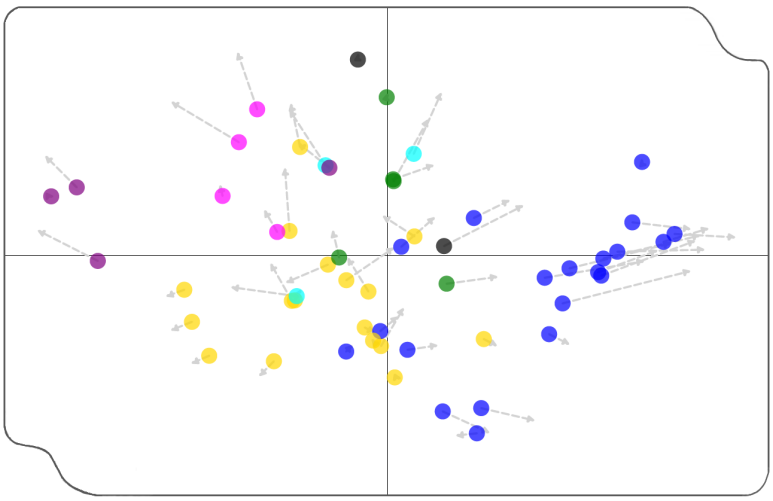
CHALMERS



On nonlinear machine learning methodology for dose-response data in drug discovery

Master's Thesis for the Degree of Master of Science
in Engineering Mathematics & Computational Science

KLARA GRANBOM

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2020

Thesis for the Degree of Master of Science

On nonlinear machine learning methodology for dose-response data in drug discovery

Klara Granbom

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2020

On nonlinear machine learning methodology for dose-response data in drug discovery
Klara Granbom

Supervisor: Dr. Adam Andersson, Smartr
Examiner: Prof. Aila Särkkä, Department of Mathematical Sciences

Master's Thesis 2020
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Dimensionality reduction visualization from a t-distributed stochastic neighbor embedding of dose-response data.

Typeset in LaTeX

On nonlinear machine learning methodology for dose-response data in drug discovery
Klara Granbom
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

This thesis investigates novel approaches to use nonlinear methodology for dose-response data in drug discovery. Such methodology could potentially create insights and value within the field, saving resources such as time and usage of animals in experiments. Methods for dimensionality reduction and visualization, as well as methods for classification of compounds into clinical classes based on therapeutic usage, are investigated. The thesis builds partly upon previous research where linear methods, based on partial least squares and principal component analysis, have been used for dimensionality reduction in drug discovery. By using results from linear methods as a benchmark, this thesis investigates the nonlinear methods kernel partial least squares and t-distributed stochastic neighbor embedding for dimensionality reduction. Moreover, methods for classification of compounds are investigated using the linear method multinomial logistic regression as well as the nonlinear methods random forest and multi-layer perceptron networks.

Results from nonlinear methods for dimensionality reduction do not detect any distinctly new patterns or clusters, compared to linear methodology. However, some results are promising to build upon in further methodology development.

The best performing classification method shows results corresponding to well-known effects for 70.6% of the compounds evaluated. Moreover, classifications of 11.8% of the compounds indicate potentially unknown effects, which are considered interesting and could be a springboard for further analysis and innovation. Therefore, this classification methodology can create insight and potentially high value.

Keywords: drug discovery, machine learning, classification, multi-layer perceptron, random forest, dimensionality reduction, partial least squares, kernel partial least squares, t-sne

Acknowledgements

I want to thank everyone at IRLAB and Smartr for contributing in different ways to make this thesis possible.

A special thank you to Adam Andersson, my supervisor at Smartr, for going above and beyond by providing invaluable guidance and answering all of my questions with so much patience.

I also want to send a special thank you to those at IRLAB who introduced me to their drug discovery process. Thank you, Susanna Waters and Peder Svensson, for helping me develop the data and evaluate the results with such passion. Further, I want to thank Johan Kullingsjö and Fredrik Wallner for enthusiasticly assisting me in the data exploration.

In addition, I want to thank Mattias Sundén for supporting me in the writing process.

Thank you, Gustav, for involuntarily volunteering to help me reduce the dimensions of my concerns.

Finally, I want to thank my family and friends for their outstanding support throughout my studies. You are all (classified as) fantastic (with 100% probability).

Klara Granbom, Gothenburg, June 2020

Contents

1

Introduction

This thesis is a collaboration with Integrative Research Laboratories (IRLAB), a biotech company focusing on the discovery and development of drugs to fight Parkinson's disease. IRLAB has data from experiments performed during two decades and works very rigorously with data quality. In their drug discovery process, parallel assessment of biological measures is performed on experimental rats and hundreds of variables are measured. This process and the rigorous data approach result in more flexible data which can be used to answer questions of different origin, compared to more conventional drug screening programs [1].

To evaluate data from drug discovery processes efficiently, previous research have discussed the potential usage of machine learning [2, 3]. Machine learning methods seek to detect mathematical relationships and can potentially be used to, for instance, predict chemical and biological properties of novel chemical compounds. Generalization of machine learning methods to big data sets can be made without the need for extensive computational resources, compared to more conventional physical models based on explicit equations from e.g. quantum chemistry or molecular dynamics [4]. Usage of machine learning methods can potentially extract additional information from drug discovery experiments. Hopefully, this results in both faster processes and that fewer experiments are needed. Through an ethical perspective, fewer experiments would reduce the harm of using animals in experiments.

The methods for data analysis used by IRLAB today are mainly linear dimensionality reduction methods based on *principal component analysis* and *partial least squares*. Since it might create new insights, IRLAB now wants to investigate non-linear methodology, possibly from machine learning, and evaluate qualitatively and quantitatively if such methods can extract patterns and clusters which the linear methods are unable to detect. IRLAB contributes to this thesis with data expertise, knowledge about their current methodology and input on what they want from the new methodology.

The purpose of this thesis is to explore, investigate and evaluate nonlinear methodology on dose-response data. Both methodology for dimensionality reduction and classification of compounds in clinical classes, based on therapeutic findings and usage, are investigated. The most promising nonlinear methods are implemented and evaluated to see qualitatively and quantitatively if the dose-response data contains nonlinearities and if the methods can extract patterns and clusters which the linear methods are unable to detect.

The first phase of this thesis consists of getting up to date with the current methodology at IRLAB. The linear dimensionality reduction methods used in previous work [1] are implemented to have as a benchmark for further exploration. Nonlinear methods for dimensionality reduction are then investigated. Then the most promising are considered for visual evaluation to help to answer if the nonlinear methods can detect patterns and clusters which the linear methods are unable to detect. The results of nonlinear methods differ only slightly from the results of the linear methods. Therefore, nonlinear methods do not detect any distinctly new patterns or clusters. However, some results could be built upon in further work.

Regarding classification methods, one linear method and two nonlinear methods are investigated and implemented. Results from the linear method are compared to results from the nonlinear methods to evaluate if there are nonlinearities in the dose-response data. Classifiers are evaluated quantitatively by considering performance metrics and qualitatively by consulting people with expertise in the data. The qualitative evaluation is performed by training classifiers on data with verified clinical classes and using the classifiers to predict compounds with unverified clinical classes. Results are qualitatively evaluated to see if they correspond to the intuition of IRLAB and tentative clinical classes since this can create insights and value. Results from the best performing classifier are promising, where predictions of 70.6% of the compounds are consistent with well-known effects. Moreover, predictions of 11.8% of the compound are considered to be possible foundations of future innovation.

This thesis is structured as follows. Chapter 2 and 3 presents theory of methods used for dimensionality reduction and classification, respectively. In Chapter 4, information on data used in this thesis is provided. Chapter 5 presents the current methodology and implementation of this and nonlinear methodology, both for dimensionality reduction and classification. Results are presented and commented in Chapter 6 and finally, conclusions and possible future work are presented in Chapter 7. In the Appendix, additional results are presented.

2

Dimensionality Reduction

In this chapter, the algorithms used for dimensionality reduction in this thesis are described. First, the linear dimensionality reduction method *partial least squares* regression that is part of currently used methodology is presented in Section 2.1. Further, the nonlinear dimensionality reduction methods, *kernel partial least squares* and *t-distributed stochastic neighborhood embedding*, are presented in Sections 2.2 and 2.3.

2.1 Partial Least Squares Regression

Partial least squares (PLS) regression is a linear multivariate model used to relate two data matrices to each other [5]. The model is useful for analyzing data with many, noisy and collinear variables in both matrices [6]. As an example, PLS can be used when measuring different types of properties on the same substances or compounds in clinical research. PLS is a regression extension of *principal component analysis* (PCA). Section 2.1.1 below presents an overview of the PLS problem and a famous algorithm for solving it. Additionally, it is explained how PLS can be used for dimensionality reduction and visualization purposes in Section 2.1.2.

2.1.1 Partial Least Squares and the NIPALS algorithm

Consider a setting with $n \geq 1$ input and output data pairs $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$. The input data matrix $\mathbf{X}$ of dimension $n \times K$ and an output data matrix $\mathbf{Y}$ of dimension $n \times M$ have observations of different variables as coordinates $\mathbf{x}_i \in \mathbb{R}^K$ and $\mathbf{y}_i \in \mathbb{R}^M$ in the rows, respectively. In the full partial least squares (PLS) problem, orthogonal matrices $\mathbf{P}$ and $\mathbf{C}$ of dimensions $K \times K$ and $M \times K$, where $K, M \geq 1$, are sought such that the transformed data matrices $\mathbf{T}$ and $\mathbf{U}$, both of dimension $n \times K$, given

by

$$\mathbf{T} = \mathbf{XP}$$
$$\mathbf{U} = \mathbf{YC},$$

satisfy

1. the coordinates, i.e. columns, of $\mathbf{T}$ are uncorrelated,
2. the first coordinate of $\mathbf{T}$ has largest variance among the coordinates and the succeeding variances are non-increasing,
3. every coordinate has *roughly* maximal variance, and
4. the sample covariance between coordinate l of $\mathbf{T}$ and coordinate l of $\mathbf{U}$ is *roughly* maximal for every $l = 1, ..., K$.

Conditions 1 and 2 are precise. Conditions 3 and 4 can in general not be satisfied simultaneously, which is why they are notated by *roughly*. The precise optimization problem, explaining how conditions 3 and 4 are optimized, can be found in [7, p. 81]. It is not very intuitive and the interested reader is referred to this book. Removing condition 4 and using condition 1–3 for $\mathbf{X}$ only gives the principal component decomposition of $\mathbf{X}$, that is used in PCA. The matrices $\mathbf{P}$ and $\mathbf{C}$ are called *loadings* and the matrices $\mathbf{T}$ and $\mathbf{U}$ are called *scores*.

Since $\mathbf{P}$ and $\mathbf{C}$ are orthogonal it equivalently holds that

$$\mathbf{X} = \mathbf{TP}^T$$
$$\mathbf{Y} = \mathbf{UC}^T.$$

It is not usual to transform the K-dimensional data in $\mathbf{X}$ into the K-dimensional transformed data in $\mathbf{T}$, but rather to a lower L-dimensional subspace, where $L < K$. The reason can be for dimensionality reduction purposes. Moreover, it can be since essentially all information is captured in the L-dimensional subspace, and that the contribution in the remaining subspace of dimension $K - L$ is mainly noise. In this case, the problem is to find matrices $\mathbf{P}, \mathbf{C}, \mathbf{E}, \mathbf{F}$ of dimensions $K \times L$, $M \times L$, $n \times K$

and $n \times M$ such that

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T + \mathbf{E}$$
$$\mathbf{Y} = \mathbf{U}\mathbf{C}^T + \mathbf{F},$$

and conditions 1–4 above hold. The matrices $\mathbf{E}$ and $\mathbf{F}$ are error terms representing the deviation of the original K-dimensional data from the L-dimensional representation [6].

A famous algorithm for solving the PLS problem is the *Nonlinear Iterative Partial Least Squares* (NIPALS) algorithm [8, 9, 10]. Since PLS is not specified with full precision in this thesis, the NIPALS algorithm for PLS (NIPALS-PLS) is not derived. One benefit with NIPALS is that the dimensions $l = 1, \ldots, L$, are computed one at a time, and a decision on the number of dimensions to use can be done without computing all dimensions. To perform the NIPALS-PLS algorithm, one starts for $l = 1$ by randomly initializing $\mathbf{u}_l \in \mathbb{R}^n$ and iteratively computing the steps

1. $\mathbf{w}_l = \mathbf{X}^T\mathbf{u}_l$
2. $\mathbf{t}_l = \mathbf{X}\mathbf{w}_l$, $\mathbf{t}_l \leftarrow \mathbf{t}_l/\|\mathbf{t}_l\|$
3. $\mathbf{c}_l = \mathbf{Y}^T\mathbf{t}_l$
4. $\mathbf{u}_l = \mathbf{Y}\mathbf{c}_l$, $\mathbf{u}_l \leftarrow \mathbf{u}_l/\|\mathbf{u}_l\|$,

until convergence of $\mathbf{t}_l$. After convergence, the $\mathbf{X}$ and $\mathbf{Y}$ matrices are deflated, meaning that

$$\mathbf{X} \leftarrow \mathbf{X} - \mathbf{t}_l\mathbf{t}_l^T\mathbf{X} \quad \text{and} \quad \mathbf{Y} \leftarrow \mathbf{Y} - \mathbf{t}_l\mathbf{t}_l^T\mathbf{Y}.$$

The steps 1-4 and the deflation of matrices are repeated for $l = 2, \ldots, L$. This gives $(\mathbf{t}_l)_{l=1}^L$, $(\mathbf{u}_l)_{l=1}^L$ and $(\mathbf{c}_l)_{l=1}^L$. The score matrices $\mathbf{T}$ and $\mathbf{U}$ and the loading matrix $\mathbf{C}$ are constructed by by having $(\mathbf{t}_l)_{l=1}^L$, $(\mathbf{u}_l)_{l=1}^L$ and $(\mathbf{c}_l)_{l=1}^L$ as columns. Moreover, the $K \times L$ *weight* matrix $\mathbf{W}$ is constructed by having $(\mathbf{w}_l)_{l=1}^L$ as columns. The loading matrix $\mathbf{P}$ is finally given by $\mathbf{P} = \mathbf{X}^T\mathbf{T}(\mathbf{T}^T\mathbf{T})^{-1}$. Since $\mathbf{T}^T\mathbf{T}$ is diagonal this is equivalent to the columns of $\mathbf{P}$ satisfying $\mathbf{p}_l = \mathbf{X}^T\mathbf{t}_l/(\mathbf{t}_l^T\mathbf{t}_l)$. When using PLS for dimensionality reduction purposes, the columns $\mathbf{t}_l$, $\mathbf{u}_l$, $\mathbf{p}_l$, $\mathbf{c}_l$ and $\mathbf{w}_l$ are often referred to as the l^{th} *principal component* scores, loadings respectively weights.

2.1.2 Principal Components for Visualization

When using PLS for dimensionality reduction purposes, plots of loadings, scores and weights from the PLS problem can be considered to potentially gain insight in the data. Usually the first couple of columns, corresponding to the first couple of principal components, of the transformed data is used since they capture most of the variance. These kinds of principal component plots can then be considered to gain insight into how variables and observations affect each other.

Plots of different principal component scores can be used to gain insight into how different observations relate. To plot scores in two dimensions, the values of $\mathbf{t}_l$ are plotted against corresponding values of $\mathbf{t}_j$ of another principal component $j \neq l$ where $j, l \in [1, L]$. Grouped observations then have similar properties and observations close to the origin have average properties in the principal component space. Scores $\mathbf{T}$ and $\mathbf{U}$ of both data matrices $\mathbf{X}$ and $\mathbf{Y}$ can be plotted simultaneously to see how the data matrices relate to each other.

Plots of principal component loadings can be used to gain insight into how variables affect each other. In a loading plot of two dimensions, the values of $\mathbf{p}_l$ are plotted against corresponding values of $\mathbf{p}_j$ of another principal component $j \neq l$, $j, l \in [1, L]$. If different variables correlate strongly, they have similar principal component values, i.e. they are located very close to each other. The distance to the origin of a loading plot also holds information. Variables further away from the origin have a more substantial impact on the principal components, and vice versa [6]. Figure 2.1 shows an illustrative example of a two dimensional plot of loadings and scores for one data matrix $\mathbf{X}$, corresponding to PCA.

In addition to studying the loadings to gain insight into how variables affect each other, the weights $\mathbf{C}$ and $\mathbf{W}$ can be considered. The transformed $\mathbf{X}$ weights $\mathbf{W}^* = \mathbf{W}(\mathbf{P}^T\mathbf{W})^{-1}$ are sometimes used instead of $\mathbf{W}$ [6], which is the approach of [1]. Similar as for the loadings, the weight vectors corresponding to different principal components can be plotted against each other. These kinds of weight plots can be used to gain insight into which variables contribute to forming the relationship between $\mathbf{X}$ and $\mathbf{Y}$. Figure 2.2 shows an illustrative example of a PLS weight plot.

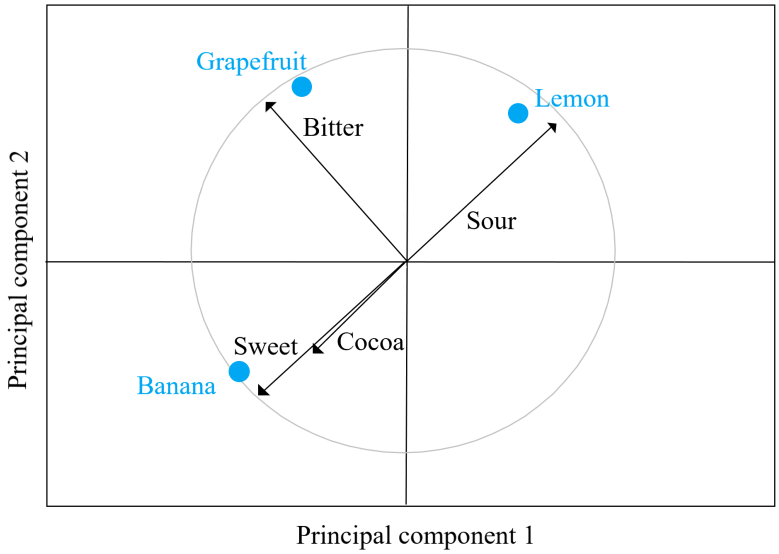


Figure 2.1: Illustrative example of a PCA plot. The circle in the plot is only to help the reader with a consistent distance to the origin.
Suppose a data table **X** where different variables, in this case flavors, have been measured for different observations, in this case fruits. The loadings of different variables are represented as arrows, where loadings that are pointing in similar directions, in this case 'Sweet' and 'Cocoa', are similar and positively correlated. However, the loading of 'Cocoa' is closer to the origin than the loading of 'Sweet', which reflects that 'Cocoa' is not as highly described by the two first principal components as 'Sweet' is. 'Sour' has its loading in the opposite direction of 'Sweet', indicating a negative correlation. The loading of 'Bitter' is orthogonal to the loadings of both 'Sweet' and 'Sour' which means they are not correlated at all to 'Bitter'. Scores of observations are represented as blue dots. According to the first two principal components, Lemon is mostly sour, slightly bitter, but almost no sweet at all. Banana, on the other hand, is mostly sweet according to the first two principal components.

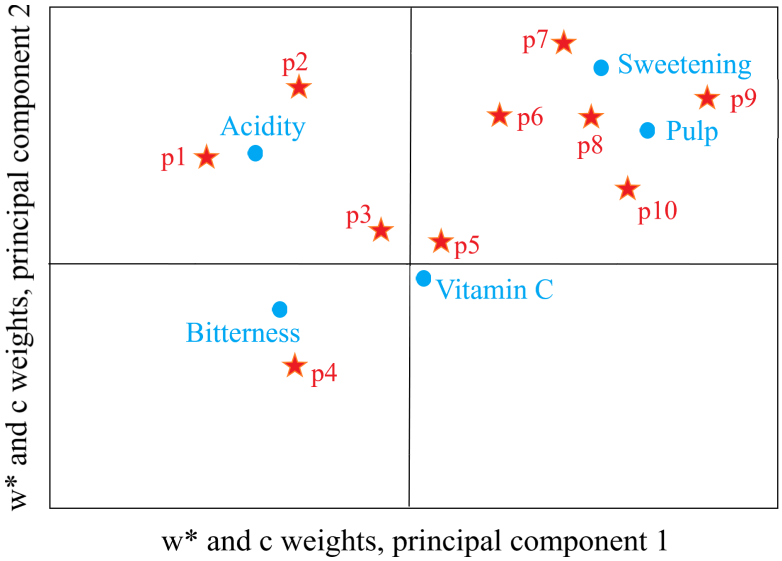


Figure 2.2: Illustrative PLS plot of weights $\mathbf{w}*$ and $\mathbf{c}$ in blue dots and red stars, respectively. Suppose two data matrices of measurements of different properties on different juice brands. Let the $\mathbf{X}$ matrix contain explanatory variables such as the level of sweetening, bitterness, pulp and vitamin C of the juice brands. Suppose also that several persons (p1,...,p10) have rated the juice brands depending on their preference, stored in the $\mathbf{Y}$ data matrix. From this, the relationship between the data matrices can be modeled using PLS and potentially, the organizations behind the juice brands can examine what kind of characteristics that potential customers enjoy. In the plot, there are more persons located close to sweetening and pulp compared to the number of persons close to bitterness. Hence, the plot illustrates that more persons seem to fancy sweet juice with pulp in it and fewer persons seem to fancy bitter juice. Moreover, the level of vitamin C is not very well represented by the first two principal components. This fact can be interpreted as that the level of vitamin C only explains a small amount of the preference of persons, which seems reasonable since it does not affect the taste or other criteria that could influence the preference.

2.2 Kernel Partial Least Squares

Partial least squares is a linear method. One way to obtain a nonlinear counterpart, suggested by [8], is to introduce a nonlinear map $\varphi\colon \mathbb{R}^K \to V$ that maps data into some vector space V and solves the linear PLS problem for the transformed data. Note that the dimension of V depends on the choice of $\varphi(\cdot)$. Consider the original input data $\mathbf{X}$ with n components $\mathbf{x}_i \in \mathbb{R}^K$, $i = 1, ..., n$. The transformed data is then saved in a matrix $\mathbf{\Phi} = \varphi(\mathbf{X})$ having rows corresponding to $\varphi(\mathbf{x}_i)$. The PLS problem for the transformed data is given by

$$\mathbf{\Phi} = \mathbf{T}\mathbf{P}^T + \mathbf{E}$$

$$\mathbf{Y} = \mathbf{U}\mathbf{C}^T + \mathbf{F}.$$

The steps 1-4 of the NIPALS-PLS algorithm in Section 2.1.1 can now be modified to use the transformed data $\mathbf{\Phi}$. Then, the algorithm do not use the weights $\mathbf{w}_l$ and starts for $l = 1$ by randomly initializing $\mathbf{u}_l \in \mathbb{R}^n$ and iteratively computing the steps

1. $\mathbf{t}_l = \mathbf{\Phi}\mathbf{\Phi}^T\mathbf{u}_l$, $\mathbf{t}_l \leftarrow \mathbf{t}_l/\|\mathbf{t}_l\|$
2. $\mathbf{c}_l = \mathbf{Y}^T\mathbf{t}_l$
3. $\mathbf{u}_l = \mathbf{Y}\mathbf{c}_l$, $\mathbf{u}_l \leftarrow \mathbf{u}_l/\|\mathbf{u}_l\|$,

until convergence of $\mathbf{t}_l$. After convergence the matrices are deflated according to

$$\mathbf{\Phi}\mathbf{\Phi}^T \leftarrow (\mathbf{\Phi} - \mathbf{t}_l\mathbf{t}_l^T\mathbf{\Phi})(\mathbf{\Phi} - \mathbf{t}_l\mathbf{t}_l^T\mathbf{\Phi})^T \quad \text{and} \quad \mathbf{Y} \leftarrow \mathbf{Y} - \mathbf{t}_l\mathbf{t}_l^T\mathbf{Y}.$$

The steps 1-3 and the deflation of matrices are repeated for $l = 2, \ldots, L$. This approach can be computationally expensive in practice. Especially when the map $\varphi(\cdot)$ transforms data into a vector space being very high, or even infinitely, dimensional. Fortunately there is a trick called the *kernel trick* to avoid this problem. The same trick is used for support vector classification and regression, see e.g. [11].

Before we proceed we notice that

$$(\mathbf{\Phi}\mathbf{\Phi}^T)_{i,j} = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle_V, \tag{2.1}$$

where $\langle \cdot, \cdot \rangle_V$ is the inner product on the vector space V.
A function $K\colon \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}$ is called positive semidefinite if for all functions $f\colon \mathbb{R}^K \to \mathbb{R}$ satisfying that $\int_{\mathbb{R}^K} f(x)^2 \, \mathrm{d}x < \infty$ it holds that

$$\int_{\mathbb{R}^K} \int_{\mathbb{R}^K} f(x) K(x, y) f(y) \, \mathrm{d}x \, \mathrm{d}y \geq 0.$$

Mercer's Theorem [12] states for all positive semidefinite kernels $K\colon \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}$ there exists a so called *Reproducing Kernel Hilbert space*, denoted V, and a function $\varphi\colon \mathbb{R}^K \to V$ such that $K(x, y) = \langle \varphi(x), \varphi(y) \rangle_V$. This fact is used together with (2.1) to present the NIPALS algorithm for kernel PLS.

Consider a positive semidefinite kernel $K\colon \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}$ and define the Gram matrix $\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^K$. Inspired by (2.1) and Mercer's Theorem, $\mathbf{\Phi}\mathbf{\Phi}^T$ is replaced by $\mathbf{K}$ in the NIPALS-PLS algorithm. The NIPALS algorithm for kernel PLS is then to start for $l = 1$ by randomly initializing $\mathbf{u}_l \in \mathbb{R}^n$ and iteratively computing the steps

1. $\mathbf{t}_l = \mathbf{K}\mathbf{u}_l$, $\mathbf{t}_l \leftarrow \mathbf{t}_l / \|\mathbf{t}_l\|$
2. $\mathbf{c}_l = \mathbf{Y}^T\mathbf{t}_l$
3. $\mathbf{u}_l = \mathbf{Y}\mathbf{c}_l$, $\mathbf{u}_l \leftarrow \mathbf{u}_l / \|\mathbf{u}_l\|$

until convergence of $\mathbf{t}_l$. Finally, matrices are deflated by

$$\mathbf{K} \leftarrow (\mathbf{I} - \mathbf{t}_l\mathbf{t}_l^T)\mathbf{K}(\mathbf{I} - \mathbf{t}_l\mathbf{t}_l^T)^T \quad \text{and} \quad \mathbf{Y} \leftarrow \mathbf{Y} - \mathbf{t}_l\mathbf{t}_l^T\mathbf{Y},$$

where $\mathbf{I}$ is an $n \times n$ identity matrix. The steps 1-3 and the deflation of matrices are repeated for $l = 2, \ldots, L$. By using the kernel matrix $\mathbf{K}$ in the NIPALS algorithm for kernel PLS above, the algorithm avoids using the nonlinear map in the first step. This corresponds to the kernel trick. In this way, even if the implicitly given space V is very high dimensional, even infinitely dimensional, the NIPALS algorithm has not higher computational cost than the linear setting.

2.2.1 Kernel functions

As briefly described in the section above, expressing a nonlinear map as a kernel function allows the kernel PLS regression to use the kernel matrix in the iterations. The theory of the kernel functions used for kernel PLS in this thesis is presented below.

Radial Basis Kernel

The *radial basis function* (RBF) kernel is a kernel that is in the form of a radial basis defined as

$$K_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2), \tag{2.2}$$

where γ is the width parameter. The nonlinear function $\varphi(\cdot)$ implied by the RBF Kernel and Mercer's Theorem is in fact mapping into the infinitely dimensional space [11]. Hence, the kernel trick is useful to reduce computational complexity. The RBF kernel represents the similarity of vectors as a decaying function of the distance between the vectors. This means that the closer vectors are to each other,

the larger the RBF kernel value. The width of RBF kernel values are determined by the width parameter $\gamma > 0$. Geometrically, a smaller value of γ corresponds to a wider and lower RBF kernel and a larger value corresponds to a more narrow and higher RBF kernel.

Polynomial Kernel

Polynomial kernels can be defined in various ways but all consist of polynomial kernel functions. In this thesis, polynomial kernels of the form

$$K_{\text{Poly}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + b)^a \tag{2.3}$$

of different orders a and parameters b are used. It should be noted that in the case of $a = 1$ and $b = 0$, the polynomial kernel function in (2.3) becomes the linear kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$, which is equivalent to linear PLS. A particular case of polynomial kernels is the quadratic kernels, i.e. kernels where $a = 2$.

2.3 T-distributed Stochastic Neighbor Embedding

T-distributed stochastic neighbor embedding (t-SNE) is a technique used to visualize high-dimensional data in a lower-dimensional space, usually of two or three dimensions. According to [13], t-SNE is capable of capturing much of the local structure of high-dimensional data very well, while also revealing global structure such as the presence of clusters at several scales.

In the t-SNE method, a probabilistic structure is assumed with conditional probabilities $p_{j|i}$ in the high-dimensional space. These conditional probabilities represent that a data point $\mathbf{x}_i$ would pick $\mathbf{x}_j$ as its neighbor if neighbors were picked in proportion to the probability density under a Gaussian kernel centered on $\mathbf{x}_i$. The conditional probabilities are given by

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}, \tag{2.4}$$

where σ_i is the variance of the Gaussian kernel centered on $\mathbf{x}_i$. Only pairwise similarities are of interest, so $p_{i|i}$ is set to zero. Now, let P_i represent the conditional probability distribution over all other data points given a data point $\mathbf{x}_i$. Then, the *perplexity* parameter needed for the t-SNE algorithm is defined as

$$\text{Perplexity}(P_i) = 2^{H(P_i)}, \tag{2.5}$$

where $H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$ is the Shannon entropy of P_i measured in bits. If the variance σ_i is known, the conditional probabilities (2.4) can be computed and so also the perplexity. However, the case in the t-SNE method is that the perplexity is specified by the user. Since this is the case, t-SNE performs a binary search for the values of σ_i to use in the conditional probabilities (2.4) [13].

In the t-SNE method, the joint probability density of the Gaussian distribution centered on $\mathbf{x}_i$ is used to measure the similarity of the point $\mathbf{x}_i$ to all other points $\mathbf{x}_j$ in the high-dimensional space [13]. The joint probabilities are given by

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}, \tag{2.6}$$

where n is the dimension of the high-dimensional space and $p_{j|i}$ is given by (2.4).

Furthermore, a probabilistic structure is assumed in the lower-dimensional space. In the regular SNE method, Gaussian distributions are used in both the high- and low-dimensional spaces, which results in that points tend to get too crowded in the low-dimensional space. Using a Student t-distribution with one degree of freedom, which is more heavy-tailed than the Gaussian distribution, results in that the distances between points in the lower-dimensional space increase. Therefore, SNE is extended to t-SNE by employing a Student t-distribution with one degree of freedom in the lower-dimensional space [13].

Let $\mathbf{y}_i$ and $\mathbf{y}_j$ be the low-dimensional points corresponding to $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively. The joint probability densities q_{ij} under the Student t-distribution with one degree of freedom are used to measure similarities in the lower-dimensional space. The joint probabilities are given by

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}. \tag{2.7}$$

Using the probabilistic structures in the low- and high-dimensional spaces, t-SNE seeks to model each high-dimensional data point as a point in the lower-dimensional space in a way such that similar objects have a high probability of being visualized close to each other. In the t-SNE method, a lower-dimensional representation is constructed that minimizes the mismatch between the two sets of joint probabilities, p_{ij} and q_{ij}, in (2.6) and (2.7). More precisely, t-SNE seeks to minimize the sum of

Kullback-Leibler divergences [13]. The corresponding cost-function is

$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{2.8}$$

where p_{ii}, q_{ii} and $\log(p_{ii}/q_{ii})$ are set to zero. The cost function in (2.8) is minimized by a gradient descent method using

$$\frac{\delta C}{\delta \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}. \tag{2.9}$$

For a derivation of (2.9), see [13].

The perplexity parameter (2.5) is related to the number of nearest neighbors, which means that it determines how to balance attention between local and global features of data in the t-SNE algorithm. A higher perplexity value corresponds to more attention to global features, while a lower value corresponds to more attention to local features. Hence, a too high perplexity value might not reflect variations within clusters, while a too low perplexity value might not show any distinct clusters of the data points on a global level. The perplexity value should be adjusted when data points are added since a higher number of data points require a higher perplexity value. Typical values of the perplexity are between 5 and 50 and t-SNE is fairly robust to changes in perplexity [13]. Nevertheless, when using t-SNE in practice, users would still have to interactively choose perplexity value by visually comparing results under multiple parameter settings [14]. This process often requires knowledge and understanding of the t-SNE algorithm, which could lead to non-expert users misinterpreting data [15].

Finally, it should be noted that the objective function of t-SNE is non-convex and minimized using a gradient descent optimization that is initiated randomly. Therefore, different runs may result in different solutions [13].

3

Classification

In this chapter, the algorithms and concepts used for classification in this thesis are described. The linear method *multinomial logistic regression* is presented in Section 3.1, followed by the nonlinear methods *random forest* and *multi-layer perceptron* networks in Sections 3.2 and 3.3. A technique for handling imbalanced data sets is presented in Section 3.4 and finally concepts for evaluation of classifiers are presented in Section 3.5.

The main idea behind classification problems is to, given input data, find the class giving the highest expected probability score. A classifier is trained to distinguish classes from each other and can be applied to predict class given input data.

3.1 Multinomial Logistic Regression

Logitstic regression is a linear method that can be used for classification. In this thesis, *multinomial logistic regression* for multilabel classification, also called *softmax regression*, using *cross-entropy loss* is used. The objective of multinomial logistic regression is to construct a linear predictor function that constructs a score from the explanatory variables of a given observation.

Assume there are n data points, each consisting of a set of M variables $\mathbf{x}_i = x_{1,i},..., x_{M,i},\ i = 1,..,n$. Further, assume that a data point can take one class $c_j \in C$, $j = 1,...,K$, where C is the set of the $K = |C|$ possible classes. The linear predictor function, for each class $c_j \in C$, is then defined as

$$f(j,i) = \boldsymbol{w}_j \cdot \mathbf{x}_i + b_j, \tag{3.1}$$

where $\boldsymbol{w}_j = w_{1,j},..., w_{M,j}$ are vectors of regression weights/coefficients and b_j are real valued regression bias/interception terms.

The target, or true class, y in multinomial logistic regression is a variable ranging over more than two different classes $c_j \in C$. The method calculates the probabilities of y being in each potential class, given an input $\mathbf{x}$. To compute the probabilities, the *softmax function* is used. Suppose $\mathbf{z} = [z_1, z_2, ..., z_k]$ is a vector of dimension k, then the softmax function of this vector is defined as the vector

$$\text{softmax}(\mathbf{z}) = \left[\frac{e^{z_1}}{\sum_{i=1}^{k} e^{z_i}}, \frac{e^{z_2}}{\sum_{i=1}^{k} e^{z_i}}, ..., \frac{e^{z_k}}{\sum_{i=1}^{k} e^{z_i}} \right]. \tag{3.2}$$

In multinomial logistic regression, the input to the softmax is the linear predictor function (3.1) for each class. Note that weights and biases are different for each of the K classes. The probability that an input observation $\mathbf{x}$ has the true class y being $c_j \in C$, $j = 1, .., K$, is then given by

$$P(\mathbf{x} \text{ has true label } c_j) = \frac{e^{\mathbf{w}_j \cdot \mathbf{x} + b_j}}{\sum_{k=1}^{K} e^{\mathbf{w}_k \cdot \mathbf{x} + b_k}}. \tag{3.3}$$

The cross-entropy loss, also called logistic loss, function used in multinomial logistic regression, with true class y and predicted class $\hat{y}$, is

$$L_{CE}(\hat{y}, y) = -\sum_{j=1}^{K} \mathbb{1}\{y = c_j\} \log \frac{e^{\mathbf{w}_j \cdot \mathbf{x} + b_j}}{\sum_{k=1}^{K} e^{\mathbf{w}_k \cdot \mathbf{x} + b_k}}, \tag{3.4}$$

which is how much the predicted $\hat{y}$ differs from the true y, for one single example $\mathbf{x}$ [16]. In the cross-entropy loss function (3.4), $\mathbb{1}\{\cdot\}$ is 1 if the expression between the brackets is true and 0 otherwise. The objective of multinomial logistic regression is to minimize this loss as a function of weights and biases, which means to find a set of weights and biases that makes the loss as small as possible. There are different solvers available to solve this minimization problem. One such solver is the *Limited-memory Broyden–Fletcher–Goldfarb–Shanno* (L-BFGS) algorithm, that is a quasi-Newton optimization method for real-valued and multivariate functions [17]. From the minimization problem, the multinomial logistic regression method outputs a set of weights $\boldsymbol{w}_j$ and a bias b_j for each class $c_j \in C$ which can be used for prediction of class for unseen observations. The predicted class is the class giving the highest prediction probability (3.3).

To avoid overfitting training data and gain better performance on unseen data, *regularization* can be used. Regularization adds a penalty that increases with model complexity to the loss function. In this thesis, $L2$-regularization is used. Assume

the weights $w_0, ..., w_M$ in the predictor function, then the $L2$ penalty has the form $\lambda \sum_{i=1}^{M} w_i^2$, where λ is the penalty parameter. Hence, using $\lambda = 0$ corresponds to no penalty at all. Further, it is important to use an appropriate value of the penalty parameter λ since a too small value might result in overfitting and a too large value might result in underfitting.

3.2 Random Forest

Multinomial logistic regression is, as mentioned in the section above, a linear method for classification. Consequently, it can not capture nonlinearities in given data and is not advantageous when having nonlinear relationships. *Random forest* is, in contrast, a nonlinear method that can be used for supervised classification. It is an ensemble method, meaning that it uses multiple learning algorithms, usually resulting in better performance and prediction ability. More specifically, the random forest method is an ensemble method consisting of several *decision trees* [18].

A decision tree can be used for supervised classification and consists of a tree-like structure where each leaf node holds a class label. An observation is classified by starting at the root node of the tree. A feature, i.e. variable, specified by the node is considered. Depending on the value, a new node is selected at the next level in the tree. This process continues recursively for the subtree rooted at the new node until a leaf node is reached.

When training a decision tree classifier, all features of the training data are considered at each level in the tree, and different binary partitions, so-called *splits*, are tried. The best split is selected based on a *cost function*, where the split having the lowest cost is selected. From each branch in this split, a new split is decided based on the cost function. In this way, the tree is built. One commonly used cost function, which is also used in this thesis, is the *gini impurity*. The gini impurity measures the probability of misclassification for a randomly picked observation in the data set that is randomly classified according to the class distribution in the data set. For a classification problem of K classes, the gini impurity at a node j is defined as

$$\text{GINI}(j) = \sum_{i=1}^{K} P(i|j) \sum_{l \neq i} P(l|j) = \sum_{i=1}^{K} P(i|j)(1 - P(i|j)),$$

where $P(i|j)$ is the probability of an item being classified with label i at node j. When selecting a good split at a node, the gini impurity is calculated for each branch

and results are weighted according to the number of elements in each branch. The split with the lowest branch weighted gini impurity is selected. If this split gives a lower branch weighted gini impurity than the current node, the current node is split. Different criteria can be incorporated to decide when to stop splitting a tree. One criterion could be to decide a minimum number of samples that are required to split a node. Another is to decide the maximum depth of a tree. If the tree at a point during training reaches the criterion decided, then the nodes are not further split.

To predict the decision tree response y of an input observation $\mathbf{x}$, pass $\mathbf{x}$ down the decision tree until it reaches a leaf node. Let k denote the leaf node and let $y_{k_1}, ..., y_{k_n}$ denote response values of the training data in node k. The prediction function of the input observation $\mathbf{x}$ is then

$$h(\mathbf{x}) = \underset{y \in C}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbb{1}\{y_{k_i} = y\}, \tag{3.5}$$

where C is the set of available classes and $\mathbb{1}\{\cdot\}$ is 1 if the condition inside the paranthesis holds and 0 otherwise.

Returning to ensemble methods, they usually result in better predictive performance than by using a single decision tree. In decision tree ensemble methods, the whole original data set can be used for each tree, or the data set can be split into subsets. Choosing subsets randomly with replacement is called *bagging* or *bootstrap aggregation*. The random forest method is an extension to bagging where in addition to taking the random subsets of data, also taking a random selection of features rather than using all features when deciding the best splits and growing trees. The optimal parameters to use in a random forest, such as the number of trees and how many features to consider for each split, is dependent on the data and application area. To decide on optimal parameters, one technique is to optimize on some performance measurement during cross-validation which is described in Sections 3.5.1 and 3.5.3.

Suppose an input observation $\mathbf{x}$ and the corresponding response class y. Then the goal of the random forest method is to find a prediction function $f(\mathbf{x})$ for predicting y. The prediction function of random forest is constructed by combining a number of multiple decision trees, $h_1(\mathbf{x})$,..., $h_J(\mathbf{x})$. The decision trees are predicted independently and the prediction function is the most frequently predicted class of all

decision trees [18],

$$f(\mathbf{x}) = \underset{y \in C}{\operatorname{argmax}} \sum_{j=1}^{J} \mathbb{1}\{y = h_j(\mathbf{x})\},$$

where $\mathbb{1}\{\cdot\}$ is 1 if the condition inside the parenthesis holds and 0 otherwise.

3.3 Multi-layer Perceptron

Multi-layer perceptron (MLP) networks are a class of *feedforward artificial neural networks* that can detect nonlinear relationships in data and be used for classification purposes [16]. A MLP network consists of multiple *layers*, where the first layer is the input layer, then there are one or several *hidden layers* and finally, there is an output layer.

Suppose a MLP network of L hidden layers. The state of neuron i in a layer $l = 0, ..., L + 1$ in the network is denoted by $x_i^{(l)}$, where $l = 0$ corresponds to the input layer, which is simply the input data vector, and $l = L+1$ corresponds to the output layer. Each hidden layer and the output layer has a connection weight $w_{ji}^{(l)}$, $l = 1, .., L+1$, from the previous layer and a bias $\theta_j^{(l)}$, where i are indices of the previous layer and j are indices of the current layer. The states of hidden neurons, $x_j^{(l)}$, $l = 1, ..., L$, are calculated from the neurons $x_i^{(l-1)}$ at the previous layer as

$$x_j^{(l)} = g\left(\sum_{i=1}^{n_i} w_{ji}^{(l)} x_i^{(l-1)} - \theta_j^{(l)}\right),$$

where n_i is the number of neurons of layer $l - 1$ and $g(\cdot)$ is the *activation function*. Common choices of the activation function are the logistic sigmoid function $\sigma(x) = 1/(1+\exp(-x))$, the hyperbolic tan function $\tanh(x)$, and the rectified linear unit function $\text{ReLu}(x) = \max(0, x)$.

In the case of multilabel classification, it is common to calculate the output, $\hat{y} = x_j^{(L+1)}$, of a MLP network using the softmax function (3.2), as

$$\hat{y} = x_j^{(L+1)} = \text{softmax}\left(\sum_{i=1}^{n_i} w_{ji}^{(L+1)} x_i^{(L)} - \theta_j^{(L+1)}\right).$$

Let C be the set of $K = |C|$ number of classes available. The objective when training a MLP network is to minimize a loss function. One common such function is the

cross-entropy loss function [16], defined as

$$L_{CE}(\hat{y}, y) = -\sum_{j=1}^{K} \mathbb{1}\{y = c_j\} \log P(y = c_j|\mathbf{x}),$$

where y is the true class, $\hat{y}$ is the predicted class and $\mathbb{1}\{y = k\}$ is 1 for $y = k$ and 0 otherwise. Using the softmax function, and defining $z_j := \sum_{i=1}^{n_i} w_{ji}^{(L+1)} x_i^{(L)} - \theta_j^{(L+1)}$, the cross-entropy loss can be expressed as

$$L_{CE}(\hat{y}, y) = -\sum_{j=1}^{K} \mathbb{1}\{y = c_j\} \log \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}.$$

To minimize the loss, there are different optimization solvers. One solver is the *Adam* solver, proposed by [19]. It is a commonly used optimizer and it is based on stochastic gradient descent.

3.4 Synthetic Minority Over-sampling Technique

A class imbalanced data set is a data set with one or some classes containing significantly fewer or more observations. There exist various approaches to adjust a data set to a set where the number of observations is more equally distributed between the classes.

Previous research [20] has discussed sampling with replacement, but argues that this approach does not significantly increase the recognition of minority classes. With this background, the *synthetic minority over-sampling technique* (SMOTE), proposed by [21], is used in this thesis. The SMOTE technique performs over-sampling by creating 'synthetic' examples rather than by sampling with replacement.

Using SMOTE, the user can decide if over-sampling should be performed on the minority class, a number of minority classes, or on all classes. For each data sample of the selected class or classes, the technique creates synthetic samples along the line segments connecting the samples with their k nearest neighbors. Depending on the number of over-samples required, all or only some randomly chosen of the k nearest neighbors are considered. Consider a vector from a sample to one of its nearest neighbors multiplied by a random number between 0 and 1. This multiplied vector is added to the original sample to create a synthetic sample. For more information on the over-sampling procedure in SMOTE, see [21].

3.5 Classifier evaluation

When training a classifier model, it is of high importance to design the classifier in a way so that it can be generalized and used on data it has not seen before. A classifier that can classify the data it has been trained on very well but performs poorly on unseen data is probably having problems with overfitting. When performing classification experiments, a common approach is to divide the data set into different subsets for different usage. Usually, the data is divided into one set used for training, one set for validation and one set used for testing.

3.5.1 K-fold Cross-Validation

During model development and when tuning model and regularization/tuning parameters, some classifiers might perform better on some specific held out data but perform worse on other held out data. Such behavior could result in unfair performance measures. To avoid this, the *k-fold cross-validation* technique can be used. In this technique, instead of splitting training and validation data into two parts, one for training and one for validation, data are divided into k parts, often of equal size. One set at a time is then used as validation data on a model trained with the other $k - 1$ parts of the data. Performance metrics, see Section 3.5.3, can then be averaged over the k trials [22].

3.5.2 Confusion Matrix

Results from a classifier can be visually presented as a *confusion matrix*. Let C be the set of $K = |C|$ number of classes available. In the classification problem, the confusion matrix is a $K \times K$ matrix where rows correspond to true labels and columns correspond to predicted labels. Each element m_{ij}, $i = 1, ..., K$, $j = 1, ..., K$, in a confusion matrix is the number of samples from class c_i being predicted to class c_j. Hence, a perfect classifier would only have non-zero elements on the diagonal of the confusion matrix. The structure of a confusion matrix can be seen in Table 3.1 for a classifier of three classes.

		Predicted label		
		Class 1	**Class 2**	**Class 3**
True label	**Class 1**	m_{11}	m_{12}	m_{13}
	Class 2	m_{21}	m_{22}	m_{23}
	Class 3	m_{31}	m_{32}	m_{33}

Figure 3.1: Structure of a confusion matrix for a multilabel classification problem of $K = 3$ classes. Each element m_{ij}, $i = 1, 2, 3$, $j = 1, 2, 3$, is the number of samples from a true class c_i being predicted to class c_j.

3.5.3 Performance Metrics

To measure the performance of different trained classifiers, the classification performance metrics *accuracy*, *precision*, *recall* and *F1*-score can be used. In this section, these performance metrics are presented for a multilabel setting. Let C be the set of $K = |C|$ number of classes available. The formulas are presented using the confusion matrix definition from the Section above, where m_{ij} is the number of samples from class $c_i \in C$ being predicted to class $c_j \in C$.

First, *accuracy* is the number of correct predictions out of the total number of predictions, defined as

$$\text{Accuracy} = \frac{\sum_{i=1}^{K} m_{ii}}{\sum_{i=1}^{K} \sum_{j=1}^{K} m_{ij}},$$

for a multilabel classifier. However, accuracy alone does not cover the robustness of a classifier when having a class-imbalanced data set, where there is a significant disparity between the number of samples in each class. Since this is the case in this thesis, additional classification metrics are needed.

For a class $c_i \in C$, *precision* is the proportion of all observations predicted to this class that are truly in this class, defined as

$$\text{Precision}_i = \frac{m_{ii}}{\sum_{j \in C} m_{ji}}.$$

For a class $c_i \in C$, *recall* is the proportion of all observations truly belonging to this class that are also being correctly classified to this class, defined as

$$\text{Recall}_i = \frac{m_{ii}}{\sum_{j \in C} m_{ij}}.$$

Depending on the classification problem and the application of it, precision and recall can be considered of higher or lower importance. Precision and recall scores are sometimes summarized into one single $F1$-*score*. In the multilabel setting, there is one $F1$-score for each class $c_i \in C$ defined as

$$F1_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i},$$

which is the harmonic mean of the precision and recall scores [23].

So for multilabel classifiers, precision, recall and $F1$-score can be calculated for each class. However, there exist various definitions of how to summarize scores for multilabel classifiers. Since there is an imbalanced data set used for classification in this thesis, the *average weighted macro* summarized score is used, where the scores of different classes are weighted according to the number of observations in the corresponding class [24]. For the $F1$-score, if there are K classes, each with n_i, $i = 1, .., K$, number of observations, and $F1$-scores $F1_i$, $i = 1, .., K$, then the average weighted macro $F1$-score is defined as

$$F1_{macro}^{weighted} = \frac{\sum_{i=1}^{K} n_i \cdot F1_i}{\sum_{i=1}^{K} n_i}.$$

It should be noted that much information gets lost when generalizing the precision and recall scores to a $F1$-score and that even more information gets lost when generalizing further to an average weighted macro $F1$-score. Therefore it is of importance to consider all scores during the development of classifiers to have more robust evaluation.

4

Data

In this chapter, information on the data used in this thesis is presented. First, the overall structure of the dose-response data is presented. Second, specific information on the two data sets used in this thesis is presented, including information on clinical classes and the number of observations and variables.

4.1 Structure and Experiments

The dose-response data in this thesis is from experiments on rats performed further back by IRLAB. In total, data from experiments on 3730 rats is used. In these experiments, rats are given different chemical compounds in various doses, given by the $\mathbf{Y}$ data matrix. For more information on the compounds, see [1]. How rats respond to different doses of compounds are measured as biological response profiles and saved as a data matrix $\mathbf{X}$. More information on the data matrices is given below.

X Data

The $\mathbf{X}$ data matrix of biological response profiles contain both *behavioral* and *neurochemical* variables. Behavioral variables are, for instance, the average speed of a rat and the time spent in the central part of the cage. During one hour, time series of positions of the rats are measured. Afterward, 11 different main behavioral variables, such as average speed, are calculated based on the time series. The main variables are calculated from the time series at seven sampling frequencies from 0.25 to 25 Hz and pooled into 15 minute periods, giving 308 behavioral variables in total for each rat. Neurochemical variables are levels of different chemical substances in the brain. They are measured in the different brain regions striatum, cortex, and limbic. The measurements of neurochemical variables are done by killing the rats and dissecting brains immediately after measuring behavioral variables [1].

Y Data

All **X** variables are obtained from experiments and measured for rats that are given different compounds in various doses, given by the **Y** matrix. The **Y** matrix contains compound dose vectors from experiments. For each compound in one experiment, there are between three and four dose-levels examined and each dose-level is examined on four different rats. The dose-levels are given as milligrams or micro-moles per kilogram of the rat's weight. For different compounds, different dose-levels are relevant and therefore, the dose-levels vary between different compounds.

The **Y** variables are formed as a sparse matrix of 'dummy variables', where one column corresponds to one compound. Variable values in a column are numbers from 1 to 4 representing specific dose-levels, in ascending order, of the column compound. By using the dose-response data, it is possible to relate the biological response profiles of different compounds. For more information on the experiments and different compounds, see [1].

4.2 Data Sets

There are two data sets used in this thesis, one for dimensionality reduction and another for classification. The compounds in the data sets belong to different clinical classes, based on therapeutic findings and usage. These clinical classes are defined slightly different for the two data sets and more details are given below.

Data for Dimensionality Reduction

The data set used for dimensionality reduction consists of 850 observations of 55 different compounds where 228 behavioral variables and 20 neurochemical variables are considered. This data set contains both **X** and **Y** data matrices. The clinical classes in this data set are antidepressants (AD), drugs for attention deficit hyperactivity disorder (ADHD), cognitive/drugs for dementia (cogn), drugs for Parkinson, antipsychotics (AP) and drugs of abuse. There are also some compounds without clinical class, and these are denoted by *other*.

Data for Classification

The data set used for classification consists of 2880 observations where 20 neurochemical variables and 308 behavioral variables are considered. This data set contains only a **X** data matrix, but no **Y** data matrix. Hence, dose-levels are not taken

into account when developing classifiers. The class labels used for classifiers are the clinical classes based on therapeutic usage and previous research. The distribution of the these classes in the classification data set can be seen in Figure 4.1.

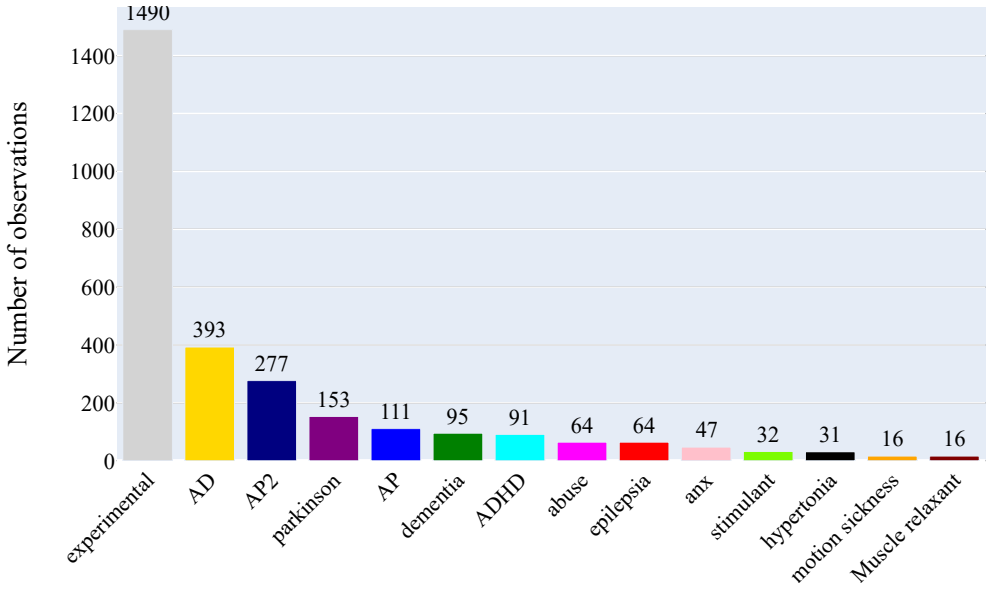


Figure 4.1: Distribution of clinical classes in the data set used for classification.

Classes with a too small number of observations are kept out from classification. These classes are compounds inhibiting anxiety (anx), stimulant, drugs for hypertonia, motion sickness and muscle relaxant. The class of epilepsy drugs (epilepsia) is kept out from classification since no such class is present in the data set used for dimensionality reduction.

The observations with label *experimental* do not have any verified clinical class. Therefore, these observations are only used for prediction and not for training, validation nor testing of classifiers.

5

Methods

In this chapter, the methods of this thesis are presented. First, the current methodology at IRLAB is presented in Section 5.1. Second, the pre-processing procedures used in this thesis are presented in Section 5.2. Finally, information on the implementations of methods used for dimensionality reduction and classification in this thesis is presented in Sections 5.3 and 5.4, respectively.

5.1 Current Methodology at IRLAB

The purpose of using PLS is often to use it to obtain a model for prediction where the model can predict $\mathbf{Y}$ from $\mathbf{X}$, see Section 2.1.1. However, this is not the case in this thesis. Instead, PLS is used to model the relationship between dose-response data obtained for each compound, since this is the approach described in [1]. In this section, the current visualization methodology at IRLAB is described followed by data pre-processing procedures.

5.1.1 Dimensionality Reduction Visualization

The weights $\mathbf{w}^*$ and $\mathbf{c}$ for $\mathbf{X}$ respectively $\mathbf{Y}$ variables are plotted for the principal components from linear PLS, see [1], to compare results visually. An example of such a weight plot can be found in Figure 2.2 in Section 2.1.2. Visualizations are made easier to interpret, especially to people with limited knowledge of compounds, by putting all compounds into different clinical classes, based on therapeutic findings and usage, where each class is represented by a specific color.

5.1.2 Pre-processing of Data in Current Methodology

The software used for linear PLS regressions by IRLAB is called SIMCA, see [6], and it performs some pre-processing, so-called *auto-scaling*, on data before regression.

The first step of auto-scaling is to replace zeros in the **X** data, to later be able to transform it logarithmically. If a zero is encountered, it is replaced by half of the smallest value of that variable among all observations. This means that all zeros are replaced by half of the smallest value in its column.

Moreover, PLS performs better if the data is fairly symmetrically distributed [6]. Therefore, if the data has an asymmetric distribution, the data is usually logarithmically transformed before analysis [6]. The next step in auto-scaling is therefore to logarithmically transform the **X** data with base 10. The logarithmic transformation is followed by *mean-centering* of both **X** and **Y** matrices, which is to subtract the average value of each variable from the data.

Further, variables are scaled, meaning that each coordinate axis in the variable space is regulated according to some criteria. Since a variable with a wide numerical range has a higher variance, and principal components seek to maximize the variance, it is more likely for high variance variables to be expressed in a model. This fact results in a model that will not reflect low variance variables as much as high variance variables, why it is important to scale variables. The next step after mean-centering is therefore to perform *unit variance* (UV) scaling of both data matrices **X** and **Y**. UV-scaling means that the variables, stored in the columns of the data matrices, are divided by the standard deviation of the corresponding variable. In this way, each variable has equal unit variance, corresponding to variables being equally important *a priori*. In cases when there is no a priori knowledge about the data, it is essential not to perform scaling subjectively, resulting in a modification of the model that is adjusted towards some expected or wanted result. Therefore it is crucial to perform scaling objectively and UV-scaling is, in general, an objective approach [6].

Lastly, *block scaling* is performed on the **X** variables. Block scaling is a technique that can be used when there are different *types* of variables in a data matrix, for instance corresponding to that they have different origins. In the data of this thesis, the different types of variables are neurochemical and behavioral variables, see Chapter 4. If it is a considerable difference in the number of variables of different types in the data, then different types of variables will affect the model to a variously large extent. However, if the intention is that the different types of variables should be equally reflected in the model, the variables can be divided into different blocks based on the variable type. The variables in different blocks are then multiplied by different relevant factors. In this thesis, the variables are divided by the square root of the number of variables in its block.

5.2 Pre-processing of Data

In this thesis raw data was provided. Therefore, the same pre-processing steps as in the auto-scaling process in SIMCA, see Section 5.1.2 above, are performed in `Python` when replicating the linear PLS results of IRLAB. Since the replicated linear PLS results act as a benchmark for parts of the explorative work, the pre-processing steps described in Section 5.1.2 are also performed before applying the non-linear dimensionality reduction methods and classification methods of this thesis.

5.3 Dimensionality Reduction

Different methods for dimensionality reduction are investigated in the scope of this thesis and some of the most promising ones are implemented and finally evaluated visually. The evaluation of different methods is qualitative and consists of visually comparing results with each other and against results from linear PLS, which is the method used in the current methodology, see Section 5.1. Visualizations are also evaluated by consulting IRLAB and the intuition they possess.

In the upcoming Sections 5.3.1, 5.3.2 and 5.3.3 it is described how different methods for dimensionality reduction are implemented in this thesis. Afterwards, in Section 5.3.4, follows a remark on other methods for dimensionality reduction and finally, in Section 5.3.5, it is described how results from dimensionality reduction methods are visualized in this thesis.

5.3.1 Implementation of Partial Least Squares

A NIPALS-PLS regression, see Section 2.1.1, is implemented using the `Python` library `python-nipals` [25]. In this package, missing data is handled as in the `R` package `nipals` [26]. The handling of missing values in the NIPALS-PLS algorithm is based on the idea that single elements in the loading and score vectors corresponding to missing values in the original data should be skipped in calculations. More precisely, suppose a certain variable value x_{ik} of observation, i.e. row, $i = 1, ..., n$ of variable, i.e. column, $k = 1, ..., K$ in $\mathbf{X}$ is missing. Then the corresponding score elements t_{ih}, $h = 1, ..., K$, should be skipped in the calculation of the loadings p_{hk}. In the same way, if a certain variable value x_{ik} is missing, the corresponding loading elements p_{kh} must be skipped when calculating the score values t_{ih}. In the NIPALS-PLS regression algorithm, see Section 2.1.1, this idea is implemented by making sure that all normalizations of score vectors $\mathbf{t}$ and $\mathbf{u}$ and loading vectors $\mathbf{w}$

and $\mathbf{c}$ are done in a manner such that numbers corresponding to missing values in $\mathbf{X}$ respectively $\mathbf{Y}$ are not included. The algorithm saves positions of missing values in the $\mathbf{X}$ and $\mathbf{Y}$ matrices and replaces them with zeros. After calculating scores and loadings, but before normalizing, the values in the score and loading matrices corresponding to missing values in $\mathbf{X}$ and $\mathbf{Y}$ are replaced by zeros.

5.3.2 Implementation of Kernel Partial Least Squares

Kernel PLS is implemented in this thesis using the `python-nipals` package as a basis. In this package, normalizations are performed on the loadings $\mathbf{w}$ and $\mathbf{c}$. To be able to add a kernel, normalizations are applied on the scores $\mathbf{t}$ and $\mathbf{u}$ instead, as in Section 2.1.1. Apart from changing the normalization, the only other adjustment of `python-nipals` to perform kernel PLS is to add the calculation of the kernel matrix $\mathbf{K}$ from the $\mathbf{X}$ data, see Section 2.2. The kernel functions used in this thesis are radial basis functions (2.2) with width parameters in the range $\gamma \in [0.5, 2]$. Polynomial kernels with kernel functions of the form (2.3) are also used. Parameters considered are combinations of $a = 1, 2, 3, 4$ and $b = 0, 1, 2, 3, 4, 5, 10, 50, 100$.

Due to the use of the kernel matrix, it is not possible to handle missing values in the implementation of kernel PLS, as in the implementation of linear PLS described in Section 5.3.1. This fact is a result of that the kernel NIPALS-PLS algorithm, see Section 2.2, does not calculate $\mathbf{t}$ scores directly from the $\mathbf{X}$ matrix, as in the NIPALS-PLS algorithm, see Section 2.1.1. Instead, the kernel NIPALS-PLS algorithm calculates $\mathbf{t}$ scores from the kernel matrix $\mathbf{K}$. Hence, it is not possible to make sure that all normalizations of $\mathbf{t}$ vectors are done in a manner such that numbers corresponding to missing values in $\mathbf{X}$ are not included, as in the implementation of the NIPALS-PLS algorithm, described in Section 5.3.1. Therefore, the somewhat more naive approach to replace missing values with zeros is used when using kernel PLS in this thesis. The replacement with zeros is implemented after the pre-processing steps. In the pre-processing steps, the missing values are excluded from calculations of mean values and variances of variable columns. Since pre-processing includes mean-centering, replacing missing values with zeros corresponds to replacing missing values with the mean value of its variable among all observations that do not have missing values.

5.3.3 Implementation of T-distributed Stochastic Neighbor Embedding

To implement t-SNE, the `Python` library `sklearn.manifold.TSNE` [27] is used. In the implementation, various values of the perplexity parameter are used in a range from 5 to 50 and embeddings are fit during 1500 iterations. For each run, the embedding is set to a random initial state and a learning rate of 200 is used. Just as for kernel PLS, missing data is replaced with zeros when using t-SNE in this thesis.

5.3.4 Remark on Methods for Dimensionality Reduction

Except from the methods described in the subsections above, some other non-linear methods for dimensionality reduction were tested. These methods are *Isometric Embedding* [28], *Locally Linear Embedding* [29], *Multi-dimensional Scaling* [30], *Spectral Embedding* [31] and *Uniform Manifold Approximation and Projection* [32]. However, the experiments performed using these methods gave poor results or results very similar to linear PLS. Therefore, these methods are not further evaluated in the scope of this thesis.

5.3.5 Visualization of Observations

The current methodology at IRLAB consists of visualizing $\mathbf{w}^*$ and $\mathbf{c}$ weights from linear PLS, as mentioned in Section 5.1.1. In a plane of two principal components, this yields one point per compound from $\mathbf{c}$ and one point per variable from $\mathbf{w}^*$. All the methods for dimensionality reduction in this thesis do not result in corresponding results for variable positions. Therefore, visualizations in this thesis only consider positions of compounds and observations where the observations are the rats given different compounds. Also, the methods in this thesis do not give one point per compound in the same manner as in the linear PLS. Therefore, a technique for summarization of observations of the same compound is developed. The output from the methods gives one score point per single observation. All points of observations corresponding to the same compound are summarized by an average and a weighted average where the weights are the compound dose of an observation. The averages are visualized as points colored according to the clinical class of the corresponding compounds. From these points, there are grey arrows to the weighted average. Results visualized using this technique are presented in Section 6.1.

5.4 Classification

In this section, it is described how the different methods for classification, see Chapter 3, are investigated in the scope of this thesis. First, Section 5.4.1 explains how data is treated along with information on the training process and how evaluation is performed. Finally, the implementations of all classifiers, using the `scikit-learn` library [33] in `Python`, are described in more detail in Sections 5.4.2, 5.4.3 and 5.4.4.

5.4.1 Classifier training and evaluation

Before applying classification methods, the data with selected and known clinical classes, described in Chapter 4, are divided into two sets. One set is used for training and validation and the other set is used for testing. The two sets are obtained using stratified sampling, which means that the proportion of classes in the two sets are as similar as possible to the distribution of classes in the original data set. To obtain this, two observations of each compound are in the test set while the others are in the training and validation set. The two observations picked to be in the test set are selected so that lower and higher dose levels are distributed as equally as possible between the test set and the set for training and validation. From this, 12.8% of the observations are in the test data set and 87.2% are in the data set used for training and validation. The distribution of different classes in the test data set and in the training and validation data set can be seen in Figure 5.1.

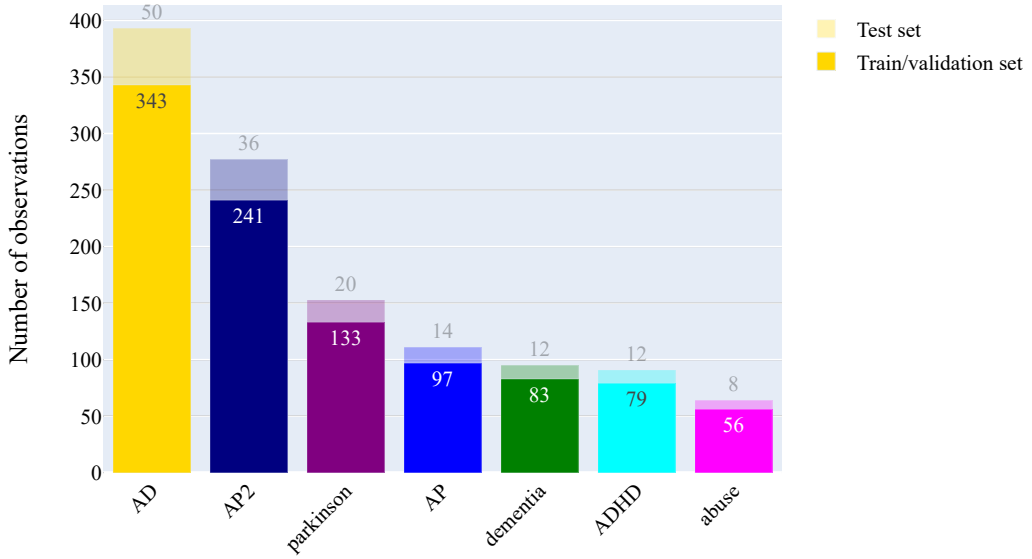


Figure 5.1: Distribution of clinical classes in the test data set and in the training and validation data set used for classification. The test data set can be seen as lighter colors on top of the stacks, and the training and validation data set can be seen as brighter colors in the bottom of the stacks.

All classifiers are developed using the training and validation set and, in the very end, evaluated on the test set. Classification methods considered are multinomial logistic regression, random forest and multi-layer perceptron networks. Each method is trained both on the initial training and validation data set and on a SMOTE data set, which gives six different classifiers to evaluate. When evaluating the training process of classifiers, k-fold cross-validation is used, see Section 3.5.1. More specifically, the data set used for training and validation is divided into $k = 5$ parts using stratified sampling and performance metrics are calculated for each of the k trials and the average of these trials is considered for evaluation. The best suitable parameters for each of the classification methods are obtained by searching through different sets of parameters to find the ones giving the highest average weighted macro $F1$-score from the k-fold cross-validation, see Sections 3.5.1 and 3.5.3. The $F1$-score is used since precision and recall are considered to be equally important in the scope of this thesis. The average weighted macro $F1$-score is chosen to incorporate the performance of the smaller classes to a larger extent even though the data set is class imbalanced.

The confusion matrices and classification metrics of the different classifiers are considered to evaluate classifier performance. Also, qualitative evaluation is performed by consulting IRLAB and the expertise they possess. The data of experiments on compounds classified as *experimental* compounds that do not have any verified clinical class are predicted by the classifiers. From this, the probabilities of each observation being in the different classes are obtained. Then the probabilities are averaged over dose-levels, mostly containing 4 observations. The averages are presented as histograms where there is one histogram of each experiment having stacks for each dose-level and class. These results are evaluated by comparing them with tentative clinical classes from IRLAB and by consulting people with expertise.

5.4.2 Implementation of Multinomial Logistic Regression

To implement multinomial logistic regression, the method `sklearn.linear_model.LogisticRegression` [34] is used with the `multi-class` parameter set to 'multinomial'. Further, $L2$-regularization is used with the parameter set to the `scikit-learn` default value of 1. When the gain from updating the model weights falls below a tolerance of $1 \cdot 10^{-4}$, training is stopped.

5.4.3 Implementation of Random Forest

The method `sklearn.ensemble.RandomForestClassifier` [35] is used to implement random forest. The gini impurity is used to find the best splits in each decision tree, see Section 3.2. Different combinations of parameters are tried and the combination giving the highest average weighted macro $F1$-score is chosen. For the optimal minimum samples required to split a node, values tried are in the range $[2, 10]$. For the number of decision trees used in the ensemble, values in the range $[10, 200]$ are tried. The number of features to consider at each branch split is varied between half of the square root of the total number of variables up to the total number of variables.

5.4.4 Implementation of Multi-layer Perceptron

For the multi-layer perceptron, `sklearn.neural_network.MLPClassifier` [36] is used. Networks of one and two hidden layers with between 50 and 350 neurons are tested and the network giving the highest average weighted macro $F1$-score is used in further evaluation. The classifier is fit using the ReLu-activation function and a constant learning rate of 0.001 is used. Further, $L2$-regularization is used and the penalty parameter is set to the `scikit-learn` default value of 0.0001. The maximum number of iterations allowed is set to 500 and when the gain from updating the model weights falls below a tolerance of $1 \cdot 10^{-4}$, training is stopped.

6

Results

In order to evaluate the methods described in Chapters 2 and 3, this chapter presents some visual results and performance metrics from the explorative work described in Chapter 5. Visual results from dimensionality reduction methods are presented in Section 6.1 and results from classification methods are presented in Section 6.2.

6.1 Dimensionality Reduction

Results from dimensionality reduction methods are visually presented in the following Sections. Results from the linear PLS, acting as a benchmark for further analysis, are presented in Section 6.1.1. In the following Sections 6.1.2 and 6.1.3, results from the nonlinear methods kernel PLS and t-SNE are presented and compared to each other and to the results from linear PLS. Finally, the results are qualitatively evaluated, partly based on the expertise of IRLAB in Section 6.1.4.

It should be noted that the scales of the axes in a dimensionality reduction visualization are only relevant for comparison within the plot and not between plots.

6.1.1 Partial Least Squares

The result from using linear PLS shows a clustering pattern consistent with results from the SIMCA software used by IRLAB, i.e. shows the same patterns and clusters. The result is shown in a **c** weight plot in Figure 6.1. Figure 6.2 shows the result from the same linear PLS implementation using the technique for summarizing scores, described in Section 5.3.5.

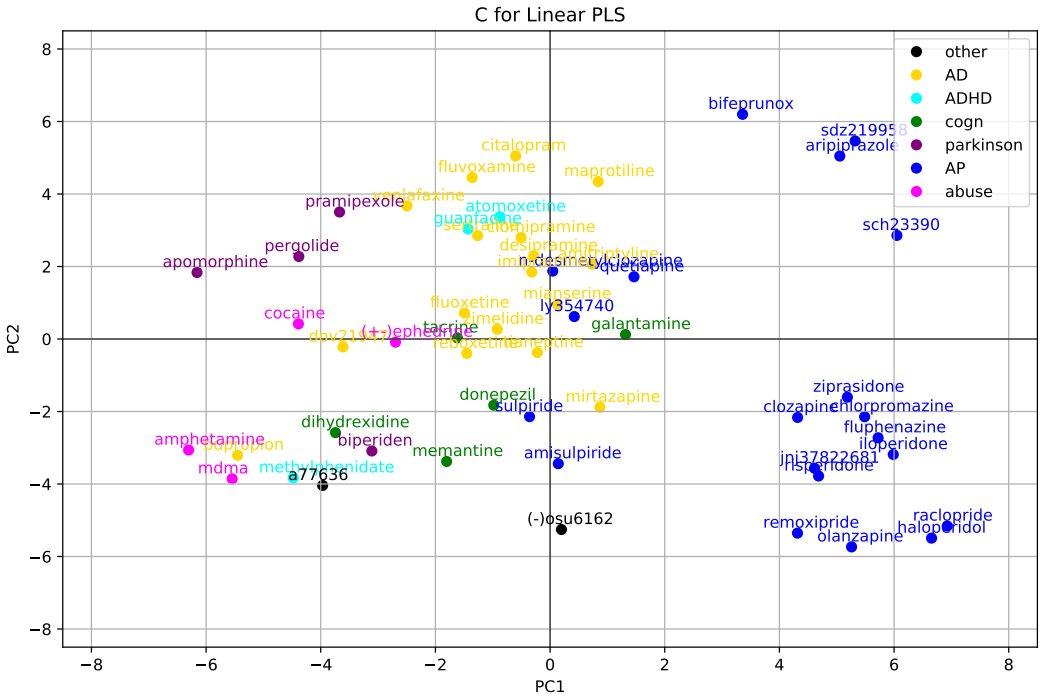


Figure 6.1: Linear PLS **c** weights of the first and second principal components.

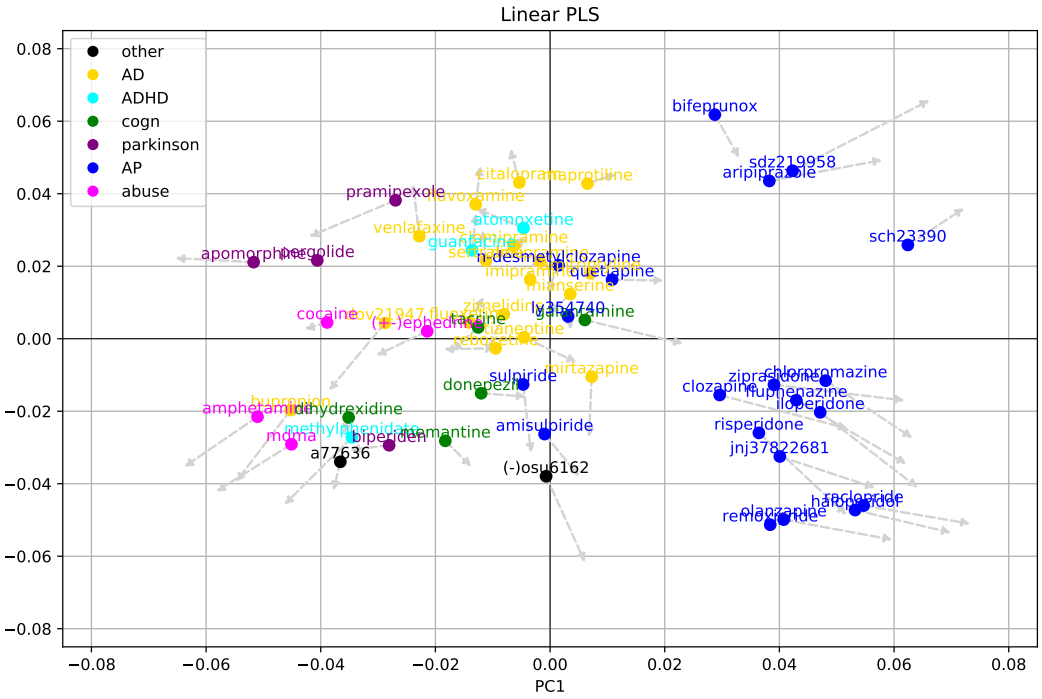


Figure 6.2: Plot of the first and second principal component scores, $\mathbf{t}_1$ and $\mathbf{t}_2$, for the linear PLS method. Colored dots are the average of all observations, i.e. scores, for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

6.1.2 Kernel Partial Least Squares

In this section, results from the implementation of kernel PLS are visualized for some of the most interesting kernel functions.

For RBF kernels (2.2), results for the width parameter γ between approximately 0.8 and 1.3 give very similar results. Having the width parameter γ lower and higher than these values result in similar, but not as distinct, clustering patterns and are therefore considered less informative. These results can be found in Appendix A.1. The result with parameter $\gamma = 1$ can be seen in Figure 6.3. In this figure, the antipsychotic (AP) compounds are somewhat more densely clustered compared to the linear PLS in Figure 6.2. However, compounds in other classes are a bit more spread out compared to linear PLS. Compared to linear PLS, it can also be seen that the drugs of abuse are less densely clustered and more spread out among the other classes.

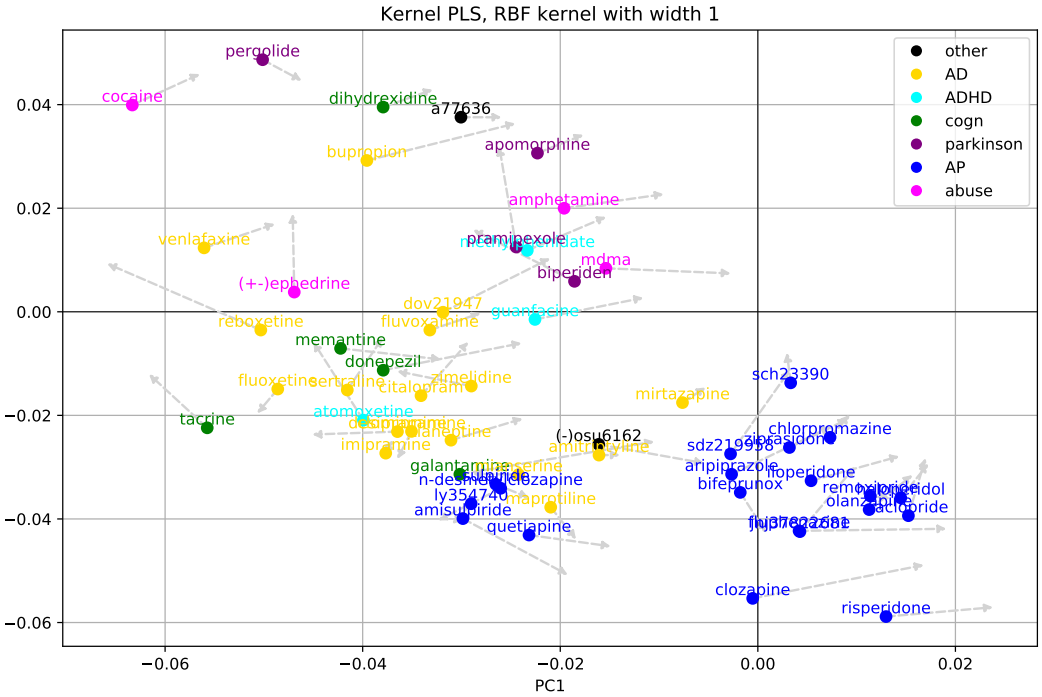


Figure 6.3: Principal component scores, $\mathbf{t}_1$ and $\mathbf{t}_2$, of RBF Kernel PLS with width $\gamma = 1$. Colored dots are the average of all observations, i.e. scores, for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

For polynomial kernel PLS with kernel functions given by (2.3), mainly the special case of quadratic kernels, $a = 2$, shows interesting results. Higher a values seem to generate poor plots, meaning they do not show any patterns or clusters among the compounds. Polynomial kernels with $a = 1$ do not provide additional information to the linear PLS for any value of b. Results supporting these statements can be found in Appendix A.2.

For the quadratic kernel with $b = 0.5$, the antipsychotic (AP) compounds are distinctly spread out while it does not give much information about compounds in the other classes, see Figure 6.4. Quadratic kernels with even smaller b values show similar behavior, but a bit shifted, not providing any additional information. Increasing the b value shows a more and more similar behavior as the linear PLS. The result for the polynomial kernel with $a = 2$ and $b = 4$ in Figure 6.5, shows a behavior similar to the linear PLS in Figure 6.2. Increasing the b value further shows a more and more similar behavior as the linear PLS.

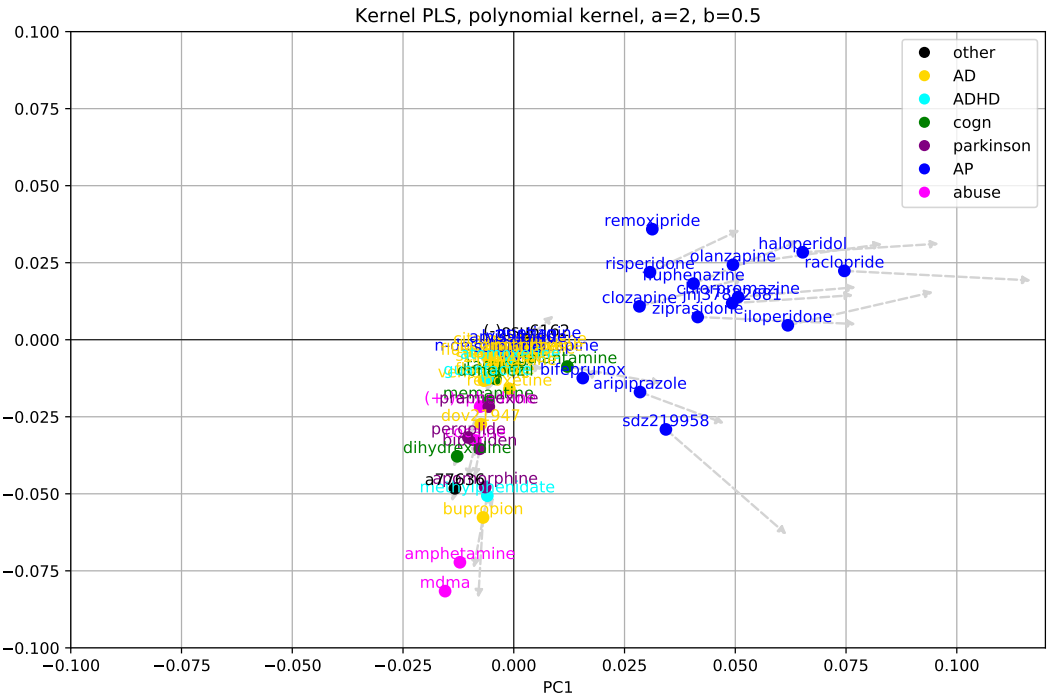


Figure 6.4: Principal component scores, $\mathbf{t}_1$ and $\mathbf{t}_2$, of polynomial Kernel PLS with $a = 2$ and $b = 0.5$. Colored dots are the average of all observations, i.e. scores, for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

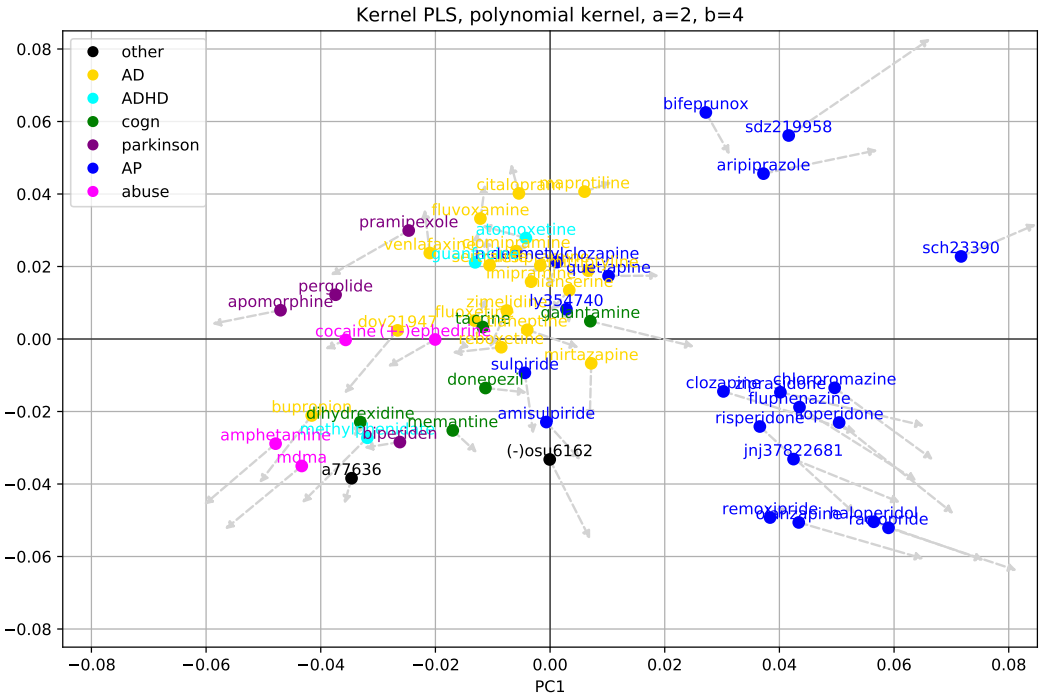
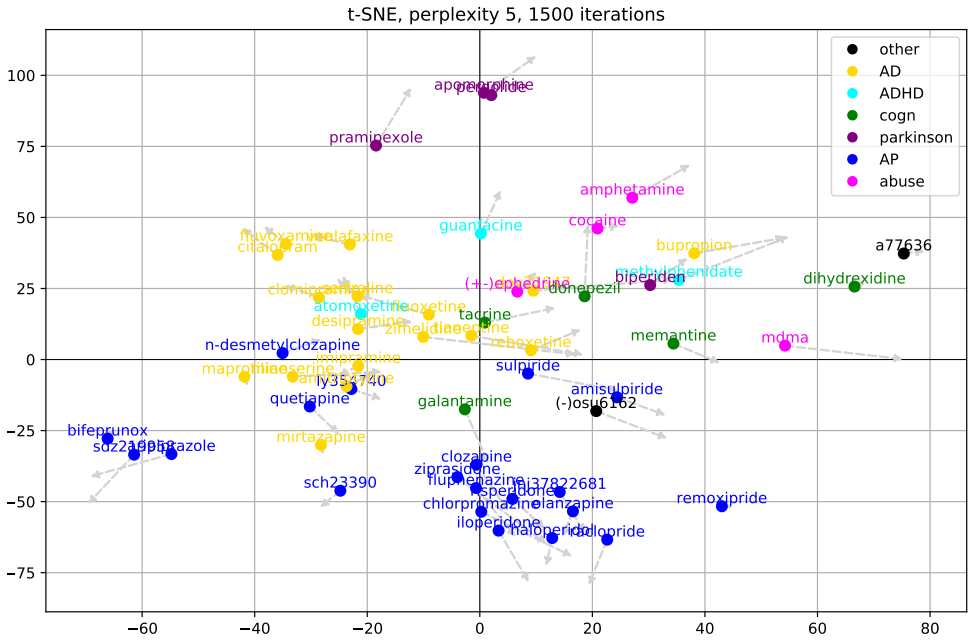


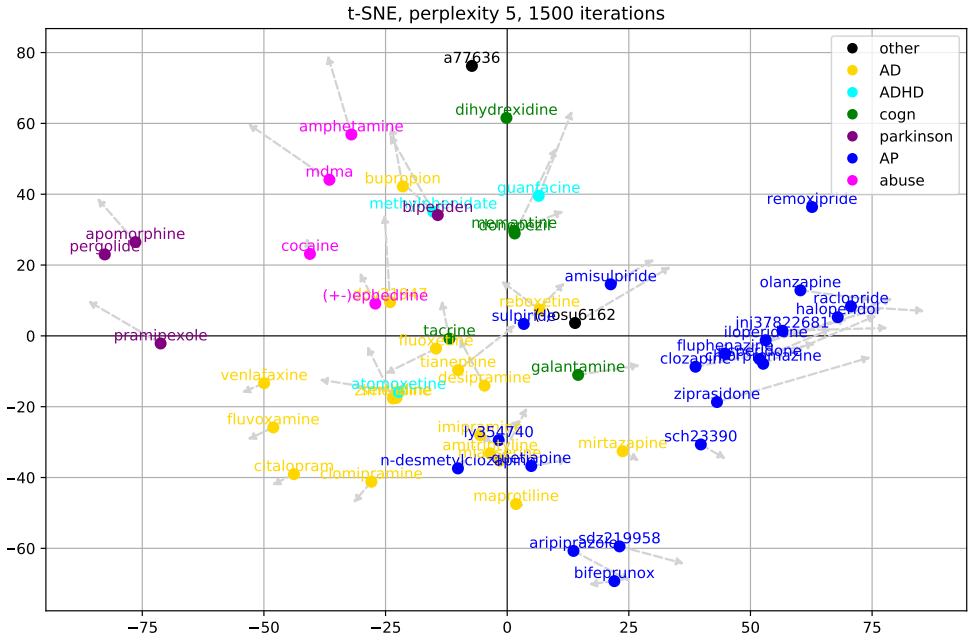
Figure 6.5: Principal component scores, $\mathbf{t}_1$ and $\mathbf{t}_2$, of polynomial Kernel PLS with $a = 2$ and $b = 4$. Colored dots are the average of all observations, i.e. scores, for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

6.1.3 T-distributed Stochastic Neighbor Embedding

In this section, results from the implementation of t-SNE using some different values of the perplexity parameter are presented. The results in Figure 6.6 are two embeddings yielded with the perplexity parameter set to 5. Both these embeddings are presented to illustrate that different runs of t-SNE with the same perplexity value result in different solutions, see Section 2.3. The global location of clusters differs quite much between these embeddings, while they show a similar behavior within clusters on a more local level. As an example, most of the compounds in the Parkinson's class are clustered together in the two embeddings but in different areas of the plots. This behavior might be due to the low perplexity value of 5 that, according to theory, gives more attention to local features, see Section 2.3.



(a) First t-SNE run with perplexity 5.



(b) Second t-SNE run with perplexity 5.

Figure 6.6: Two dimensional embeddings from two runs of t-SNE with perplexity 5. Colored dots are the average of all observations for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

A result from t-SNE with a perplexity value of 50 can be seen in Figure 6.7. Compared to the results in Figure 6.6 for smaller perplexity value, this result is more similar to the linear PLS, which might be due to the fact that higher perplexity values give more attention to global features, see Section 2.3.

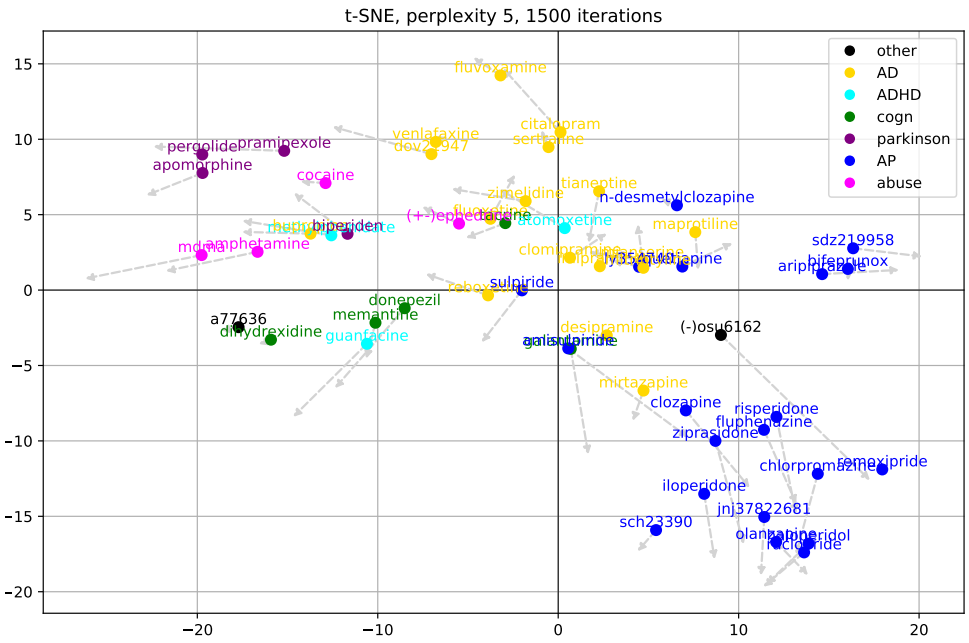


Figure 6.7: Two dimensional embedding from t-SNE with perplexity 50. Colored dots are the average of all observations for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

6.1.4 Qualitative Evaluation

Qualitatively, IRLAB considers some results to be promising for further exploration even though the results do not show any very informative or new patterns or clusters. The t-SNE results, in particular the result in Figure 6.6b, show behavior that could be interesting as a springboard for further methodology development and analysis. This behavior is that the antipsychotic and antidepressant compounds are quite distinctly spread out and that the overlap is consistent with the intuition of the compounds included. However, the result from the other run with the same perplexity value, in Figure 6.6a, are somewhat different. These observations illustrate that different runs of the t-SNE algorithm with the same perplexity parameter can show different results, as described in Section 2.3. Therefore, a possible methodology approach could be to run t-SNE with the same perplexity value several times

and evaluate if patterns and clusters that seem to be recurring in different runs can be found.

Concerning kernel PLS, the RBF kernel in Figure 6.3 does not give very distinct information on antipsychotic compounds but more distinct information of the other compounds. On the other hand, the polynomial kernel in Figure 6.4 show more distinct information on antipsychotic compounds, but not the behavior of other compounds. Therefore, one idea to create insight from kernel PLS could be to consider them complementary.

6.2 Classification

In this section, results from investigating the classification methods of Chapter 3 are presented, commented on and evaluated. First, in Section 6.2.1, the parameter values showing best performance for the different methods are presented. Further, in Sections 6.2.2 and 6.2.3, performance metrics from cross-validation on the training data set and confusion matrices of the test data set are presented for each method using the parameter values giving best performance. Finally, in Section 6.2.4, some selected and interesting results from the experimental data set are presented and qualitatively evaluated for the method showing best performance.

6.2.1 Parameter Values

For the random forest classifier, the combination of parameter values giving the highest average weighted macro $F1$-score is 3 for the minimum number of samples required to split a node and 40 for the number of decision trees used in the ensemble. Further, the maximum features to consider for each split set to the square root of the total number of variables gives the highest average weighted macro $F1$-score. The depths of individual decision trees in the forest are in the range $[16, 21]$.

For the multi-layer perceptron, a network with 2 hidden layers, each having 250 neurons, gives the highest average weighted macro $F1$-score. The maximum number of iterations is never reached. These parameters are used in the following results.

6.2.2 Performance Metrics from K-fold cross-validation

In this section, performance metrics from using the parameter values of the previous section are presented for the different classification methods.

	Multinomial logistic regression			**SMOTE Multinomial logistic regression**		
Clinical class	Precision	Recall	$F1$-score	Precision	Recall	$F1$-score
AD	0.5164	0.6977	0.5814	0.6578	0.4294	0.4908
ADHD	0.3048	0.1792	0.2243	0.1902	0.3433	0.2408
AP	0.3330	0.2037	0.2507	0.3461	0.5216	0.4133
AP2	0.5777	0.6220	0.5877	0.6030	0.4395	0.4975
abuse	0.4111	0.1424	0.1842	0.3442	0.4333	0.3624
dementia	0.2610	0.2449	0.2481	0.2309	0.4272	0.2919
parkinson	0.3966	0.3487	0.3322	0.3398	0.3407	0.3255
macro average	0.4001	0.3484	0.3441	0.3874	0.4193	0.3746
macro weighted average	0.4551	0.4816	0.4435	0.4875	0.4215	0.4211

Table 6.1: Classification metrics of multinomial logistic regression averaged over k-fold cross-validation, $k = 5$. Using the original data set for training and validation (left) and the SMOTE data set for training and validation (right).

	Random forest			**SMOTE Random forest**		
Clinical class	Precision	Recall	$F1$-score	Precision	Recall	$F1$-score
AD	0.5084	0.7585	0.6045	0.5321	0.6710	0.5886
ADHD	0.4590	0.1133	0.1689	0.2943	0.1775	0.2123
AP	0.5333	0.1642	0.2277	0.3558	0.2458	0.2873
AP2	0.5863	0.6425	0.6026	0.5872	0.6425	0.6079
abuse	0.6087	0.4697	0.4971	0.5048	0.3606	0.3671
dementia	0.4128	0.1838	0.2422	0.3174	0.2824	0.2885
parkinson	0.3956	0.2877	0.3172	0.3991	0.2641	0.3080
macro average	0.5006	0.3743	0.3800	0.4272	0.3777	0.3800
macro weighted avgerage	0.5079	0.5029	0.4628	0.4740	0.4854	0.4632

Table 6.2: Classification metrics of random forest averaged over k-fold cross-validation, $k = 5$. Using the original data set for training and validation (left) and the SMOTE data set for training and validation (right).

	Multi-layer perceptron			**SMOTE Multi-layer perceptron**		
Clinical class	Precision	Recall	$F1$-score	Precision	Recall	$F1$-score
AD	0.6000	0.5928	0.5715	0.5928	0.5780	0.5666
ADHD	0.2630	0.2683	0.2409	0.2874	0.2692	0.2611
AP	0.4064	0.4184	0.4049	0.3960	0.4289	0.4073
AP2	0.6331	0.6137	0.6180	0.6166	0.5846	0.5966
abuse	0.5542	0.4833	0.5017	0.5151	0.5197	0.5065
dementia	0.3847	0.4265	0.3869	0.3767	0.4162	0.3664
parkinson	0.5237	0.3638	0.3477	0.5261	0.3872	0.3734
macro average	0.4807	0.4524	0.4388	0.4730	0.4548	0.4397
macro weighted average	0.5347	0.5068	0.4937	0.5265	0.5000	0.4905

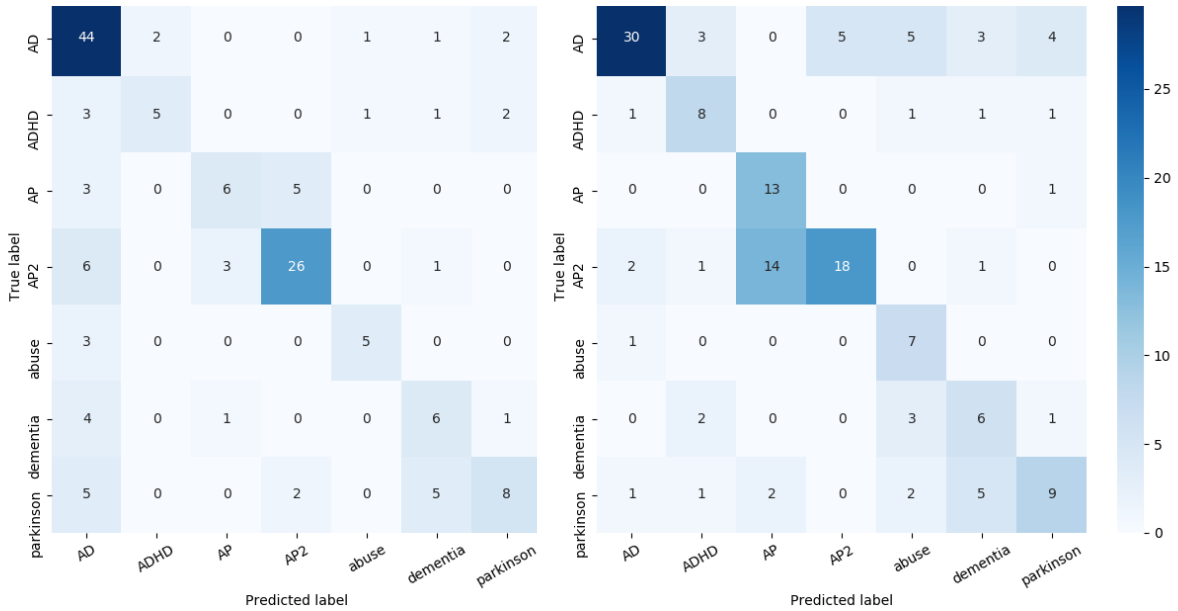
Table 6.3: Classification metrics of a multi-layer perceptron with 2 hidden layers, 250 neurons each, averaged over k-fold cross-validation, $k = 5$. Using the original data set for training and validation (left) and the SMOTE data set for training and validation (right).

The results presented in the tables above are performance metrics averaged over $k = 5$ cross-validation rounds for the training and validation data set, see Section 3.5.1. The different performance metrics are presented and defined in Section 3.5.3. For all metrics, a higher value corresponds to better performance. A perfect classifier that predicts all observations correctly would have all values of precision, recall and $F1$-scores equal to 1.

From the tables above, it can be seen that the average weighted macro $F1$-score is highest for the multi-layer perceptron network without SMOTE data. Moreover, the random forest classifiers in Table 6.2 give higher average weighted macro $F1$-scores than the multinomial logistic regression classifiers in Table 6.1. For the random forest classifiers, the average weighted macro $F1$-score is slightly higher for the classifier trained with SMOTE data than the classifier trained on original data. For the multinomial logistic regression and multi-layer perceptron networks, using SMOTE data does not improve the average weighted macro $F1$-score.

6.2.3 Confusion matrices from Classification of Test Data

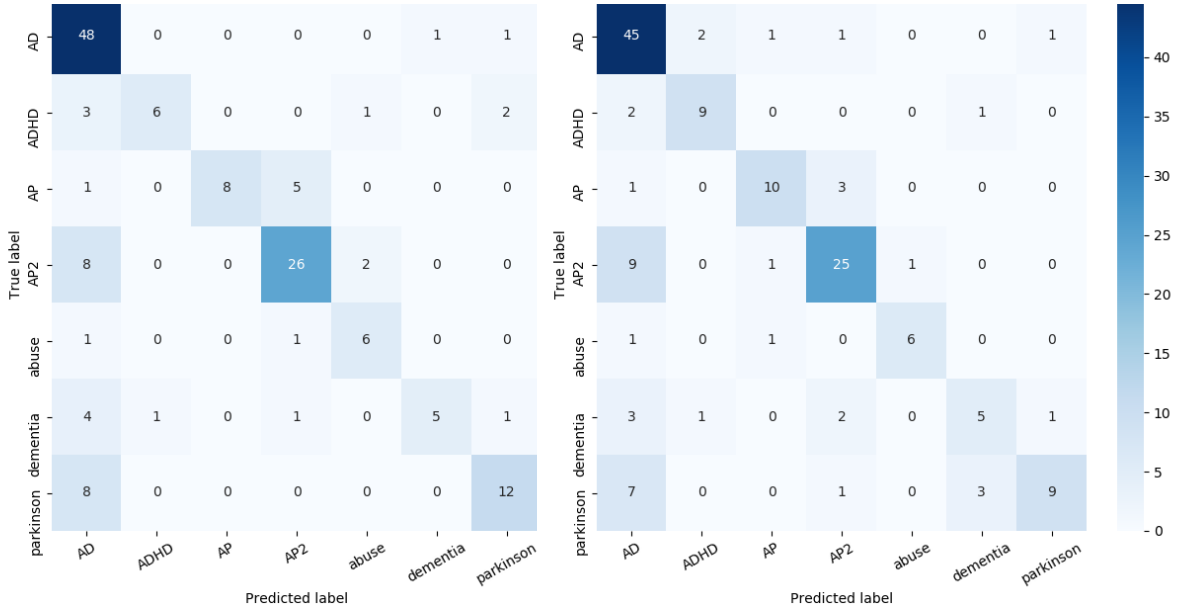
In this section, confusion matrices are presented for predictions of the test set for all classifiers from the Section above.



(a) Multinomial logistic regression.

(b) SMOTE Multinomial logistic regression.

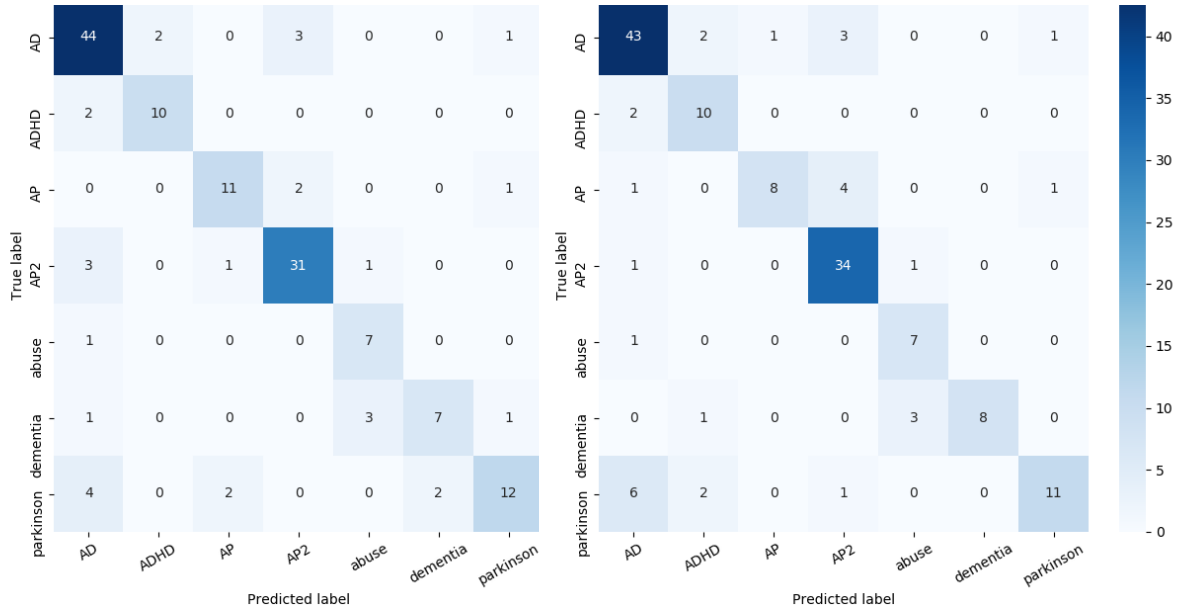
Figure 6.8: Confusion matrix of the test set classified by a multinomial logistic regression trained on the original training set (left) respectively trained on the SMOTE training set (right).



(a) Random forest.

(b) SMOTE Random forest.

Figure 6.9: Confusion matrix of the test set classified by a random forest trained on the original training set (left) respectively trained on the SMOTE training set (right).



(a) Multi-layer perceptron. **(b)** SMOTE Multi-layer perceptron.

Figure 6.10: Confusion matrix of test set classified by a multi-layer perceptron with 2 hidden layers, 250 neurons each, trained on the original training set (left) respectively trained on the SMOTE training set (right).

From the confusion matrix of the test data in Figure 6.10a above, it is indicated that the multi-layer perceptron classifier without SMOTE performs well. Compared to the same classification method trained on SMOTE data, see Figure 6.10b, it is not very clear which one performs better. The classifier trained with SMOTE data is somewhat better at correctly classifying classes AP2 and dementia, while the classifier trained on original data is better at correctly classifying AD and AP. However, the difference is not very high.

For the random forest classifiers, see Figure 6.9, there are small differences when comparing the classifier trained on original data to the classifier trained on SMOTE data. By comparing the random forest classifiers to the multi-layer perceptron classifiers, the random forest classifiers seem to be more likely to classify different classes as AD falsely. The number of correctly classified experiments in the AD class is slightly higher for the random forest classifiers, while the numbers of correctly classified experiments in most of the other classes are higher for the multi-layer perceptron classifiers.

Finally, the multinomial logistic regression classifiers in Figure 6.8 show a smaller amount of correctly classified experiments. Just as for the other classifiers, several experiments are falsely classified as AD. One could argue that this is due to the

high amount of samples of this class in the training set. When performing SMOTE, the multinomial logistic regression classifier does not classify as many experiments falsely as AD, which supports this argument. Another argument for the reason of falsely classified experiments as AD is that it is due to that AD compounds often have lower effects than compounds in the other clinical classes. Since there is no class for control groups, i.e. experiments with compounds that show no effect, it could be that compounds giving small or no effect at all are falsely classified as AD.

Overall, results from the classification methods indicate that the multi-layer perceptron classifier trained without SMOTE data performs better than the other classifiers. It also has the highest average weighted macro $F1$-score. Therefore, the results from classifying experimental data, in the section below, will be presented using this classifier.

6.2.4 Qualitative Evaluation of Experimental Data

The experiments with label experimental, i.e. data without verified clinical class, are predicted using a multi-layer perceptron classifier trained on the original training set since this classifier gives the highest macro weighted average $F1$-score. The results are compared to tentative clinical classes and previous research. Note that evaluation is performed by consulting people with expert knowledge at IRLAB, and therefore, it is partly based on their expertise and intuition.

The classifications are evaluated and considered by IRLAB to be of one of three groups. First, one group contains classifications showing *well-known effects* which are consistent with tentative clinical classes and the intuition of IRLAB. Second, another group contains classifications showing *potentially unknown effects* which are interesting or surprising results that could, in some cases, be a springboard for further analysis and innovation. Finally, the last group contains classifications showing *wrong effects*, which are results utterly different from the intuition of IRLAB and tentative clinical classes.

Out of the experimental compounds classified, IRLAB has previous knowledge and intuition of possible clinical classes for 85 of them. Of the predictions of these compounds using the multi-layer perceptron classifier, IRLAB considers classifications of 70.6% of the compounds to show *well-known effects*, 11.8% to show *potentially unknown effects*, and 17.6% to show *wrong effects*. In the following, some examples of classifications in the three groups are presented.

Well-known effects

The compound JNJ37822681 is classified as AP2, see Figure 6.11, which is consistent with known characteristics [1].

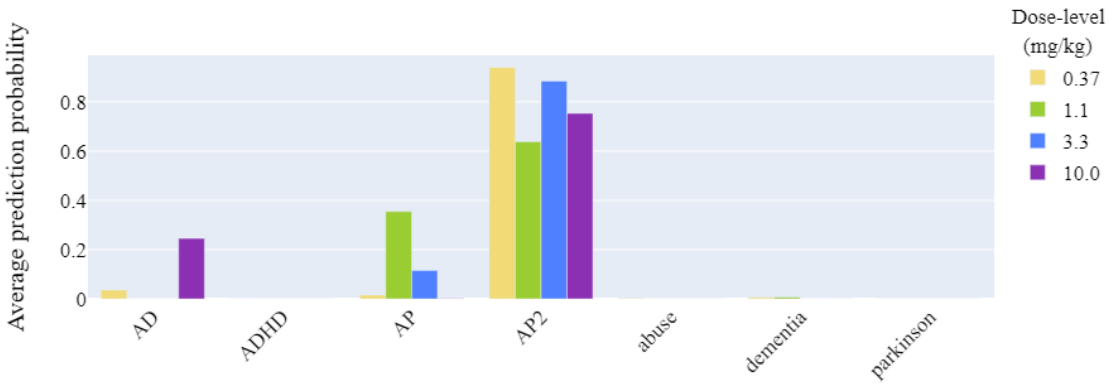


Figure 6.11: Average prediction probabilities of compound JNJ37822681, clinical tentative class AP2. Classification considered showing well-known effects.

The compound Lamotrigin is classified as AD, see Figure 6.12, which is consistent with known characteristics even though it has a completely different mechanism of action compared to multiple compounds of class AD in the training data set.

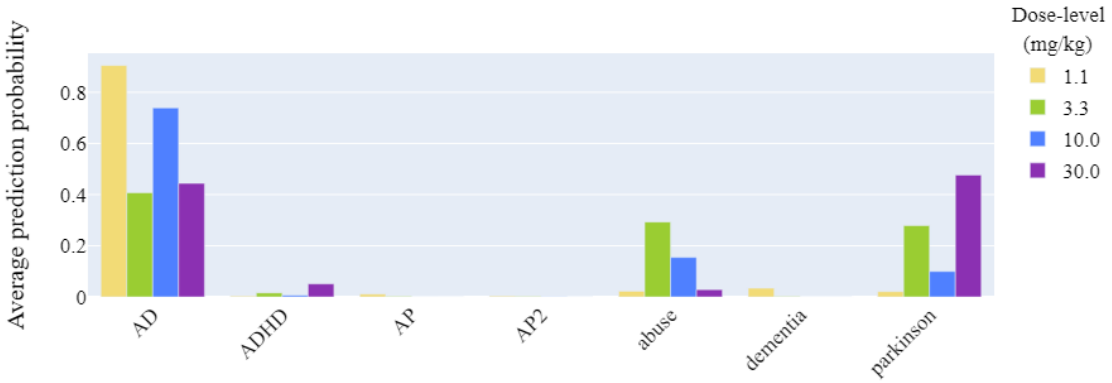
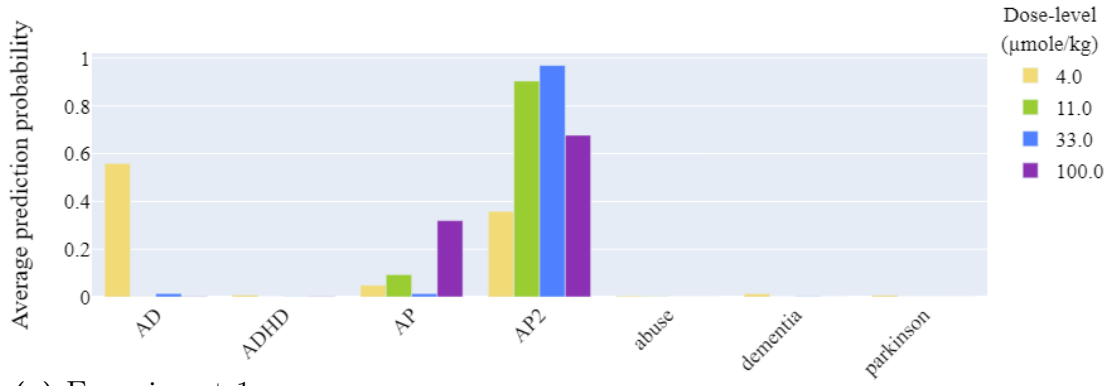
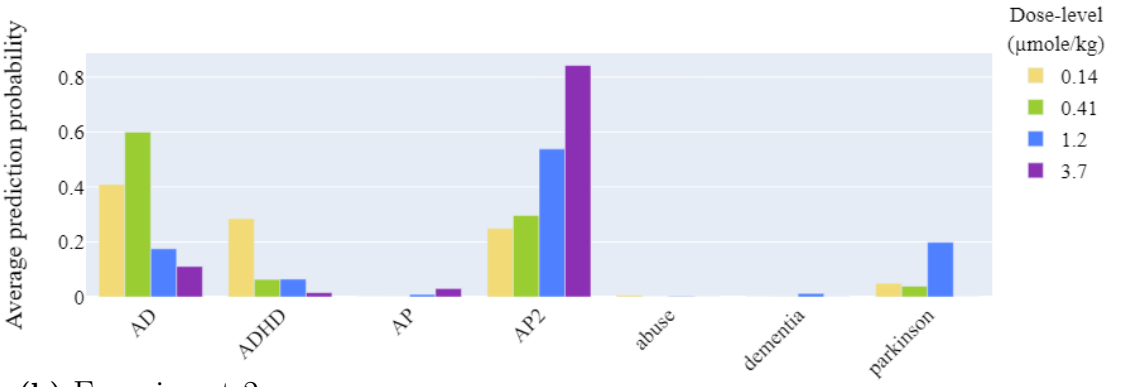


Figure 6.12: Average prediction probabilities of compound Lamotrigin, clinical tentative class epilepsia. Classification considered showing well-known effects.

In the data set for classification, there are two experiments performed for the compound Bifeprunox. The classifier results on these two can be seen in Figure 6.13 and the majority classification is AP2. For lower dose-levels, there are some experiments classified as AD as well. These results are consistent with the experience of IRLAB, which is that Bifeprunox belongs to a class of compounds with biphasic effect, i.e. having two phases, of antidepressant and antipsychotic effects.



(a) Experiment 1.

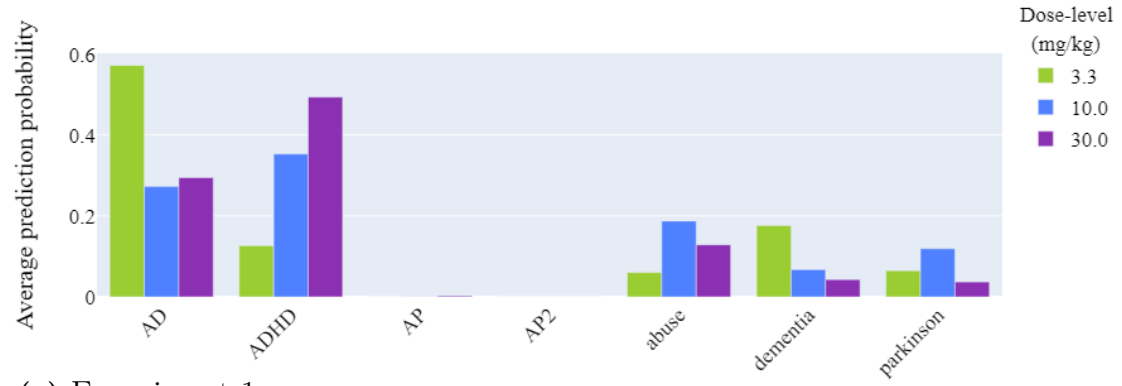


(b) Experiment 2.

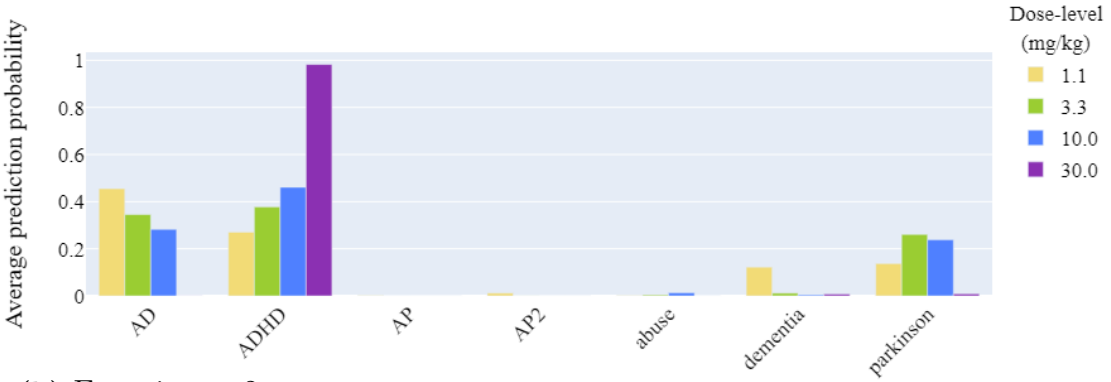
Figure 6.13: Average prediction probabilities of compound Bifeprunox, clinical tentative class AP2. Classification considered showing well-known effects.

Potentially unknown effects

The compound dov21947 is classified as an antidepressant (AD) and a compound for treating ADHD in different ratios in two different experiments, see Figure 6.14. Based on its pharmacological mode of action, it was developed as an AD compound [37], but has substantial effects on the dopamine-system. This fact makes ADHD classification an interesting and reasonable possibility since classical ADHD drugs typically increase dopamine levels.



(a) Experiment 1.



(b) Experiment 2.

Figure 6.14: Average prediction probabilities of compound dov21947, clinical tentative class AD. Classification considered showing potentially unknown effects.

The compound Fipamezole is classified as a drug for Parkinson's, see Figure 6.15, which is consistent with previous studies where Fipamezole has been suggested as a possible treatment of Parkinson's [38]. Nevertheless, Fipamezole is an entirely different kind of Parkinson's drug, compared to the Parkinson's drugs in the training data, in the way that it addresses other symptoms of Parkinson's disease. The fact that the classifier still predicts it as Parkinson's is an interesting observation.

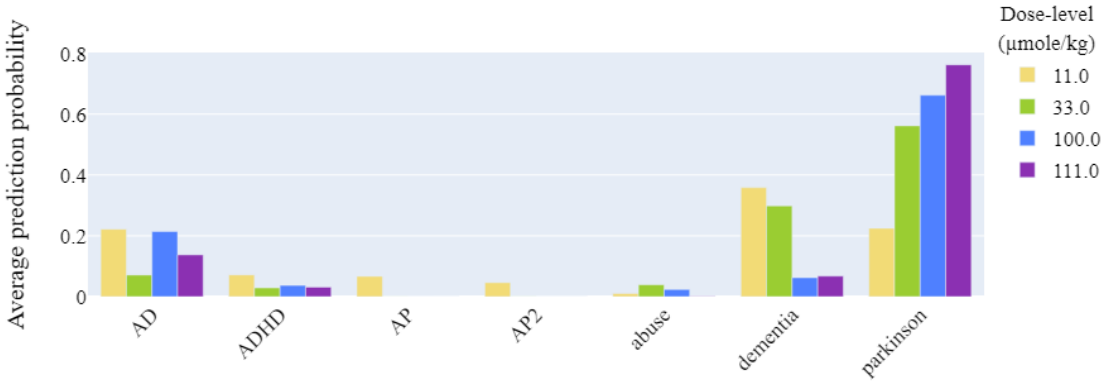


Figure 6.15: Average prediction probabilities of compound Fipamezole, no clinical tentative class. Classification considered showing potentially unknown effects.

The compound sk609 is classified as AD, see Figure 6.16. It is a dopamine receptor agonist, meaning that it mimics the actions of dopamine to relieve symptoms related to low levels of dopamine. Therefore, it is used to treat Parkinson's disease, among other conditions, which is probably the reason behind some of the observations of sk609 being classified as Parkinson. The compound sk609 is also an inhibitor of noradrenaline transporters, which is a common characteristic of AD drugs [39]. This characteristic seems to be captured by the classifier.

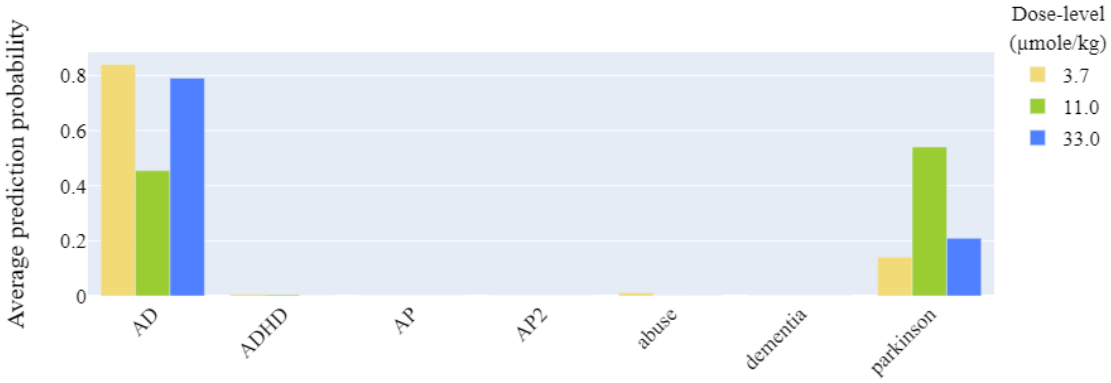
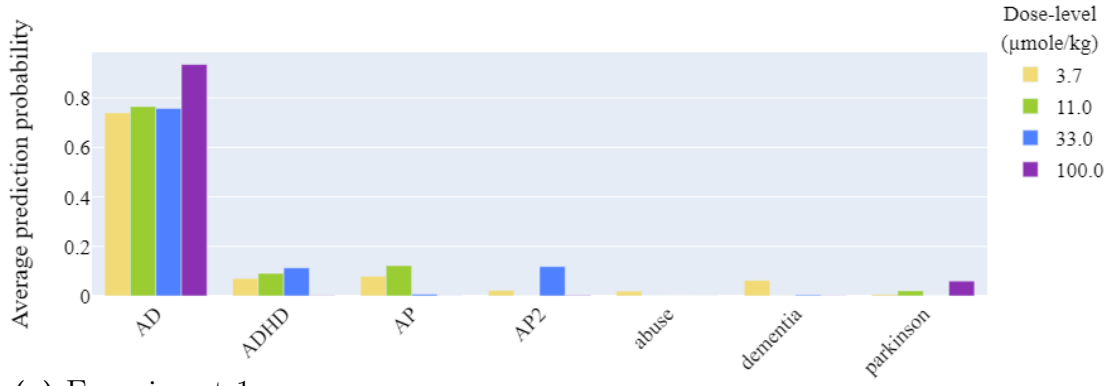


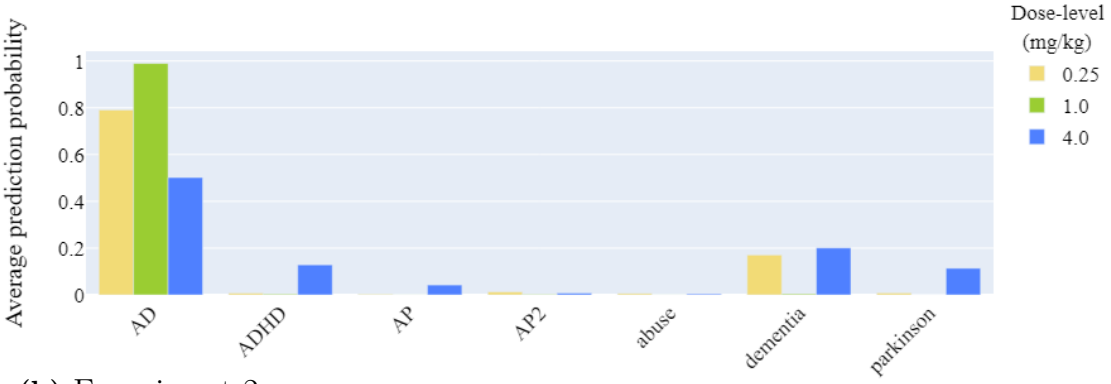
Figure 6.16: Average prediction probabilities of compound sk609, clinical tentative class parkinson. Classification considered showing potentially unknown effects.

Wrong effects

Two experiments are performed on the compound mCPP, and both are mainly predicted as AD, see Figure 6.17. The compound is often used abusively and together with other compounds. However, the effects are not typical for drugs of abuse since the compound gives mostly unpleasant effects and anxiety when taken alone. Moreover, mCPP has some mechanisms of action related to serotonin, which is a neurotransmitter often considered as a contributor to feelings of happiness or well-being. This mechanism is partly overlapping with the mechanisms of AD. Also, mCPP is a metabolite to AD compounds [40], which could explain the classification. The results of mCPP highlight an overall, more general, limitation of using classifiers. This limitation is a consequence of that classifiers can not classify better than how the classes available are defined.



(a) Experiment 1.



(b) Experiment 2.

Figure 6.17: Average prediction probabilities of compound mCPP, clinical tentative class abuse. Considered showing Wrong effects.

The experiment of the compound MK801 is predicted as atypical antipsychotic (AP2) for the highest dose-level, see Figure 6.18. However, this compound is used for modeling psychosis [41], which is basically the opposite of antipsychotics. The AP2 classification could be due to the compound having neurochemical effects similar to

antipsychotics and to the highest dose-level being very high, or even too high, so the behavioral variables are reduced. If this is the case, the antipsychotic prediction is not very surprising. The lowest dose-level has clear effects and the predicted class is AD, which is reasonable since it has a similar mechanism as norketamine which has been reported to have antidepressant effects [42].

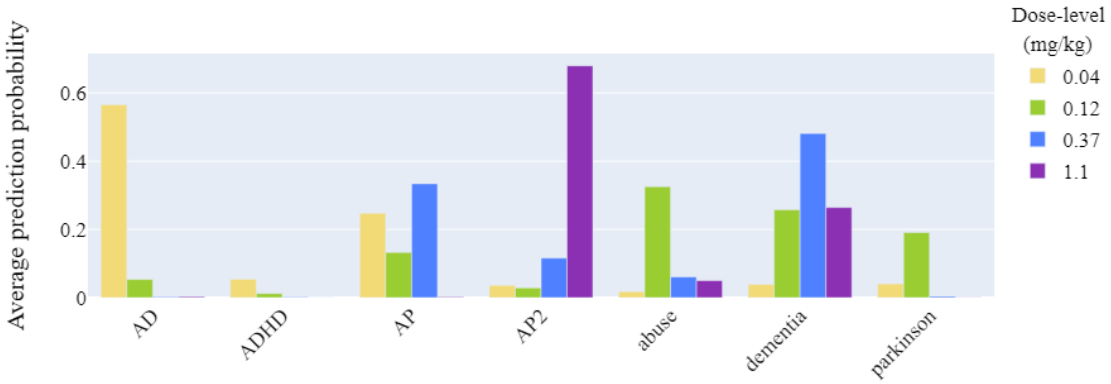


Figure 6.18: Average prediction probabilities of compound MK801, no clinical tentative class. Considered showing wrong effects.

The compound Gaboxadol is predicted as AP2, see Figure 6.19, but is not known to have antipsychotic effects. The prediction being AP2 might be due to that Gaboxadol is similar to antipsychotics in that it can reduce dopamine transmission, but with a different mechanism [43]. For lower dose-levels, Gaboxadol is tentatively known as a compound for sleep [44], which means that it is calming and makes the user drowsy. In the smallest dose-level, the average prediction probability is highest for the AD class, which could be due to a small increase in behavior. Nevertheless, the clinical effects of Gaboxadol are not verified, making it difficult to draw any conclusions.

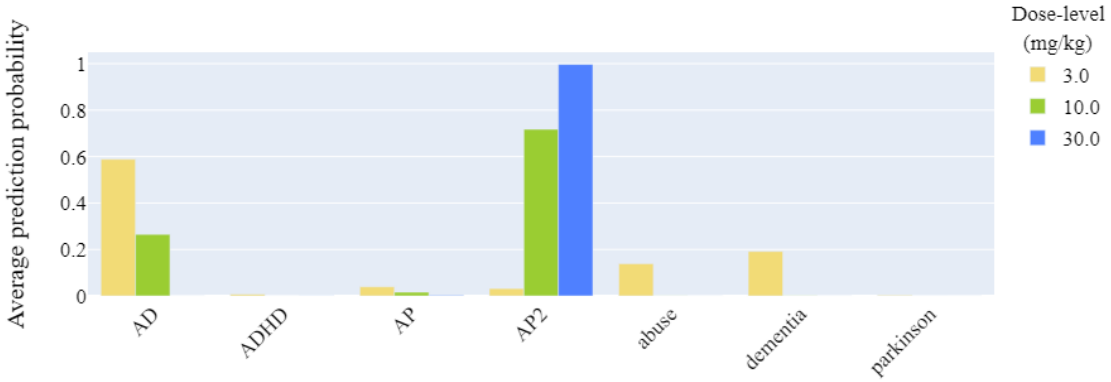


Figure 6.19: Average prediction probabilities of compound Gaboxadol, clinical tentative class sleep. Considered showing wrong effects.

7

Conclusions & Future Work

The main topic of this thesis is to investigate if there are any nonlinearities in the dose-response data as well as evaluating if different methods for dimensionality reduction and classification can capture information of these nonlinearities or provide other insights. In this chapter, the conclusions of this thesis are presented, followed by possible improvements and suggestions for future work.

7.1 Conclusions

The results from nonlinear methods for dimensionality reduction in this thesis do not show any results of very high interest. The patterns and clusters found using nonlinear methods are overall similar to the ones from the linear method. Some patterns are slightly different but do not detect any distinct new patterns, compared to the linear method. Since evaluation methods are only qualitative, it is hard to draw any general conclusions. Nevertheless, some methods show an indication of being promising for future methodology development. However, more work needs to be done in order to be able to create potential insights and value.

Results from classification indicate that there are nonlinearities in the dose-response data since nonlinear methods seem to capture the structure of the data better than the linear method and give better classification performance. The classifiers are overall promising to use for further analysis and innovation, especially the multi-layer perceptron networks. Compared to methods for dimensionality reduction, the classifier approach can reveal how different dose-levels effect dose-response, which can create additional insights. The classifications of experimental compounds using the multi-layer perceptron network show results where the predictions of 70.6% of the compounds are consistent with tentative clinical classes and previous research. Also, predictions of 11.8% of the compounds are extra interesting in the sense that they show potentially unknown effects and could be the basis of further innovation. Therefore, the classifiers can create insight and potentially high value.

7.2 Future work

As for most exploratory projects, there is more work that can be done on the subject of this thesis project. Additional methods could be explored and finely tuned to see if more interesting results can be found. Also, the methods investigated in the scope of this thesis could be investigated further by exploring different parameter settings and using different data sets. For classifications purposes, it would be of particular interest to study different network architectures of a multi-layer perceptron network since the results from this method are most promising. Moreover, the pre-processing of data before applying any methods in this thesis is only performed in one way, and other ways of doing it and of handling missing values could be a subject of further investigation.

Concerning the methods for dimensionality reduction, methods are only investigated on one data set and it would be interesting to investigate these methods on other data sets as well. For example, it could be investigated how well methods perform on more 'zoomed in' data sets, of some specific domain of compounds as well as by using only behavioral or neurochemical variables, as in [1].

Concerning future investigation of classifiers, it could be of interest to use a smaller or larger number of clinical classes to see how it affects the results. Moreover, dividing data into clinical classes could be done on a more precise level, such as dividing the Parkinson's compounds into different classes based on what kind of symptoms they treat. Another idea for future work when investigating some specific questions could be to use training data that only contains relevant data. The data used for classification in this thesis is raw, meaning it might contain dose-levels that are, in fact, irrelevant as well as poorly executed experiments. Excluding such observations could be of interest for future work. It would also be of interest to investigate classifiers on mechanical classes that are based on which target protein, such as a receptor, transporter or enzyme, is affected in the body. Further, the handling of class imbalanced data sets can be done in different ways. In the scope of this thesis, SMOTE is investigated. Other approaches to handle imbalanced data sets could be a subject for future work.

The classifiers in this thesis all have a high tendency to predict observations into the majority class, antidepressants (AD). Also, due to the nature of classifiers, each observation is forced to have one of the defined clinical classes when performing classification, while the case might be that the observation is not of any available

class. Therefore, it could be of use to add an additional *inactive* class to the group of classes used for classification. Training data of the *inactive* class could, for instance, be data from experiments performed on rats not given any compound at all, i.e. from control groups. The intuition of IRLAB suggests that some observations using compounds that should not have a very high active effect at all are classified as AD, and an additional *inactive* class might solve this problem.

A suggestion for evaluation of the robustness of classifiers as well as of methods for dimensionality reduction is to use multiple experiments of the same compound at the same dose-levels. Suppose training a classifier with a data set including at least one of the experiments with identical compound and dose-level setting, and then see if the other experiments with identical compound and dose-level setting are predicted similarly. If not, this could be a sign of a weak classifier or just a poorly performed experiment.

Observations of the same compound are of different dose-levels, but these dose-levels are not taken into account when performing classification in this thesis. However, low doses might not give as high effect as higher doses, or low and high doses might not show the same behavior at all. A suggestion for future work is, therefore, to include dose-levels in the classification methodology. One idea is to examine different dose-levels by training a classifier on higher doses and see if a lower dose is predicted in the same way. Then there might be possible to examine if lower doses could be sufficient for treatment.

References

[1] Waters, S., Svensson, P., Kullingsjö, J., Pontén, H. P., Andreasson, T., Sunesson, Y., Ljung, E., Sonesson, C., and Waters, N., "In Vivo Systems Response Profiling and Multivariate Classification of CNS Active Compounds: A Structured Tool for CNS Drug Discovery", in *Acs Chemical Neuroscience*, vol. 8, 2017, pp. 785–797.

[2] Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., Li, B., Madabhushi, A., Shah, P., Spitzer, M., and Zhao, S., "Applications of machine learning in drug discovery and development", in *Nature Reviews Drug Discovery*, vol. 18, Nature Publishing Group, 2019, pp. 463–477.

[3] Mitchell, J. B. O., "Machine learning methods in chemoinformatics", in *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 4, Blackwell Publishing Inc., 2014, pp. 468–481.

[4] Lo, Y. C., Rensi, S. E., Torng, W., and Altman, R. B., "Machine learning in chemoinformatics and drug discovery", in *Drug Discovery Today*, vol. 23, Elsevier Ltd, 2018, pp. 1538–1546.

[5] Rosipal, R. and Krämer, N., "Overview and Recent Advances in Partial Least Squares", in *Lecture Notes in Computer Science*, 2006, pp. 34–51.

[6] Eriksson, L., Byrne, T., Johansson, E., Trygg, J., and Vikström, C., *Multi- and Megavariate Data Analysis Basic Principles and Applications*. Umeå: Umetrics AB, 2013.

[7] Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. New York: Springer, 2009.

[8] Rosipal, R. and Trejo, L. J., "Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space", in *Journal of Machine Learning Research*, vol. 2, 2001, pp. 97–123.

[9] Rosipal, R., "Nonlinear partial least squares: An overview", in *Chemoinformatics and Advanced Machine Learning Perspectives: Complex Computational Methods and Collaborative Techniques*, IGI Global, 2010, pp. 169–189.

[10] Wold, H., "Causal flows with latent variables. Partings of the ways in the light of NIPALS modelling", in *European Economic Review*, vol. 5, North-Holland, 1974, pp. 67–86.

[11] Steinwart, I. and Christmann, A., *Support Vector Machines*. New York: Springer, 2008.

[12] Mercer, J., "Functions of positive and negative type, and their connection the theory of integral equations", in *Philosophical Transactions of the Royal Society of London. A 209*, 1909, pp. 415–446.

[13] Maaten, L. van der and Hinton, G., "Visualizing Data using t-SNE", in *Journal of Machine Learning Research*, vol. 1, 2008, pp. 1–48.

[14] Cao, Y. and Wang, L., "Automatic Selection of t-SNE Perplexity", 2017.

[15] Wattenberg, M., Viégas, F., and Johnson, I., "How to Use t-SNE Effectively", in *Distill*, vol. 1, Distill Working Group, 2016.

[16] Jurafsky, D. and Martin, J. H., *Speech and Language Processing*, 2nd ed. Prentice Hall, 2008.

[17] Liu, D. C. and Nocedal, J., "On the limited memory BFGS method for large scale optimization", in *Mathematical Programming*, vol. 45, 1989, pp. 503–528.

[18] Zhang, C. and Ma, Y., *Ensemble Machine Learning: Methods and Applications*. Springer-Verlag New York, 2012.

[19] Kingma, D. P. and Lei Ba, J., "Adam: A method for stochastic optimization", 2015.

[20] Ling, C. X. and Li, C., "Data Mining for Direct Marketing: Problems and Solutions", in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, New York: AAAI Press, 1998, pp. 73–79.

[21] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P., "SMOTE: Synthetic Minority Over-sampling Technique", in *Journal of Artificial Intelligence Research*, vol. 16, 2002, pp. 321–357.

[22] Wong, T.-T., "Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation", in *Pattern Recognition*, vol. 48, 2015.

[23] Goutte, C. and Gaussier, E., "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation", in *Lecture Notes in Computer Science*, vol. 3408, 2005, pp. 345–359.

[24] Opitz, J. and Burst, S., *Macro F1 and Macro F1*, 2019. [Online]. Available: http://arxiv.org/abs/1911.03347.

[25] Wallner, F., *python-nipals*, 2018. [Online]. Available: https://github.com/fredrikw/python-nipals.

[26] Wright, K., *The NIPALS algorithm*, 2017. [Online]. Available: https://cran.r-project.org/web/packages/nipals/vignettes/nipals_algorithm.html.

[27] *sklearn.manifold.TSNE*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.

[28] Tenenbaum, J. B., Silva, V. de, and Langford, J. C., "A Global Geometric Framework for Nonlinear Dimensionality Reduction", in *Science*, vol. 290, 2000, pp. 2319–2323.

[29] Roweis, S. T. and Saul, L. K., "Nonlinear dimensionality reduction by locally linear embedding", in *Science*, vol. 290, 2000, pp. 2323–2326.

[30] Borg, I. and Groenen, P., *Modern Multidimensional Scaling*, 2nd ed. Springer New York, 2005.

[31] Ng, A. Y., Ng, A. Y., Jordan, M. I., and Weiss, Y., "On Spectral Clustering: Analysis and an algorithm", in *Advances in Neural Information Processing Systems*, 2001, pp. 849–856.

[32] McInnes, L., Healy, J., and Melville, J., "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction", Feb. 2018.

[33] 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, É., "Scikit-learn: Machine Learning in Python", in *Journal of Machine Learning Research*, vol. 12, 2011, pp. 2825–2830. [Online]. Available: http://scikit-learn.sourceforge.net..

[34] *sklearn.linear_model.LogisticRegression*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[35] *sklearn.ensemble.RandomForestClassifier*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[36] *sklearn.neural_network.MLPClassifier*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[37] Skolnick, P., Popik, P., Janowsky, A., Beer, B., and Lippa, A. S., "Antidepressant-like actions of DOV 21,947: A "triple" reuptake inhibitor", in *European Journal of Pharmacology*, vol. 461, Elsevier, 2003, pp. 99–104.

[38] LeWitt, P. A., Hauser, R. A., Lu, M., Nicholas, A. P., Weiner, W., Coppard, N., Leinonen, M., and Savola, J. M., "Randomized clinical trial of fipamezole for dyskinesia in Parkinson disease (FJORD study)", in *Neurology*, vol. 79, Lippincott Williams and Wilkins, Jul. 2012, pp. 163–169.

[39] Zhou, J., "Norepinephrine transporter inhibitors and their therapeutic potential", in *Drugs of the Future*, vol. 29, NIH Public Access, Dec. 2004, pp. 1235–1244.

[40] Preskorn, S. H., Stanga, C. Y., Feighner, J. P., and Ross, R., *Antidepressants: Past, Present and Future*, 1st ed. Springer-Verlag Berlin Heidelberg, 2004, pp. 68–69.

[41] Andiné, P., Widermark, N., Axelsson, R., Nyberg, G., Olofsson, U., Mårtensson, E., and Sandberg, M., "Characterization of MK-801-Induced Behavior as a Putative Rat Model of Psychosis", in *Journal of Pharmacology and Experimental Therapeutics*, vol. 290, 1999, pp. 1393–1408.

[42] Hashimoto, K. and Yang, C., "Is (S)-norketamine an alternative antidepressant for esketamine?", in *European Archives of Psychiatry and Clinical Neuroscience*, vol. 269, Dr. Dietrich Steinkopff Verlag GmbH and Co. KG, 2019, pp. 867–868.

[43] Lopes, E. F., Roberts, B. M., Siddorn, R. E., Clements, M. A., and Cragg, S. J., "Inhibition of nigrostriatal dopamine release by striatal GABA A and GABA B receptors", in *Journal of Neuroscience*, vol. 39, Society for Neuroscience, 2019, pp. 1058–1065.

[44] Wafford, K. A. and Ebert, B., "Gaboxadol - A new awakening in sleep", in *Current Opinion in Pharmacology*, vol. 6, Elsevier Ltd, 2006, pp. 30–36.

A

Additional Results from Kernel Partial Least Squares

A.1 Radial Basis Function Kernels

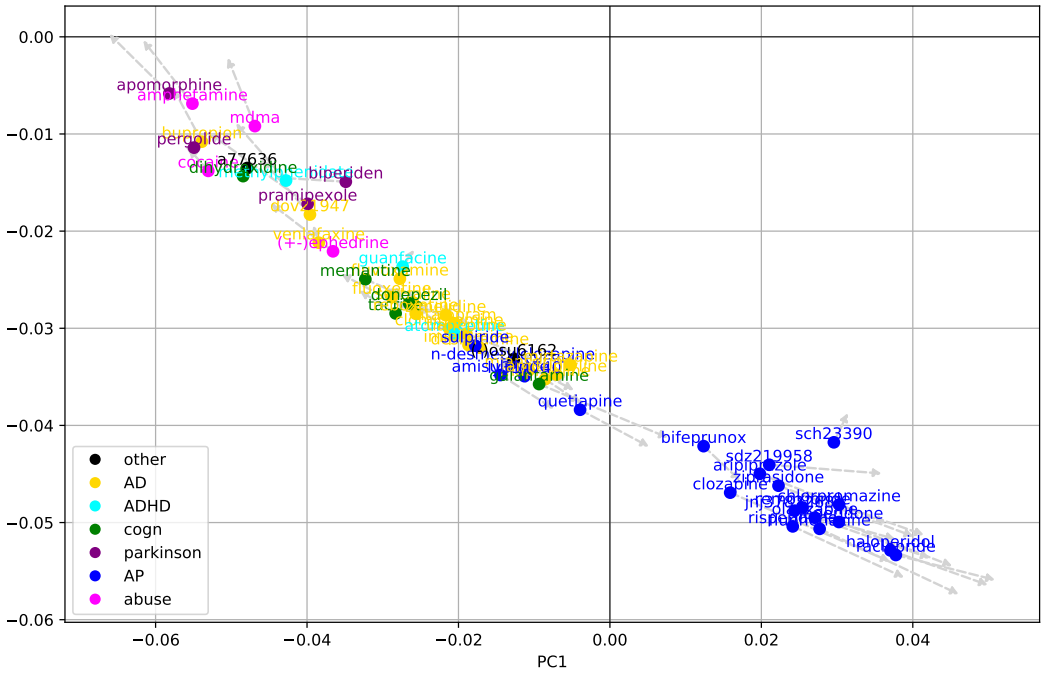


Figure A.1: Principal components of RBF Kernel PLS with width $\gamma = 0.1$. Colored dots are the average of all observations for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

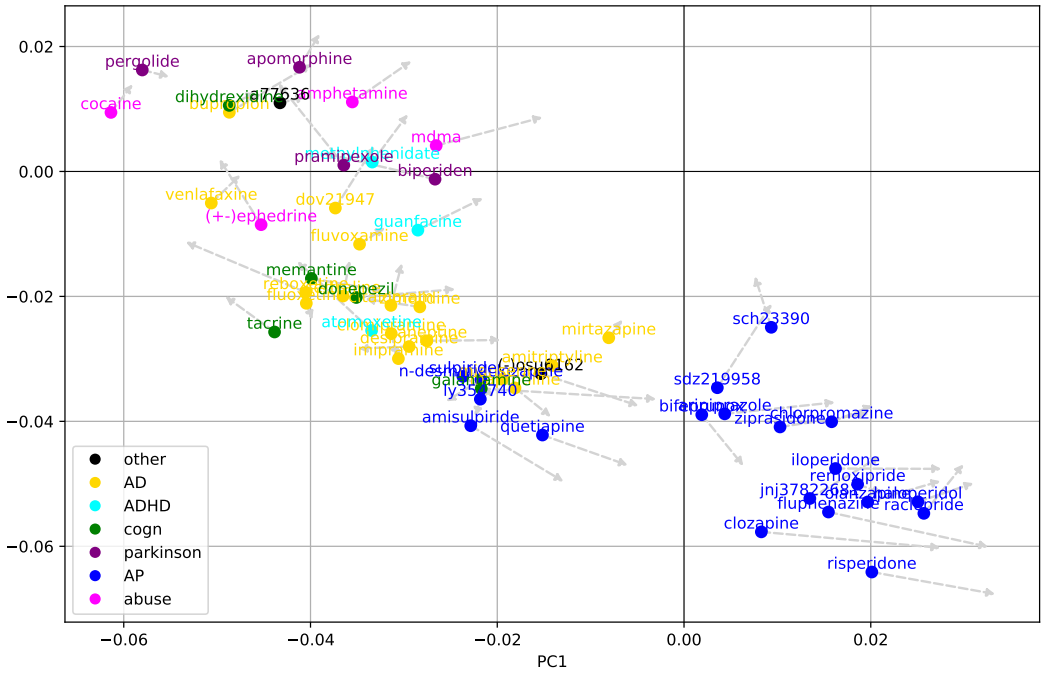


Figure A.2: Principal components of RBF Kernel PLS with width $\gamma = 0.5$. Colored dots are the average of all observations for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

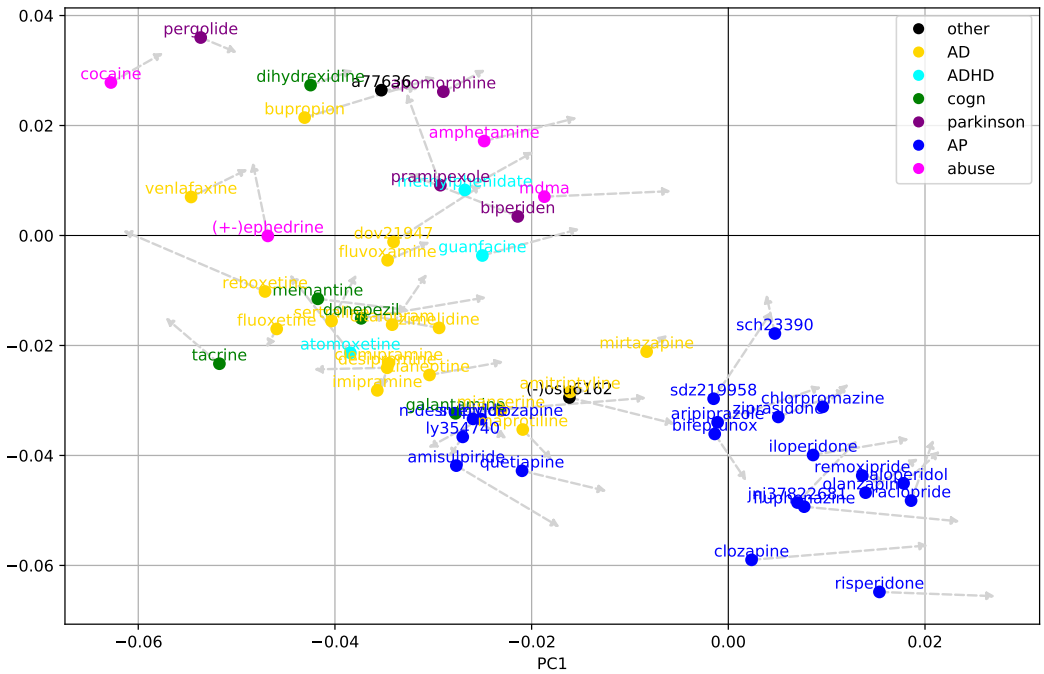


Figure A.3: Principal components of RBF Kernel PLS with width $\gamma = 0.8$. Colored dots are the average of all observations for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

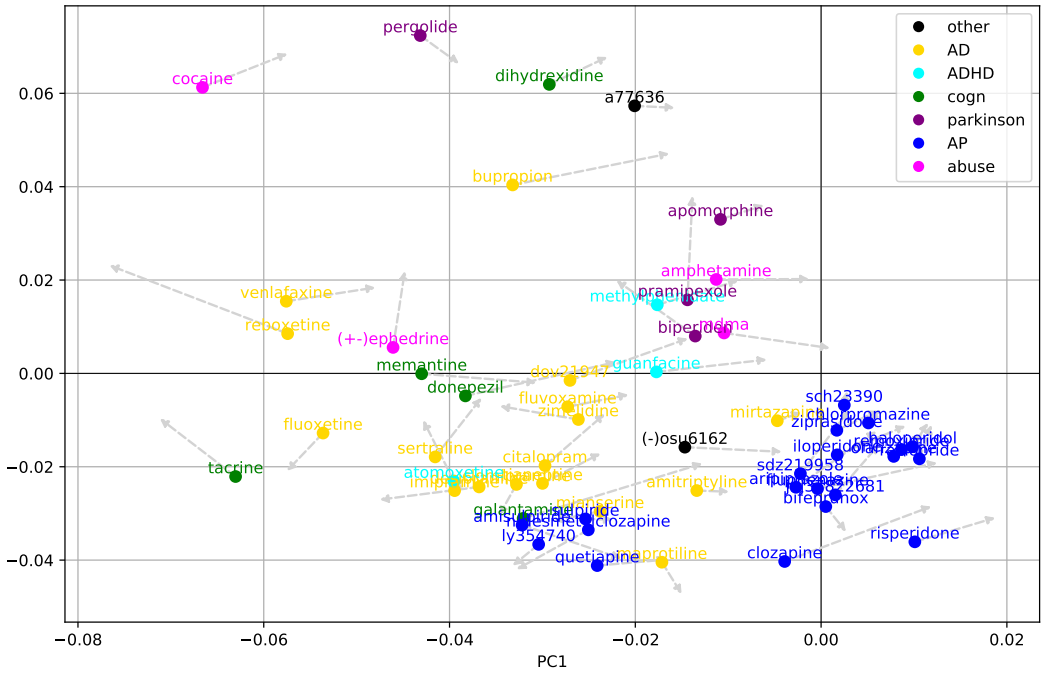


Figure A.4: Principal components of RBF Kernel PLS with width $\gamma = 1.5$. Colored dots are the average of all observations for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

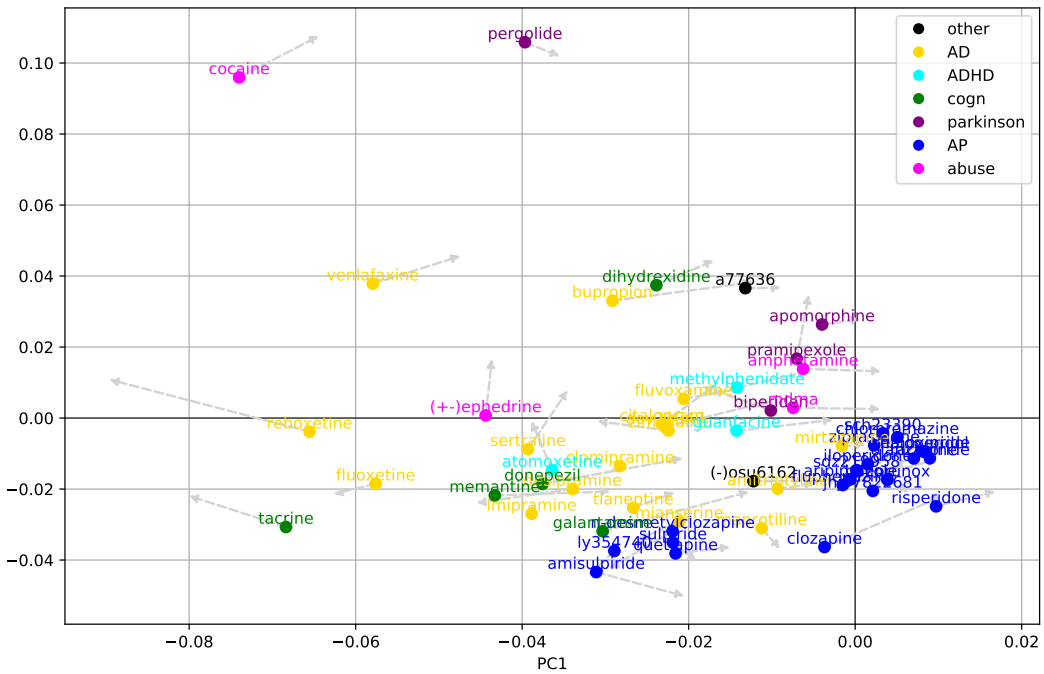


Figure A.5: Principal components of RBF Kernel PLS with width $\gamma = 2$. Colored dots are the average of all observations for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

A.2 Polynomial Kernels

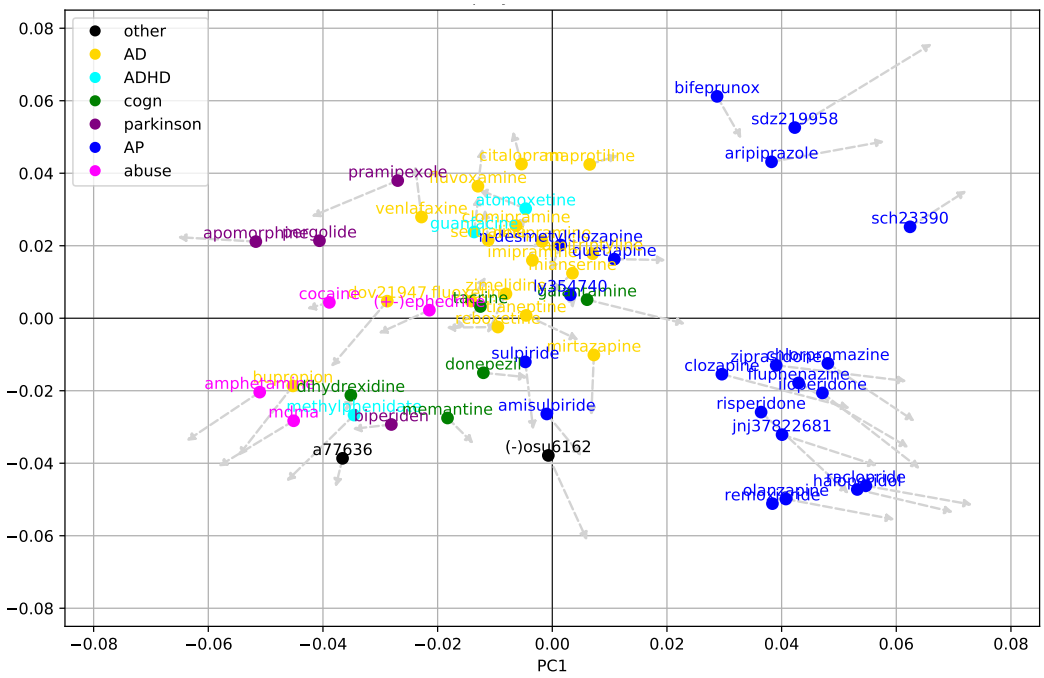


Figure A.6: Principal components of polynomial Kernel PLS, $a = 1$ and $b = 2$. Colored dots are the average of all observations for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

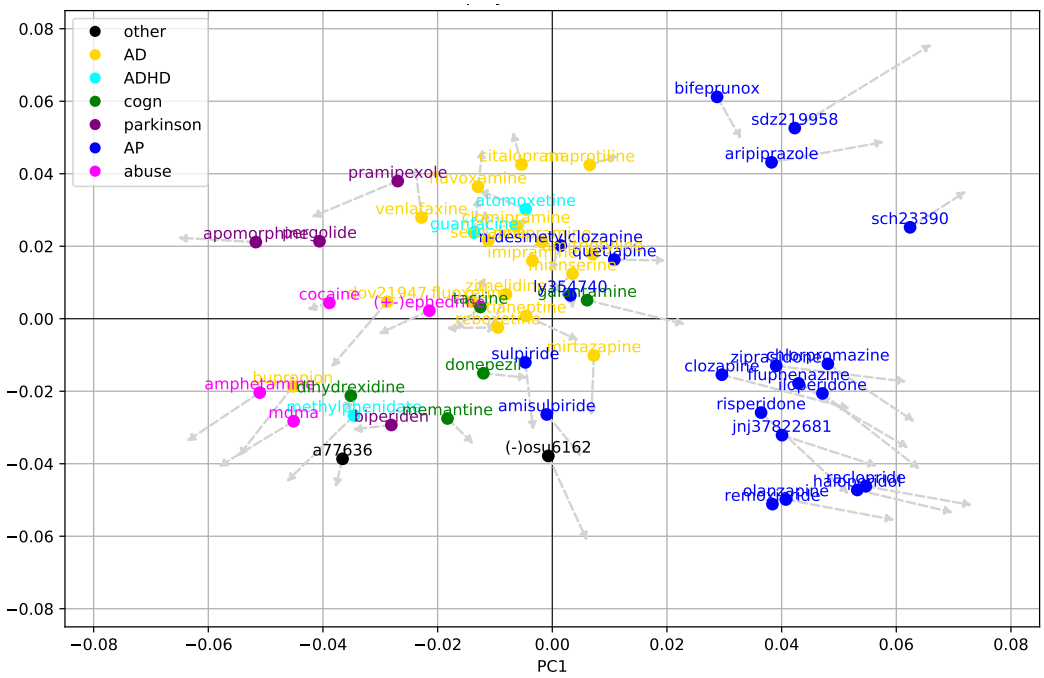


Figure A.7: Principal components of polynomial Kernel PLS, $a = 1$ and $b = 10$. Colored dots are the average of all observations for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

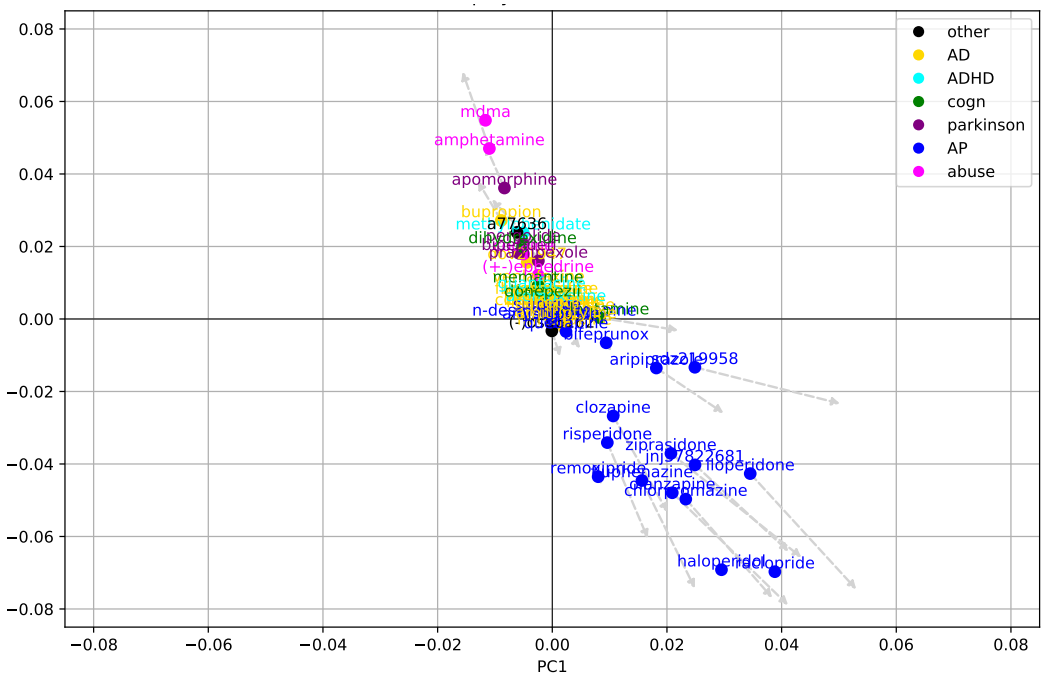


Figure A.8: Principal components of polynomial Kernel PLS, $a = 3$ and $b = 1$. Colored dots are the average of all observations for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.

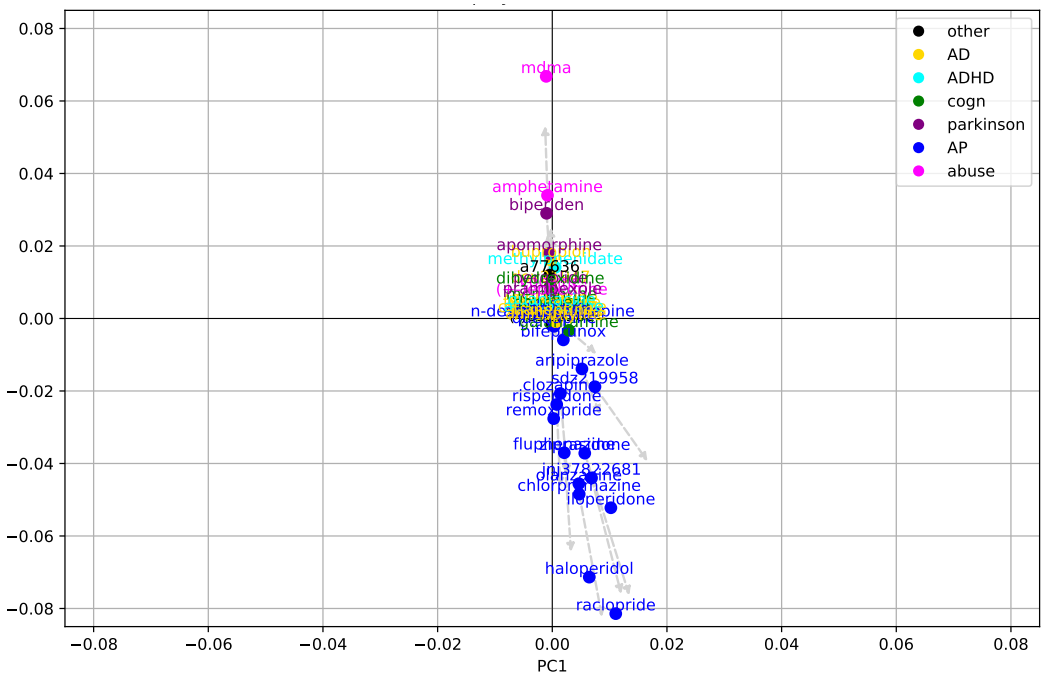


Figure A.9: Principal components of polynomial Kernel PLS, $a = 4$ and $b = 1$. Colored dots are the average of all observations for one compound. From each dot, there is a grey arrow to a weighted average where weights are the compound dose-level of an observation.