search with cross-validation. The presented results are only valid when using seed=0 for the train-test-split and model initialization. Using other seeds can result in different results, as a model's performance depends on the selected test data. The comparison of these scores to the scores measured by k-fold cross-validation is limited due to the train-test-split. For a better estimation, optimization by grid search needs to be repeated multiple times and its average results would need to be calculated, which would be the same process as k-fold cross-validation and would not yield an optimized model as a model for further usage, which is the distinct goal of this step.

The inability to achieve higher scores when optimizing the models is a further indicator that the used data set does not enable models to perform better due to samples that cannot be clearly separated into the two classes.

4.4.2 Feature elimination

Recursive Feature Elimination with Cross-Validation (RFECV) was performed to investigate whether a reduction in features results in a better performance. A fresh Logistic Regression model without prior optimization was used as algorithm. Macro-average F1 score was selected as scoring parameter. The highest macro-average F1 score of 0.70 was achieved by using the 6 features listed with Rank 1 in Table 4.6.

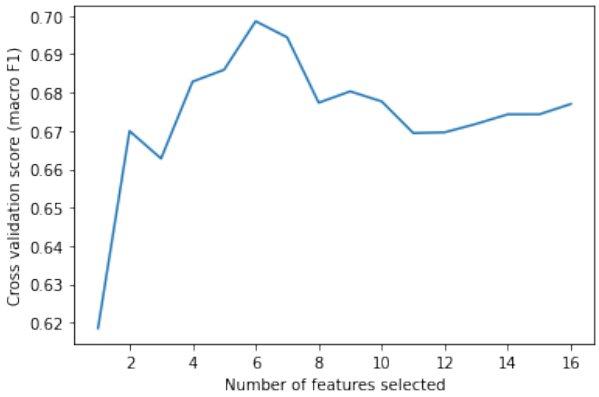


Figure 4.4: Graph of number of selected features and cross-validated macro F1 score.

Rank	Feature	Rank	Feature
1	BMI	4	age
1	TMT_A	5	sex
1	STROOP_INT	6	STROOP_FSB
1	CVLT_WA1	7	STROOP_FWL
1	CVLT_WA2	8	CVLT_A_LS_COR
1	vereint_KL	9	CVLT_DG1
2	CVLT_VFW1	10	TMT_B
3	CVLT_VFW2	11	CVLT_DG5

Table 4.6: Features ranked by RFECV. Features with Rank 1 are those selected.

To compare the performance when using all features and the reduced set of features, a default model and the model optimized by grid search were scored using 5-fold cross-validation with both sets of features.

Model	No. of Features	micro-average F1	macro-average F1
Base model	16	0.69 ± 0.07 [0.63-0.75]	0.68 ± 0.07 [0.62-0.74]
Base model	6	0.72 ± 0.07 [0.66-0.77]	0.71 ± 0.07 [0.65-0.77]
Optimized model	16	0.68 ± 0.07 [0.62-0.74]	0.67 ± 0.06 [0.61-0.73]
Optimized model	6	0.72 ± 0.08 [0.65-0.79]	0.71 ± 0.08 [0.64-0.78]

Table 4.7: Comparison of performance using the complete (16 features) and the reduced (6 features) data set.

A minimal increase in performance can be seen, but overlapping 95% confidence intervals suggest no statistically significant difference. Results of RFECV are similar to the results of optimization with grid search. These results seem to be the upper performance limit when using the available data set.

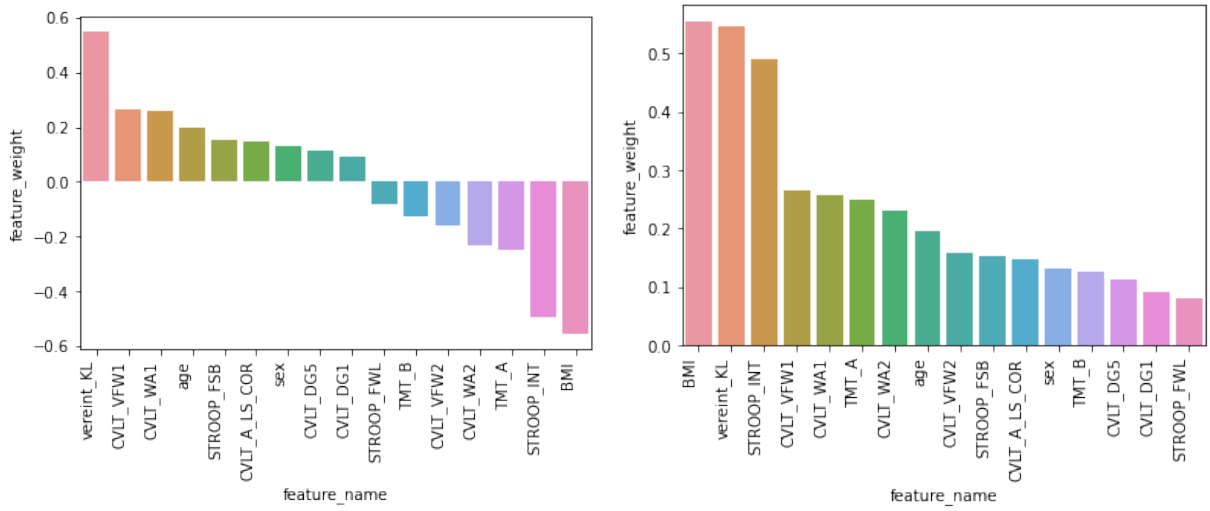
A trial with RFECV using the model optimized by grid search did yield 0.70 ± 0.08 [0.63-0.76] as the highest macro-average F1 score with 9 selected features.

4.5 Feature analysis

4.5.1 Feature ranking

Logistic Regression as implemented in scikit-learn performs classification by calculating the sum of all features multiplied by the features' coefficient, also called the feature's weight. If the value is greater than 0, the feature is classified as the positive label, in this case "bipolar", and otherwise as the negative label. Therefore, the feature coefficients have signed values: positive values add to a classification as "bipolar"' and negative values to a classification as "control". The plot 4.5(a) shows the features coefficients

of the grid-search optimized model as their signed values, and for better comparison of the feature's absolute contribution to a classification, the unsigned coefficient values are shown in plot 4.5(b). The coefficient values are shown in Table 4.8. Notably, the features "BMI", "vereint_KL" and "STROOP_INT" have visually higher coefficient values than all the other features. These three features are also selected as part of the top six features when performing RFECV as shown in Table 4.6. The other three selected features, "TMT_A", "CVLT_WA1" and "CVLT_WA2", are also part of the top seven ranked features by absolute coefficient with "CVLT_VFW1" being the only exception.



(a) Signed values

(b) Absolute values for easier comparison

Figure 4.5: Feature ranking of the optimized Logistic Regression model.

Feature	Coefficient	Feature	Coefficient
vereint_KL	0.546738	CVLT_DG1	0.089955
CVLT_VFW1	0.265095	STROOP_FWL	-0.079407
CVLT_WA1	0.257938	TMT_B	-0.124755
age	0.196262	CVLT_VFW2	-0.157302
STROOP_FSB	0.152752	CVLT_WA2	-0.229595
CVLT_A_LS_COR	0.146818	TMT_A	-0.248136

sex	0.131749	STROOP_INT	-0.491695
CVLT_DG5	0.112270	BMI	-0.555347

Table 4.8: Importance of features sorted by their signed coefficient value.

Figure 4.6 shows the feature ranking encoded by cognitive test and patient information. The three features with the highest coefficients are from the groups *patient information*, *d2/d2-R* and *Stroop Test*. The order of the remaining features shows no clear pattern or hint, that one of these tests is more important for classification than the others. This correlates with the finding that the six features selected by RFECV as shown in table 4.6 include at least one member from every group.

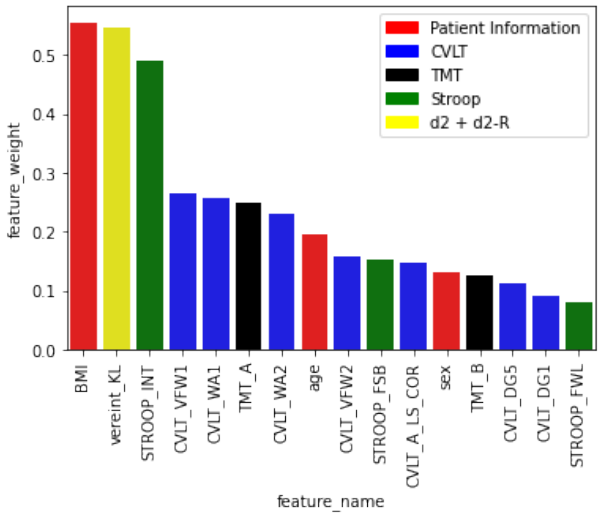


Figure 4.6: Feature ranking by absolute values colored according to cognitive test.

4.5.2 Model Analysis with ELI5

The library "ELI5" is used to provide information about a classification made by a model. To demonstrate its usage, the results for the first two elements of the test set, created before the grid search, are used. As the model for the prediction, the grid search optimized

Logistic Regression model is used. Figure 4.7 shows the output of the first two samples of the test data set, which are Pat_ID 125 and Pat_ID 95.

The classification is made by calculating the sum of contributions, which equals each feature's value multiplied by its coefficient. If this sum is larger than 0, the sample is classified as group "2" (control), else as group "1" (bipolar). The feature "<BIAS>" is added by the model with a constant value of 1 and a weight of -0.356 for each sample. The feature coefficients are the same as shown in Table 4.8 and therefore not displayed a second time. During preprocessing, feature scaling was performed as described in Section 3.2.1. Therefore, the feature's values are not the same as in the original data set file and cannot be interpreted as their corresponding clinical values.

y=2.0 (probability **0.891**, score **2.098**)

Contribution[?]	Feature	Value
+1.461	vereint_KL	2.673
+0.355	CVLT_VFW1	1.339
+0.317	STROOP_INT	-0.644
+0.299	TMT_A	-1.204
+0.149	TMT_B	-1.198
+0.132	CVLT_WA1	0.513
+0.124	sex	0.940
+0.117	CVLT_DG5	1.040
+0.082	STROOP_FSB	0.537
+0.069	CVLT_DG1	0.771
+0.027	STROOP_FWL	-0.334
+0.003	BMI	-0.005
-0.000	CVLT_A_LS_COR	-0.000
-0.180	CVLT_VFW2	1.147
-0.243	age	-1.240
-0.256	CVLT_WA2	1.116
-0.356	<BIAS>	1.000

(a) Pat_ID 125, correct prediction

y=1.0 (probability **0.645**, score **-0.598**)

Contribution[?]	Feature	Value
+0.356	<BIAS>	1.000
+0.209	STROOP_INT	0.424
+0.183	CVLT_WA2	0.798
+0.134	CVLT_VFW2	0.854
+0.115	CVLT_VFW1	-0.433
+0.095	STROOP_FSB	-0.622
+0.070	STROOP_FWL	0.887
+0.024	vereint_KL	-0.043
+0.008	CVLT_DG5	-0.071
+0.000	CVLT_A_LS_COR	-0.000
-0.020	CVLT_DG1	0.224
-0.057	TMT_B	-0.458
-0.058	TMT_A	-0.233
-0.081	CVLT_WA1	0.315
-0.094	BMI	-0.169
-0.124	sex	0.940
-0.163	age	0.829

(b) Pat_ID 95, incorrect prediction

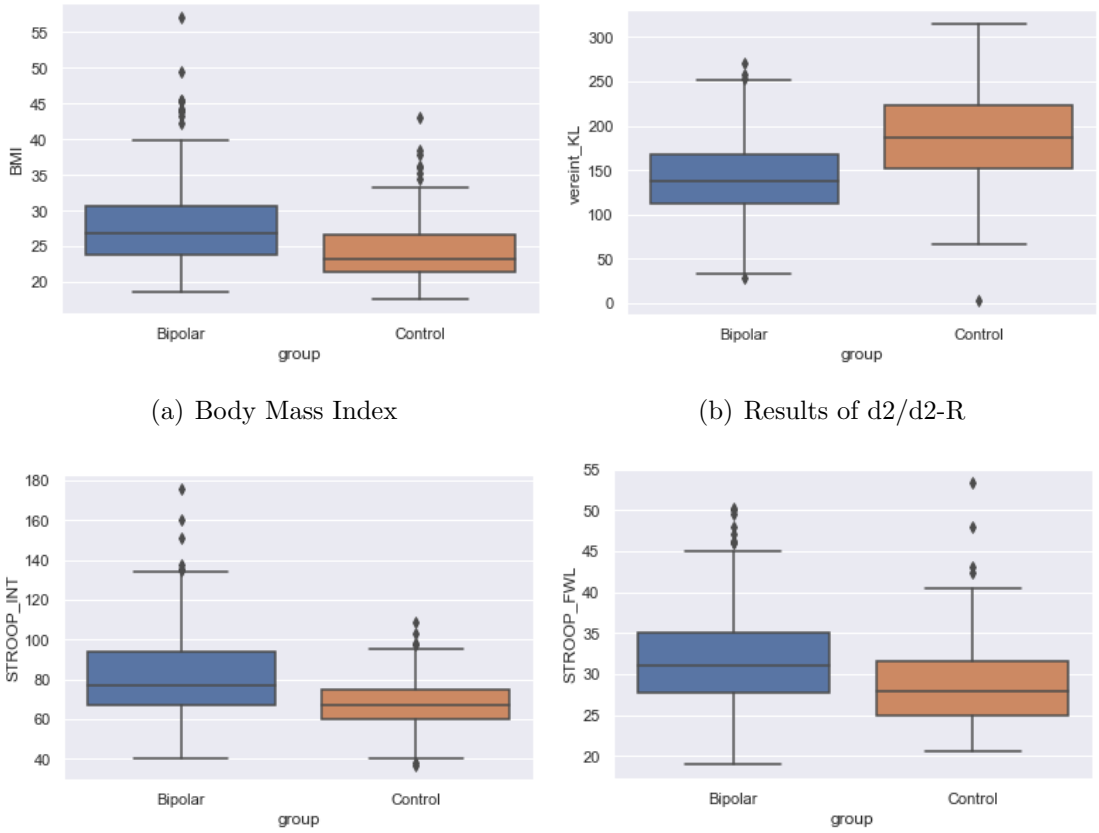
Figure 4.7: ELI5 output of two predictions.

The sample with Pat_ID 125 is classified correctly as a member of the healthy control group "2". The second sample with Pat_ID 95 is also a member of the healthy control group. This sample was misclassified as "bipolar" or group "1".

Visualizing a model's decision using ELI5 can help understand how the model came to a decision. Especially in a clinical context, this can help to understand why a patient was classified as healthy or as a patient with a disease. When using the Pat_ID 125 as an example, it can be seen that the feature "vereint_KL" had the largest contribution. The original value for this feature is 297, the third highest "vereint_KL" value in the data set (min: 2.7, max: 315, mean 160.22). Looking at Table 4.8, "vereint_KL" has a coefficient of 0.546738, which is the highest positive coefficient and the second-highest coefficient when comparing absolute numbers. When interpreting "vereint_KL" as one of the most important features based on its high coefficient, changes in this value have a higher impact on the model's decision compared to, for example, the feature "sex" with a coefficient of 0.131749. When deploying a model intended for clinical application, this kind of analysis can be provided at the same time as a prediction to understand how this prediction was made and to possibly gain some information about the patients themselves.

4.5.3 Analysis of the three most important features

As discussed in Section 4.5.1, "BMI", "vereint_KL" and "STROOP_INT" are the three features with significantly higher coefficients than the rest and therefore contribute the most to the classification of the model. When visually analyzing the distributions of these features as shown in Figure 4.8, patients labelled as "bipolar" seem to have, on average, higher values of "BMI" and "STROOP_INT" and lower values of "vereint_KL". For comparison, the feature with the lowest coefficient "STROOP_FWL" is also shown. When looking at "STROOP_FWL", both groups have a very similar range of values as compared to larger differences in the other three plots. This might be an explanation for the lower contribution of this feature, as overlapping values for both groups are not helpful when trying to separate these two groups using machine learning.



(a) Body Mass Index

(b) Results of d2/d2-R

(c) Results of Stroop Test - Interference

(d) Results of Stroop Test - Reading Color Names

Figure 4.8: Boxplots showing the distributions of the top three features and "STROOP_FWL" as the feature with the lowest contribution.

To see how well a model performs when only these three features are used for classification, a model was trained and optimized by grid search and then compared to the optimized model from Section 4.4.1. Very notably, it reached an almost identical performance of a macro-average F1 score of 0.74 and a micro-average F1 score of 0.77. For comparison of these two models, ROC curves with their AUROC values are shown in Figure 4.9. The performance of this reduced model further highlights the importance of the selected features. Again, this exact results are only valid for this specific train-test-split of the data.

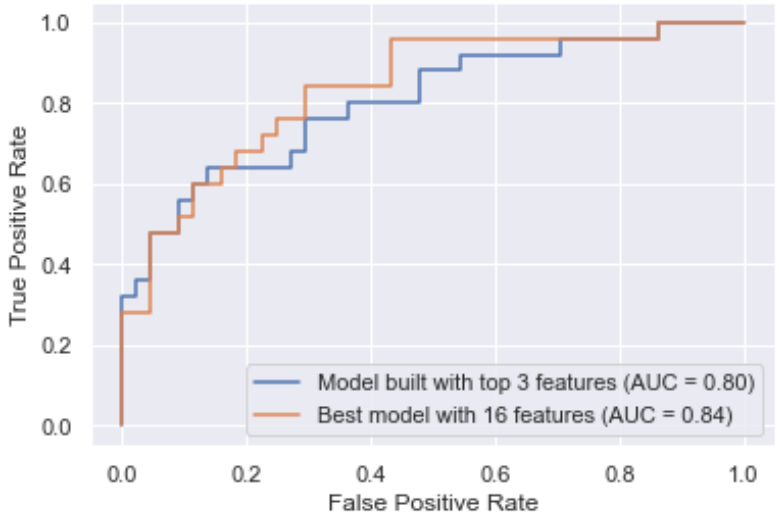


Figure 4.9: ROC curves of the model built with the top 3 features and the optimized model built with all 16 features.

Bipolar disorder and obesity

As "BMI" is one of the top three features and the "BIPFAT" study researched a link between obesity and bipolar disorder, this hypothesis is examined with the available data. In their report on obesity from 2000, the WHO defines several categories of obesity based on the BMI [61, Table 2.1]. When applying these criteria to the data set, 28.1% of bipolar samples fulfil the criteria of obesity, with the smaller number of 14.5% of controls. When including samples categorized as preobese, 64.8% of bipolar samples match these criteria, as opposed to 36.6% of controls. The proportion of accordingly categorized samples of the group "bipolar" is approximately the double of control samples. A visualization of both groups and the distribution of BMI's is shown in Figure 4.10. An important limiting factor to this observation is that the used data set was selected manually from a larger data set which implies a significant bias in this data, therefore these observations cannot be generalized for bipolar disorder.

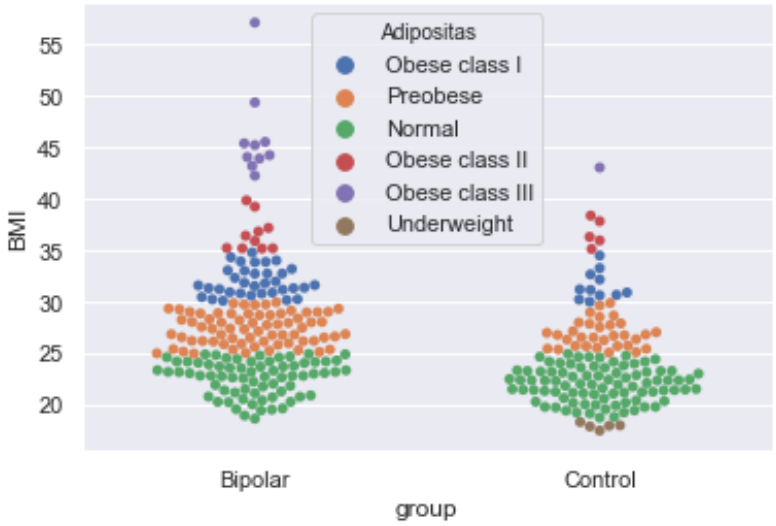


Figure 4.10: Samples of each group colored according to their WHO obesity class.

Chapter 5

Discussion

5.1 Conclusion

Applying machine learning models on the "BIPFAT" dataset is a trial to develop new methods for diagnosing bipolar disorder, especially to aid the early recognition of this disorder. In the following chapter, a summary of the results of the application of machine learning to the data set is given, as well as limitations and possible solutions and ideas for further research are discussed.

5.1.1 Model performance

In this study, all evaluated algorithms performed equally well with only minimal differences, with the best macro-average F1 score of 0.69 [0.66-0.73] achieved by Logistic Regression. These only small differences can probably be attributed to the properties of the data set.

When comparing the results of this study to the results from similar studies (Wu et al. [25], Sawalha et al. [27], Sonkurt et al. [28]), it is relevant to select which model should be compared. These three studies used optimized models to calculate their performance metrics, therefore the comparison should be made with the results of the optimized model as described in section 3.2.4. A limiting factor when comparing these results is that this

is not a representation of the model's true generalization performance, as these metrics are optimized only for a specific selection of test data used. The small difference in performance of the optimized model and the performance measured using nested cross-validation allows for a comparison of the results to get an idea of how these results compare to the results of the three studies cited, but it can be different for other subsets of the data. Table 5.1 lists the test characteristics of the other studies and results from the optimized model with an 80/20 train-test-split and initialization of the split method with random_state=0. The results of this study fit in well with the others. Looking at this data suggests that is might not be possible to achieve higher classification performance by using cognitive data alone.

Using sensitivity and specificity, likelihood ratios can be calculated for the results of the optimized model. When using these ratios to calculate pre-test and post-test probabilities, the resulting positive likelihood ratio (LR+) of 2.56 increases the probability of having bipolar disorder by approximately 15 to 20% after a positive classification as bipolar disorder, and the negative likelihood ratio (LR-) of 0.27 decreases the probability by 25 to 30% [62]. When implemented as a diagnostic test, a model with a similar performance can contribute to the overall diagnostic process, but cannot be used as a single tool for making diagnoses.

Study	Best Algorithm	Accuracy	F1 (macro)	Sensitivity	Specificity	AUROC
Sonkurt et al. [28]	Modified O-PCF	0.78	0.80	0.80	0.76	0.78
Wu et al. [25]	LASSO	0.71	0.73*	0.76	0.67	0.71
Sawalha et al. [27] (Chronic BPD)	Linear SVM	0.77	0.76*	0.76	0.77	N/A
Sawalha et al. [27] (First Episode)	Linear SVM	0.77*+	0.77*	0.75	0.79	0.77
ML4BP	Log. Regression	0.77	0.75	0.82	0.68	0.84

Table 5.1: Results of the similar studies and this thesis. * Metrics calculated based on given confusion matrices. +Accuracy given as 0.76 in original paper.

5.1.2 Data set

The performance of machine learning models depends to a large extent on the quality of the underlying data. Based on this, the assumption that better data leads to better results can be made.

The evaluation of algorithms resulted in very similar performance, with no algorithm performing considerably better than the rest. This observation leads to the conclusion that every algorithm is suited for this particular task, and that the underlying data might not allow for a better classification of bipolar and healthy patients using the available data. The inability to find clearly visible clusters in the data when applying PCA and t-SNE also fits this picture. With the assumption that these plots are a valid 2-dimensional projection of the 16-dimensional feature space, regions where one class is dominant and areas where both classes are mixed can be observed. In the regions with class dominance, outliers from the other class can be detected. This can be best seen in the t-SNE plots with perplexities of 10 and 50 shown in figure 4.2. When assuming that classification of these outliers results in misclassification and that gains in performance are made when optimizing for regions of mixed classes, this might explain why no significant gains in performance could be made by model optimization and why the results for all algorithms are similar. Dimensionality reduction and clustering might therefore be a tool to guess if a data set will achieve good results before applying it to machine learning. This approach could be used to reduce the effort necessary to find high quality data sets with additional features for future studies.

5.1.3 Limitations

The data set used for this thesis is a manually selected subset of patient from the “BIPFAT” study. The data was selected explicitly for this classification task to get a data set with an almost equal number of male and female samples. This is a limitation, as classification of a manually selected data set using machine learning adopts potential bias in the data. Therefore, the results of this study are only valid for this specific data set and might not be generalizable for classification of unselected patients.

Another limitation might be that if patients are given the wrong diagnosis in the data set, the machine learning model learns the false diagnosis as correct and might classify future patients with similar features incorrectly. As this approach is tested for aiding the diagnosis and possibly the early recognition of bipolar disorder, the underlying data set must be free of errors to avoid introducing false classifications into the model.

5.1.4 Considerations for the implementation in clinical use

Model optimization is implemented in scikit-learn to maximize the performance of a metric, which can be defined as desired. In this study, the F1 score was chosen as it is a balanced metric of precision and recall and allows for an easy comparison of different models. Accuracy and F1 score are commonly used to evaluate machine learning models, but metrics like sensitivity, specificity, positive (LR+) or negative (LR-) likelihood ratio might be more useful in the context of a medical application. Which of those metrics should be used depends entirely on the context in which a machine learning application might be put into practice. A possible scenario could be the implementation of a screening program for early recognition of bipolar disorder used by general practitioners. This scenario might require the optimization for maximum specificity to generate as few false positive tests as possible. In a scenario, where machine learning is used as a diagnostic test to aid decision-making, optimizing the model to achieve a maximum LR+ to achieve a high post-test probability might be desirable. Any real-live clinical application should therefore have a specific purpose and should then be optimized to maximize the model's performance as required by optimizing for certain metrics.

When looking at the performance of an algorithm in combination with the complexity and requirement of computing power of said algorithm, Logistic Regression showed the best F1 score and the shortest runtime required for the model. This consideration might be relevant in settings where only low computing power is available. In a setting where a machine learning model is used in a clinical context, this might be less relevant as the training only needs to be done once and a single classification task is not going to strain the resources of this environment. In case of frequent updates of the model with new data, this might become relevant. When implementing such a model, a cloud-based approach

could be taken where the model is run on a high-performance server where all calculations are made, taking into account data protection regulations.

5.1.5 Proposals for further research

The features of the data set used is a selection of all features from the original "BIPFAT" study and is only focused on cognitive data. A more exploratory approach might be to use the complete data set and train machine learning models, and the subsequently perform feature analysis and selection to find out which features are relevant for classification and which can be excluded. Using such an approach, completely new features and clinical measures, like for example laboratory values, vital signs, imaging data or possibly subjective and self-reported mood states, could be identified which might have a future role in the diagnosis of bipolar disorder. Alternatively, such an analysis could help speed up the diagnostic process by eliminating the necessity to perform tests and examinations which do not add any useful information for the diagnosis of bipolar disorder. Similar to the results by Ma et al. [36], this could be used to simplify diagnostic criteria and therefore improve patient care.

This approach was basically done by building the model with only the three features with the highest importance coefficients. As this model has an almost equal performance when compared to the model with all 16 features, it can be assumed that using such a simplified model as a tool for decision-making in a clinical context would have similar accuracy as the original model. This reduced model represents an interesting finding, as the features used are "BMI" and the results of only two out of four cognitive tests used in the "BIPFAT" study. The BMI can be easily measured, and requiring only two cognitive tests to reach the same diagnostic accuracy means that the time and the administrative and logistic effort required to gather the necessary information can be drastically reduced. When applied in a clinical setting or as a low-threshold screening tool, such a diagnostic aid can be helpful and play a future role in the early recognition of bipolar disorder. Using this finding as a basis for further research, this approach could be expanded by analyzing the complete "BIPFAT" data set to gain further insight into the data and possibly find other features with high contribution to the classification which, as a result, make the diagnosis easier and faster.

In this study, results from only one execution of the cognitive tests is used for classification. An approach to possibly improve the early recognition of bipolar disorder could be to look at cognitive performance over a longer period of time. A machine learning model could be trained to recognize changes in performance which might indicate the progression of the disease, or even could identify episodes before they manifest. Such an approach would require to gather a large amount of data from patients and controls over a long period of time to produce viable results.

To validate results of all these possible approaches, prospective validation should be done before using machine learning in clinical context.

5.2 Outlook

The objective of this thesis was to test how well machine learning performs when being applied to the task of classifying healthy controls from patients with bipolar disorder when using results from cognitive tests and basic patient information. A macro-average F1 score of 0.75 was reached when measuring a highly optimized model. A model with such a performance might not yet be suited for application in a clinical context, but it suggests what is possible with the available data and technology. Due to the ongoing digitization of medicine, more data becomes available every day. Using the approach tested in this thesis and expanding the data set with newly gathered information might lead to improved classification performance and maybe one day to a real-world application where such a model is used to actively support clinicians in decision-making and the diagnosis of bipolar disorder, which subsequently can lead to improved patient care. It should therefore be an objective of every healthcare system to develop ways for the automated collection of unbiased data to fully harness the potential of machine learning and other applications of artificial intelligence.

Bibliography

[1] GBD 2019 Diseases and Injuries Collaborators . Global burden of disease 2019 disease, injury, and impairment summaries - bipolar disorder. *Lancet*, 396, October 2020. URL https://www.thelancet.com/pb-assets/Lancet/gbd/summaries/diseases/bipolar-disorder.pdf. (accessed 21.Oktober 2020).

[2] Kessler R. C., Berglund P., Demler O., Jin R., Merikangas K. R., and Walters E. E. Lifetime Prevalence and Age-of-Onset Distributions of DSM-IV Disorders in the National Comorbidity Survey Replication. *Archives of General Psychiatry*, 62(6):593–602, 06 2005. ISSN 0003-990X. doi: 10.1001/archpsyc.62.6.593. URL https://doi.org/10.1001/archpsyc.62.6.593.

[3] Merikangas K., Akiskal H., Angst J., Greenberg P., Hirschfeld R., Petukhova M., and Kessler R. Lifetime and 12-month prevalence of bipolar spectrum disorder in the national comorbidity survey replication. arch gen psychiatry, 2007 May. 2007 Sep.

[4] Pini S., De Queiroz V., Pagnin D., Pezawas L., Angst J., Cassano G., and Wittchen H. Prevalence and burden of bipolar disorders in european countries. *Eur Neuropsychopharmacol*, 15(4):425–459, 2005 Aug. doi: 10.1016/j.euroneuro.2005.04.011.

[5] Baldessarini R. J., Tondo L., Baethge C. J., Lepri B., and Bratti I. M. Effects of treatment latency on response to maintenance treatment in manic-depressive disorders. *Bipolar Disorders*, 9(4):386–393, jun 2007. doi: 10.1111/j.1399-5618.2007.00385.x.

[6] World Health Organization . International classification of diseases for mortality and morbidity statistics (11th revision), 2018.

[7] American Psychiatric Association . *Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition*. American Psychiatric Association, Arlington, VA, 2013. ISBN 089042554X.

[8] World Health Organization . International classification of diseases for mortality and morbidity statistics (11th revision) - 06 mood disorders - bipolar or related disorders, 2018. URL https://icd.who.int/browse11/l-m/en#/http://id.who.int/icd/entity/613065957. (accessed 21.02.2021).

[9] Rowland T. A. and Marwaha S. Epidemiology and risk factors for bipolar disorder. *Therapeutic Advances in Psychopharmacology*, 8(9):251–269, apr 2018. doi: 10.1177/2045125318769235.

[10] Haack S., Pfennig A., and Bauer M. Bipolare depression. *Der Nervenarzt*, 81(5): 525–530, mar 2010. doi: 10.1007/s00115-009-2849-3.

[11] Marneros A. and Brieger P. Prognosis of bipolar disorder. In *Bipolar Disorder*, pages 97–189. John Wiley & Sons, Ltd, apr 2002. doi: 10.1002/047084650x.ch2.

[12] Benazzi F. Prevalence and clinical correlates of residual depressive symptoms in bipolar II disorder. *Psychotherapy and Psychosomatics*, 70(5):232–238, 2001. doi: 10.1159/000056260.

[13] Plans L., Barrot C., Nieto E., Rios J., Schulze T., Papiol S., Mitjans M., Vieta E., and Benabarre A. Association between completed suicide and bipolar disorder: A systematic review of the literature. *Journal of Affective Disorders*, 242:111–122, jan 2019. doi: 10.1016/j.jad.2018.08.054.

[14] Pompili M., Gonda X., Serafini G., Innamorati M., Sher L., Amore M., Rihmer Z., and Girardi P. Epidemiology of suicide in bipolar disorders: a systematic review of the literature. *Bipolar Disorders*, 15(5):457–490, jun 2013. doi: 10.1111/bdi.12087.

[15] Rothenhäusler H.-B. and Täschner K.-L. *Kompendium Praktische Psychiatrie*. Springer Vienna, 2012. doi: 10.1007/978-3-7091-1237-3.

[16] DGBS e.V. and DGPPN e.V. . S3-Leitlinie zur Diagnostik und Therapie Bipolarer Störungen Langversion. 2019.

[17] Atkins D., Best D., Briss P. A., Eccles M., Falck-Ytter Y., Flottorp S., Guyatt G. H., Harbour R. T., Haugh M. C., Henry D., Hill S., Jaeschke R., Leng G., Liberati A., Magrini N., Mason J., Middleton P., Mrukowicz J., O'Connell D., Oxman A. D., Phillips B., Schünemann H. J., Edejer T. T.-T., Varonen H., Vist G. E., Williams J. W. J., and Zaza S. Grading quality of evidence and strength of recommendations. *BMJ (Clinical research ed.)*, 328:1490, Jun 2004.

[18] Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften (AWMF)- Ständige Kommission Leitlinien . AMWF-Regelwerk " Leitlinien" . 2nd Edition 2020. URL http://www.awmf.org/leitlinien/awmf-regelwerk.html. (accessed 12.10.2021).

[19] Ahmed G. K., Elbeh K., Khalifa H., and Samaan M. R. Impact of duration of untreated illness in bipolar i disorder (manic episodes) on clinical outcome, socioeconomic burden in egyptian population. *Psychiatry Research*, 296:113659, feb 2021. doi: 10.1016/j.psychres.2020.113659.

[20] Altamura A. C., Dell'Osso B., Berlin H. A., Buoli M., Bassetti R., and Mundo E. Duration of untreated illness and suicide in bipolar disorder: a naturalistic study. *European Archives of Psychiatry and Clinical Neuroscience*, 260(5):385–391, nov 2009. doi: 10.1007/s00406-009-0085-2.

[21] McCraw S., Parker G., Graham R., Synnott H., and Mitchell P. The duration of undiagnosed bipolar disorder: Effect on outcomes and treatment response. *Journal of Affective Disorders*, 168:422–429, oct 2014. doi: 10.1016/j.jad.2014.07.025.

[22] World Health Organization . International classification of diseases for mortality and morbidity statistics (11th revision) - 06 mood disorders - depressive disorders, 2018. URL https://icd.who.int/browse11/l-m/en#/http://id.who.int/icd/entity/1563440232. (accessed 08.03.2021).

[23] Hirschfeld R., Lewis L., and Vornik L. Perceptions and impact of bipolar disorder: how far have we really come? results of the national depressive and manic-depressive association 2000 survey of individuals with bipolar disorder. *J Clin Psychiatry*, 64 (2):161–74, 2003 Feb.

[24] Drancourt N., Etain B., Lajnef M., Henry C., Raust A., Cochet B., Mathieu F., Gard S., MBailara K., Zanouy L., Kahn J. P., Cohen R. F., Wajsbrot-Elgrabli O., Leboyer M., Scott J., and Bellivier F. Duration of untreated bipolar disorder: missed opportunities on the long road to optimal treatment. *Acta Psychiatrica Scandinavica*, 127(2):136–144, aug 2012. doi: 10.1111/j.1600-0447.2012.01917.x.

[25] Wu M.-J., Passos I. C., Bauer I. E., Lavagnino L., Cao B., Zunta-Soares G. B., Kapczinski F., Mwangi B., and Soares J. C. Individualized identification of euthymic bipolar disorder using the cambridge neuropsychological test automated battery (cantab) and machine learning. *Journal of affective disorders*, 192:219–225, March 2016. ISSN 1573-2517. doi: 10.1016/j.jad.2015.12.053.

[26] Cambridge Cognition . CANTAB [Cognitive assessment software], 2019. URL www.cantab.com.

[27] Sawalha J., Cao L., Chen J., Selvitella A., Liu Y., Yang C., Li X., Zhang X., Sun J., Zhang Y., Zhao L., Cui L., Zhang Y., Sui J., Greiner R., Li X.-M., Greenshaw A., Li T., and Cao B. Individualized identification of first-episode bipolar disorder using machine learning and cognitive tests. *Journal of affective disorders*, 282:662–668, March 2021. ISSN 1573-2517. doi: 10.1016/j.jad.2020.12.046.

[28] Sonkurt H. O., Altinöz A. E., Cimen E., Kösger F., and Öztürk G. The role of cognitive functions in the diagnosis of bipolar disorder: A machine learning model. *International journal of medical informatics*, 145:104311, January 2021. ISSN 1872-8243. doi: 10.1016/j.ijmedinf.2020.104311.

[29] Fernandes B. S., Karmakar C., Tamouza R., Tran T., Yearwood J., Hamdani N., Laouamri H., Richard J.-R., Yolken R., Berk M., Venkatesh S., and Leboyer M. Precision psychiatry with immunological and cognitive biomarkers: a multi-domain prediction for the diagnosis of bipolar disorder or schizophrenia using machine learning. *Translational psychiatry*, 10:162, May 2020. ISSN 2158-3188. doi: 10.1038/s41398-020-0836-4.

[30] Wechsler D. *Wechsler Adult Intelligence Scale, 3rd Edition*. The Psychological Corporation, San Antonio, 1997.

[31] Nelson H. E. *National Adult Reading Test (NART): For the Assessment of Premorbid Intelligence in Patients with Dementia: Test Manual*. NFER-Nelson, Windsor, UK, 1982.

[32] Walsh-Messinger J., Jiang H., Lee H., Rothman K., Ahn H., and Malaspina D. Relative importance of symptoms, cognition, and other multilevel variables for psychiatric disease classifications by machine learning. *Psychiatry research*, 278:27–34, August 2019. ISSN 1872-7123. doi: 10.1016/j.psychres.2019.03.048.

[33] Endicott J., Spitzer R. L., Fleiss J. L., and Cohen J. The global assessment scale. a procedure for measuring overall severity of psychiatric disturbance. *Archives of general psychiatry*, 33:766–771, June 1976. ISSN 0003-990X. doi: 10.1001/archpsyc.1976.01770060086012.

[34] Kay S. R., Fiszbein A., and Opler L. A. The positive and negative syndrome scale (panss) for schizophrenia. *Schizophrenia bulletin*, 13:261–276, 1987. ISSN 0586-7614. doi: 10.1093/schbul/13.2.261.

[35] Nurnberger J. I., Blehar M. C., Kaufmann C. A., York-Cooler C., Simpson S. G., Harkavy-Friedman J., Severe J. B., Malaspina D., and Reich T. Diagnostic interview for genetic studies. rationale, unique features, and training. nimh genetics initiative. *Archives of general psychiatry*, 51:849–59; discussion 863–4, November 1994. ISSN 0003-990X. doi: 10.1001/archpsyc.1994.03950110009002.

[36] Ma Y., Ji J., Huang Y., Gao H., Li Z., Dong W., Zhou S., Zhu Y., Dang W., Zhou T., Yu H., Yu B., Long Y., Liu L., Sachs G., and Yu X. Implementing machine learning in bipolar diagnosis in china. *Translational psychiatry*, 9:305, Nov 2019.

[37] Samuel A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, jul 1959. doi: 10.1147/rd.33.0210.

[38] Raschka S. and Mirjalili V. *Python Machine Learning - Third Edition*. Packt Publishing, 2019. ISBN 1789955750. URL https://www.ebook.de/de/product/38434617/sebastian_raschka_vahid_mirjalili_python_machine_learning_third_edition.html.

[39] Geron A. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Sebastopol, CA, 2017. ISBN 9781491962299.

[40] Varma S. and Simon R. Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics*, 7(1):91, 2006. doi: 10.1186/1471-2105-7-91.

[41] Raschka S. " nested-cv.png". URL https://sebastianraschka.com/images/blog/2018/model-evaluation-selection-part4/nested-cv.png. licensed under CC BY 4.0 (accessed 23.Novenber 2020).

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

[43] Cox D. R. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[44] Vapnik V. and Chervonenkis A. *Theory of Pattern Recognition [in Russian]*. Nauka, Moscow, 1974. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979).

[45] scikit-learn developers , 2022. URL https://scikit-learn.org/stable/_images/sphx_glr_plot_separating_hyperplane_001.png. (accessed 16.01.2022).

[46] Freund Y. and Schapire R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1): 119–139, aug 1997. ISSN 0022-0000. doi: 10.1006/jcss.1997.1504.

[47] Breiman L. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/a:1010933404324.

[48] Mccullock W. and Pitts W. A logical calculus of ideas immanent in nervous activity. archive copy of 27 november 2007 on wayback machine. *Avtomaty [Automated Devices] Moscow, Inostr. Lit. publ*, pages 363–384, 1956.

[49] Rosenblatt F. The perceptron - a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York, January 1957.

[50] Kluyver T., Ragan-Kelley B., Pérez F., Granger B., Bussonnier M., Frederic J., Kelley K., Hamrick J., Grout J., Corlay S., Ivanov P., Avila D., Abdalla S., and Willing C. Jupyter notebooks – a publishing format for reproducible computational workflows. In Loizides F. and Schmidt B., editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[51] Reininghaus E. Z., McIntyre R. S., Reininghaus B., Geisler S., Bengesser S. A., Lackner N., Hecht K., Birner A., Kattnig F., Unterweger R., Kapfhammer H.-P., Zelzer S., Fuchs D., and Mangge H. Tryptophan breakdown is increased in euthymic overweight individuals with bipolar disorder: a preliminary report. *Bipolar Disorders*, 16 (4):432–440, dec 2013. doi: https://doi.org/10.1111/bdi.12166.

[52] Llinàs-Reglà J., Vilalta-Franch J., López-Pousa S., Calvó-Perxas L., Rodas D. T., and Garre-Olmo J. The trail making test. *Assessment*, 24(2):183–196, jul 2016. doi: 10.1177/1073191115602552.

[53] Stroop J. R. Studies of interference in serial verbal reactions. *Journal of Experimental Psychology*, 18(6):643–662, 1935. doi: 10.1037/h0054651.

[54] Scarpina F. and Tagini S. The stroop color and word test. *Frontiers in Psychology*, 8, apr 2017. doi: 10.3389/fpsyg.2017.00557.

[55] Delis D., Kramer J., Kaplan E., and Ober B. California verbal learning test research edition manual. *San Antonio: The Psychological Corporation*, 1987.

[56] Brickenkamp R., Schmidt-Atzert L., and Liepmann D. Test d2 - revision aufmerksamkeits und konzentrationstest (manual). *Hogrefe*, 2010. Göttingen.

[57] Pearson K. LIII. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2 (11):559–572, nov 1901. doi: 10.1080/14786440109462720.

[58] van der Maaten L. and Hinton G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

[59] Guyon I., Weston J., Barnhill S., and Vapnik V. Gene selection for cancer classification using support vector machines. *Mach. Learn*, 46(1-3):389–422, 2002.

[60] Korobov M. and Lopuhin K. Eli5. URL https://pypi.org/project/eli5/. (accessed 24.November 2020).

[61] World Health Organization . *Obesity: Preventing and Managing the Global Epidemic*, volume 894 of *WHO Technical Report Series*. World Health Organization, January 2000. ISBN 9241208945. URL https://www.ebook.de/de/product/6551038/world_health_organization_obesity_preventing_and_managing_the_global_epidemic.html.

[62] McGee S. Simplifying likelihood ratios. *Journal of General Internal Medicine*, 17(8):647–650, aug 2002. doi: 10.1046/j.1525-1497.2002.10750.x.

Appendix A

Jupyter Notebook

[1]:
```
# imports
import eli5
import platform

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

from sklearn import manifold
from sklearn.feature_selection import RFECV
from sklearn.metrics import classification_report, plot_confusion_matrix,␣
 ↪roc_auc_score, plot_roc_curve
from sklearn.model_selection import cross_validate, GridSearchCV, KFold,␣
 ↪train_test_split, cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA

# models
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.neural_network import MLPClassifier
```

```
/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-
packages/sklearn/utils/deprecation.py:143: FutureWarning: The
sklearn.metrics.scorer module is  deprecated in version 0.22 and will be removed
in version 0.24. The corresponding classes / functions should instead be
imported from sklearn.metrics. Anything that cannot be imported from
sklearn.metrics is now part of the private API.
  warnings.warn(message, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-
packages/sklearn/utils/deprecation.py:143: FutureWarning: The
sklearn.feature_selection.base module is  deprecated in version 0.22 and will be
removed in version 0.24. The corresponding classes / functions should instead be
imported from sklearn.feature_selection. Anything that cannot be imported from
sklearn.feature_selection is now part of the private API.
  warnings.warn(message, FutureWarning)
```

[2]:
```
def calc_ci(sample, z_val=1.96, digits=2):
    """
    Returns the confidence interval for a sample as string with the format:␣
 ↪[lower-upper]

    Parameters
    ---------
    sample : array
        sample of values for confidence interval calculation
    z_val : floar, default=1.96
        z-value for confidence interval
    digits : int, default=2
        number of digits to round

    Returns
    -------
    string of the CI in the format [lower-upper]
    """
    mean = sample.mean()
    std = sample.std()
    se = std / np.sqrt(len(sample))
    return "[{}-{}]".format(round(mean - se * z_val, digits), round(mean + se *␣
 ↪z_val, digits))
```

1 Loading the data set for descriptive analysis

[3]:
```
# check which OS to load from correct path
os = platform.system()
if os == 'Darwin':
    # path to vault for macOS
    ml4bp = pd.read_csv(
        "/Volumes/tresorML4BP/Data/BIPFAT 15.09.2020 kOGINTION_SELCTED.CSV", sep=';')
# Windows
else:
    ml4bp = pd.read_csv(
        "G:\Data\BIPFAT 15.09.2020 kOGINTION_SELCTED.CSV", sep=';')
```

```
# drop the last line (empty values)
ml4bp.drop(ml4bp.tail(1).index, inplace=True)

# drop Patient ID and group from DataFrame
ml4bp = ml4bp.drop(["Pat_ID"], axis=1)
```

2 Descriptive statistics

2.1 Descrpitve statistics for every column of the data set

[4]:
```
# nr. of samples in both groups
ml4bp.group.value_counts()
```

[4]:
```
1.0    196
2.0    145
Name: group, dtype: int64
```

[5]:
```
# separated into two views for better readability
# 50% percentile = median
ml4bp.iloc[:, :9].describe(percentiles=[.5])
```

[5]:
```
                age         sex       group         BMI       TMT_A       TMT_B  \
count   336.000000  336.000000  341.000000  336.000000  341.000000  338.000000
mean     41.186310    1.529762    1.425220   26.426964   31.966921   73.483580
std      14.779413    0.499858    0.495103    5.760030   13.764583   39.684469
min      17.890000    1.000000    1.000000   17.530000   10.760000   24.200000
50%      38.515000    2.000000    1.000000   25.150000   29.000000   64.970000
max      77.950000    2.000000    2.000000   57.130000  105.300000  309.940000

        STROOP_FWL  STROOP_FSB  STROOP_INT
count   334.000000  334.000000  334.000000
mean     30.658892   47.156976   75.906467
std       5.937043    8.900113   20.059469
min      18.900000   25.900000   36.300000
50%      29.785000   45.970000   72.195000
max      53.440000   92.870000  176.000000
```

[6]:
```
# 50% percentile = median
ml4bp.iloc[:, 9:].describe(percentiles=[.5])
```

[6]:
```
        CVLT_A_LS_COR    CVLT_DG1    CVLT_DG5   CVLT_VFW1    CVLT_WA1  \
count      239.000000  337.000000  337.000000  334.000000  334.000000
mean        54.820084    7.771513   13.192878   11.467066   12.416168
std         12.299017    5.527907    2.720329    3.426245    5.093467
min         14.000000    1.000000    6.000000    2.000000    2.000000
50%         56.000000    7.000000   14.000000   12.000000   13.000000
max         81.000000   97.000000   16.000000   16.000000   86.000000

         CVLT_VFW2    CVLT_WA2  vereint_KL
count   333.000000  333.000000  322.000000
mean     12.087087   12.489489  160.225776
std       3.456474    3.187520   52.746820
```

```
min       2.000000    3.000000    2.700000
50%      13.000000   13.000000  156.500000
max      16.000000   16.000000  315.000000
```

2.2 Number of missing values for each feature

[7]:
```
print(ml4bp.isnull().sum(axis=0))
```

```
age               5
sex               5
group             0
BMI               5
TMT_A             0
TMT_B             3
STROOP_FWL        7
STROOP_FSB        7
STROOP_INT        7
CVLT_A_LS_COR   102
CVLT_DG1          4
CVLT_DG5          4
CVLT_VFW1         7
CVLT_WA1          7
CVLT_VFW2         8
CVLT_WA2          8
vereint_KL       19
dtype: int64
```

2.3 Descriptive statistics for each feature per group/sex

Varying counts are due to missing values of this feature

[8]:
```
# 50% percentile = median, precision set to .01
for column in ml4bp:
    # skip group
    if str(column) != 'group':
        print(ml4bp.filter(['group', column], axis=1).groupby(
            ['group']).describe(include='all', percentiles=[.5]).round(2))
        print()  # empty line
```

```
          age
        count   mean    std    min    50%    max
group
1.0     195.0  44.17  13.67  17.89  43.78  77.95
2.0     141.0  37.07  15.30  18.86  30.20  76.56

          sex
        count  mean   std  min  50%  max
group
1.0     195.0  1.45  0.50  1.0  1.0  2.0
2.0     141.0  1.64  0.48  1.0  2.0  2.0

          BMI
        count   mean   std    min    50%    max
```

```
group
1.0    195.0  27.90  6.10  18.69  26.79  57.13
2.0    141.0  24.39  4.55  17.53  23.26  43.06

       TMT_A
       count   mean    std    min    50%    max
group
1.0    196.0  36.03  14.41  13.27  32.96  105.3
2.0    145.0  26.47  10.65  10.76  24.43   71.7

       TMT_B
       count   mean    std   min   50%     max
group
1.0    193.0  84.16  45.62  31.2  72.0  309.94
2.0    145.0  59.28  23.55  24.2  55.0  168.00

      STROOP_FWL
           count   mean   std   min    50%    max
group
1.0        189.0  31.99  6.12  18.9  31.00  50.18
2.0        145.0  28.93  5.23  20.5  27.95  53.44

      STROOP_FSB
           count   mean   std   min    50%    max
group
1.0        189.0  49.28  8.83  31.0  48.00  79.00
2.0        145.0  44.39  8.23  25.9  43.05  92.87

      STROOP_INT
           count   mean    std    min    50%     max
group
1.0        189.0  82.45  22.43  39.97  77.00  176.00
2.0        145.0  67.38  12.01  36.30  67.07  108.54

      CVLT_A_LS_COR
              count   mean    std   min   50%   max
group
1.0           130.0  51.33  12.29  22.0  51.0  81.0
2.0           109.0  58.98  10.99  14.0  61.0  78.0

      CVLT_DG1
         count  mean   std  min  50%   max
group
1.0      192.0  6.90  2.45  1.0  6.5  15.0
2.0      145.0  8.93  7.81  2.0  8.0  97.0

      CVLT_DG5
         count   mean   std  min   50%   max
group
1.0      192.0  12.42  2.87  6.0  13.0  16.0
2.0      145.0  14.22  2.12  6.0  15.0  16.0

      CVLT_VFW1
          count   mean   std  min   50%   max
```

```
group
1.0       189.0  10.37  3.44  2.0  11.0  16.0
2.0       145.0  12.90  2.84  3.0  14.0  16.0

      CVLT_WA1
         count   mean   std  min   50%   max
group
1.0      189.0  11.25  3.20  2.0  12.0  16.0
2.0      145.0  13.94  6.52  4.0  14.0  86.0

      CVLT_VFW2
          count   mean   std  min   50%   max
group
1.0       189.0  11.17  3.52  2.0  12.0  16.0
2.0       144.0  13.28  2.99  2.0  14.0  16.0

      CVLT_WA2
         count   mean   std  min   50%   max
group
1.0      189.0  11.62  3.27  3.0  12.0  16.0
2.0      144.0  13.63  2.68  3.0  14.0  16.0

      vereint_KL
           count    mean    std   min    50%    max
group
1.0        186.0  139.81  44.82  28.0  138.0  270.0
2.0        136.0  188.14  50.03   2.7  186.5  315.0
```

3 Reloading the data set from scratch and performing imputation and preprocessing

[9]:
```python
# check which OS to load from correct path
os = platform.system()
if os == 'Darwin':
    # path to vault for macOS
    ml4bp = pd.read_csv(
        "/Volumes/tresorML4BP/Data/BIPFAT 15.09.2020 kOGINTION_SELCTED.CSV", sep=';')
# Windows
else:
    ml4bp = pd.read_csv(
        "G:\Data\BIPFAT 15.09.2020 kOGINTION_SELCTED.CSV", sep=';')

# drop the last line (empty values)
ml4bp.drop(ml4bp.tail(1).index, inplace=True)

# save column group as y (classes)
y = ml4bp.group
# drop Patient ID and group from DataFrame before storing in X (features)
ml4bp = ml4bp.drop(["Pat_ID"], axis=1)
ml4bp = ml4bp.drop(["group"], axis=1)
```

```
# save resulting DF as X
X = ml4bp.loc[:]

# imputation and preprocessing
# impute missing sex with random sex 1 or 2 before SimpleImputer is called
# to avoid imputation with float value between 1 and 2
# set randint(1,3) for random 1 or 2
# fixed seed for reproducible results between multiple runs
np.random.seed(0)
X['sex'] = X['sex'].apply(
    lambda x:  np.random.randint(1, 3) if (np.isnan(x)) else x)
# impute all NaN's with the columns mean value
X = SimpleImputer(missing_values=np.nan, strategy='mean').fit_transform(X)
# Scaling with StandardScaler
X = StandardScaler().fit_transform(X)
```

4 Clustering analysis and dimensionality reduction

4.1 Principal Component Analysis

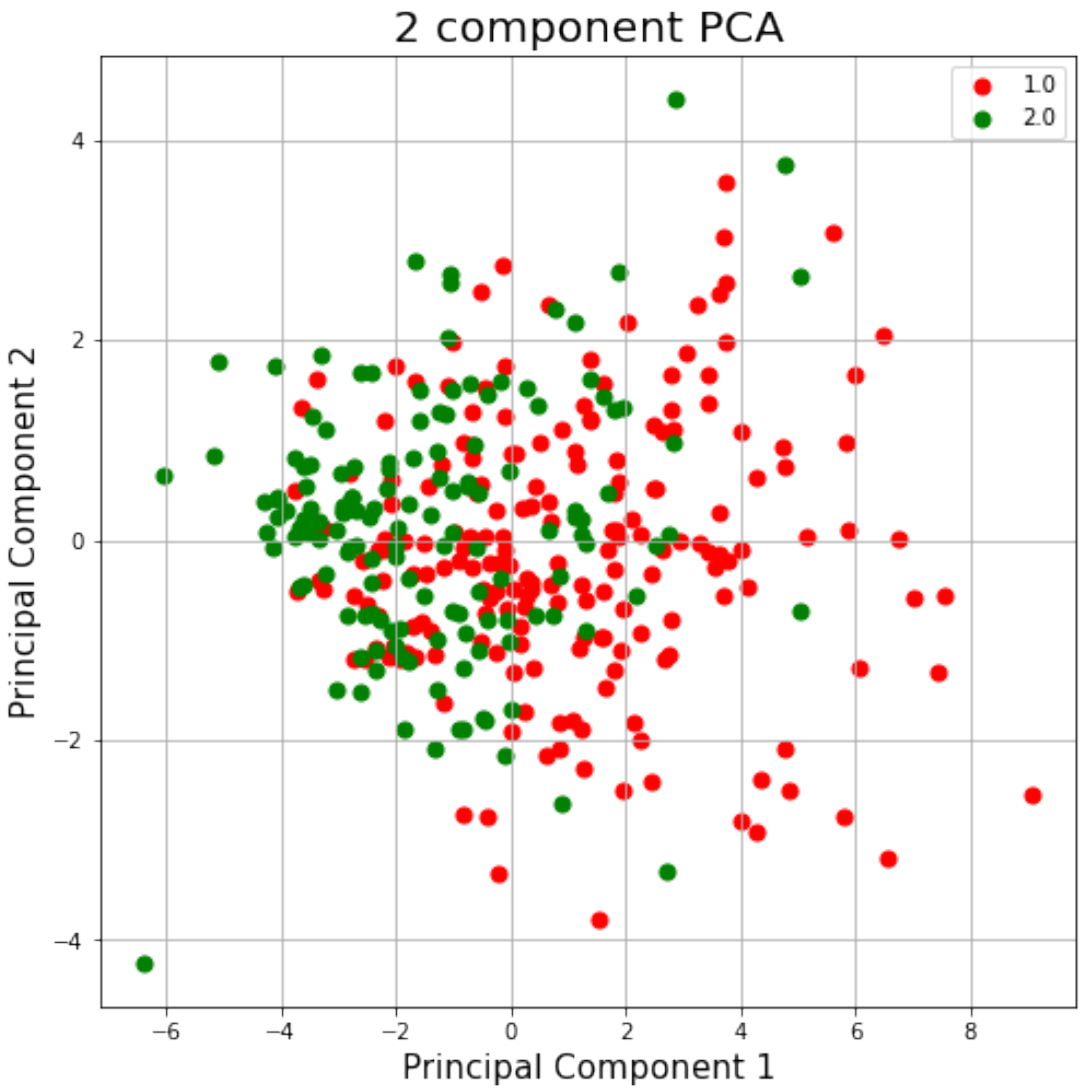
[10]:
```
# calculates pca with n_components = n_features and lists the results in descending␣
 ↪order
pca_model = PCA().fit(X)
pca_model.transform(X)
n_pcs = pca_model.components_.shape[0]
most_important_features = [
    np.abs(pca_model.components_[i]).argmax() for i in range(n_pcs)]
initial_feature_names = ml4bp.columns
most_important_names = [
    initial_feature_names[most_important_features[i]] for i in range(n_pcs)]
pca_result = {'PC{}'.format(i): most_important_names[i] for i in range(n_pcs)}
pca_result_DF = pd.DataFrame(pca_result.items(), columns=['PC', 'feature'])
# add all ratios of explained variances for each PC to DataFrame
pca_result_DF['variance ratio'] = pca_model.explained_variance_ratio_
pca_result_DF
```

[10]:

	PC	feature	variance ratio
0	PC0	CVLT_VFW1	0.436010
1	PC1	STROOP_FSB	0.108661
2	PC2	sex	0.071194
3	PC3	BMI	0.064334
4	PC4	CVLT_DG1	0.055407
5	PC5	BMI	0.048681
6	PC6	CVLT_WA1	0.037827
7	PC7	CVLT_WA1	0.035407
8	PC8	vereint_KL	0.032666
9	PC9	STROOP_INT	0.025062
10	PC10	TMT_B	0.024625
11	PC11	CVLT_A_LS_COR	0.020803
12	PC12	STROOP_FSB	0.014341
13	PC13	CVLT_DG5	0.010996
14	PC14	CVLT_VFW1	0.008884

```
15  PC15      CVLT_VFW2        0.005102
```

[11]:
```
# 2 component PCA and visualization in 2D-Plot
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
pcaDF = pd.DataFrame(data=principalComponents, columns=[
    'principal component 1', 'principal component 2'])
y_df = pd.DataFrame(y)
pca_result_DF = pd.concat([pcaDF, y], axis=1)

# plot
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(1, 1, 1)
ax.set_xlabel('Principal Component 1', fontsize=15)
ax.set_ylabel('Principal Component 2', fontsize=15)
ax.set_title('2 component PCA', fontsize=20)
targets = [1.0, 2.0]
colors = ['r', 'g']
for target, color in zip(targets, colors):
    indicesToKeep = pca_result_DF['group'] == target
    ax.scatter(pca_result_DF.loc[indicesToKeep, 'principal component 1'],
               pca_result_DF.loc[indicesToKeep, 'principal component 2'], c=color,␣
 ↪s=50)
ax.legend(targets)
ax.grid()
plt.show()
print("Components: \n" + str(abs(pca.components_)))
print("PCA explained variance ratio: "+str(pca.explained_variance_ratio_))
```

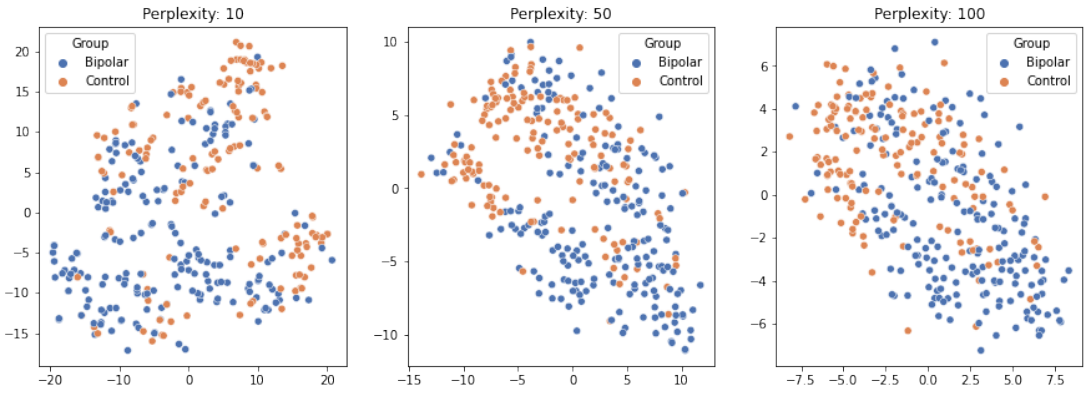



```
Components:
[[0.21540464 0.11651393 0.12806592 0.24093066 0.25721079 0.22083169
  0.20956156 0.25460504 0.27935676 0.15637777 0.32456945 0.3278435
  0.22900379 0.32037383 0.32373513 0.25827433]
 [0.11380785 0.20911653 0.04657783 0.23496649 0.24791147 0.37098692
  0.377936   0.3598652  0.18964392 0.02783    0.19791834 0.24665101
  0.26176008 0.28308908 0.2499337  0.25814141]]
PCA explained variance ratio: [0.43600968 0.10866097]
```

4.2 t-SNE

[12]:
```
# t-SNE plot of the date, perplexity influences the shape
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 5))
perplexities = [10, 50, 100]
axes = [ax1, ax2, ax3]
```

```
for i in range(3):
    tsne_DF = pd.DataFrame(X)
    tsne_DF['Group'] = y

    #fig = plt.figure(figsize=(10,20))
    tsne = manifold.TSNE(perplexity=perplexities[i], n_iter=1000, n_jobs=-
                         1, random_state=0, n_components=2)
    tsne = tsne.fit_transform(tsne_DF)
    # rename groups for correct labeling in plot
    tsne_DF.loc[(tsne_DF.Group == 1), 'Group'] = "Bipolar"
    tsne_DF.loc[(tsne_DF.Group == 2), 'Group'] = "Control"
    plot = sns.scatterplot(
        x=tsne[:, 0],
        y=tsne[:, 1],
        hue="Group",
        data=tsne_DF,
        palette='deep',
        ax=axes[i],
    )
    axes[i].set_title("Perplexity: "+str(perplexities[i]))

plt.show()
```



5 Algorithm selection using nested cross-validation

5.1 Classifiers and ParameterGrid's

[13]:
```
# classifiers
classifier_names = ("LogReg", "SVC", "AdaBoost",  "RF", "FFNN")
clf_logreg = LogisticRegression(random_state=0)
clf_svc = SVC(random_state=0)
clf_ada = AdaBoostClassifier(random_state=0)
clf_rf = RandomForestClassifier(random_state=0)
clf_nn = MLPClassifier(random_state=0)

# pipelines (redundant as data is already scaled)
# pipelines
```

```
pipeline_logreg = Pipeline([('std', StandardScaler()),
                            ('c_logreg', clf_logreg)])
pipeline_svc = Pipeline([('std', StandardScaler()),
                         ('c_svc', clf_svc)])
pipeline_ada = Pipeline([('std', StandardScaler()),
                         ('c_ada', clf_ada)])
pipeline_rf = Pipeline([('std', StandardScaler()),
                        ('c_rf', clf_rf)])
pipeline_nn = Pipeline([('std', StandardScaler()),
                        ('c_nn', clf_nn)])

# Parameter Grids
# Logistic Regression
param_grid_logreg = [{'c_logreg__penalty': ['l1', 'l2', 'elasticnet'],
                      'c_logreg__C': np.logspace(-4, 4, 50),
                      'c_logreg__solver': ['newton-cg', 'lbfgs', 'liblinear'],
                      }]

# SVC
param_grid_svc = [{'c_svc__C': np.append(np.logspace(1, -3, 50), [10, 1, 0.1]),
                   'c_svc__gamma': np.append(np.logspace(1, -3, 50), [10, 1, 0.1]),
                   'c_svc__kernel': ['rbf', 'poly', 'linear'],
                   'c_svc__degree': [1, 2, 3, 4, 5]}]
# adaboost
param_grid_ada = {'c_ada__n_estimators': [10, 50, 100, 200, 500, 1000, 2000, 3000,␣
 ↪4000, 5000],
                  'c_ada__learning_rate': [.001, 0.01, .1, 1]}

# random forrest
param_grid_rf = {
    'c_rf__bootstrap': [True, False],
    'c_rf__max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
    'c_rf__max_features': ['sqrt'],
    'c_rf__min_samples_leaf': [1, 2, 4],
    'c_rf__min_samples_split': [2, 5, 10],
    'c_rf__n_estimators': [10, 20, 30, 50, 100, 150, 200, 400],
    'c_rf__oob_score': [True, False]}  # , 600, 800, 1000, 1200, 1400, 1600, 1800,␣
 ↪2000]}

# neuronal net
param_grid_nn = {'c_nn__hidden_layer_sizes': [(100,), (50,), (100, 100), (50, 50),␣
 ↪(100, 50), (100, 100, 100), (50, 50, 50), (100, 50, 50), (10, 30, 10), (20,)],
                 'c_nn__activation': ['tanh', 'relu'],
                 'c_nn__solver': ['sgd', 'adam'],
                 'c_nn__alpha': [0.0001, 0.05],
                 'c_nn__learning_rate': ['constant', 'adaptive'], }
```

5.1.1 inner CV

[14]:
```
# define folds
inner_cv = KFold(n_splits=5, shuffle=True, random_state=0)

gridcvs = {}
```

```
# create inner folds
for pgrid, est, name in zip((param_grid_logreg, param_grid_svc, param_grid_ada,␣
 ↪param_grid_rf, param_grid_nn),
                            (pipeline_logreg, pipeline_svc,
                             pipeline_ada, pipeline_rf, pipeline_nn),
                            classifier_names):

    # gridsearch by optimizing for model with best f1-score for label '1'=biploar
    # (no averaging)
    gridsearch = GridSearchCV(estimator=est,
                              param_grid=pgrid,
                              scoring='f1_macro',
                              n_jobs=-1,
                              cv=inner_cv,
                              verbose=1,
                              refit=True)
    gridcvs[name] = gridsearch
```

5.1.2 outer CV

[15]:
```
# create outer folds
outer_cv = KFold(n_splits=10, shuffle=True, random_state=0)
# scores to calculate
scores = ["accuracy", "f1", "f1_macro", "f1_micro", "f1_weighted", 'roc_auc']

for name, gs_estimator in gridcvs.items():
    # cross validate and calculate all scores
    nested_score = cross_validate(gs_estimator,
                                  X=X,
                                  y=y,
                                  cv=outer_cv,
                                  n_jobs=-1,
                                  verbose=1,
                                  scoring=scores)
    # print detailed scores
    # print(nested_score)
    print(name)
    # print meam of all scores + stddev
    for score in scores:
        print("{}: score: {:.2f} +/- {:.2f} 95% CI {}".format(score,
                                                              ␣
 ↪round(nested_score["test_" +
                                                                                ␣
 ↪score].mean(), 2),
                                                              ␣
 ↪round(nested_score["test_" +
                                                                                ␣
 ↪score].std(), 2),
                                                              ␣
 ↪calc_ci(nested_score["test_"+score])))
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
```

```
[Parallel(n_jobs=-1)]: Done   6 out of  10 | elapsed:   32.6s remaining:   21.7s
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   49.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.

LogReg
accuracy: score: 0.70 +/- 0.06 95% CI [0.67-0.74]
f1: score: 0.72 +/- 0.07 95% CI [0.68-0.77]
f1_macro: score: 0.69 +/- 0.06 95% CI [0.66-0.73]
f1_micro: score: 0.70 +/- 0.06 95% CI [0.67-0.74]
f1_weighted: score: 0.70 +/- 0.06 95% CI [0.67-0.74]
roc_auc: score: 0.78 +/- 0.06 95% CI [0.75-0.82]

[Parallel(n_jobs=-1)]: Done   6 out of  10 | elapsed: 75.7min remaining: 50.5min

SVC
accuracy: score: 0.69 +/- 0.07 95% CI [0.65-0.73]
f1: score: 0.73 +/- 0.07 95% CI [0.69-0.78]
f1_macro: score: 0.68 +/- 0.06 95% CI [0.64-0.72]
f1_micro: score: 0.69 +/- 0.07 95% CI [0.65-0.73]
f1_weighted: score: 0.69 +/- 0.07 95% CI [0.65-0.73]
roc_auc: score: 0.77 +/- 0.07 95% CI [0.73-0.81]

[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed: 103.4min finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   6 out of  10 | elapsed: 24.5min remaining: 16.3min

AdaBoost
accuracy: score: 0.69 +/- 0.05 95% CI [0.66-0.72]
f1: score: 0.73 +/- 0.07 95% CI [0.69-0.78]
f1_macro: score: 0.67 +/- 0.05 95% CI [0.64-0.69]
f1_micro: score: 0.69 +/- 0.05 95% CI [0.66-0.72]
f1_weighted: score: 0.68 +/- 0.05 95% CI [0.65-0.71]
roc_auc: score: 0.78 +/- 0.04 95% CI [0.75-0.8]

[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed: 37.9min finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   6 out of  10 | elapsed: 109.1min remaining:
72.7min

RF
accuracy: score: 0.69 +/- 0.06 95% CI [0.65-0.73]
f1: score: 0.74 +/- 0.07 95% CI [0.7-0.78]
f1_macro: score: 0.68 +/- 0.06 95% CI [0.64-0.71]
f1_micro: score: 0.69 +/- 0.06 95% CI [0.65-0.73]
f1_weighted: score: 0.69 +/- 0.06 95% CI [0.65-0.73]
roc_auc: score: 0.77 +/- 0.07 95% CI [0.73-0.81]

[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed: 171.5min finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   6 out of  10 | elapsed: 18.4min remaining: 12.3min

FFNN
accuracy: score: 0.70 +/- 0.08 95% CI [0.65-0.74]
f1: score: 0.73 +/- 0.09 95% CI [0.68-0.79]
f1_macro: score: 0.68 +/- 0.07 95% CI [0.64-0.73]
f1_micro: score: 0.70 +/- 0.08 95% CI [0.65-0.74]
f1_weighted: score: 0.70 +/- 0.08 95% CI [0.65-0.74]
roc_auc: score: 0.76 +/- 0.06 95% CI [0.72-0.8]
```

```
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed: 26.9min finished
```

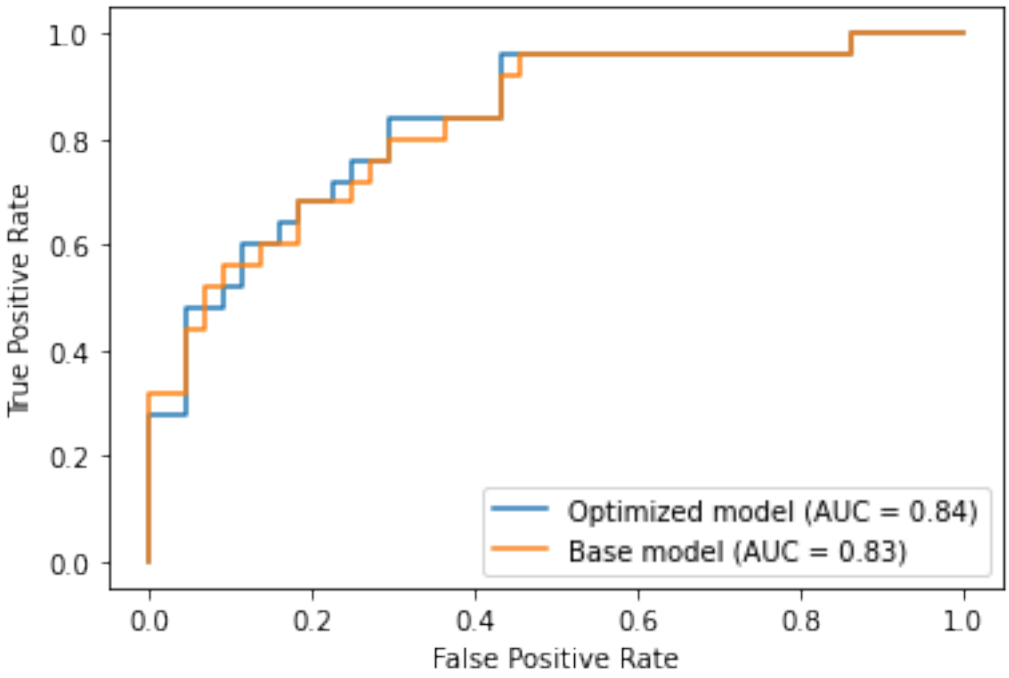
5.2 Optimiziation using Grid Search

An 80/20 train-test-split is used, random_state=0 for reproduce the results during multiple runs

[16]:
```
# defining the models
base_model = LogisticRegression(random_state=0)
tuned_model = LogisticRegression(random_state=0)

# split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0)

# training and scoring of base model
base_model.fit(X_train, y_train)
print("Base model")
print(classification_report(base_model.predict(X_test), y_test))
print("AUC: {:.3f}".format(roc_auc_score(
    y_test, base_model.decision_function(X_test))))

# optimzing the second model using gridsearch and scoring
# defining the parameter grid
param_grid = [{'clf__penalty': ['l1', 'l2', 'elasticnet'],
               'clf__C': np.logspace(-4, 4, 50),
               'clf__solver': ['newton-cg', 'lbfgs', 'liblinear'],
               }]

# making a pipeline (redundant due to prior scaling)
pipeline = Pipeline([('std', StandardScaler()),
                     ('clf', tuned_model)])

# gridsearch, optimizing for best label "1" f1-score
grid = GridSearchCV(pipeline, param_grid, scoring='f1_macro',
                    refit=True, verbose=1, n_jobs=-1, cv=5)
grid.fit(X_train, y_train)

# scoring and best parameters
print("Tuned model")
print(classification_report(grid.predict(X_test), y_test))
print(grid.best_params_)
print(grid.best_estimator_)
print("AUC: {:.3f}".format(roc_auc_score(
    y_test, grid.decision_function(X_test))))

# plot ROC curves for base and optimized model
ax = plt.gca()
# calculate ROC's
gridroccurve = plot_roc_curve(grid.best_estimator_, X_test, y_test, ax=ax)
baseroccurve = plot_roc_curve(base_model, X_test, y_test, ax=ax)
# get legend and set custom labels
legend = plt.legend().get_texts()
# some string magic to get AUROC from original legend
legend[0].set_text('Optimized model (AUC = {})'.format(
```

```
    str(legend[0]).split()[-1][0:4]))
legend[1].set_text('Base model (AUC = {})'.format(
    str(legend[1]).split()[-1][0:4]))
plt.show()
```

```
Base model
              precision    recall  f1-score   support

         1.0       0.82      0.80      0.81        45
         2.0       0.64      0.67      0.65        24

    accuracy                           0.75        69
   macro avg       0.73      0.73      0.73        69
weighted avg       0.76      0.75      0.75        69

AUC: 0.831
Fitting 5 folds for each of 450 candidates, totalling 2250 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  60 tasks      | elapsed:    0.5s
[Parallel(n_jobs=-1)]: Done 2146 tasks      | elapsed:    4.4s
[Parallel(n_jobs=-1)]: Done 2250 out of 2250 | elapsed:    4.6s finished
Tuned model
              precision    recall  f1-score   support

         1.0       0.82      0.82      0.82        44
         2.0       0.68      0.68      0.68        25

    accuracy                           0.77        69
   macro avg       0.75      0.75      0.75        69
weighted avg       0.77      0.77      0.77        69

{'clf__C': 0.18420699693267145, 'clf__penalty': 'l2', 'clf__solver':
'liblinear'}
Pipeline(steps=[('std', StandardScaler()),
                ('clf',
                 LogisticRegression(C=0.18420699693267145, random_state=0,
                                    solver='liblinear'))])
AUC: 0.835
```



[17]:
```
# Scoring of the base model with cross-validation as estimation of genealization␣
 ↪performance
print("Base model")
base_model = LogisticRegression(random_state=0)
scores = cross_val_score(base_model, X, y, scoring='f1_micro')
print("5-fold F1_micro: {} +- {} {}".format(round(scores.mean(), 2),
                                            round(scores.std(), 2), calc_ci(scores)))
scores = cross_val_score(base_model, X, y, scoring='f1_macro')
print("5-fold F1_macro: {} +- {} {}".format(round(scores.mean(), 2),
                                            round(scores.std(), 2), calc_ci(scores)))

# Scoring of the optimized model with cross-validation as estimation of␣
 ↪genealization performance
print("Tuned model")
scores = cross_val_score(grid.best_estimator_, X, y, scoring='f1_micro')
print("5-fold F1_micro: {} +- {} {}".format(round(scores.mean(), 2),
                                            round(scores.std(), 2), calc_ci(scores)))
scores = cross_val_score(grid.best_estimator_, X, y, scoring='f1_macro')
print("5-fold F1_macro: {} +- {} {}".format(round(scores.mean(), 2),
                                            round(scores.std(), 2), calc_ci(scores)))
```

```
Base model
5-fold F1_micro: 0.69 +- 0.07 [0.63-0.75]
5-fold F1_macro: 0.68 +- 0.07 [0.62-0.74]
Tuned model
5-fold F1_micro: 0.68 +- 0.07 [0.63-0.74]
5-fold F1_macro: 0.67 +- 0.06 [0.62-0.73]
```

5.3 Optimization using Feature Selection

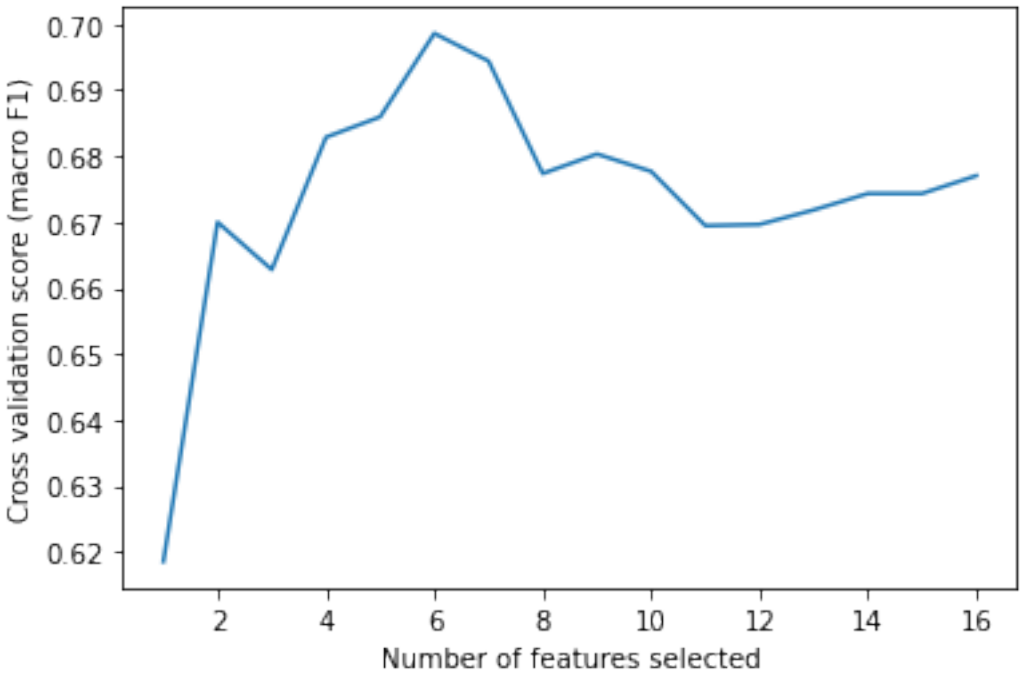
5.3.1 Recursive feature elimination with cross-validation

Evaluation using base model and best model from gridsearch

[18]:
```
base_model = LogisticRegression(random_state=0)
model = LogisticRegression(random_state=0)

rfecv = RFECV(estimator=LogisticRegression(), step=1,
              cv=5, scoring='f1_macro', n_jobs=-1,)
#rfecv.fit(X_test, y_test)
rfecv.fit(X, y)
print("Optimal number of features : %d" % rfecv.n_features_)
print(
    "Max. macro-avg F1 score: {:.2f}".format(round(np.max(rfecv.grid_scores_), 2)))

# Plot number of features VS. cross-validation scores
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (macro F1)")
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
plt.show()

# print and sort features by rank, rank=1 for all selected features
rfe_selected = pd.DataFrame({'rank': rfecv.ranking_, 'feature': ml4bp.columns})
rfe_selected.sort_values(by=['rank'], inplace=True)
print(rfe_selected)

# reduce data set to selected features
X_reduced = rfecv.transform(X)


# performance comparison
# base model, all features, no optimization
print("\nBase model: all 16 features, no prior hyperparameter tuning")
base_model.fit(X, y)
base_scores = cross_val_score(base_model, X, y, scoring='f1_micro')
print("5-fold F1_micro for complete data set: {} +- {} {}".format(
    round(base_scores.mean(), 2), round(base_scores.std(), 2), calc_ci(base_scores)))
base_scores = cross_val_score(base_model, X, y, scoring='f1_macro')
print("5-fold F1_macro for complete data set: {} +- {} {}".format(
    round(base_scores.mean(), 2), round(base_scores.std(), 2), calc_ci(base_scores)))

# base model after reduced data set by RFECV
print("\nBase model: reduced to {} features, no prior hyperparameter tuning".format(
    rfecv.n_features_))
base_model.fit(X_reduced, y)
base_reduced_score = cross_val_score(
    base_model, X_reduced, y, scoring='f1_micro')
print("5-fold F1_micro for reduced data set: {} +- {} {}".format(
    round(base_reduced_score.mean(), 2), round(base_reduced_score.std(), 2),␣
 ↪calc_ci(base_reduced_score)))
base_reduced_score = cross_val_score(
    base_model, X_reduced, y, scoring='f1_macro')
```

```
print("5-fold F1_macro for reduced data set: {} +- {} {}".format(
    round(base_reduced_score.mean(), 2), round(base_reduced_score.std(), 2),␣
 ↪calc_ci(base_reduced_score)))

# RFEVC model with reduced data set
print("\nRFECV model with {} features, no prior hyperparameter tuning".format(
    rfecv.n_features_))
rfecv_scores = cross_val_score(rfecv, X_reduced, y, scoring='f1_micro')
print("5-fold F1_micro for reduced data set: {} +- {} {}".format(
    round(rfecv_scores.mean(), 2), round(rfecv_scores.std(), 2),␣
 ↪calc_ci(rfecv_scores)))
rfecv_scores = cross_val_score(rfecv, X_reduced, y, scoring='f1_macro')
print("5-fold F1_macro for reduced data set: {} +- {} {}".format(
    round(rfecv_scores.mean(), 2), round(rfecv_scores.std(), 2),␣
 ↪calc_ci(rfecv_scores)))

# gridsearch best model with complete data set
print("\nOptimized model from grid search with all 16 features".format(rfecv.
 ↪n_features_))
grid_scores_complete = cross_val_score(
    grid.best_estimator_['clf'], X, y, scoring='f1_micro')
print("5-fold F1_micro for complete data set: {} +- {} {}".format(
    round(grid_scores_complete.mean(), 2), round(grid_scores_complete.std(), 2),␣
 ↪calc_ci(grid_scores_complete)))
grid_scores_complete = cross_val_score(
    grid.best_estimator_['clf'], X, y, scoring='f1_macro')
print("5-fold F1_macro for complete data set: {} +- {} {}".format(
    round(grid_scores_complete.mean(), 2), round(grid_scores_complete.std(), 2),␣
 ↪calc_ci(grid_scores_complete)))

# gridsearch best model with reduced data set
print("\nOptimized model from grid search with {} features".format(rfecv.
 ↪n_features_))
grid_scores_reduced = cross_val_score(
    grid.best_estimator_['clf'], X_reduced, y, scoring='f1_micro')
print("5-fold F1_micro for reduced data set: {} +- {} {}".format(
    round(grid_scores_reduced.mean(), 2), round(grid_scores_reduced.std(), 2),␣
 ↪calc_ci(grid_scores_reduced)))
grid_scores_reduced = cross_val_score(
    grid.best_estimator_['clf'], X_reduced, y, scoring='f1_macro')
print("5-fold F1_macro for reduced data set: {} +- {} {}".format(
    round(grid_scores_reduced.mean(), 2), round(grid_scores_reduced.std(), 2),␣
 ↪calc_ci(grid_scores_reduced)))
```

```
Optimal number of features : 6
Max. macro-avg F1 score: 0.70
```



	rank	feature
2	1	BMI
3	1	TMT_A
7	1	STROOP_INT
12	1	CVLT_WA1
14	1	CVLT_WA2
15	1	vereint_KL
11	2	CVLT_VFW1
13	3	CVLT_VFW2
0	4	age
1	5	sex
6	6	STROOP_FSB
5	7	STROOP_FWL
8	8	CVLT_A_LS_COR
9	9	CVLT_DG1
4	10	TMT_B
10	11	CVLT_DG5

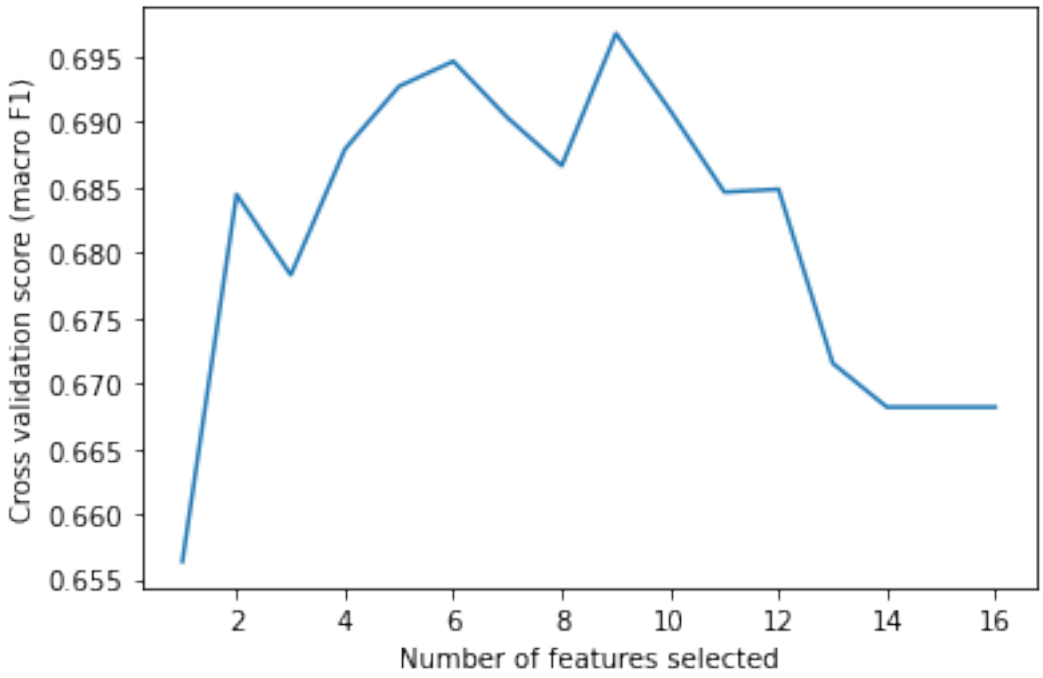
```
Base model: all 16 features, no prior hyperparameter tuning
5-fold F1_micro for complete data set: 0.69 +- 0.07 [0.63-0.75]
5-fold F1_macro for complete data set: 0.68 +- 0.07 [0.62-0.74]

Base model: reduced to 6 features, no prior hyperparameter tuning
5-fold F1_micro for reduced data set: 0.72 +- 0.07 [0.66-0.77]
5-fold F1_macro for reduced data set: 0.71 +- 0.07 [0.65-0.77]

RFECV model with 6 features, no prior hyperparameter tuning
5-fold F1_micro for reduced data set: 0.7 +- 0.07 [0.64-0.76]
5-fold F1_macro for reduced data set: 0.69 +- 0.07 [0.63-0.75]
```

```
Optimized model from grid search with all 16 features
5-fold F1_micro for complete data set: 0.68 +- 0.07 [0.62-0.74]
5-fold F1_macro for complete data set: 0.67 +- 0.06 [0.61-0.73]

Optimized model from grid search with 6 features
5-fold F1_micro for reduced data set: 0.72 +- 0.08 [0.65-0.79]
5-fold F1_macro for reduced data set: 0.71 +- 0.08 [0.64-0.78]
```

5.3.2 RFECV from scratch with best estimator from grid search

[19]:
```
# rfecv with best estimator from grid
rfecv = RFECV(estimator=grid.best_estimator_[
              'clf'], step=1, cv=5, scoring='f1_macro', n_jobs=-1,)
#rfecv.fit(X_test, y_test)
rfecv.fit(X, y)
print("Optimal number of features : %d" % rfecv.n_features_)
print(
    "Max. macro-avg F1 score {:.2f}".format(round(np.max(rfecv.grid_scores_), 2)))


# Plot number of features VS. cross-validation scores
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (macro F1)")
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
plt.show()

# print and sort features by rank, rank=1 for all selected features
rfe_selected = pd.DataFrame({'rank': rfecv.ranking_, 'feature': ml4bp.columns})
rfe_selected.sort_values(by=['rank'], inplace=True)
print(rfe_selected)

X_reduced = rfecv.transform(X)

# gridsearch best model with reduced data set
print("\nGrid search best model with features reduced by RFECV: reduced to {}␣
 ↪features, no prior hyperparameter tuning".format(
    rfecv.n_features_))
base_scores = cross_val_score(rfecv, X_reduced, y, scoring='f1_micro')
print("5-fold F1_micro for reduced data set: {} +- {} {}".format(
    round(base_scores.mean(), 2), round(base_scores.std(), 2), calc_ci(base_scores)))
base_scores = cross_val_score(rfecv, X_reduced, y, scoring='f1_macro')
print("5-fold F1_macro for reduced data set: {} +- {} {}".format(
    round(base_scores.mean(), 2), round(base_scores.std(), 2), calc_ci(base_scores)))
```

```
Optimal number of features : 9
Max. macro-avg F1 score 0.70
```



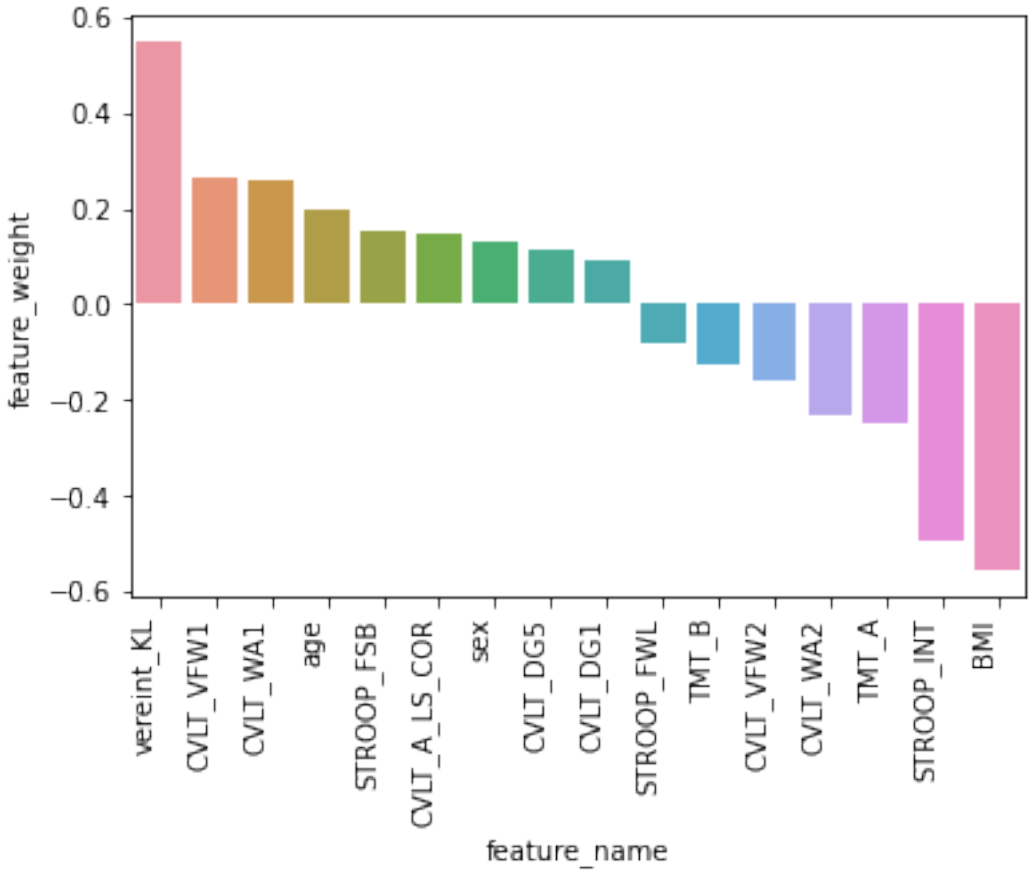
	rank	feature
1	1	sex
2	1	BMI
3	1	TMT_A
7	1	STROOP_INT
11	1	CVLT_VFW1
12	1	CVLT_WA1
13	1	CVLT_VFW2
14	1	CVLT_WA2
15	1	vereint_KL
0	2	age
9	3	CVLT_DG1
8	4	CVLT_A_LS_COR
4	5	TMT_B
6	6	STROOP_FSB
5	7	STROOP_FWL
10	8	CVLT_DG5

Grid search best model with features reduced by RFECV: reduced to 9 features, no prior hyperparameter tuning
5-fold F1_micro for reduced data set: 0.71 +- 0.08 [0.64-0.77]
5-fold F1_macro for reduced data set: 0.7 +- 0.08 [0.63-0.76]

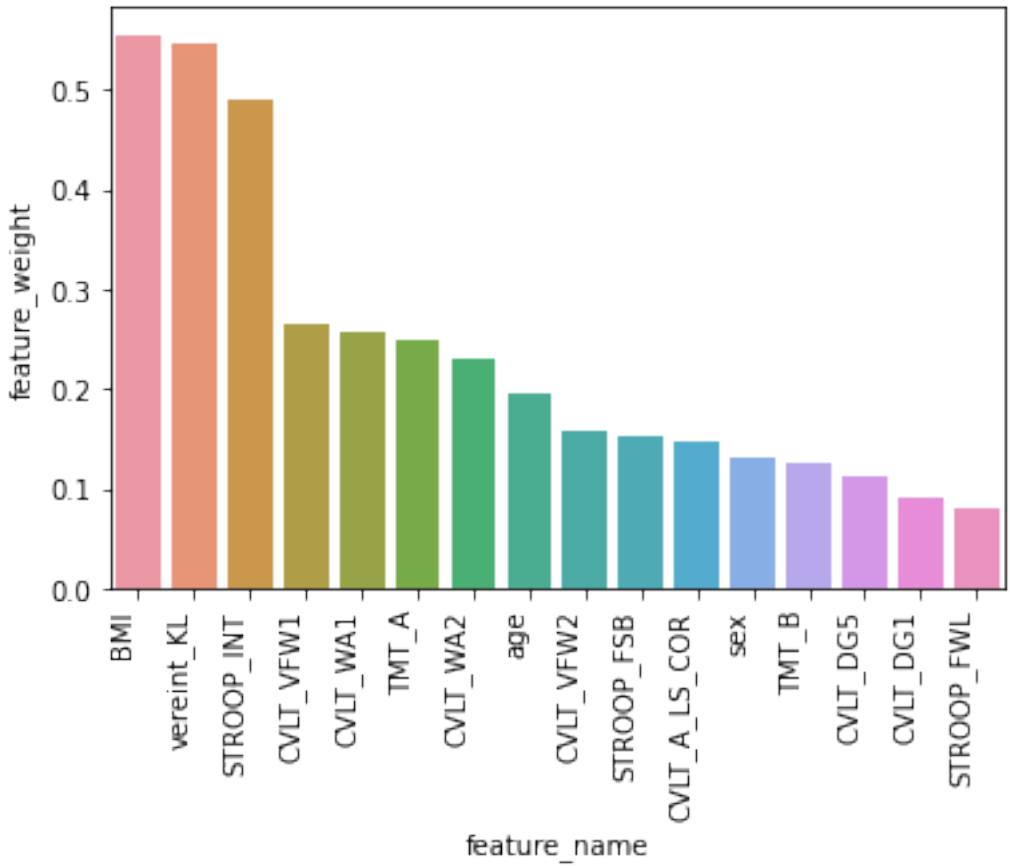
6 Feature analysis

6.1 Feature ranking

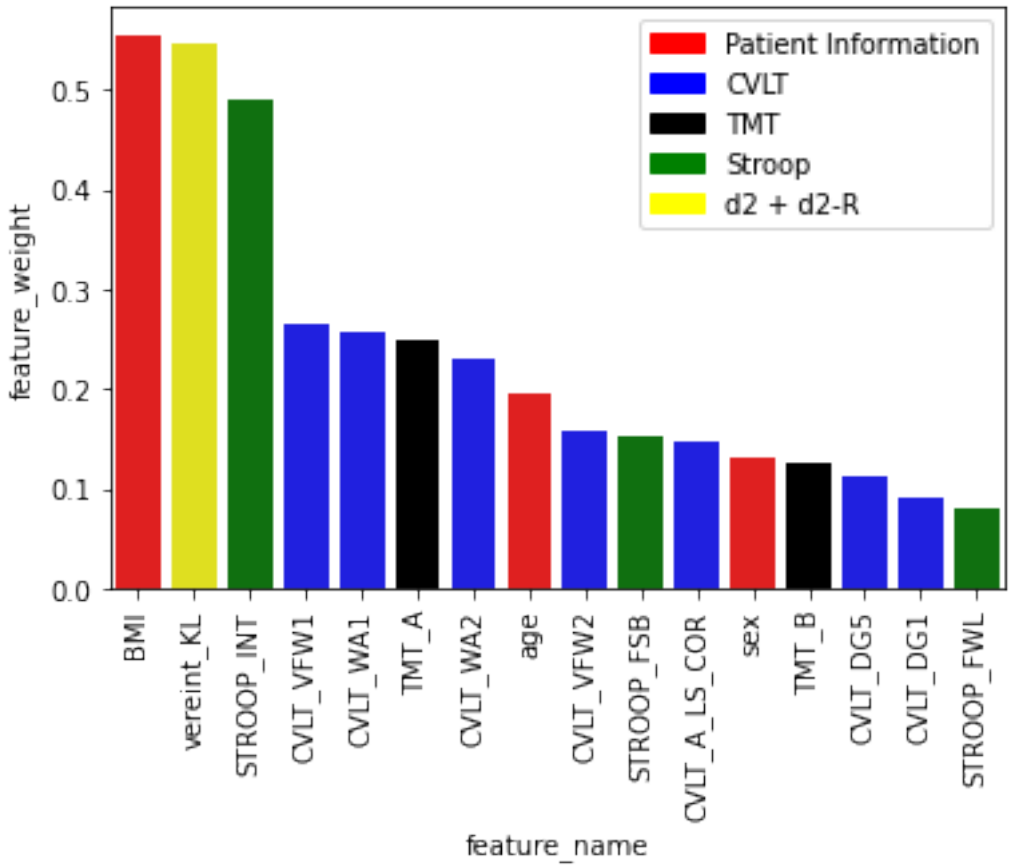
[20]:
```
# get feature importances from gridsearch optimized model
# absolute values are used to sort from highest coef
feature_importances = grid.best_estimator_['clf'].coef_[0]

# signed values
feature_df = pd.DataFrame({'feature_name': ml4bp.columns,
                           'feature_weight': feature_importances})
plot = sns.barplot(y='feature_weight',
                   x='feature_name',
                   data=feature_df.sort_values(['feature_weight'], ascending=False))
plot.set_xticklabels(plot.get_xticklabels(), rotation=90,
                     horizontalalignment='right')
plt.show()

print(feature_df.sort_values(by=['feature_weight'], ascending=False))

# absolute values
feature_importances = np.abs(feature_importances)
feature_df = pd.DataFrame({'feature_name': ml4bp.columns,
                           'feature_weight': feature_importances})
plot = sns.barplot(y='feature_weight',
                   x='feature_name',
                   data=feature_df.sort_values(['feature_weight'], ascending=False))
plot.set_xticklabels(plot.get_xticklabels(), rotation=90,
                     horizontalalignment='right')
plt.show()
```



	feature_name	feature_weight
15	vereint_KL	0.546738
11	CVLT_VFW1	0.265095
12	CVLT_WA1	0.257938
0	age	0.196262
6	STROOP_FSB	0.152752
8	CVLT_A_LS_COR	0.146818
1	sex	0.131749
10	CVLT_DG5	0.112270
9	CVLT_DG1	0.089955
5	STROOP_FWL	-0.079407
4	TMT_B	-0.124755
13	CVLT_VFW2	-0.157302
14	CVLT_WA2	-0.229595
3	TMT_A	-0.248136
7	STROOP_INT	-0.491695
2	BMI	-0.555347



[21]:
```
# hardcoded the palette for this specific feature ranking to save time
palette = sns.color_palette(['r', 'yellow', 'g', 'b', 'b', 'black',
                             'b', 'r', 'b', 'g', 'b', 'r', 'black', 'b', 'b', 'g'],␣
 ↪16)

feature_df = pd.DataFrame({'feature_name': ml4bp.columns,
                           'feature_weight': feature_importances})  # ),
#                            'group': group})
plot = sns.barplot(y='feature_weight',
                   x='feature_name',
                   data=feature_df.sort_values(['feature_weight'], ascending=False),␣
 ↪palette=palette)
plot.set_xticklabels(plot.get_xticklabels(), rotation=90,)

# legend with colors
patinfo = mpatches.Patch(color='r', label='Patient Information')
cvlt = mpatches.Patch(color='b', label='CVLT')
tmt = mpatches.Patch(color='black', label='TMT')
stroop = mpatches.Patch(color='g', label='Stroop')
d2 = mpatches.Patch(color='yellow', label='d2 + d2-R')
plt.legend(handles=[patinfo, cvlt, tmt, stroop, d2])
plt.show()
```



6.2 Model analysis with ELI5

[22]:
```
# show ELI5 weights, values are the same as the coefficients in feature ranking
eli5.show_weights(grid.best_estimator_, feature_names=list(ml4bp.columns))
```

[22]: <IPython.core.display.HTML object>

[23]:
```
# re-read the original data to get the Pat_ID (dropped previously)
# check which OS to load from correct path
os = platform.system()
if os == 'Darwin':
    # path to vault for macOS
    ml4bp_forpateid = pd.read_csv(
        "/Volumes/tresorML4BP/Data/BIPFAT 15.09.2020 kOGINTION_SELCTED.CSV", sep=';')
# Windows
else:
    ml4bp_forpateid = pd.read_csv(
        "G:\Data\BIPFAT 15.09.2020 kOGINTION_SELCTED.CSV", sep=';')

# get a feature to check by index from y_test (test subset)
# feature index 0 = first sample from test data set
feature_index = 0
```

```
sample = pd.DataFrame(y_test).iloc[feature_index]
# print Pat_ID and group
print("Pat_ID: {} group: {}".format(
    ml4bp_forpateid.iloc[sample.name]['Pat_ID'], sample[0]))
# print predict probas for sample
print(grid.best_estimator_['clf'].predict_proba(
    X_test[feature_index:feature_index+1]))
# print eli5 explanation
eli5.show_prediction(grid.best_estimator_['clf'], doc=X_test[feature_index],␣
 ↪feature_names=list(
    ml4bp.columns), show_feature_values=True)
```

```
Pat_ID: 95.0 group: 2.0
[[0.64509196 0.35490804]]
```

[23]: <IPython.core.display.HTML object>

```
# get a feature to check by index from y_test (test subset)
# feature index 1 = second sample from test daata set
feature_index = 1
sample = pd.DataFrame(y_test).iloc[feature_index]
# print Pat_ID and group
print("Pat_ID: {} group: {}".format(
    ml4bp_forpateid.iloc[sample.name]['Pat_ID'], sample[0]))
# print predict probas for sample
print(grid.best_estimator_['clf'].predict_proba(
    X_test[feature_index:feature_index+1]))
# print eli5 explanation
eli5.show_prediction(grid.best_estimator_['clf'], doc=X_test[feature_index],␣
 ↪feature_names=list(
    ml4bp.columns), show_feature_values=True)
```

```
Pat_ID: 125.0 group: 2.0
[[0.10928839 0.89071161]]
```

[24]: <IPython.core.display.HTML object>

6.3 Analysis of the top 3 features

```
# building a new DF
topDF = pd.DataFrame(data=[ml4bp.vereint_KL, ml4bp.BMI, ml4bp.STROOP_INT, y]).T
topDF
```

[25]:

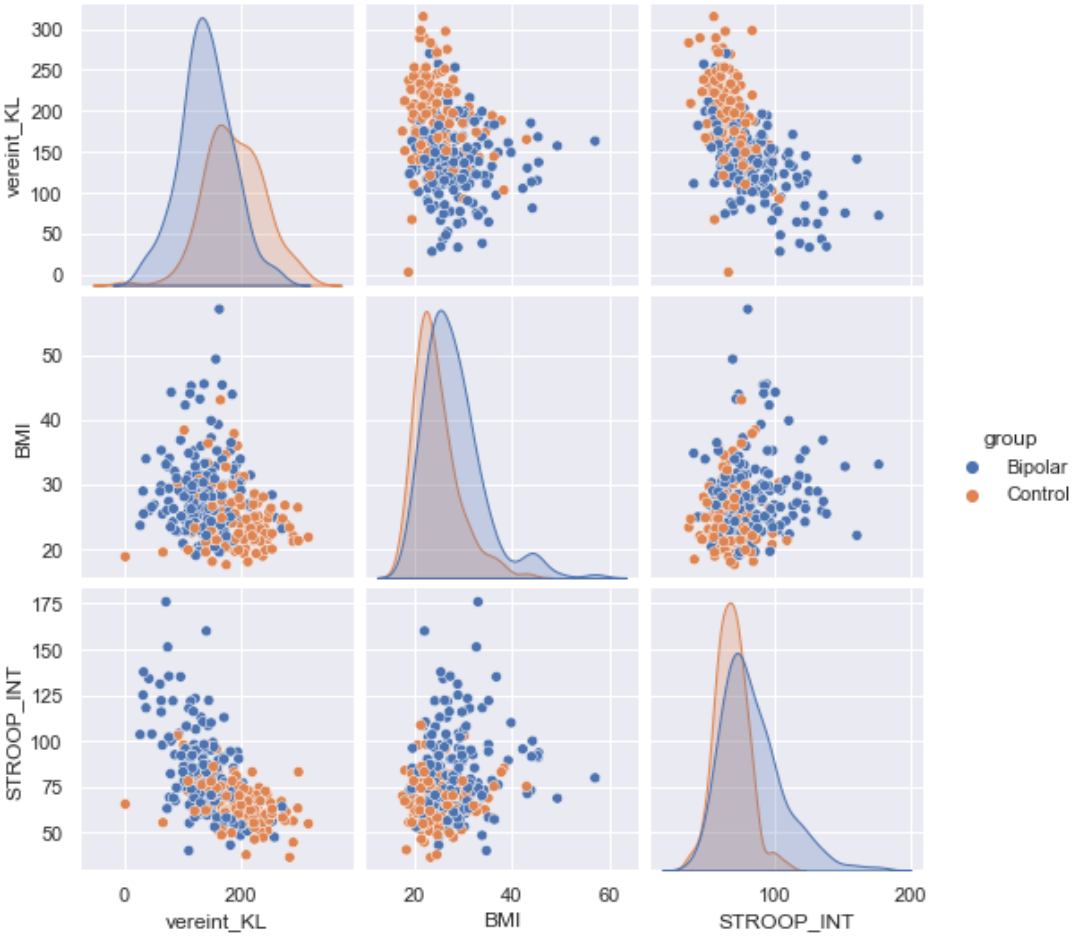
	vereint_KL	BMI	STROOP_INT	group
0	122.0	30.93	123.2	1.0
1	192.0	33.04	77.0	1.0
2	115.0	30.83	117.0	1.0
3	131.0	26.88	71.0	1.0
4	109.0	29.37	90.0	1.0
..	...	...	...	...
336	149.0	24.24	70.3	1.0
337	141.0	22.08	160.0	1.0

```
338       315.0  21.80        54.7    2.0
339         NaN    NaN        66.0    2.0
340       219.0  22.00        83.0    2.0

[341 rows x 4 columns]
```

[26]:
```
# required to activate seaborn design, unknown bug
sns.set()

# decsriptive statistics again (same as above)
# assigning labels to groups for better readability for pairplot
topDF.loc[topDF['group'] == 1, 'group'] = 'Bipolar'
topDF.loc[topDF['group'] == 2, 'group'] = 'Control'

# pairplot to visualize possible correlations
sns.pairplot(topDF, hue='group')

topDF.describe()
```

[26]:

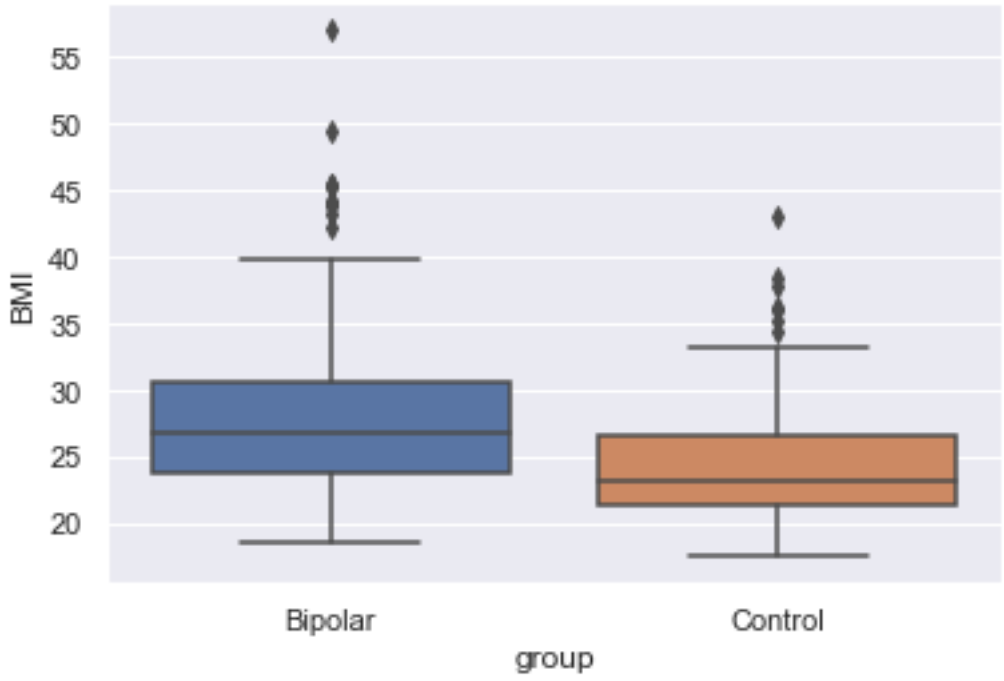
	vereint_KL	BMI	STROOP_INT
count	322.000000	336.000000	334.000000
mean	160.225776	26.426964	75.906467
std	52.746820	5.760030	20.059469
min	2.700000	17.530000	36.300000
25%	124.250000	22.487500	62.430000
50%	156.500000	25.150000	72.195000
75%	194.000000	29.032500	84.190000
max	315.000000	57.130000	176.000000



6.4 Boxplots of top three features and the least important

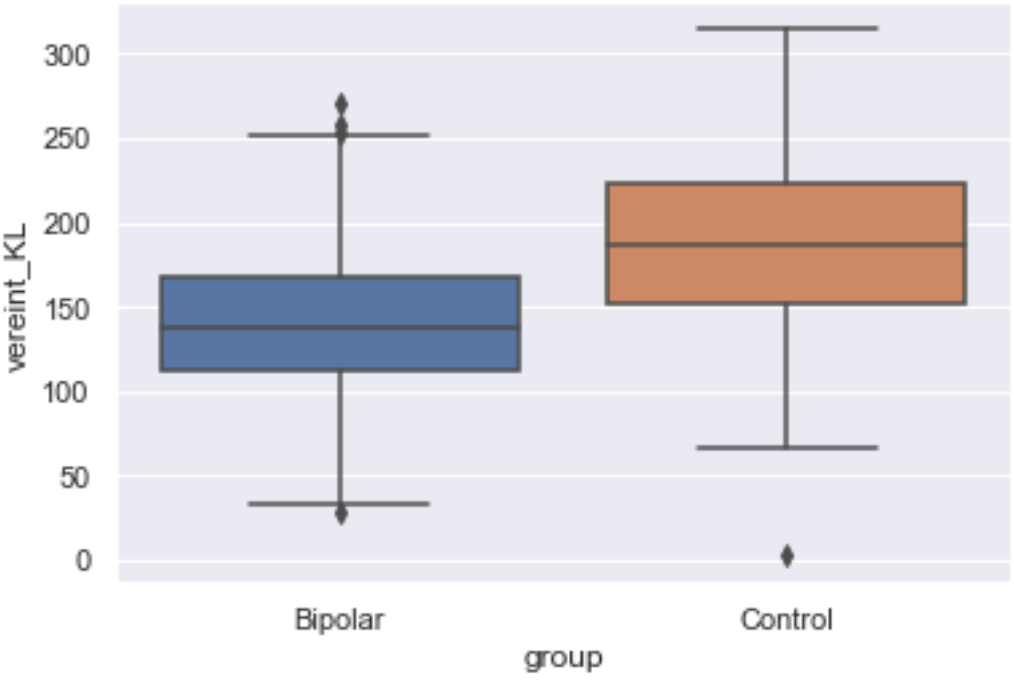
[27]:
```
sns.boxplot(x=topDF.group, y=topDF.BMI)
```

[27]:
```
<AxesSubplot:xlabel='group', ylabel='BMI'>
```



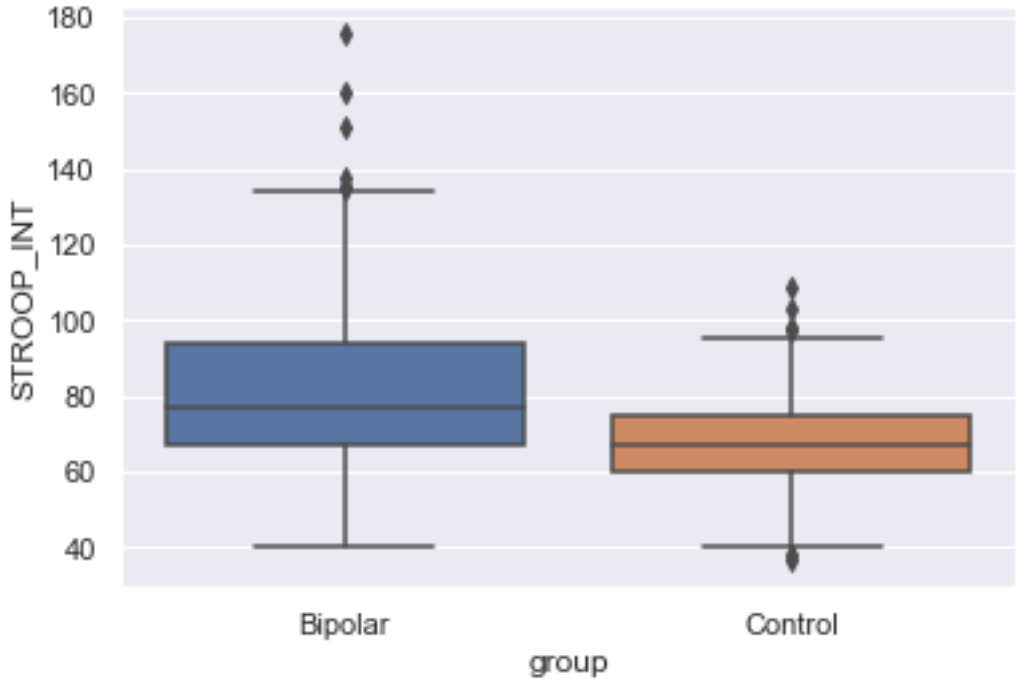
[28]:
```
sns.boxplot(x=topDF.group, y=topDF.vereint_KL)
```

[28]:
```
<AxesSubplot:xlabel='group', ylabel='vereint_KL'>
```



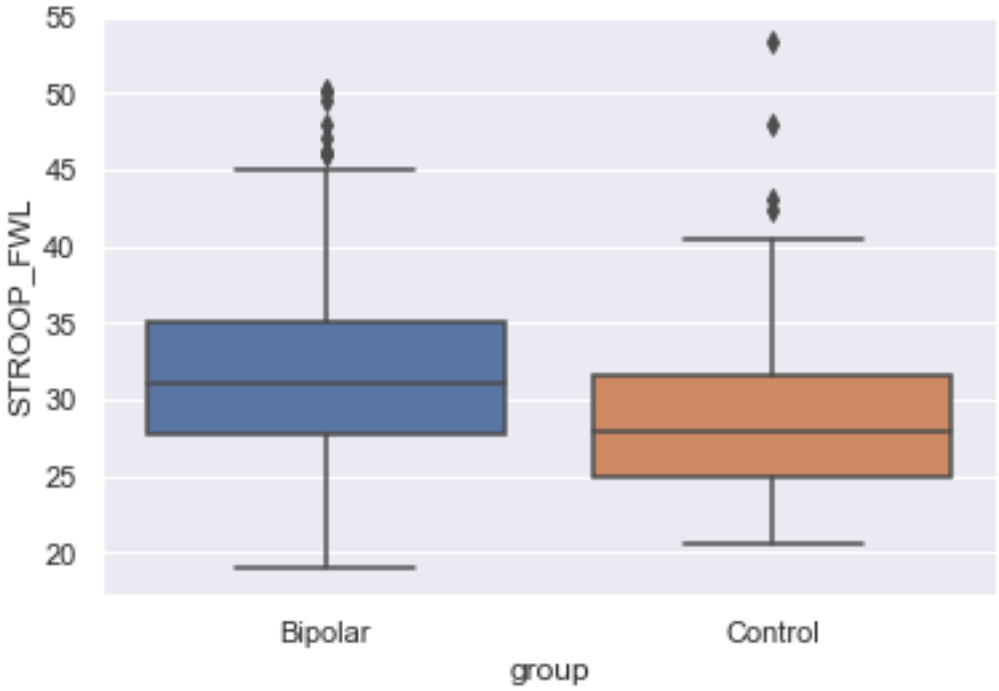
[29]:
```
sns.boxplot(x=topDF.group, y=topDF.STROOP_INT)
```

[29]:
```
<AxesSubplot:xlabel='group', ylabel='STROOP_INT'>
```



[30]:
```
sns.boxplot(x=topDF.group, y=ml4bp.STROOP_FWL)
```

[30]:
```
<AxesSubplot:xlabel='group', ylabel='STROOP_FWL'>
```



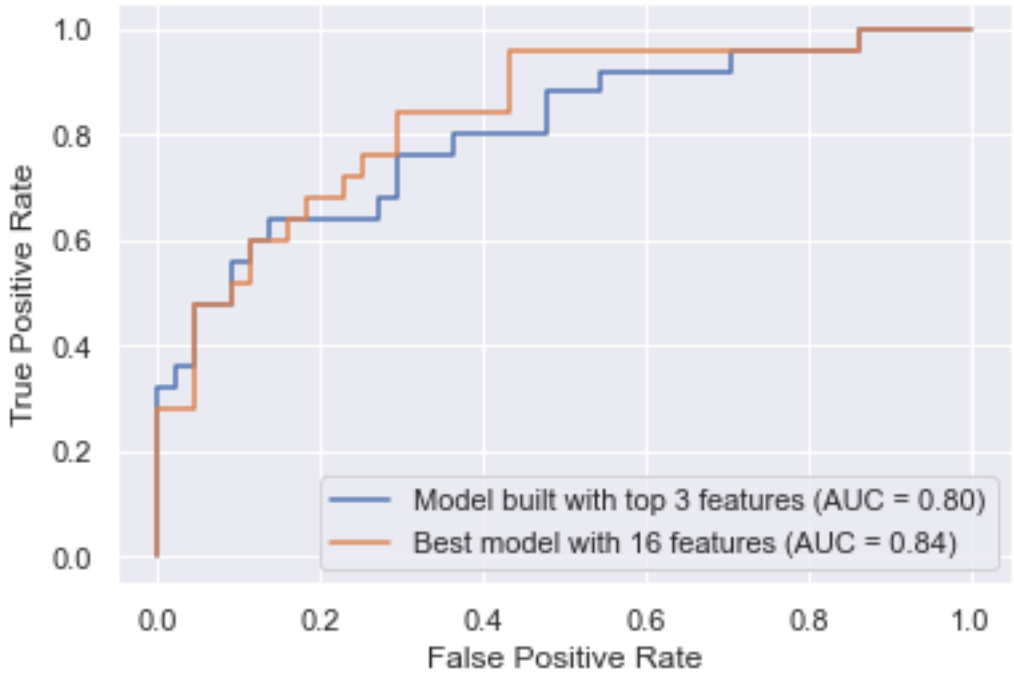
6.5 Training a model with optimization using only the top 3 features

[31]:
```
topmodel = LogisticRegression(random_state=0)
# build new X and y for topmodel
topX = pd.DataFrame(data=[ml4bp.vereint_KL, ml4bp.BMI, ml4bp.STROOP_INT]).T
topy = y
# impute all NaN's with the columns mean value
topX = SimpleImputer(missing_values=np.nan,
                     strategy='mean').fit_transform(topX)
# train test split
topX_train, topX_test, topy_train, topy_test = train_test_split(
    topX, topy, test_size=0.2, random_state=0)

# optimzing the second model using gridsearch and scoring
# defining the parameter grid
topparam_grid = [{  # 'clf__penalty': ['l1', 'l2', 'elasticnet'],
    'clf__C': np.logspace(-4, 4, 50),
    'clf__solver': ['newton-cg', 'lbfgs', 'liblinear'],
}]

# making a pipeline (redundant due to prior scaling)
toppipeline = Pipeline([('std', StandardScaler()),
                        ('clf', topmodel)])

# gridsearch, optimizing for best label "1" f1-score
topgrid = GridSearchCV(toppipeline, topparam_grid, scoring='f1_macro',
                       refit=True, verbose=1, n_jobs=-1, cv=5)
```

```
topgrid.fit(topX_train, topy_train)

# scoring and best parameters
print("Tuned model")
print(classification_report(topgrid.predict(topX_test), topy_test))
print(topgrid.best_params_)
print(topgrid.best_estimator_)
print("AUC: {:.3f}".format(roc_auc_score(
    topy_test, topgrid.decision_function(topX_test))))
```

```
Fitting 5 folds for each of 150 candidates, totalling 750 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:    0.3s

Tuned model
              precision    recall  f1-score   support

         1.0       0.84      0.80      0.82        46
         2.0       0.64      0.70      0.67        23

    accuracy                           0.77        69
   macro avg       0.74      0.75      0.74        69
weighted avg       0.77      0.77      0.77        69

{'clf__C': 0.3906939937054613, 'clf__solver': 'newton-cg'}
Pipeline(steps=[('std', StandardScaler()),
                ('clf',
                 LogisticRegression(C=0.3906939937054613, random_state=0,
                                    solver='newton-cg'))])
AUC: 0.805

[Parallel(n_jobs=-1)]: Done 720 tasks      | elapsed:    1.5s
[Parallel(n_jobs=-1)]: Done 750 out of 750 | elapsed:    1.6s finished
```

[32]:
```
# same code as for comparison of base and optimzed model
ax = plt.gca()
topcurve = plot_roc_curve(topgrid.best_estimator_, topX_test, topy_test, ax=ax)
baseroccurve = plot_roc_curve(grid.best_estimator_, X_test, y_test, ax=ax)
legend = plt.legend().get_texts()
legend[0].set_text('Model built with top 3 features (AUC = {})'.format(
    str(legend[0]).split()[-1][0:4]))
legend[1].set_text('Best model with 16 features (AUC = {})'.format(
    str(legend[1]).split()[-1][0:4]))
plt.show()
```



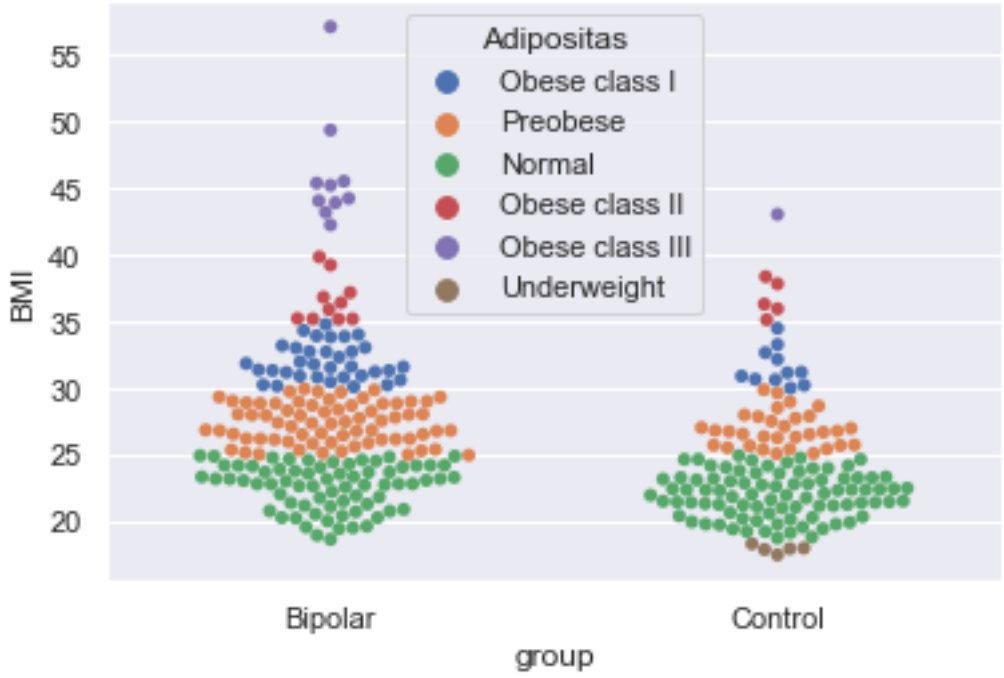
6.6 Bipolar disorder and obesity

[33]:
```
#bmi and adipositas

# bmi as parameter and return WHO classification for obesity
def bmicalc(bmi):
    #overweight > 25
    # preobese 25 - 29.99
    # obese 1 30 -34.99
    # obese 2 35 - 39.99
    # obese 3 > 40
    #under <18,5

    if bmi < 18.5:
        return "Underweight"
    elif bmi < 25:
        return "Normal"
    elif bmi < 30:
        return "Preobese"
    elif bmi < 35:
        return "Obese class I"
    elif bmi < 40:
        return "Obese class II"
    else:
        return "Obese class III"
```

```
# build new column with obesity class for every sample
adipositas = []
for i in topDF.BMI:
    adipositas.append(bmicalc(i))
topDF['Adipositas'] = adipositas

# count classes per group and print percentages of obese members
group_count = topDF['group'].value_counts()
adipositascount = topDF.groupby(['group'])['Adipositas'].value_counts()

print("Percentage of obese members of bipolar group: {:.1%}".format(
    sum(adipositascount[2:5])/group_count[0]))
print("Percentage of obese members of control group: {:.1%}".format(
    sum(adipositascount[7:10])/group_count[1]))
print("Percentage of obese and preobese members of bipolar group: {:.1%}".format(
    sum(adipositascount[1:5])/group_count[0]))
print("Percentage of obese and preobese members of control group: {:.1%}".format(
    sum(adipositascount[6:10])/group_count[1]))

# descriptive statistics for both groups
topDF.groupby(['group'])['BMI'].describe()
# plot
sns.swarmplot(y=topDF.BMI, x=topDF.group, hue=topDF.Adipositas)
```

```
Percentage of obese members of bipolar group: 28.1%
Percentage of obese members of control group: 14.5%
Percentage of obese and preobese members of bipolar group: 63.8%
Percentage of obese and preobese members of control group: 36.6%
```

[33]: <AxesSubplot:xlabel='group', ylabel='BMI'>



6.7 versions of mainly used libraries

[34]:
```
# versions
import matplotlib
import jupyterlab as lab
import sklearn as sklearn
import sys
print("Python: {}".format(sys.version))
print("scikit-learn: {}".format(sklearn.__version__))
print("jupyterlab: {}".format(lab.__version__))

#rint("jupyter-core: {}".format(core.__version__))
print("Numpy: {}".format(np.__version__))
print("Pandas: {}".format(pd.__version__))
print("Seaborn: {}".format(sns.__version__))
print('matplotlib: {}'. format(matplotlib. __version__))
```

```
Python: 3.9.0 (v3.9.0:9cf6752276, Oct  5 2020, 11:29:23)
[Clang 6.0 (clang-600.0.57)]
scikit-learn: 0.23.2
jupyterlab: 3.0.1
Numpy: 1.19.4
Pandas: 1.1.5
Seaborn: 0.11.0
matplotlib: 3.3.3
```