FACULTY OF BIOSCIENCE ENGINEERING

A COMPARISON OF FLAT AND HIERARCHICAL CLASSIFICATION FOR AUTOMATIC ANNOTATION OF SINGLE-CELL TRANSCRIPTOMICS DATA

word count: 20.928

Lauren Theunissen
Student ID: 01602801

Supervisors: Prof. Dr. Willem Waegeman, Prof. Dr. Yvan Saeys and Prof. Dr. Pieter De Bleser

A dissertation submitted to Ghent University in partial fulfilment of the requirements for the degree of master in Science in Bioinformatics: Bioscience Engineering.
Academic year: 2020 - 2021

GHENT UNIVERSITY

Gent, June 10, 2021

The promotors,
Prof. Dr. Willem Waegeman,
Prof. Dr. Yvan Saeys,
Prof. Dr. Pieter De Bleser

The author,
Lauren Theunissen

PREFACE

This dissertation is the result of a collaboration between the KERMIT laboratory from the Department of Data Analysis and Mathematical Modelling at the University of Ghent and the Saeys lab at the VIB (Vlaams Instituut voor Biotechnology), their joined knowledge led to the needed insights and ideas for the creation of this dissertation.

Therefore, I would sincerely like to thank prof. dr. Willem Waegeman, prof. dr. Yvan Saeys and prof. dr. Pieter De Bleser for their help, advice and time. I wasn't quite familiar with the given topic when I started this dissertation, but their input and advice helped me with gaining a thorough understanding of the topic. Sadly enough, due to COVID19, it was impossible to hold meetings in person and digital meetings are never quite the same as face-to-face meetings. Regardless, during the meetings I always felt interest, support and enthusiasm from all the professors, and for that I am really grateful.

I also own a great deal of thanks to my tutor, Thomas Mortier. I am quite aware that he has had a very busy year. But he still always found the time to answer my questions, even if they came at a less convenient time. Furthermore, his answers were always quite elaborate. So thank you for taking the time to make sure I understood everything fully.

Lastly, I want to thank my family. The past year was not always easy for everyone. Social contact lessened greatly, and this is not always the most fun to experience while working on your dissertation. My parents, my brother Julian and Emma were there for me at the lesser times, the times were I was stressing or panicking or when I just needed to vent. I hope (and think) that they know that I appreciate them greatly for this.

LIST OF ABBREVIATIONS

scRNA	single-cell RNA
qRT-PCR	quantitative reverse transcription polymerase chain reaction
cDNA	coding DNA
UMI	unique molecular identifiers
FACS	fluorescence-activated cell sorting
mRNA	messenger RNA
PCR	polymerase chain reaction
IVT	in vitro transcription
SVC	support vector classifier
SVM	support vector machine
DAG	directed acyclic graph
LCN	local classifier per node
LCPN	local classifier per parent node
LCL	local classifier per level
TP	true positives
FP	false positives
TN	true negatives
FN	false negatives
TPR	true positive rate
TNR	true negative rate
FPR	false positive rate
AUC	area under the ROC curve
ROC	receiver operating curve
VIB	Vlaams Instituut voor Biotechnologie
MAD	median absolute deviation
AMB	Allen mouse brain
MAIT cells	mucosal associated invariant T cells
IFN	interferon
MHC	major histocompatibility complex
CITE-Seq	cellular indexing of transciptomes and epitopes by sequencing

CONTENTS

SUMMARY

With the development of single-cell sequencing techniques, a whole new world of possibilities opened up to further resolve and discover biological phenomena. But, in order to use the single-cell data, cell type annotation is a crucial, first step. Manual cell type annotation is a laborious process that generates non-reproducible results with a not-standardised vocabulary. So, in the past couple of years interest increased in the development of automatic cell type annotation tools. At the moment several of these tools exist, either based on marker gene detection or supervised classification with a labelled dataset. In a recent benchmarking study involving 22 annotation methods with different approaches, a supervised approach with a simple linear SVM classifier performed best (Abdelaal et al., 2019). As a first step in this dissertation, these results were confirmed. Incorporation of cell type hierarchy in the automatic annotation process is not a common practice, but can have certain advantages and could thus possibly lead to improved annotations. The main goal of this dissertation was to compare automatic cell type annotation with and without the incorporation of this cell type hierarchy. For this purpose, two datasets were used and three different classifiers: the logistic regression, random forests and linear SVM classifier. The analyses show that hierarchical classification can lead to better performances and the order of improvement seems to depend on the dataset's complexity. With the comparison analyses, the linear SVM performance is rivalled by the logistic regression classifier if not sometimes improved, which shows promise for the use of the logistic regression classifier for single-cell data annotation. Furthermore, it is observed that for hierarchical classification, the presence of a correct cell type hierarchy is crucial, yet not always easily generated.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

SINGLE-CELL TRANSCRIPTOMICS

1.1 Introduction

The transcriptome is the entirety of coding and non-coding RNA transcripts present in a cell or tissue sample. It encodes the sample's or cell's functional activity state and can give insight into biological responses, regulatory processes, disease development, etc. The transcriptome is mostly analysed with bulk tissue samples that give snapshots of the transcriptome (Lowe et al., 2017; Srivastava et al., 2019). Analysis of these samples results in one expression profile per sample that is averaged over the thousands or millions of cells present in the bulk sample, which is then assigned to all the cells in the sample (see Figure 1.1). However, these samples usually consist out of different cell types, each with their own role, function and activity state, which makes these bulk techniques less suitable for transcriptome analysis. Furthermore, there is always inherent stochasticity present in biological cells that will result in variability. Even if a very pure tissue sample is obtained with only one cell type present, the use of bulk RNA sequencing techniques will typically still result in loss of information due to biological variation that is averaged out. This biological variation is attributed to randomness in translational and transcriptional processes and should be controlled. Furthermore, extrinsic factors such as environmental factors and cell cycle stages are also present in varying forms inside tissue samples and they will induce cell to cell variation. A direct consequence is that the use of bulk methods will result in all or none conclusions, while the biological responses are gradual processes (Kanter and Kalisky, 2015; Wang and Navin, 2015; Raj and van Oudenaarden, 2008; Sandberg, 2014).

Due to the shortcomings mentioned above, together with the need for more detailed transcriptomic information to further resolve biological questions and help in disease treatment development, the need for transcriptomics techniques with a higher reso-

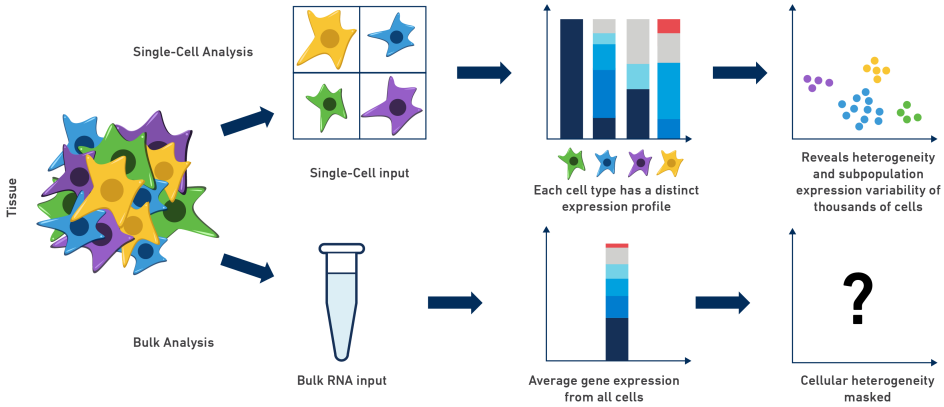

Figure 1.1: A single-cell transcriptome analysis results in one expression profile per sample or cell and gives more information than a bulk transcriptome analysis, which results in one averaged expression profile that is assigned to all the cells in the sample (10x Genomics, 2017).

lution rose, resulting in the rapid development of a myriad of new techniques in the last decade. Today, it is possible to profile transcriptomic signals at a single-cell level.

Single-cell transcriptome analysis provides a multitude of new opportunities (Wang and Navin, 2015; Sandberg, 2014; Aldridge and Teichmann, 2020):

- It allows the observation of gene expression on a single-cell level.
- It can help with resolving cell to cell variation.
- It can help with resolving transcriptional landscapes and regulatory networks.
- It can lead to the discovery of rare cell types that may play a large role in disease progression.
- It allows the discovery of new, previous unknown cell types.
- It broadens research opportunities and allows a deeper understanding of inter- and intracellular workings of different cell types and tissue's microenvironments.

1.2 Techniques

Multiple single-cell transcriptome analysis techniques exist. The three main ones are quantitative reverse transcription polymerase chain reaction (qRT-PCR), plate-based systems and RNA sequencing methods. Here we will focus on single-cell RNA sequencing (scRNA sequencing), which is a high-throughput, unbiased method. All possible single-cell sequencing protocols follow roughly the same steps: a single cell is isolated, the RNA is extracted from the cells, converted to cDNA for stability and

amplificated. This cDNA library is then sequenced using the same protocols as with bulk RNA sequencing to obtain the RNA transcript's codes and these are then further analysed with bioinformatics tools (see Figure 1.2). A lot of different scRNA sequencing technologies exist that can be distinguished based on the following aspects: cell isolation, cell lysis, reverse transcription, amplification, transcript coverage, strand specificity and UMI availability (see Section 1.2.1 and 1.2.2 for more details). The main criteria used to classify the single-cell sequencing technologies is whether they produce full length transcript reads or whether they only capture the 3' end of the transcript (Chen et al., 2019).

1.2.1 Single cell isolation

The isolation of a large number of cells is not an easy task. Especially since a proper sampling of the tissue is desired, which is random and includes all the cell types present, even the rare ones. Different techniques and strategies have been developed and used like FACS (fluorescence-activated cell sorting) or other flow cytometric based tools and microfluidic-based systems. FACS allows the sorting (and thus the identification) of specific cell types and can be used to obtain very pure samples (Kannan et al., 2019). Microfluidic devices or chips isolate single cells on a chip by the addition of very small volumes of cell suspension in the different chambers of the chip. In these chambers, amplification of the RNA is possible after which the cDNA is collected and sequenced (Svensson et al., 2018).

A more recently developed method that is gaining a lot of popularity, is the inDrop and Drop-seq method, where the cells are captured in nanoliter droplet emulsions. Here, two flows are brought together, one containing the cell suspension and another containing the materials and substances needed for amplification and sequencing. The combined flow is then separated in droplets or alternatively the cells are brought into a picoliter well, before continuing the protocol. By modifying the flow rates and dilution rates, while taking Poission statistics into account, one obtains single-cell isolation droplets, containing one or no cells (Svensson et al., 2018).

In some cases the spatial context of the cells can be of importance, which is usually lost when the cells are brought into suspension for isolation. Certain cell isolating systems like laser-capture microdissection, which cuts out very small tissue regions after microscopic identification, exist to prevent this loss. A multitude of other methods for retaining spatial information together with transcriptomics information exist as well, using various approaches (Asp et al., 2020). The capture of really rare cell type (<1%) can sometimes also lead to difficulties. Specialised techniques have been developed based on antibodies or other characteristics like the size or charge of the specific cell

type, to very specifically target and capture these rare cell types (Wang and Navin, 2015).

Certain specific cell types, like neurons, are morphologically very complex, making the intact capture of these cell types as a single cell from a tissue sample difficult. Normally, single cells are obtained from tissue samples with the help of harsh dissociation steps. But not all cell types can handle this, resulting in under- or overrepresentation of cell types and or harm to certain cells. Single-nuclei RNA sequencing circumvents this problem. By only isolating the nuclei of the cells, the cell type levels will be unaffected by the isolation process since nuclei are a lot more resistant than (whole) cells (Kulkarni et al., 2019).

1.2.2 Amplification

A single cell contains around 1 - 50 pg RNA, from which 0,1 pg is mRNA. For a successful RNA-seq experiment 0,1 - 1,0 μg RNA is needed. Therefore, prior to sequencing, an amplification step is needed to obtain sufficient RNA. Amplification of the transcripts can be achieved in two ways, either by PCR or IVT. PCR or polymerase chain reaction amplification results in an exponential increase of transcripts. Amplification is obtained by subsequently denaturing the double-stranded cDNA, adding adaptor sequences to the single strands and replicating the 2 single stranded cDNA parts with the help of DNA polymerases that recognize the adaptor sequences. This process is repeated for several cycles until sufficient starting material is obtained for sequencing. IVT or in vitro transcription results in a linear increase of cDNA and is better suited for longer cDNA fragments, which troubles the PCR procedure. In vitro transcription uses living mammalian cells to amplify the cDNA (Svensson et al., 2018). The cDNA template together with the polymerase and nucleotides are injected in the cell following the protocol described by Eberwine et al. (1992) (Wang and Navin, 2015). IVT's downside is the minimum amount of starting cDNA needed (around 400 pg) to be able to perform the experiment. IVT is however still used in some protocols like Cel-seq where the problem of input material is circumvented by multiplexing (Hashimshony et al., 2012). Multiplexing is the practice of analysing multiple samples at once by adding unique barcodes to the cDNA fragments of every sample before pooling and sequencing the cDNA fragments in one run. This is a common practice in sequencing to increase the output of the sequencing methods and to decrease the costs and time of the sequencing procedure (Svensson et al., 2018).

Amplification can lead to distorted transcript levels due to errors induced during the amplification step, which leads to wrong expression profiles. To counter this, UMI's are introduced. UMI's or unique molecular identifiers are barcodes added to the orig-

inal extracted cDNA fragments (before amplification) and are thus specific to each original copy of cDNA extracted from one single cell. UMI's allow to keep track of the original transcripts present in the single cell, reduce amplification bias and provide estimations of absolute molecular counts (Wang and Navin, 2015; Islam et al., 2014). UMI's are often combined with multiplexing in various sequencing protocols.

1.2.3 Single-cell sequencing data versus bulk sequencing data

As single-cell sequencing requires different, specialised methods compared to bulk sequencing, single-cell data has different characteristics than bulk data. Single-cell data will contain a much higher variability than bulk data. On the one hand, more biological variability will be present that is missed with bulk technologies and on the other hand more technical variability will be present, introduced by the single-cell technologies. It is possible to estimate technical variability with RNA spike-ins and unique molecular identifiers (UMI's). RNA spike-ins are known RNA transcripts that are added to the extracted RNA (before amplification) in a specific amount and can be used for calibration purposes. UMI's provide estimations of absolute molecular counts. However, UMI's and spike-ins are not compatible with every single-cell technology (Chen et al., 2019; Haque et al., 2017).

Single-cell data is also typically sparse, this is caused by a high frequency of drop-out effects and the temporal fluctuation of gene expression (Haque et al., 2017). A drop-out event is the event of observing moderate or high expression levels of a gene in one cell, that is absent in another cell (Kharchenko et al., 2014). This is probably the result of bad detection of certain transcripts (Haque et al., 2017).

1.3 Cell type identification

1.3.1 Introduction

A single-cell analysis usually consists out of 4 phases: a data acquisition phase, a data cleaning phase, a cell assignment phase and a gene identification phase. In the first phase (data acquisition) the raw sequence reads are converted to a cell-gene expression matrix. To obtain this matrix, the transcripts are mapped to a reference transcriptome and the expression of all the genes is quantified. In the second phase, the data cleaning phase, a quality control is performed on the data. Low quality cells and uninformative genes are filtered out. Low quality cells are samples (cells) with

only a few number of reads, a very low mapping ratio or only a few detected genes. Additionally, batch effect correction is performed together with a normalisation to adjust for unwanted biases. This normalisation step can either be performed within one sample or between samples, depending on the objective of the analysis. Imputation of the missing values (dropouts) can also be performed, to benefit the further downstream analyses. The third phase is the cell assignment phase, in which all samples (cells) are assigned to cell types. This is usually done by dividing the samples in groups using an unsupervised clustering approach based on the gene expression patterns. All the samples in a cluster are then assigned the same cell type, which is found by manual inspection of the high differentially expressed genes and a thorough search in the literature for cell types that are characterised by the same differential expressed genes. In the last and fourth phase, it is then possible to try to answer earlier defined hypotheses by looking for interesting, differential expressed genes across different cell types, by looking for specific marker genes or by searching for certain, interesting expression patterns (Zappia et al., 2018; Chen et al., 2019; Diaz-Mejia et al., 2019).

The cell assignment phase is a complex phase. The manual assignment is a very time-consuming task, which gives results with a low reproducibility that are usually obtained using a non-standardised vocabulary. Meaning that it is almost impossible to use the datasets generated across different experiments or with different research groups in order to obtain correct conclusions. Furthermore, high quality curated datasets are sparse (Diaz-Mejia et al., 2019; Abdelaal et al., 2019). These caveats could be solved with fast, automated cell assignment. Automated cell type assignment can lead to reproducible cell labelling with a standardised vocabulary that is performed in a reasonable time frame, improving the dataset's quality and the data analysis in general.

1.3.2 Machine learning implementations for automated cell type identification

A lot of automatic cell identification methods already exist for single-cell RNA sequencing data, each differing in their classifying method and criteria. In general two types can be identified: the methods that use a labelled training dataset as starting point for classification such as scmap (Kiselev et al., 2018), CelFishing (Sato et al., 2019), scPred (Alquicira-Hernández et al., 2018), scReClassify (Kim et al., 2019) and SingleR (Aran et al., 2019). And methods like scCATCH (Shao et al., 2020) and CellAssign (Zhang et al., 2019) that use a specific set of marker genes per cell type, found in the literature, for classification (Bernstein et al., 2021).

The first type of methods, those that use a labelled training dataset as a starting point, operate in varying ways. Some start with a feature reduction step to reduce computation time before using machine learning techniques on expression data to actually perform the classification (Kiselev et al., 2018; Alquicira-Hernández et al., 2018). Others don't use a feature reduction step, but for instance allow reclassification after the initial classification to potentially correct for mislabelled cells in the labelled training dataset (Kim et al., 2019). Machine learning is a very vast field that encompasses many different classification techniques, of which multiple have already been implemented for cell type classification such as transfer learning (Lieberman et al., 2018; Hu et al., 2020), neural networks (Ma and Pellegrini, 2019) or the more classical classifiers like support vector machines (Alquicira-Hernández et al., 2018). But methods outside the domain of machine learning have also been applied. SingleR (Aran et al., 2019) is somewhat different from the techniques mentioned above, as it classifies entries based on comparison of the expression profiles of the entries with cell type specific profiles derived from bulk RNA datasets, not single-cell datasets. The biggest problem with this type of methods in general is that a high quality, curated and labelled dataset needs to be available. The quality of the classifiers generated from the labelled reference dataset will highly depend on the quality of this dataset. But highly quality, curated and labelled single-cell datasets are not common. This is why SingleR uses bulk RNA datasets. However, as mentioned previously, there are vast differences in characteristics of datasets generated with bulk or single-cell techniques (see Section 1.2.3).

Marker-based methods like CellAssign (Zhang et al., 2019) and scCATCH (Shao et al., 2020) depend on prior knowledge of cell type specific marker genes, and the quality of classification will highly depend on the chosen set of marker genes that will identify the different cell types. Difficulties can arise due to the fact that one cell type is commonly assigned to a set of multiple marker genes which are not always uniquely associated with that cell type. And so, finding a canonical set of markers for a cell type can be difficult (Shao et al., 2020; Zhang et al., 2018; Bernstein et al., 2021)

A small group of cell identification methods also take the cell type hierarchies into account. Flat classifiers, which don't take taxonomical information into account when assigning labels, will assign one cell type to every sample if the classifier predicted that cell type with enough confidence. Hierarchical classifiers will use the taxonomy or hierarchy present in the dataset as a starting point. They will build a tree with the different levels of hierarchy and will start classifying along the different levels of hierarchy in tree: starting at the top of the tree, which contains the most general labels, and continuing along the tree towards the bottom, which contain the most specific labels. At the end multiple labels will be assigned to the samples, each with

varying levels of detail, consistent with the label hierarchy present. For instance one cell will be labelled as an immune cell at the top level of the tree, a T cell at the second level of the tree and a CD4 T cell at the third and bottom level of the tree.

Garnett (Pliner et al., 2019) is a hierarchical marker-based method that requires a full definition of the hierarchy, with for every cell type specific marker genes, in order to classify single-cell data. CHETAH (de Kanter et al., 2019) and CellO (Bernstein et al., 2021) are hierarchical methods that use a labelled training dataset to build their classifiers. CHETAH (de Kanter et al., 2019) uses a labelled reference dataset for classification and will infer a hierarchy from the reference dataset based on correlations between self-build reference profiles per cell type. Classification is implemented using the similarities of the reference profiles of the cells and the cell types. CellO (Bernstein et al., 2021) uses Cell Ontology terminology to build a hierarchy and the classifier is trained on bulk RNA-seq training datasets for each cell type in the hierarchy. To be able to use the bulk-trained classifier on single-cell data, the single-cell data is first clustered and mean profiles are calculated per cluster. The samples in the cluster are then classified using this mean expression profile. Classification is implemented using two different approaches. The first approach uses a cascade logistic regression model for classification, where a binary logistic regression classifier is trained for every cell type (and thus in every node of the tree) on all the samples in the dataset belonging to the cell type's parents. The second approach uses independent one versus the rest binary classifiers for each cell type together with a follow-up correction method that reconciles the predictions with the Cell Ontology hierarchy.

Abdelaal et al. (2019) benchmarked 22 automatic cell type identification methods for single-cell RNA-seq data on 27 publicly available single-cell datasets. They concluded that a simple support vector machine approach with a linear kernel performed best overall. Among the cell type identification methods tested were Garnett, ACTINN, scmap, scPred, scCHETAH, SingleR and scID, which are all mentioned above, together with the following general classification methods: linear discriminant analysis, random forests, support vector machines with a linear kernel and k-nearest neighbours. It should be noted that all these methods were implemented with the default parameters.

1.3.3 Objectives and outline of this dissertation

The objective of this dissertation is to evaluate and implement hierarchical classifiers, which are already generally used in machine Learning, on single-cell data and explore their use for the automation of cell type identification of single-cell RNA sequencing data.

The following chapters and an overview of their content are listed down below:

- **Chapter 2: Machine Learning.** This chapter provides an introduction to machine learning and all the machine learning concepts that were implemented on single-cell transcriptomics data for automatic cell type assignment.
- **Chapter 3: Data and Methods.** This chapter gives an overview of the different datasets used in this dissertation together with more information on the implementation of all the analyses performed on these datasets.
- **Chapter 4: Results and discussion.** This chapter details and discusses the results from all the performed analyses.
- **Chapter 5: Conclusion and future work.** This chapter recapitulates the most important conclusions and suggests several future research paths.

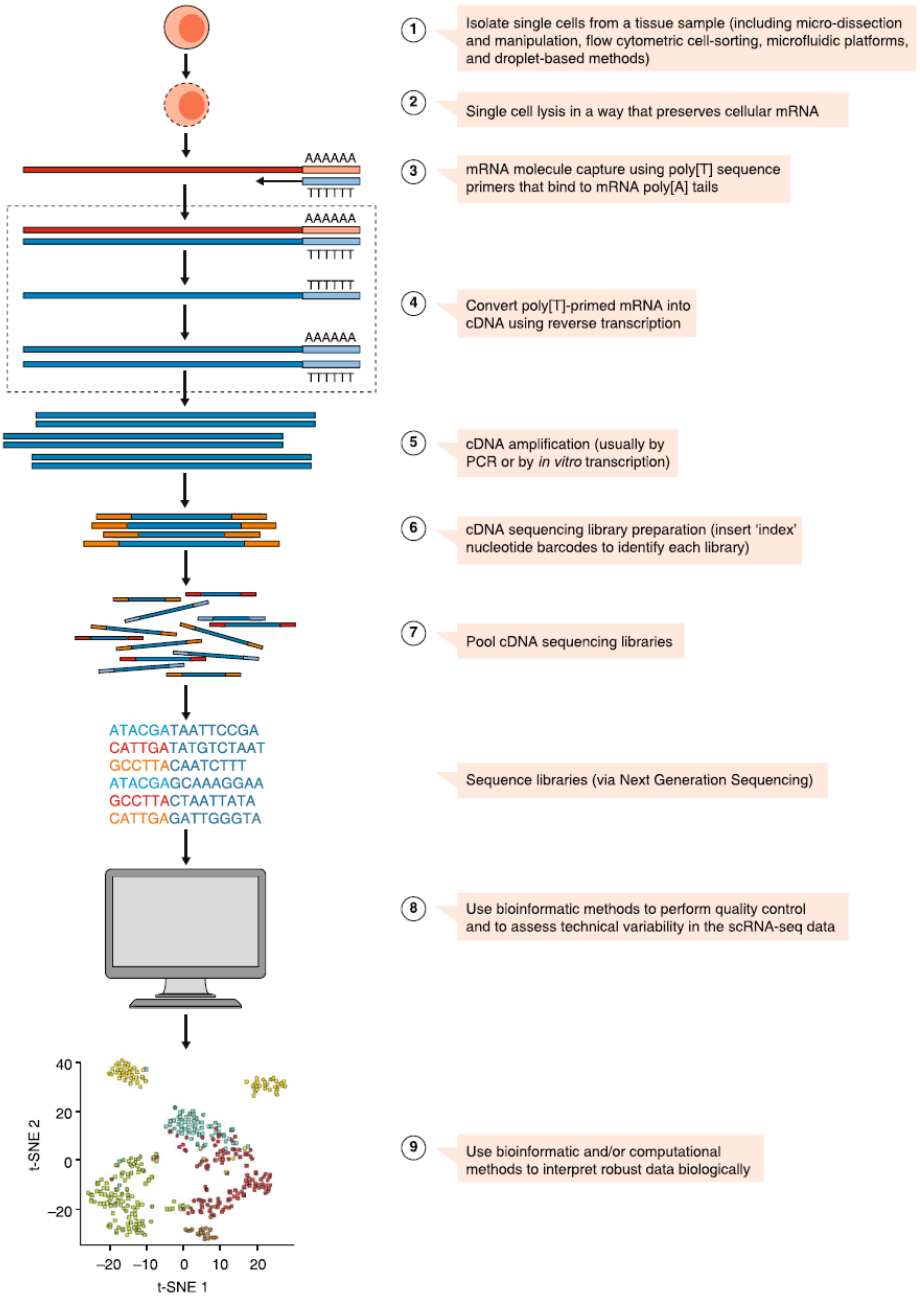

Figure 1.2: General workflow of a single-cell RNA sequencing experiment (Haque et al., 2017).

CHAPTER 2

MACHINE LEARNING

This chapter introduces the machine learning concepts implemented in this dissertation on single-cell transcriptomics data for automatic cell type identification. The purpose of this chapter is to give insight in these concepts to improve the reader's understanding of the following chapters.

2.1 Introduction

Machine learning builds mathematical models based on known information in order to be able to make predictions or decisions without being explicitly programmed how to do so (Zhang, 2020). Its goal is to use this known information or experience to improve performance or to make accurate predictions (Mehryar Mohri et al., 2018). A machine learning model learns from the known experience. This learning can be categorized into different scenarios depending on the type of data available. The two main scenarios are supervised and unsupervised learning.

Supervised learning can only occur when for each predictor's observation a label or response measurement is present. In a supervised learning scenario the goal is to relate the responses to the observations of the predictors to help predict future responses. The predictors are the input variables of the model and are also often referred to as the independent variables or features. In our case of cell type identification, the input will be a cell-gene matrix containing expression levels for different genes for every sample or observation. The cells will thus be the samples and the genes the features or predictors. The output variable is called the label, response or dependent variable and for automatic cell type identification this will correspond to the cell type labels.

Unsupervised learning occurs when unlabelled training data is present and seeks to understand the relationships between the observations. The goal of unsupervised learning may be to discover groups of similar samples within the data (clustering), to determine the distribution of data with the input space (density estimation) or to project the data from a high-dimensional space to two or three dimensions in order to visualise the data. In general, this scenario results in a better understanding of the

data. An intermediate of these two types exists as well and is called semi-supervised learning. Here one part of the training data is labelled, the other part is unlabelled. Other learning scenario's have been defined in the literature, but are not important in this context (Bishop, 2006; James et al., 2013; Mehryar Mohri et al., 2018).

Output variables can be either quantitative, which means that they are numerical, or qualitative and take on values of K distinct classes. Qualitative problems tend to be referred to as classification problems, while quantitative problems are known as regression problems. The cell type identification problem, which is at interest here, is a categorical or qualitative problem and thus a classification problem.

In this chapter the machine learning concepts that are implemented later on for automatic cell type identification are explained. For the cell type identification problem we will focus on supervised methods for classification. An overview of the following sections and their content is given below.

- **Section 2.2** describes some basic concepts of a supervised learning scenario:
  - It gives a more detailed description of the goal of a supervised method together with the distinction between parametric and non-parametric methods.
  - It introduces the concept of model assessment, the bias-variance trade-off and regularization.
  - It explains the use of resampling methods and K-fold cross-validation.
- **Section 2.3** details three supervised multi-class classification methods:
  - Logistic regression
  - Support vector machines
  - Random forests
- **Section 2.4** introduces hierarchical classification, the difference between local and global hierarchical classifiers and the local classifier per parent node approach.
- **Section 2.5** defines model performance metrics for binary, hierarchical and multi-class classification.

The following notation will be used in this chapter: n will represent the number of samples, observations or cells for single-cell datasets. The letter p will denote the number of independent variables, features or genes for single-cell datasets. The complete matrix with size nxp containing the observations on all the features will be denoted by

a capital letter X and the response variable will be denoted by a capital letter Y with one response observation depicted as y_i for i = 1,..,n. Features will be depicted as followed: $\mathbf{x}_1, ...\mathbf{x}_p$, with a single observation i of a feature $\mathbf{x}_1$ notated as x_{i1}.

2.2 Supervised learning

2.2.1 Model estimation

In a supervised learning setting we have access to a set of p features or predictors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p$ and a corresponding response Y measured on n observations. The goal is to fit a model $\widehat{f}(\cdot)$ that approximates the actual relationship $f(\cdot)$ between the response Y and the predictors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p$, with the aim to be able to make accurate predictions for future, unknown responses $Y^* \approx \widehat{f}(X)$. In a supervised learning setting the dataset will be split in a training and test set in order to be able to estimate the generalisation of the model towards future, unknown observations in an unbiased way. The model $\widehat{f}(\cdot)$ will be fit on the training dataset, where it will be taught how to estimate the actual relationship $f(\cdot)$, and the model's performance will be tested on the test dataset. We can estimate $f(\cdot)$ or thus obtain $\widehat{f}(\cdot)$ in two ways: using parametric methods or non-parametric methods.

Parametric methods first make an assumption about the functional form or shape of the actual relationship $f(\cdot)$ between observations and responses. This reduces the problem of estimating $f(\cdot)$ to the problem of estimating a rather limited set of parameters. But the potential disadvantage is that the assumed model shape can differ from the actual relationship $f(\cdot)$ between the observations and responses, which can result in poor predictions. Logistic regression and support vector machines with a linear kernel (see Sections 2.3.1 and 2.3.2) are examples of parametric methods.

Non-parametric methods do not make these explicit assumptions about the true relationship $f(\cdot)$, and are thus not limited by choosing a certain form of the model prior to fitting that model. They estimate $f(\cdot)$ by fitting a model as close as possible to the data points. To do this, non-parametric methods usually require a lot more training data to accurately estimate the shape of the relationship in the data approximated by the model (James et al., 2013). Random forests (see Section 2.3.2) is a non-parametric method.

2.2.2 Model assessment

To introduce the concepts of model assessment we assume for the moment that we have a regression problem at hand. The concept of the bias-variance trade-off introduced below is also applicable to a classification setting, but will only be explained in this section for a regression setting. With a regression problem, the measure of how close the predictions $\widehat{y}_i = \widehat{f}(\mathbf{x}_i)$ are to the future, unknown responses y_i^* is called the expected prediction error. This expected prediction error can be defined as:

$$E\left\{\left(\widehat{Y} - Y^*\right)^2\right\} = Var(\widehat{Y}) + Var(\epsilon) + bias^2 \tag{2.1}$$

where $\widehat{Y} = (\widehat{f}(\mathbf{x}_1), \widehat{f}(\mathbf{x}_2), ..., \widehat{f}(\mathbf{x}_n))$ are the predicted values, ϵ is the error term and $Y^* = (y_1^*, y_2^*, ..., y_n^*)$ are the unknown (future) responses. This equation shows that the expected prediction error can be decomposed into three terms: an irreducible error ($Var(\epsilon)$), the variance of the predictions ($Var(\widehat{Y})$) and the bias of the predictions ($Bias(\widehat{Y})$). The bias and variance terms are calculated in the following manner:

$$\begin{aligned} Bias\left(\widehat{Y}\right) &= E\left\{\widehat{Y}\right\} - E\left\{Y^*\right\} \\ Var\left(\widehat{Y}\right) &= E\left[(Y - E\left\{Y^*\right\})^2\right] \end{aligned} \tag{2.2}$$

The variance term refers to the amount by which $\widehat{Y}$ would vary when a different training dataset is used. The bias term is a measure for how close on average the predictions are to the ground truth.

The irreducible error, which is typically present due noise, will remain constant since it is independent of the model. A high bias originates from a model that does not decently capture the true relationship present in the data. This will usually be the result of a model with very few parameters. When too little parameters are incorporated in the model, the model will be a simplification of reality and will thus not underfit the data. The actual relationship will not be captured because of oversimplification. A high variance will occur when a lot of parameters are present in the model due to the fact that the model will not generalise to new data. The more parameters are included, the better the estimated relationship will fit to the (training) data points and thus the lower the (training) expected prediction error will be. But the training data is not an exact representation of reality, so if we adapt the model too much to the training data, the actual relationship will be lost. This phenomenon is called overfitting and will result in the fact that once the model's performance gets tested on the independent test dataset, the expected prediction error will be larger than previously estimated by the training data. The model complexity (number of parameters) will

be chosen as a trade-off between the bias and variance to obtain a minimal expected prediction error (see Figure 2.1) (James et al., 2013; Thas, 2020; Hastie et al., 2007).

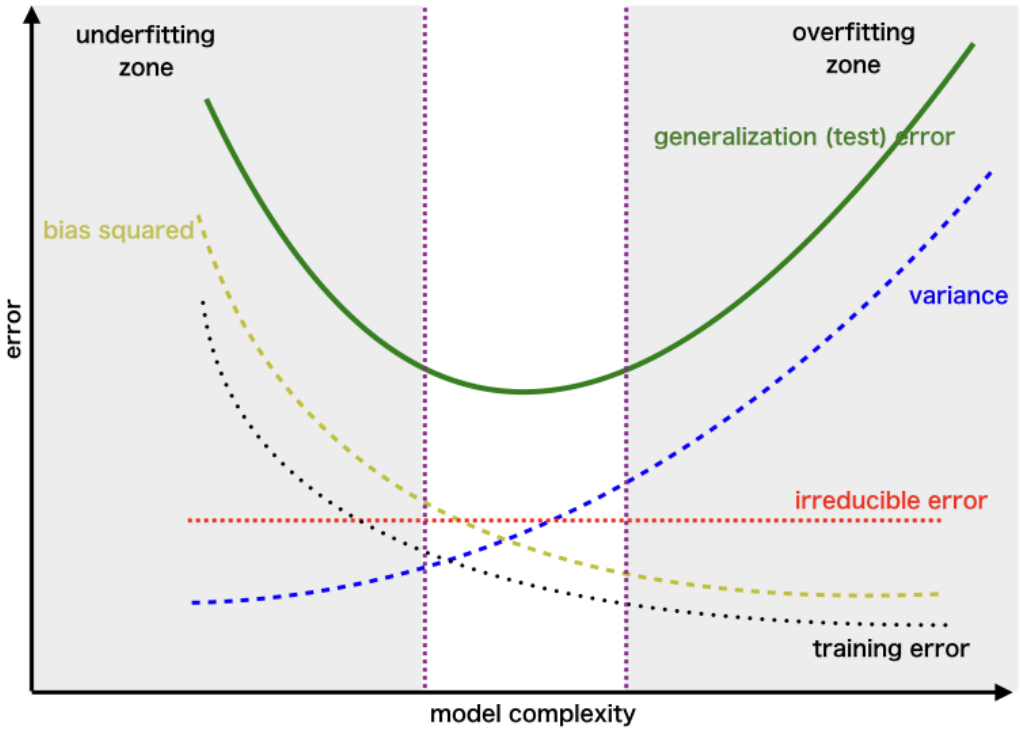

Figure 2.1: The influence of model complexity (the number of parameters included in a model) on the test error, training error, $bias^2$, variance and irreducible error (Papachristoudis, 2019).

In order to prevent overfitting, regularization is applied to penalise the amount of parameters used in the model. Regularization methods minimize the complexity of models by adding penalty terms to the model's objective that are dependent on the amount of parameters included in the model. As more parameters are included in the model, the penalty term will get bigger. The most used regression methods with regularization are Lasso regression, Ridge regression and Elastic Net regression. Ridge regression shrinks the coefficients of the estimated parameters towards zero with the use of an extra shrinkage term in the objective that needs to be minimised. If we assume here that the objective that needs to be minimised to obtain the most accurate predictions is the least squares criterion, the model that needs to be fitted will have the following form:

$$\hat{Y} = \hat{f}(X) = w_0 + \sum_{j=1}^{p} \mathbf{x}_j w_j, \tag{2.3}$$

with $X^T = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p)$ our predictors. The least squares criterion estimates those coefficients $w = (w_0, w_1, ..., w_p)^T$ that minimise the residual sum of squares:

$$RSS(\beta) = \sum_{i=1}^{N} \left(y_i - \hat{f}(\mathbf{x}_i)\right) = \sum_{i=1}^{N} \left(y_i - w_0 - \sum_{j=1}^{p} x_{ij} w_j\right)^2. \tag{2.4}$$

The ridge regression objective will then have the following form:

$$RSS + \lambda \sum_{j=1}^{p} w_j^2, \tag{2.5}$$

with the second term being the shrinkage term, and λ the penalty or tuning parameter. Ridge regression will return different estimated coefficients for different penalty parameters λ.

The Lasso regularization forces some coefficients to zero, unlike ridge regression which never sets the coefficients to zero. And thus the Lasso can result in sparse, less complex models and can be seen as a form of predictor or feature selection. If we assume the same model and objective as before with ridge regression, the Lasso is formulated as follows:

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|. \tag{2.6}$$

The Lasso penalty term is based on the l_1 norm, while the penalty term of ridge regression is based on the l_2 norm. Elastic net combines both the Lasso and the ridge regression penalty terms (James et al., 2013). All the models implemented in this thesis to automatically classify single-cell data will include approaches to avoid overfitting.

2.2.3 Resampling methods

When only a small dataset is available, it is impossible to split the dataset in a training and test dataset of sufficient size that are a decent representation of the population from which the samples were drawn. So, in order to still be able to confirm the model's performance on data independent from the training data, resampling methods can be used. The most commonly used ones are the cross-validation techniques.

Cross-validation allows for a reliable estimate of the expected test or prediction error of a model. Different forms of cross-validation exist such as Leave-one-out cross-validation or K-fold cross-validation. These two approaches are both similar, but K-fold cross-validation is computationally more feasible and is thus more commonly used.

K-fold cross-validation splits the training dataset in K equal-sized subsets or folds. Then each subset is used once as a test set for calculation of the test error and the model is fit on the other K-1 subsets. This results in K fitted models and test errors. The reliable test error estimate is then the average of the K test errors (James et al., 2013).

When a model is implemented that depends on a set of hyperparameters (parameters used to control the learning process of the machine learning method), the optimal values for these hyperparameters can be found by performing a hyperparameter tuning step. Hyperparameter tuning can be performed using a separate tuning or validation subset. This procedure is also referred to as nested K-fold cross-validation. The dataset will be separated in K equal parts or folds. A loop will be constructed over the K-folds and hold one subset separate for final testing. The remaining part of the dataset (K-1 folds) will be randomly divided in L equal parts and an inner loop over these L folds will hold one of the L folds separately as a tuning subset. The model will then be trained on the remaining L-1 subsets with different hyperparameters. The optimal hyperpamameter(s) will be decided by evaluating the different models on the validation or tuning subset. The performance of the best model can then be tested on the test subset (see Figure 2.2) (Waegeman, 2021).

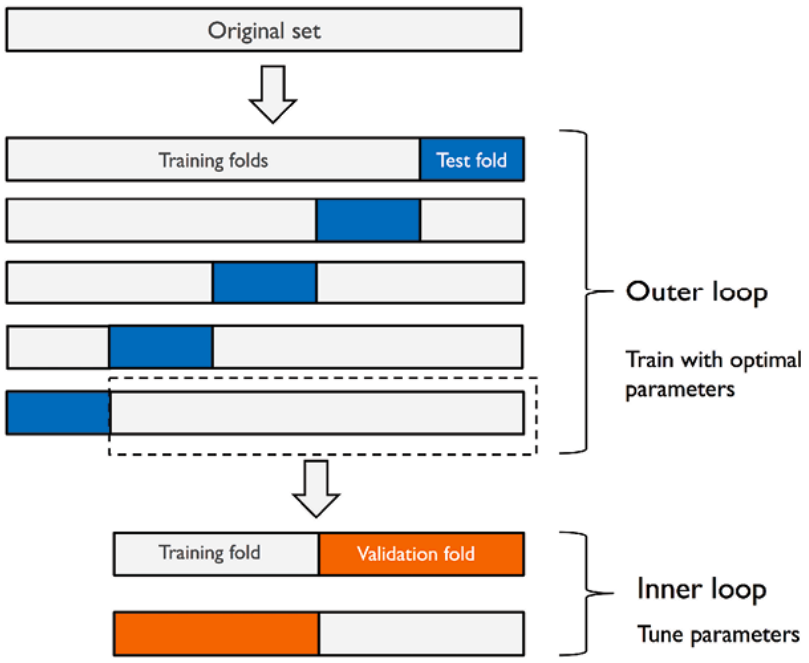

Figure 2.2: Nested cross-validation for hyperparameter tuning: K-fold cross-validation with a training, tuning and test dataset. In this example K = 5, since there are 5 folds constructed in the outer loop and L = 2, since the data present in the inner loop is splitted in 2 equal subsets (Kumar, 2020).

2.3 Supervised Multi-class classification methods

As mentioned earlier, the cell type identification problem is a classification problem, more specifically a multi-class classification problem: every sample is to be assigned one class or cell type from a group of possible candidate classes (> 2). In this section a couple of multi-class classification methods will be introduced that are later on implemented on single-cell datasets.

2.3.1 Logistic regression

Binary classification

Logistic regression is a classification method specifically designed for binary classification. Logistic regression will not model the response Y directly in function of the predictors $X = \mathbf{x}_1, ...\mathbf{x}_p$, but it will model the relationship between the probability that the response Y belongs to a particular category (for instance class 1), given the input variables, and the predictors X: $P(X) = P(Y = 1|X) = w_0 + w_1\mathbf{x}_1 + ... + w_p\mathbf{x}_p$. The logistic regression model will then output the conditional probability that a sample belongs to a certain class. A threshold can be set for the values of the conditional probabilities to assign the classes to the different samples. For instance all samples where $P(Y = 1|X) > 0.5$ will be assigned to class 1. To ensure that the output of the logistic regression model lies between 0 and 1, as expected for probabilities, the logistic regression model is implemented using the logistic function:

$$\phi(z) = \frac{e^z}{1 + e^z}, \tag{2.7}$$

The logistic model will then have the following form:

$$P(X) = \frac{e^{w_0 + w_1\mathbf{x}_1 + ... + w_p\mathbf{x}_p}}{1 + e^{w_0 + w_1\mathbf{x}_1 + ... + w_p\mathbf{x}_p}}, \tag{2.8}$$

which is equivalent to:

$$log\left(\frac{P(X)}{1 - P(X)}\right) = w_0 + w_1\mathbf{x}_1 + ... + w_p\mathbf{x}_p, \tag{2.9}$$

with p predictors $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p)$. The left term of Equation 2.9 was constructed by performing a logit or log-odds transformation on $P(X)$ and shows that the logistic model has a logit that is linear in X. Based on the output of the model, the conditional probability of a certain observation to belong to a certain class, class-membership can be assigned. The model will be built by estimating the coefficients $w_1, ..., w_p$ that

maximize the log likelihood of the data, which is an alternative to the least squares method mentioned earlier in Section 2.2.2. In practice a penalty term gets added to this model to prevent overfitting (see Section 2.2.2) (Hastie et al., 2007; James et al., 2013; Waegeman, 2021).

Multi-class classification

Logistic regression in the form discussed above is only valid for binary responses (0/1). Logistic regression can be adapted to a multi-class setting, where a sample can be assigned to more than 2 classes, by using various strategies.

A first strategy is the one-versus-one approach, where for every pair of classes a binary model is built on the observations of these two classes and predictions are made. The observation then gets assigned to its most predicted class. A second strategy is the one-versus-all approach where for every class a (binary) model is constructed that considers it's own observations as one class and all the other observations as another class. The final classification is then performed based on the conditional probabilities (Waegeman, 2021).

A third strategy is a specialised extension of logistic regression to a multi-class setting, called multinomial logistic regression. Suppose that K classes are present: $C_1, ..., C_K$. With multinomial logistic regression, one class gets assigned as reference class C_r and multiple binary logistic regression models will be build based on the reference class and the other classes, resulting in K-1 models of the following form (Forsyth, 2019):

$$X^T w_i = log\left(\frac{Pr(Y = i|X)}{P(Y = r|X)}\right), \tag{2.10}$$

which easily gives way to the predicted conditional probabilities of the different classes:

$$\begin{aligned} P(Y = 1|X) &= \frac{e^{X^T w_1}}{1 + \sum_{l=1}^{K-1} e^{X^T w_l}} \\ P(Y = r|X) &= \frac{1}{1 + \sum_{l=1}^{K-1} e^{X^T w_l}} \\ &... \\ P(Y = K|X) &= \frac{e^{X^T w_K}}{1 + \sum_{l=1}^{K-1} e^{X^T w_l}} \end{aligned} \tag{2.11}$$

And these probabilities can then be used to decide a sample's class assignment (Waegeman, 2021).

2.3.2 Support vector machines

The support vector machine is a generalisation of the maximal margin classifier that allows classification when the classes can not be separated by a linear boundary. Support vector machines were originally designed for binary classification but can be extended to multi-class classification and regression settings. Here they will be discussed in a binary classification setting. To broaden this to a multi-class setting, the one-versus-one and one-versus-all strategy mentioned in Section 2.3.1 can be applied. To properly grasp the concept of the support vector machine, understanding of the maximal margin classifier and support vector classifier is useful. So, these two classifiers will be introduced first.

For a p-dimensional dataset (n observations and p predictors), the maximal margin classifier will construct a hyperplane ($w_0 + w_1x_{i1} + w_1x_{i2} + ... + w_px_{ip} = 0 \ with \ i = 1, .., n$), which will act as a linear decision boundary that separates the training observations according to class labels and which will also be located as far away from the training observations as possible (see Figure 2.3). In order to construct this classifier, the concept of the margin M is introduced. The margin M is the minimal distance from the hyperplane to the observations. The goal is thus to construct a hyperplane or decision boundary that maximizes this margin and does not lead to misclassifications. This problem can be formulated as follows:

$$\begin{aligned} &\underset{w_0, w_1, \ldots, w_p}{\text{Maximize}} \ M \\ &\text{subject to} \sum_{j=1}^{p} w_j^2 = 1 \\ &Y_i\left(w_0 + w_1x_{i1} + w_1x_{i2} + ... + w_px_{ip}\right) \geq M, \ \forall \ i = 1, ...n \end{aligned} \tag{2.12}$$

The maximal margin classifier requires that all the observations are correctly classified. This is only possible when the classes are linearly separable, which is not often the case. If the classes are not linearly separable, the problem formulated above (Equation 2.12) becomes unsolvable and the maximal margin classifier cannot be constructed. When this is the case, support vector classifiers can be used.

The support vector classifier (SVC) is a variation on the maximal margin classifier that allows a couple of observations to be misclassified if the end-result, the overall classification, will benefit from it. A support vector classifier is constructed from the maximal margin classifier by adding slack variables $\xi = (\xi_1, \xi_2, ..., \xi_n)$ to the maximal margin optimization problem detailed in Equation 2.12, which allows for a certain

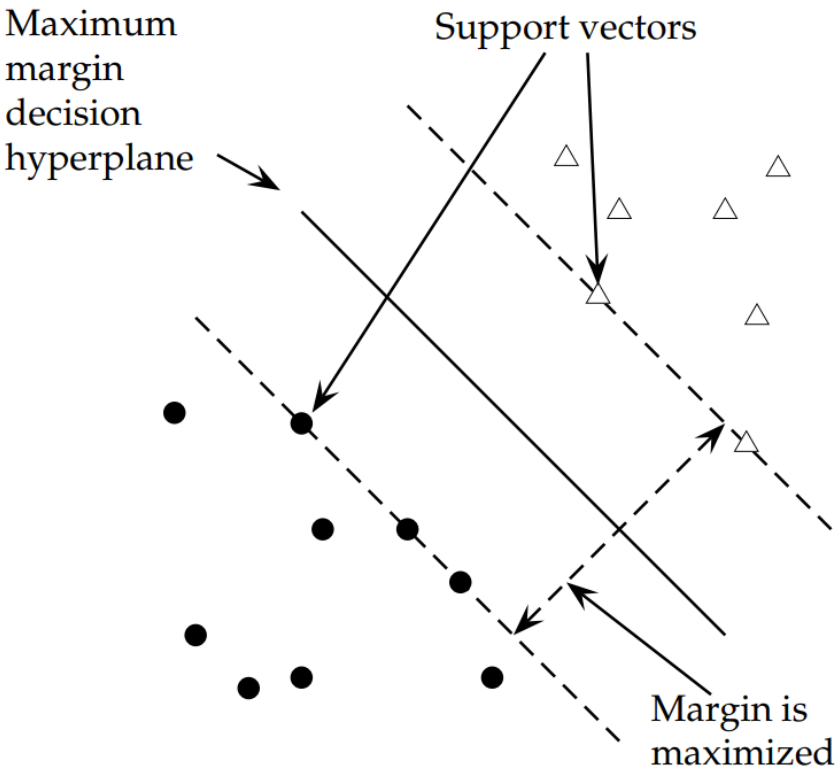

Figure 2.3: The margin, maximum margin decision plane and support vectors for a binary classification problem (Manning et al., 2009).

number of observations (C) to be on the wrong side of the hyperplane:

$$\begin{aligned} &\underset{w_0, w_1, \ldots, w_p}{\text{Maximize}}\ M, \\ &\text{subject to} \sum_{j=1}^{p} w_j^2 = 1, \\ &Y_i\left(w_0 + w_1 x_{i1} + w_1 x_{i2} + \ldots + w_p x_{ip}\right) \geq M(1 - \xi_i), \\ &\xi_i \geq 0,\ \sum_{i=1}^{n} \xi_i \leq C \end{aligned} \tag{2.13}$$

A large value of C will thus lead to many misclassifications and a smaller margin than a small value of C. The parameter C thus also controls the bias-variance trade-off (see Section 2.2.2): a small C leads to a high variance but a low bias, a large C has the opposite effect. A consequence is that a hyperplane constructed with a support vector classifier is only influenced by the observations that either lie on the hyperplane or are wrongly classified. These observations are called support vectors (see Figure 2.3).

The support vector classifier will only construct linear decision boundaries. To extend this method to also work with non-linear decision boundaries, an approach is

needed to make the procedure more flexible. For this purpose, the feature space can be enlarged by incorporating for instance quadratic or higher order polynomial functions or interaction terms of the predictors. In this enlarged feature space, the decision boundary will still be linear. But in the original feature space, the projected decision boundary (the projection of the hyperplane in the enlarged feature space to the original feature space) will be non-linear, which allows for more flexible decision boundaries. If the following original feature space is considered: $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p$. It is possible to fit a support vector classifier using 2p features: $\mathbf{x}_1, \mathbf{x}_1^2, \mathbf{x}_2, \mathbf{x}_2^2, ..., \mathbf{x}_p, \mathbf{x}_p^2$, instead of p features to enlarge the feature space.

The support vector machine (SVM) allows the feature space to become very large, sometimes even infinite. To make this computationally feasible, kernels $K(\cdot)$ are used to enlarge the feature space in a specific way. Kernels are functions that allow to perform operations in an enlarged feature space, without explicitly constructing this space. They are able to do this, because they only depend on the inner product of observations in the original feature space. Suppose that in order to construct a decision boundary with an SVC, we construct the following hyperplane: $w^T X - b = 0$ with w a vector of coefficients, but the data can not be correctly classified with a linear decision boundary. So, a SVM is implemented that constructs a non-linear decision boundary: $w^T \phi(X) - b = 0$ where the function $\phi(\cdot)$ transforms the data to an enlarged feature space where the data can be separated by a linear boundary. The computation of $\phi(\cdot)$ for every data point will be very expensive, since it involves computations in an enlarged and thus high-dimensional feature space. If instead of $\phi(\cdot)$ a kernel function is used, it is not needed to go to a high-dimensional feature space, since the kernel is solely based on the dot product of the data points in the original feature space and the calculations can be done in the original feature space. So, by using different kernel functions the feature space gets enlarged in a computationally feasible way (Waegeman, 2021).

In the support vector machines performed on the single-cell data later on, a linear kernel will be used. A linear kernel is just the dot product of the observations. This implies that a linear decision boundary is constructed (in the original feature space) and that the support vector machine in essence will do the same thing as a support vector classifier (James et al., 2013).

2.3.3 Random Forests

The random forests method is a tree-based method. With tree-based methods the feature space is divided into smaller, non-overlapping regions and the observations present in those regions are all assigned the same response. For regression purposes

this is usually the mean or mode of the observations in that region, for classification purposes this is usually a certain class.

The classification tree for a nxp dataset consisting of n observations on p predictors with labels corresponding to K classes, is built by recursively performing greedy, binary splits. Each split is chosen to obtain a maximum increase in the overall node or region homogeneity. This means that the region will be more homogeneous in its class or label distribution with as end goal only one class or label per region. This is done by using purity measures. One such purity measure is the Gini index (G):

$$G = \sum_{k=1}^{K} \widehat{p}_{mk}(1 - \widehat{p}_{mk}) \tag{2.14}$$

where $\widehat{p}_{mk}$ are the training observations in the m-th region, belonging to the k-th class. The resulting 'end'-regions will be called terminal nodes or leaves of the classification tree, the points along the tree where the binary splits were performed to the predictor space are called internal nodes. The Gini index is a measure of node purity. A small value means that predictors of the same classes are correctly divided in homogeneous regions depending on their labels.

To prevent overfitting of the tree (too many splits), tree pruning is performed. A classification tree is grown while taking a penalty term into consideration on top of the purity measure to perform the splits. This penalty measure will penalise based on the amount of leaves or lowest nodes present in the model. The resulting classification tree unfortunately will suffer from high variance: a different split of the training data will result in a very different tree. This occurs predominantly because of the hierarchy present in the classification process, which results in easy propagation of errors further down to lower level splits. So these classification trees usually have a high variance but a low bias when they are grown sufficiently deep. The internal nodes are prone to be variable, but when the tree is grown sufficiently deep, the end result (the classification at the lowest level of the tree) will be equal to other classification trees (if they are also grown to a sufficient level).

To decrease the tree's variance, bagging can be performed. Bagging consists out of taking repeated samples from the training dataset and using these samples as datasets to perform the tree building process. If m samples are taken, m classification trees will be built. The final bagged tree will then be built by the majority vote rule: the correct prediction is the most commonly occurring one over the m trees. This decreases the variance and will lead to a final, low bias, low variance classification tree.

The random forest approach has a slight improvement over the bagged tree approach. It decorrelates the trees before bagging, which improves the variance reduction significantly. The bagging procedure described above will average over m trees. These m trees will be correlated when a couple or one strong predictor is present in the dataset, because then a majority of the m trees will start with a split based on the same strong predictor, resulting in very similar trees. A random forest procedure will only allow a part of the predictors to be considered for the performance of binary splits, which decorrelates the trees (James et al., 2013; Hastie et al., 2007).

2.4 Hierarchical classification

Hierarchical classification can be applied when a predefined hierarchy is present in the classes or labels that are to be predicted. This hierarchy can be represented as a tree or a directed acyclic graph (DAG). DAG and tree structures are both graphs that consist out of nodes and edges. In the context of automatic cell type identification a node will contain a certain cell type class. A tree is a connected, acyclic undirected graph where a node cannot have more than one parent node, which is possible in a DAG (see Figure 2.4). Here, we will only consider tree structures, as they correspond to the biological nature of cell type hierarchies. In the tree or DAG representations, parent and children nodes are connected with edges that can be directed or undirected. The root node is the top node and encompasses all the classes and observations. The leaves are the nodes corresponding to the lowest level of classification. The other nodes besides the leaves and root node are called internal nodes.

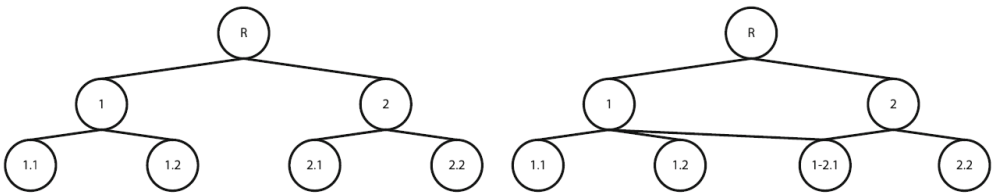

Figure 2.4: A tree structure (left) and a DAG structure (right) (Silla and Freitas, 2011)

Hierarchical classifiers can be subdivided into two groups: top-down or local classifiers and big-bang or global classifiers, which are further discussed below (Silla and Freitas, 2011).

2.4.1 Local classifiers

Local classifiers or top-down approaches start at the top of the hierarchy (in the root) where first for every observation the most generic class is predicted. Based on this result, a classification is then performed to assign class labels one level lower (from the set of class labels that have the first predicted labels as parents) and so on until the desired level of classification is predicted. This procedure ensures that no predictions are made that are inconsistent with the predefined hierarchy. The hierarchy is incorporated by only considering the local information per level, all the levels in the hierarchy are never considered at the same time during the process. The local information perspective leads to the disadvantage that errors are easily propagated through the entire tree. A mistake at a high hierarchical level can have severe repercussions at the lower levels.

There are three standard ways of using this local information: a local classifier per node (LCN) approach, a local classifier per parent node approach (LCPN) and a local classifier per level approach (LCL). Each of these methods differ in the amount of classifiers that are built and the nodes at which these classifiers are built (Silla and Freitas, 2011). For the automatic cell type identification problem a local classifier per parent node (LCPN) strategy will be applied.

Local classifier per parent node approach

With the local classifier per parent node approach, a multi-class classifier is trained for every parent node to assign children nodes to the observations classified under the parent node (see Figure 2.5). The training of the classifiers in every parent node is only performed on the observations that are labelled in the data with the same label as the parent node. This ensures that the trained model will adhere to the hierarchy present in the data (Silla and Freitas, 2011).

2.4.2 Global classifiers

With global classifiers, one global model or classifier is used to predict all the hierarchical labels of the observations. The entire hierarchy is taken into account by one global hierarchical model. The advantage of this is that the size of the total classification model will be a lot smaller than with an approach that uses only the local information. But the model will also be a lot more complex (Silla and Freitas, 2011).

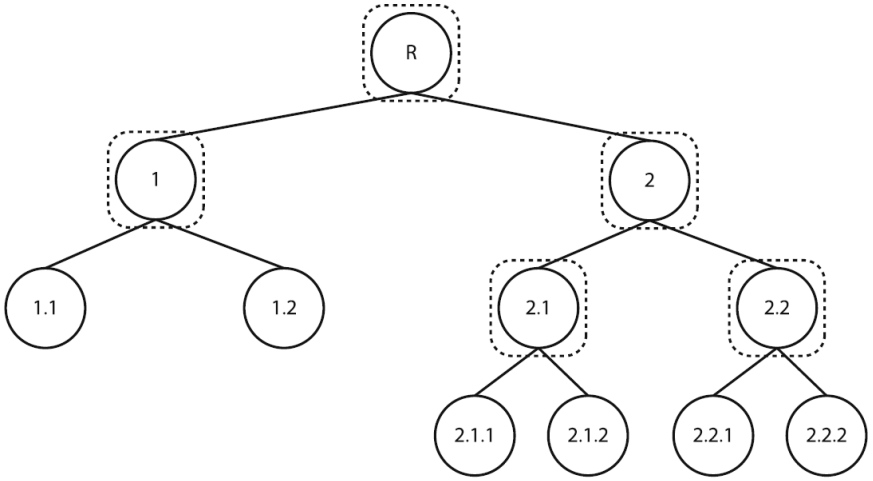

Figure 2.5: A visualisation of the local classifier per parent node approach. A classifier will be built in all the nodes which are circled with dashed lines (Silla and Freitas, 2011).

2.5 Model performance measurements

In order to compare different models, the performance of these models needs to be assessed. For this, different metrics can be used depending on the classification setting. First, metrics for a binary classification setting will be introduced. These metrics can then all be extended to a multi-class classification setting. The performance of the hierarchical classifiers will be estimated with the same metrics used to assess the performance of the multi-class classifiers.

2.5.1 Binary classification

In a supervised binary classification setting, metrics pertaining to the performance of classifiers are built up starting from a confusion matrix. A confusion matrix gives an overview of the amount of false positive (FP), true positive (TP), false negative (FN) and true negative (TN) predictions as shown in Figure 2.6.

Data class	Classified as pos	Classified as neg
pos	true positive (tp)	false negative (fn)
neg	false positive (fp)	true negative (tn)

Figure 2.6: A confusion matrix for binary classification (Sokolova and Lapalme, 2009).

Some of the metrics that are used a lot in a binary classification setting are accuracy, precision, recall and the F1-score. Accuracy describes the overall effectiveness of a classifier: how much observations were correctly classified?

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.15}$$

The recall or the sensitivity of a method gives the number of correctly identified positives divided by the total amount of positives present in the data. Or in other words, the proportion of correctly classified inputs in a class out of all the inputs belonging to that class.

$$recall = \frac{TP}{TP + FN} \tag{2.16}$$

Precision depicts the proportion of correct classified inputs of a certain class out of all the inputs that are classified in this class.

$$precision = \frac{TP}{TP + FP} \tag{2.17}$$

The F1-score is defined as the harmonic mean of precision and recall (Sokolova and Lapalme, 2009; Sokolova et al., 2006; Zhang and Zhao, 2015).

Accuracy and F1-score

The F1-score is often used as an alternative for accuracy as overall effectiveness metric, since accuracy can give problems when class imbalance is present in the dataset. Class imbalance occurs when there are large differences in class sizes in the data. If class imbalance is present, accuracy will be biased towards the majority class(es), the class(es) with the most amount of observations present, resulting in a classifier with a good accuracy score that might have trouble classifying the instances of all the small class(es). This effect is more notable in a multi-class setting, but can still be present in a binary classification setting. The F1-score will not be troubled by class imbalance. However, the F1-score is calculated independently of the amount of true negative predictions and shows asymmetric behaviour with respect to handling the positive and negative labels. So in a binary classification setting, the F1 score will be different when class swapping is performed (negative labels are replaced with positive labels and positive labels are replaced with negative labels). And thus the F1-score will depend on arbitrary choices (Sokolova and Lapalme, 2009; Yedida, 2016; Sokolova et al., 2006; Zhang and Zhao, 2015).

Balanced accuracy

An alternative metric, often used when imbalance is present, is balanced accuracy. For a binary classification setting, balanced accuracy is defined as the arithmetic mean of the sensitivity or the true positive rate (TPR) and the specificity or the true negative rate (TNR):

$$\begin{aligned} \text{Sensitivity} &= \frac{TP}{TP+FN} = \text{TPR} \\ \text{Specificity} &= \frac{TN}{TN+FP} = \text{TNR} \\ \text{Balanced accuracy} &= \frac{sensitivity + specificity}{2} \end{aligned} \tag{2.18}$$

Sensitivity reflects a classifier's probability to correctly predict a positive outcome, specificity reflects a classifier's probability to correctly predict a negative outcome. If this metric is used to assess a classifier that performs equally well on both classes, this metric reduces to the 'normal' accuracy. Meaning that if class imbalance is not present and all the classes have roughly the same size, accuracy and balanced accuracy tend to converge to the same value. Calculation of this metric is done by dividing the diagonal elements in the confusion matrix with their row sums and averaging the results (Grandini et al. (2020); Mosley (2013)).

Area Under the ROC Curve

Another metric that is often used is the Area under the ROC curve (AUC). This metric measures the area under the receiver operating curve (ROC), which plots the false positive rate (FPR) on the x-axis and the true positive rate (TPR), on the y-axis for varying decision or discrimination thresholds.

$$\begin{aligned} TPR &= \frac{TP}{TP+FN} = \text{recall or sensitivity} \\ FPR &= \frac{FP}{FP+TN} = 1 - \text{specificity} \end{aligned} \tag{2.19}$$

The AUC reflects the classifier's ability to correctly distinguish between classes. The higher the AUC value, the more correct classifications are performed and thus the better the predictive performance. An AUC value of 0.5 corresponds to a 45°-line in the ROC plot, which indicates that the performance of the classifier is equal to that of flipping a coin. An AUC value of 1 represents a perfect classification (Thas, 2020). The AUC is commonly used on imbalanced data, but the AUC metric can be unreliable in the case if severe imbalance when there are low sample sizes for minority classes.

There is no consensus about the optimal metric to evaluate a model's performance or to optimise a model's performance. For the optimisation process choices need to be made, for the final evaluation of the model it might be best to include several metrics as to truly give a complete view of the model's performance and avoid incorrect conclusions due to imbalanced data.

2.5.2 Extension towards multi-class classification

Based on the binary performance measures, similar measures have been designed for multi-class classification.

For a multi-class classification setting with l classes the following measures can be defined:

$$\begin{aligned}
\text{Recall}_{\mu} &= \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} (tp_i + fn_i)} \\
\text{Precision}_{\mu} &= \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} (tp_i + fp_i)} \\
\text{Recall}_{M} &= \frac{\sum_{i=1}^{l} \frac{tp_i}{(tp_i + fn_i)}}{l} \\
\text{Precision}_{M} &= \frac{\sum_{i=1}^{l} \frac{tp_i}{(tp_i + fp_i)}}{l}
\end{aligned} \tag{2.20}$$

with tp_i, tn_i, fp_i and fn_i respectively the number of true positives, true negatives, false positives and false negatives of a certain class C_i.

Precision and recall (and thus also the F1-score) can be calculated using macro- or micro-averaging (respectively indicated in Equation 2.20 with M and μ). With macro-averaging, precision and recall are calculated per class and then averaged. With micro-averaging, precision and recall are calculated based on the total amount of TP, FP, TN and FN over all the classes. So with macro-averaging, all the classes are considered equally important, while micro-averaging gives equal weight to all the inputs. The micro-averaged F1-score will thus also have problems when class imbalance is present in the data, seeing that whenever large classes will be correctly identified, a good performance will be concluded. Even when smaller classes samples are completely misclassified. (Sokolova and Lapalme, 2009; Sokolova et al., 2006; Zhang and Zhao, 2015).

In a multi-class classification setting, calculating the balanced accuracy score comes down to dividing the diagonal elements in the confusion matrix with their row sums

and averaging all the values, as done in the binary classification setting, which equals calculating the macro-averaged recall defined in Equation 2.19 (Grandini et al., 2020).

The extension of the AUC towards a multi-class setting is not quite that straightforward. Usually the implementation is performed with the use of the one-versus-one strategy or one-versus-all strategy, which was mentioned previously in Section 2.3.1 with the discussion of several strategies to extend binary classifiers for multi-class classification use. With these strategies multiple AUC scores are calculated based on binary comparisons, after which all the scores are averaged. With the one-versus-one strategy, AUC scores are calculated for all pairwise combinations between classes, with the one-versus-all strategy, the AUC is calculated for each class against the rest of the classes (Hand and Till, 2001).

CHAPTER 3

DATA AND METHODS

The following chapter will provide details about the datasets and methods used to answer the main hypothesis of this dissertation: Does incorporation of cell type hierarchy information in the classification process of single-cell data lead to better cell type annotation?

3.1 Datasets

In total four single-cell datasets that contain gene expression count data are used in this dissertation. Two of these are used to reproduce the results of the study by Abdelaal et al. (2019), where 22 automatic cell type identification methods for single-cell RNA-seq data were benchmarked: the Segerstolpe dataset (Segerstolpe et al., 2016) and the Baron Human dataset (Baron et al., 2016). These two datasets were randomly chosen from the 27 datasets that were used in the benchmarking paper and both of these datasets contain gene expression data of human pancreatic cells. The two datasets are open source datasets and can be obtained from the Hemberg Group at the Sanger Institute with the following link: `https://hemberg-lab.github.io/scRNA.seq.datasets/`. But for the reproduction analyses, filtered versions of these datasets are used, provided by Abdelaal et al. (2019). On these datasets, the following filtering steps were performed by Abdelaal et al.:

- The cells labelled as doublets debris or unlabelled cells were filtered out.
- The genes with zero counts across all the cells were filtered out.
- The median number of detected genes per cell was calculated and from that the median absolute deviation (MAD) was obtained across all cells in the log scale. All the cells for which the total detected number of genes was below three MAD from the median number of genes per cell were filtered out.

The label distributions of these two datasets can be found in Figure 3.1.

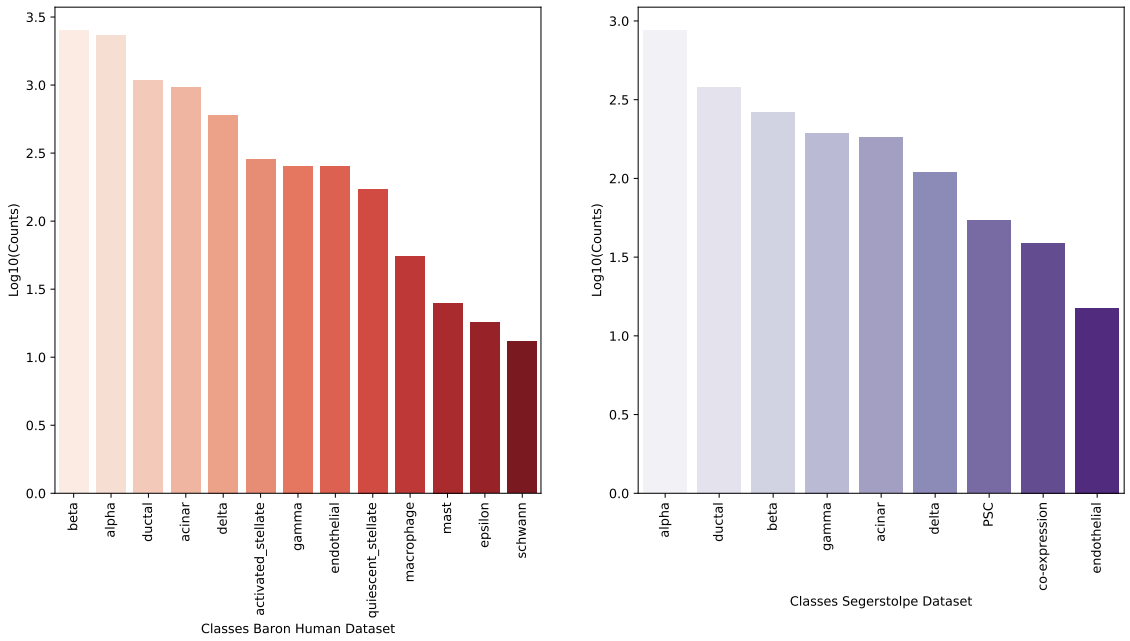

Figure 3.1: The logarithmic label distributions of the filtered Human Baron dataset (left) and the filtered Segerstolpe dataset (right). The height of the bars represents the log10-transformation of the total number of occurrences of a specific cell type (depicted on the x-axis) in the dataset.

Two other datasets are used to compare hierarchical classification with flat classification: The Allen Mouse Brain (AMB) dataset (Tasic et al., 2018), which contains single-cell gene expression information of several locations in the primary mouse visual cortex and was also used in the benchmarking study of Abdelaal et al. (2019), and the COVID19 dataset, which was provided by the Saeys lab at the VIB (Vlaams Instituut voor Biotechnologie) center. The COVID19 dataset contains gene expression data of single cells from lung tissue of a COVID-19 positive patient. The cells of the lung tissue were obtained with the bronchoalveolar lavage procedure.

The AMB dataset can be obtained from the Allen Institute Brain Atlas (`https://portal.brain-map.org/atlases-and-data/rnaseq`), but for this dissertation again the filtered version provided by Abdelaal et al. (2019) is used. The filtering details are the same as discussed previously for the other two datasets and details can be found in the previous paragraph. The AMB dataset contains the following hierarchical labelling: for each cell a cell type, subtype and cluster label is present. The label distributions of the AMB dataset for the three levels of hierarchy are visualised in Figure 3.2. The COVID19 dataset is currently not publicly available. More information on how this dataset was obtained can be found in Appendix C. The hierarchy of the COVID19 dataset is visualised in Figure 3.3 and the label distributions of the first three hierarchical levels are visualised in Figure 3.4. During the preprocessing of this dataset a couple of filtering steps were already performed:

- Empty droplets and outlier cells were identified and removed based on the gene expression profile.
- Cells with counts in less than 200 genes and genes expressed in less than 3 cells were removed from the count matrix.
- Cells that were more than 5 mean absolute deviations (MAD) from the median library size or median number of expressed genes were also removed, as well as cells where the percentage of mitochondrial reads exceeded the median by 5 mean absolute deviations.

For the COVID19 dataset a couple of extra filtering steps were also performed prior to classification. The cells that were labelled during the quality control were manually removed together with the cells that were labelled as undefined at the first level of the hierarchy. Furthermore, if a subset of cells belonging to a certain subtype had an extra label and thus an extra level in the hierarchy, but the other cells belonging to this subtype did not, the former's extra label was removed.

On top of the previously mentioned filtering steps, the cell populations with less than 10 cells were also manually excluded from all datasets. The characteristics of all the four datasets (after filtering) can be found in Table 3.1.

Datasets	**Number of cells (n)**	**Number of genes (p)**	**Number of cell populations**	**Tissue**	**Protocol**
Baron Human	8 562	17 499	13	Human pancreas	inDrop
Segerstolpe	2 111	22 757	9	Human pancreas	SMART-Seq2
AMB	12 781	42 625	3/16/92	Primary mouse visual cortex	SMART-Seq v4
COVID19	16 725	24 444	See Figure 3.3	Bronchoalveolar lavage fluid (BALF)	NovaSEQ S4

Table 3.1: The characteristics of the 4 filtered datasets used in this dissertation.

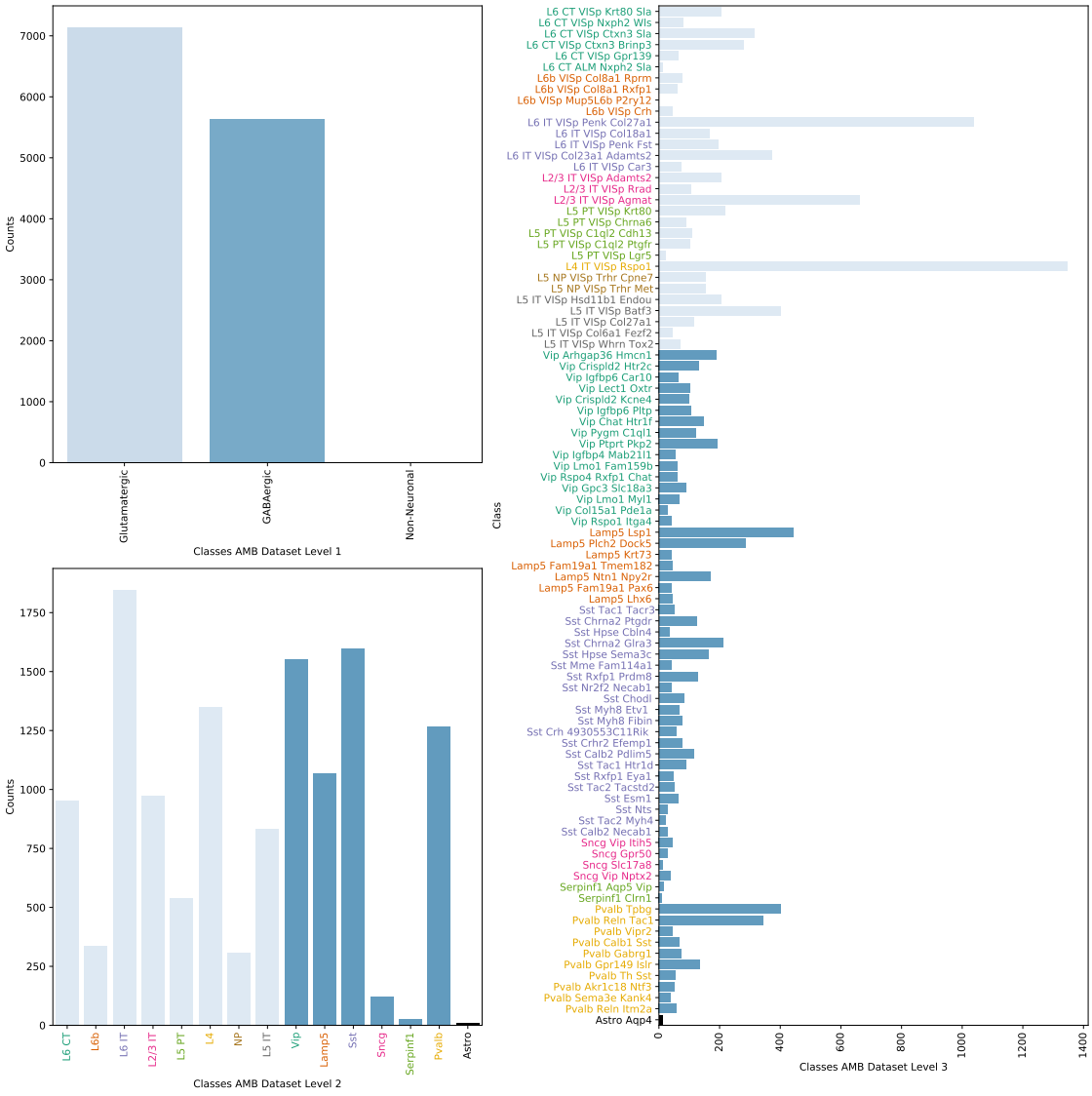

Figure 3.2: The label distributions of the three levels of hierarchy present in the AMB dataset. The height of the bars represents the total number of occurrences of a specific cell type at a certain level of hierarchy in the dataset. The label distribution of the first level of the hierarchy can be seen in the upper left plot, the label distribution of the second level of the hierarchy in the bottom left plot and the label distribution of the third level of the hierarchy in the right plot. The first level of the hierarchy is represented based on the bar colours in all the subplots, the second level of the hierarchy is represented on the third level hierarchy plot by the colours of the names depicted on the y-axis.

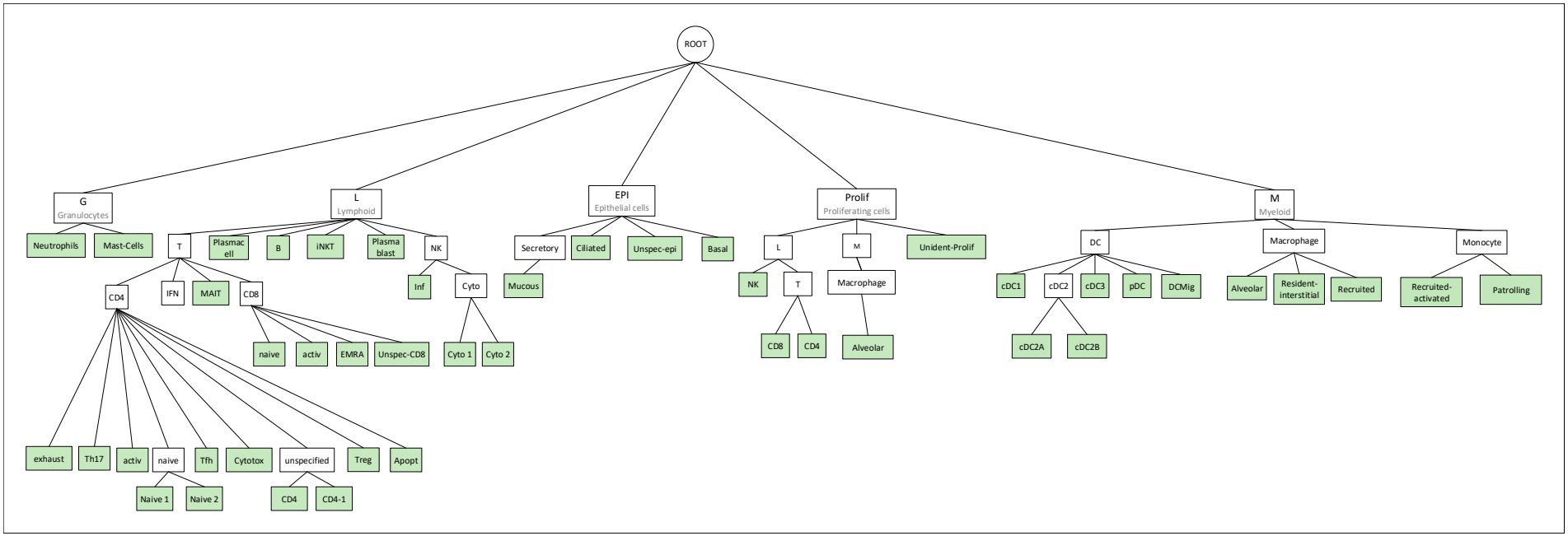

Figure 3.3: The predefined hierarchy of all the cell type labels present in the COVID19 dataset. The green-coloured nodes represent the leaves of this hierarchy, all these nodes contain the lowest level of classification possible for certain samples.

3.2 Methods

3.2.1 Reproduction of the benchmarking results

First, the results from Abdelaal et al. (2019) are reproduced. There, the authors concluded that the general-purpose support vector machine had the best performance across all the experiments they performed. Abdelaal et al. (2019) used multiple datasets to test for both the intra- and inter-dataset performance of the different classification methods. Here, only the intra-dataset performance is reproduced.

For the reproduction of these results, two randomly-chosen filtered datasets provided by Abdelaal et al. (2019) are used: the Human Baron dataset and the Segerstolpe dataset. Abdelaal et al. (2019) tested the performance of the different classifiers using ln(1+x) normalised data and implemented the different classifiers with default parameters and 5-fold cross-validation.

In this dissertation, the results of the linear SVM classifier and random forest classifier are verified with (stratified) nested k-fold cross validation and additional hyperparameter tuning for each fold. For the linear SVM classifier, optimisation of the regularisation parameter is implemented and for the random forest classifier the maximum depth of the trees is optimised. The option to not restrict the trees' depths with the random forest classifier and to grow the trees until all the leaves are pure is included during hyperparameter tuning. The details of the implementation of these classifiers can be found in Appendix B.2. For each classifier two analyses are performed: one with ln(1+x) normalisation and one with ln(1+x) normalisation and standardization. Standardization in this dissertation refers to the following transformation: $z_{ij} = \frac{x_{ij}-\mu}{\sigma}$, with z_{ij} the standardized value, μ the mean of all the training observations and σ the standard deviation of all training the observations. The performance is then evaluated in the same way as Abdelaal et al. (2019), by calculation of the median F1 score over all the cell type populations based on the pooled results of the 5 folds. As an additional metric the (pooled) accuracy score is calculated as well.

3.2.2 Testing the performance of flat and hierarchical classification

After reproduction of Abdelaal et al.'s (2019) results, flat and hierarchical classification are implemented on the two single-cell datasets with a hierarchy present in their labelling: the AMB dataset and the COVID19 dataset, to see whether incorporation of the cell type hierarchy in the classification process leads to improved results.

Flat classification

For flat classification, the labels with the highest level of detail are used. Classification is performed with three different classification methods: logistic regression, linear SVM and random forests and each of these classifiers is implemented with (stratified) nested 5-fold cross-validation and hyperparameter tuning. The hyperparameter tuning for the linear SVM and random forest classifier is performed as with the reproduction of the benchmarking results. For the logistic regression classifier, the regularisation parameter is optimised, similarly as with the linear SVM classifier. Details of the flat classification implementations can be found in Appendix B.2. For each of these classifiers two analyses are implemented: one analysis on ln(1+x) transformed data and one analysis on ln(1+x) normalised and standardized data. Performance of the different classifiers is measured with the following metrics:

- **Accuracy**: the average accuracy over all the 5 folds.
- **Balanced accuracy**: the average balanced accuracy over all the 5 folds.

More information on these metrics can be found in Section 2.5.

Hierarchical classification

Hierarchical classification is implemented on the hierarchical datasets with a local classifier per parent node (LCPN) approach (see Section 2.4.1). The same three classifiers as with flat classification are used: logistic regression, linear SVM and random forests. Again these analyses are performed with (stratified) nested 5-fold cross-validation and hyperparameter tuning (see Appendix B.2). For evaluation of classifier performance, the same metrics are used as with flat classification. But for hierarchical classification not one analysis is performed per setup, but several with different hierarchies:

- Hierarchical classification is performed with the predefined hierarchy, the hierarchy present in the labels provided together with the dataset. For the AMB dataset this is a three-level hierarchy, for the COVID19 dataset this hierarchy can be observed in Figure 3.3.
- Hierarchical classification is performed with a randomly generated hierarchy.
- For the COVID19 dataset, hierarchical classification is also performed with two newly-constructed hierarchies:

- A hierarchy where at the top a distinction is made between proliferating cells and not-proliferating cells. This hierarchy is visualised in Figure A.1.
- A hierarchy where the proliferating label is added at the bottom of the hierarchy as an extra label. This hierarchy is visualised in Figure A.2.

Feature selection

Lastly, feature selection is also implemented on the COVID19 dataset to evaluate whether this would improve classification. Feature selection is performed as a preprocessing step with the use of a threshold on feature importances implied by a random forest classifier. After this step, classification is implemented with the same classifiers as before: the linear SVM classifier, the random forest classifier and the logistic regression classifier. Evaluation of these classifiers is performed with (stratified) nested 5-fold cross-validation, where the tuning subset is split in two equal parts to render two independent tuning datasets: one for the feature selection step and one for the classification step. For the feature selection step, three different thresholds are tested: 0.3*mean, 0.7*mean and mean, with 'mean' referring to the mean value of all the feature importances assigned by the random forest classifier. For the classification step, hyperparameter tuning is performed similarly to the analyses before, but in a somewhat more limited range. Details on this can be found in Appendix B.2. Feature selection is performed on top of both flat classification and hierarchical classification with the predefined hierarchy. Performance of this strategy is evaluated with the average accuracy and the average balanced accuracy over the performed folds.

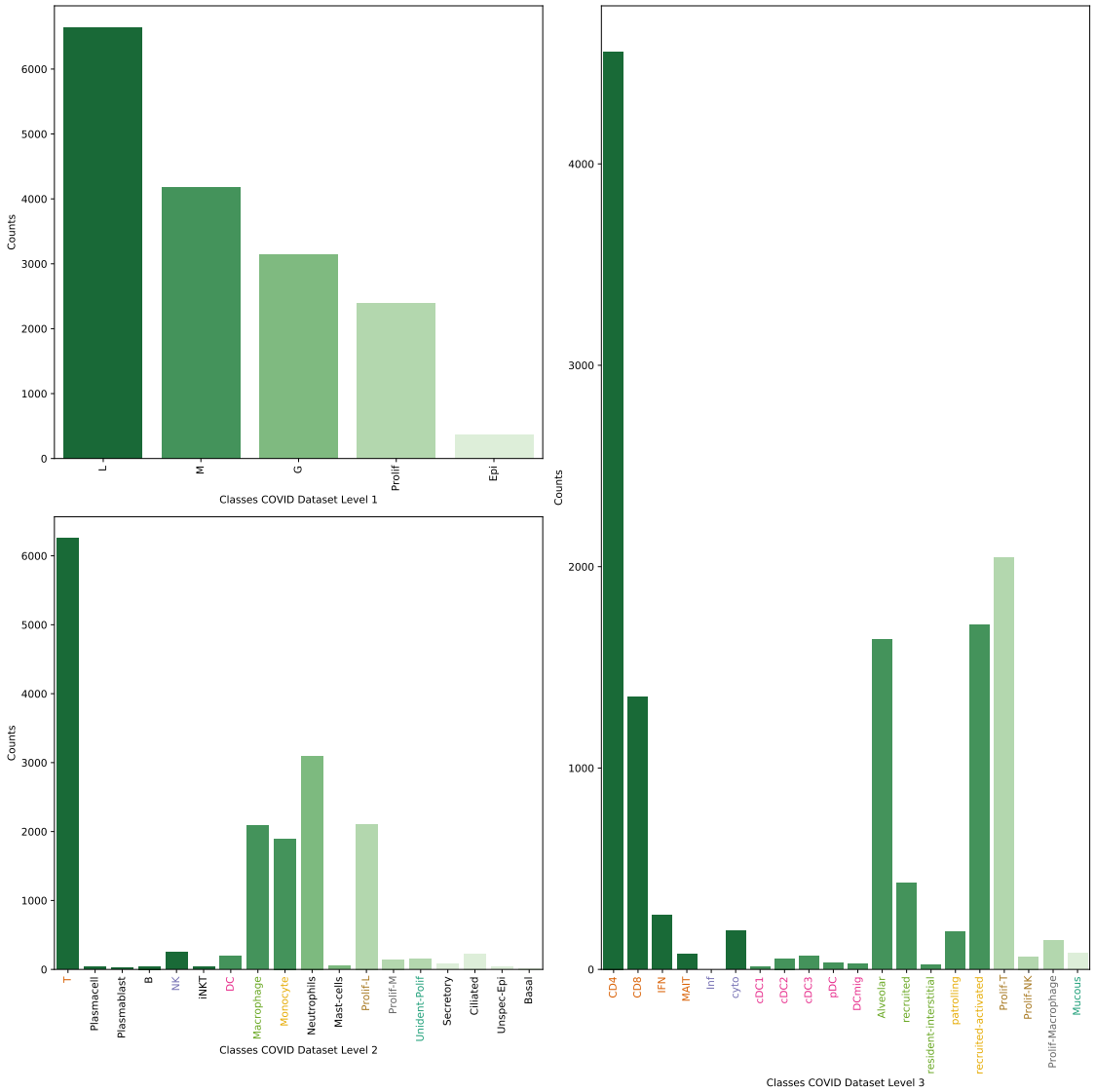

Figure 3.4: The label distributions of the first three levels of hierarchy present in the COVID19 dataset. The height of the bars represents the total number of occurrences of a specific cell type at a certain level of hierarchy in the dataset. The label distribution of the first level of the hierarchy can be seen in the upper left plot, the label distribution of the second level of the hierarchy in the bottom left plot and the label distribution of the third level of the hierarchy in the right plot. The first level of the hierarchy is represented based on the bar colours in all the subplots, the second level of the hierarchy is represented on the third level hierarchy plot by the colours of the names depicted on the x-axis.

CHAPTER 4

RESULTS AND DISCUSSION

This chapter presents and discusses the results from all the performed analyses described in Chapter 3.

4.1 Reproduction of the benchmarking results

First, reproduction of the benchmarking results from Abdelaal et al. (2019) is performed with two classifiers and two datasets. Abdelaal et al. (2019) assessed the performance of all the classifiers in their study by using the median F1-score over all the cell populations present in the different datasets. Their results on the Baron Human dataset and Segerstolpe dataset for the linear SVM classifier and random forest classifier can be found in Table 4.1. The obtained results by Abdelaal et al. (2019) for both classifiers are very good, with even a maximum median F1-score of 1 for the classification on the Segerstolpe dataset, generated with the linear SVM classifier. However, it should be noted that the median F1-metric takes the median over a number of different F1-scores. This means that a classification with a median F1-score of 1 can be imperfect. A dataset of 5 classes with class specific F1-scores of 0, 0, 1, 1 and 1 will also have a median F1-score of 1, even when 0 F1-scores were obtained for 2 classes, which indicates that for these classes not one sample was classified correctly. Because of this, the (pooled) accuracy score is also calculated with the reproduction of these analyses, since the accuracy score gives equal weight to every sample.

	Baron Human dataset	**Segerstolpe dataset**
Linear SVM	0,98	1
Random forests	0,94	0,98

Table 4.1: The median F1-scores obtained by Abdelaal et al. (2019) for classification of the Human Baron and Segerstolpe dataset with a linear SVM and random forest classifier. These results were extracted from Figure 1 in the benchmarking paper by Abdelaal et al. (2019).

The reproduction results can be found in Table 4.2. In the benchmarking paper the influence of standardization on the classification process was not evaluated. So, the

comparable values between Table 4.2 and Table 4.1 are the median F1 scores for the different classifiers on the normalised data depicted in Table 4.2. Comparison of these results shows great similarity between the newly-generated results and the results from Abdelaal et al. (2019). There is a small improvement in the median F1 score generated in this dissertation with the linear SVM classifier on the Baron Human dataset and a small decrease in performance for the Baron Human dataset with the random forest classifier in comparison to the results of the benchmarking study. But this can be caused by the different parameters used during the classification process as Abdelaal et al. (2019) used the default parameters for the different classifiers, which is not the case for this dissertation. More information on the implementation differences can be found in Section 3.2.1 and Appendix B.2.

			Baron Human dataset	**Segerstolpe dataset**
Linear SVM	Normalised data	Accuracy	**0,987**	**0,985**
		Median F1	**0,985**	**0,997**
	Normalised and standardized data	Accuracy	0,983	0,961
		Median F1	0,984	0,992
Random forests	Normalised data	Accuracy	0,965	0,978
		Median F1	0,932	0,980
	Normalised and standardized data	Accuracy	**0,966**	**0,981**
		Median F1	**0,942**	**0,995**

Table 4.2: The results of the reproduction analyses of the benchmarking paper by Abdelaal et al. (2019). Automatic cell type annotation of both the Baron Human dataset and Segerstolpe dataset is performed with the use of a linear SVM classifier and random forest classifier while assessing the importance of data standardization. The performance of these classifications is evaluated with the (pooled) accuracy score and the median F1-score over all the cell populations present in the dataset. The bold values indicate the best performance for the different metrics when comparing the classification performed with and without standardization of the data (so the best performance per metric within one quadrant of the table).

In general, the newly-generated results seem to reinforce the conclusion that the linear SVM classifier performs really well on single-cell transcriptomics data and not just in terms of the median F1-scores as concluded by Abdelaal et al. (2019), but also in terms of the accuracy scores. From the results in Table 4.2, it can also be concluded that standardization worsens the performance of the linear SVM classifier, but improves the performance of the random forests classifier.

In general a really good performance of both classifiers can be observed for these two datasets, with very high accuracy and median F1 scores. A possible explanation of this can be the limited complexity of the datasets present. As visible in Figure

3.1, both datasets contain a limited number of labels, which makes the classification process less complicated.

4.2 the AMB Dataset

After the reproduction analyses, the performance of flat versus hierarchical classification is assessed on the AMB dataset together with the influence of data standardization on all the different classification processes. For hierarchical classification, two hierarchies were evaluated: the predefined hierarchy present in the labelling of the dataset (see Figure 3.3) and a random generated hierarchy. The results of these comparison analyses, represented by the average accuracy and average balanced accuracy score over the 5 performed folds, can be found in Table 4.3. There, the best-performing scores over the three strategies are indicated in bold.

When comparing all the analyses in Table 4.3 , the following conclusions can be made:

- Standardization of the data worsens the performance of all the classifiers across the different strategies.
- Hierarchical classification with a randomised hierarchy generates worse results in comparison to the other two strategies.
- Hierarchical classification with a predefined hierarchy outperforms flat classification across the two metrics, albeit sometimes just slightly. The one exception to this, is the balanced accuracy score with the logistic regression classifier on data that isn't standardized. There, flat classification outperforms hierarchical classification with the predefined hierarchy.
- When comparing the three classifiers across flat and hierarchical classification with the predefined hierarchy, the random forests classifier performs worse than the other two.
- The best results are generated with the logistic regression classifier on non-standardized data. But it is hard to tell which strategy performs the best: flat classification or hierarchical classification with predefined hierarchy, since the accuracy scores are quite close. Furthermore, the best balanced accuracy score is generated with flat classification while the best accuracy score is generated with hierarchical classification that uses the predefined hierarchy, which gives a nuanced result.

			Flat classification	Hierarchical classification	
				Predefined Hierarchy	**Random Hierarchy**
Logistic regression	Normalised data	Accuracy	0,9050	**0,9054**	0,834
		Balanced accuracy	**0,801**	0,795	0,686
	Normalised and standardized data	Accuracy	0,845	**0,870**	0,754
		Balanced accuracy	0,690	**0,767**	0,581
Linear SVM	Normalised data	Accuracy	0,891	**0,899**	0,839
		Balanced accuracy	0,788	**0,793**	0,692
	Normalised and standardized data	Accuracy	0,867	**0,872**	0,772
		Balanced accuracy	0,711	**0,749**	0,600
Random forests	Normalised data	Accuracy	0,816	**0,838**	0,710
		Balanced accuracy	0,628	**0,649**	0,481
	Normalised and standardized data	Accuracy	0,813	**0,837**	0,705
		Balanced accuracy	0,620	**0,646**	0,475

Table 4.3: The automatic cell type annotation results on the AMB dataset. Automatic cell type annotation is performed with flat classification and hierarchical classification while making use of three different classifiers and assessing the influence of standardization on the classification processes. Two different hierarchies were considered for hierarchical classification: the hierarchy present in the labels of the dataset (the predefined hierarchy, see Figure 3.3) and a random generated hierarchy. Evaluation of the different classifiers' performance is assessed with the average accuracy over the 5 folds and the average balanced accuracy over the 5 folds. The bold values indicate the best obtained score for the respective metric when comparing the different classification settings (the best score per row in the table).

For the assessment of the performance of all the classifiers with all the different setups, both the accuracy score and the balanced accuracy score are used. As mentioned previously in Section 2.5.1, the balanced accuracy score is included since the accuracy score can sometimes give misleadingly-good scores when class imbalance is present in the dataset. The balanced accuracy score corrects for this by given equal weight to each class. As you can see in Table 4.3, the balanced accuracy scores are indeed lower than the accuracy scores and Figure 3.2 does show some class imbalance present in the AMB dataset. But the class imbalance is rather limited, because the smallest class in the first level of the hierarchy 'Non-neuronal' has only one class representation in the second and third level of the hierarchy and the presence of these classes is very small with only 11 samples (the total number of samples in the AMB dataset is 12 781). Care should thus be taken while interpreting the balanced accuracy score, because while it can deflate the overly good accuracy scores, it can also deflate the score too much when very small classes are present in the data. And this can lead the interpreter to believe that a classifier performs badly, while in reality it classifies the majority of the instances correctly.

Given the nature of hierarchical classification, it is expected to perform better on small classes since for the detailed labels, only a limited amount of options are still present to chose from while classifying the sample. So, if distinguishing features of the detailed labels are present in the data, you expect the classifier to detect these features more easily and if the smaller classes are classified correctly, the balanced accuracy score will improve. Table 4.3 shows indeed that hierarchical classification with the predefined hierarchy improves the balanced accuracy score for the different analyses, with the exception of the logistic regression classifier on non-standardized data. It seems that with this setup, some smaller classes were better classified with flat classification.

Another interesting phenomenon is that across all the analyses, standardization worsens the results even though standardization is a common preprocessing step to improve classifier performance. It is especially commonly used when classification is performed based on support vector machines, since the scale of the input values will largely influence the constructed decision boundaries of the classifier and thus the SVM results. In general, models that make use of distance measures are heavily influenced by the scale of input values. Moreover, standardization is highly recommended with the use of the stochastic gradient descent optimisation method, which is implemented in this dissertation for the linear SVM and logistic regressor classifier, since it eases the optimisation process (Ketkar, 2017). Thus, the current results for the logistic regression and linear SVM classifier are not really expected. A possible explanation as to why standardization worsens the classification process, comes with the

nature of single-cell transcriptomics data. Single-cell gene expression data is count data, with a lot of zero values and some very large values. It can be argued that the features with very large values, the genes for which a lot of counts were registered during sequencing, also contain the most information and are likely to be very important for classification. These large values will probably be influenced the most by standardization and they can lose their importance, particularly if a ln(1+X) normalisation is already performed prior to standardization. Therefore, it can be argued that the use of standardization possibly leads to a loss of information that is important for the classification process. For the random forest classifier, standardization is not expected to influence the results. However somehow, the performance does worsen slightly when standardization is performed, which gives further prove for the aforementioned hypothesis that standardization can lead to a loss of valuable information for the cell type annotation.

4.3 The COVID19 dataset

Similarly as with the AMB dataset, the COVID19 dataset also contains hierarchical labels and thus serves the main purpose of assessing the influence of incorporation of cell type hierarchy on the performance of classification. But for the COVID19 dataset this was not the only investigated effect. All the performed analyses can be categorized in three categories based on their purpose and will also be discussed separately before the main conclusions will be recapitulated. The three categories are:

- **Flat versus hierarchical classification**. Here flat versus hierarchical classification was performed similarly as with the analyses performed on the AMB dataset. For hierarchical classification two hierarchies were used, a random one and the predefined one that is visualised in Figure 3.3.
- **The influence of the proliferating label**. As visualised in Figure 3.3, a distinction is made in the predefined hierarchy on the first level between multiple labels with among these the proliferating label. All the labels on the first level pertain to a cell type, except for the proliferating label, which depicts a cell state. Thus, the question arises whether this distinction is correctly positioned within the hierarchy on the first level. In this section the proliferating label was moved within the hierarchy to find out how this influenced the classification results.
- **The influence of feature selection**. In this section feature selection was performed as a preprocessing step with flat classification and hierarchical classification with the predefined hierarchy, to see whether this improved the classification results.

4.3.1 Flat versus hierarchical classification

Table 4.4 depicts the results of the flat versus hierarchical classification comparison. Flat classification is implemented together with hierarchical classification with two different hierarchies both on standardized and non-standardized data. The evaluation of these classifications is performed by calculating the average accuracy and average balanced accuracy score over each performed fold. Hierarchical classification is, as mentioned before, performed with a random hierarchy and the hierarchy depicted in Figure 3.3. The bold values in the table visualise again the best performing strategy out of the three, for every setup.

Based on Table 4.4, the following statements can be formulated:

- Standardization worsens the results of the logistic regression classifier and the linear SVM classifier across the three strategies. For the random forest classifier this behaviour differed between the three strategies.
- Hierarchical classification with the predefined hierarchy performs better than the other two strategies, except when applied with the logistic regression classifier on standardized data, then the accuracy score decreases slightly.
- Hierarchical classification with a random hierarchy performs worse than the other two strategies across all the performed analyses.
- With flat classification, the best performing classifier on non-standardized data is the logistic regression classifier. For hierarchical classification with the predefined hierarchy this is the logistic regression classifier based on the accuracy score and the linear SVM classifier based on the balanced accuracy scores, but between these classifiers, the two metrics only differ with 0,001. So, it can be stated that the two classifiers perform more or less equally in this setting and that the overall best performance is observed with both the logistic regression classifier and the linear SVM classifier used for hierarchical classification with the predefined hierarchy.

			Flat classification	Hierarchical classification	
				Predefined Hierarchy	**Random Hierarchy**
Logistic regression	Normalised data	Accuracy	0,775	**0,782**	0,710
		Balanced accuracy	0,547	**0,561**	0,432
	Normalised and standardized data	Accuracy	**0,737**	0,735	0,673
		Balanced accuracy	0,413	**0,551**	0,378
Linear SVM	Normalised data	Accuracy	0,771	**0,781**	0,720
		Balanced accuracy	0,526	**0,562**	0,420
	Normalised and standardized data	Accuracy	0,742	**0,745**	0,687
		Balanced accuracy	0,428	**0,505**	0,379
Random forests	Normalised data	Accuracy	0,670	**0,678**	0,658
		Balanced accuracy	0,275	**0,323**	0,285
	Normalised and standardized data	Accuracy	0,667	**0,681**	0,658
		Balanced accuracy	0,283	**0,331**	0,284

Table 4.4: The results of the comparison of flat and hierarchical classification with the predefined hierarchy (Figure 3.3) and a random hierarchy on the COVID19 dataset. For all classification settings (flat versus hierarchical classification), three different classifiers are implemented and the influence of standardizing the data is also assessed. The performance of the different classifications is evaluated by calculating the average accuracy and average balanced accuracy over the 5 folds. The bold values indicate the best obtained score for the respective metric when comparing the different classification settings, so the best values per row in the table.

Similarly to the results on the AMB dataset, standardization also worsens the performance of the logistic regression and linear SVM classifier. For the random forest classifier this is mostly the case, with an exception for the random forest classifier in the hierarchical classification setting with the predefined hierarchy where an improvement is observed with standardization. But, as mentioned previously, standardization is expected to influence the random forest classifier to a lesser manner. So the hypothesis presented while discussing the AMB dataset, which states that standardization can interfere with important signals present in the features and in this way negatively influence the classification, can also here give a possible explanation for the observed results.

With all the analyses performed on the COVID19 dataset, the balanced accuracy scores are lower than the accuracy scores, as observed with the AMB dataset. But the differences are a lot bigger and looking at the balanced accuracy scores it could seem like the different classifiers perform badly on the COVID19 dataset. As mentioned with the AMB dataset, the balanced accuracy score gives equal weight to each class and corrects for inflation of the accuracy score when class imbalance is present. As visualised in Figure 3.4 for the first three levels in the predefined hierarchy, there is a lot of imbalance present in the COVID19 dataset. In the COVID19 dataset, some samples have up to 5 levels of labelling and thus this imbalance will probably be even worse than showed in Figure 3.4 at the lowest levels of labelling, resulting in larger corrections with the balanced accuracy score. However, the accuracy scores of the different analyses are not at all bad. This implies that the classifiers don't have a bad performance when the number of correctly classified samples are observed, but that the classifiers struggle with the smaller classes, of which quite a lot are present in the COVID19 dataset.

Another interesting observation worth mentioning is that classification of this dataset seems to be more difficult as the accuracy scores and balanced accuracy scores are lower than for the previous three datasets. This could be caused by the increase in complexity present in the COVID19 dataset as presented in Figure 3.3. While there are less lower level classes than the AMB dataset, it seems like the relations between the different entries in the COVID19 dataset are more complicated. The AMB dataset has three cell types on the first level in the hierarchy: 'GABAergic', 'Glutamatergic' and 'Non-Neuronal', which represent respectively GABAergic neurons, Glutamatergic neurons and Non-neuronal cells. GABAergic and Glutamatergic neurons are two major neuronal classes, which establish inhibitory and excitatory synapses respectively and have differences in anatomy, physiology and developmental origin (Turko et al., 2019). So, given the functional differences and the differences mentioned before, you would expect very different expression profiles across the genes for these three

cell types and thus an easier classification. Furthermore, the third level labels (the clusters), refer to the different areas in the brain where cell sampling occurred (Baron et al., 2016). Again it seems not unreasonable that different areas in the brain would be susceptible to detectable changes in gene expression. In comparison, the first level hierarchy of the COVID19 dataset contains the distinction between three types of immune cells: myeloid cells, lymphoid cells and granulocytes together with epithelial cells and the proliferating label which depicts a cell state. The three types of immune cells contain most of the samples, and are then classified into more detailed subtypes with sometimes up to 5 levels of labellings. The immune system is quite an intricate system that makes use of a very broad range of specialised cells all with similar functions. So, given this labelling of the COVID19 dataset, it seems not unreasonable to think that these distinctions are harder to pick up and thus more difficult to classify.

To get more insight in the classification process, confusion matrices are constructed for the first three levels of the hierarchy, comparing flat classification and hierarchical classification with the predefined hierarchy. These confusion matrices were constructed for the classifications performed with the logistic regression classifier on non-standardized data. It should be noted that in these figures, the absolute counts are depicted and that the total amount of samples for this dataset is 16 725. Based on these visualisations, the following observations can be made:

- For level 1 (Figure A.3):
  - Flat classification seems to have a bit more problems with distinguishing the samples in the proliferating state than the hierarchical classifier with the predefined hierarchy.
- For level 2 (Figure A.4):
  - Both classifiers have problems distinguishing the monocyte cells from the macrophage cells.
  - On this level, the hierarchical classifier also performs a little better with the classification of the proliferating cells (this can however be a propagated result downwards from level 1).
- For level 3 (Figure A.5):
  - A couple of confusions occur between the different T cell subtypes::
    - the MAIT subtype gets often confused with the CD4 and CD8 subtype.
    - The CD4 subtype and CD8 subtype also get confused, but to a smaller extent.

    - the IFN subtype is quite often misclassified as the CD4 subtype.
- Not one instance of the resident-interstitial subtype of the macrophage cells gets classified correctly with the hierarchical classification with predefined hierarchy. Only one instance gets correctly classified with flat classification. Instead the most assigned subtype is 'Alveolar'.
- The monocyte cells that were wrongly assigned to the macrophage class in level 2 are mostly classified as the alveolar subtype of macrophage cells, which is the subclass of macrophage cells with the largest amount of samples present in the COVID19 dataset.

Some of these observations raise questions as to whether or not they make sense biologically. In order to answer these questions, more information is given down below in concern to the statements regarding the confusion between different cell type labels.

On the second level in the hierarchy, the classifiers seem to have problems with distinguishing monocyte and macrophage cells. Monocytes and macrophages are cells of the innate immune system, which originate from a common myeloid progenitor cell. Monocytes are differentiated myeloid cells that under normal circumstances circulate in the bloodstream for a short period of time before undergoing apoptosis. In response to external or internal differentiation signals, monocytes differentiate into macrophages that have a longer life span and can exit the blood stream. Macrophages can thus be present in almost every organ. Monocytes and macrophages share a number of similar functions and are responsible for surveillance to detect the presence of pathogens and for initiation of the inflammatory response (Parihar et al., 2010). Knowing this, it is unsurprising that these two cell types get confused by the classifier.

More surprising is that the distinction between CD4 and CD8 T cells is sometimes not clear. Both cells do share common characteristics and a similar origin, like most immune cells, but these two T cell subtypes have very different functions. CD4 T cells are helper or inducer cells that help establish the immune reaction, while CD8 T cells are cytotoxic T cells that will kill the infected cells (Miceli and Parnes, 1993). However, relatively speaking, only a small part of the total number of samples belonging to these classes gets misclassified. So, maybe an explanation for the confusion between these two cell type lies with the classifiers' discriminative power instead of the biological content of the data.

On the third level of the hierarchy, problems arise with the classification of MAIT cells, CD4 T cells, CD8 T cells and IFN T cells. For each of the specific confusions listed above, explanations can be found in the biology of the immune system. One of these

observations is that MAIT cells tend to get misclassified as either CD4 T cells or CD8 T cells. MAIT cells or mucosal associated invariant T cells are unconventional innate-like T cells that are defined by a very specific T cell receptor. The T cell receptor on T cells is responsible for the recognition of antigen fragments presented by major histocompatibility complex (MHC) proteins. MAIT cells recognize very specifically microbial riboflavin metabolites presented by MHC I proteins. But they also express CD4 and CD8 receptors like the conventional CD4 and CD8 T cells with the conventional T receptor to then perform a helper or killer function (Hinks and Zhang, 2020). So, the MAIT cells will probably have similar expression profiles to normal CD4 and CD8 T cells (depending on whether they are MAIT CD4 or MAIT CD8 cells), which will result in a more difficult classification.

IFN T cells were also often misclassified as CD4 T cells. The IFN sub-label assigned to T cells indicates that the T cells are interferon gamma (IFN-γ) producing T cells, which is again a special type of T cells. Gamma or immune interferon interacts with specific cellular receptors, which promote the production of second messengers, ultimately leading to the expression of antiviral and immune modulatory genes. IFN-γ can be secreted by CD4 T cells and CD8 T cells (Le Page et al., 2000; Castro et al., 2018). Since interferon gamma production will most likely only alter the expression of one gene, at most a couple of genes. It is possible that there is more evidence present in the data that points towards classification of a sample as a CD4 T cell instead of an IFN-γ producing T cell. It can be argued based on the knowledge above that the MAIT cell label and IFN T cell label are not properly positioned in the current hierarchy and that an extra distinction above the CD4 and CD8 labels, between conventional T cells, MAIT cells and IFN T cells could possibly improve the classification.

Lastly, it is also observed that the classification of resident-interstitial macrophages is very hard, with none and one correctly classified sample for the hierarchical classification strategy with predefined hierarchy and the flat classification strategy respectively. Macrophages are the most abundant immune cell population in healthy long tissue. And based on their location they can be classified as alveolar macrophages, when they are present in the alveolar and airway lumen, or as interstitial macrophages, when they are present in the long tissue interstitial (Liegeois et al., 2018). A recruited macrophage is a macrophage that was recruited from elsewhere as part of a local immune response in the long tissue. It seems logical that recruited macrophages show a different expression profile. But since both alveolar macrophages and resident-interstitial macrophages are constantly present in long tissue, it seems not unreasonable to think that maybe the data did not contain enough distinguishable features for a clear classification of these two subtypes.

4.3.2 The influence of the proliferating label

A second set of analyses are performed on the COVID19 dataset to see how the position of the proliferating label in the hierarchy influences the classification results. As already briefly mentioned previously, in the predefined hierarchy (Figure 3.3) the proliferating label is present on the first level. However, the proliferating label does not depict a cell type, like the other labels in the hierarchy, but a cell state. Cell proliferation is the process of cell growth (both an increase in mass and size) and duplication of the cell's content, followed by cell division (Conlon and Raff, 1999). In multicellular organisms, this process occurs at different times and rates for different cells and cell types depending on the needs of the entire organism and the capability of the individual cells. A cell can either be in a non-proliferating state or stable state, which is the case for most cells, or undertake the process of the cell cycle and enter the proliferating state (Yang et al., 2014).

Since the proliferating label refers to a cell state rather than a cell type, this label can be seen as a parallel label, because every cell belonging to a specific cell type can be in the proliferating cell state. Regularisation of cell proliferation occurs in all mammalian cells by making use of various signal pathways that detect developmental cues, growth factors, DNA damage, ... (Duronia and Xiong, 2013). Since all the different mammalian cell types use the same pathways to regulate cell proliferation and the same process to proliferate, it is expected that a common set of genes across all the cell types can be correlated with cell proliferation and that maybe even common expression patterns can be found across the cell types. If this would be the case, it would be possible to make a distinction between proliferating and not-proliferating cells across all the cell types, at the top of the hierarchy. This could possibly improve the classification results obtained with hierarchical classification in comparison to the hierarchy used previously where no distinction was made between cell type and cell state labels. If this would not be the case and if certain cell types would have cell type specific gene sets responsible for cell proliferation, it would be better to make a distinction between proliferating and not-proliferating cells after cell type classifications, within the group of cells belonging to one cell type.

So, two different hierarchies were constructed: one with a distinction between proliferating and not-proliferating cells at the top of the hierarchy, prior to cell type classification (this hierarchy is visualised in Figure A.1). And a hierarchy where the distinction between proliferating and not-proliferating cells is made after cell type classification: a hierarchy where the proliferating labels are added at the bottom (see Figure A.2). The performance of hierarchical classification with these two newly-constructed hierarchies is then compared to the performance of hierarchical classification with the

predefined hierarchy (Figure 3.3) with the help of the average accuracy and balanced accuracy scores across the performed folds. Classification is performed with the same three classifiers as before: the logistic regression classifier, the linear SVM classifier and the random forest classifier. The influence of standardization on the classification results is again assessed. The results of these analyses can be found in Table 4.5, where the results of hierarchical classification with the predefined hierarchy are recapitulated for comparison purposes. Based on this table, the following conclusions can be formulated:

- Standardization worsens the results of the linear SVM and logistic regression classifier across the three strategies. For the random forest classifiers a uniform conclusion across the three strategies cannot be made.
- For the linear SVM and logistic regression classifier the following observations are made in concern to the performance of the three strategies:
  - A clear trend is visible for both classifiers implemented on standardized data: the best accuracy score is obtained with the proliferation distinction at the bottom of the hierarchy, the best balanced accuracy score with the predefined hierarchy.
  - For non-standardized data and with the use of the linear SVM classifier, the accuracy scores obtained with the predefined hierarchy and hierarchy with a proliferating distinction at the top are equal. The balanced accuracy score is better when the proliferation distinction is made at the top.
  - For non-standardized data and with the use of the logistic regression classifier, the best accuracy score is obtained with the predefined hierarchy and the best balanced accuracy score is only slightly better when the proliferation distinction is made at the top of the hierarchy.
- The overall best performance is observed with the logistic regression classifier and the hierarchy with the proliferation distinction at the top.
- The random forest classifier performs again worse than the other two classifiers across the strategies and all the obtained scores lie quite close in value.

			Predefined Hierarchy	**Hierarchy with proliferation differentiation at the top**	**Hierarchy with proliferation differentiation at the bottom**
Logistic regression	Normalised data	Accuracy	**0,782**	**0,782**	0,778
		Balanced accuracy	0,561	**0,573**	0,566
	Normalised and standardized data	Accuracy	0,735	0,727	**0,744**
		Balanced accuracy	**0,551**	0,533	0,477
Linear SVM	Normalised data	Accuracy	**0,781**	0,778	0,777
		Balanced accuracy	0,562	**0,563**	0,558
	Normalised and standardized data	Accuracy	0,745	0,745	**0,749**
		Balanced accuracy	**0,505**	0,483	0,499
Random forests	Normalised data	Accuracy	0,678	**0,680**	0,679
		Balanced accuracy	0,323	0,324	**0,326**
	Normalised and standardized data	Accuracy	**0,681**	0,676	0,677
		Balanced accuracy	**0,331**	0,320	0,330

Table 4.5: The results of hierarchical, automatic cell type annotation for the COVID19 dataset with three different hierarchies: the predefined one (Figure 3.3), a hierarchy where a distinction between proliferating and not-proliferating cells is present at the top (Figure A.1) and a hierarchy where the proliferating label is present as an extra label at the bottom (Figure A.2). For each of these three hierarchies classification is performed with three different classifiers and with and without standardizing the data. The performance of the different classifications is evaluated by calculating the average accuracy and average balanced accuracy score over the 5 folds. The bold values indicate the best scoring hierarchy for the respective metric for all the different analyses, so the best value per row in the table.

Confusion matrices are also constructed for the hierarchical classification with the two new hierarchies and the logistic regression classifier on non-standardized data. They can be found in Appendix A. Based on these visualisations, the following conclusions can be made:

- Figure A.6 shows the confusion matrices of the first level for both the newly-constructed hierarchies, and the following observations can be made based on this figure:
  - For the hierarchy with the proliferation distinction at the top, the first level is a newly-added level and it seems like a pretty good classification is possible there with an accuracy score of 98%.
  - For the hierarchy with the proliferation distinction at the bottom, granulocyte cells seem to be confused a bit more often as lymphoid cells in comparison to the predefined hierarchy. Though, relatively speaking, this encompasses still only a small number of samples.
- Figure A.7 then visualises the confusion matrix of the second level in the hierarchy with the proliferation distinction at the top. The labels here correspond to the first level labels of the other hierarchies. Here, the following observations can be made:
  - Not one sample gets assigned to the unidentified proliferating cell label.
  - The performance of the not-proliferating part of the hierarchy is very similar to that of the first level of the confusion matrix generated for hierarchical classification with the predefined hierarchy.
- Figure A.8 depicts the confusion matrices for the second level in the hierarchy with the proliferation distinction at the bottom and for the third level in the hierarchy with the proliferation distinction at the top. Here, the monocyte and macrophage distinction is somewhat more difficult for both hierarchies, which was also the case with the predefined hierarchy and the flat classification. With the hierarchy where proliferation differentiation is present at the bottom, not one sample gets assigned to the unidentified proliferating class.
- Figure A.9 and A.10 depict the confusion matrices of the levels further down in the hierarchies. Interesting to note here is that classification of the specific proliferation cell labels at the bottom of the hierarchy seems to go quite well, with only a few misclassified samples. Furthermore, the same confusions between the different cell types, observed and discussed previously with the predefined hierarchy and flat classification, seem to persist here as well with both the newly-constructed hierarchies.

It should be noted that the interpretation of these confusion matrices is not easy, since there are a lot of values to compare and since the mistakes can be propagated down to lower levels. Plus the COVID19 hierarchy is also imbalanced, meaning that not every cell type has en equal number of levels within the hierarchy. So, the confusion matrices can help detect for instance difficulties within the classification process (as was done in the previous section), but the comparison of performance across the different strategies is not always that easy. If there are large differences between the confusion matrices, these can easily be detected. But for small differences, this is more difficult. Since there are a lot of small classes present in the dataset, small differences in the number of misclassified samples do matter. Thus, these confusion matrices do not always help with finding an explanation for the differences in performance across the different strategies.

Based on the conclusions mentioned above, it can be stated that the best classification method is still obtained with the logistic regression classifier on non-standardized data. For this classifier, the hierarchy with the proliferation distinction at the top performed better in terms of the balanced accuracy score by comparison with the predefined hierarchy, while the accuracy score is equal. This would indicate that the smaller classes are better classified when the proliferation distinction is made at the top, but no definite evidence is found for this within the confusion matrices. It is also not possible to conclude that the hierarchy with the proliferation distinction at the top outperforms the predefined hierarchy, since the analysis with the linear SVM on non-standardized data showed a decrease in accuracy score and only a similar balanced accuracy score. A possible explanation here could be that proliferating specific gene expression is equally detectable for both hierarchies and thus equally detectable in the first couple of levels in the hierarchy.

Interesting is the fact that the scores obtained with the proliferating labels at the bottom are mostly lower on non-standardized data, but that the confusion matrices for the specific proliferating labels at the bottom of the hierarchy don’t show a lot of misclassified samples, which indicates that the reasons as to why the scores are lower are probably situated higher up the hierarchy. So, making a proliferating distinction somewhere at the top of the hierarchy does positively influence the results because then apparently, less mistakes get made in between the higher-situated labels. A possible explanation for this is that the presence of the common set of proliferation gene expression signals interferes with the cell type classification.

The results on the standardized data are also quite peculiar. As stated previously, standardization most likely interferes with important gene expression signals. And this interference is probably the cause of these peculiar results. Standardization will most likely meddle with multiple gene expression signals, and these results show

that the gene expression proliferation signals are most likely among them. Since hierarchical classification with the hierarchy with proliferation distinction at the top always renders the worst results across all the analyses.

Lastly, also interesting to discuss are the results obtained for the unidentified proliferating category. Both of the newly-constructed hierarchies don't assign one sample to this category. The unidentified proliferating label is assigned when the proliferating cell type is unclear. So it seems normal that these instances would get assigned to other classes by the different classifiers. With the hierarchy where the proliferation labels are added at the bottom, it is unclear where the unidentified proliferating samples end up. For the hierarchy with the proliferation distinction at the top, the unidentified proliferating samples are classified as either myeloid or lymphoid cells. But these are the only other two categories present under the proliferating label, so this might not give a good insight into what the actual cell types of these proliferating cells are. Figure A.4 visualises the confusion matrices of the second level of classification with flat classification and hierarchical classification with the predefined hierarchy. It can be seen that both for flat classification and hierarchical classification with the predefined hierarchy these cells get confused with myeloid cells, lymphoid cells and granulocytes cells. No confusion is present with the epithelial cells, indicating that most likely no proliferating epithelial cells were present in the COVID19 dataset (this label is also not present in the COVID19 dataset).

4.3.3 The influence of feature selection

Lastly, a set of analyses is performed on the COVID19 dataset to evaluate if feature selection prior to classification improves the cell type annotation. Omics data in general is often high dimensional, meaning that more features are present than observations. High dimensional data can lead to overfitting of certain machine learning models and thus suboptimal performances of these models (Thas, 2020; Hastie et al., 2007). Single-cell transcriptomics data specifically, is also quite sparse, as many features will have zero counts over the majority of the samples, which makes the classification of this data not easier. Therefore, feature selection is commonly performed as a preprocessing step when machine learning methods get implemented on this type of data.

Multiple feature selection methods exist, some more often used than others. Since the goal of these analyses was just to see whether feature selection improved the classification results in both a flat and hierarchical classification setting, a rather simple approach was implemented. Prior to classification, a random forest classifier was fitted on the data and feature importance scores, assigned by the random forest clas-

sifier, were extracted. The features inputted in the actual classifier, responsible for cell type annotation, were then selected based on a threshold for the feature importance scores. Three thresholds were tested in a parameter tuning step: the mean of all the feature importance scores, 0,3 times the mean of all the feature importances and 0,7 times the mean of all the feature importance scores. Features with a feature importance score larger or equal to this threshold were retained for the classification process.

This feature selection preprocessing method is implemented on flat classification and hierarchical classification with the predefined hierarchy by making use of three different classifiers (logistic regression, linear SVM and random forests), while assessing the influence of data standardization on the complete classification process (feature selection + classification). The performance of these analyses is again assessed with the use of the average accuracy score and average balanced accuracy score across the performed folds. The results of these analyses can be found in Table 4.6. The bold values indicate the best performing strategy out of the two (flat classification and hierarchical classification with the predefined hierarchy) and the underlined values indicate an improvement in metric score in comparison to the analyses that were performed without feature selection.

Based on Table 4.6, the following conclusions can be formulated:

- Feature selection does not improve the accuracy scores of the best performing strategy: hierarchical classification with the logistic regression classifier or linear SVM classifier and the predefined hierarchy on non-standardized data, the accuracy scores even decreased a little bit. The balanced accuracy scores for these analyses did increase with feature selection and these strategies remained the best performing ones.
- The performance of the random forest classifier is a lot better with feature selection.
- Feature selection improves almost all the results on the standardized data and even results in flat classification outperforming hierarchical classification with the predefined hierarchy, in terms of the accuracy scores for both the logistic regression classifier and linear SVM classifier. The balanced accuracy scores however remain higher with hierarchical classification with the predefined hierarchy in this setting.

			Feature selection and flat classification	**Feature selection and hierarchical classification with the predefined hierarchy**
Logistic regression	Normalised data	Accuracy	0,762	**0,781**
		Balanced accuracy	0,547	**0,574**
	Normalised and standardized data	Accuracy	**0,763**	0,758
		Balanced accuracy	0,498	**0,523**
Linear SVM	Normalised data	Accuracy	0,765	**0,778**
		Balanced accuracy	0,543	**0,579**
	Normalised and standardized data	Accuracy	**0,756**	0,754
		Balanced accuracy	0,466	**0,547**
Random forests	Normalised data	Accuracy	**0,721**	**0,721**
		Balanced accuracy	0,366	**0,416**
	Normalised and standardized data	Accuracy	0,717	**0,724**
		Balanced accuracy	0,354	**0,424**

Table 4.6: The results of flat and hierarchical classification with the predefined hierarchy on the COVID19 dataset with feature selection as a preprocessing step. For both classification settings three classifiers are implemented and the influence of standardizing the data is assessed. The performance of the different classifications is evaluated by calculation of the average accuracy and average balanced accuracy over the 5 folds. The bold values indicate the best classification setting for each analysis (and for the respective metric). The underlined values indicate an improvement in the respective score due to the additional feature selection step.

So, based on these results, it can be concluded that feature selection can lead to classification improvements. But, given the fact that almost all specifically-designed automatic cell type annotation tools make use of a feature selection step, better results were expected, especially since the accuracy score of the best performing strategy on this dataset decreases. However, for this specific dataset 24 444 features are present, which is quite a lot. And so the retention of features with the thresholds mentioned above might not actually be strict enough. It can be argued that maybe even only 1000 genes or 1% of these features are of importance for the correct classification of the samples. And thus, stricter thresholds could lead to better results. Furthermore, the feature selection method applied here is not commonly used when it comes to automatic single-cell annotation. Most strategies base themself on selecting the most variable genes with principal component based analyses (Zhang et al., 2019; Bernstein et al., 2021; Alquicira-Hernández et al., 2018; Kim et al., 2019; Hao et al., 2020) or highly-variable gene detection methods (Huang and Zhang, 2021; Hu et al., 2020; Kiselev et al., 2018). Moreover, the parameter tuning performed with these analyses was rather limited due to time restrictions (see Appendix B.2), which can also lead to a decrease in performance.

As the results in Table 4.6 indicate, feature selection seems to lead to serious improvements in the classification process of standardized data. This could be explained by reintroduction of some features' importance with the use of feature selection, which were previously lost due to standardization. And so again, these results validate the standardizing hypothesis made in the previous sections that states that standardization could lead to a loss of important signals in the gene expression data by decreasing the largest signals and thus decreasing feature importances for certain observations.

4.3.4 General conclusions for the COVID19 dataset

The previous three sections each detail and discuss the different analyses quite in depth. To retain some overview of the results for all the analyses performed on the COVID19 dataset, the most important conclusions will be briefly recapitulated here:

- The best cell type annotation was obtained with the use of hierarchical classification with a hierarchy where proliferation distinction is present at the top and with the logistic regression classifier on non-standardized data.
- The logistic regression classifier and linear SVM classifier both perform really well on this dataset, the random forest classifier might not be that suited for annotation of this dataset.
- Standardization worsens the performance of classification on this dataset with the linear SVM and logistic regression classifier. But the best performance with the random forest classifier is obtained on standardized data with hierarchical classification that makes use of the predefined hierarchy.
- The best performing classification strategy for this dataset is hierarchical classification. Implementation with the predefined hierarchy and hierarchy with proliferation distinction at the top generates similar results. This leads to the conclusion that the proliferation signals present in the data are probably detectable at the first couple of levels in the hierarchy. Implementation of hierarchical classification with a hierarchy with proliferation distinction at the bottom results in a decrease in performance. This is most likely caused by mistakes that are made in the upper part of the hierarchy since the confusion matrices show that for the proliferation specific labels not a lot of samples are misclassified.
- Feature selection seems to improve most of the classification results, except those of the best performing setups. Most likely bigger improvements can be made with specific feature selection methods tailored to single-cell data.
- Certain cell types are quite hard to distinguish from each other as discussed in Section 4.3.1 due to the relatedness of immune cell types. This makes the COVID19 dataset a more complex dataset to classify and shows that the complexity of a biological dataset is not solely dependent on the amount of cell populations present (which is quite limited here with only 46 cell populations).

CHAPTER 5

CONCLUSION AND FUTURE WORK

The goal of this dissertation was to answer the following question: 'Does incorporation of cell type hierarchy information improve the annotation process of single-cell transcriptomics data?'. And the answer to this question is yes, hierarchical classification can lead to a better annotation. In this dissertation, hierarchical classification was performed on two datasets that both contained hierarchical labelling: the AMB dataset and the COVID19 dataset and in both cases hierarchical classification improved the classification results. Yet, the improvements were not that large for the AMB dataset. This can be explained by the lack of hierarchical complexity of this dataset with only three main labels at the top, which seem to have very different biological functions. Plus, the classification complexity of this dataset seems to also be smaller by comparison to the COVID19 dataset, where apparently a very close relatedness is present between (some of) the labels.

The performed analyses show that both the linear SVM classifier and the logistic regression classifier are great options for automatic single-cell type annotation, which was also concluded by Huang and Zhang (2021), where the performance of several machine learning approaches was compared for cell type annotation of single-cell data. Furthermore, standardization of single-cell transcriptomics data for cell type annotation on single-cell expression data seems to worsen the cell type annotation.

For the COVID19 dataset, the cell proliferating label present in this dataset gave rise to a couple of extra analyses. This label is not a cell type label, like the other labels present in the dataset, but a cell state label and is thus part of a parallel labelling process. In this dissertation the overall classification of this dataset was considered as a multi-class classification problem where no distinction was made between the difference in nature of cell type labels and cell state labels. The results show that if the cell state label is present at the top levels in the hierarchy, distinction between the cell state label and other labels is possible. However, due to the parallel labelling present, this annotation process can also be considered as a multi-label classifica-

tion problem where each sample is assigned two labels: a cell type and a cell state. This could possibly increase the cell type classification results, as with the multi-class strategy the presence of some of the proliferating labels in the dataset is quite small, which would not be the case in a multi-label classification setting. It would thus be interesting to implement a multi-label classification strategy for this dataset. Especially since recently-developed sequencing techniques are making it possible to generate more information than just gene expression information on single cells, like for instance CITE-Seq where it is possible to measure proteomics information besides transcriptomics information (Stoeckius et al., 2017). Furthermore, for single cell transcriptomics data, spatial information is often also generated, since this can give a lot of insight into biological processes. This extra information can thus generate extra (parallel) label classes which could then be estimated with multi-label approaches, thus rendering more possibilities for the multi-label setting.

The impact of feature selection on the classification process was also assessed on the COVID19 dataset and multiple improvements can still be made in this area. Section 4.3.3 gives more information about this and already suggests several alternatives that could improve the classification results.

As mentioned in Chapter 2, hierarchical classification has the benefit of never violating the class hierarchy. But another possible advantage to hierarchical classification, which was not implemented here, is that it could allow for the labelling of certain samples to be stopped if uncertainty is present during class assignment at a certain level. This has the potential to improve the classification results, but moreover it could lead to a more curated cell annotation. Possibilities then also exist to make a distinction between epistemic and aleatoric uncertainty and to correct for these two types separately. Epistemic uncertainty refers to uncertainty caused by a lack of knowledge, aleatoric or statistical uncertainty is present due to a notion of randomness (Hüllermeier and Waegeman, 2021).

Lastly, it should be noted that in this dissertation only intra-dataset analyses were performed. However, a good inter-dataset performance of a classifier is a very desirable quality and it could thus be very interesting to compare the performance of hierarchical and flat classification in this setting.

BIBLIOGRAPHY

10x Genomics (2017). Single-cell rna-seq: An introductory overview and tools for getting started. https://www.10xgenomics.com/blog/single-cell-rna-seq-an-introductory-overview-and-tools-for-getting-started.

Abdelaal, T., Michielsen, L., Cats, D., Hoogduin, D., Mei, H., Reinders, M. J., and Mahfouz, A. (2019). A comparison of automatic cell identification methods for single-cell rna sequencing data. *Genome Biology*, 20:194.

Aldridge, S. and Teichmann, S. A. (2020). Single cell transcriptomics comes of age. *Nature Communications*, 11.

Alquicira-Hernández, J., Sathe, A., Ji, H. P., Nguyen, Q., and Powell, J. E. (2018). scpred: Cell type prediction at single-cell resolution. *bioRxiv*.

Aran, D., Looney, A. P., Liu, L., Wu, E., Fong, V., Hsu, A., Chak, S., Naikawadi, R. P., Wolters, P. J., Abate, A. R., Butte, A. J., and Bhattacharya, M. (2019). Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. *Nature Immunology*, 20:163–172.

Asp, M., Bergenstrahle, J., and Lundeberg, J. (2020). Spatially resolved transcriptomesnext generation tools for tissue exploration. *BioEssays*, 42.

Baron, M., Veres, A., Wolock, S., Faust, A., Gaujoux, R., Vetere, A., Ryu, J., Wagner, B., Shen-Orr, S., Klein, A., Melton, D., and Yanai, I. (2016). A single-cell transcriptomic map of the human and mouse pancreas reveals inter- and intra-cell population structure. *Cell Systems*, 3(4):346–360.e4.

Bernstein, M. N., Ma, Z., Gleicher, M., and Dewey, C. N. (2021). Cello: comprehensive and hierarchical cell type classification of human cells with the cell ontology. *iScience*, 24(1):101913.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer Science+Bussiness Media.

Castro, F., Cardoso, A. P., Gonçalves, R. M., Serre, K., and Oliveira, M. J. (2018). Interferon-gamma at the crossroads of tumor immune surveillance or evasion. *Frontiers in Immunology*, 9:847.

Chen, G., Ning, B., and Shi, T. (2019). Single-cell rna-seq technologies and related computational data analysis. *Frontiers in Genetics*, 10:317.

Conlon, I. and Raff, M. (1999). Size control in animal development. *Cell*, 96:235–244.

de Kanter, J. K., Lijnzaad, P., Candelli, T., Margaritis, T., and Holstege, F. C. P. (2019). CHETAH: a selective, hierarchical cell type identification method for single-cell RNA sequencing. *Nucleic Acids Research*, 47(16):e95–e95.

Diaz-Mejia, J. J., Meng, E. C., Pico, A. R., MacParland, S. A., Ketela, T., Pugh, T. J., Bader, G. D., and Morris, J. H. (2019). Evaluation of methods to assign cell type labels to cell clusters from single-cell rna-sequencing data [version 3; peer review: 2 approved, 1 approved with reservations]. *F1000Research*, 8:296.

Duronia, R. J. and Xiong, Y. (2013). Signaling pathways that control cell proliferation. *Cold Spring Harbor Perspectives in Biology*, 5.

Eberwine, J., Yeh, H., Miyashiro, K., Cao, Y., Nair, S., Finnell, R., Zettel, M., and Coleman, P. (1992). Analysis of gene expression in single live neurons. *Proceedings of the National Academy of Sciences*, 89(7):3010–3014.

Forsyth, D. (2019). *Applied Machine Learning*. Springer Nature.

Grandini, M., Bagli, E., and Visani, G. (2020). Metrics for multi-class classification: an overview.

Hand, D. J. and Till, R. J. (2001). A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45:171–186.

Hao, Y., Hao, S., Andersen-Nissen, E., Mauck, W. M., Zheng, S., Butler, A., Lee, M. J., Wilk, A. J., Darby, C., Zagar, M., Hoffman, P., Stoeckius, M., Papalexi, E., Mimitou, E. P., Jain, J., Srivastava, A., Stuart, T., Fleming, L. B., Yeung, B., Rogers, A. J., McElrath, J. M., Blish, C. A., Gottardo, R., Smibert, P., and Satija, R. (2020). Integrated analysis of multimodal single-cell data. *bioRxiv*.

Haque, A., Engel, J., Teichmann, S. A., and Lönnberg, T. (2017). A practical guide to single-cell rna-sequencing for biomedical research and clinical applications. *Genome Medicine*, 9.

Hashimshony, T., Wagner, F., Sher, N., and Yanai, I. (2012). Cel-seq: Single-cell rna-seq by multiplexed linear amplification. *Cell Reports*, 2(3):666 – 673.

Hastie, T., Tibshirani, R., and Friedman, J. (2007). *The elements of statistical learning: Data Mining, Inference and Prediction*. Springer-Verlag. Second edition.

Hinks, T. S. C. and Zhang, X.-W. (2020). Mait cell activation and functions. *Frontiers in Immunology*, 11:1014.

Hu, J., Li, X., Hu, G., Lyu, Y., Susztak, K., and Li, M. (2020). Iterative transfer learning with neural network for clustering and cell type classification in single-cell rna-seq analysis. *Nature Machine Intelligence*, 2:607–618.

Huang, Y. and Zhang, P. (2021). Evaluation of machine learning approaches for cell-type identification from single-cell transcriptomics data. *Briefings in Bioinformatics*. bbab035.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine learning*, 110:457–506.

Islam, S., Zeisel, A., Joost, S., Kasper, G. L. M. P. Z. M., Lönnerberg, P., and Linnarsson, S. (2014). Quantitative single-cell rna-seq with unique molecular identifiers. *Nature Methods*, 11:163 – 166.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer Science+Bussiness Media.

Kannan, S., Miyamoto, M., Leei Lin, B., Zhu, R., Murphy, S., Kass, D. A., Andersen, P., and Kwon, C. (2019). Large particle fluorescence-activated cell sorting enables high-quality single-cell rna sequencing and functional analysis of adult cardiomyocytes. *Circulation Research*, 125(5):567–569.

Kanter, I. and Kalisky, T. (2015). Single cell transcriptomics: methods and applications. *Frontiers in Oncology*, 5:53.

Ketkar, N. (2017). *Stochastic Gradient Descent*, pages 113–132. Apress, Berkeley, CA.

Kharchenko, P. V., Silberstein, L., and Scadden, D. T. (2014). Bayesian approach to single-cell differential expression analysis. *Nature Methods*, 11:740–742.

Kim, T., Lo, K., Geddes, T. A., Kim, H. J., Yang, J. Y. H., and Yang, P. (2019). screclassify: post hoc cell type classification of single-cell rna-seq data. *BMC Genomics*, 20(913).

Kiselev, V. Y., Yiu, A., and Hemberg, M. (2018). scmap: projection of single-cell rna-seq data across data sets. *Nature Methods*, 15:359–362.

Korsunsky, I., Millard, N., Fan, J., Slowikowski, K., Zhang, F., Wei, K., Baglaenko, Y., Brenner, M., Po-ru, L., and Raychaudhuri, S. (2019). Fast, sensitive and accurate integration of single-cell data with harmony. *Nature Methods*, 16:1289–1296.

Kulkarni, A., Anderson, A. G., Merullo, D. P., and Konopka, G. (2019). Beyond bulk: a review of single cell transcriptomics methodologies and applications. *Current Opinion in Biotechnology*, 58:129 – 136. Systems Biology Nanobiotechnology.

Kumar, A. (2020). Python - nested cross validation for algorithm selection. https://vitalflux.com/python-nested-cross-validation-algorithm-selection/.

Le Page, C., Génin, P., Baines, M., and Hiscott, J. (2000). Interferon activation and innate immunity. *Reviews in immunogenetics*, 2(3):374386.

Lieberman, Y., Rokach, L., and Shay, T. (2018). Correction: Castle - classification of single cells by transfer learning: Harnessing the power of publicly available single cell rna sequencing experiments to annotate new experiments. *Nature Methods*, 13(11).

Liegeois, M., Legrand, C., Desmet, C. J., Marichal, T., and Bureau, F. (2018). The interstitial macrophage: A long-neglected piece in the puzzle of lung immunity. *Cellular Immunology*, 330:91–96. Special Issue: A Tissue Macrophage Compendium.

Lowe, R., Shirley, N., Bleackley, M., Dolan, S., and Shafee, T. (2017). Transcriptomics technologies. *PLOS Computational Biology*, 13:1–23.

Ma, F. and Pellegrini, M. (2019). Actinn: automated identification of cell types in single cell rna sequencing. *Bioinformatics*, 36(2):533–538.

Manning, C. D., Raghavan, P., and Schütze, H. (2009). *An Introduction to Information Retrieval*. Cambridge University Press. Online edition.

Mehryar Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of Machine Learning (second edition)*. The MIT Press.

Miceli, M. C. and Parnes, J. R. (1993). Role of cd4 and cd8 in t cell activation and differentiation. volume 53 of *Advances in Immunology*, pages 59–122. Academic Press.

Mosley, L. (2013). A balanced approach to the multi-class imbalance problem. Graduate Theses and Dissertations.

Papachristoudis, G. (2019). The bias-variance tradeoff. https://towardsdatascience.com/the-bias-variance-tradeoff-8818f41e39e9.

Parihar, A., Eubank, T., and A.I., D. (2010). Monocytes and macrophages regulate immunity through dynamic networks of survival and cell death. *Journal of Innate Immunity*, 2(3):204–215.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau,

D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pliner, H. A., Shendure, J., and Trapnell, C. (2019). Supervised classification enables rapid annotation of cell atlases. *Nature Methods*, 16:983–986.

Raj, A. and van Oudenaarden, A. (2008). Nature, nurture, or chance: Stochastic gene expression and its consequences. *Cell*, 135:216–226.

Sandberg, R. (2014). Entering the era of single-cell transcriptomics in biology and medicine. *Nature Methods*, 11:22–24.

Sato, K., Tsuyuzaki, K., Shimizu, K., and Nikaido, I. (2019). Cellfishing.jl: an ultrafast and scalable cell search method for single-cell rna sequencing. *Genome Biology*, 20(31).

Segerstolpe, A., Palasantza, A., Eliasson, P., Andersson, E.-M., Andréasson, A.-C., Sun, X., Picelli, S., Sabirsh, A., Clausen, M., Bjursell, M. K., Smith, D., Kasper, M., Ämmälä, C., and Sandberg, R. (2016). Single-cell transcriptome profiling of human pancreatic islets in health and type 2 diabetes. *Cell Metabolism*, 24(4):593–607.

Shao, X., Liao, J., Lu, X., Xue, R., Ai, N., and Fan, X. (2020). sccatch: Automatic annotation on cell types of clusters from single-cell rna sequencing data. *iScience*, 23(3):100882.

Silla, C. N. and Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22:31–72.

Sokolova, M., Japkowicz, N., and Szpakowicz, S. (2006). Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation. In Sattar, A. and Kang, B.-h., editors, *AI 2006: Advances in Artificial Intelligence*, pages 1015–1021, Berlin, Heidelberg. Springer Berlin Heidelberg.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427–437.

Srivastava, A., George, J., and Karuturi, R. K. (2019). Transcriptome analysis. In Ranganathan, S., Gribskov, M., Nakai, K., and Schönbach, C., editors, *Encyclopedia of Bioinformatics and Computational Biology*, pages 792 – 805. Academic Press, Oxford.

Stoeckius, M., Hafemeister, C., Stephenson, W., Houck-Loomis, B., Chattopadhyay, P. K., Swerdlow, H., Satija, R., and Smibert, P. (2017). Simultaneous epitope and transcriptome measurement in single cells. *Nature Methods*, 14:856–868.

Svensson, V., Vento-Tormo, R., and Teichmann, S. A. (2018). Exponential scaling of single-cell rna-seq in the past decade. *Nature Protocols*, 13.

Tasic, B., Yao, Z., Graybuck, L. T., Smith, K. A., Nguyen, T. N., Bertagnolli, D., Goldy, J., Garren, E., Economo, M. N., Viswanathan, S., Penn, O., Bakken, T., Menon, V., Miller, J., Fong, O., Hirokawa, K. E., Lathia, K., Rimorin, C., Tieu, M., Larsen, R., Casper, T., Barkan, E., Kroll, M., Parry, S., Shapovalova, N. V., Hirschstein, D., Pendergraft, J., Sullivan, H. A., Kim, T. K., Szafer, A., Dee, N., Groblewski, P., Wickersham, I., Cetin, A., Harris, J. A., Levi, B. P., Sunkin, S. M., Madisen, L., Daigle, T. L., Looger, L., Bernard, A., Phillips, J., Lein, E., Hawrylycz, M., Svoboda, K., Jones, A. R., Koch, C., and Zeng, H. (2018). Shared and distinct transcriptomic cell types across neocortical areas. *Nature*, 536(4):72–78.

Thas, O. (2019-2020). High-dimensional data analysis. Course offered by the University of Ghent.

Turko, P., Groberman, K., Browa, F., Cobb, S., and Vida, I. (2019). Differential dependence of gabaergic and glutamatergic neurons on glia for the establishment of synaptic transmission. *Cerebral cortex (New York, N.Y. : 1991)*, 29(3):12301243.

Waegeman, W. (2020-2021). Predictive modelling. Course offered by the University of Ghent.

Wang, Y. and Navin, N. (2015). Advances and applications of single-cell sequencing technologies. *Molecular Cell*, 58(4):598 – 609.

Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021.

Yang, N., Ray, S., and Krafts, K. (2014). Cell proliferation. In Wexler, P., editor, *Encyclopedia of Toxicology (Third Edition)*, pages 761–765. Academic Press, Oxford, third edition edition.

Yedida, A. (2016). Against f-score. Blogpost. Accessed: 2021-03-02.

Zappia, L., Phipson, B., and Oshlack, A. (2018). Exploring the single-cell rna-seq analysis landscape with the scrna-tools database. *PLoS Computational Biology*, 14.

Zhang, A. W., OFlanagan, C., Chavez, E. A., Lim, J. L. P., Ceglia, N., McPherson, A., Wiens, M., Walters, P., Chan, T., Hewitson, B., Lai, D., Mottok, A., Sarkozy, C., Chong, L., Aoki, T., Wang, X., Weng, A. P., McAlpine, J. N., Aparicio, S., Steidl, C., Campbell, K. R., and Shah, S. P. (2019). Probabilistic cell-type assignment of single-cell rna-seq for tumor microenvironment profiling. *Nature Methods*, 16:1007–1015.

Zhang, D.and Wang, J. and Zhao, X. (2015). Estimating the uncertainty of average f1 scores. ICTIR '15, page 317320, New York, NY, USA. Association for Computing Machinery.

Zhang, X. (2020). *A Matrix Algebra Approach to Artificial Intelligence*. Springer.

Zhang, X., Lan, Y., Xu, J., Quan, F., Zhao, E., Deng, C., Luo, T., Xu, L., Liao, G., Yan, M., Ping, Y., Li, F., Shi, A., Bai, J., Zhao, T., Li, X., and Xiao, Y. (2018). Cellmarker: a manually curated resource of cell markers in human and mouse. *Nucleic Acids Research*, 47(D1):D721–D728.

APPENDIX A

ADDITIONAL FIGURES

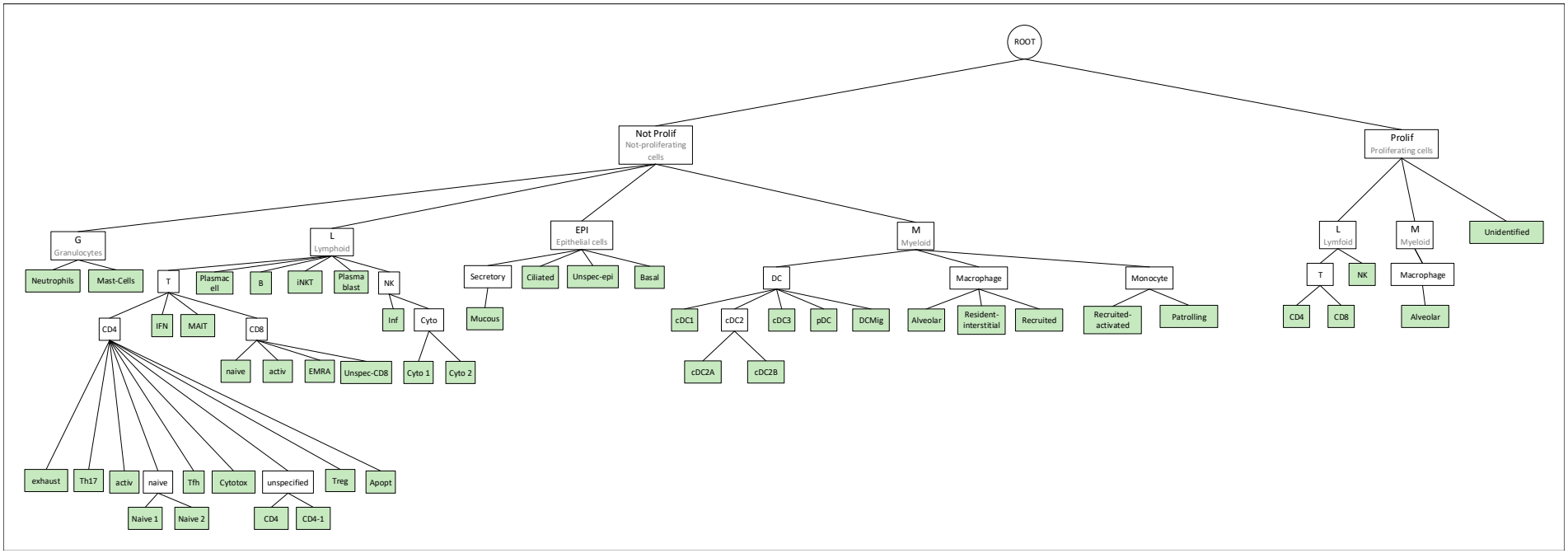

Figure A.1: The adapted hierarchy of the COVID19 dataset with a distinction made at the top of the hierarchy between proliferating and not-proliferating cells. The green-coloured nodes represent the leaves of this hierarchy, all these nodes contain the lowest level of classification possible for certain observations.

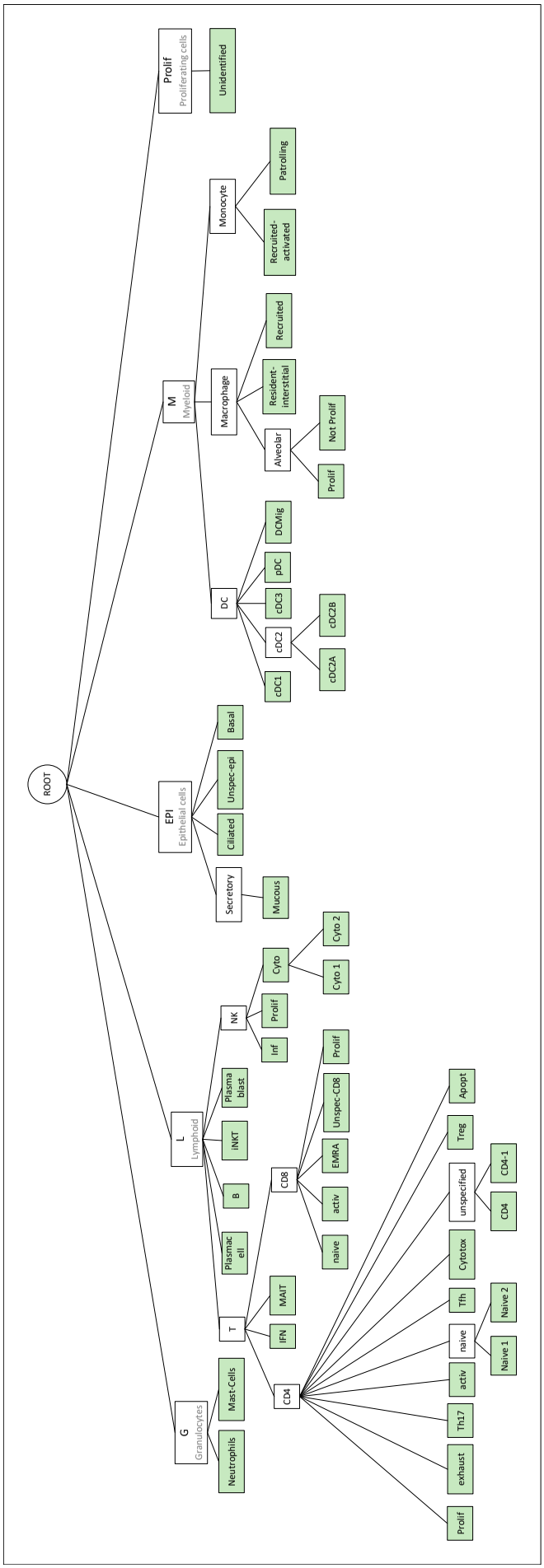

Figure A.2: The adapted hierarchy of the COVID19 dataset with the proliferating labels added to the corresponding cell type node as an extra label at the bottom. The green-coloured nodes represent the leaves of this hierarchy, all these nodes contain the lowest level of classification possible for certain observations.

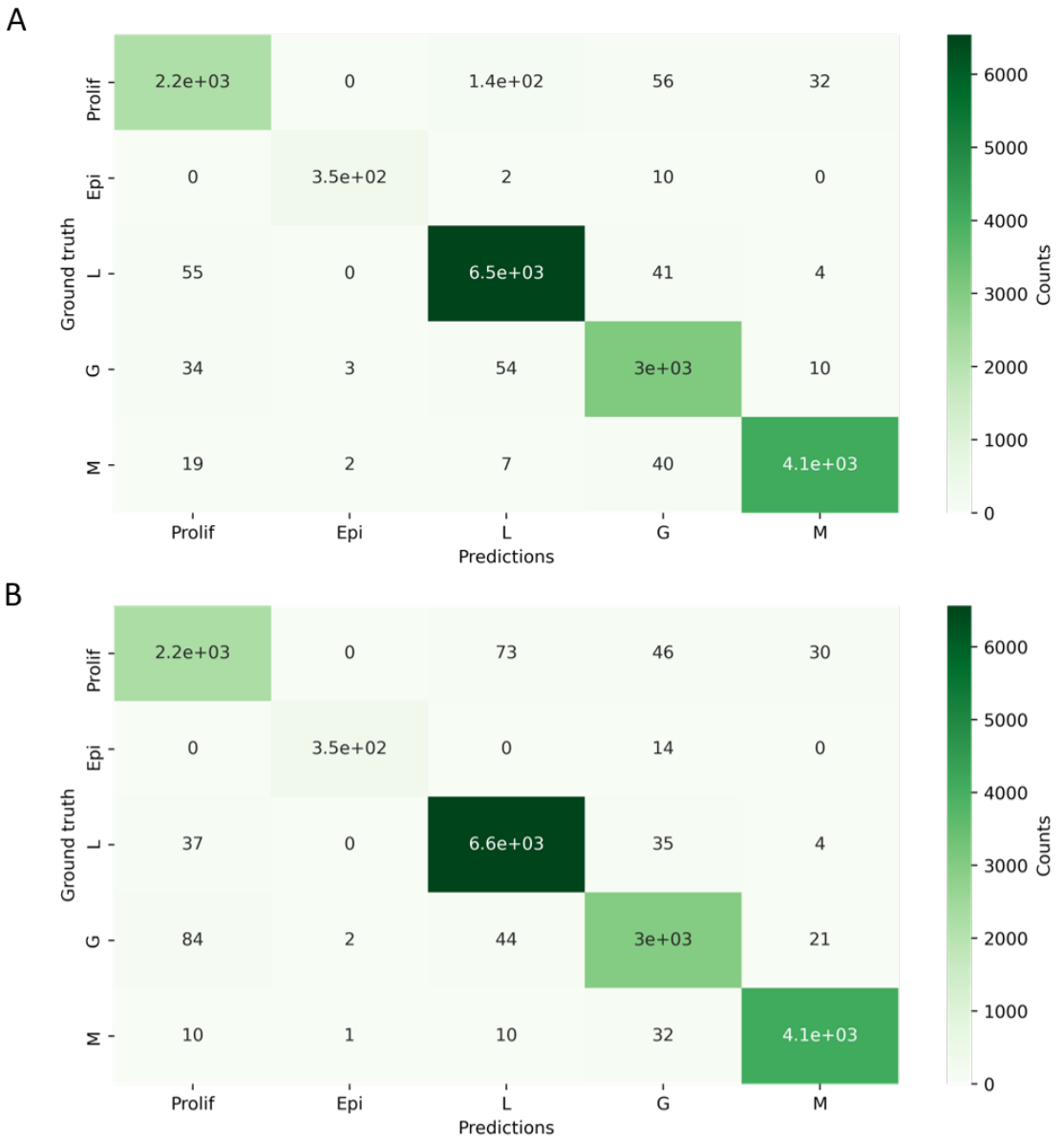

Figure A.3: Confusion matrices of the first level of classification of the COVID19 dataset, implemented with **A**: flat classification and **B**: hierarchical classification with the predefined hierarchy, on non-standardized data while making use of the logistic regression classifier. The y-axis represents the true labels and the x-axis the predictions. The values depicted in the matrices are absolute counts.

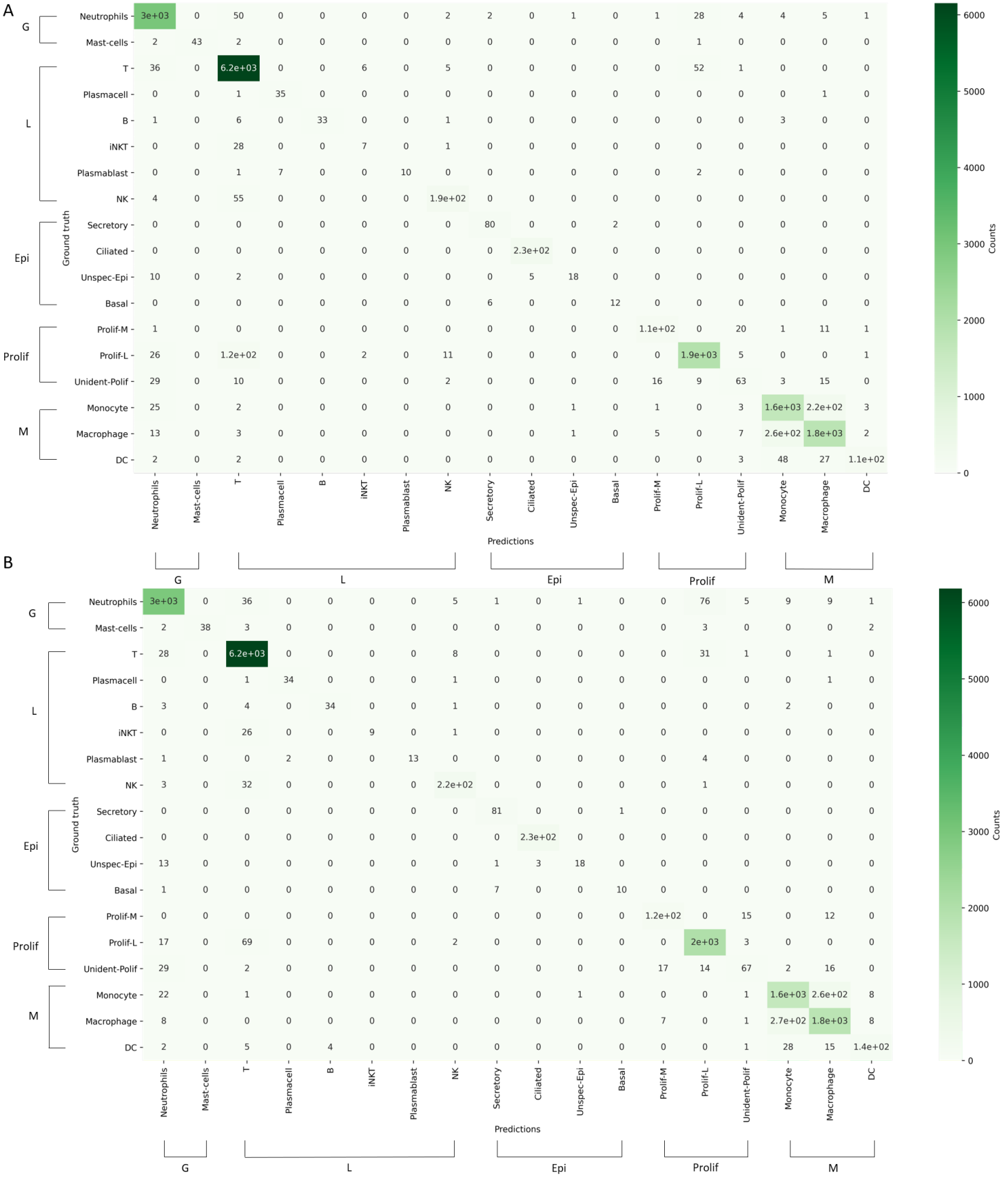

Figure A.4: Confusion matrices of the second level of classification of the COVID19 dataset, implemented with **A**: flat classification and **B**: hierarchical classification with the predefined hierarchy, on non-standardized data while making use of the logistic regression classifier. The y-axis represents the true labels and the x-axis the predictions. The values depicted in the matrices are absolute counts.

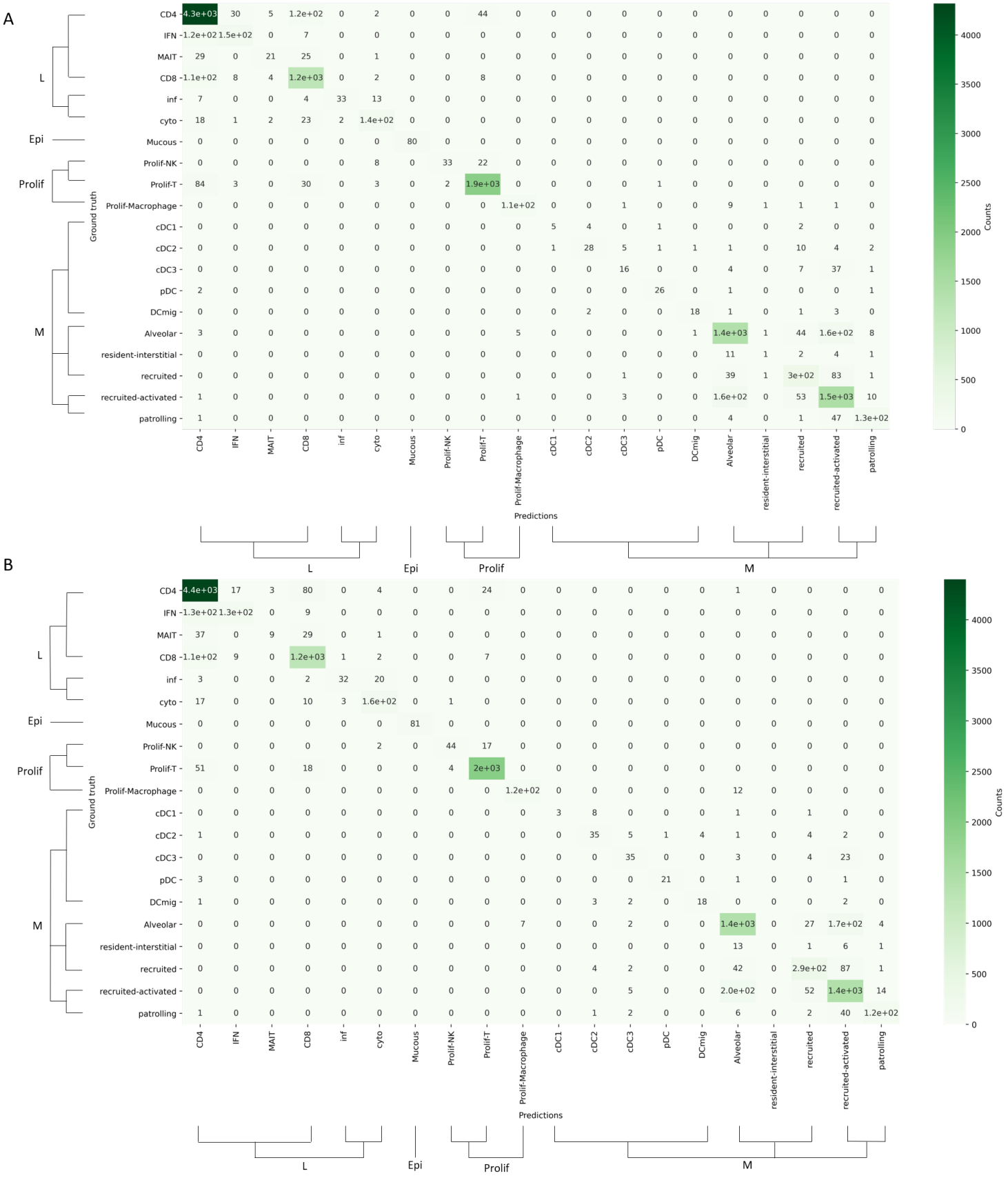

Figure A.5: Confusion matrices of the third level of classification of the COVID19 dataset, implemented with **A**: flat classification and **B**: hierarchical classification with the predefined hierarchy, on non-standardized data while making use of the logistic regression classifier. The y-axis represents the true labels and the x-axis the predictions. The values depicted in the matrices are absolute counts.

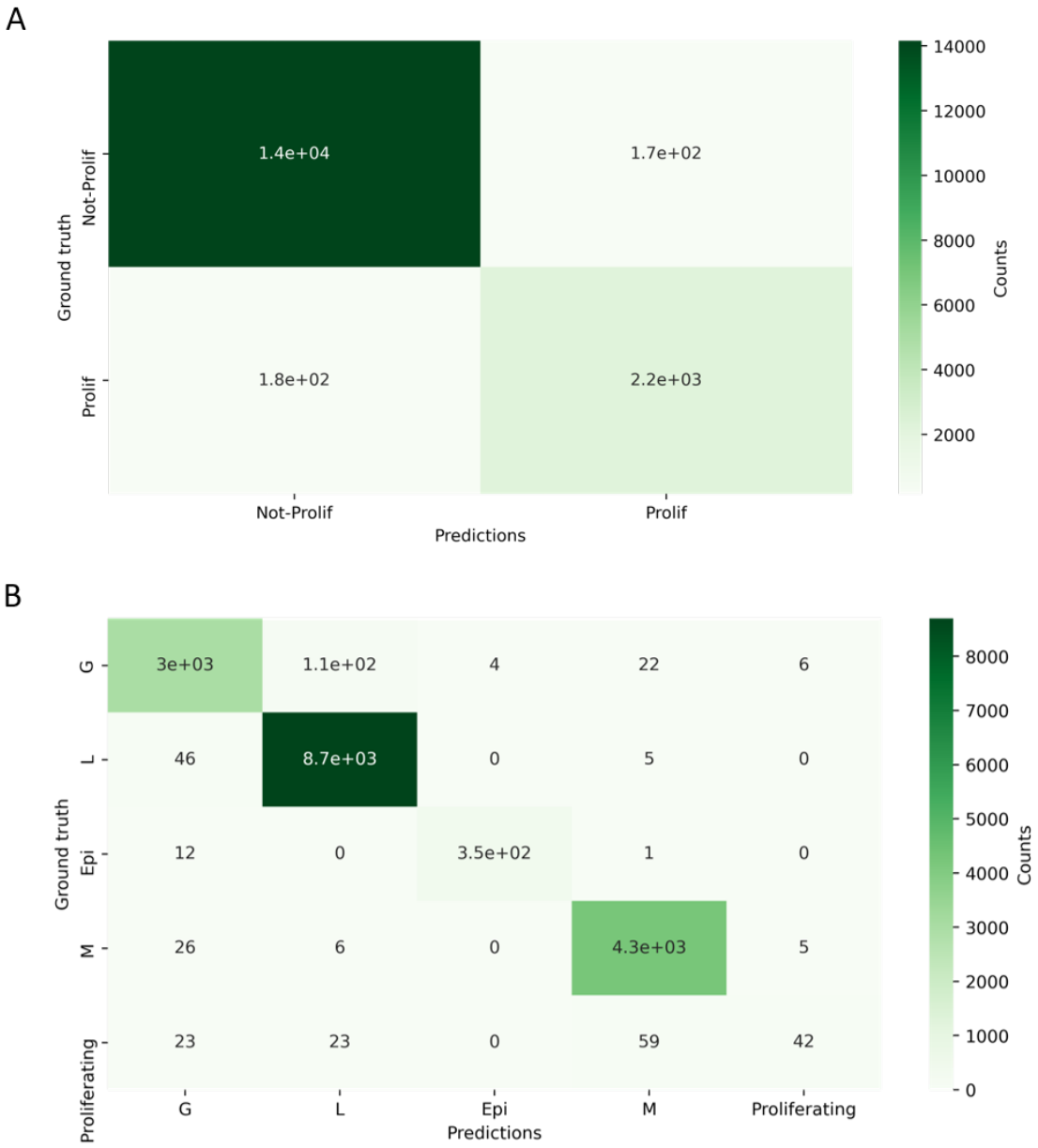

Figure A.6: Confusion matrices of the first level of hierarchical classification of the COVID19 dataset with **A**: the hierarchy with the proliferation distinction at the top and **B**: the hierarchy with the proliferation distinction at the bottom. Classification is performed with the logistic regression classifier on non-standardized data. The y-axis represents the true labels and the x-axis the predictions. The values depicted in the matrices are absolute counts.

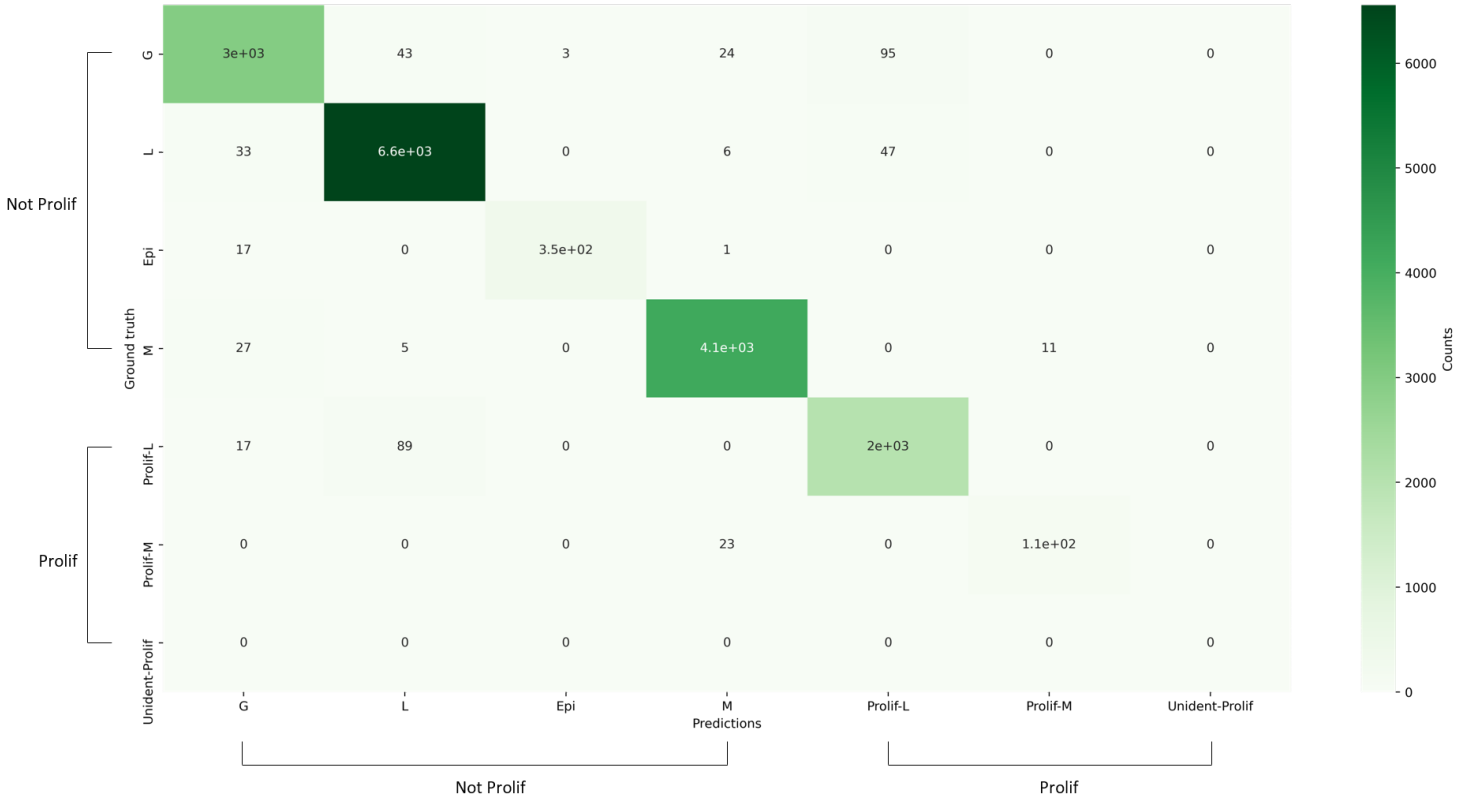

Figure A.7: The confusion matrix of the second level of hierarchical classification of the COVID19 dataset with the hierarchy with the proliferation distinction at the top. Classification is performed with the logistic regression classifier on non-standardized data. The y-axis represents the true labels and the x-axis the predictions. The values depicted in the matrix are absolute counts.

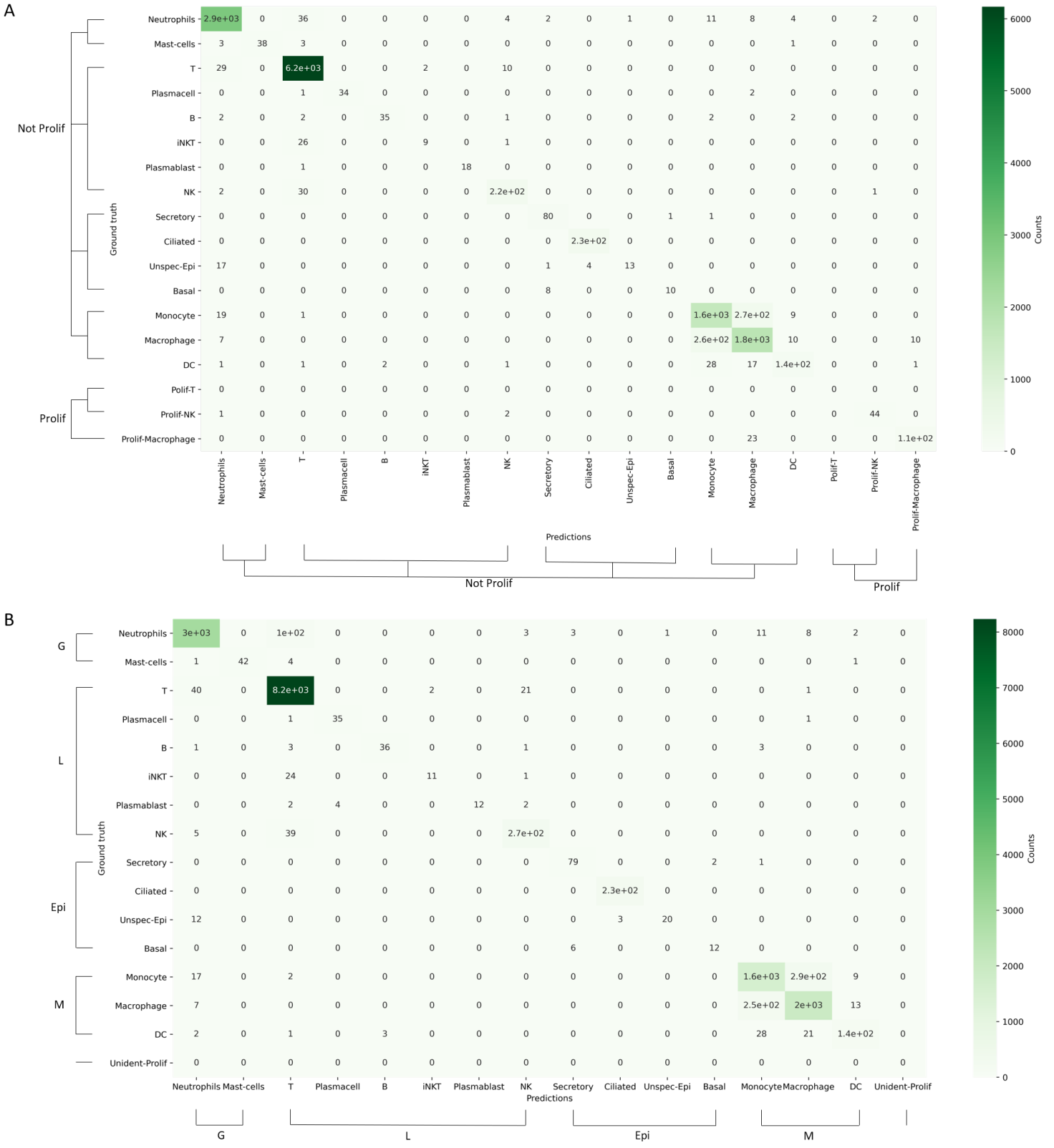

Figure A.8: **A**: the confusion matrix of the third level of hierarchical classification with the hierarchy with the proliferation distinction at the top. **B**: the confusion matrix of the second level of hierarchical classification with the hierarchy with the proliferation distinction at the bottom. Both classifications are implemented with the logistic regression classifier on non-standardized data. The y-axis represents the true labels and the x-axis the predictions. The values depicted in the matrices are absolute counts.

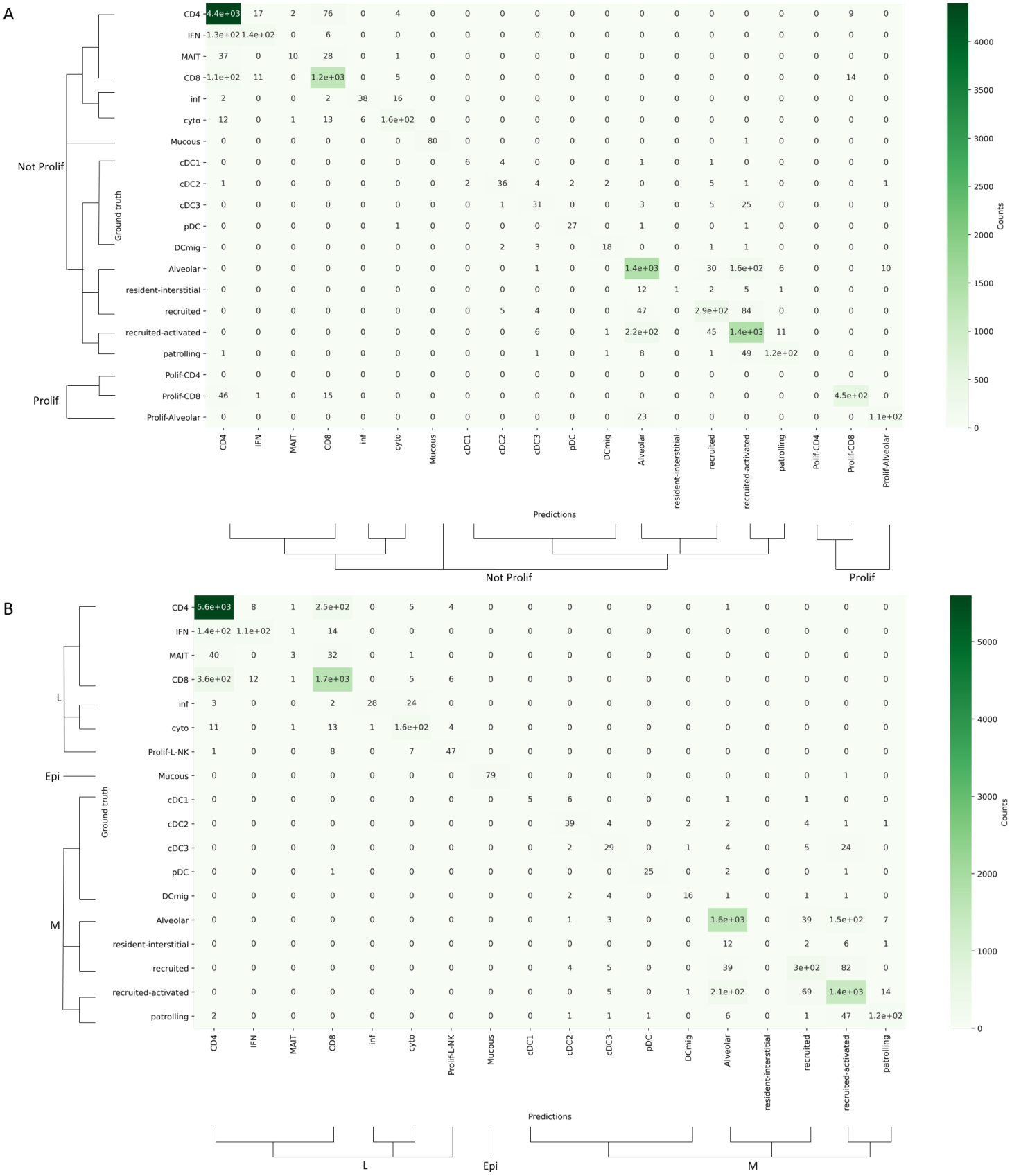

Figure A.9: **A**: the confusion matrix of the fourth level of hierarchical classification with the hierarchy with the proliferation distinction at the top. **B**: the confusion matrix of the third level of hierarchical classification with the hierarchy with the proliferation distinction at the bottom. Both classifications are performed with the logistic regression classifier on non-standardized data. The y-axis represents the true labels and the x-axis the predictions. The values depicted in the matrices are absolute counts.

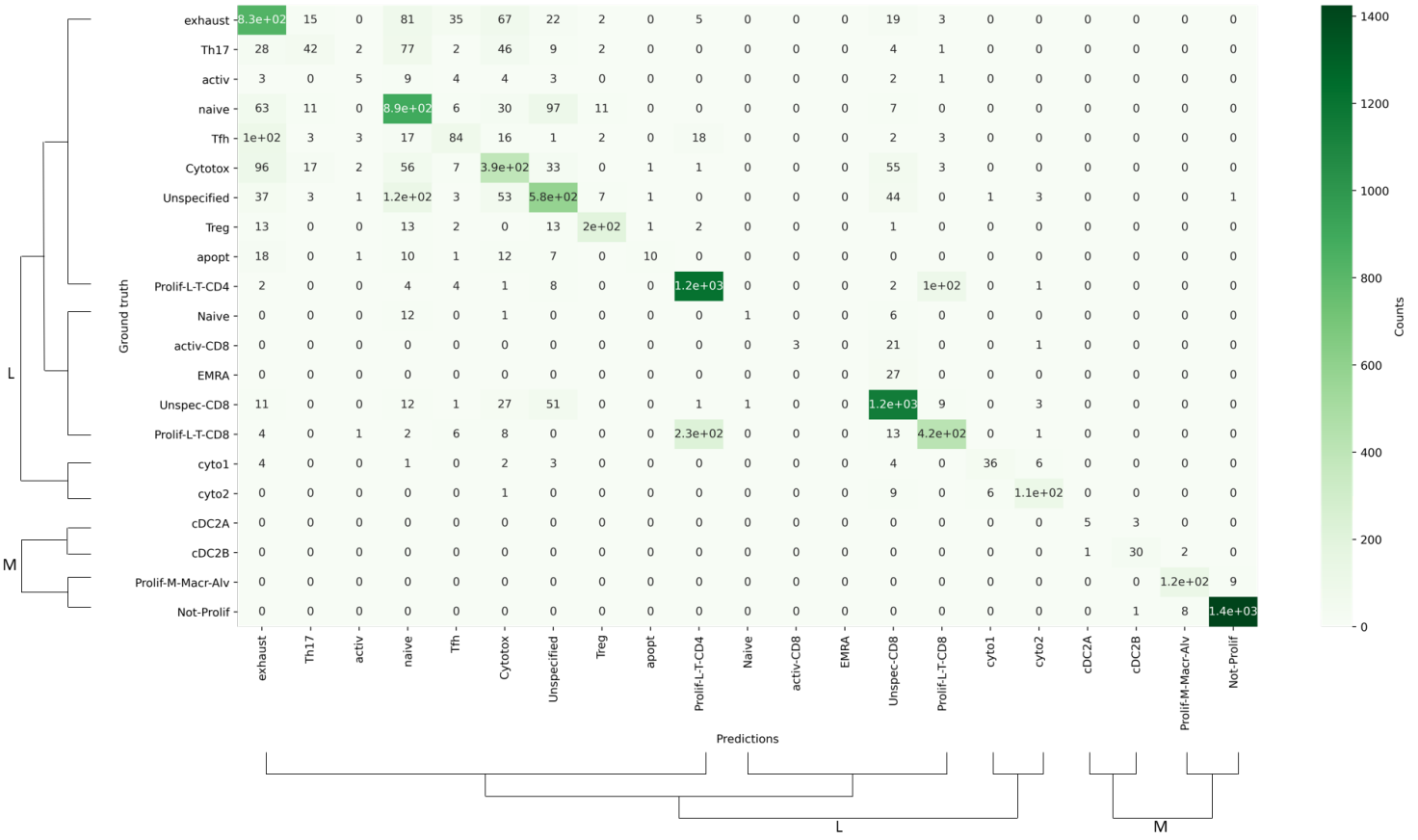

Figure A.10: The confusion matrix of the fourth level of hierarchical classification of the COVID19 dataset with the hierarchy with the proliferation distinction at the bottom. Classification is performed with the logistic regression classifier on non-standardized data. The y-axis represents the true labels and the x-axis the predictions. The values depicted in the matrix are absolute counts.

APPENDIX B

SOFTWARE SPECIFICATIONS

The following software was used in this dissertation:

- **Python** 3.8.8, **R** 4.1.0
- **NumPy** 1.19.2, **Pandas** 1.2.3, **Matplotlib** 3.3.2, **Seaborn** 0.11.1, **Seurat** 4.0.1
- **scikit-learn** 0.24.1, **Pytorch** 1.4.0

R and the Seurat package (Hao et al., 2020) was used to access the COVID19 dataset. All the other (parts of) the analyses were performed in Python. For this mostly the scikit-learn library (Pedregosa et al., 2011) was used, the Matplotlib (Hunter, 2007) and Seaborn (Waskom, 2021) packages were used to render the visualisations.

For all the datasets, standardization was performed by the *StandardScaler* function of scikit-learn, stratified k-fold cross-validation was implemented using the *StratifiedKFold* function and the *train_test_split* function to render tuning subsets (see Section 2.2.3), both of these functions were also provided by scikit-learn. The assessment of all the classifiers' performance was done by making use of multiple functions provided by scikit-learn, namely: the *confusion_matrix* function, the *accuracy_score* function, the *balanced_accuracy_score* function and the *f1-score* function. Both the *accuracy_score* function and the *balanced_accuracy_score* function were implemented with the default parameters, meaning that no sample weights were used, *the accuracy_score* function was not normalised and the adjusted option of the *balanced_accuracy_score* function was not used. The *f1-score* function was also implemented with the default parameters and the 'micro' average option was used to calculate the micro-averaged F1-score per cell population. The median F1-score was then obtained by calculating the median of all the cell population specific micro-averaged F1-scores.

Three different classifiers were used for classification purposes in this dissertation: the random forest classifier, the logistic regression classifier and the linear SVM classifier. For the reproduction of the benchmarking results, the random forests classifier was implemented using the *RandomForestClassifier* from scikit-learn and the linear SVM

classifier was implemented using the *LinearSVC* classifier provided by scikit-learn. The logistic regression classifier was not implemented for the purpose of reproducing the benchmarking results. For the analyses on the AMB and COVID19 datasets, the random forest classifier was also implemented using the *RandomForestClassifier* provided by the scikit-learn library, the linear SVM classifier and logistic regression classifier were implemented using the *SGDClassifier* present in the scikit-learn library.

The parameters used for all these classifiers together with the parameters tested for hyperparameter tuning for each classifier can be found down below. For all the classifiers most of the parameters were used with the default settings, so not all the parameters will be listed. Only those that are conceptually important and those for which the parameters were changed from the default values will be mentioned.

RandomForestClassifier:

- *criterion*: 'gini' (default), the function to measure the quality of the split (see Section 2.3.3).
- *max_depth*: The maximum depth of a tree. This parameter was tuned to find an optimal value, the following max_depth values were considered : 1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 and 'None' ('None' results in no restriction on the maximum depth of a tree, expansion occurs until all the leaves are pure or all the leaves contain less than 2 samples).
- *bootstrap*: 'True' (default).

LinearSVC:

- *penalty*: 'l2' (default), the regularisation penalty used ('l2' refers to the l2-norm, more info on this can be found in Section 2.2.2).
- *loss*: 'hinge', the loss function used. The hinge loss is standardly used with support vector machines.
- *C*: The regularisation parameter. For this parameter, the following values were tested: 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10, 10^{1}, 10^{2} and 10^{3}.
- *multi_class*: 'ovr' (default), the approach used to perform a linear SVM on multi-class data (see section 2.3.1). The 'ovr' option indicates that the one-versus-all approach is used.

SGDClassifier:

- *loss*: The loss function used, 'hinge' or 'loss' was used dependent on the classification setting.
- *penalty*: 'l2' (default), the regularisation penalty used ('l2' refers to the l2-norm, more info on this can be found in Section 2.2.2).
- *alpha*: A constant that multiplies the regularization term. This parameter was used for hyperparameter tuning, the following values were considered: 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10, 10^{1}, 10^{2} and 10^{3}.

For feature selection purposes, the *RandomForestClassifier* was implemented as a feature selection preprocessing step with the *Pipeline* function provided by scikit-learn. A number of features was extracted based on a threshold by making use of the *SelectFromModel* function provided by scikit-learn. Both the feature selection step and the classification step with the *SGDClassifier* and *RandomForestClassifier* were performed using the parameters specified above but with a limited number of test values for hyperparameter tuning:

- *Max_depth* values considered for the *RandomForestClassifier* during the feature selection part of the analysis: 10, 50, and 'None'.
- *Tresholds* considered to extract the most important features using the *SelectFromModel* function: 0,3*mean, 0,7*mean and mean (with mean, the mean value of all the generated feature importances).
- *Max_depth* values considered for the *RandomForestClassifier* during the classification part of the analysis: 10, 50, 80 and 'None'.
- *Alpha* values considered for hyperparameter tuning of the *SGDClassifier* during the classification step of the analysis: 10^{-6}, 10^{-4}, 10^{-1} and 1.

Hierarchical classification was implemented with of the *hclf* package provided by Thomas Mortier, this module was downloaded locally for implementation of the analyses and so the respective versions used for the different analyses can be found with the corresponding scripts on Github.

APPENDIX C

THE COVID19 DATASET

The COVID19 dataset used in this dissertation contains gene expression count data of a COVID-19 positive patient obtained during a single-cell sequencing experiment. This dataset was obtained as part of a lager study, the details of this study can be found below.

C.1 Study overview and design

Matching bronchoalveolar lavage fluid (BALF) and blood samples were profiled from patients who have been hospitalized with a high clinical suspicion of COVID-19 (n=17) and control individuals (n=2). The analysis includes single-cell 3'RNA-sequencing along with the quantitative measurement of surface proteins using panels of more than 250 oligo-conjugated antibodies (TotalSeq A - CITEseq). The study population entails adult patients with a diagnostic or therapeutic need for bronchoscopy and the cohort consists of COVID-19 patients (n=8), control cases with a non-SARS-CoV-2 respiratory disease (n=9) and healthy controls (n=2). Patients aged 18-100 years old were eligible for study inclusion if they had clinical symptoms suggestive of COVID19 and if hospitalization was required. Healthy controls were asymptomatic and were selected from a group of patients requiring a bronchoscopy with BAL for diagnostic work-up or follow-up of other diseases. In these cases, lavage was always performed in a healthy lung lobe and SARS-CoV-2 was formally ruled-out by rRT-PCR. For 5 out of 19 patients, PBMC samples were analyzed at a secondary time point as well. This study was performed in accordance with the principles expressed in the Declaration of Helsinki. Written informed consent was obtained from all patients or a legal representative. The study was approved by the Ethics Committee of Ghent University Hospital (Belgium), AZ Jan Palfijn (Belgium) and AZ Maria Middelares (Belgium), where all samples have been collected.

C.2 Sample collection and processing for CITEseq/scRNAseq

Bronchoscopy with BAL was performed bedside using a single use disposable video bronchoscope. Bronchoscopy was only performed in hemodynamically and respiratory stable patients. In spontaneously breathing patients, an additional oxygen need of 3L/min. in rest was required. Recommended personal protective equipment was used: full face mask, disposable surgical cap, medical protective mask (N95/FFP2/FFP3), work uniform, disposable medical protective gown, disposable gloves. Three to five aliquots of 20 mL sterile normal saline were instilled into the region of the lung with most aberrations on chest CT. Retrieval was done by suctioning of the scope. BAL fluid was collected in siliconized bottles to prevent cell adherence and kept at 4 °C. BAL fluid was filtered through a 100 μm cell strainer (BD Biosciences) and centrifuged for 7 min. at 1300 rpm at 4 °C. The supernatant was removed and the BAL fluid cells were counted and subsequently processed fresh for CITEseq/scRNAseq. One million of cells was used for subsequent single-cell RNA sequencing while the remaining cells were frozen in 1 mL 90% fetal calf serum (FCS, Sigma), 10% dimethyl sulphoxide Hybri-Max (DMSO, Sigma) in a cryovial using a 5100 Cryo 1 °C Freezing Container (Nalgene) to 80 °C. Afterwards the cells were stored stored in liquid nitrogen (196 °C). Whole blood was collected in EDTA tube and processed within a maximum of 1.5 hours after collection. Whole blood separation was performed by bringing whole blood, diluted with PBS 7.2 (ThermoFisher Scientific, #20012027), in a Leucosep tube, (Greiner Bio-One, #227290), prefilled with 15 mL Lymphoprep^{TM} (Stemcell technologies, #07851), followed by a centrifugation step of 30 minutes at 1500 rpm (acceleration 5, brake 3). After isolation, the PBMCs were twice washed in PBS 7.2 and centrifuged at 350 xg for 10 minutes in a cooled centrifuge at 4 °C. Isolated PBMCs were counted, cryopreserved in 1mL FCS/DMSO 10% and stored in liquid nitrogen (196 °C).

C.3 Single-cell capture method and library preparation

All experiments have been conducted at a containment laboratory with inward directional airflow (BSL-3). BALF cells have been processed fresh, PBMC cells were frozen first and subsequently processed. One million of cells were stained with the CITE-seq antibody mix containing >250 barcoded antibodies (TotalSeq-A, BioLegend), CD45 FITC (Clone HI30, BioLegend, 3040050), and CD235a APC (2.5 μL, Clone HIR2, BD

Biosciences, 561775). When cell hashing was applied, TotalSeq-A hashing antibodies were supplemented to the CITEseq antibody cocktail. After a 30 min. incubation on ice, cells were then washed with PBS/FBS2% and spun down at 500 rcf at 4 ˚C for 5min. After resuspension in 300 L of PBS and instant staining with propidium iodide (Company, catalog number, 4 μL), PI-/CD235a- viable cells (whilst excluding red blood cells) were sorted using the BD FACSJazz. Sorted cells were spun down at 450rcf at 4 ˚C for 8 minutes. Supernatant was carefully discarded and the cell pellet was re-suspended in an appropriate volume of PBS/BSA 0.04%. Sorted cells were loaded on a GemCode NextGEM Single-Cell Instrument (10x Genomics) to generate single-cell Gel Bead-in-EMulsion (GEMs). and samples were mixed prior loading on the GemCode instrument. Single-cell RNA-Seq libraries were prepared using GemCode Single-Cell V3.1 (NextGEM) 3 Gel Bead and Library Kit (10x Genomics) according to the manufacturers instructions. Sequencing libraries were sequenced with NovaSEQ S4 flow cell with custom sequencing metrics (single-indexed sequencing run, 28/8/0/98 cycles for R1/i7/i5/R2) (Illumina). Sequencing was performed at the VIB Nucleomics Core (VIB, Leuven, Belgium).

C.4 Single-cell RNA-seq computational pipelines, processing and analysis

The raw reads were demultiplexed and mapped to a merged human/SARS-CoV-2 genome using Cell Ranger v4.0. Empty droplets and outlier cells were identified and removed based on the gene expression profile. Cells with counts in less than 200 genes and genes expressed in less than 3 cells were removed from the count matrix. Cells that were more than 5 mean absolute deviations from the median library size or median number of expressed genes were also removed, as well as cells where the % of mitochondrial reads exceeded the median by 5 mean absolute deviations. The ensuing count matrix was further processed using Seurat v3.1.5. The gene expression counts were divided by the library size and after applying a scaling factor log-transformed to normalize between cells. A centered log-ratio transform was used to normalize the antibody derived counts. Cells were clustered using the Louvain algorithm on the 50 first principal components of a subset of high variable genes and visualized on a Uniform Manifold Approximation and Projection of a batch-effect corrected embedding using Harmony (Korsunsky et al., 2019). The computational resources (Stevin Supercomputer Infrastructure) and services were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, FWO and the Flemish Government - department EWI.