Automation of a Data Analysis Pipeline for High-Content Screening Data

Simon Bergström
Oscar Ivarsson

Master Thesis in
Computer Science and Technology

Department of Science and Technology
Linköping University
Sweden
August 31, 2015

Abstract

High-content screening is a part of the drug discovery pipeline dealing with the identification of substances that affect cells in a desired manner. Biological assays with a large set of compounds are developed and screened and the output is generated with a multidimensional structure. Data analysis is performed manually by an expert with a set of tools and this is considered to be too time consuming and unmanageable when the amount of data grows large. This thesis therefore investigates and proposes a way of automating the data analysis phase through a set of machine learning algorithms. The resulting implementation is a cloud based application that can support the user with the selection of which features that are relevant for further analysis. It also provides techniques for automated processing of the dataset and training of classification models which can be utilised for predicting sample labels. An investigation of the workflow for analysing data was conducted before this thesis. It resulted in a pipeline that maps the different tools and software to what goal they fulfil and which purpose they have for the user. This pipeline was then compared with a similar pipeline but with the implemented application included. This comparison demonstrates clear advantages in contrast to previous methodologies in that the application will provide support to work in a more automated way of performing data analysis.

Acknowledgements

We would like to thank our supervisors at Scilifelab Torbjörn Nordling and Magdalena Otrocka for all support and providing us with inspiration and ideas during the process of this thesis. We would also like to thank our supervisor Katerina Vrotsou and examinator Aida Nordman at Linköping University for great support during the completion of the thesis. All personnel within Annika Jenmalm Jensen's team at LCBKI has contributed with an inspiring working environment and have made us feel welcomed at their work, which we would like to thank them all for. Thanks also to our friend Robin Berntsson that has been a constant inspiration during our time at Linköping University.

Contents

List of Figures

Chapter 1

Introduction

This chapter introduces the purpose of this thesis by describing the considered problem, together with a proposed approach for finding a solution, and how it will make an addition to the current workflow.

At the Science of Life Laboratory (SciLifeLab), stationated at Karolinska Institutet, there is a department named LCBKI[1] which is engaged with different research projects in chemical biology. They provide expertise in fields such as assay development and high-content screening (HCS) with the goal of giving a greater understanding in human biology, and in this way enhance the biomedical and pharmaceutical research sector in Sweden.

High-content screening involves the screening of cells to collect information about their behaviour when subjected to different substances. The data collected are then initially processed using image analysis for extracting information from the images that are generated from the compounds through a screening hardware. The resulting data is then analysed further using additional data processing techniques for the purpose of reaching conclusions about the experiment.

Considering the high-content screening process performed in different projects, the image analysis is performed with advanced tools that generates a lot of data. However, the processing and analysis of the data resulting from the image analysis does not reach full potential because of the amount of data that makes it problematic to analyse in full coverage with current used software. The user performing the screens and analysis is an experienced biologist with deep knowledge in the area of high-content screening. A well-known dilemma within data analysis of biological data is the required knowledge within data mining, statistics and biology to reach full potential of the analysis. This dilemma is apparent at LCBKI and yields the purpose of this thesis.

The workflow of the data analysis performed today consist of manual calculations with the help of spreadsheets, in combination with different analysis software in order to process the data. (See Appendix A for a complete walkthrough of the current workflow). There is a lack of capacity to analyse the amount of data that HCS generates with the software that is used today, which creates the need of exploring the field of data mining in a try to improve the quantity and quality of the analysis. To be able to analyse data in full coverage, this problem will be of increasing need since the amount of data increases continuously due to the constant improvement of measuring tools. A more automated manner of selecting relevant data and enabling classification of the data will support the process of drawing conclusions from experiments, both by replacing a lot of manual work that needs to be performed today and by enhancing the analysis work through giving a second opinion based on smart algorithms.

1.1 Aim

The main purpose of this thesis is to complement and support scientific expertise in molecular biology by investigating relevant analysis methods applicable to HCS data. To this end, we propose

[1] LCBKI is the abbreviation for Laboratory for Chemical Biology at Karolinska Institutet. It is a part of CBCS (Chemical Biology Consortium Sweden), a non-profit strategic resource for academic researchers across Sweden.

a solution that implements and presents these techniques for a defined end user. The new solution will contribute with a more automated way of performing analysis that will simplify the process of drawing conclusions from experiments. It will also enhance the quality of the analysis by presenting otherwise inaccessible patterns in datasets.

1.2 Questions

The following questions will be considered within this thesis:

1. *How to create an automated pipeline to perform analysis on large amounts of multidimensional data generated from HCS?*

The main assignment of this thesis is to propose and create a solution for performing analysis of HCS data in an automated structure that can replace or complement the manual work performed today, by giving good support in the process of finding significance in biological experiments.

2. *Which techniques and methods are adequate to use for managing the large amount of data that is generated from high-content screening?*

One of the largest issues with analysis of HCS data is the characteristics and the size of the generated datasets. This needs to be considered when solving the fundamental problem of providing a solution for data analysis because everything is dependable and revolves around the data.

3. *What kind of learning algorithms are applicable for the specific problem of mining cellular data generated from HCS?*

Large and complex datasets tend to behave in ambiguous ways that cannot be explained by using simple metrics. Learning algorithms are thus used for providing classification or clustering of such data. The question relates to what kind of algorithms that are suitable for this purpose.

4. *What is the most accurate method for selecting a subset of the data that is relevant for applying a learning algorithm?*

The selection of specific features in a dataset is an indispensable stage of analysing multivariate data. The adopted method must be specifically implemented for the purpose of enhancing the data for further exploration and it must also be implemented in an efficient and robust manner.

5. *How shall the result of the data analysis be presented for the end user to provide further possibilities of understanding it?*

The end user shall be able to interpret the results received from the analysis stage and discover patterns useful with their expertise in the field of molecular biology. The solution shall thus provide abilities for further investigation.

6. *How to design a system so that the results in crucial stages can be manually curated?*

The solution provided shall only act as a support tool in the process of analysing data in the process. It must be adaptable so that the user can be aware of every action taken and have control within the important stages of the process. This is due to the requirement of biological expertise in some decision making within the analysis process.

1.3 Approach

The approach will be described according to the questions established in section 1.2.

1. This thesis will start with conducting an investigation with the aim of discovering the existing HCS analysis methods performed today, this investigation is described in Appendix A. The next step in the process includes identification of possible techniques and algorithms that can provide automatisation and extended analysis into the workflow. Finally, an evaluation shall be conducted of what is that can be improved in the current workflow and implement it. The initial phase will also consist of a literature study in the fields of feature selection and machine learning in order to identify appropriate techniques and methods associated with HCS. Some background information on HCS will also be reviewed for a better understanding of the subject.

2. The proposed solution for the specified problem is a cloud based software that is available for authorised users. The application shall include features for input and output of data such that it can be integrated as a part of the current workflow. The data uploading phase requires a well developed data management system to be able to handle the amount of data that is generated from HCS. This requires a scalable system where operations can be performed on large datasets. The input can also appear in odd formats which creates a requirement of adaptable parsing options.

3 and 4. For the purpose of conducting data analysis, multiple different algorithms will be investigated and implemented in order to be able to perform a comparison. Feature selection techniques will be assessed due to the multidimensional nature of HCS data, such that a dataset can be filtered to only include relevant features.

5. The initial investigation of the workflow also shall consist of looking into which softwares and techniques that are used by the end user for visualising the resulting data. The visualisation methods that are not possible in current workflow but would provide value for the end user shall be implemented. To enable visualisation with other softwares, export functionality for the result from data analysis will be implemented

6. To be able to create a useful application suited for a specific end user that possess expert knowledge in another domain, a close collaboration with the supposed user must be set up so that continious feedback can be given together with multiple user studies. A third-party supervisor shall also be consulted with knowledge spanning over both the field of molecular biology and computer science, such that the communication will be simplified.

1.3.1 The End User

The application will be customised according to a specific end user. This end user will be in house during the development and all functionality and design decisions will be influenced by this end user. The user is a well educated scientist within the field of cell biology and specialised in the field of high-content screening. The user has also knowledge within math and statistics but has no experience from using data mining within the research. The computer skills of the user are at a basic level, i.e. experience exists in specific computer software.

The user is familiar with software like Excel [1] for performing manual mathematical operations to analyse generated data. To visualise results for further analysis the user has great experience in the software Spotfire [2]. The user has tried working with data analysis software incorporating data mining algorithms but due to the long learning period to use this software, and requirement of data mining knowledge, these software never became of good usage for the user.

1.3.2 Limitations

This thesis is restricted to only include a few specific data mining algorithms, which are selected through a pre-study phase. The number of algorithms included is greater than one because of the purpose of providing alternative algorithms when performing analysis. However, no comprehensive analysis of different feature selection or classification techniques will be performed.

1.4 Thesis Overview

The remaining parts of this thesis are structured as follows.

Chapter 2 will mainly present the theoretical background upon which this thesis is based on. It basically covers the fields of HCS, data anlaysis and data management.

Chapter 3 covers how the implementation has been performed in this thesis to solve the fundamental problem and how the methods in the theory chapter have been utilised.

Chapter 4 presents the resulting application and how it performs on different kinds of data. This chapter also describes how the new automated pipeline for conducting data analysis in HCS differs

from the procedure used before.

Chapter 5 concludes the work of this thesis. It starts by first summarizing the major thesis contributions. I then includes directions for future work and ends with some concluding remarks about the performed work.

Chapter 2

Theory

This chapter includes all theory that is necessary for understanding the concept of this thesis. It covers basic knowledge of the screening methods that are used in projects within biological research and why it is a suitable field for adapting various data mining techniques to. An extensive review of the data analysis methods is also covered together with some background of the database management system used.

2.1 High-Content Screening

This section relates to the overall description of which biological context this thesis is performed in and what part of the research pipeline that will take advantage of the resulting outcome.

High-content screening (HCS), also denoted as high-content analysis (HCA), can be defined as a general name for a series of automated analytical methods used for biological research about cells and their behaviour in different environments. HCS is an automated platform for conducting microscopy and image analysis in the purpose of study the behavior (phenotype) of cells subjected by different substances [3]. HCS is generating data in large amounts due to the existing technology and software that provides features down to cellular level. HCS became an official technology in the mid 90s for the purpose of dealing with complex biological systems within screening and to bridge the gap between depth and throughput of biological experiments [4].

The basic concept of the screening process is that the cells are exposed to different compounds and to be able to see what happens, automated digital microscopy is performed which outputs flourescent images of cells. By utilising an automated HCS pipeline, a quantitative and qualitative analysis can be made of the outcome. HCS branches out from microscopy and the terminology was first coined in the 90s by Giuliano et al. [5]. Its predecessor High-Throughput Screening (HTS) resulted in a single read out of activity while HCS allowed measurement of multiple features per cell simultaneously. This possibility made the readouts more challenging in terms of complexity but also enabled a more effective tool for discovering new applications [6].

The research of HCS can cover multiple fields, e.g. drug discovery that can be described as a type of phenotypic screen conducted in cells. It includes analyse methods that yields simultaneous readouts of multiple parameters considering cells or compound of cells. The screening part in this process is an early discovering stage in a sequence of multiple steps that are required for finding new medications. It acts as a filter for targeting possible candidates that can be used for further development. The substances used for this purpose can be small molecules, which can be defined as an organic compound with low molecular weight, e.g. proteins, peptides or antibodies.

2.1.1 Phenotypes

When performing HCS, the target is to evaluate the phenotypes of cells when they have been affected with some sort of substance. A phenotype can be described as observable characteristics of an organism, determined by its genetic background and environmental history [7]. It can be defined on multiple different levels starting from a whole organism down to a cellular level.

2.1.2 Methods and Pipeline

HCS can be considered to be a comprehensive system for addressing biological problems and therefore many different fields of expertise are needed as proposed in [8]. Six major skill sets can be charted for the requirement of developing and running a HCS project and even though a single person can have knowledge in several fields, it is rare to have fully extensive expertise in all of them. First of all, for the ability of developing a hypothesis based on a biological problem, there needs to be an understanding of the biological background. This comprises knowledge of current methods for affecting cell behaviour as well as being able to find opportunities for exploring and discovering new ones. Two other areas where knowledge is required are microscopy and instrumentation. It is important to have good understanding of fundamental microscopy for using correct techniques so that the screenings are performed with good quality. The resulting data is also affected by the instruments used, which thus requires solid knowledge of what types of instruments to use for specific experiments. This knowledge is also important to be able to handle instrument problems, automation of the screening process or image acquisition configuration.

Image analysis is another large and important part of HCS experiments used for detecting and measuring changes in the cells. Through different algorithms suitable for specific pattern recognition, one can detect and extract information from the images. Most of the time, these methods are applied through third-party applications. With the data extracted from the images, there are requirements for utilising fields of information technology support and statistical analysis. The task of the IT expert is to find a suitable data management solution that is scalable due to the amount of data generated from experiments while the part of statistical analysis can be defined as the concluding step in the process of a HCS project. The person responsible for the analysis should understand the concept of the experiment and apply the required statistical tests to be able to draw conclusions. The difficulties of data analysis for HCS projects can vary a lot depending on the experiment outcome and the methods applied. The robustness of a screen is often relatively easy to evaluate through positive and negative controls where the response is known. Positive control relates to when a compound is setup such that it ensures effect while negative control is the opposite, it ensures that no effect is going to occur. Also cell culture performance visualised through heat maps can help to locate problematic patterns in different plates and z-scores can be calculated for each data point for identifying extreme values. The amount of generated data can however be of such amount that it becomes a hard task for extensive manual analysis. Data on a cellular level generates millions of data points per image and several hundreds of features can be extracted per data point. Therefore learning algorithms can be applied for selecting and classifying data to additionally help an analysis expert in the work of making correct conclusions.

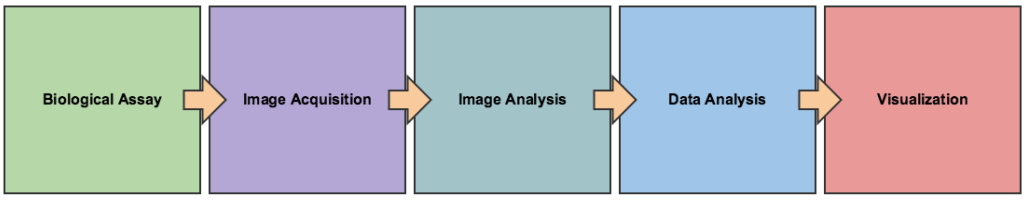


Figure 2.1: The pipeline of a High-Content Screening workflow.

A pipeline of the workflow for performing HCS can be viewed in fig. 2.1. A biological assay is a type of biological experiement that can be defined as setting up and developing the actual environment for examining the activity of an organism that has been exposed by a substance, e.g. hormone or drug. This assay is developed and screened into high resolution images. The images are processed and analysed for the purpose of finding cell features and characteristics. The resulting data is then extracted and can thus be used for further data analysis. What kind of data analysis that should be performed and why differs depending on the purpose of the experiment. For example samples can be predicted into classes that relates to positive and negative control. The output can then be visualised through mapping data to different graphical representations.

2.1.3 Data Characteristics

The data extracted from the image analysis stage can contain millions of data points due to the inclusion of data on a cellular level. The data is also of multidimensional type in that it can contain several hundreds of features per data point. The desired features can be chosen when the data is extracted during the image analysis. From the image analysis software, the data can be exported in different formats.

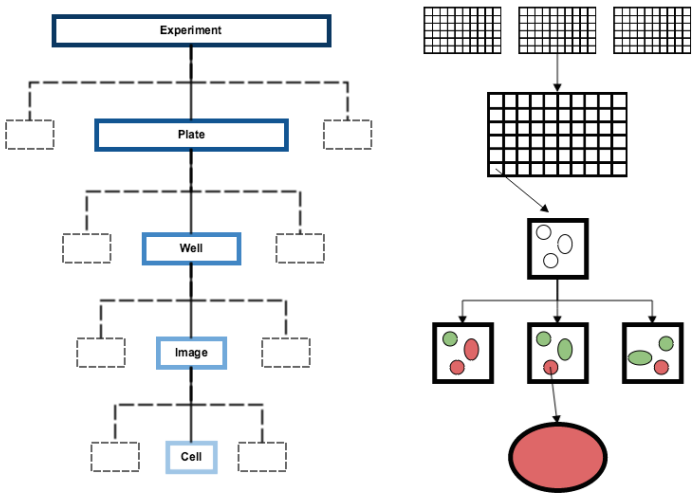


Figure 2.2: The different levels that data can be extracted from the image analysis.

The data is distributed over several different levels, which can be seen in fig. 2.2. A dataset is most of the time extracted as a specific experiment that has been performed. An experiment can contain multiple different plates with substances. The plates have a defined plate map of different wells where data can be extracted as multiple images. The data points for specific features are then stored at a cellular level.

2.2 Data Analysis

This section describes the concept of data analysis and for what purpose it will serve in this thesis.

Data analysis is the process of evaluating data using analytical and logical reasoning, the process varies depending on application area. Content within this thesis will cover the areas of data mining, feature selection and visualisation. The area of data mining includes areas like machine learning and artificial intelligence but for simplicity we will refer to data mining in this thesis since investigating the differences and similarities of these areas are not in focus. Data mining also incorporates the subject of feature selection but since this field is are crucial in this thesis, the following section will explain feature selection separately.

2.2.1 Data Mining

Data mining can be defined as: *"a set of mechanisms and techniques, realised in software, to extract hidden information from data"* [9]. Data mining is performed by a computer with a specific goal within exploration of data that is set by a user, where the data often is too complex or large for manual analysis. The subject of mining large dataset for the purpose of discovering patterns and make predictions is of increasing significance in multiple different fields, including biological data. Data mining has its roots in the late 80s within the research community and could be defined as a set of techniques for the purpose of extract hidden informations from data [9]. The interest of data mining is increasing due to the increasing amount of data produced that complicates manual interpretation and analysis.

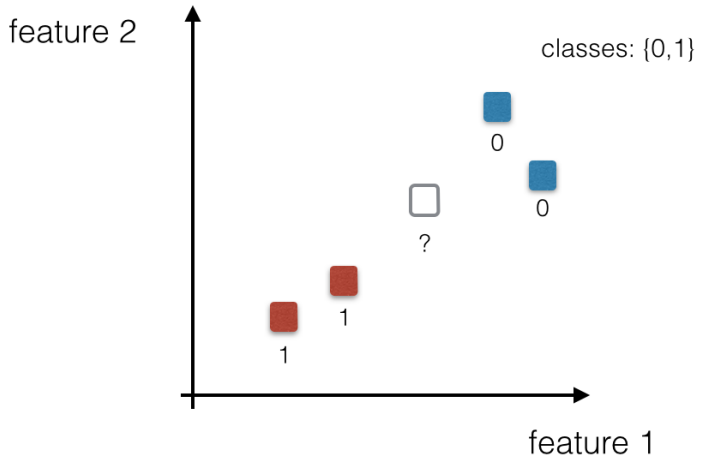


Figure 2.3: Illustration of classification in a supervised learning context. A classifier is trained based on based on the four samples with known class, denoted 0 (blue) and 1 (red) and used predict the class of the fifth sample of unknown class.

The initial application of data mining was focused on tabular data but was developed into different fields like text mining, image mining and graph mining. Different techniques within data mining can be categorised in the following three categories: pattern extraction/identification, data clustering and classification/categorisation. The aim of pattern extraction is to find patterns within data, which has been an essential focus within data mining throughout its history. Clustering aims to group data into categories with similar implicit characteristics. Unlike clustering, the classification techniques categorise data into groups/classes that are predefined, see fig. 2.3.

Modelling the relationship between a set of input variables (regressors) and another set of output variables (regressands) for the purpose of predicting output variables is often a complex process to achieve mathematically. Data mining provides techniques to solve these issues in an approximate manner which can be used for classification and regression problems.

2.2.2 Data Model

A common way of describing a model in statistics is to find the relationship between the regressors, which are the independent variables, and the dependent variable called regressand. This is explained by

$$\phi_j = \check{\phi}_j + \upsilon_j, \;\; \xi = \check{\xi} + \epsilon \tag{2.1}$$

which describes the definition of the j^{th} regressor $\check{\phi}$ and the regressand $\check{\xi}$ with errors, defined by υ_j and ϵ. All following data mining methods for classification and regression problems aims to model this relationship by solving

$$\sum_{j \in \mathcal{V}} \check{\phi}_j \check{\theta}_j = \check{\xi} \tag{2.2}$$

which specifies the sum of the regressors $\check{\phi}_j$ for all j, multiplied with a parameter $\check{\theta}_j$ that shall result in the regressand $\check{\xi}$. The purpose of data modeling is to find out how the parameter shall be constructed.

2.3 Supervised Learning Algorithms

Supervised learning can be utilised for generating profiles for each tested substance in a HCS experiment and create models made for classifying samples according to these profiles. This section covers theory and explanation of the different supervised learning methodologies that are used in this thesis.

Supervised learning is a concept in machine learning where a model is to be created from a set of data where the response is known. New data without known response can then be applied to the model and the outcome will be predicted responses. Supervised learning can be divided into two major fields: classification and regression. Classification problems apply to data that is categorised into nominal values while regression problems apply to real values. This thesis will only cover supervised learning with classification algorithms.

2.3.1 Decision Trees

Decision trees can be applied to both regression and classification problems and is a supervised learning algorithm where a tree is created for representing a decision model. To build a tree, training data is used to recursively split the data into branches. Thresholds are applied to split the tree at so called nodes.

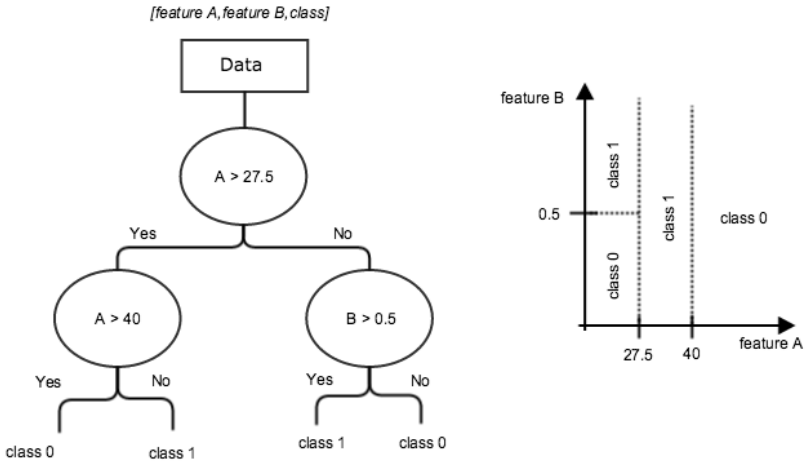


Figure 2.4: Illustration of a decision tree (left) and the corresponding regions in the feature space (right).

A threshold is a value from a feature in the training data that can easily be described as an "if-statement", check example of a decision tree and how the splitting could be done in fig. 2.4. The split to use on each node can be decided with different algorithms, some of the most common are cross entropy or gini index which are further explained in a subsection below. The tree is recursively constructed until a stopping criteria is fulfilled. The class of each leaf (where tree stops) is decided by the distribution of observations from the dataset of the specific classes that ended up on that leaf. The class with the majority of observations set the class of the leaf. When the tree is created it can be used for predicting data by letting the data traverse through the tree to get a value or get classified depending if it is a classification or regression problem. Decision trees as an algorithm in itself often produces bad results with models that overfit the data, but in other approaches like Random Forest which is an improved version of decision trees the resulting model gives much better result and two of these algorithms are described in this section.

2.3.2 Random Forest

Decision trees is a popular method for performing decision analysis within machine learning. There are however some constraints of only utilising a single decision tree, there is for example a high risk of overfitting and they are seldom very accurate in their analysis. Random forest is an ensemble learning method which makes use of multiple decision trees in its computations. It can be used as both unsupervised- and supervised learning method and could be applied to both regression and classification problems [10].

The random forest algorithm uses a large collection of decorrelated decision trees and takes an average value of the decision trees to predict and create the resulting models. This approach is derived from bagging which calculates the average values of different models. Bagging leads to

lower variance of the resulting model which results in a procedure that is less sensitive to noise. Random forest provides an improvement over the original bagging approach which reduces the correlation between the decision trees [11][12].

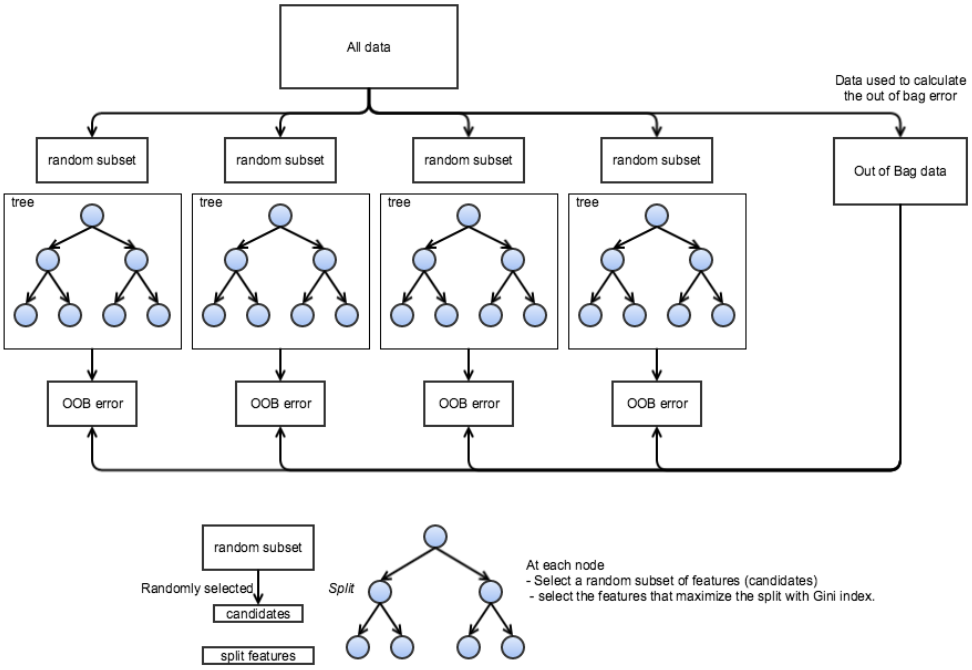


Figure 2.5: The random forest procedure.

As in bagging, the algorithm starts with building a number of decision trees of bootstrapped[1] training data. An example is given by

$$\begin{pmatrix} f_{A1} & f_{B1} & f_{C1} & f_{D1} & C_1 \\ f_{A2} & f_{B2} & f_{C2} & f_{D2} & C_2 \\ ... & ... & ... & ... & ... \\ f_{AN} & f_{BN} & f_{CN} & f_{AN} & C_N \end{pmatrix} \tag{2.3}$$

with f corresponding to samples of the $A-D$ features and C representing which class the samples belongs to. The equations

$$S_1 = \begin{pmatrix} f_{A1} & f_{B1} & f_{C1} & f_{D1} & C_1 \\ f_{A16} & f_{B16} & f_{C16} & f_{D16} & C_{16} \\ ... & ... & ... & ... & ... \\ f_{A22} & f_{B22} & f_{C22} & f_{D22} & C_{22} \end{pmatrix}, \; S_2 = \begin{pmatrix} f_{A3} & f_{B3} & f_{C3} & f_{D3} & C_3 \\ f_{A12} & f_{B12} & f_{C12} & f_{D12} & C_{12} \\ ... & ... & ... & ... & ... \\ f_{A27} & f_{B27} & f_{C27} & f_{D27} & C_{27} \end{pmatrix} \tag{2.4}$$

shows two randomised subsets of the example data that could be used for creating decision trees.

In the bagging algorithm an error estimation can be computed that is called out-of-bag (OOB) error. Approximately $\frac{2}{3}$ of the data in a learning tree are used and the residual $\frac{1}{3}$ is referred to as the out-of-bag observations. A prediction in each of the trees could be conducted with the data from OOB on each of the trees to calculate an error.

The random forest procedure is visualised in fig. 2.5 where the result is computed as the average of results from multiple decision trees. The figure also illustrate the process from the dataset where random subsets of data is created and bootstrapped from the dataset and decision trees for each subset is created. At last the splitting process for each tree is described and how the OOB data together with the generated decision trees generate an OOB error for each tree. When the splitting

[1]Original data is replaced with other data from the dataset which could result in repeated and omitted values.

occurs at each node in the decision trees, a random subset of features is selected as candidates. The optimal feature value within a specific feature from the subset is then selected for the split and this randomized procedure will decrease the correlation of the trees. The number of candidates m is usually calculated with $m = \sqrt{p}$ where p is the total number of features in the subset [10].

Another way to calculate the error within decision trees is to calculate the Gini index, which measures variance across the classes and can be used to measure the quality of a particular split in a decision tree. The Gini index can also be used to measure variable importances. This is made by adding up the total amount the Gini index is decreased for every split in a tree and then computing the average over all trees. The importance will be a coefficient between $0 - 1$ and can be further used in a feature selection. The Gini index can be referred to as an impurity measure in this field of usage and could be exchanged to other measures e.g. cross entropy [13][11]. More information about cross entropy and gini index can be found in section 2.5.2.

2.3.3 Extremely Randomized Trees

Extremely Randomized Trees (ERT) is an extension of Random Forest which is using bagging and randomized subsets for each tree but it modifies the splitting process. The selection of splitting-feature in Random Forest is determined based on the most optimal value of the candidates for splitting and then the most optimal feature according to a metric, like gini index, decides which feature to choose for the splitting. In ERT each candidate for splitting receives a random value from their observations which then are used for selecting the best splitting candidate. This procedure often results in a model with a reduced variance but with a slight increase in bias [14].

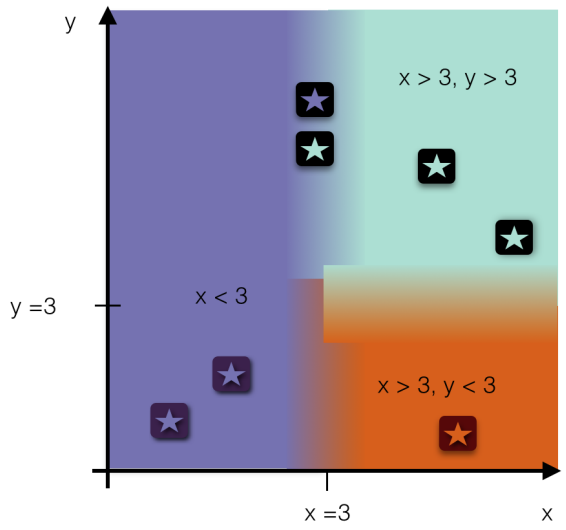


Figure 2.6: Bagged classification example for Random Forest or Extremely Randomized Trees.

Since Random Forest and Extremely Randomized Trees are both bagged classifiers which take a mean value from multiple decision trees the boundaries for a specific class is fuzzy. This is visualised in fig. 2.6 with the transitions between colours representing the fuzzy boundaries between classes. The colours in the figure represents three different classes and how the data samples (stars) are classified for the features x and y. The classification of samples within the fuzzy, areas is based on the mean value of multiple different decision trees which result in that samples closely located will not obviously correspond to the same class, It will differ for every case.The rules that are set up by a single decision tree could easily be translated as “if-statements” in programming with different boundaries as attributes.

2.3.4 Support Vector Classifier

Support vector classifier (SVC) is a supervised learning algorithm for classification and is a generalized method of the maximal margin classifier [11]. The approach of SVC is to produce a hyperplane which will separate samples in a dataset according to how the samples are delimited

by the hyperplane.

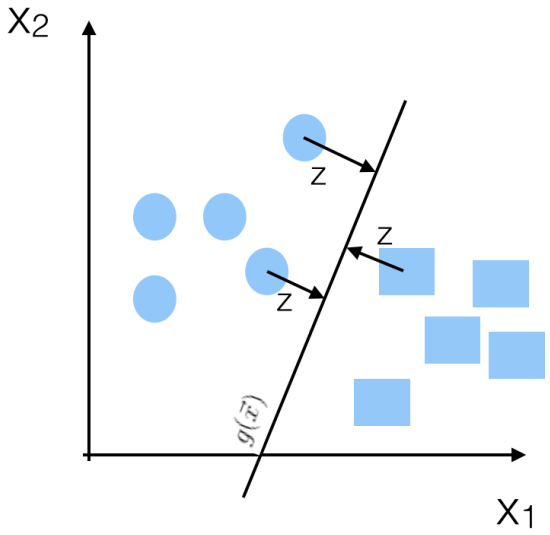


Figure 2.7: An example hyperplane $g(\vec{x})$ of a maximal margin classifier.

The hyperplane of a maximal margin classifier will be constructed to maximise margin between hyperplane and the closest observations. The closest observation will afect the hyperplane and will act as support vectors for the hyperplane, see fig. 2.7. SVC is called the soft margin classifier since the margin from the hyperplane allows violation of some of the training observations to be on the wrong side of the hyperplane or just violating the margin. This property increases the robustness of the classifier and makes it more general since the data rarely is optimal for finding a linear hyperplane.

The distance z is calculated by

$$z = \frac{|g(\vec{x})|}{\|\vec{w}\|} = \frac{1}{\|\vec{w}\|},\ g(\vec{x}) \geq 1\ \forall \vec{x} \in class1,\ g(\vec{x}) \leq -1\ \forall \vec{x} \in class2 \tag{2.5}$$

where weight w is the so called support vectors and will span up the hyperplane $g(\vec{x})$ for classification. Observations with values above 1 will belong to $class1$ and observations with values below -1 shall belong to $class2$.

Process of binary classification
Given a set of training data x with predefined classes y gives an optimization problem for minimising the weight vector to maximize distance between the closest samples of the two classes. This optimization problem is given by

$$maximize_{\beta_0,\beta_1,\beta_2,\ldots,\beta_n,\epsilon_0,\ldots,\epsilon_n} M \tag{2.6}$$

$$\sum_{j=1}^{p} \beta_j^2 = 1 \tag{2.7}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \tag{2.8}$$

where parameter β represents a weight coefficient for the different features in the training data x and M relates to the margin that one wants to maximize. Observations that get a value between -1 and 1 in eq. 2.5 will be problematic for a maximal margin classifier since those observations lie within the calculated margin or on the wrong side of the margin or hyperplane and no perfect separating hyperplane exists. This is considered by the soft margin classifier with help of slack variables ϵ which enable the soft margin to accept observations to be on the wrong side of the

margin and hyperplane. If $\epsilon_i = 0$ the observation is on the right side of the margin. If ϵ is between 0 and 1, that means that the observation has violated the margin but is on the right side of the hyperplane. Finally $\epsilon > 1$ means that the observation is on the wrong side of the hyperplane. Parameter C in

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C \tag{2.9}$$

is a tuning parameter of how tolerant the classifier will be to observations that violate the margin or are on the wrong side of the hyperplane. A high value of C allows many observations to violate the margin and potentially result in a more biased classifier but with lower variance. A low C value restricts the violation of observations on the wrong side of the margin and potentially results in a classifier that highly fits the data with low bias but is having a high variance.

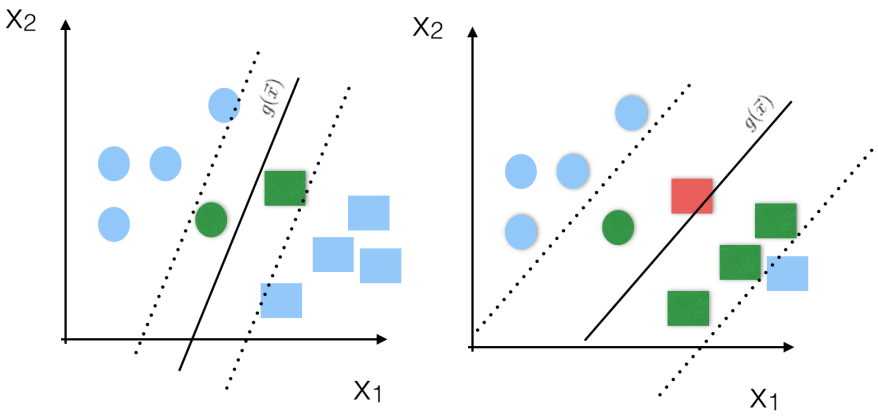


Figure 2.8: Two examples of SVM classifiers with different value of the C parameter.

The observations that exist directly on the margin or violating the margin are the variables that will affect the hyperplane and act as the support vectors. This means that a high C value will probably result in a higher number of observations that act as support vectors, see fig. 2.8 which shows an example of hypeplanes with different values of C on the same dataset. A high value of C allows more violation of the margin which will potentially result in a model less fitted to the training data but with more bias and with lower variance. A low value of C will result in the complete opposite.

Multiple classification

The SVC is a binary classifier which labels data into two classes ± 1, but it can also be constructed to handle multiple classification. The approach is to create a set of binary classifiers which will each get trained to separate one class from the other classes. This approach can be performed with two different methods, one-vs-one classification or one-vs-all classification. One-vs-one classifies all data samples and when all sets of classifiers have been executed, the final classification is determined by the frequency of which class the samples were assigned to. The one-vs-all method compares one class at a time with all other classes to make the classification [11].

Non-linear classifier

In some datasets a linear classifier is not good enough. For those situations there are different functions for creating a hyperplane which are called kernel functions that produce hyperplanes of different shapes. The creation of kernel functions is a research area in itself but some well known kernel functions are: linear, polynomial, radial basis function and the sigmoid function that will create hyperplanes of different shapes. This extended approach to use kernel functions for producing both linear and non-linear classifiers is called Support Vector Machine (SVM) [11].

2.4 Feature Selection

The usage of increasingly advanced tools for performing HCS results in that the number of features that can be extracted per sample can grow rapidly. This increases the need for techniques that can be used for extracting relevant features from a multidimensional dataset. A set off possible techniques will be covered in this thesis and they are explained in this section.

For performing advanced analysis on HCS data, the analysis method must be able to handle all the generated readouts. With such many parameters describing all the data points together with data on a cellular level generating a high number of data points, a characterization for a specific biological response becomes harder to identify. The data generated from HCS also consists of noisy and irrelevant data that contributes to a less accurate depiction of it. This yields the use of feature selection (FS) for selecting relevant features, which is important for creating a model that can be utilised for prediction and classification. The importance of feature selection has increased over the past decade due to the same reason as the increasing popularity of data mining since these two are closely related and often used together. This has resulted in a gain of ongoing research within this area but feature selection is still an unsolved fundamental problem of science [15].

Feature selection (FS) can be seen as a preprocessing step in data mining for selecting data which is relevant and exluding data which can be seen as irrelevant and in such cases does not bring any value for further analysis. Feature selection is important in order to create a good classification model since methods for classification decrease in quality when data consist of noise or irrelevant data.

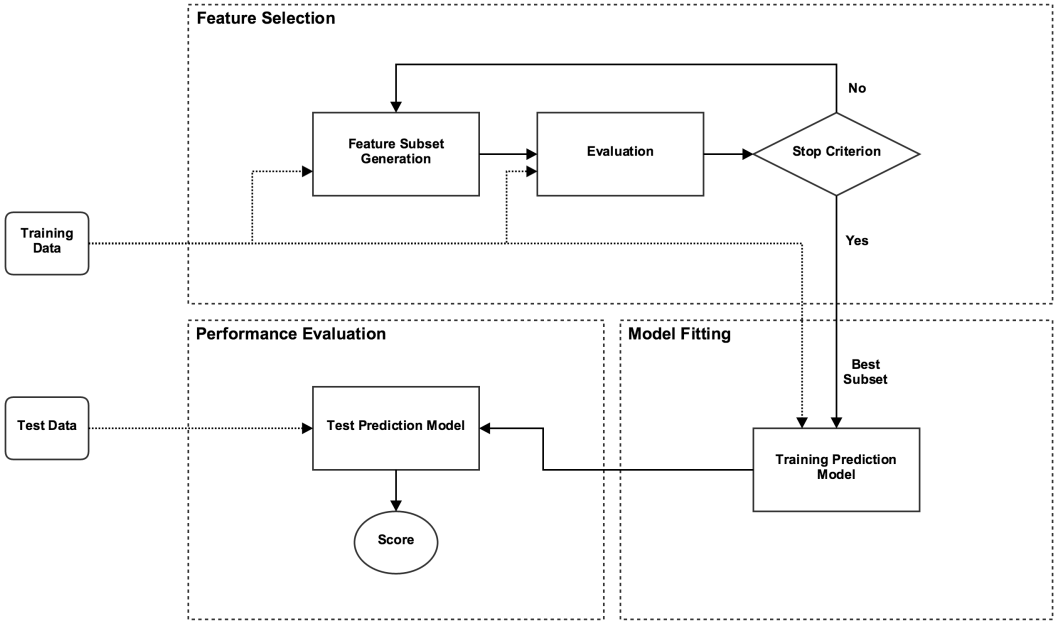


Figure 2.9: The data flow in feature selection. Training data is used to select a subset of features and fit a model, which then is evaluated on test data.

The process of feature selection usually consists of two phases, selecting the features and model fitting and evaluation of the performance/relevance of the selected features. The selection of features has training data as input which is constructed by a percentage of the total number of samples. The features in the subset get evaluated and are either discarded or added to the selection of features according to their relevance. This process is iterated until the selection of features satisfies a stop criterium and the final selection can later be used to filter the training data for model fitting and prediction, see fig. 2.9 [16].

The evaluation of feature selection can be divided into three different categories which are named filters, wrappers and embedded functions [17]. The filter approach separates the selection from

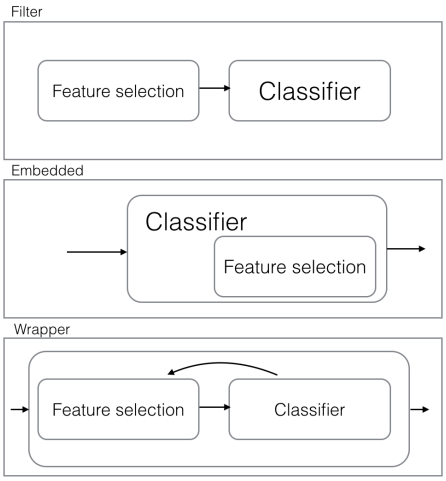


Figure 2.10: The three different groups that feature selection algorithms can be divided into.

the model construction [18]. In most cases the filter techniques only look at intrinsic properties of the data and calculate a score for each feature and threshold features with low score [19]. This approach is easy, fast and scalable for big data sets but often lacks in quality due to the lack of consideration of dependencies between features. The wrapper methods include the evaluation to the selection of features. These methods are tailored to a specific classification algorithm and are called wrappers since the feature selection are wrapped around a classification model. They also take feature dependencies into consideration when performing selection and include interaction between model construction and feature selection. The wrapper methods are usually more suitable for multidimensional data than filters but are often computationally very heavy and suffer from high risk of overfitting. Embedded methods are very similar to the wrapper methods with cooperation between classifier and the feature selection, but the difference is that the embedded methods are embedded into the classifier when wrapper methods distinct the feature selection from the classifier, see fig. 2.10. Embedded methods obtain the same advantages as wrapper methods but do not have the disadvantages of overfitting and expensive computations. But as well as the wrapper methods, the embedded methods are dependent of a specific classification method which gives the filter methods the advantages of having better generalisation ability [20].

The training data could be either labeled, unlabeled or partially labeled, which yields three different categories which are called supervised, unsupervised and semi-supervised feature selection. In the case where the training data is labeled (supervised) the features relevance could be established by evaluating correlation with their class or utility [16]. The unsupervised algorithms with unlabeled data need to calculate variance or distribution of data in its evaluation of features. Finally the semi supervised methods are combinations of both supervised and unsupervised techniques that use the provided labels as additional information for performing unsupervised selection. In multidimensional data one can often find nonlinear patterns and many of the regression and classification methods are built to provide linear models which could affect the quality of the whole data mining. When linear correlations are known the linear classification methods are less expensive computationally and the quality is good enough.

2.4.1 Recursive Feature Elimination

Recursive feature elimination (RFE) is a feature selection algorithm which repeatedly removes the worst performing feature from a given set. This is performed until a predefined number of features are left or if a specifically chosen evaluation method criterion is fulfilled. An external estimator is used and trained in every step of the process and the estimator is responsible for giving weights to the given features and thus also responsible for selecting which features that shall be pruned. A common approach is to use RFE together with a linear SVM algorithm where the feature ranking consist of weight magnitudes which are given by the correlation coefficients of the support vectors [21].

2.4.2 Exhaustive Feature Selection

In order to find the optimal subset for a given set of features, one has to consider a brute force approach that looks at every possible subset [22]. The problem with using a method that calculates the performance of every possible subset is the computational complexity.

If the optimal solution was to be found in a set of N features, and every feature has 2 states in that they are either included or not in the subset, then there would exist 2^N different possibilities which can be considered to be a prohibitive task. If the task was simplified to only include every subset of N features of the total M it would generate $c(M, N)$ subsets calculated by

$$c(m, n) = \frac{m!}{n!(m-n)!} \tag{2.10}$$

where m represents total number of features and n the number of features for a given subset. This is still a heavily computational task, even with parallelization. Such an approach would thus require some constraints to be implemented in practice. The general approach is to make some pre-defined ranking criterion before entering the actual exhaustive search, e.g. it would be possible to look at every subset of 2 features for a total set of 10 features since $c(10, 2) = 45$ different possibilities.

2.4.3 Robust Feature Selection

A new approach of feature selection called Robust feature selection, that has been derived from the field of system biology, can be applied to problems accounting low signal-to-noise ratios, errors-in-variables and near collinearity. The method can be labeled as a filter method which is separated from an objective function. Measurement data contains errors and the features can thus be defined as a set of realizations. Robust Feature Selection (RFS) provides a method for checking all realizations by classifying the features and interactions into the following four classes:

- **Present/Existing**
  The feature is present in every combination of realizations of a target feature and is thus required for explaining the data.
- **Absent/Non-existing**
  The feature must be excluded for explaining the data since it is absent in some combination of all realizations of a target feature.
- **Non-evidental**
  The feature lacks information and thus do not affect the ability to explain data.
- **Alternative**
  Can be selectable, excludable or negligable for explaining data since it is required in some combination but not required in another.

RFS requires a defined error/uncertainty-model to the data in order to check all models within a chosen class that cannot be rejected and construct uncertainty sets based on that data which represent the uncertainty of samples within the dataset. By considering all realizations of unrejectable variables with an error model at a desired significance level, robustness is achieved [15]. The following formulas and definition will describe the procedure of creating uncertainty sets, separate features into classes and how the feature selection works in general.

The procedure of performing Robust feature selection is accomplished through calculating Nordling's confidence score [15] $\gamma(j)$, given by

$$\gamma(j) \triangleq \sigma_n(\Psi(\chi, j)) \tag{2.11}$$

where each feature in the dataset is represented through j, and only selecting those with a score above 1 to the final subset. The resulting value is computed as the smallest non-zero singular value and denoted as σ_n. The matrix Ψ is given through calculating each element ψ_{kl} in

$$\psi_{kl}(\chi, j) \triangleq \frac{\psi_{kl}(j)}{\sqrt{\chi^{-2}(\alpha, nm)\lambda_{kl}}} \tag{2.12}$$

where k and l represents indexes of row and column in a matrix with a total of m rows and n columns. The computation of the confidence score requires that a dataset is given together with a matrix describing the variance, denoted as λ, of the measurement errors v_j and ϵ in the data model, see eq. 2.1 and 2.2. Parameter $\psi_{kl}(j)$ is recieved from the matrices

$$\Psi(j) \triangleq [\phi_1, \ldots, \phi_{j-1}, \phi_{j+1}, \ldots, \phi_n, \xi] \text{ for } j \in \mathcal{V} \tag{2.13}$$

$$\Psi(0) \triangleq [\phi_1, \ldots, \phi_j, \ldots, \phi_n] \text{ for } j \in \mathcal{V} \tag{2.14}$$

$$\Psi(\infty) \triangleq [\phi_1, \ldots, \phi_j, \ldots, \phi_n, \xi] \text{ for } j \in \mathcal{V} \tag{2.15}$$

where ϕ_j corresponds to a regressor, ξ to the regressand and $\mathcal{V}$ to a given set of features. The inverse of the chi-square cumulative distribution $\chi^{-2}(\alpha, nm)$ is calculated with nm degrees of freedom for the corresponding probability which is defined as the desired significance level α. The value for α is typically set to the standard level of significance for justifying a statistically significant effect, $\alpha = 0.05$.

A signal-to-noise ratio is also used in the process by calculating

$$\text{SNR}(\phi_j) \triangleq \frac{1}{\sqrt{\chi^{-2}(\alpha, m)}} \sqrt{\sum_{k=1}^{m} \frac{\phi_{kj}^2}{\lambda_k}} \tag{2.16}$$

and it is used for comparing the level of noise with each regressor ϕ_j.

The algorithm for computing the confidence scores starts with adding all considered feature to an index set $\mathcal{V} = \{1, 2, \ldots, n\}$. If the number of rows (samples) m for a given matrix (dataset) is less than the number of columns (features) n, then the $n - m$ features with the smallest signal-to-noise ratio $\text{SNR}(\phi_j)$ must be removed from feature index set $\mathcal{V}$. The feature with the smallest signal-to-noise ratio $\text{SNR}(\phi_j)$ of the remaining features in feature index set $\mathcal{V}$ is then removed if both the confidence scores $\gamma(0)$ and $\gamma(\infty)$ are less than 1. This step is iterated and features are removed from the index set until one of the confidence scores equals or goes above the score of 1. The features that are removed will have scores of 0 and the rest of the features will be used for calculating new confidence scores $\gamma(j)$. Of the resulting scores, the features with a score above 1.0 are required for explaining the regressand and thus included in the final subset of relevant features for describing the dataset. The features resulting in scores $0 - 1$ are not required but can be included for noise realisations.

2.5 Evaluation Methods

The creation of data models can be considered as more art than science, there is no defined way of creating a perfect model for predicting data. Different techniques can however be applied for estimating the performance and these are described in this section.

Different quality measures can be used for validating the performances of prediction algorithms and estimate how accurately they will perform in practice. These methods are commonly used for determining if a chosen subset of features performs better than another for a given estimator but also make sure that no overfitting is occurring. Overfitting can be described as when a model is too complex for making good predictions on real world data and thus only customized for the training data.

For evaluating the performance of a created prediction model, one often splits the original dataset into two parts where one defines the training set and the other the test set. The training set is used for building the prediction model, which tries to fit itself according the samples. The test set is used for computing the performance of the prediction model in its final state and on unseen data, i.e. data that has not been involved in the fitting steps.

2.5.1 Cross Validation

Cross validation (CV) is a commonly used validation technique for prediction models. It comes in variations that can be separated into exhaustive and non-exhaustive methods. Exhaustive cross validation splits the data into a training set and validation for all possible combinations while a non-exhaustive approach only considers a certain amount of those combinations.

The standard technique to use for a non-exhaustive approach is to divide the dataset into two parts where one acts for training the prediction model and then validates the model with help of the other part. Different methods exist for improving the result for performing cross validation, e.g. the K-fold method [11]. This method divides the data into k number of subsets, with the variable k specified externally. The standard method of evaluating the model with a validation set is performed k-times with one of the subsets used as validation set and the others used for training the model. The mean square error MSE is calculated by

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 \tag{2.17}$$

where $\hat{f}(x_i)$ is the predictions of the observations y_i for a total of n samples. This is computed for the samples in the validation set and the performance of the prediction model is then calculated by

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i \tag{2.18}$$

where $CV_{(k)}$ relates to the average of all k mean square errors.

2.5.2 Gini Index and Cross Entropy

The Gini index (also called Gini coefficient) is an old measurement of inequalities among values [23]. It can for example be defined as a measurement of the total variance across the different classes in a dataset containing multiple features [11]. It is used by e.g. decision tree classifiers as a classification criteria for measuring the quality of a specific split. It is considered to be a node purity measurement where small values are significant for nodes with samples that are predominant from one specific class. The purity of a node is measured by how the data is split by that node, if the major part of the data within a specific class got split on one side of the binary split the purity is high and if the data is equally split by the node the purity is low.

The computation of the Gini index can be given as

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{2.19}$$

where $\hat{p}_{mk}$ represents the ratio of training observations of the m^{th} region from the k^{th} class and K the total amount of classes. Small values for G will be received if $\hat{p}_{mk}$ is close to 0 or 1. An alternative for the Gini index measurement is Cross entropy which can be computed by

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log\ \hat{p}_{mk} \tag{2.20}$$

and it behaves in a similar way in that D will result in small values if the m^{th} region is pure, i.e. it will have a predominantly dominance of a single class.

2.6 Data Handling with SciDB

HCS generates data on a cellular level which can be of large proportions and this creates requirements of scalable and robust data handling techniques. This section describes the data management tools used for this project and their essential functionality.

SciDB is an open-source array database management system made for handling large amounts of scientific data [24]. It is developed for the purpose of making out-of-memory computations available through different statistical and linear algebra operations.

2.6.1 Data Model

The native data model used in SciDB is defined as a multidimensional array data model. For a database utilising complex analytics computations there is an advantage of using this kind of data model because most analytics are computed through core linear algebra operations and these can be performed with support from arrays. An array in SciDB can be specified with N number of dimensions and every individual cell in that array can contain an arbitrary number of attributes. The attributes can be of any defined data type and are uniform throughout the array. This means that the SciDB database contains a collection of n-dimensional arrays with cells that each consists of a tuple with values that are distinguishable by a specifically given key. The attributes must be conformant throughout the array.

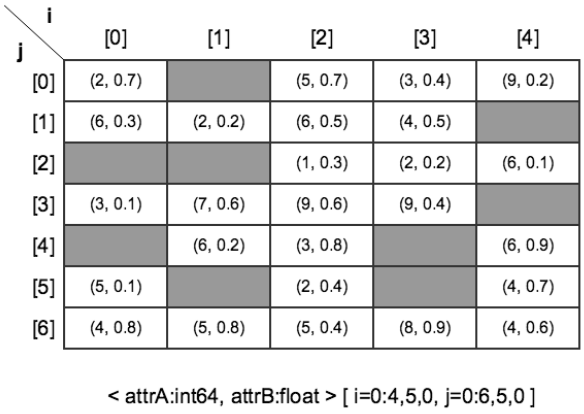


Figure 2.11: An example of a two dimensional sparse array in SciDB.

For an example of a sparse array together with its schema, see fig. 2.11 which describes a two dimensional array with index i and j together with two attributes at each index. The schema below the grid in the figure defines type of attributes, how many index in each dimension, chunk size and memory overlap.

SciDB supports two types of query language; AQL (array query language) uses an SQL-like syntax and is, when executed, compiled into AFL (array function language) which holds the most common functionality for performing operations in the database. In addition there exist interfaces for the ability of processing data from R (SciDB-R) and Python (SciDB-Py). This is performed through Shim which is a SciDB client that exposes functionality through an HTTP API. The Python interface SciDB-Py provides interconnection to multiple other Python libraries related to scientific computations, e.g. NumPy, SciPy and Pandas.

A SciDB database has functionality for storing sparse arrays, i.e. arrays that contain empty cells. The functionality of managing empty cells is important when applying data manipulation operations because these need to be ignored. When applying multiple dimensions, the amount of empty cells also tends to become large. An array can also consist of NULL values but they are distinguished from empty cells in that they are treated as existing cells in the array but with no containing value. The data stored in an array can be of any numerical or string type but needs to be explicitly defined when creating an array. There is also support for user defined data types.

An array must be defined with at least one dimension which forms the coordinate system to use. When creating an array, the dimension is created with a name, lower and higher boundary index together with values for chunk size and chunk overlay. An array dimension can be created as an unbounded dimension by declaring no higher boundary index. This enables the dimensions to update dynamically as new data are added to the array.

2.6.2 Design and Architecture

SciDB is created with scalability in mind due to that an instance can be deployed over a network of computers. A shared nothing design is adopted where each node in the cluster runs its own SciDB engine together with a local storage [25]. A central coordinator stores information of all nodes and is responsible for distributing query processes and providing communications between them. The storage manager of the database adapts a no-overwrite approach and thus, there is no functionality for updating data, only appending new.

The arrays in the database are decomposed into different parts. The different attributes are partitioned in arrays where each attribute is stored individually and all low level operations in SciDB are performed on these single value arrays. The arrays are then further broken down into equally sized parts called chunks. The chunks in SciDB can be defined as the units which all processes and communications operate on. The size of the chunks shall be specified for each dataset and the performance of operations can have a large difference in selecting correct chunk sizes contra wrong ones. Chunks can also be specified together with overlays for achieving parallelization of operations utilising the cell neighborhood, which otherwise would require stitching of adjacent chunks.

2.6.3 Comparison

The most significant property of SciDB is its definition of being a computational database. SciDB offers both storage and an analysis platform in one package, data is not required to be extracted or reformatted for performing mathematical operations on it. This advantage is why most kinds of highly faceted data such as bioinformatic data, sensor data and financial data are well suited for use in array data models rather than tables which are used in relational databases [26]. The term relational database is given for databases structured by entities in a tabular form containing rows and columns, which have different types of relations between eachother. This kind of database is not designed for performing complex analytics on scientific data which gives poor performance. Schema-less NoSQL alternatives are also considered as bad options because schema enforcement is required for highly structured data and the process of receiving that moves the burden from the storage layer to the application layer.

The main problem with other analysis software is that they most of the time do not store data which creates requirements of data extraction, formatting and exporting to a specific software or package where the analysis is going to be performed. These in-memory solutions also limit the amount of data that can be processed at a given time. A solution to get rid of this problem can be MapReduce which is a programming model that can be applied to process and generate large datasets by distributing the computations across multiple instances and perform map and reduce methods in parallel [27]. One ecosystem that uses this kind of computations is Hadoop, created for performing massively parallel computing [28]. These kind of techniques can be used for processing large datasets but are given as extensive frameworks, which makes it more heavy for implementation. The reason for selecting SciDB to work with is mainly based on its promising references for usage within bioinformatics. The possibility of utilising out-of-memory computations together with the ability of scaling the system over multiple instances creates good support for using even larger datasets in the future.

2.7 Summary of Related Work

This section presents a summary of the research related to this thesis. A plot of how many publications that have been published over the last decade is also shown to map how the popularity and importance in this area of research is evolving.

Many of the relevant publications have focused on making a comparative study of different classifiers and feature selection methods used on different types of datasets in an experiment to map if specific feature selection methods suits better for specific kinds of datasets.

Figure 2.12 describes the evolvement of the amount of search hits for the different combination of keyword over the last decade. The different lines corresponds to the different combination of the key

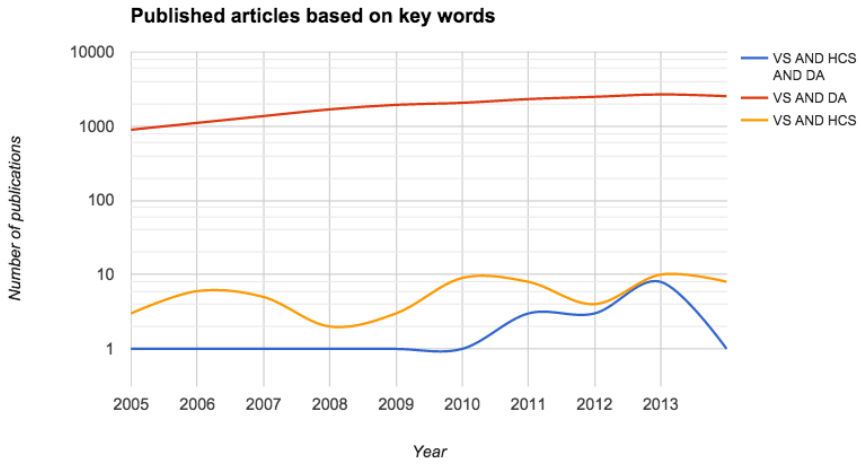


Figure 2.12: Number of search hits that was found during literature search with the different combination of key words.

words "Variable Selection" (VS), "High Content Screening" (HCS) and "Data Analysis" (DA) with synonyms that were used. In detail how this literature search was performed is described in Appendix A. This shows that the field of HCS in combination with data analysis and feature selection are not widely considered within the data mining area of research but have an increasing interest after the year 2010. However, the amount of articles found within this area can be considered to be a small number and is thus proving that there is not so much research yet performed that spans over the three fields. Some of the relevant articles in this figure have been used for the purpose of conducting this thesis. The main use of the search terms is only to provide an overview of how the research within these fields has changed over time and how much that spans over different fields.

Most of the found related research has focused on a more general perspective of how and which data mining algorithm to use for bioinformatic data in an attempt to find best practises and benchmarks for different methods for different types of data sets. This thesis focuses on comparing a smaller amount of recommended methods specific for HCS data and how different settings for classifier, preprocessing and feature selection effects the result.

Abraham et al. [7] enlighten the development of screening where single readouts from biological assays have grown to multiple readouts which result in multidimensional data to analyse. They present different assay methods including HCS and discuss them from a biological perspective. The authors also describe the main approaches within data mining for analyzing multidimensional data which are feature selection, distance measures, supervised- and unsupervised learning and describe their main objective and example of usage. This publication brings answers to the questions of how to visualise multidimensional data of this nature and which data mining algorithms that are most popular, but not which algorithms that provide good results.

Bolón-Canedo et al. [20] test several types of feature selection methods with different classifiers on synthetic datasets. This publication focused on comparing a lot of different feature selection algorithms on different types of dataset and generate results to see which algorithms that performs best. Especially interesting was the experiments that were performed on microarray datasets with large amount of features and noise affected data which have similar properties of the datasets generated from HCS. This publication answers questions about which feature selection that seems to be most stable and provide good result on different datasets but do not consider datasets that are generated from a HCS.

For gene expression microarray data which has similar properties as data generated from HCS, a feature selection algorithm was developed by Yu and Liu [29]. The algorithm used was an unsupervised classifier which was compared to a small collection of well known feature selection methods on three different microarray datasets. The result of the different algorithms was calculated with

help of the evaluation methods leave-one-out cross-validation.

Publications that consider HCS from a data analysis perspective and focusing on which methods that bring best result were not found. Most of the results for the blue line in fig. 2.12 were general reviews which either consider HCS as a method for generating multidimensional data or data analysis as a minor part of the HCS pipeline. The result from the literature search in fig. 2.12 which consisted of all the keywords were publications that consists of general reviews which consider HCS as a small part of the increasing problem to solve in analysing multidimensional data.

Chapter 3

Method

This chapter will describe the implementation of the theory described in the previous chapter for developing an application that will enable efficient analysis of ceullular data from HCS experiments.

The following decisions have been taking of how to design the application that has been created within this thesis. These follow the general requirements that have been decided by the end user together with the authors of this thesis. The implemented application:

- Shall manage data up to multiple GB in size.
- Shall be easy to access and use.
- Shall perform data analysis operations for classification and feature selection in an automated manner.
- All decisions shall be controlled by the user.
- Exporting functionality for ability of performing visualisation in other software.

The aim is to fill the gap in the workflow with the data analysis by providing an automatic tool for handling data generated from MetaXpress directly and perform data analysis on multidimensional datasets that is infeasible to manage manually.

The solution was to create a web application mainly because of the platform independence and the opportunity to perform cloud computing and handle big data. The client side of the application will perform parsing of files but all other calculation and managing of data for the data analysis will be performed on server side. The application will enable uploading data from CSV file customised from MetaXpress data and optionally enable to add additional data through simultaneously upload an Excel file template with additional information for the data and automatically match data in the dataset with the annotation and merge the information in the database. A smaller collection of well known supervised learning algorithms and feature selection methods were implemented and coupled together. Focus will be on creating an automated pipeline for performing data analysis, but in order to make the user in control of all decisions that are made in the analysis, simple visual response will be provided to the user with options to proceed with any action prefered. See fig. 3.6 for how the feature selection shall be performed in the application and the subsection about prediction for how the result of feature selection can be used to predict data. When the analysis is finished the result shall easily be exported for further visualisation and analysis in other software.

3.1 Establishing the Core Functionality

To map which functionality that is missing or forming a bottle neck in the current workflow an investigation was conducted with the end user. This investigation is described in Appendix A an this section will summaries this investigation, describe the current used softwares and conclude which functionality that is missing that the application created in this thesis will cover.

To establish what current software lacks in functionality and prove state of the art with the

application that is created in this thesis, an investigation of the existing tools, which serve similar purposes as the defined end user was using or had access to was conducted. Read Appendix A for a more detailed description of how the tools are used in the current workflow. To map the functionalities of the software and prove the need of an application with specific features, different functionalities of the software where listed in table 3.1. This list of tools were only based on the tools the end user was using, but more software was considered during the research. A list of relevant software can be found in [4] together with applications for handling data.

Software/ Features	Cloud computing	Advanced data analysis	Manual data analysis	Easy to use	Visualisation	Export functionality	Specialized for HCS
Excel			**X**	x	x	x	
KNIME		x	x		**X**	x	
Spotfire	x				**X**	x	
Meta Xpress						x	x
Cell Profiler Analyst		**X**			x		
Goal for application in this thesis	x	**X**	x	**X**		x	**X**

Table 3.1: To map functionality that exist in current used software and which functionality that is possible to bridge in our application this table was created from the information about the different software analysed below. Big cross means that the feature is one of the key feature of the specific software and small cross means that it is a regular feature in the software.

The tools analysed in this section have been tested and evaluated to gain experience, get inspiration and establish which functionality that current software lacks that the software created in this thesis will provide. The chosen tools for evaluation are *MetaXpress*, *Cell Profiler Analyst*, *Excel*, *KNIME* and *Spotfire*. A brief description of these software is included below as a summarised table of the established division of the software can be seen in table 3.1.

MetaXpress [30] is a software for acquisition and processing of HCS images. The extracted data is exported in CSV format which will be the input format for data for the implementation in this thesis. This program was evaluated to test possibilities with extracted features and to see the structure and size of exported data.

The open-source software *CellProfiler Analyst* [31] is specialised for cell analysis and closely related to the image analysis software *CellProfiler* that extracts data from cell images. These two software could be used separately but work well in combination. *CellProfiler Analyst* provides features for processing the data with machine learning algorithms for exploring the data and gives some basic visualisation options for analysing the result. This program was tested for inspiration of how the analysis could be performed and which techniques that could be used. The end user assessed this software as too complex to start using and that learning how to use it with profit would require too much time. To make use of data extracted from image analysis, the two programs are also constricted for usage together. There is also a requirement of setting up an database to be able to load data into the program makes the program inflexible for the user which have to adjust the data to the program.

The program is well suited for cell biology and consist of a big variety of features and tools to be used, which are configured to work for all cases of usage and. This makes the program hard to manage due to the many configurations that needs to be performed, which require knowledge in both data mining and cell biology. This variety of tools makes it confusing for the user to know which tools to use when and lacks of easy accessed documentation to be able to solve this confusion without conducting research.

Excel is the most used tool for managing and performing analysis today thanks to its intuitive spreadsheet layout and easy navigation. The main focus within this software is data management and processing table data with manual parallelized operations with basic math, statistical and text manipulative operations. Excel also have options for creating visualisations in form of basic charts like pie charts, bar charts and linear diagrams. Excel [1] is limited to the RAM of the computer in use for how much data it can handle. It is also limited to how many rows and columns each sheet can have to approximately 10^6 rows which is considered as too low for cellular data gener-

ated from HCS. See document about specifications and limit [32] for more info. Excel was tested and evaluated to get inspiration and point out which operations that are difficult and/or too time consuming to perform.

KNIME [33] is a workflow system that enables the user to control the analysis from data acquisition to visualisation with a flow chart layout of the interface where every module describes a certain step in the workflow. The reason for testing this software was to evaluate this way of performing the analysis.

Spotfire [2] provides visualisation and enables the user to interact with the visualisation and filter interesting data. The evaluation of this software gives inspiration of which visualisation methods that could be used as well as its constraints.

Table 3.1 shows that the core functionality for the application created in this thesis is focused on an easy to use tool for performing advanced data analysis that can provide functionality of exporting results for conducting visualisation in other software.

Since no restrictions want to be made in the size of data that could be handled, cloud computing[1] was a key feature that was wanted, which many of the compared software did not provide. Also some general functionality needs to exist to perform the data analysis like basic manually editing options for loaded data to increase result to the data analysis. Usability will be achieved by only providing required functionality that has significance for the end user. It is also important to continuously perform user testing to make the application customised for the end user. This is because the automatic steps, that will replace the manual preprocessing of the data, are required to work as expected.

3.2 Overview, Architecture and Tools

This section describes the main structure of the application and its high-level architecture. It will also provide a compilation of the different tools, libraries and frameworks used in the development process of the application.

The chosen approach for the application developed during this thesis project is a cloud based *SaaS* (Software as a Service) solution that is reachable through the web browser. The main idea behind this is to provide a cross-platform availability that has no requirement of installation or other configurations which is believed to give the best possible user experience for an end user without extensive technical knowledge about the service itself. The utilization of cloud computing also provides possibilities of building a system that is scalable in that more hardware can always be provided for more performance and computing power. Such a system also supports maintainability in that it eases integration of new features, which could quickly be deployed to a new version of the application. The user has no responsibility of updating any software on their own.

Besides the benefits of deploying a cloud based web application, there also exist some disadvantages that need to be considered. The biggest concern relates to security of the application and the data maintained within. A remote cloud based infrastructure gives anyone the ability of accessing the public content that is distributed. This requires all information that has some sort of confidential status to be protected from unauthorized users which requires the use of a strong security layer.

A high-level design for the implementation is provided in fig. 3.1 and it describes an overview of the data flow within the system. The layered structure shown is separated into one part of the server side implementation and another part of the client side implementation. The client side in this architecture represents the code received by the user from the first request made from the browser. A brief description of the responsibilities from the different modules is described in the following subsections together with information about which tools, libraries and frameworks

[1] Cloud computing is when all calculation and data is stored on a server somewhere where capacity of the server easily could be scaled to fit the required data.

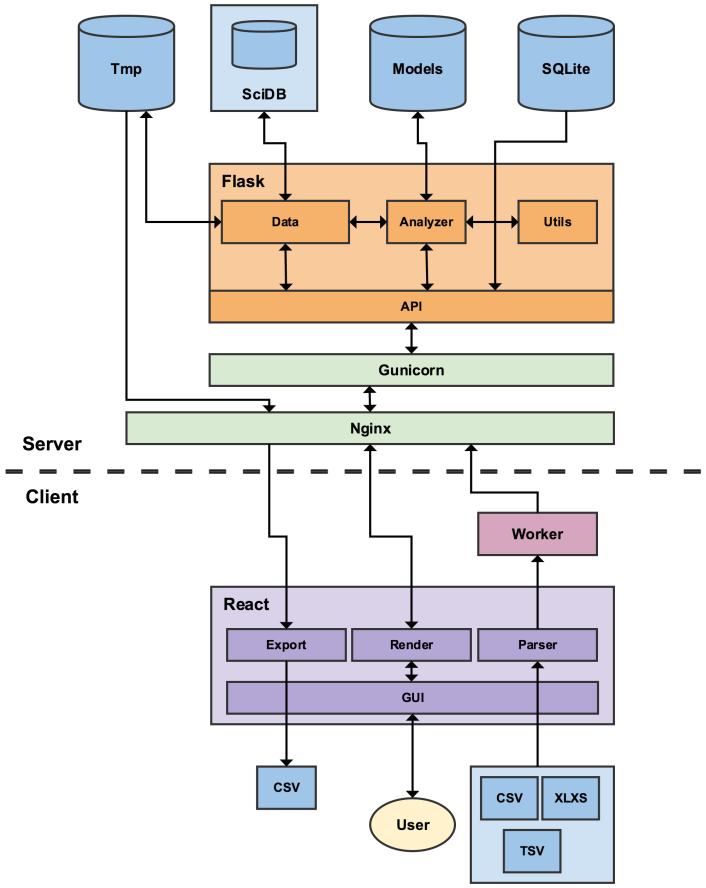


Figure 3.1: High-level design of the application.

are utilised. The third-party software that have been used for this application have mainly been selected according to these criterias:

- Will it ease the development and prevent reinventing the wheel?
- Does it have a good community and good reputation from other users?
- Does it have a good future predicted?

3.2.1 Client Side

The client side is composed with React [34] which is a JavaScript framework for building user interfaces. It is developed by Facebook and has a good reputation of handling large amounts of data in the DOM[2] which affected the choice of client side framework to work with. React works with components that contain states of the data and renders DOM elements with an XML[3] like structure provided by a syntax extension for JavaScript called JSX[4]. React utilises a virtual DOM such that when a state changes in a component, only the part rendered by that specific component gets re-rendered. The structure of the code written with React also automatically becomes very modular which makes components reusable and easily combined with other components. Another framework that is used for the client side and in combination with React is Flux [35]. This framework provides extended functionality to React and provides a MVC[5]-like pattern to the client side

[2]Document Object Model is an interface for representation and interaction with objects in HTML, XHTML and XML documents.
[3]Extensible Markup Language is a markup language, designed to describe data.
[4]JSX is a XML-like JavaScript syntax extension.
[5]Model-view-controller is a architectural design pattern commonly used for describing applications with user interfaces.

that can be utilised for a more manageable data flow within different parts of the application. The main component of the client side is the Graphical User Interface (GUI) which gives the user a visual appearance and provides interaction with the implemented features. A lot of the design and feature functionality of the GUI is given through Bootstrap [36] and jQuery [37]. Bootstrap provides responsive design and intuitive component interfaces and was chosen for setting up an acceptable design of the application with a minimum amount of time. The inclusion of jQuery is almost standard when creating JavaScript applications and it is mainly used for quick manipulations of HTML components but it also has an interface for making simple Ajax requests.

The Parser module and the Worker module contain functionality for loading and parsing files, see section 3.3.1 for further information. The Parser is responsible for loading the files locally and then distributes the work of parsing and uploading the files to the Worker module which is wrapped around the Web Worker API. This module uses a fast and powerful library called PapaParse [38] for streaming and parsing comma- or tab-separated values. This parsing library was chosen as it is considered as one of the fastest available for the browser.

The client side also includes an export service through the Export module. This module makes a request via the server for externally downloading a file which content is made public the moment it asks for it. Other features on the client side exist in the Render module which manages remaining user requests, makes calls to the server and distributes the responses to the GUI. A simple Excel-like grid was implemented in the application to get some visual response of loaded data with ability of some basic functionality like editing cell values and reordering columns for example. This was implemented with the jQuery based spreadsheet library Slickgrid.

3.2.2 Server Side

The server side is deployed in a Linux environment and is composed of multiple different layers. The bottom of the stack contains the web server Nginx [39]. It is used as a proxy server serving all static files, e.g. the client side code, but also manages all requests to the rest of the server side implementation and all responses to the client side. The server side application is implemented as a Python application which means that a Web Server Gateway Interface (WSGI) is required for communication between the application and the proxy server interface. For that purpose, Gunicorn [40] is used as an application web server running on localhost and providing an interface to the application. The reason for choosing to implement the system with Python is because it has well established reputation for usage in big data productions with a large stack of powerful packages useful for data analysis. It is also well proven for building massively scalable web applications and it is an easy process to setup a platform independent python environment and start implementing.

As a server side framework, Flask [41] is chosen. Flask is a microframework with support for extension packages with functionality of providing e.g. user authentication and RESTful APIs. The API module is built with a RESTful approach and supplies a layer of communication between the server functionality and other resources. Basic security is given by token based authentication with every request. A user enters his/her credentials and makes a login request and in response gets a token that is used to verify other requests with. It is important for an application of this sort to have some layer of security because of the open accessibility that exists when deploying in the cloud. Also the data handled in this thesis is used for research purposes and is thus considered confidential which makes the application useless if the application functionality was made public. The user management within the application is made very simple due to that it focuses on providing support for one single user only. The user credentials are stored in an SQLite [42] provided database.

One of the main modules, the “Data” module, provides an interface to the database management system SciDB [24], see more in section 3.3.3. The other one, the “Analyzer” module, contains functionality for applying analysis methods on the data. This module includes multiple different Python specific resources for its purpose, e.g. scikit-learn [43] is used for providing machine learning techniques. Scikit-learn was selected because it is an open source library, widely used by developers and well supported. Other basic Python libraries are also used for computing purposes. The storage module, “Models”, handles file storage of objects created in the Analyzer module,

more information of how this works can be found in section 3.4. The "Utils" module contains additional tools not covered by the other modules. The "Tmp" module is another file storage made for serving files that can be either downloaded or loaded into the database.

3.2.3 Tools

A set of tools has been used in the development process in order to provide extensions to the application in the future. Virtualenv [44] is a tool to create isolated Python environments with specifically selected dependencies and versions for a chosen Python installation. In that way, libraries can be updated or changed without automatically affecting the application. Gulp [45] is used as an automation tool for the client side workflow. Tasks are e.g. provided for setting up local development servers and for building a production version of the application. As package managers, Bower [46] and npm [47] are used. Bower is a optimized for providing packages for front-end production. Npm is most commonly used for managing Node.js [48] modules but can be extended to handle front-end dependencies with use of Browserify [49]. Since React-specific code has to be compiled from JSX to pure JavaScript, one has to use external libraries for using Browserify together with React, e.g. reactify [50] is one such tool.

3.3 Data Management

Data management is an important part of the analysis pipeline since it affects both the usability of the application as well as how good different algorithms perform. This section describes how the data are being handled through the system.

In order to process the data in an appropriate way, a flexible data model needs to be created which provides all functionality required for manipulating the data sets. The whole pipeline of data management includes how to actually load data into the application and how it is stored and managed within the application.

3.3.1 Formats and Parsing

Parsing strict CSV formatted files is an easy task today with the help of the large number of different tools and libraries there are today. But when there is an internal structure in the CSV file which violates the strict CSV standard for example, where first row should consists of equal number of columns and the first row is the only row that should consist of header data, parsing can get really difficult. To create a general parser that interprets the internal structure of the files and parse the data according to that is an interesting problem per se, but will not be taken into consideration. In this thesis a limitation has been set to only consider CSV files which follow the strict CSV format and the structure the dataset retrieves from Meta Xpress.

The data from Meta Xpress [30] is in tab separated format with a customized structure where plate specific data comes first, followed by column headers together with cell level data. One dataset can consist of multiple plates and when a new plate occurs in the dataset new plate specific data appear which is followed by column header and cell leveled data until a new plate begins.

The information in the dataset retrieved from Meta Xpress needs to be complemented with general information about experiments which our end user want to fill in manually. The solution was to create a template in Excel where additional data about the experiment and some plate specific annotation could be added, which optionally could be uploaded together with the dataset. The datasets will be merged to one dataset and matched on specific keys that appear in both the dataset and the annotation file. Before the parsing started a preview of the datasets first 1000 rows is done in order to check file types for each column. Since the data can be quite sparse, all samples do not necessarily have values in the first samples, so the preview will keep looking until all columns have found a sample value separated from null or until it reaches the 1000th row. If no sample is found then the type will be set to string since strings can handle both numbers and strings.

The parsing is processed in a sequence where firstly the annotation data is parsed from the excel

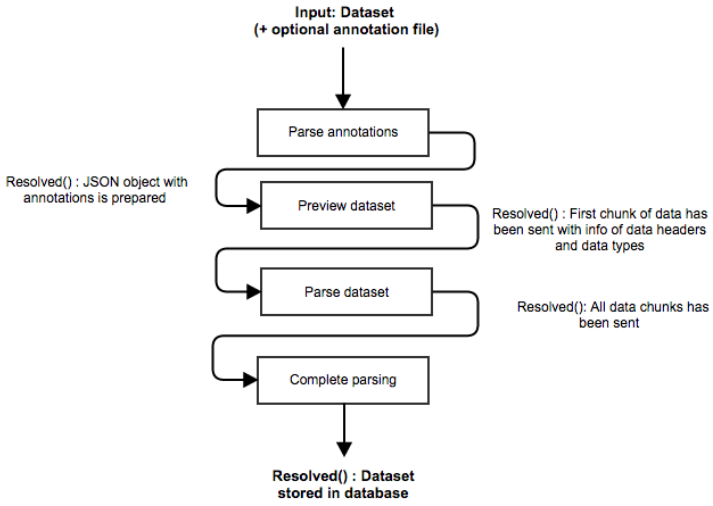


Figure 3.2: The flow of data through the different stage of parsing.

file and stored in JSON. Then the preview is performed on the dataset in order to prepare a first chunk to send to database to prepare the storage, followed by sending the chunks of data and finally send a complete response to the client which tells that the progress is completed. This is performed with help of promises to handle the asynchronous way of executing code in JavaScript such that the different steps is executed sequentially. The sequence is controlled with callback functions "resolve" and "reject" which indicate if the step has succeeded or not. The resolve function will proceed to the next step in the promise stack and the reject function will immediately abort the process and send an error message, see fig. 3.2.

3.3.2 Uploading the Data

When data it parsed from CSV format to JSON and optionally matched with an annotation file 3.2 it needs to be uploaded to the server and stored into a database.

Since no limitation in file size is set the uploading needs to be performed bitwise in chunks. The first chunk of data will only contain information about headers of the different columns and one row of sample data to be able to initialize an array in the database with the right structure and types of the attributes that will be filled with data. The following chunks will consist of data.

The optimal chunk size was established by testing to send different sizes of chunks with regard to the number of rows. Chunk sizes of 1000, 10000, 100000 and 200000 rows were tested and the fastest and most stable upload was by using 100000 rows. The difference between 100000 and 200000 rows was small but to prevent overloading of the server the smaller chunk of those was the most stable choice.

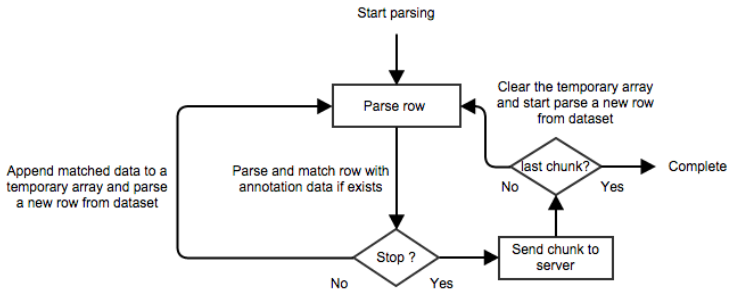


Figure 3.3: The process of parsing and upload chunks

To not have the whole file in memory on client side the file also gets read in a stream so the parsing

performs until a certain stop criteria is fulfilled, see fig.3.3. The chunk of parsed data is sent via an AJAX call to the server and a new chunk starts parsing and then sent to the server. The stop criteria of when to start a new chunk differs depending on format of the CSV file (see section Format and Parsing) and how many rows of data there are left in the file.

To not lock the browser when performing upload a dedicated web worker [51] was used to run the uploading on a different thread. This enables the application to perform other tasks in the application in parallel when upload is in progress.

3.3.3 Data Layer

SciDB is used as a database management system within the application. The reason for choosing SciDB was to have a system made for scalable systems, engaging out-of-memory computations and the ability of storing n-dimensional arrays. Due to the lack of available APIs for communicating between Python and SciDB, an extensive data layer has been implemented for handling the communication. SciDB-Py is an API that was applied and tested in the process but neglected due to different limits in its behaviour of handling data that could not be worked around, see further details in the following paragraphs.

The layer is executing queries to SciDB through a client called *iquery* that can be passed with multiple different parameters, e.g. for handling output formats. The queries are run by functions used for creating schemas and arrays but also for getting data in different formats, for manipulation of cells in existing arrays and addition of new attributes. Also other support functionality exists e.g. calculation of chunk sizes, serialization of strings and loading/writing to file.

SciDB in its current status is yet lacking of basic functionalities, e.g. inserting data into arrays via the memory. The methods fully supported now are primarily through using existing parsing and loading scripts for different file formats. These scripts are provided by the SciDB community but require the data to be contained in a file rather than in the memory, which makes it hard to utilise when transmitting data over e.g. Ajax requests. The Python interface SciDB-Py [52] has functionality for converting NumPy arrays to objects for storage in SciDB. The downside is that it takes a large amount of time for this type of converting, which could probably be worked around by inserting data directly into the database. Data can be directly inserted by using the build function in the AFL language. However this function has constraints in that it only allows single attribute arrays to be built with a bounded dimension. This can be overcome by also utilising the AFL join function which intersects two arrays into one. The problem in such a case is that the HTTP Shim interface has a max limit of 1000000 characters for the queries which becomes a constraint if the data is going to be stringified and sent through use of Shim queries. The method chosen in this thesis is therefore to write all uploaded chunks of data to a temporary CSV file and when the last chunk has been written, load the file into the database via existing scripts.

One important task of the data management is the handling of different data types. SciDB can utilise all the common data types as well as user defined ones. The difficulties that exist are in knowing which type to use for a specific attribute in the array. A preview of the data set is performed at the client side where the data is scanned for a chosen number of rows. This is required because the dataset can appear as sparse in that all cells do not always contain data. In this way one can check if a value exists and then find out which data type the parser is using. The data types used are restricted to strings and floats. This limitations is required because if a numeric value exists for a specific feature and also defined as e.g. an integer, there is no knowledge if it has a uniform data type through all data points. The data type selected for a specific feature needs to be able to fit all values for that feature. Strings and floats fulfill that criteria in that they fit for all expected values. The downside is that the data occupies more data storage even if there is no requirement for it. Another approach would be to select the datatype with the least memory storage and cast to a larger when it is necessary. Most of the values is suited for string and float storage, so such an approach would require a large amount of insert operations that will fail and thus have negative affection on the performance. The interface of SciDB also have limitations in its inserting functionality that contradicts such an approach.

3.4 Data Analysis

This section defines the process of performing analysis on multidimensional HCS generated data. It covers all steps from training to prediction and how they are implemented in the application.

The analysis part of the application is implemented as a two-step process. The first step considers the feature selection phase where features are selected as representatives for a specific dataset and also the training phase of the classification model where the parameters of each model are adjusted accordingly to the selected features. The second step is about the usage of the classification models which can be used for prediction.

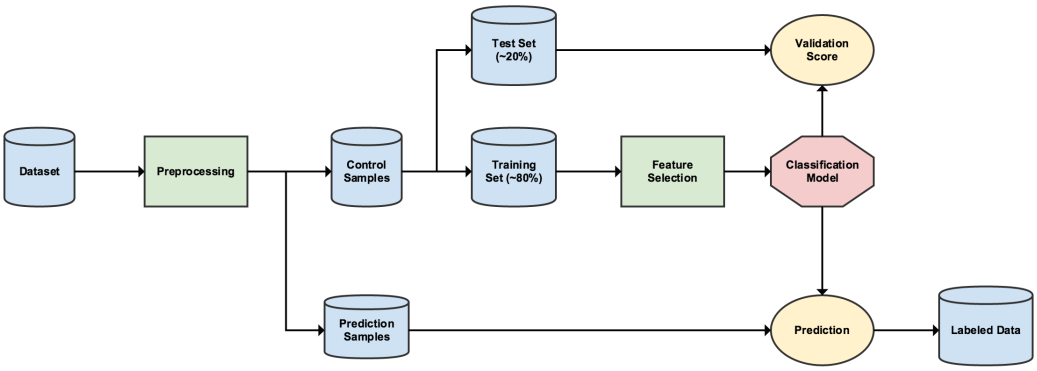


Figure 3.4: The pipeline for performing analysis on a dataset that results in predicted labels for data samples through a classification model.

Figure 3.4 shows the pipeline of creating a classification model that can be used for predicting unlabeled data. The original dataset needs to be preprocessed and transformed into a shape that is suitable for applying learning algorithms to. The data is then divided into separate parts depending on whether it can be used as control samples, i.e. data that contains predefined classes, or prediction samples, i.e. unlabeled data. The control samples are required to create the classification model while the prediction samples need to utilise the classification model for predicting labels. In the case of HCS extracted data, control samples and prediction samples often belong to the same dataset and it is also unknown how the user will divide them both. The target feature is therefore specifically chosen by the user and also which labels to use for training and prediction.

The control samples are further split into a training set and a test set, the former to train the model and the latter to test against and calculate a validation score describing the performance of the model. Note that a validation set used by e.g. cross validation methods, is not included in this pipeline but is part of the feature selection process and thus also part of the training dataset. The feature selection module performs feature selection for a dataset and uses the chosen features for creating a classification model. The model can then be applied on data that requires classification.

3.4.1 Preprocessing

Preprocessing is a step performed before the occurrence of any learning or filtering algorithms. It is a preparation process of the data that is required since the implemented feature selection and classification methods are in need of a specific format for the dataset.

At first, selected data are going to be extracted from the database. Including all data is possible for a single dataset but a manual filtering process can be made for only including the features considered. Since a dataset can contain both training and prediction data, the dataset is filtered to only include samples with a chosen label for a target feature.

The next step in the preprocessing is to format and clean the data. The algorithms used can for example, not handle text-related formats and therefore the features with string datatypes need to be reformatted. The approach for doing this is to create binary features for every unique value

in a string feature. The binary features show if a value is represented for a specific data sample or not. The amount of values per feature can however be of extremely large amounts if e.g. a unique value exist for each sample. This would create a great amount of new features and therefore a limit is chosen for how many new features a string feature can create. If it exceeds this value, the string feature cannot be represented and is thus neglected from the subset. The limit is arbitrarily chosen as 20 so a categorial feature can maximum create that many new boolean features. The reason for not creating a singular feature with unique integers matching unique strings is that such a case generates categories that appears to be ordered. Most of the time this is not desired since strings tend to not contain any information about the order.

Before the data can be used in machine learning methods, it also needs to be cleaned from missing values. The user can choose between a number of different imputation strategies for the purpose of creating a full dataset. Three of them consider filling up the dataset with aggregated data from the different features, e.g. mean, median or most frequent. These approaches calculate aggregated data from each feature and replace missing values with the resulting values. Another method is to fully exclude features that contain missing data. The the removal of data samples with missing data has not been considered, however, because of the amount of data that would be removed in such a case.

After the data has been manipulated by cleaning, formatting and imputation, it can be transformed to a better fitting shape. The approach handled in this step is primarily scaling. Before this step, there is no information of how the distribution is scaled or how this will affect the machine learning methods. Therefore each feature can be standardized to represent a normally distributed data with zero mean and unit variance. A min-max approach can also be selected to normalize the data between 0 and 1.

3.4.2 Creation of the Classification Model

The creation of the classification model is performed in a sequential pipeline with a first step of manually adjusting the settings of the different algorithms and information about the features. The chosen dataset is then preprocessed before entering the stage of feature selection and model training. Depending on the chosen approach, this step differs for the different algorithms, see fig. 2.10 in section 2.4. Features are selected and a model is trained based on these features. All information about this process is then stored. The model and information about what preprocessing that has occurred is stored as objects in files while information about the selected features is stored in arrays in the SciDB.

Three separate methods are implemented in order to perform feature selection, i.e. filter, wrapper and embedded, that behave in separate ways. Recursive Feature Selection is implemented as an embedded method, integrated as a part of the classification process where a feature is removed in each iteration. This method was chosen because of its ability of being incorporated in classification methods and let them be decisive of which feature to strip away in each step. As a wrapper method, an Exhaustive Feature Selection was implemented where a classification algorithm is used as an objective function that works external from the feature selection but generates quality measurements of the chosen subsets. This approach was chosen due to the ability of finding the best optimal subset since its approach is to search through all possible combinations. However, the execution of it requires a manual input of a maximum amount of features since the time complexity gets too large for extensive amounts of subsets. Robust Feature Selection is implemented as a filter approach occurring before the actual model training part. Statistical methods are applied to the dataset for filtering out a subset of features and the classifying model has no part of it. This method is however somewhat incomplete since it requires a generated variance matrix for the error model in the dataset and this is not always provided.

The methods selected as classification algorithms are Support Vector Classifier, Random Forest and Extremely Randomized Trees. These were selected based on a brief investigation of related literature to see which methods that provided best result and were often mentioned with good judgements. All of these are available in the scikit-learn library which also affected the selection.

A general class structure, see fig. 3.5, has been constructed to easy categorize different feature

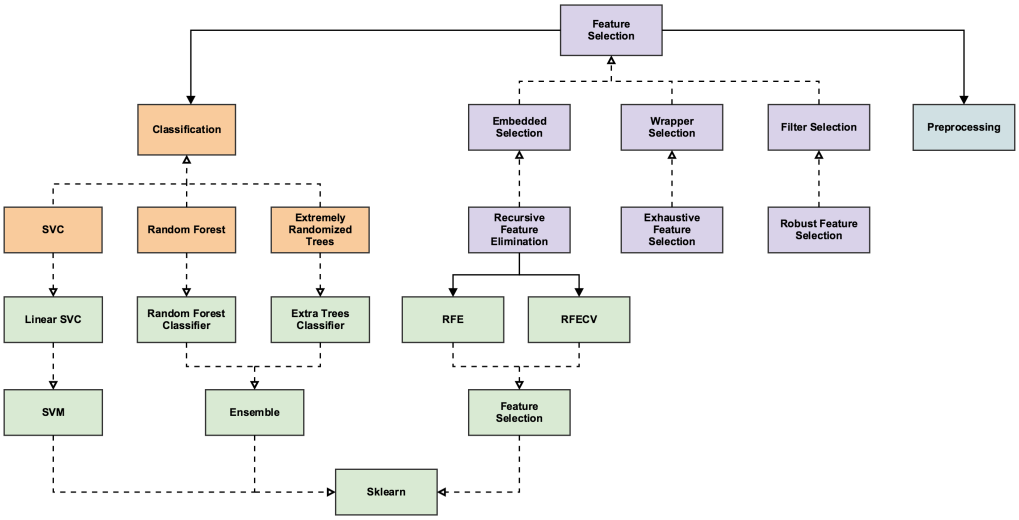


Figure 3.5: Low-level hierarchy of the feature selection, classification and preprocessing classes.

selection methods, couple them to different objective functions and extend the collection of the methods. Dashed line in the figure symbolizes that a class inherits from another while a full line embodies object representation of another class. The different types have been based on the categories of feature selection that were described in the Theory chapter, see section 2.4.

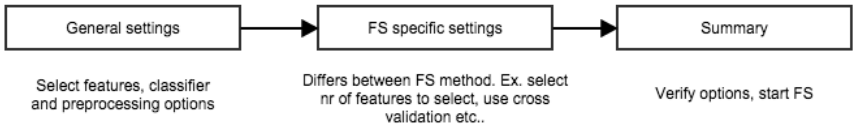


Figure 3.6: Different steps for feature selection to proceed for the user.

On the client side there are three different steps that the user will have to manually go through to perform a FS, see fig. 3.6. The first step is a general step for all FS methods which consists of selecting which feature to include in the process, which classifier to use as objective function and which imputation strategy to use to handle missing values and increase quality of the FS. The next step involves adjusting specific settings, like using cross validation and how big the set of selected features are going to be. The last step is also general for all methods like the first step and it is a confirmation step for the user where a summary of all performed settings is visible and an option to go back and change or start the FS is given. The different steps on the client side are also structured in a way that is easy to extend for new feature selection methods.

3.4.3 Prediction

The created classification models can be used for predicting unlabeled data. An important criteria is that the features that the models have been fitted with also exist in the dataset used for prediction. The user is thus provided with information about the performance of each model, the methods used and which features that are required. The target feature does not need to exist in the dataset and if it does, the prediction algorithm will filter out all samples that already have one of the chosen labels for the classification model. The procedure of prediction creates a new feature where each unlabeled sample gets predicted based on the classification model. This results in a feature, named by the user, which consists of predicted labels as well as original labels for a dataset containing both training and prediction samples.

3.5 Graphical User Interface

This section describes how the graphical user interface was established and how the result of the different data mining operation is visualised to the user.

The main concept was to design the GUI in a way that is familiar to the user, with a menu of options at the upper left corner and popover windows for editing options in the application. The requirement was to create an application that should work for desktop sized screen of different sizes and resolutions from laptop up to big desktop screens. Bootstrap was used to create a responsive layout of the application and many of the components from Bootstrap have been used such as buttons, glyphicons and popovers to provide a basic clean design.

In addition to designing the application as clean and natural for the user as possible, a goal was to make the user in control of every decision and action that is taken in the application. This was established by providing a control field every time an action has been performed in the application that will affect the data or the result. In the feature selection a summary of all settings is visible before starting the process and all other actions either provide a confirmation window for the action and/or provide a message in the status log, see fig 3.7.

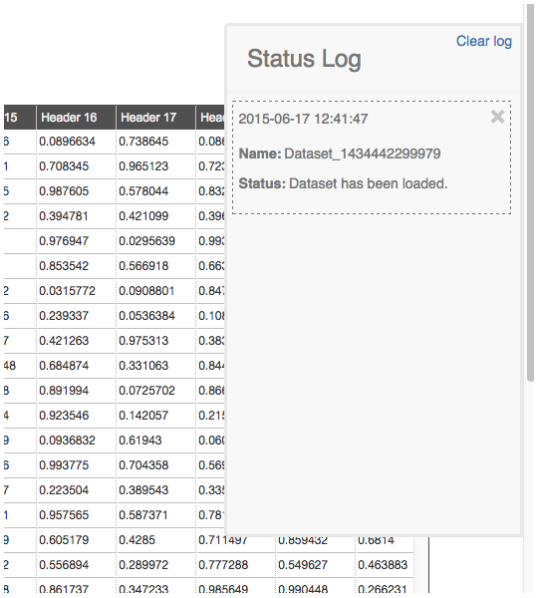


Figure 3.7: Status log for messages.

In the status log in fig. 3.7 all status messages stored are located at the right side of the window and only visible when hovering within 1 cm from the right side of the window. This design was decided to save as much space as possible for the grid but still have it easily accessable on the screen without having to navigate through any menu.

The grid in the current version of the application provides basic functionality of view and edit specific cells and reordering columns, see fig. 3.8. The grid is designed as a spreadsheet to provide a natural experience for our end user who usually works a lot in Excel [1].

The focus on providing the result from classification and feature selection has been to provide good export functionality to let the user use other software specialized for visualisation, see fig. 3.10. The result of feature selection can be seen first in the status log but in more description under the menu Analyze where prediction also can be used for created models, see fig. 3.9.

Since data mining is an scientific area which is not well known by our end user, help buttons, see fig 3.11, is provided on all options to inform the user of the purpose of the specific options and information about what they mean and if possible when to use what.

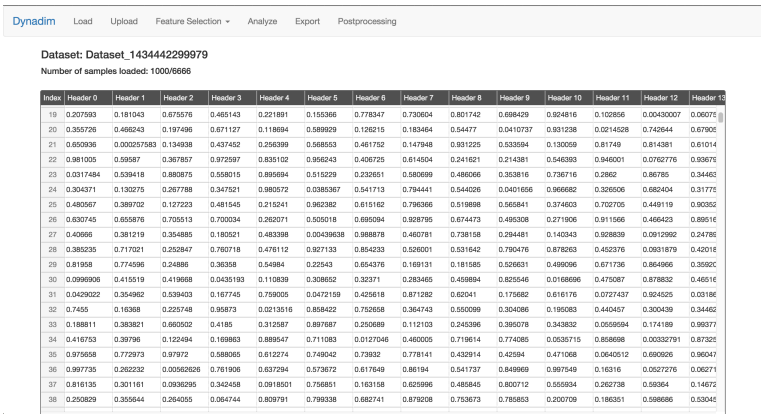


Figure 3.8: The data grid.

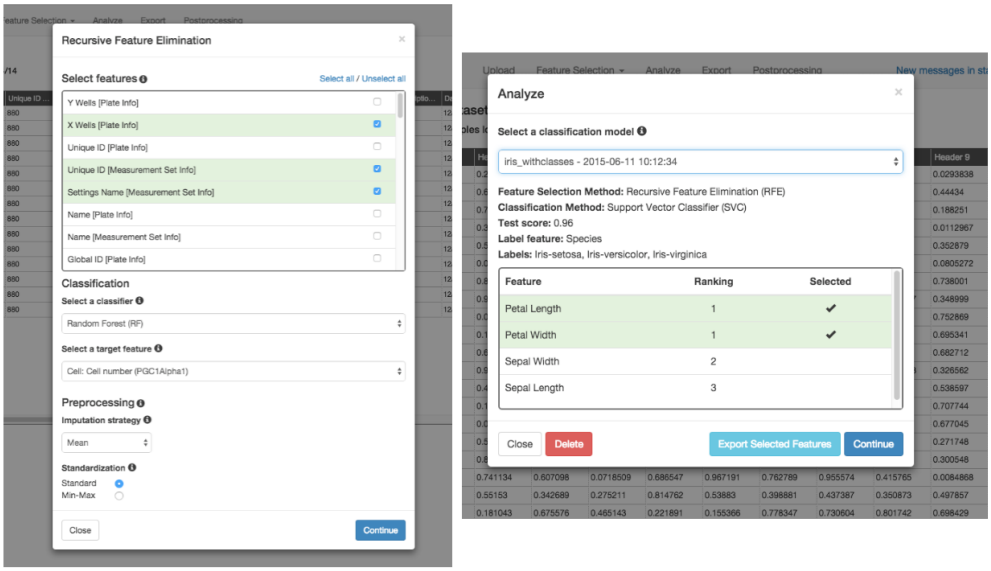


Figure 3.9: Feature selection settings modal (left) and the resulting classification modals in the Analyse modal (right).

3.5.1 Usability test

In order to ensure that the GUI was designed in a intuitive way that was natural for the user a usability test was conducted with the end user. The test was performed by letting the end user test the application and complete defined tasks under observation. The defined tasks are available in Appendix C. The test was a think aloud session performed as unsupervised as possible with only a few questions about the predefined use cases/tasks that were performed where the user described her opinion verbally about the application and the tasks. The tests resulted in that the status log in fig. 3.7 was hard to find for a new user which was solved by creating a clickable message in the top corner every time it has been updated which opened the log when clicking. Another result from the test was to separate the continue and action buttons from cancel, delete and back button in the options windows to separate the "positive" from the "negative" actions which was done by grouping them on each side of the window.

Other comments where positive and the user really liked the summary and messages of what settings and actions that are being performed.

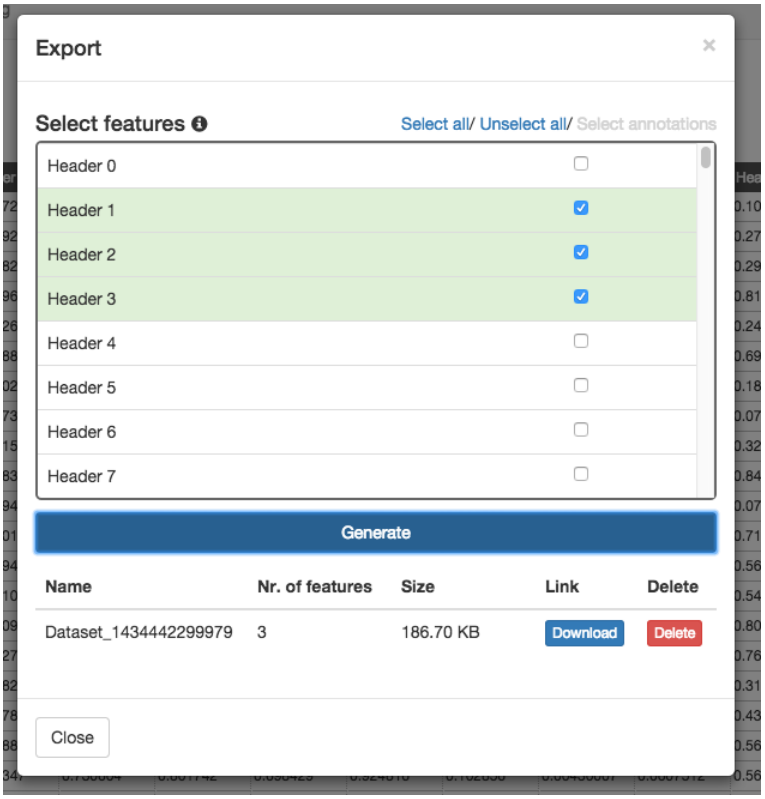


Figure 3.10: Export menu.

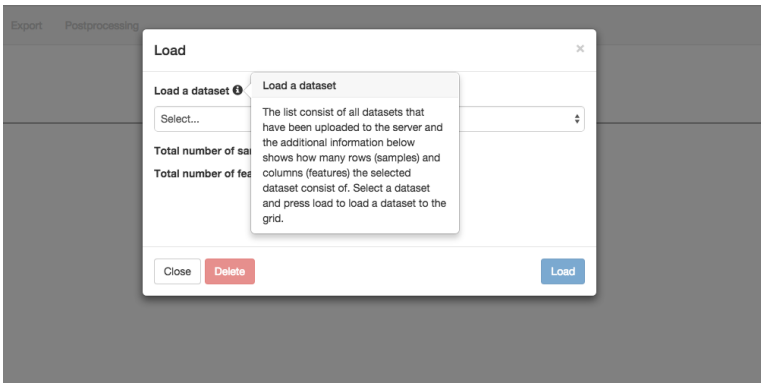


Figure 3.11: Information popup.

Chapter 4

Result

This chapter describes the resulting application and explains how it influences the workflow when performing HCS experiments. It also presents different types of measurements for calculating its performance.

4.1 The Application

This section describes the resulting application and how it complements the existing workflow.

The resulting application extends the manual workflow, see Appendix A, and provides a more automated way of performing analysis on multidimensional datasets with cellular level data samples. This section describes how the created application influences this workflow.

Figure 4.1 describes the resulting workflow that this thesis suggests. The created application is included and it shows how new features have been enabled and how old features have been improved. The new features include classification and feature selection, which can be performed on generated data to find relevant features and predict data to predefined classes. The improved functionality is mainly the ability to analyse multiple features at the same time which is made with help of the feature selection and machine learning algorithms. This was previously performed by an iterated manual analysis of one single feature at a time. Compare fig. 4.1 with fig. A.1 to see in picture how the application has affected the estimated workflow.

4.1.1 Data Preparation

To start using the application, a dataset is required, which can be uploaded from a file. The dataset shall be a CSV formatted file with any of the well known separators tab, comma or semicolon. The application provides two different ways for uploading and parsing datasets.

If the dataset is generated from MetaXpress an annotation file can be included within the uploading step, see fig. 4.2a. The annotation file is an Excel template which is explained in detail in Appendix E. If the dataset is not generated from MetaXpress, the data has to be in strict CSV format with first row as headers and the following rows with data.

When the file is uploaded to the server, the dataset can be found and loaded within the loading menu, see fig. 4.3a. The dataset will not be able to load before it has been uploaded completely, this information is visible in the status log which provides information of all action performed by the user, see figure 4.2b. The parsing phase utilises a background thread on the computer so that the GUI still can be used while uploading files.

When a dataset is loaded the features for a limited amount of the samples are visible in the grid, see fig. 4.3b. The cells in the grid can easily be manipulated by changing values. The menu

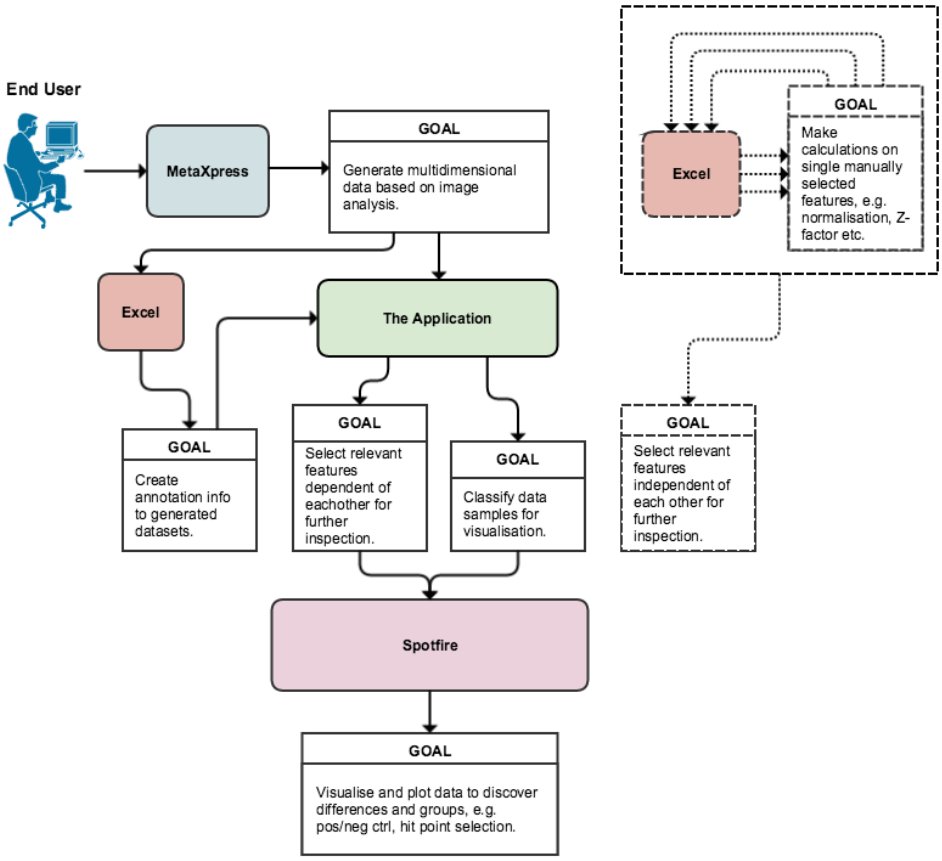
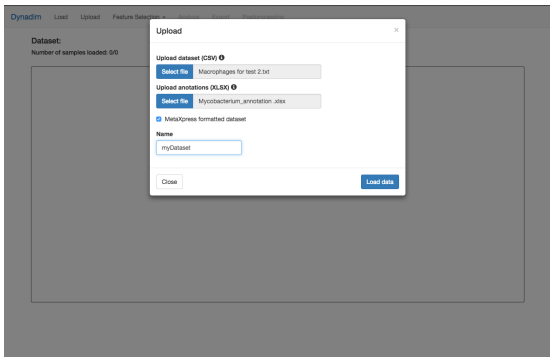
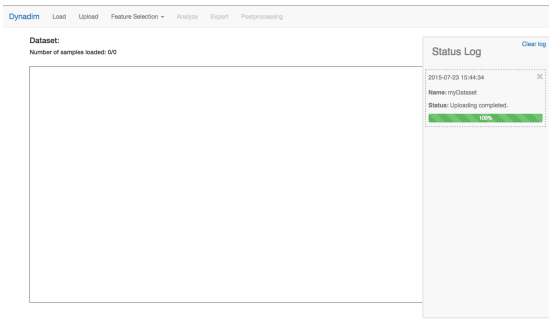


Figure 4.1: Proposed workflow with the new application involved.



(a) Select files to upload, parse and save in the database.



(b) You can follow the progress of parsing and saving into the database in the status log.

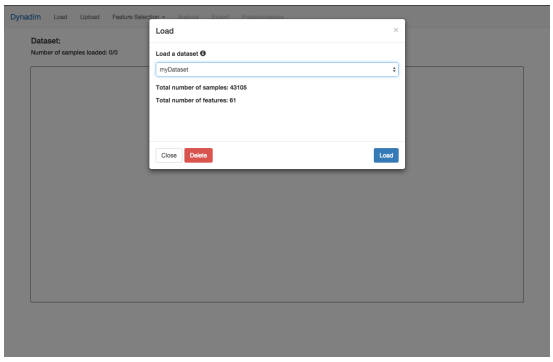
Figure 4.2: The uploading procedure.

options performing feature selection, prediction, exporting and features manipulation also becomes available when a dataset has been loaded.

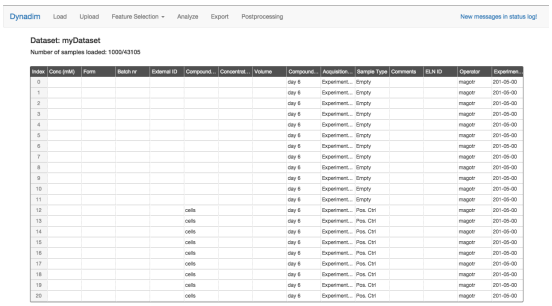
4.1.2 Feature Selection

The feature selection option is available in the top menu and offers three different feature selection algorithms, see fig. 4.4.

When a FS method is chosen you manually need to select which features to include, which classification algorithm to use for training the model and set some preprocessing options for how to



(a) All datasets that have been uploaded can be chosen here to load into the application for analysis.



(b) When a dataset has been loaded the grid gets updated with the data.

Figure 4.3: Loading a dataset

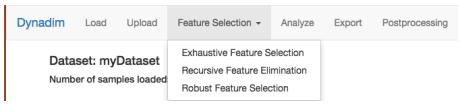


Figure 4.4: Alternative FS methods to choose between.

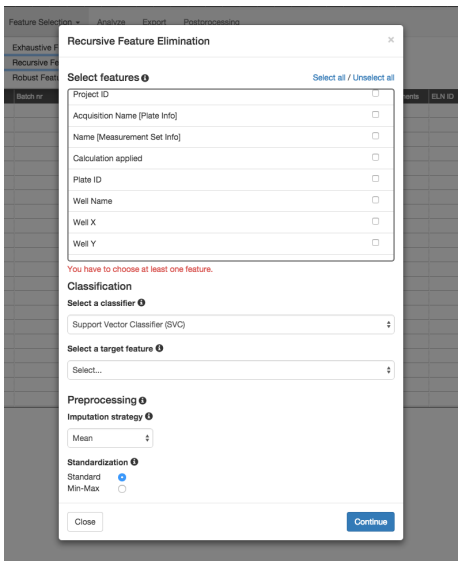
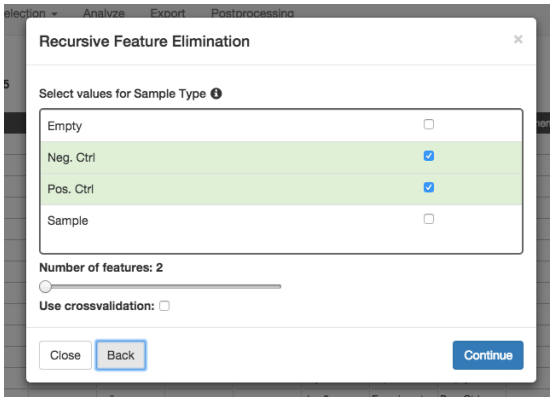


Figure 4.5: First step of settings for performing FS.

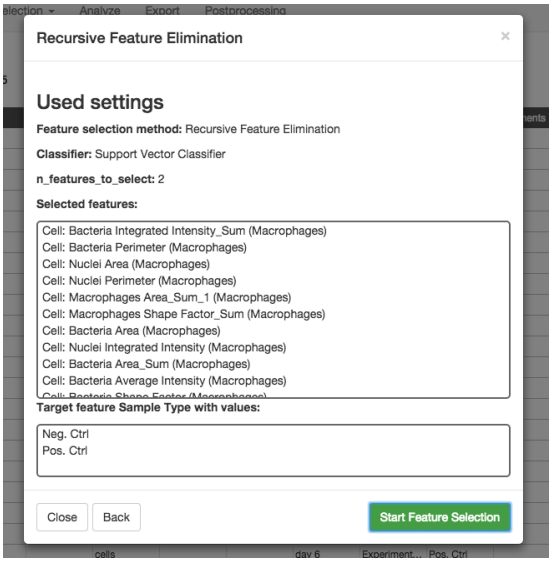
handle empty values and normalize data, see figure 4.5. The reason to involve a manual selection before training the model is to make the user involved in the process and always be in charge of what is happening within the application.

The next phase is to set the specific settings for the feature selection method that has been chosen. This can for example be the number of features that one wants the algorithm to select, or if crossvalidation should be used to decide the optimal number of features, see fig. 4.6a. A selection is also made based on what classes one wants to train the data with. A target feature is provided from the previous step and this step shows all possible labels within that feature, for the user to choose among. This makes the classification very flexible because the user decides what feature that is useful for grouping data samples into classes. The user also has the option of performing training and prediction on the same dataset without having to divide them manually.

The last step before starting the training is to confirm the settings that have been set in previous steps. All configurations are visible in a summary and if any option needs to be changed you



(a) Specific settings for FS method and selecting values for the labeled feature.



(b) Last steps with summary of all settings that have been set and button to start the FS.

Figure 4.6: Feature Selection, the final steps.

can press the back button for the ability of going backwards. When all settings are as preferred, the training can be started by pressing the "Start Feature Selection" button. When the feature

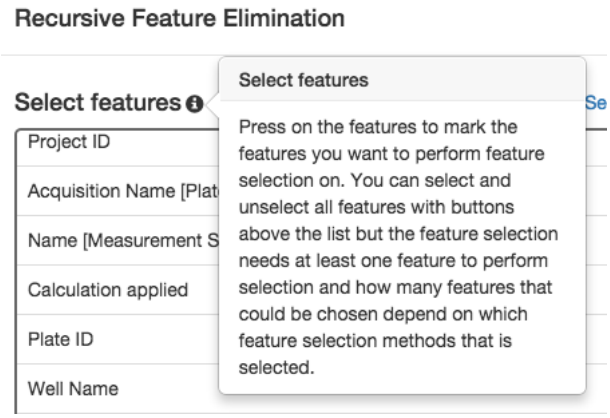


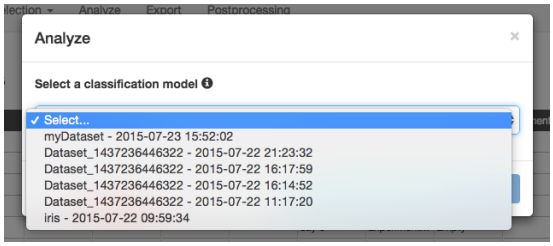
Figure 4.7: Popovers for information about the different settings.

selection is completed the status log will be updated with a message of the test score. To further ease the use of the application, popovers has been implemented at all places where configuration can be performed marked with an "i" sign, see fig. 4.7. They provide help for the user so that no option or configuration generate confusion.

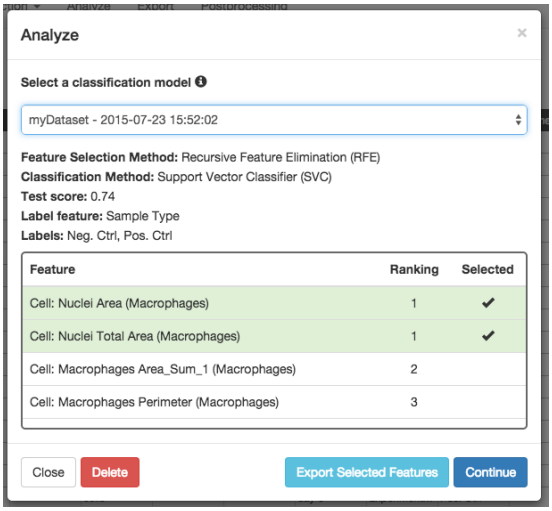
4.1.3 Analyze

The analyse section of the application provides functionality to predict classes based on the different classification models that have been created.

When entering this section, a list is given for all existing classification models based on their name together with a timestamp see fig. 4.8a. Detailed information will show for each classification model about which methods have been used, test score and the feature used for labeling samples. Also a ranking is included for features used in the training phase, see fig. 4.8b. When proceeding with the Analyze section, the classification model chosen will be used for predicting new labels for samples in the dataset. A new feature will be created with the new predicted labels,



(a) All classification models is available here for further analysis.



(b) When choosing a classification model, information about the performed FS is visible below with alternative proceedings.

Figure 4.8: Analyze modal.

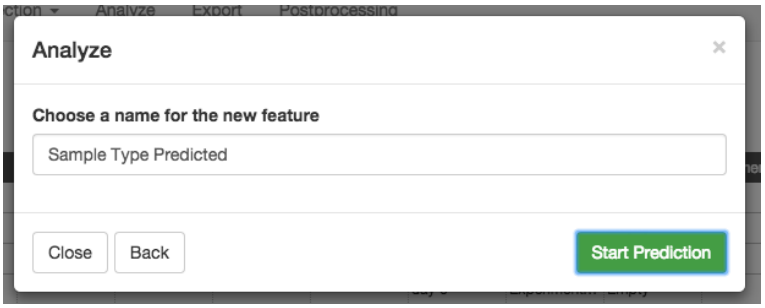


Figure 4.9: When a prediction is conducted, a new feature is created with the predicted data.

see fig. 4.9.

4.1.4 Export

All data that has been saved in the database can be exported to a file with CSV format in the menu Export.

The feature that are to be exported shall be selected in this menu, generated to a file, and then downloaded locally, see fig. 4.10. This yields the usage of other visualisation tools that can be used to view results from the analysis steps performed in this application.

4.1.5 Feature Processing

A section of the application also exists for performing feature manipulation of the data loaded into the database.

If some features contain a lot of empty values for some reason, these values can be filled by utilising one of the methods provided in this section, see fig. 4.11. For example some features extracted from MetaXpress only supply data samples at an image level, so these features need manipulation in order to make a correct representation in the machine learning algorithms.

For this purpose a method is provided for filling data samples with the closest value above in the dataset. The other method instead fills empty values with a mean value of all existing values

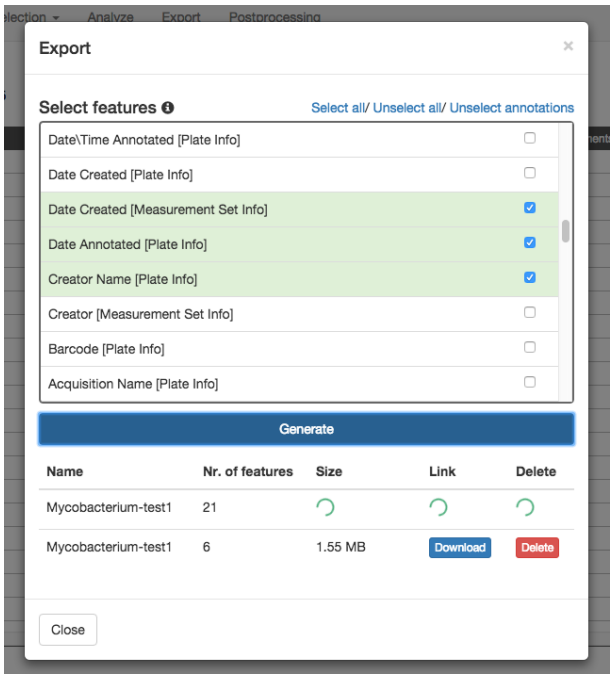


Figure 4.10: Export selected features to CSV format.

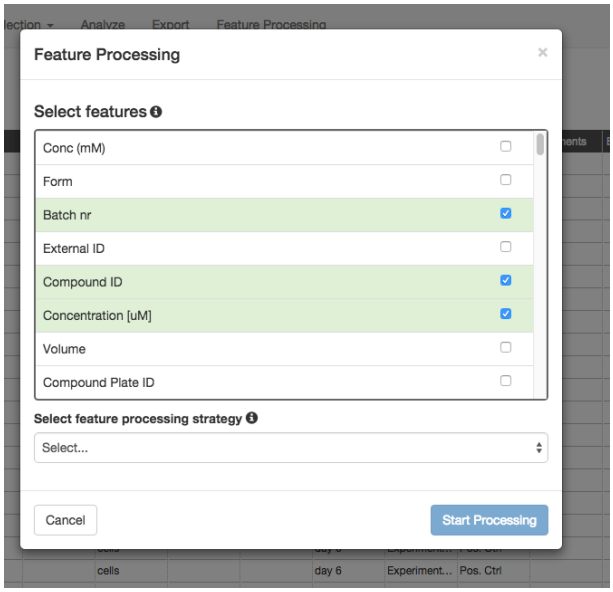


Figure 4.11: The feature processing modal.

in the feature. The latter method utilises the functionality of performing calculations directly in SciDB and is thus somewhat faster then the former.

4.1.6 Summary

To summarise the workflow of the application, a dataset is uploaded from a file format and stored into the application. The user can then increase the quality of the data by processing it manually in the grid or with the option to perform automatic feature processing through a set of predefined methods. The user can then perform feature selection for creating a classification model and to extract the most important features. The relevant features can be exported together with other manually chosen features for the purpose of performing visualisation and further exploration in other software. If the user wants samples to be predicted in a new feature, this is also provided by the classification models that have been created. This flow of decisions is visualised in fig. 4.12.

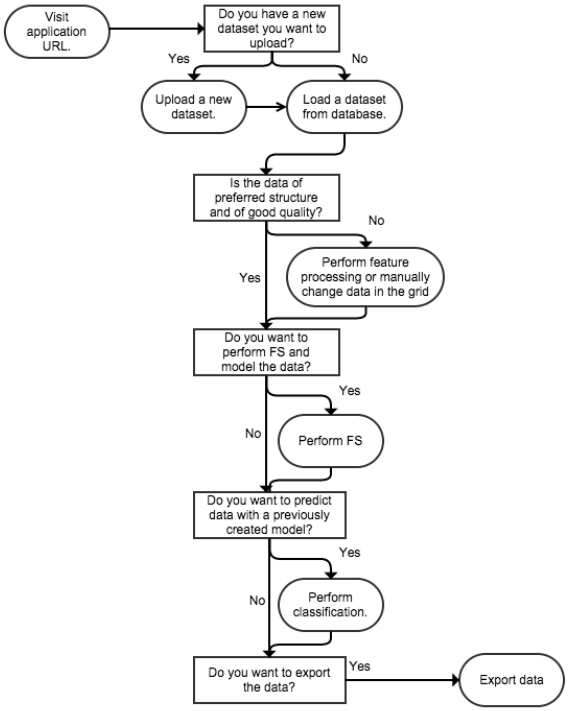


Figure 4.12: Typical workflow for usage of the application.

4.2 Data Uploading Performance

This section includes benchmarking results by measuring the speed of uploading data.

The performance of the uploading of datasets has been tested by measuring the duration of uploading datasets of different sizes. All tests have been performed on a Ethernet connection (at the date 2015-07-20) with a measured speed of : 212 Mbit/s / 244 Mbit/s (send files / receive files). The files that have been tested are in strict csv with 100 columns and various amount of rows depending on file size. All upload has been performed on a DigitalOcean cloud server with 2 GB RAM and a 1 core processor.

File size	**Duration (hh:mm:ss)**
1 MB	00:00:03.6
5 MB	00:00:07.5
10 MB	00:00:10.7
50 MB	00:00:27.5
100 MB	00:00:47.2
500 MB	00:04:12
1 GB	00:08:07
2 GB	00:14:50
5 GB	00:38:00

Table 4.1: Table of collected data from uploading benchmark

Figure 4.13 shows a visualisation of the data collected from the uploading phase, see table 4.1. It shows a linear relationship between duration and size of dataset. From this plot an average velocity of 2.03 MB per second can be used to predict that 10GB of CSV would take less than 90 minutes.

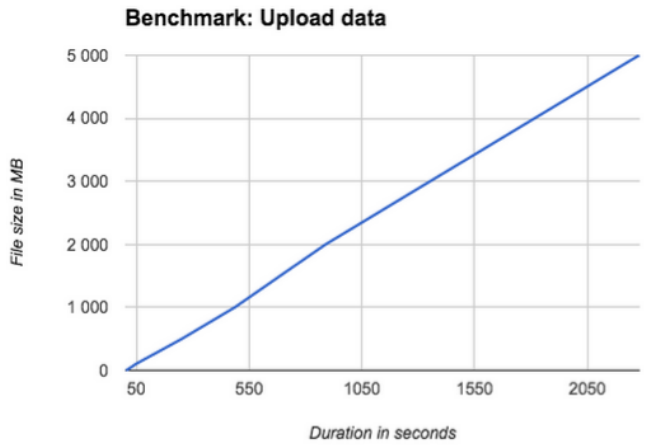


Figure 4.13: A line graph showing the duration growth for uploading CSV when increasing file size with data from table 4.1.

4.3 Feature Selection and Classification

This section covers the test results on two different datasets, one result from the well known Iris data set and one result from a real HCS dataset.

4.3.1 Test Data

The Iris dataset, see more in Appendix D, was used for testing the application and see what it accomplished on a commonly used classification problem.

The results in tables 4.2 and 4.3 include a variety of runs to provide a measurement of how the different algorithms perform on a commonly used classification problem. Both the tables include 5 test runs per setting, i.e. a specific feature selection and classification method. The number of test runs is selected based on the fact that each specific setting does not yield classification models that are identical. To recieve resulting values that tell a bit more than a single run, but within a reasonable amount of time, that number is set to 5.

Table 4.2 tests the different feature selection algorithms available while table 4.3 has manually chosen features for all the runs. The reason for the manually selected features is to prove a point by comparing feature selection with how the outcome can be if the wrong features are chosen. The test score is calculated as a mean accuracy of the predicted labels with a range $0-1$ where 0 means that no samples were predicted correct while 1 has 100% correct prediction results. As can be seen in table 4.2, the three different feature selection methods perform somewhat different from each other. The other feature selection algorithm that has been implemented, Robust feature selection, has been omitted (read more under section 5.3.2).

The ERT algorithm looked for the best possible combination of all features. The resulting features were spread through a lot of different combinations. There are only two combinations of features that never occur for the ERT algorithm and that is “Sepal Length” - “Sepal Width” and “Sepal Width” - “Petal Width”. The scores for ERT have a min value of 0.83, max value of 1.00 and a mean value of 0.92.

The number of features selected for the RFE algorithm was set to 2 since it is a number that reduce the number of features with half. The use of 2 features can also be useful when describing samples in 2D plots. However, this number is up to the user to choose for this algorithm. The result was consistent for the selected features in all runs with the occurrence of only one combination. The scores for RFE have a min value of 0.90, max value of 1.00 and a mean value of 0.94.

The RFECV algorithm looked for the most optimal subset of features based on cross-validation

Method	Classifier	Score	Sepal Length	Sepal Width	Petal Length	Petal Width
EFS	SVC	0.83				
EFS	SVC	0.93				
EFS	SVC	0.96				
EFS	SVC	0.96				
EFS	SVC	1.00				
EFS	RF	0.86				
EFS	RF	0.90				
EFS	RF	0.90				
EFS	RF	0.86				
EFS	RF	0.93				
EFS	ERT	0.96				
EFS	ERT	0.96				
EFS	ERT	0.96				
EFS	ERT	0.86				
EFS	ERT	0.96				
RFE	SVC	0.93				
RFE	SVC	0.96				
RFE	SVC	0.93				
RFE	SVC	0.93				
RFE	SVC	0.90				
RFE	RF	0.90				
RFE	RF	0.90				
RFE	RF	1.00				
RFE	RF	0.96				
RFE	RF	0.93				
RFE	ERT	0.96				
RFE	ERT	0.96				
RFE	ERT	0.90				
RFE	ERT	0.96				
RFE	ERT	1.00				
RFECV	SVC	0.96				
RFECV	SVC	0.96				
RFECV	SVC	0.96				
RFECV	SVC	0.96				
RFECV	SVC	0.96				
RFECV	RF	0.90				
RFECV	RF	0.93				
RFECV	RF	0.93				
RFECV	RF	0.93				
RFECV	RF	0.96				
RFECV	ERT	0.93				
RFECV	ERT	1.00				
RFECV	ERT	0.93				
RFECV	ERT	0.96				
RFECV	ERT	0.96				

Table 4.2: Feature selection and classification test score of the Iris dataset. 5 test runs are made per setting. Methods used: EFS - Exhaustive Feature Selection, RFE - Recursive Feature Elimination, RFECV - Recursive Feature Elimination Cross Validation. Classifiers used: SVC - Support Vector Classifier, RF - Random Forest, ERT - Extremely Randomized Trees. Green color represent that the feature has been selected by the algorithm while red color represent unselected feature.

and in some runs it found more optimal solutions by incorporating more than two features. The scores for RFECV has a min value of 0.90, max value of 1.00 and a mean value of 0.95.

Method	Classifier	Score	Sepal Length	Sepal Width	Petal Length	Petal Width
WORST	SVC	0.76				
WORST	SVC	0.76				
WORST	SVC	0.73				
WORST	SVC	0.76				
WORST	SVC	0.86				
WORST	RF	0.60				
WORST	RF	0.76				
WORST	RF	0.66				
WORST	RF	0.73				
WORST	RF	0.73				
WORST	ERT	0.66				
WORST	ERT	0.76				
WORST	ERT	0.66				
WORST	ERT	0.73				
WORST	ERT	0.80				
ALL	SVC	0.90				
ALL	SVC	0.90				
ALL	SVC	0.80				
ALL	SVC	0.96				
ALL	SVC	0.96				
ALL	RF	0.93				
ALL	RF	0.96				
ALL	RF	0.96				
ALL	RF	0.96				
ALL	RF	1.00				
ALL	ERT	0.90				
ALL	ERT	0.93				
ALL	ERT	0.96				
ALL	ERT	0.93				
ALL	ERT	0.93				

Table 4.3: Classification test score of the Iris dataset with feature selected manually. 5 test runs are made per setting. Methods used: WORST - The features selected are assumed to result in the worst possible test score, ALL - All features are selected. Classifiers used: SVC - Support Vector Classifier, RF - Random Forest, ERT - Extremely Randomized Trees. Green color represent that the feature has been selected while red color represent unselected feature.

Table 4.3 contains data for test runs on manually selected features. Two different settings are tested. The case "WORST" relates to that the features selected are the 2 that are assumed to perform the worst in a classification problem. This finding is based on table D.1 and fig. D.1 in Appendix D where one can see that these features contain the lowest correlation coefficients and behave in unstructured ways when using a scatter plot. The score for WORST has a min value of 0.60, max value of 0.86 and a mean value of 0.73. The "ALL" setting has all the features selected to see how the performance would have been without feature selection. It has a min value of 0.80, max value of 1.00 and a mean value of 0.93.

The scatter plots in fig. 4.14 - 4.16 visualise the prediction result of 3 different classification models. In fig. 4.14, the features are manually selected and one can see that two of the Iris species (Iris Setosa and Iris Virginica) are very hard to separate linearly which results in that a lot of predictions fail. Figure 4.15 shows features selected by RFE algorithm with a SVC classifier and unlike the last picture, it proves that the samples are better separated with the use of the selected features. Figure 4.16 shows that even better success rate can be given by utilising another classifier, in this case an ERT-algorithm.

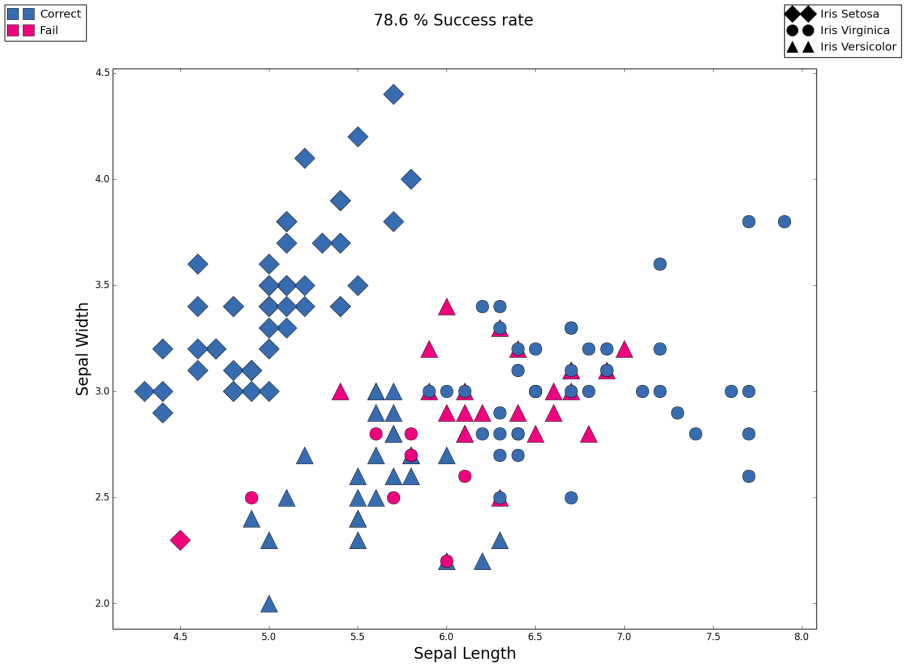


Figure 4.14: A scatter plot of the prediction results from a model created by an SVC-algorithm (Support Vector Classifier) and two manually chosen features that are assumed as the worst for describing the Iris dataset.

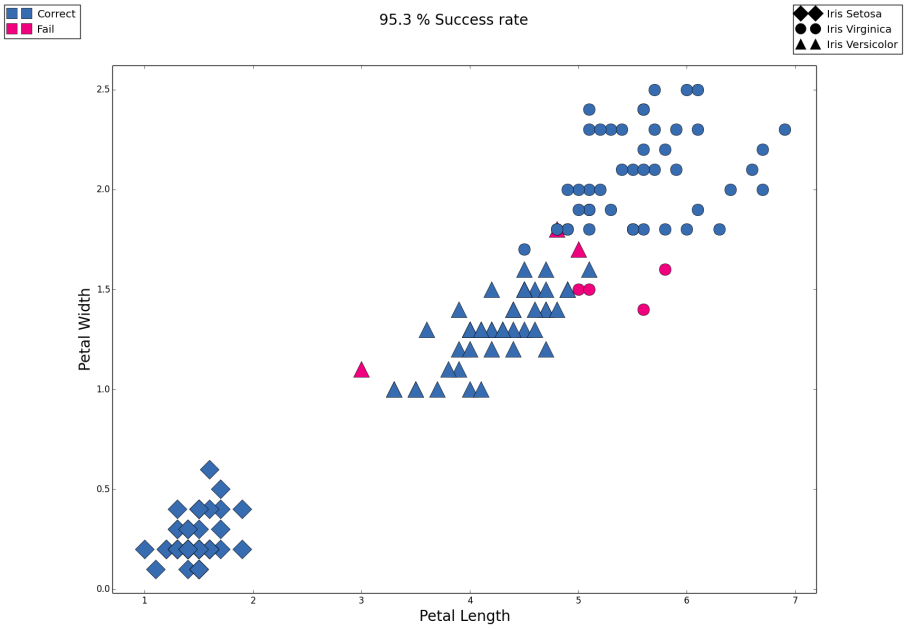


Figure 4.15: A scatter plot of the prediction results from a model created by an SVC-algorithm (Support Vector Classifier) together with two features selected by RFE (Recursive Feature Elimination).

4.3.2 Case Study

The goal with the presented case study is to demonstrate an improvement in using the application implemented within this thesis in comparison with the manual analysis methods. The data used in this study is generated during development of a screening assay for a project performed in collaboration between CBCS and Maria Lerm laboratory at Linköping University. The aim of the project

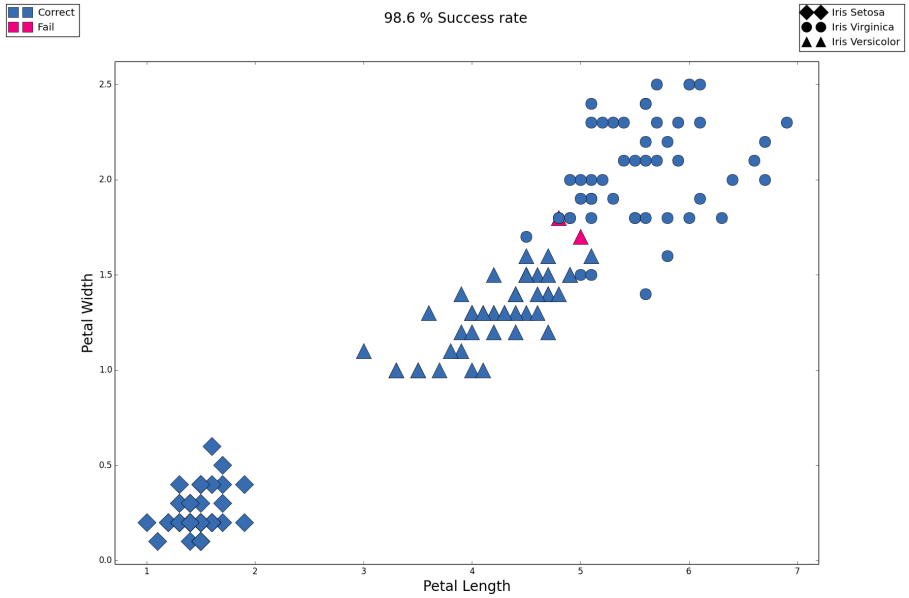


Figure 4.16: A scatter plot of the prediction results from a model created by an ERT-algorithm (Extremely Randomized Trees) together with two features selected by RFE (Recursive Feature Elimination).

is to identify compounds that can prevent the intracellular pathogen[1] *Mycobacterium tuberculois* from causing damage to the macrophages, which are important cells in the human immune system.

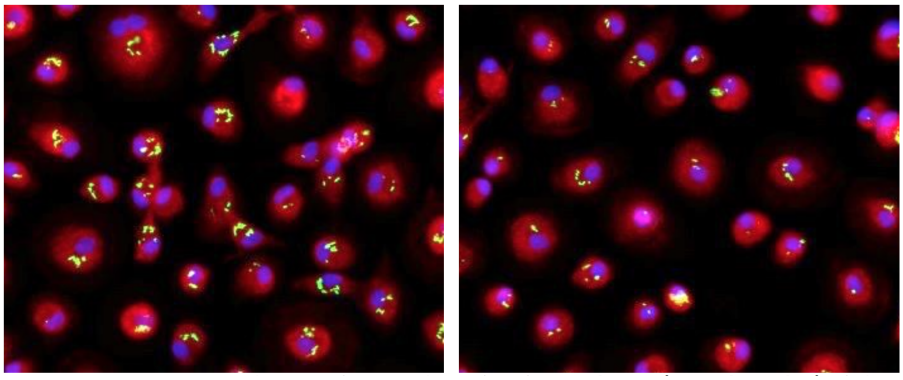


Figure 4.17: Images from ImageXpress of macrophages infected with *Mycobacterium tuberculosis* (left) and after treatment with *drug1* (right). Green areas are bacteria, red areas are macrophages and blue areas show the cell nuclei.

The experiment includes treatment with different known antibacterial drugs to explore their ability to inhibit the growth of *Mycobacterium tuberculosis* in macrophages. The images of the cells were taken with ImageXpress, which is an microscope for performing automated screenings. The bacteria in the macrophages were then identified and quantified by using the image analysis software MetaXpress, see fig. 4.17. In total, 34 different features were extracted from the image analysis and further data analysis was required to be able to identify and select the features that had the best description of the desired phenomenon. To make a comparison, the data analysis was performed both manually and with the workflow proposed in this thesis.

The manual workflow of performing data analysis started with extracting 34 features in a well based format. The well based format provides samples per well with mean values of all cells in the

[1]A pathogen can be defined as anything that can cause a disease, e.g. virus, bacterium and parasites.

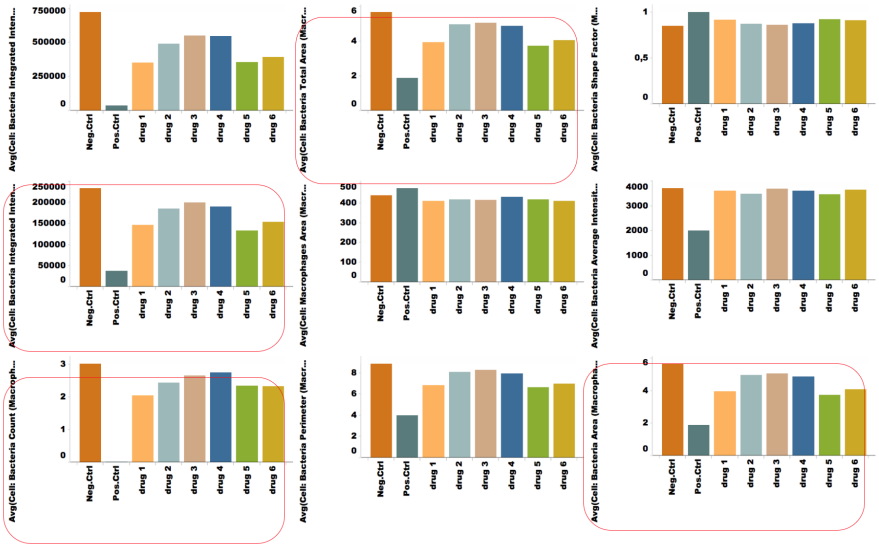


Figure 4.18: Visualisation from Spotfire of a limited number of manually selected features. The bars represent the mean value of the number of replicates for Neg. Ctrl (infected cells), Pos. Ctrl (non-infected cells) and cells infected and treated with different drugs.

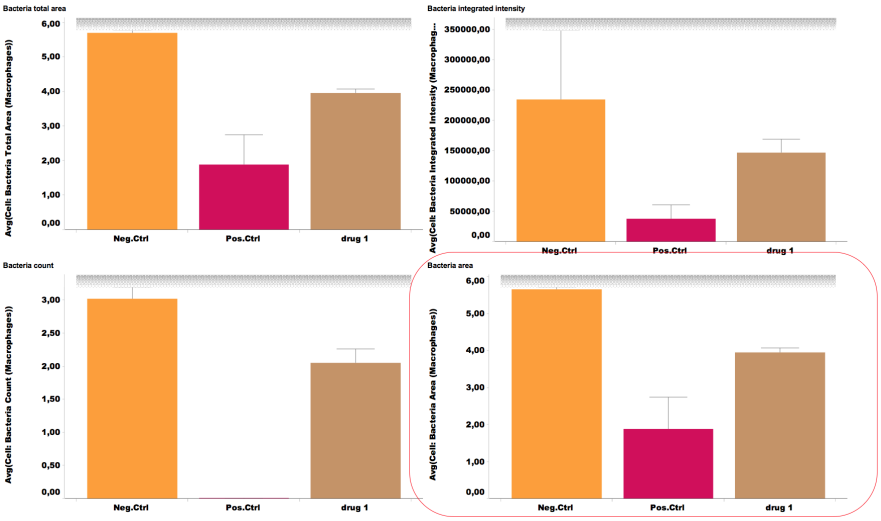


Figure 4.19: Visualisation from Spotfire of four manually selected features. The bars represent the mean value of the number of replicates for Neg. Ctrl (infected cells), Pos. Ctrl (non-infected cells) and cells infected and treated with single drug together with a standard deviation measurement.

well. The data were extracted to a text file that was then processed in Excel where it was manually annotated. A preliminary review was also performed in Excel to select a limited number of features that were used for plotting in Spotfire. The visualisation in Spotfire, see fig. 4.18, provided functionality for comparing the selected features by showing relations between control samples and samples treated with drugs. The highlighted graphs in the figure show features selected for further analysis (*Bacteria Integrated Intensity*, *Bacteria Count*, *Bacteria Total Area* and *Bacteria Area*). Figure 4.19 shows the four selected features and the best separation of infected and non-infected control samples were noticed for *Bacteria Count*. The feature *Bacteria Area* was however selected as the best feature to identify an inhibitory activity because it has the highest window between mean values together with the lowest variability between the samples.

In the case of using the application developed in this thesis, a more automated selection of features can be conducted. The inital step of the more automated workflow was to extract 34 features to a text file. An advantage with using support of the application is that a lot more data can

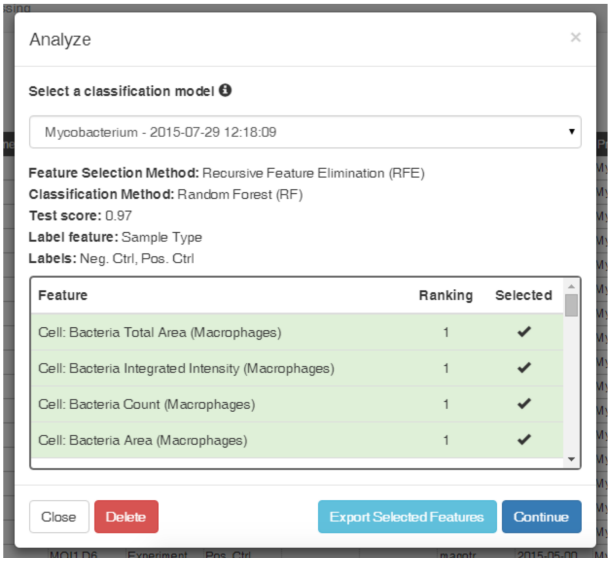


Figure 4.20: The results of the feature selection performed using the implemented application.

be handled in the analysis phase which means that samples can be extracted at a cellular level instead of mean values per well. The text file is then imported to the application together with a compiled annotation file. Feature selection was performed using a Recursive Feature Selection algorithm with a Random Forest classifier and samples with positive and negative control were used as training data. The same four features as in the previous manual analysis were selected, see fig. 4.20. The features were then exported from the application and imported into Spotfire for further examination.

Chapter 5

Discussion and Conclusion

This chapter will discuss how the problems presented in section 1.2 have been answered and highlight problems that have been crucial in the process of solving those problems within the subjects of data mining, feature selection and bioinformatics.

5.1 The Application

To map which needs the application to be developed should cover, an investigation of the workflow was conducted with the end user before this application was created. A survey of the current workflow with existing functionality and non-existing, but wanted, functionality was made. When the application was completed, a new survey with the new functionalities together with the previous was established. When comparing the surveys in figures A.1 and 4.1, there is a difference in that the application has replaced the manual ways of performing data analysis. The new approach offers methods of performing multidimensional analysis simultaneously, and also on a cellular level, which could not be made with the old workflow. The comparison can be summarized as that the wanted functionality is established and some of the previously existing functionality has been improved. The outcome of the application can be considered to complement existing analysis methods and improve the workflow with more automated tasks.

5.1.1 Future Work

This application is the first step towards an application that will hopefully grow in functionality and quality to provide even more support for performing analysis within molecular biology. The application consists of several steps and settings to configure when performing the analysis. To make this in an even more automatic manner, configuration templates could be an interesting extention to create and save settings of the analysis process. This would render in a more customized interface for the user and probably save some time using the application. Other functionality that could be performed as future work are discussed in the following subsections.

5.2 Data Management

The question about how to manage large amount of data in a robust manner has partly been solved by enabling cloud computing and by building a web application, since this is easily scalable and puts no requirement on the client. The procedure when uploading data “chunk-by-chunk” puts no limit on how much data that the uploading functionality could handle. The choice of using SciDB as database management system, with an array based structure, provides fast access to parts of data in big datasets. The uploading phase has been tested and an approximation can be made, based on the consolidated data, that a file of 10 GB can be parsed, transferred and stored in less than 90 minutes. That can be considered as a reasonable amount of time. The question of how to manage large amounts of data has many different solutions depending on which database to use and how to process the data that is stored. We are satisfied with the solution of using SciDB which seems to be a good fit for this type of data, but improvements could be made here which is described in future work. This improvement was not implemented during this thesis due to lack of experience in this system, lack of support from community and documentation in order to solve

the problems, together with the lack of time.

Another part of the data management was the parsing of files, into which much effort has been put into. The parsing of files, which were generated from MetaXpress, followed a seemingly weird structure. This generated a requirement of a customized parser that also could match the annotated data, which made this task rather complex. The parsing of files of this type is fully implemented together with parsing of files which follow strict CSV structures and this satisfies the requirements that had been set up. For providing parsing support other types of file structure, a more advanced parser is required which can anticipate how the supposed data is structured.

5.2.1 Future Work

Out-of-memory computations
SciDB is mainly used for its basic functionality like create, read, update and delete arrays. Only some calculations for feature processing are made directly in the database, e.g. there is well provided functionality for computing the mean value of a feature and this can be used for filling empty values in an array. If all calculations in the application were made in SciDB, this would remove the dependence of RAM size since heavy calculations would be performed out-of-memory and could be a good improvement to investigate in the future.

Data parsing
The file structure of different datasets can be very complex and often differs between sources. This opens a task for investigating the problem of how to build a general parser which can interpret and understand the internal structure of the file before parsing. This task would be rather large and could be a thesis in itself to investigate.

5.3 Feature Selection

Three feature selection methods were implemented. Exhaustive feature selection was implemented because it tests all combinations of features and is the only method that will provide the best possible subset and can be good to compare with other selection methods. The recursive feature elimination was chosen because of its popularity in literature [21] and other software [43]. The robust feature selection was chosen for experimental reasons and has promising ability of coping with errors in the dataset [15]. This way of providing robustness in the selection of features was not found in other methods. The experimental part was due to that it has never been implemented in a real application before.

As can be seen in section 4.3.1, the feature selection algorithms work well for selecting relevant features for the test data. Conclusions can also be drawn from section 4.3.2 which provides results of a real experiment. It is given as an example where relevant features is required to be selected to be able to describe if an output is good or bad. The presented example obtains a full overlap between manual analysis and feature selection techniques which indicates that the implemented application can be used to address biological questions. The application significantly simplifies the workflow of conducting analysis by eliminating most of the manual steps. For example the step of evaluating single features in Excel is replaced by the implemented feature selection algorithms. The outcome of this can be considered to be time saving as well as reducing the possibilities of human mistakes during the analysis process. This is an improvement, since it becomes easier to miss potientially important features when analysis is performed manually for large and complex datasets.

5.3.1 Preprocessing

Preprocessing is a vital part of machine learning algorithms, since it affects the actual outcome. In this thesis, some techniques have been implemented for the purpose of enhancing the data by transformation and manipulation of the different features. The decision of letting the user decide settings for some of these methods is also important because the needs for different datasets may differ. The most important preprocessing that can be made for HCS extracted data is how to handle empty values, i.e. data with incomplete values (null values), since this can be common.

The application offers multiple different methods for filling empty values. One such, that is not included, is the ability of removing data samples which contain missing values. Removal of whole features with missing data exists but for a dataset with a few missing data points, sample reduction would probably be better suited.

Conversion of nominal string values is another preprocessing that does not exist within the application. The only provided option is to convert unique string values to binary features. The reason for this is that nominal string values rarely exist and was thus not a priority. Another component that does not exist, but can be very important for enhancing data, is the removal of outliers. Biological data have a tendency to create outliers, i.e. samples that are far away from other samples. This can have an impact when scaling the data and a method for handling outlier removal would probably contribute to better quality of the outcome. If it proves that the outliers are relevant to look at, the user would probably want insight in these samples, and in such case a technique that separates outliers from the other samples would be eligible.

5.3.2 Robust Feature Selection

The resulting application consists of 3 different feature selection algorithms, one of which is Robust feature selection. This technique is implemented but it is not fully functional. Therefore, it has not been tested. The reason for this is that it, requires a variance matrix based on the measurement errors for the different features, besides the actual dataset. The variance matrix is used for creating uncertainty sets for all features. If no such matrix can be provided, then no robustness is achieved in the selection step.

All datasets containing measured data also consist of measurement errors. However they are very rarely provided with a model describing these errors. When working with HCS data, it becomes even harder to provide an error model due to the fact that multiple analysis and data acquisition steps occur before the actual data analysis. The data provided is also based on biological experiments, which can be affected by many unknown parameters during the assay development.

A possible solution to the problem described above is to provide specific tools for calculating statistic parameters, for each feature, so that the user can compute an estimation of the error model. An example is to create histograms for every feature and let the user select cutoff values that can estimate the variances. A created error model that is conservative will be useful in practice and give a robust selection of features.

5.3.3 Future Work

Enabling at client side to upload a variance matrix when performing Robust feature selection and enable calculating such an matrix would be a feature to implement in the future for the purpose of enabling the user to use and test Robust feature selection. Some further implementation is also necessary for more developed methods to process the data, e.g. dealing with outliers and more extensive methods for coping with empty values.

5.4 Classification

The classification algorithms that are implemented perform well on known datasets, where conclusions easily could be drawn, but with HCS data the result is hard to analyse. Since the datasets are large and consist of unknown and complex data in a biological manner, which we have almost no experience in, it is up to the end user to answer if the classifiers result in any useful information for the HCS data. We can establish results by computing test scores and with these results, all the classifiers perform well on the HCS datasets.

Results of feature selection and classification can be redundant and misleading. The high-content screening consists of many steps where each step can affect the resulting data and the outcome of the data analysis phase is highly depedent on the preceding steps. It is crucial that no preceding step is error-prone. For example the assay development puts the creator of the assays and the

instruments in charge of the quality and the following image processing is dependent on the performance of the algorithms used in MetaXpress.

Biological data does not have to consist of statistical relationships between features, which makes it important to say that this is only a complement for the researchers in their work. The process of creating a mathematical model to simulate the characteristics of real data can be seen as more art than science, which yields the usage of having several methods for classification and feature selection in order to compare their results with different datasets.

5.4.1 Future Work

This thesis included only supervised learning for the classification problem. Other learning problems such as clustering and regression were therefore neglected but can still be of importance in terms of analysis and should be stated for future work. Since no almighty algorithm exists for perfect classification different algorithms exist which have different pros and cons. That brings the need of including more classification algorithms that could be tested to see how they perform on HCS data. An example can be to investigate the possibilities of using genetic algorithms, which differs in behaviour from the implemented methods in this thesis.

5.5 User Interface

This section discusses how to design a system to put the user in the centre of every decision that is made and how to create intuitive feedback when actions are performed. The application has been designed to focus on making the analysis steps as user friendly as possible by following an architecture of how the methods have been implemented, where different level of data is filled in and a summary of the settings is provided before starting an analysis. All actions together with result from the actions are present in the status log which is always accessible in the application. This solution was assessed as a good solution according to the usability test performed with the end user. The focus has been on enabling as good data analysis as possible, extract results, and provide export functionality together with the requirement of making an application that is easy to use. This yielded the decision of making a user interface as clean as possible with as few input options as possible on the screen simultaneously.

5.5.1 Future Work

Instead of having menus for performing the different calculations and algorithms, the grid (see fig. 3.8) could be developed to perform more tasks in a more effective way. The grid currently implemented is rather unstable and implemented with a library which restricts the possibility of what can be performed with the grid. A future task could be to create a more interactive grid where more manipulations could be performed directly from the grid, e.g. manual processing of features and filtering options on different level of data. As a suggestion, a React component can be created for this task to take advantage of the benefits with its virtual DOM rendering. This could possibly handle a large amount of data in an effective manner.

5.6 Conclusion

This thesis investigates how the use of software and machine learning algorithms could provide a more automated workflow of conducting data analysis within high-content screening for drug discovery. This problem is particularly relevant in the context of bioinformatics. The resulting outcome is a web application made for supporting experts in molecular biology by selecting relevant features in a multidimensional dataset that can be of importance in the analysis process. Data samples can also be classified for the purpose of finding patterns within a dataset and this has been made flexible with the end user in mind so that it can be performed differently depending on the specific research question that one wants to answer. Several well established data mining techniques have been used, e.g. SVM and Random forest, together with more unexplored methods of performing data handling and feature selection, e.g. SciDB and Robust feature selection.

Something that has been realized through coming in contact with the subject of this thesis is that the possibilities of working with bioinformatics within high-content screening spans over a much broader span than the extent of this thesis. A lot of implementations specified for a specific kind of target user within biological research could and needs to be done. This could for example be different machine learnings algorithms that can support in decision making but also additional tools like parsers that are adapted to process a specific kind of dataset to minimize the amount of manual work. Current software provides a lot of functionalities for analysing data, but the feeling is that they are made far too general and often lack of support in some aspects, e.g. performing feature selection for multidimensional data. It is hard to provide an extensive software solution that shall work for all kinds of data and for all sorts of purposes. Many ideas have come up to discussion for implementation, but neglected due to the time limit or they being too far away from the scope of this thesis.

As mentioned in this chapter, a lot of improvements can be made upon different parts of the application and the idea is that this will be worked on in the future via other theses. The most relevant improvements are:

- A more adaptable parser that works for all dataset structures and formats.
- Implementing visualisation tools that can provide further insight for the user.
- Tools for approximating a variance matrix of the measurement errors in a dataset to provide support for Robust feature selection.

The compilation of this thesis will therefore act as a starting point through providing an extendable code basis and also investigations of which areas that requires further development and research. Our hope is that in the future, this work will contribute to a set of tools that is used continuously in the work of conducting data analysis within high-content screening.

Appendix A

HCS Current Manual Workflow

This section describes the manual workflow of the current data analysis methods the user used before this thesis was performed. It covers the different formats used for data management, the multiple software used for visualization as well as the techniques utilised for finding results. This workflow was documented in the spring of 2015 at the prestudy phase of this thesis.

Note that the workflow of whole process of performing HCS is not given in this section, only the parts related to the actual data analysis and this considers that data has been provided from an extensive image analysis on the screening results. However, some basic knowledge of HCS is required and can be acquired in chapter 2.

A.1 Summary

The described workflow can be concluded as a bit disorganized beacuse there is no standardized way of working. The main reasons for this are that the available tools are very flexible and need some deeper knowledge within the software for being able to fully use them or that the tools miss some functionalities. This has resulted in a large collection of software that are not used to their full potential. The analysis is performed differently depending on the biological questions that are addressed for the specific experiment and what kind of data that is the output from it. A important aspect to also consider with the current workflow tools is the limitations of handling larger amounts of data to make more extensive analysis.

Figure A.1 summarises the investigated workflow, which starts with the end user performing high-content screening and producing images as output. These images can be processed in MetaXpress which is the most preferred software today by the end user. CellProfiler is another software that also is available but it is rarely used. The image processing results in data at an image or cellular level where different features have been extracted and calculated. Analysis has been restricted to approximated data at an image level since more detailed data at a cellular level will produce an amount of data which is unmanageable to handle manually. This is because the selection of features has to be performed manually in Excel by utilising different computed parameters for each feature. There is also a restriction of only do this at one feature a time and this creates a requirement for iterating this process for some, by the user selected, features. This takes time and the user can miss significant features by neglecting them in this stage. Features that shows relevance in the Excel analysis are selected and visualised further in Spotfire. In Spotfire, the user can discover and group data to find conjuction in the data.

A.2 Data Extraction

The resulting data from the image analysis software are exported as matrices in CSV- or XLSX-format. The data is stored into a database for enabling data export at others occasions while annotation data is generated manually and only stored partially.

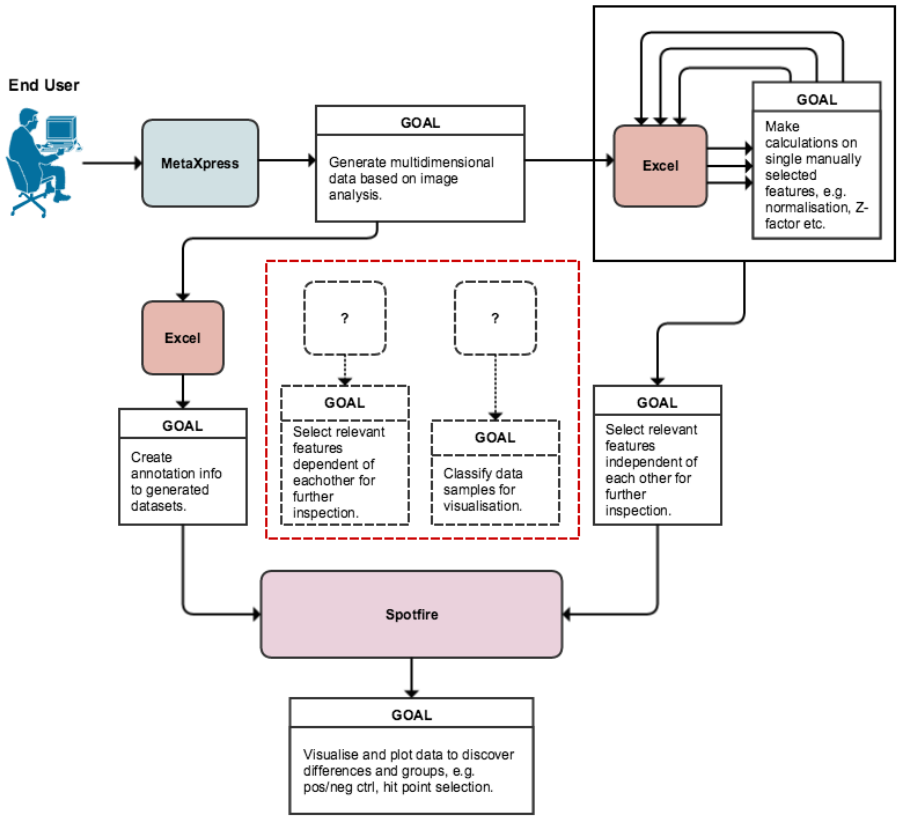


Figure A.1: Summarised working pipeline for the end user where the red dotted line describes parts that are not yet available but wanted for enhancing the workflow. Multiple lines defines that a task has to be iterated several times.

A.3 Analysis and Visualisation Software

This section describes the software that are used today by the end user. The different software has different input formats and the area of use for the different software are described below:

A.3.1 Excel

There are several customized templates for spreadsheets in *Excel* which include guidance for how to perform a standardised analysis of image level HCS data. These templates can however only handle single read-out data so one cannot analyze multiple features at the same time. The different templates are created for the purpose of handling one specific plate format, which makes it an extensive task to add functionality for a new plate format.

The templates provide calculations, e.g. Z' value per well, or computations for positive and negative control. Different plots like scatter plots, line plots were enabled but have customization problems, e.g. the axes adapt bad to the data. Histograms were enabled but difficult to implement so other software are better for that purpose.

The use of these templates was perceived as laborious which resulted in that calculations for analysis were performed manually in *Excel* without the templates.

A.3.2 Spotfire

When visualisation of the data was needed the software *Spotfire* was used. *Spotfire* offered plenty of different ways to represent the data for analysis together with the ability to manually filter the represented data.

A.4 Other Tools

This section describes tool that are rarely or never used but available for the end user and of interest in this thesis. At site there were several tools available for data analysis that were rarely used for different reasons which are listed below:

A.4.1 CellProfiler

CellProfiler is a flexible tool that provides machine learning methods through its Analyst version of the software. However the data needs to be extracted from image analysis performed by CellProfiler so it cannot be used in combination with another software, e.g. MetaXpress. The idea with this software creates good possibilities to perform analysis since it spans the pipeline of both image and data analysis but it also has some constraints that makes it difficult to use in some experiments. For example the images that are going to be analyzed need to be exported to files prior the analysis and this is not very convenient for analyzing multiple plates of screening data, which often is the case. The analysis methods also operate on compounds and make classification based on the wells in a plate. Often a more unbiased feature selection approach is preferred that makes classification on a cellular level. The software also demands the user to have a SQL database setup with the data to be able to use it.

A.4.2 Columbus

Columbus is an image data storage and analysis system with possible plugins e.g. PhenoLogic and export options for further analysis in other software. It is a big overall solution from data acquisition to analysis which is not open source and has a cost for each license per user. The tool is not used because it is perceived by the user as it is too time consuming to learn and also restricted to perform limited data analsysis, e.g. it cannot handle feature selection. See `http://www.perkinelmer.com/pages/020/cellularimaging/products/columbus.xhtml` for further information.

A.5 Limitations

One of the limitations with performing the current manual way of analysis is the amount of data that can be handled. With the current approach, there is a limitation of only including data on an image level, see fig. 2.2 in chapter 2. One single data sample on an image level can represent hundreds of cells. This restriction exists because the analysis is performed manually and even looking at one feature at a time gets too complex for data at cellular level. Image level data can also be defined as data with measured values per well and the acquired values become an approximation of all cellular data in the well.

The initial idea was to make a user study by following a HCS experiment from start to finish and make a more detailed walkthrough of the workflow. To perform an investigation of the analysis work for a specific experiment would be too time consuming since the work can be ongoing for a very long time and prepare data for such a task would be very much work. Therefore this study was conducted from an interview with a biological expert where different tools and techniques were described. The conclusion is that there is no real established way of working with analysis since the approach varies a lot between experiments.

Appendix B

Literature Study

This appendix describes the progress of finding relevant research and literature for this thesis and some interesting discoveries from this search.

B.1 Databases

The databases that were investigated for finding relevant literature together with a argumentation of how they were selected are described in this section.

Multiple web services have been used in order to make the search as extensive as possible. The following list of services has been investigated:

- Web of science
- Scopus
- Inspec
- Pubmed
- Google Scholar
- Arxive
- IEEE database
- JSTOR
- Microsoft Academic Research
- MathSciNet

A reduced selection of these has been used for the search and the selection of services have been chosen according to the following criteria:

- Which databases the service cover
- How big search query can the service handle
- How relevant the results are

These criteria ensure that the search result from different services does not come from the same databases, covers as many databases as possible and provides relevant search results. To make the search as extensive as possible all synonyms of the key words need to be included in the search, this yields a very long search query the services require to be able to handle. To explore the last criterium, relevance of the search result at the specific services was established through a brief investigation of the result by reading the abstract and title of resulting literature. A service which is very popular is Google Scholar which provides many results with quite good relevance. But search results from this service were inconsistent due to frequent changes of source databases and

also not providing enough size for search query which made the searches incomplete [53]. For this reason the service has been excluded for usage in this literature study. The selection of services was chosen according to the previous reasoning with criteria, popularity and reviews.

The resulting services used are presented below:

B.1.1 Web of science

One of the largest research databases of scholarly research data which is acknowledged by almost 7000 of the world's leading scholarly institutions. Web of science provides a general source of database which consists of data from more than 250 disciplines [54].

Web of science passed all the criterias and provided literature with relevance. Search result with different search queries are presented in figure 2.12.

B.1.2 Scopus

Scopus is an extensive database for scientific content that specifies their coverage of subjects in five different areas of science where the health and physical science have the largest part (over 60%) [55].

B.1.3 Pubmed

A service with focus on biomedical literature with more than 24 million articles [56]. All these services are well known and are some of the most popular sources for biomedical science [53].

B.2 Search Queries

This section includes a description of which combination of search queries that were used.

The resulting literature was desired to cover three different areas, high content screening, feature selection and data analysis. All these areas have several synonyms, thus all synonyms found needed to be included.

The search queries have been performed in different combinations since the services provided poor results on all areas combined which indicates that this area is an unexplored field of research.Searches with different combinations of queries with results from 10 years back to present are represented in figure 2.12

The publications which consist of data analysis in combination of variable selection (red line in fig. 2.12) steadily increase over the years which shows an increasing popularity for the subject. The big difference between the search result from data analysis and feature selection and the result which included High content screening (blue and yellow line) shows that this area of result is a smaller research area. But the trend over time shows that a big increase of released publications that considers HCS after year 2010.

The synonyms used are listed below:

Variable selection (VS): "feature selection" "feature reduction" "feature ranking" "attribute selection" "attribute reduction" "attribute ranking" "variable selection" "variable reduction" "variable ranking" "feature subset selection" "feature subset reduction" "attribute subset selection" "attribute subset reduction" "variable subset selection" "variable subset reduction" "selection of feature" "selection of features" "reduction of feature" "reduction of features" "ranking of feature" "ranking of features" "selection of attribute" "selection of attributes" "reduction of attribute" "reduction of attributes" "ranking of attribute" "ranking of attributes" "selection of variable" "selection of variables" "reduction of variable" "reduction of variables" "ranking of variable" "ranking of variables" "selection of feature subset" "selection of feature subsets" "selection of attribute subset" "selection of attribute subsets" "selection of variable subset" "selection of variable subsets"

"reduction of feature subset" "reduction of feature subsets" "reduction of attribute subset" "reduction of attribute subsets" "reduction of variable subset" "reduction of variable subsets" "ranking of feature subset" "ranking of feature subsets" "ranking of attribute subset" "ranking of attribute subsets" "ranking of variable subset" "ranking of variable subsets" "dimensionality reduction" "reduction of dimensionality" "dimension reduction"

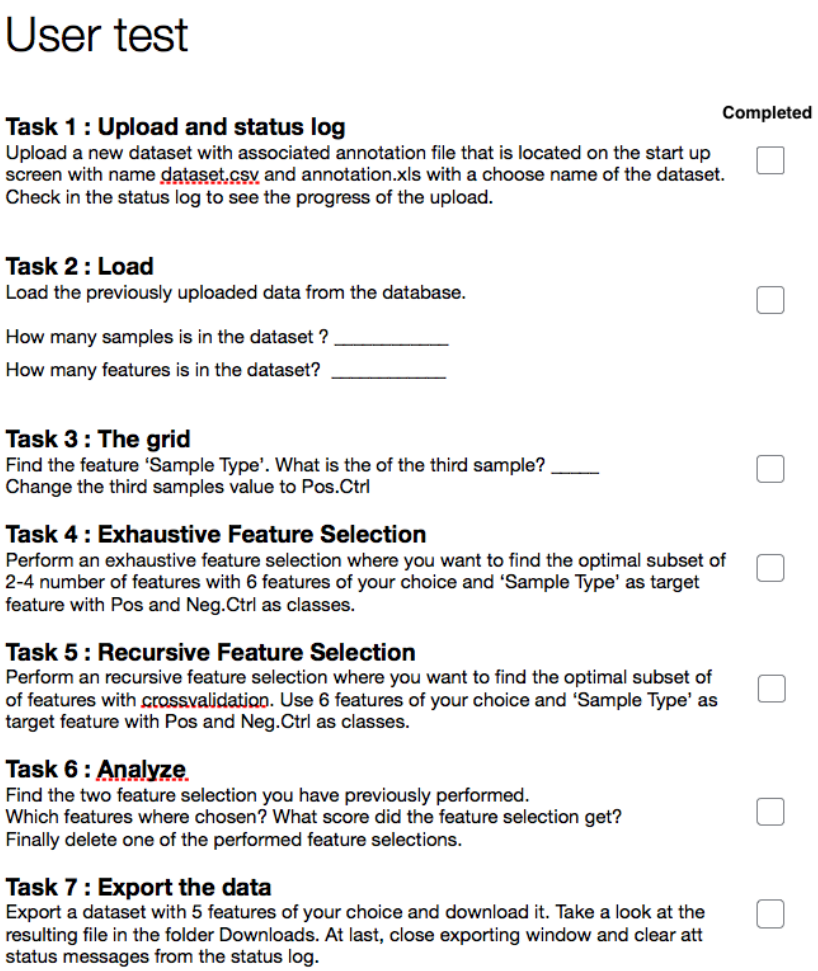
High-content screening (HCS): "high content screening" "hcs" "high-content analysis" "high content analysis" "hca" "high-content imaging" "high content imagning" "cellomics" "cellular imaging" "automated microscopy" "phenotypic screening"

Data analysis (DA): "data processing" "data mining" "data analysis" "machine learning" "signal processing" "big data" "knowledge discovery" "knowledge discovery in databases" "kdd" "eda" "business intelligence" "business analytics" "business analysis" "data science" "informatics" "data modeling" "data prediction" "information analysis" "predictive analytics" "data visualization" "data dissemination"

Appendix C

Usability Test

This appendix consist of the usability test that was used during usability testing. Results from the usability test is discussed in Method section 3.5.1.



User test

Completed

Task 1 : Upload and status log
Upload a new dataset with associated annotation file that is located on the start up screen with name dataset.csv and annotation.xls with a choose name of the dataset. Check in the status log to see the progress of the upload.

Task 2 : Load
Load the previously uploaded data from the database.

How many samples is in the dataset ? _____

How many features is in the dataset? _____

Task 3 : The grid
Find the feature 'Sample Type'. What is the of the third sample? _____
Change the third samples value to Pos.Ctrl

Task 4 : Exhaustive Feature Selection
Perform an exhaustive feature selection where you want to find the optimal subset of 2-4 number of features with 6 features of your choice and 'Sample Type' as target feature with Pos and Neg.Ctrl as classes.

Task 5 : Recursive Feature Selection
Perform an recursive feature selection where you want to find the optimal subset of of features with crossvalidation. Use 6 features of your choice and 'Sample Type' as target feature with Pos and Neg.Ctrl as classes.

Task 6 : Analyze
Find the two feature selection you have previously performed.
Which features where chosen? What score did the feature selection get?
Finally delete one of the performed feature selections.

Task 7 : Export the data
Export a dataset with 5 features of your choice and download it. Take a look at the resulting file in the folder Downloads. At last, close exporting window and clear att status messages from the status log.

Figure C.1: Usability test.

Appendix D

Iris Dataset

This appendix contains information about the Iris dataset.

The Iris dataset [57] is a well known dataset that has been used in numerous pattern recognition problems in the past. It was first published by Sir Ronald Fisher in 1936 [58] and contains 3 different species (Iris Setosa, Iris Virginica and Iris Versicolor) of the Iris plant with 50 samples of each. Every sample includes 4 attributes besides the different classes that represent species:

- Sepal Length
- Sepal Width
- Petal Length
- Petal Width

These attributes represent width and length of the different leafs on the flower.

Attribute	Min	Max	Mean	Standard Deviation	Class Correlation
Sepal Length	4.3	7.9	5.84	0.83	0.78
Sepal Width	2.0	4.4	3.05	0.43	-0.42
Petal Length	1.0	6.9	3.76	1.76	0.95
Petal Width	0.1	2.5	1.20	0.76	0.96

Table D.1: Iris dataset statistics.

Figure D.1 contains scatter plots and histograms for all attributes in the dataset. Every color represent one of the classes. Table D.1 provide some statistics for the dataset.

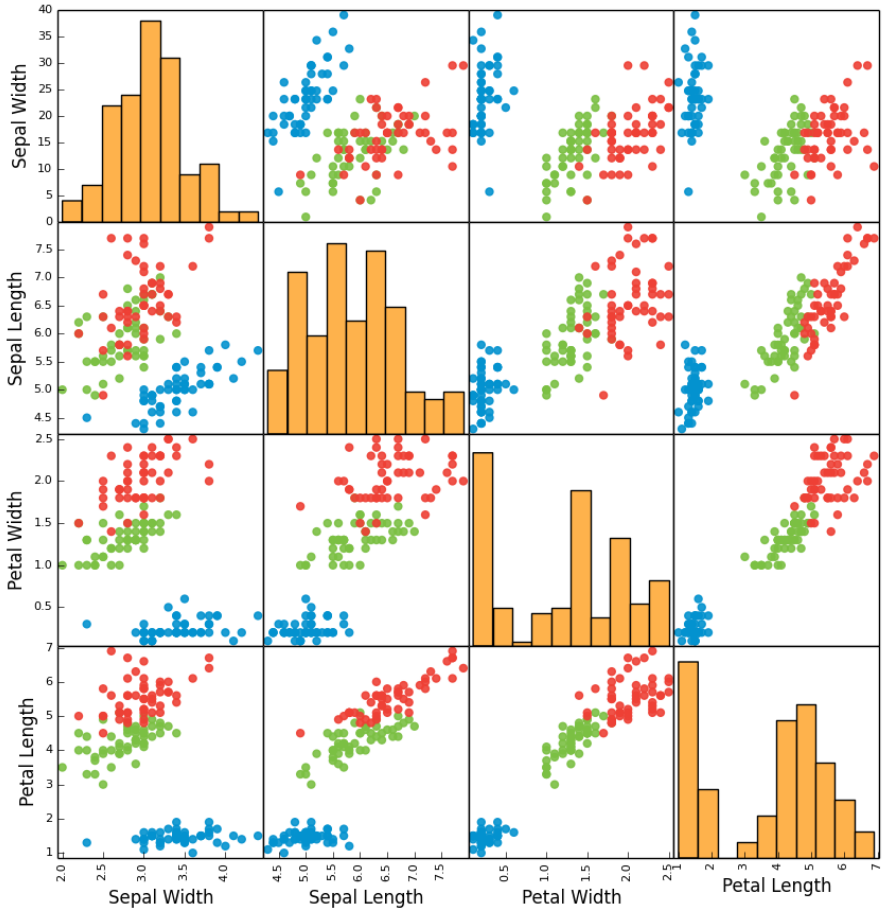


Figure D.1: Scatter matrix and histogram plots for the Iris dataset. Every color represents a specific species of the flower.

Appendix E

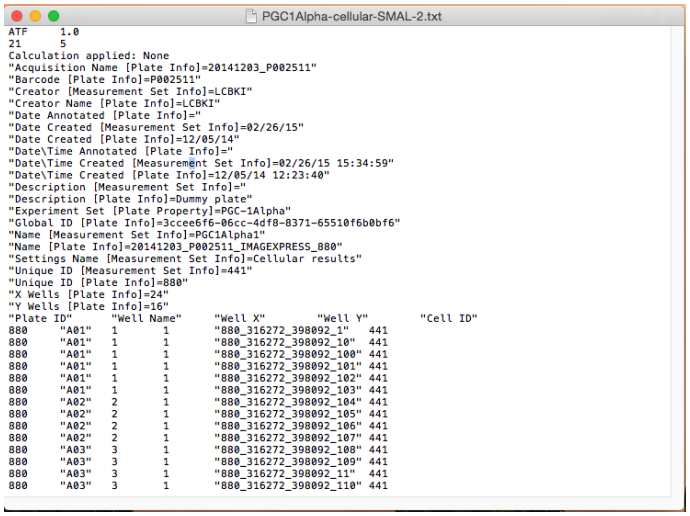
HCS Dataset

This appendix explains the structure of a HCS dataset and the annotation data.

E.1 Dataset Generated From MetaXpress

This section explains the format and structure of the data set generated from MetaXpress.

The data is generated in a text file with tab separated format. All files start with "ATF 1.0" followed by a row that tells how many rows of plate level data the current plate has and how many different features that exist in the cell. The next information contains plate level data followed by the header for all features in cell level data. Finally the actual cell data fills up the rest of the rows for the plate. If there are more than one plate the format is the same, but the dataset is appended so when a new plate begins the first row begins with ATF 1.0 and then the same structure as previously mentioned.


PGC1Alpha-cellular-SMAL-2.txt

```
ATF     1.0
21      5
Calculation applied: None
"Acquisition Name [Plate Info]=20141203_P002511"
"Barcode [Plate Info]=P002511"
"Creator [Measurement Set Info]=LCBKI"
"Creator Name [Plate Info]=LCBKI"
"Date Annotated [Plate Info]="
"Date Created [Measurement Set Info]=02/26/15"
"Date Created [Plate Info]=12/05/14"
"Date\Time Annotated [Plate Info]="
"Date\Time Created [Measurement Set Info]=02/26/15 15:34:59"
"Date\Time Created [Plate Info]=12/05/14 12:23:40"
"Description [Measurement Set Info]="
"Description [Plate Info]=Dummy plate"
"Experiment Set [Plate Property]=PGC-1Alpha"
"Global ID [Plate Info]=3ccee6f6-86cc-4df8-8371-65510f6b0bf6"
"Name [Measurement Set Info]=PGC1Alpha1"
"Name [Plate Info]=20141203_P002511_IMAGEXPRESS_880"
"Settings Name [Measurement Set Info]=Cellular results"
"Unique ID [Measurement Set Info]=441"
"Unique ID [Plate Info]=880"
"X Wells [Plate Info]=24"
"Y Wells [Plate Info]=16"
"Plate ID"      "Well Name"     "Well X"        "Well Y"        "Cell ID"
880     "A01"   1       1       "880_316272_398092_1"   441
880     "A01"   1       1       "880_316272_398092_10"  441
880     "A01"   1       1       "880_316272_398092_100" 441
880     "A01"   1       1       "880_316272_398092_101" 441
880     "A01"   1       1       "880_316272_398092_102" 441
880     "A01"   1       1       "880_316272_398092_103" 441
880     "A02"   2       1       "880_316272_398092_104" 441
880     "A02"   2       1       "880_316272_398092_105" 441
880     "A02"   2       1       "880_316272_398092_106" 441
880     "A02"   2       1       "880_316272_398092_107" 441
880     "A03"   3       1       "880_316272_398092_108" 441
880     "A03"   3       1       "880_316272_398092_109" 441
880     "A03"   3       1       "880_316272_398092_11"  441
880     "A03"   3       1       "880_316272_398092_110" 441
```

Figure E.1: Example for the structure of a dataset generated from MetaXpress.

The dataset in figure E.1 is an example of a dataset generated from MetaXpress which consists of one plate of data with 21 rows of plate specific data and 5 different features at the cellular leveled data.

E.2 Annotation Data

This section includes structure of the template for annotations that optionally could be added to the data set from MetaXpress.

The annotation template that shall complement the data generated from MetaXpress is filled in manually and is in the Excel format. The template consist of 4 different sheets of information and their name is Experiment description, Plate layout, Plate map and Plates.

E.2.1 Experiment Description

This sheet consist of some general experiment data and some additional some optional comments. All info in this sheet is optional.

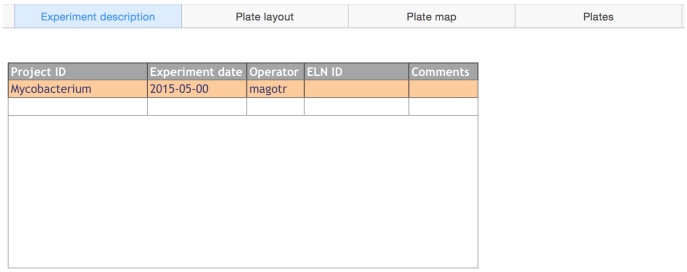


Figure E.2: Example of an experiment description in annotation data.

See fig. E.2 for example of experiment description.

E.2.2 Plate Layout

This section consist of information about the layout of the plates in the dataset.

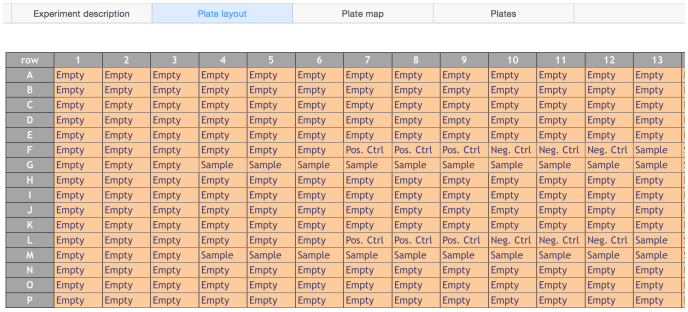


Figure E.3: Example of a plate layout in annotation data.

.

See fig. E.3 for example of the plate layout.

E.2.3 Plate Map

The sheet Plate map consist of information about which compounds and concentrations the different wells in the different plates had. Compound Plate ID and Well Name are required fields to fill in for this sheet.

See fig. E.4 for example of plate map information.

E.2.4 Plates

Plates consist of summarized data over the plates. It consist of the Accquisition Name which the annotation matches with from the dataset from MetaXpress. Required cells to fill here in Compound plate ID and Accquisition Name.

See fig. E.1 for example of plates information.

Experiment description | Plate layout | Plate map | Plates

Compound Plate ID	Volume	Well Name	Compound ID	External ID	Batch nr	Form	Conc (mM)
day 1		F07	cells				
		F08	cells				
		F09	cells				
		F10	cells				
		F11	cells				
		F12	cells				
		F13	cells				
		F14	cells				
		F15	cells				
		G04	cells				
		G05	cells				
		G06	cells				
		G07	cells				
		G08	cells				
		G09	cells				
		G10	cells				
		G11	cells				
		G12	cells				
		G13	cells				
		G14	cells				
		G15	cells				
day 3		F07	cells				

Figure E.4: Example of a plate map in annotation data.

Experiment description | Plate layout | Plate map | Plates

Plate number	Compound Plate ID	Volume	Acquisition Name	Concentration [uM]
1	day 1		XXX	
2	day 3		XXX	
3	day 6		XXX	
4				
5				
6				
7				

Figure E.5: Example of plates information in annotation data.

Bibliography

[1] *Excel*. URL: `http://microsoft-excel.sv.softonic.com/` (visited on 06/15/2015).

[2] *Spotfire*. URL: `http://spotfire.tibco.com/products/spotfire-desktop` (visited on 06/15/2015).

[3] Steven a Haney et al. "High-content screening moves to the front of the line." In: *Drug discovery today* 11.19-20 (2006), pp. 889–894. ISSN: 1359-6446.

[4] Fabian Zanella, James B Lorens, and Wolfgang Link. "High content screening: seeing is believing." In: *Trends in biotechnology* 28.5 (2010), pp. 237–245. ISSN: 1879-3096.

[5] K. a. Giuliano. "High-Content Screening: A New Approach to Easing Key Bottlenecks in the Drug Discovery Process". In: *Journal of Biomolecular Screening* 2 (1997), pp. 249–259. ISSN: 1087-0571.

[6] William Buchser, Mark Collins, Tina Garyantes, Rajarshi Guha, Steven Haney, Vance Lemmon, Zhuyin Li, and O. Joseph Trask. "Assay Development Guidelines for Image-Based High Content Screening, High Content Analysis and High Content Imaging". In: *Sittampalam GS, Coussens NP, Nelson H, et al., editors. Assay Guidance Manual* ().

[7] Yann Abraham, Xian Zhang, and Christian N Parker. "Multiparametric Analysis of Screening Data Growing Beyond the Single Dimension to Infinity and Beyond". In: *Journal of biomolecular screening* 19.5 (2014), pp. 628–639.

[8] Anthony Davies et al. *An Introduction To High Content Screening: Imaging Technology, Assay Development, and Data Analysis in Biology and Drug Discovery*. John Wiley & Sons, 2014.

[9] Frans Coenen. "Data mining: past, present and future". In: *The Knowledge Engineering Review* 26.01 (2011), pp. 25–29.

[10] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[11] Gareth James et al. *An introduction to statistical learning*. Springer, 2013.

[12] Andy Liaw and Matthew Wiener. "Classification and regression by randomForest". In: *R news* 2.3 (2002), pp. 18–22.

[13] Victor Robles Pedro Larrañaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Iñaki Inza, José A. Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez. *Machine Learning in Bioinformatics*. 2005, pp. 86–112. ISBN: 9780470116623.

[14] Pierre Geurts, Damien Ernst, and Louis Wehenkel. "Extremely randomized trees". In: *Machine learning* 63.1 (2006), pp. 3–42.

[15] Torbjörn E M Nordling. "Robust inference of gene regulatory networks: System properties, variable selection, subnetworks, and design of experiments". Ph.D. thesis. Stockholm, Sweden: KTH Royal Institute of Technology, 2013, pp. xi, 350. ISBN: 978-91-7501-762-4.

[16] Huan Liu et al. "Feature Selection: An Ever Evolving Frontier in Data Mining". In: *JMLR Workshop and Conference Proceedings Volume 10: Feature Selection in Data Mining*. Ed. by Neil Lawrence. Hyderabad, India: JMLR, 2010, pp. 4–13.

[17] Ms Shweta Srivastava, Ms Nikita Joshi, and Madhvi Gaur. "A Review Paper on Feature Selection Methodologies and Their Applications". In: *International Journal of Computer Science and Network Security* 14.5 (2014), p. 78.

[18] Shuangge Ma and Jian Huang. "Penalized feature selection and classification in bioinformatics". In: *Briefings in bioinformatics* 9.5 (2008), pp. 392–403.

[19] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. "A review of feature selection techniques in bioinformatics." In: *Bioinformatics (Oxford, England)* 23.19 (2007), pp. 2507–2517. ISSN: 1367-4811.

[20] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. "A review of feature selection methods on synthetic data". In: *Knowledge and Information Systems* 34.3 (2012), pp. 483–519. ISSN: 0219-1377.

[21] Isabelle Guyon et al. "Gene Selection for Cancer Classification using Support Vector Machines". In: *Machine Learning* 46.1 (2002), pp. 389–422–422. ISSN: 0885-6125.

[22] Matthew Shardlow. "An Analysis of Feature Selection Techniques". In: ().

[23] Philip M Dixon et al. "Bootstrapping the Gini Coefficient of Inequality". In: *Ecology* 68.5 (1987), pp. 1548–1551. ISSN: 00129658.

[24] *SciDB*. URL: `http://www.paradigm4.com/` (visited on 06/15/2015).

[25] Paul G Brown. "Overview of sciDB: Large Scale Array Storage, Processing and Analysis". In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. SIGMOD '10. New York, NY, USA: ACM, 2010, pp. 963–968. ISBN: 978-1-4503-0032-2.

[26] Michael Stonebraker et al. "The Architecture of SciDB". In: *Proceedings of the 23rd International Conference on Scientific and Statistical Database Management*. SSDBM'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 1–16. ISBN: 978-3-642-22350-1.

[27] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". In: *Commun. ACM* 51.1 (2008), pp. 107–113. ISSN: 0001-0782.

[28] Tom White. *Hadoop: The definitive guide.* " O'Reilly Media, Inc.", 2012.

[29] Lei Yu and Huan Liu. "Redundancy based feature selection for microarray data". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, pp. 737–742.

[30] *MetaXpress*. URL: `http://www.moleculardevices.com/systems/high-content-imaging/metaxpress-high-content-image-acquisition-and-analysis-software` (visited on 06/15/2015).

[31] *CellProfiler*. URL: `http://www.cellprofiler.org/` (visited on 06/15/2015).

[32] *MsOffice: Excel technical specification*. 2015. URL: `https://support.office.com/en-nz/article/Excel-specifications-and-limits-1672b34d-7043-467e-8e27-269d656771c3` (visited on 04/07/2015).

[33] *KNIME*. URL: `https://www.knime.org/` (visited on 06/15/2015).

[34] *React*. URL: `https://facebook.github.io/react/` (visited on 06/15/2015).

[35] *Flux*. URL: `https://facebook.github.io/flux/` (visited on 06/15/2015).

[36] *Bootstrap*. URL: `http://getbootstrap.com/` (visited on 06/15/2015).

[37] *jQuery*. URL: `https://jquery.com/` (visited on 06/15/2015).

[38] *PapaParse*. URL: `http://papaparse.com/` (visited on 06/15/2015).

[39] *Nginx*. URL: `http://nginx.org/` (visited on 06/15/2015).

[40] *Gunicorn*. URL: `http://gunicorn.org/` (visited on 06/15/2015).

[41] *Flask*. URL: `http://flask.pocoo.org/` (visited on 06/15/2015).

[42] *SQLite*. URL: `https://www.sqlite.org/` (visited on 06/15/2015).

[43] *scikit-learn*. URL: `http://scikit-learn.org/stable/` (visited on 06/15/2015).

[44] *Virtualenv*. URL: `https://virtualenv.pypa.io/en/latest/` (visited on 06/15/2015).

[45] *Gulp*. URL: `http://gulpjs.com/` (visited on 06/15/2015).

[46] *Bower*. URL: `http://bower.io/` (visited on 06/15/2015).

[47] *npm*. URL: `https://www.npmjs.com/` (visited on 06/15/2015).

[48] *Node.js*. URL: `https://nodejs.org/` (visited on 06/15/2015).

[49] *Browserify*. URL: `http://browserify.org/` (visited on 06/15/2015).

[50] *reactify*. URL: https://github.com/andreypopp/reactify (visited on 06/15/2015).

[51] *Web Worker*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers (visited on 06/17/2015).

[52] *SciDB-Py*. URL: http://scidb-py.readthedocs.org/en/latest/ (visited on 06/17/2015).

[53] Matthew E Falagas et al. "Comparison of PubMed, Scopus, web of science, and Google scholar: strengths and weaknesses". In: *The FASEB journal* 22.2 (2008), pp. 338–342.

[54] *Web of science*. URL: http://thomsonreuters.com/content/dam/openweb/documents/pdf/scholarly-scientific-research/fact-sheet/wos-next-gen-brochure.pdf (visited on 06/17/2015).

[55] *Scopus*. URL: http://www.elsevier.com/online-tools/scopus/content-overview (visited on 06/02/2015).

[56] *Pubmed*. URL: http://www-ncbi-nlm-nih-gov.e.bibl.liu.se/pubmed/ (visited on 06/17/2015).

[57] M Lichman. {*UCI*} *Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml.

[58] RA Fisher. "The Use of Multiple Measurements in Taxonomic Problems". In: *Annals of Eugenics* 7.2 (1936), pp. 179–188. ISSN: 1469-1809.

[59] Shantanu Singh, Anne E Carpenter, and Auguste Genovesio. "Increasing the Content of High-Content Screening: An Overview." In: *Journal of biomolecular screening* 19.5 (2014), pp. 640–650. ISSN: 1552-454X.

[60] *SlickGrid*. URL: https://github.com/mleibman/SlickGrid (visited on 06/15/2015).