Internal Report 2012–22 August 2012

# Universiteit Leiden
# Opleiding Informatica

## Automated classification of skin diseases using tile-based texture features

Erwin Haas

Supervisors: Dr. Ir. Fons Verbeek & Dr. Arno Knobbe

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

**Abstract**

The Department of Dermatology at the Leiden University Medical Center keeps a large database of medical images, used for educational purposes. This thesis is a feasibility study on automated classification of skin diseases with the use of annotated images from this database. Each annotated image is converted into tiles of which color features and Haralick's texture features are calculated. These features are then used to train the 1R, J48, NaiveBayes, Multilayer Perceptron and SMO classifiers. After classification the predictions were linked back to their original image and with a majority vote the classification of the image was obtained. This resulted in a recall on the images of 70%.

# Contents

# Chapter 1

# Introduction

At the Leiden University Medical Center (LUMC) the department of dermatology keeps a large database of several thousands of photographs of patients with skin diseases. The database is internally used for educational purposes and is supplemented regularly with new photographs.

This thesis is a feasibility study on automated classification of skin diseases by learning from the dermatology database. Next to finding a good approach to do so, it will focus also on the problems that might occur by using a dataset that is not specifically created with machine learning in mind.

## 1.1 Possible application

Not everywhere in the world are there good doctors available when you need them. Especially in third world countries, where hospitals are few and transport is limited, it is hard to find good medical care. One of the things that are done to tackle this problem is the deployment of 'flying doctors', which are general practitioners trained to do their jobs in remote villages. Although they are skilled, they are no specialists and because of this can't always rely on their own skills and knowledge.

At the same time, the availability of technology, mobile phones and internet connectivity has spread throughout the continent. This means that both doctors and patients can look up medical information on the internet.

But the availability of technology brings more possibilities in the medical world: with the use of computer vision, technology can assist patients and doctors in making diagnoses. Dermatology is one of the medical disciplines that could benefit the most from this kind of disease classification system. Imagine an iPhone application that automatically recognizes diseases from pictures that are taken with it or gives a listing of possible diseases with an advice on what medical steps to proceed with. While it is not likely that it could replace the knowledge of a real dermatologist, it could provide a simple replacement for situations in which a specialist is not available, like in Africa often is the case.

At the moment no such general solution for classifying skin diseases exists, although there are some niche problems in which computer vision is successfully deployed like determining the nature of melanomas Isasi et al. [2011].

## 1.2 The data

Most of the images in the dermatology database are taken at the hospital by one of the medical photographers, as part of the medical examination, although a smaller number is taken by patients or doctors with mobile devices or compact camera's.

The medical photographer works in a photo studio with an even background and good lighting in order to ensure that no shadows present at the subject tissue. The equipment and the setup of the photo studio, including the colour of the background canvas used has changed over the years. The photographs taken by doctors and patients vary greatly in setting and quality. Lastly, parts of an old physical non-digital photo archive have been digitalised and added to the database. Because of the large timespan and different sources, the quality, size and dimensions vary greatly across these photographs.

The database consists of a low quality version of the image, linked together with extra information in a relational database. This database contains information about

- the photograph itself, with a reference to the original image file, the date the photo was taken, the camera that has been used, the Exif[1] data and the photographer;
- the patient, with fields including sex, date of birth, name and address and patient ID for linking to the Electronic Health Record[2];
- the part of the body that is photographed;
- the disease the patient is diagnosed with, if known.

The photographs that are present in the database itself are lossy JPEG images, which are less than ideal to use with computer vision. The original images are stored in TIFF or PNG where these qualities originally existed, which is the case for all recent professionally taken photographs. Where these formats were not available, the original file is also a JPEG. In the research only the PNG / TIFF formats were used, so the lossy JPEG encoding was not an issue.

More details on the image dataset that is used can be found in Section 2.1.

## 1.3 Research goals

This research project is a feasibility study to see if images in this dermatology database can be used to automatically diagnose dermatological diseases, although the photographs were not taken with such application in mind. From the start it is clear that the database as is, isn't going to give the easiest, most accurate solution and therefore it makes little sense to try to build the best possible classifier for skin diseases. This project will focus upon the following questions, which together will answer the main research question:

- Is it possible to train a classifier that can discriminate between four skin diseases, learning from the images in the database and what accuracy can be achieved?
- Are all images in this database of use, or is there a clear distinction in characteristics?

[1] EXIF, or Exchangeable Image File, format is a metadata specification for image files taken by digital cameras among others.

[2] Electronic Heath Record is known in dutch as EPD or Electronisch Patienten Dossier.

- What information is needed in addition to the images, for example skin colour, in order to create a classifier that can discriminate between diseases?
- What kind of classifier can be used best for the purpose of discriminating between the four diseases?
- Is it possible to locate the area of the affected skin or is it only possible to mark an entire image as containing a specific disease?
- How can the creation of new pictures be improved in order to increase their usefulness in automated classification?
- What kind of features are most useful for discriminating between diseases?

## 1.4 Plan of action

To answer these questions, the following plan of action has been composed:

1. **Obtaining the dataset**, including the selection and filtering of the data;
2. **Investigating the data**, in order to see the differences between classes of pictures;
3. **Generating features for training**, converting the images into features;
4. **Annotating the images**, where necessary;
5. **Choosing a classifier**, selecting and comparing different classifiers;
6. **Experimenting**, varying some parameters and setups in order to find the best approach;
7. **Selecting areas for improvement**, in order to help the hospital build a dataset which is more valuable for future uses.

# Chapter 2

# Materials and Methods

## 2.1 Image dataset

There are over 40.000 images stored in the database at the LUMC. Because of practical reasons and privacy issues a filtered subset of 13k images was extracted. This subset excluded faces, genitals and other bodyparts that featured tattoos, spanning 311 different diagnosis. The extraction of the subset was a tedious process as many of the 'bodypart' fields were left blank or didn't state all bodyparts that were visible in the image. Because of this, all images had to be inspected manually before the dataset could be approved. This is a very tedious work to do for all 13k images. Because of this, a refined set of only the hands was manually inspected and approved for use with the research. This resulted in a dataset of 912 images of which 815 were provided with a diagnosis.

As can be seen in Figure 2.1, there are few diagnoses in the dataset with more than fifteen samples of a disease. To train and test a classifier, the more samples of a single class (diagnosis) available, the better. That's why there is again extracted a subset. This time only four diagnoses remained: *Spinocellulaircarcinoom*, *Contact Dermatitis*, *Dermatitis* and *Palmoplantar Keratoderma Hereditaria*. These four diagnoses vary in appearance, an example of each of them can be seen in Figure 2.2. The total sample count in the dataset is 77. The content of the four diagnoses is shown in Table 2.1.

The risk of such low number of samples is that of undertraining. This means that there are not enough examples for a classifier to learn some general rules for separating the diseases. For a feasibility study this is not a big problem, although a bigger data set would be recommended for future research.

Each of the images is in the PNG format or converted from TIFF to PNG. The PNG format provides a lossless encoding, preserving as much detail as possible. All images are around three megapixel in size and have 8-bit RGB channels.

## 2.2 Software components

In this section, the different software components used in this project are briefly explained.

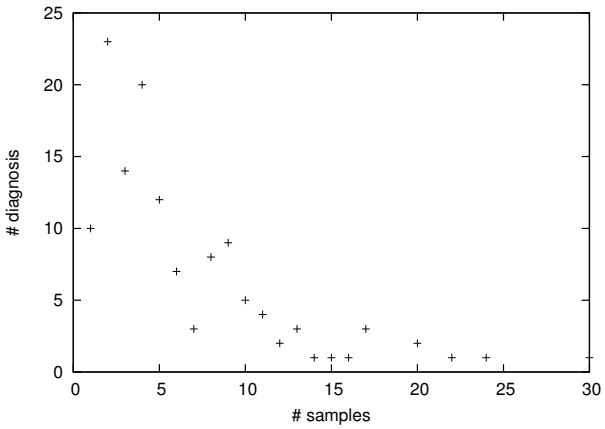



*Figure 2.1: Distribution of diagnoses and sample sizes. There are many diagnoses with a small samples set and few with a sample set $> 15$*

*(a) diagnosis-1739023, Spinocellulaircarcinoom*

*(b) diagnosis-6929067, Contact Dermatitis*

*(c) diagnosis-6929070, Dermatitis*

*(d) diagnosis-7573943, Palmoplantar Keratoderma Hereditaria*

*Figure 2.2: The four different diseases in the final dataset.*

| diagnosis | samples | # annotated samples | % annotated samples |
|---|---|---|---|
| diagnosis-1739023 | 19 | 7 | 37% |
| diagnosis-6929067 | 20 | 7 | 35% |
| diagnosis-6929070 | 13 | 11 | 85% |
| diagnosis-7573943 | 21 | 14 | 66% |
| Totals | 73 | 39 | |

*Table 2.1: Contents of the dataset*

### 2.2.1 TDR

TDR of LIACS is a software tool originally designed to annotate slices of 3D images for 3D reconstruction. With TDR it is possible to annotate images by drawing contours with different labels. This software is used in the setup to annotate the dataset used for classifying. Three kinds of contours were drawn in the images: skin, affected skin and nails. These contours divide the image in three parts: background, healthy skin and affected skin. The latter, the contour of the nails, hasn't been used in the classifier, but was added to be able to tell more about the size and orientation of the hand.

Annotating the images is a time consuming process, because of this, in a part of the annotated images only the affected parts of the skin were indicated.

### 2.2.2 OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions for real time computer vision ope, developed by Intel. The library is used by the feature detection algorithm, providing functions for reading and writing images, performing matrix operations and more.

### 2.2.3 Weka

Weka (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization wek.

## 2.3 Tiling

The amount of samples of the four diseases we are trying to classify is quite small. The diseases manifest themselves in various forms, like spots, vesicles, or an overall redness of the skin. Therefore it is not easy to look for a single kind of feature or template in the image to determine which areas of the skin are affected by the disease. The best approach is to look at the entire image in smaller parts and try to classify each of the smaller parts.

The easiest way to divide the image into smaller parts is to place a grid on it and create square tiles. Given that the tiles are small enough, most of the tiles will contain

one class each (skin, affected skin or background). Tiles that contain transitions between two classes are small in number and won't disrupt a classifier. In case it will, tiles not containing a single class in majority can be omitted from the training data feeding the classifier or can be specified as another class called *edge*.

Using the pixels of the tiles as input for a classifier won't work that well: spots and vesicles are not always in the same place within a tile and using the pixels directly would make the classifier very vulnerable to rotation and scaling. That is why the classifier uses features that are calculated from the pixels of these tiles.

## 2.4 Colour features

Images are stored in RGB colour space, defining colour as a mixture of red, green and blue. But this is not a colour space mimicking the way our human eyes would perceive colour. A more useful way for this purpose is specifying the colour of a pixel as a mixture of hue, saturation and value Zarit et al. [1999].

The colour features are simple to calculate as they are the averages and standard deviation of the values of each of the RGB and HSV channels. These are most useful separating skin from background as the difference between healthy skin and affected skin is not an absolute difference. 'Redness' of the skin, an affected region, is dependent on the skin colour and varies from sample to sample.

The colour features describe one channel at a time, making it hard for classifiers to filter for example blueish tiles and whitish tiles on a single RGB value: both contain a high value of blue. To circumvent this problem, features containing the ratio between two channels can be added. There are six channels containing colour features. Adding ratios of all the combinations gives us too many features, only slowing down the training of any classifier. Therefore only the ratios $green/blue, blue/red$ and $green/red$ are computed.

## 2.5 Texture features

Colour features have the problem that they vary under many external conditions, such as lighting, camera and natural variations in skin complexion. Texture features should not be hampered by these conditions and can provide good features for separating healthy skin from affected skin and differentiate between the different diagnosis, as long as there are differences in texture between those.

There are many kinds of feature detection algorithms based on texture available. For this project, the Haralick features are chosen Robert M Haralick [1973].

The Haralick features are a set of features which are calculated from the Gray-Level Co-occurrence Matrix. The gray-level means that there is only one channel which is looked at at the same time, not specifying that it should be a grayscale image. This matrix, with sides equal to the range of pixel values per channel, contains the number of co-occurring values of an image (or tile). Mathematically it can be defined as:

$$C_{\Delta x,\Delta y}(i,j) = \sum_{p=1}^{n}\sum_{q=1}^{n}\left\{ \begin{array}{ll} 1 & \text{if } I(p,q) = iI(p+\Delta x, q+\Delta y) = j \\ 0 & \text{otherwise} \end{array} \right\}, \qquad (2.1)$$

meaning that while looping over all pixels in the tile elements $a, b$ and $b, a$ are incremented if a pixel with value $a$ is found next to a pixel of value $b$.

Because this summation looks in the $x$ and $y$ direction of the image, it is not rotationally invariant. This is why the matrix is constructed not only of this summation, but also with a rotation of 45, 90 and $135 = -45$ degrees, creating the matrix as if the image was rotated by these degrees. The remaining neighbouring pixels do not have to be visited by the algorithm because co-occurrence is commutative and therefore these neighbours are already accounted for when the neighbouring pixel is visited.

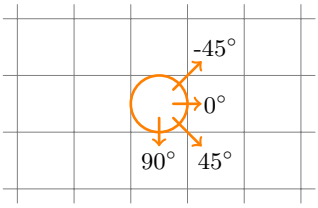



*Figure 2.3: Pixels looked at at rotational degrees of 45, 90 and 135 degrees.*

In theory a RGB image could be converted to a single channel image, by converting the three 8-bit channels to one 24-bit channel. Only would the matrix consist of an enormous $(255^3)^2 = 2^48 = 2.8 \times 10^{14}$ elements and would be a very sparse matrix[1], which would severely impact computational time and memory usage. This is why the Haralick features are calculated for one channel of the image in HSV colour space at the time. By also calculating the features for the gray level image, the interrelationship between the channels is also taken into account. The gray level image is created by averaging the three channels of the RGB channel.

Even if containing just a single 8-bit channel the co-occurrence matrix is large and sparse. This is why the Haralick features are generated from this matrix, named after R M Haralick. These features are named as follows: angular second moment, contrast, correlation, variance, inverse difference moment, sum average, sum variance, sum entropy, entropy, difference variance, difference entropy, and two information measures of correlation.

To fully cover all of these features in this paper would be too much, so three of them are shown below. First the angular second momentum,

$$f_{angular\ second\ momentum} = \sum_i \sum_j (p(i,j))^2, \tag{2.2}$$

where $p(i,j)$ is the value at $i,j$ in a normalized co-occurrence matrix. Second is the contrast,

$$f_{contrast} = \sum_{n=0}^{N_g-1} n^2 \left( \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \right), |i-j| = n, \tag{2.3}$$

where $N_g$ is the number of distict values in the images' channel, in our case 256. The

[1]A sparse matrix is a matrix populated primarily with zeros. If the majority of elements differ from zero then it is common to refer to the matrix as a dense matrix Stoer and Bulirsch [2002].

third feature displayed is the second correlation information measure,

$$f_{correlation\ info\ 2} = \sqrt{1 - e^{-2(\mathrm{HXY2}-\mathrm{HXY})}})\text{with} \tag{2.4}$$

$$\mathrm{HXY} = -\sum_i \sum_j p(i,j) \log\left(p(i,j)\right) \text{ and} \tag{2.5}$$

$$\mathrm{HXY2} = -\sum_i \sum_j p_x(i) p_y(j) \log\left(p_x(i) p_y(j)\right), \tag{2.6}$$

where $p_x(i)$ is $i$th entry in the marginal-probability matrix obtained by summing the rows of $p_x(i) = \sum_{j=1}^{N_g} p(i,j)$ and $p_y(j)$ the summation the columns.

## 2.6 Scaling the image

The Haralick texture features are based on the co-occurrence matrix, therefore are dependent on the scaling of the image. A close-up image of a single finger reveals much more detail and texture than an overview of the entire hand and forearm. By scaling images down before they are tiled and the co-occurrence matrix is build, not only the number of tiles is reduced, the amount of detail is reduced as well. This means that the algorithm will generate somewhat more alike features for a scaled down close up image of a finger, compared to a unscaled image of the entire forearm.

However, it is not possible to determine the size of the subject in the image beforehand, because no external information about the subject is present and no visual markers of known size or distance are available in the image. A way to circumvent this problem is to generate the features with different scalings of the image. Nevertheless multiple executions of the algorithm on one image increases the computational time a lot and the algorithm could benefit greatly by providing 'naturally' scaled images.

## 2.7 Classifiers

Weka provides a range of different classifiers. These classifiers have differing input and output types and can't be compared in a straight forward manner. For the testing four different classifiers are used, their differences being put aside below:

- 1R, algorithm that uses only one rule to classify examples;
- C4.5, an algorithm that creates a decision tree;
- Naive Bayes, a probability based classification algorithm;
- Platt's Sequential Minimal Optimization algorithm, a support vector machine;
- Multilayer Perceptron; a type of neural network.

### 2.7.1 1R

Described in 1993 by Holte Holte [1993], the 1R is a one level decision tree algorithm, which ranks the attributes on error rate and choses the attribute with the smallest error.

The algorithm regards all numeric attributes as continuous and uses a straight forwardly method to divide the range into several intervals. To counter the risk of over fitting when a perfect split is made, creating intervals of single items, a minimum size of the intervals is required.

### 2.7.2 C4.5

The J48 decision tree classifier is a java implementation of the well known C4.5 algorithm Quinlan [1993]. The C4.5 algorithm builds a desicion tree in the same way as ID3, using the concept of information gain. Information gain can be defined as

$$G(S, A) = E(S) - \sum_{i=1}^{m} f_S(A_i) E(S_{A_i}), \tag{2.7}$$

where $G(S, A)$ is the gain of set $S$ after split over attribute $A$, $E(S)$ is the information entropy of set $S$, $m$ is the number of different values of attribute $A$ in $S$, $f_S(A_i)$ is the proportion of the items possessing $A_i$ as a value for $A$ in $S$ and $S_{A_i}$ is the subset of $S$ containing the items where the value of attribute $A$ is $A_i$. The information entropy, $E(S)$ can be defined as

$$E(S) = -\sum_{j}^{n} = 1 f_S(j) \log_2 f_S(j), \tag{2.8}$$

where $n$ is the number of different values of the attribute in $S$ and $f_S(j)$ is the frequency (proportion) of the value $j$ in the set $S$.

The algorithm recursively splits the tree on the attribute with the highest information gain, creating subsets of the remaining data for each child node and repeating the information gain calculations for each attribute. After the algorithm is done, branches with little information gain are pruned by replacing them with leaf nodes.

### 2.7.3 Naive Bayes

The Naive Bayes classifier is a simple probabilistic classifier, which is based on applying Bayes' theorem. The algorithm assumes that all features are independent of any other feature. In reality this assumption is generally wrong, but nevertheless the Naive Bayes classifier produces good results for complex problems.

The classifier depends mostly on the application of Bayes' rule,

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \text{ en Norvig [2003].} \tag{2.9}$$

In the case of building a classifier, the algorithm perceives as evidence the *effect* of some unknown *cause* and wants to know the cause. In that case, Bayes' rule becomes

$$P(cause|effect) = \frac{P(effect|cause) \times P(cause)}{P(effect)}. \tag{2.10}$$

Because the algorithm assumes that all features (*causes*) are independent, it can use the condititional independence to combine multiple features by the definition of

$$P(X, Y|Z) = P(X|Z) \times P(Y|Z). \tag{2.11}$$

Combining the two formulas gives the naive Bayes model

$$P(Cause, Effect_1, \ldots, Effect_n) = P(Cause) \prod_i P(Effect_i|Cause), \tag{2.12}$$

which forms the basis of the algorithm. By building a probability table and applying these rules, the classifier builds a probabilistic model.

These equations are only valid for as long as the features are independent. In the case of the colour and texture features, this is does not hold up. Even in this situation, the naive Bayes could provide a pretty good model Domingos and Pazzani [1997].

These equations require that the probability distribution of some attribute is known. This is an easy task with nominal and binary attributes, as they can be estimated from the training set by dividing the number of samples of given value by the total number of samples. Numeric data is however continuous and would first have to be discretized in order to use this technique. This can be done by binning, in which the numerical data is converted in an nominal attribute and its prior can be estimated.

There is however another way, when the attributes' data is assumed to be distributed according to a Gaussian distribution for each of the classes. When confronted with a numeric attribute $x$, the algorithm first calculates the mean ($\mu_c$) and variance ($\sigma_c^2$) of $x$ in each class $c$. Now the probability of some value given a class, $P(x = v|c)$ can be calculated using the equation for the Normal distribution,

$$P(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}. \tag{2.13}$$

In reality both the discretizing technique and the Normal distributed estimation are used, where the former can outperform the Normal distributed estimation, depending on the discretizing method Dougherty et al. [1995].

### 2.7.4 Support Vector Machine

A support vector machine tries to separate instances by a linear hyperplane. Instead of minimizing the *empirical loss* on the training data, it tries to minimize the expected *generalization loss*, by maximizing the separator that is farthest away from all the examples en Norvig [2003]. This means that the hyperplane used as a separator is the maximum margin separator. The separator is defined as the set of points $\mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = 0$, with $\mathbf{w}$ the weight vector.

The optimal separator is found by solving the equation

$$\text{Maximize } L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=0}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j), \tag{2.14}$$

under the constraints $\alpha_j \geq 0$ and $\sum_j \alpha_j y_j = 0$. Once the vector $\alpha$ is calculated, the expression for the separation function itself is

$$h(\mathbf{x}) = \text{sign} \left( \sum_j \alpha_j y_j (\mathbf{x} \cdot \mathbf{x}_j) - b \right). \tag{2.15}$$

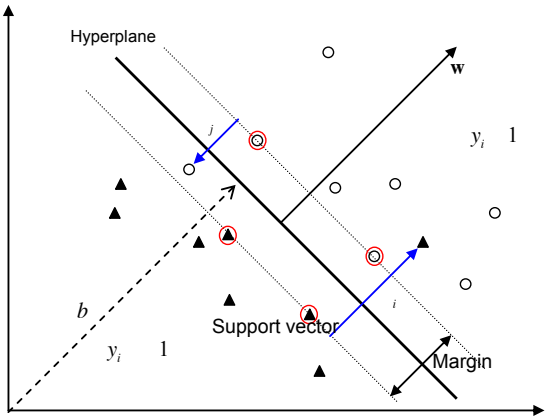



*Figure 2.4: Non-linear separable classification problem jin Wang et al. [2009]. The hyperplane separates the two classes with the highest margin possible.*

This function gives the classification for feature vector $\mathbf{x}$, -1 or 1.

If examples are not linearly separable, training samples from the input space are mapped by a function $F(\mathbf{x})$ into a higher-dimensional feature space, in such way that the examples are linearly separable in the new feature space.

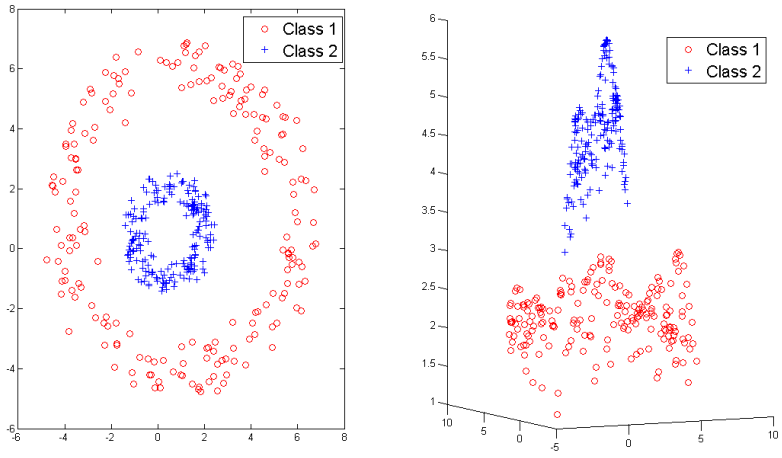



*Figure 2.5: This figureFletcher [2009] shows how a non-linear separable classification problem can be converted to a linear separable classification problem by converting the feature space into a new feature space, in this case by the radial basis kernel function* $k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\left(\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)}$.

Support vector machines can be put to use in a multi-class classification problem, by constructing separate separation functions for each pair of classes and comparing the results with each other. In this project, the SMO-class of Weka is used Wek [a], which implements John C. Platt's sequential minimal optimization algorithm for training a support vector classifier using polynomial or RBF kernels Platt [1999].

### 2.7.5 Multilayer Perceptron

A perceptron, or neural network, is a graph of nodes, divided in an input layer, one or multiple hidden layers and an output layer. Each node in one layer connects with a certain weight $w_{i,j}$ to every node in the next layer, as can be seen in Figure 2.6. The output of a node is determined by an activation function $a_j = g(in_j)$ which takes as an argument the weighted sum of the inputs of the node,

$$in_j = \sum_{i=0}^{n} w_{i,j} a_i. \tag{2.16}$$

This activation function is a sigmoid function,

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{2.17}$$

which gives a smooth transition between 0 and 1.

*Figure 2.6: Diagram of a multi-layer feedforward artificial neural network mul.*

The input nodes take normalized versions of the feature vector of an example. These values are fed forward through the network. The nodes in the output layer represent the classification. In the case the classification is not numeric but nominal, an output node for each of the nominal values is created. When selecting the classification, the node with the highest output is taken.

Training a perceptron is done by back-propagation, a technique which updates the weights of all nodes moving from the output layer through the hidden layer(s) to the input layer. First an example is fed through the network, generating values on the output nodes. These values are compared with the desired outcome: the classification of the example. The error or difference between those two values is propagated backwards through the network, adjusting the weights of each node accordingly.

The function for updating the weights of the connections to a node is

$$w_{j,k} \leftarrow w_{j,k} + \alpha \cdot a_j \cdot \Delta_k \cdot \ en\ Norvig\ [2003], \tag{2.18}$$

with $\alpha$ the learning speed of the network, $w_{j,k}$ the weight between nodes j and k, $\Delta_k$ is the error times the derivative of the sigmoid function,

$$f'(x) = f(x)(1 - f(x)). \tag{2.19}$$

By repeatedly feeding examples through the network and adjusting the weights, the network is trained.

# Chapter 3

# Implementation

This section explains how the overall setup of the classification process is set up and how the different technologies are used together.

## 3.1 Overall setup

The overall setup used to move from input images to features to a classifier is divided in five steps: annotation, tiling, feature calculation, training and testing.

The first step is annotating the images. This is done with TDR, as explained in Section 2.2.1. The annotations are stored in two files: an XML file describing the different annotations used and location of the contour data in the second file, which is a binary file. In this file, each byte represents two 4-bit directions in which the contour moves with respect to last pixel location.

The second and third step are performed by self-written software, in C++. The program reads the images of the trainingsset and reconstructs their contours, converting them into a mask per pixel. The program then scales each image if instructed and divides the image into tiles. For each tile features are calculated and stored in an output file, which is in the ARFF-format.

For the fourth step, training a classifier, Weka is used. The ARFF-file is slightly modified to remedy errors with missing values and then used as input for the classifier.

The last step, testing the classifier is done internally in Weka by a tenfold cross validation. The exact methods of testing differ per experiment and will be detailed where needed.

## 3.2 Scaling and tiling

The program responsible for creating features out of the images accepts two parameters: tile size and scale. The program scales first and then cuts the image into tiles. This means that the same number of tiles are produced with a scaling factor of one and a tile size of 32 pixels as a scaling factor of 1/2 and a tile size of 16 pixels.

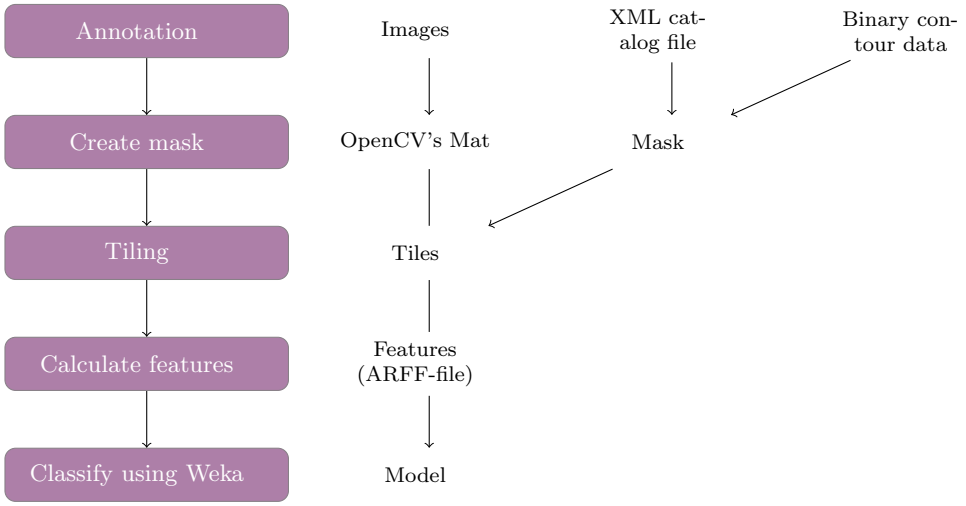



*Figure 3.1: Flow chart of data flow in the system*

For easy comparison of different scaling factors powers of two are used as the tile size. This gives the possibility to split a tile in two, four or any other power of two smaller tiles without ending up with a non integer number.

The scaling is done by the OpenCV function `resize`, which uses bilinear interpolation to scale down the image. This way, pixel values are determined by the average of their source pixels, not just one of the source pixels. As a CCD can be seen as a collection of buckets in which photons are collected, this is essentially the same process as averaging the photon density in the area of the bucket. By scaling down in a similar way, the image is scaled down as if the camera was further away from the subject, which was the reason for scaling down in the first place.

The image is cut into tiles by Algorithm 3.1

**Algorithm 3.1** Cutting the image in tiles.

```
image ← resize(image, scaling)
for x = 0 → (image.cols − 2)/tileSize.width do
    for y = 0 → (image.rows − 2)/tileSize.height do
        tileRect ← createRect(x ∗ tileSize.width + 1,
              y ∗ tileSize.height + 1, tileSize.width, tileSize.height)
        tile ← getImageRegion(image, tileRect)
        determineClass(tile)
        calculateFeatures(tile)
    end for
end for
```

### 3.2.1 Determining the tile classification

The classification of each tile is determined by the classification of the pixels it contains. This classification of the pixels is derived from the mask that was created by annotating

the image. As will be explained in Section 4.1.2, two classifications will be calculated for each image:*skin/background* and *not-disease*.

For the *skin/background*, the class of the tile is determined by a majority of 33% of the pixels, where the disease classes take precedence over skin and skin takes precedence over background. If the image isn't annotated with background and normal skin, the tile is classified as unknown.

For hte *not-disease*, classification is simpler: if more than 33% of the pixels is disease-affected skin, the classification is `disease`. If not `disease`, the classification is `not-disease`. Algorithm 3.2 shows the calculations of both classifications in more detail.

**Algorithm 3.2** Determining the classification of a tile.

**for all** *classes* **do**
  $tmpMask \leftarrow \text{threshold}(pixelMask, \text{valueOfClassInMask}(class))$
  $percentage[class] \leftarrow \text{cv :: countNonZero}(tmpMask)$
    $/(pixelMask.rows * pixelMask * cols)$
**end for**
**if** $percentage[\text{`}disease'] > 0.33$ **then**
  $classification_{skin/background} \leftarrow image.disease$
  $classification_{not-disease} \leftarrow image.disease$
**else if** $percentage[\text{`}normal - skin'] > 0.33$ & skinIsAnnotated($image$) **then**
  $classification_{skin/background} \leftarrow \text{“}normal - skin''$
**else if** $percentage[\text{`}background'] > 0.33$ & skinIsAnnotated($image$) **then**
  $classification_{skin/background} \leftarrow \text{“}background''$
**else**
  $classification_{skin/background} \leftarrow \text{`}?'$
  $classification_{not-disease} \leftarrow \text{`}no - disease'$
**end if**

## 3.3 Feature calculation and co-occurrence matrix

After the image is split into tiles, the features are calculated. For calculating the Haralick features, a co-occurrence matrix is generated for the tile. This task is done by a modified GLCM class, where GLCM stands for Grey Level Co-occurrence Matrix, which means that there is only one channel for which the matrix is calculated. Because of this, separate matrices are created for each channel in RGB and HSV-space, resulting in separated features. The channels are split by OpenCV's `split` function.

The colour based features are calculated with OpenCV's `meanStdDev` function, calculating both the mean and the standard deviation of each channel. The Haralick features are calculated by the GLCM class. All features are outputted in an ARFF file, for use with the classifier.

## 3.4 Classifiers

All classifiers that are used are written in Java for Weka. This section elaborates on Weka's implementation of the algorithm and the settings that are provided.

### 3.4.1 J48

J48 Wek [b] is Weka's implementation of the C4.5 decision tree algorithm described in Section 2.7.2 and can produce a unpruned or pruned tree. The possible settings with their values are presented in Table 3.1.

| Setting | Description | Used value |
|---|---|---|
| `binarySplits` | Whether to use binary splits on nominal attributes when building the tree | false |
| `collapseTree` | Whether parts are removed that do not reduce training error | true |
| `confidenceFactor` | The confidence factor that is used for pruning (smaller values incur more pruning) | 0.25 |
| `minNumObj` | The minimum number of instances per leaf | 2 |
| `subtreeRaising` | Whether to consider the subtree raising operation when pruning | true |
| `unpruned` | Whether pruning is performed | false |

*Table 3.1: Settings and their values for the J48 algorithm.*

### 3.4.2 NaiveBayes – Naive Bayes implementation

Weka's implementation of the Naive Bayes algorithm, NaiveBayes Wek [c], is very straight forward. From the website:

> *Class for a Naive Bayes classifier using estimator classes. Numeric estimator precision values are chosen based on analysis of the training data. For this reason, the classifier is not an* `UpdateableClassifier` *(which in typical usage are initialized with zero training instances) – if you need the* `UpdateableClassifier` *functionality, use the* `NaiveBayesUpdateable` *classifier. The* `NaiveBayesUpdateable` *classifier will use a default precision of 0.1 for numeric attributes when buildClassifier is called with zero training instances.*

For estimating the priors on numeric data, the attributes are assumed to be normal distributed. Optionally the algorithm can use kernel estimation instead of a single normal distribution or use binning to discretize the numeric attributes.

In our testing purposes, the default settings were used.

### 3.4.3 SMO – Support Vector Machine

From the website:

> *Implements John C. Platt's sequential minimal optimization algorithm for training a support vector classifier using polynomial or RBF kernels. This implementation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default. (Note that the coefficients in the output are based on the normalized/standardized*

*data, not the original data.) Multi-class problems are solved using pairwise classification.*

*To obtain proper probability estimates, use the option that fits logistic regression models to the outputs of the support vector machine. In the multi-class case the predicted probabilities will be coupled using Hastie and Tibshirani's pairwise coupling method. Note: for improved speed standardization should be turned off when operating on SparseInstances. Wek [a]*

As for the settings, the values shown in Table 3.2 were used.

| Setting | Description | Used value |
|---|---|---|
| `buildLogisticsModel` | The minimum number of instances per leaf | false |
| `c` | The complexity constant C | 1 |
| `epsilon` | The epsilon for round-off error | $10^{-12}$ |
| `filterType` | Determines how the data will be transformed | Normalize |
| `kernel` | The Kernel to use | `PolyKernel` |
| `toleranceParameter` | The tolerance parameter | $10^{-3}$ |

*Table 3.2: Settings and their values for the SMO algorithm.*

### 3.4.4 MultilayerPerceptron – Multilayer Perceptron

The MultilayerPerceptron class Wek [d] uses a network of nodes that can be build by hand or created by the algorithm. Default, it creates one hidden layer of $\frac{|\text{attributes}|+|\text{classes}|}{2}$ nodes. All data is run multiple times through the network, the default number of epochs is 500. For this classifier all the default settings were used. These settings are shown in Table 3.3

| Setting | Description | Used value |
|---|---|---|
| `hiddenLayers` | The hidden layers to be created for the network. (Value should be a list of comma separated Natural numbers or the letters 'a' = (attribs + classes) / 2, 'i' = attribs, 'o' = classes, 't' = attribs .+ classes) for wildcard values | a |
| `learningRate` | Learning Rate for the back propagation algorithm. | 0.3 |
| `momentum` | Momentum Rate for the back propagation algorithm. | 0.2 |
| `nominalToBinaryFilter` | A NominalToBinary filter will be used. | true |
| `normalizeAttributes` | Whether to normalize the attributes | true |
| `normalizeNumericClass` | Whether to normalize numeric classes | true |
| `trainingTime` | Number of epochs to train through. | 500 |

*Table 3.3: Settings of the MultilayerPerceptron classifier of Weka*

### 3.4.5 Running the classifier

Running the feature generator and the classifiers in various parameter settings is a time consuming task, especially for smaller tile sizes. This is why a set of scripts are written to automatically distribute the computation over multiple computers. It consists of a master and a slave script, the master giving out pieces of work to the slaves, like calculating the features for a given tile size and scaling, or running a classifier. When a slave computer is finished, the result is copied back to the master.

# Chapter 4

# Results

In this section the results of the tiling algorithm and the trained classifiers are shown and discussed. The experiments that are conducted will make it possible to answer our research questions.

## 4.1 About the training data

Before training the classifiers, it is necessary to look into the data that will be used. This data is created by the tiling algorithm for various parameter settings for tile size and scaling. In this section, these data sets are investigated. Also, a baseline is determined in order to be able to tell anything about the accuracy of the classifiers.

### 4.1.1 Classifications in the data set

The distribution of diseases in the source image data set was the topic of Section 2.1. This section elaborates on the training data: the calculated features from the tiles. The algorithm detailed on in Section 3.2.1 assigns two classifications to each tile. Because Weka can only train for one classification, the dataset that is created by the feature program is reformatted and saved as two seperate datasets, each containing a classification. For the rest of this chapter, the dataset that is used will be specified for all results. The dataset containing the distinction between skin and background tiles will be referenced as *skin/background*, the other as *not-disease*.

The features are calculated for various tile sizes and scaling factors. Table 4.1 shows the distribution of classes in each of these datasets. A graphical display in Figures 4.1 and 4.2 clearly shows the skewed distribution of the classes.

### 4.1.2 Using the unclassified tiles

As is visible in Table 4.1, there are a lot of missing classifications for the *skin/background* dataset. This is because in the image data set, not all images were fully annotated. A subset of the image was only provided with annotation of disease-affected areas.

| Tile | Dataset | a | b | c | d | e | f | g | ? | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| 256 | skin | 87 | 23 | 35 | 85 | 362 | 224 | - | 549 | 81 |
| | | 10,7% | 2,8% | 4,3% | 10,4% | 44,4% | 27,5% | - | 67,3% | - |
| | no-disease | 87 | 23 | 35 | 85 | - | - | 1.135 | - | 1.36 |
| | | 6,4% | 1,7% | 2,6% | 6,2% | - | - | 83,2% | - | - |
| 128 | skin | 333 | 93 | 172 | 341 | 1.443 | 1.099 | - | 2.369 | 3.48 |
| | | 9,6% | 2,7% | 4,9% | 9,8% | 41,5% | 31,6% | - | 68,1% | - |
| | no-disease | 333 | 93 | 172 | 341 | - | - | 4.911 | - | 5.85 |
| | | 5,7% | 1,6% | 2,9% | 5,8% | - | - | 83,9% | - | - |
| 64 | skin | 1.268 | 372 | 758 | 1.383 | 5.942 | 5.321 | - | 10.345 | 15.04 |
| | | 8,4% | 2,5% | 5,0% | 9,2% | 39,5% | 35,4% | - | 68,8% | - |
| | no-disease | 1.268 | 372 | 758 | 1.383 | - | - | 21.608 | - | 25.38 |
| | | 5,0% | 1,5% | 3,0% | 5,4% | - | - | 85,1% | - | - |
| 32 | skin | 4.909 | 1.386 | 2.995 | 5.360 | 23.903 | 22.397 | - | 42.244 | 60.950 |
| | | 8,1% | 2,3% | 4,9% | 8,8% | 39,2% | 36,7% | - | 69,3% | - |
| | no-disease | 4.909 | 1.386 | 2.995 | 5.360 | - | - | 88.544 | - | 103.194 |
| | | 4,8% | 1,3% | 2,9% | 5,2% | - | - | 85,8% | - | - |
| 16 | skin | 19.277 | 5.383 | 11.893 | 21.187 | 96.563 | 93.844 | - | 172.604 | 248.147 |
| | | 7,8% | 2,2% | 4,8% | 8,5% | 38,9% | 37,8% | - | 69,6% | - |
| | no-disease | 19.277 | 5.383 | 11.893 | 21.187 | - | - | 363.011 | - | 420.751 |
| | | 4,6% | 1,3% | 2,8% | 5,0% | - | - | 86,3% | - | - |
| 8 | skin | 76.512 | 21.210 | 47.298 | 84.133 | 388.530 | 383.440 | - | 697.687 | 1.001.123 |
| | | 7,6% | 2,1% | 4,7% | 8,4% | 38,8% | 38,3% | - | 69,7% | - |
| | no-disease | 76.512 | 21.210 | 47.298 | 84.133 | - | - | 1.469.657 | - | 1.698.810 |
| | | 4,5% | 1,2% | 2,8% | 5,0% | - | - | 86,5% | - | - |
| Average | skin | - | 8,7% | 2,4% | 4,8% | 9,2% | 40,4% | 34,5% | - | 68,8% |
| | no-disease | - | 5,2% | 1,4% | 2,8% | 5,4% | - | - | 85,1% | - |

*Table 4.1: Distribution of classes in the training sets. For each tile size it displays the absolute number of tiles and the percentage distribution for both the datasets. Legend: a = diagnose-1739023, b = diagnose-6929067, c = diagnose-6929070, d = diagnose-7573943, e = normal-skin and f = background, g=no-disease.*

Other parts can thus be either `normal-skin` or `background`. This is why the tiling algorithm can't determine the class of these tiles, which is why they are marked as `unknown` or `?`. Weka just ignores unknown classifications, so the size of the *skin/background* dataset is decreased by the number of unknown classes, which is why the percentages of the diseases are higher in these sets.

In order to use the entire data set, the *disease/not-disease* dataset was created. In this dataset the classes `normal-skin` and `background` were grouped together. Because there is only a single class aside the disease classes, the unknown values – containing tiles that are either background or normal-skin, see Section 3.2.1 – can also be classified as `not-disease`.

Putting these three classes together further shifts the distribution to the non-disease side, because all of the unknown tiles are now seen as well.

### 4.1.3 Effects of the tile size on the distribution

The Figures 4.1 and 4.2 also show that the percentage share of the diseases slightly decreases as the tile size decreaeses. This happens because smaller tile sizes correspond with more detail. Each halving of the tile size splits each large tile into four smaller tiles. Where as a third of the original pixels had to be annotated as disease in order to get the entire tile classified as such, with the four smaller tiles each of the smaller tiles have to contain a majority of 33% before all of the tiles are classified as diseaes. This is not likely to be true for tiles on the edge of an affected area. When these are divided into four smaller tiles, it is likely that the disease covers not all the tiles for more than 33%. This means that the number of diseased tiles will decrease when the tile size is also decreased.

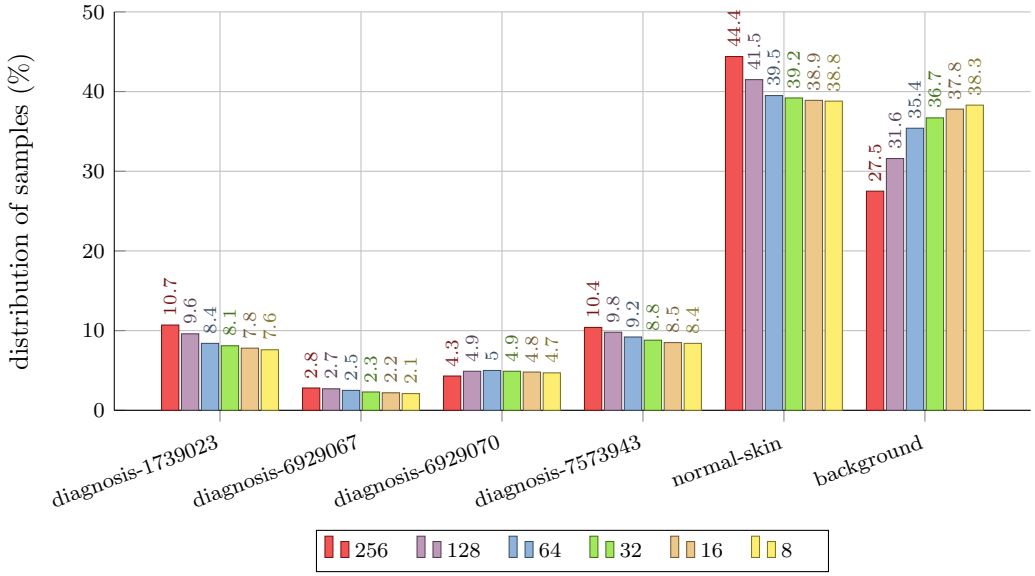



*Figure 4.1: Distributions of classes in the* skin/background *dataset, for various tile sizes. These numbers are calculated with a scaling factor of 1 and will slightly differ with other scalings. Missing values are not taken into account.*

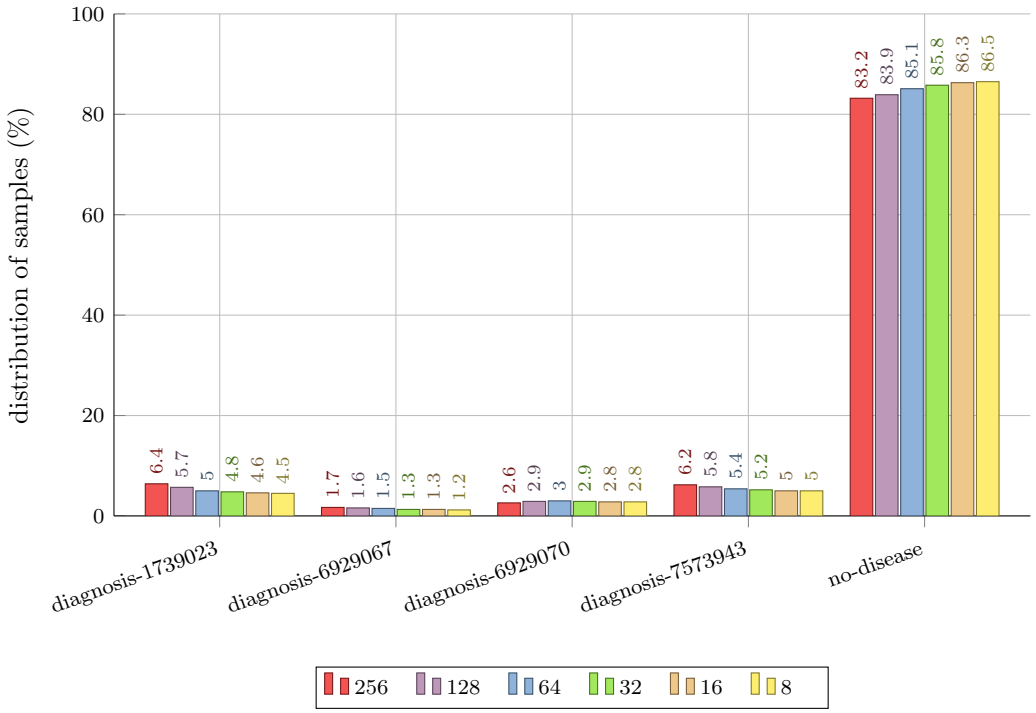



*Figure 4.2: Distributions of classes in the* not-disease *dataset, for various tile sizes. These numbers are calculated with a scaling factor of 1 and will slightly differ with other scalings.*

### 4.1.4 Determining the baseline

There are many approaches for defining a baseline accuracy. Two possibilities were considered: Weka's `ZeroR` algorithm and a random guess.

The `ZeroR` algorithm simply selects the largest class in the data set and classifies all input as that class. The accuracy that follows could be a baseline for putting the performance measured in other classifiers into perspective. Though this is a very crude algorithm, in our very unbalanced data set especially the *not-disease* dataset it focusses on the classes that are no diseases. This means that even if it tells something about the overall accuracy of the classifiers, it doesn't provide a baseline for the expected accuracy of the diseases.

To create a baseline for each class seperately, a theoretical classifier is used that randomly guesses one of the possible classifications. This gives a very low and simple baseline for each class. Because a single measure is needed, the average of the distributions shown in Table 4.1 are used to calculate the baseline. Table 4.2 shows the baseline accuracy values that are used in the following sections, where they will be referred to as "baseline".

| | *Skin/background* | | *No-disease* | |
|---|---|---|---|---|
| Class | Avg. share | Random acc. | Avg. share | Random acc. |
| Diagnosis-1739023 | 8.69% | 1.45% | 5.15% | 1.03% |
| Diagnosis-6929067 | 2.42% | 0.40% | 1.44% | 0.29% |
| Diagnosis-6929070 | 4.78% | 0.80% | 2.83% | 0.57% |
| Diagnosis-7573943 | 9.19% | 1.53% | 5.45% | 1.09% |
| Normal-skin | 40.38% | 6.73% | | |
| Background | 34.54% | 5.76% | | |
| No-disease | | | 85.13% | 17.03% |
| Average accuracy | | 2.78% | | 4.00% |
| Total random accuracy | | 1/6 | | 1/5 |
| Total `ZeroR` accuracy | | 40.38% | | 85.13% |

*Table 4.2: Baseline accuracy for each class seperately and an average accuracy. The accuracy per class is determined by a theoretical random classifier, resulting in a total accuracy of* 1/6 *for the skin/background dataset and* 1/5 *for the no-disease dataset. Total accuracy is also determined by the ZeroR algorithm, resulting in the share of the largest class as total accuracy.*

## 4.2 Contribution of tile size and scaling

The tiling algorithm can produce features for various tile sizes and can also scale down images before dividing them into tiles. To see if altering the tile size and/or the scale has any effect on the performance, datasets with various tile sizes and scalings were produced. This sections shows if there is any relationship between the tile size, scaling and the accuracy of a classifier. Because using all classifiers discussed in Section 3.4, would produce too much data, only the classifier J48 was used. Exploratory testing showed that this classifier gives relatively high results with reasonably training time.

There are two kinds of tile sizes that are worth looking at: the size of the tiles that are used to calculate the features and the original tile size before scaling. Up to now, everywhere the tile size was mentioned, a scaling of one was used, so the two kinds of tile size were the same. From this point on, we'll refer to the former as "scaled tile size" and the latter as "original tile size". The relation between those two indicated by

$$Original\ tile\ size = \frac{Tile\ size}{Scale\ factor} \tag{4.1}$$

and shown in Table 4.3.

| | Scaling factor | | | | | |
|---|---|---|---|---|---|---|
| Original tile size | 1 | 1/2 | 1/4 | 1/8 | 1/16 | 1/32 |
| 256 | 256 | 128 | 64 | 32 | 16 | 8 |
| 128 | 128 | 64 | 32 | 16 | 8 | - |
| 64 | 64 | 32 | 16 | 8 | - | - |
| 32 | 32 | 16 | 8 | - | - | - |
| 16 | 16 | 8 | - | - | - | - |
| 8 | 8 | - | - | - | - | - |

*Table 4.3: Relation between the original tile size and the scaling factor: the scaled tile size, the size of the tile where features are calculated from.*

Looking at the scaled tile size, the size of the tiles used to calculate the features, tells something about the usefulness of the features in relation to the amount of pixels that were used to calculate them. In Figure 4.3 the accuracy of the J48 classifier was plotted as a function of the tile size, for various scalings. The trend line indicates a slight decrease in accuracy as the tile size increases, suggesting that smaller tile sizes are more useful than the larger ones.

Although Figure 4.3 indicated a slight decrease of accuracy for larger tile sizes, it is to be expected that this decrease has another reason. For large tile sizes, no or moderate scaling could be done, since scaling down the image too much would result in just a few tiles for each image and classification would be pointless. Even for the data that is generated, it is difficult to compare them because the features used to train the classifiers do not originate from the same tiles and as such can vary merely due to the tiling algorithm had different input.

To take a better look at the effect of the scaling factor and tile size on the accuracy that the J48 classifier can achieve, the original tile size is taken into account. The original tile size is used to create groups of combinations of tile size and scaling, such that each

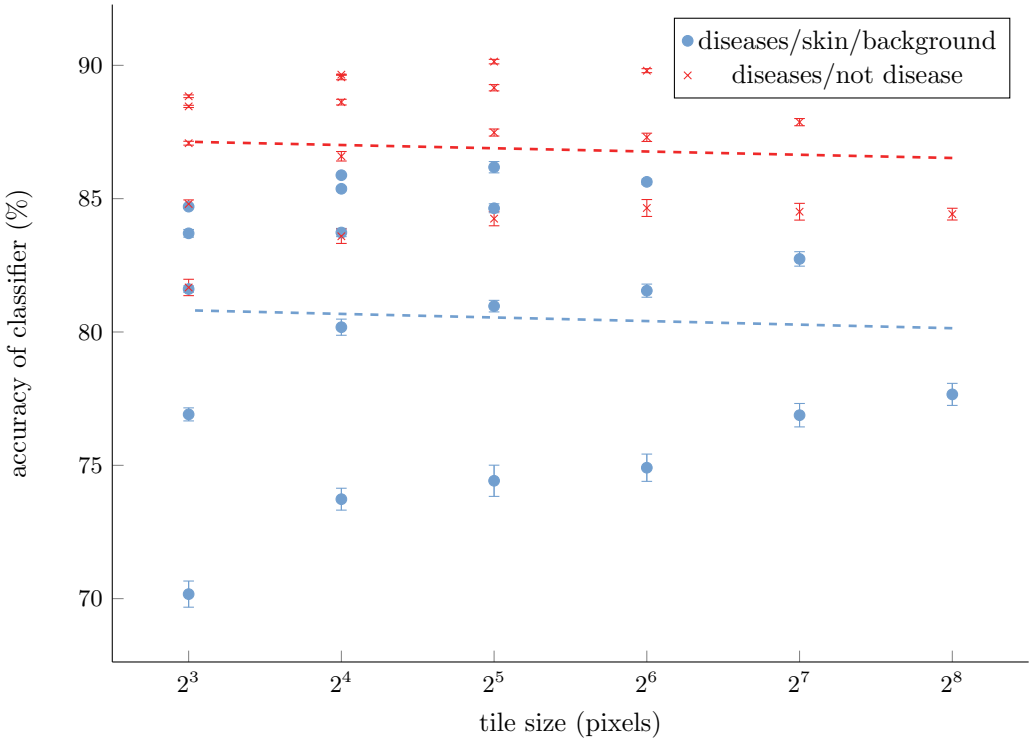
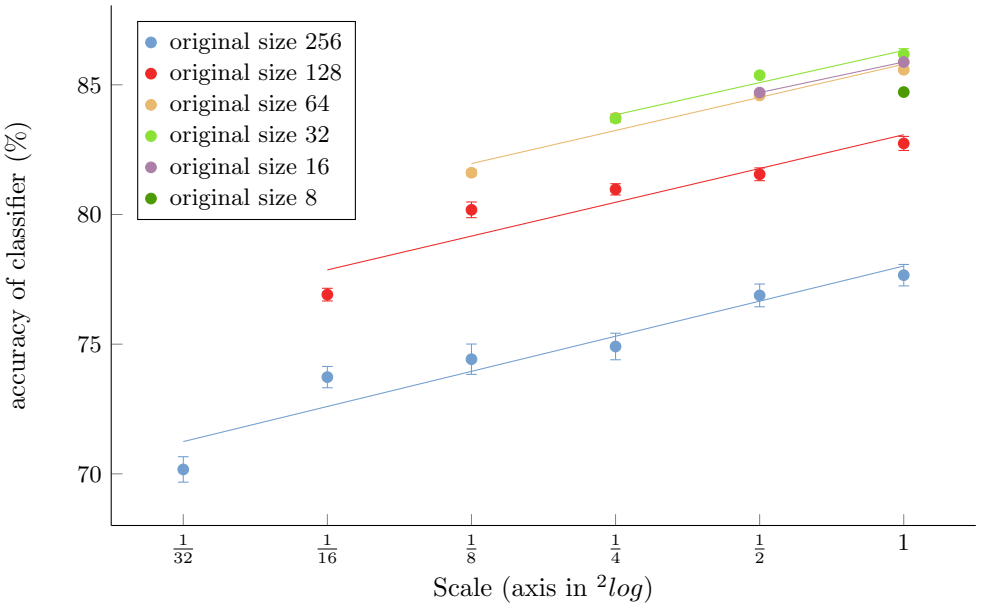



*Figure 4.3: Accuracy of the J48 classifier trained with various tile sizes and scalings plotted against the tile size. The trend line, computed by linear regression, shows a slight decline as the tile size increases.*
*Training of the classifier was done multiple times, the error bars indicating the 95% confidence intervals of the standard error. The number of repetitions vary for each combination, see Tables A.1 and A.2 in the Appendix for exact numbers.*

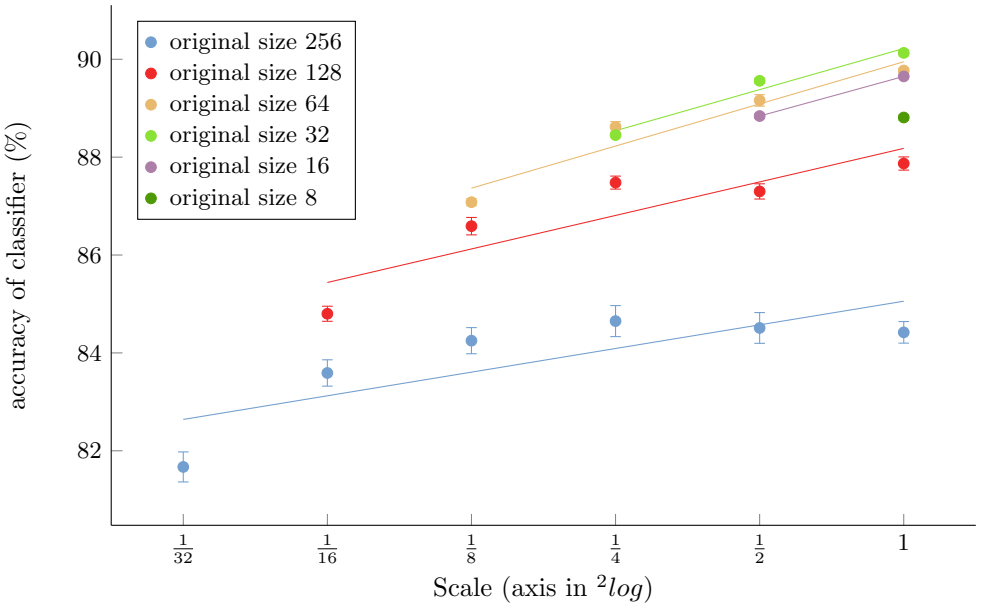
group contains all 'scaled down' versions of an original tile, as if the tile was first cut out the original image and then scaled down. In Table 4.3 the relation between these three parameters can be seen, as well as the grouping that occurs. When plotting the obtained accuracy of the J48 classifier against the scale size and colouring each grouping of scale and tile size distinctively, as can be seen in Figure 4.4, a few things may be observed.

- Within a group of tiles with the same origin in the source image, scaling down decreases the accuracy of the classifier. Each group shows about the same level of decline, with an exception of the original size 16 group in Figure 4.4b.
- There is a clear ranking between the accuracy of the different groups, the trend lines do not cross each other.
- There seems to be almost no difference in the accuracy of the original tile sizes 32, 64 and 16. Larger original tile sizes clearly perform less and also the $8 \times 8$ pixel tiles perform less than the larger ones.
- There is a much larger margin of error in the largest tile size 256 than for the other sizes, even though these experiments were repeated most. (See Tables A.1 and A.2 in the Appendix).

A possible explanation for the tile sizes $16 \times 16$, $32 \times 32$ and $64 \times 64$ performing best, with $32 \times 32$ as the maximum, could be that the texture that is present in the images is around that size. Creating tiles that are smaller than the distinctive texture would result in tiles that only contain a portion of a texture by which information would be lost. It is unlikely that all distinctive textures are exactly of the same size, which explains the small differences between the three different tile sizes.




*(a) Accuracy for diseases/skin/background dataset*




*(b) Accuracy for diseases/no disease dataset*

*Figure 4.4: Accuracy (in %) plotted against scaling factor for the diseases/skin/background dataset with 95% confidence levels on the standard error. Combinations of tile size and scaling where the combinations of tile size and scaling factor are coloured by their original tile size. For each original tile size, a trend line is plotted. The plot is made with a normal y axis and a logarithmic x axis, so each step down means a halving of each tile.*

## 4.3 Comparison of classifiers

Machine learning is a complex task and the best approach and best algorithm differs for each situation. This section contains a comparison between the different classification algorithms discussed in Section 2.7.

### 4.3.1 Experiment setup

For this experiment it isn't feasible to compare the classifiers for all combinations of tile size and scaling factor. Therefore comparison is done only for the best combination of scaling factor and tile size. According to the outcomes of Figure 4.4, the combination that gave the best overall accuracy in both the datasets was the tile size 32 and scaling factor one, closely followed by the tile size 64 and scaling 1. The latter was chosen for this experiment. Because the $64 \times 64$ tiles are larger, the overall dataset is four times smaller than that of $32 \times 32$ tiles. This makes it a lot easier to run each test multiple times to generate more accurate results. Though larger tiles could imply that smaller spots of disease are not classified as such in the tiling algorithm, Figure 4.1 indicates no such thing and even shows slightly higher percentages for the disease than at the $32 \times 32$ level.

Just comparing the overall accuracy, the percentage of correctly classified tiles, won't provide enough detail for comparing the classifiers, as preliminary tests showed that the overall accuracies were very close together. Except for the correctly predicted percentage of tiles, the classifiers also output a confusion matrix Witten et al. [2011]. In this matrix, each column represents the instances in a predicted class and each row represents the instances in the actual class. Each element thus represents the number of instances for which the actual class is the row and the predicted class is the column.

Reducing the level of detail in the confusion matrix leaves four categories for each class $i$:

**True positives** are instances of class $i$ that are correctly classified as class $i$;

**False positives** are instances of another class that are incorrectly classified as class $i$;

**False negative** are instances of class $i$ that are incorrectly classified as another class;

**True negatives** are instances which are not of class $i$ and indeed classified as another class that is not $i$, whether this prediction is correct or incorrect.

These values are directly proportional to the distribution of classes in the instance dataset and therefore not very suitable to compare the performance of the classifiers with our unbalanced datasets. With these four numbers for each of the classes, there are some other statistics that are more useful in comparing these values:

**Recall** or the true positive rate,

$$\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}, \tag{4.2}$$

is the fraction of instances of the class that are found or 'recalled' out of all instances of this class, giving information of how many tiles of a disease are found;

**Precision** ,

$$\frac{\text{true positives}}{\text{true positives} + \text{false positives}}, \tag{4.3}$$

is the fraction of correctly classified instances, the true positives, out of all instances that received the prediction of this class, providing a measure of how many tiles are incorrectly classified as a disease;

**F-measure** is the harmonic mean of recall and precision,

$$2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{4.4}$$

combining the two statistics into one.

### 4.3.2 Results

In Figures 4.5 and 4.6, the results of the experiment are displayed. Each of the classifiers was run thirty times to create numbers with a high confidence. The exact data is shown in Tables A.3 and A.4 in the appendix.

On inspecting the results, the following findings might be observed.

- Each algorithm scores very good on classifying background, both in recall and precision.
- The 1R algorithm clearly tends to generate a rule separating the largest classes, `normal skin` and `background` from the other classes as it only scores comparable results on those two classes.
- The NaiveBayes algorithm tends to classify many tiles as one of the diseases; scoring high on recall and low on precision for those classes. This also manifests itself in the low recall on normal skin. It would be interesting to investigate if the samples that are falsely classified as a disease, did come from an image containing that disease.
- The SMO algorithm has the opposite preference of classifying diseases very reticent; scoring low on recall and attaining the highest score on precision.
- The J48 algorithm shows steady performance on all classes on both precision and recall, however it seems to perform better at the diseases/skin/background dataset than at the diseases/no disease dataset.
- Most of the classifiers perform better on the normal skin and background samples than on the samples of the diseases. This could be because the normal skin and background tiles are in much greater number or because there is less variance between the tiles, as would be with a plain blue background. If this disparity still exists when resampling the dataset such that all classes are of the same size would this back up the notion that this inequality is caused by the skewed distribution.
- Each of the values are well above the baseline described in Table 4.2.

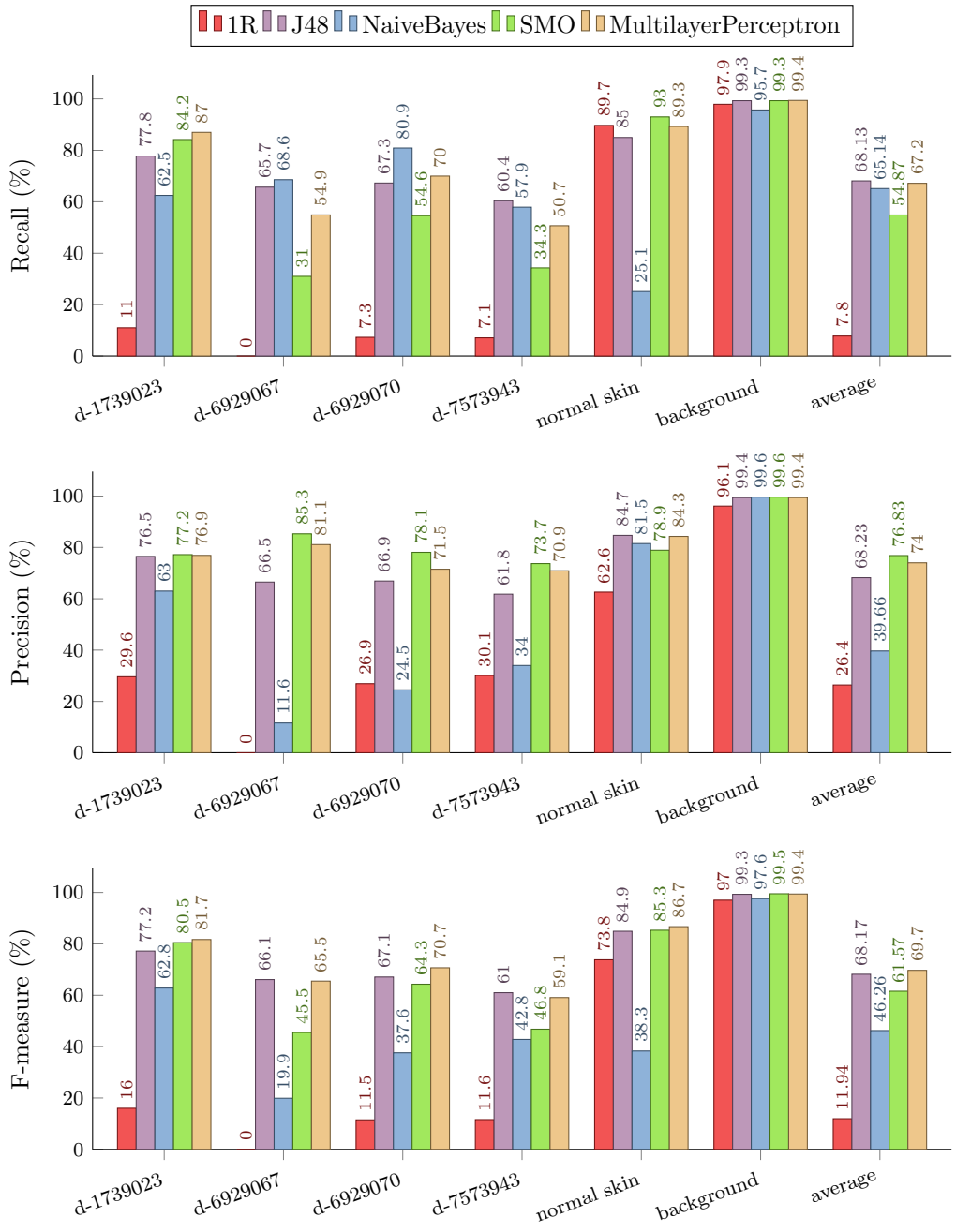



*Figure 4.5: Comparison of recall, precision and f-measure of different classifiers, trained on the disease/skin/background dataset for tilesize 32 and scale 1.0.*

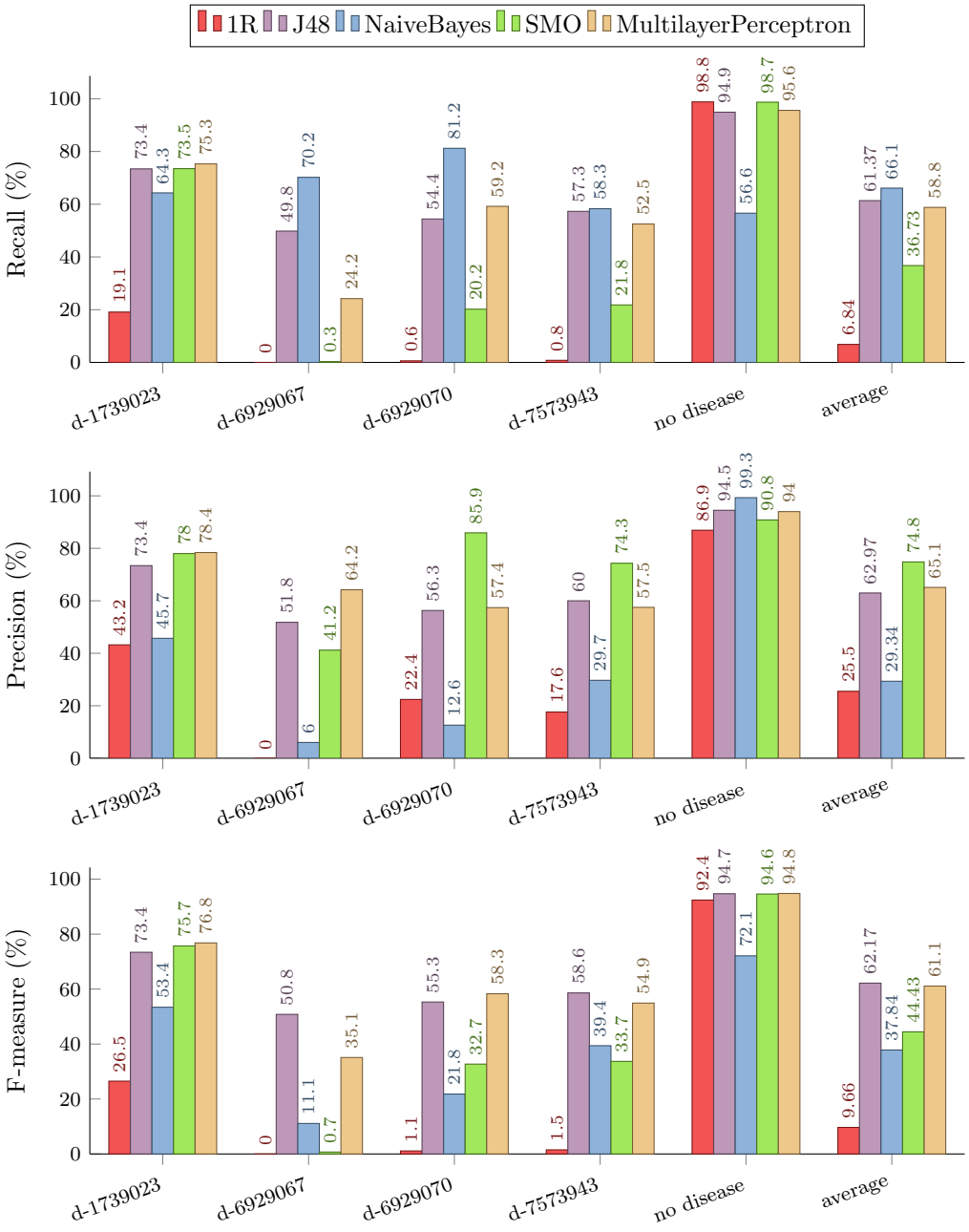



*Figure 4.6: Comparison of the recall, precision and f-measure of different classifiers, trained on the disease/no disease dataset for tilesize 32 and scale 1.0.*

## 4.4 Contribution of features

As discussed in Sections 2.4 and 2.5, there are two different groups of features: the colour features and the texture features. The composition of these features was done quite arbitrary, as in advance there was no clue about their usefulness for building a classifier. To give some insight in these features, further investigation was needed as the classifier algorithms do not give much information about this.

### 4.4.1 Experiment setup

A lot of things could be done to zoom in on the usefulness of the features. Weka provides several algorithms to rank features on individual basis, or to compose a subset of features which give the best performance. Repeatedly training the classifiers with the dataset with leaving out a feature each time would provide insight in which features do not contribute anything and thus could be left out. However, with a total of 93 features, this is not a easy task. Also, many features, like the ratios and the original colour values, are directly dependent on each other and leaving just one out would not mean that much difference.

Instead of manually trying to find the best and worst features, Weka is used to select and rank both features individually and create a subset of the best features. With just this subset a new dataset is created and the five classification algorithms are trained so their performance can be measured. Again, the dataset with tile size is 64 and a scaling factor of one is used. Because we are most interested in finding the best ranked features for discriminating between diseases, the dataset is reduced by removing the background and skin classes, such that only the four diseases remain.

To create a ranking for the features, two attribute evaluation algorithms, `InfoGainAttributeEval` and `OneRAttributeEval`, are used. The way `InfoGainAttributeEval` works is the same method as the J48 algorithm uses to calculate its first split: calculating the Info Gain on the data, using the algorithm described in Section 2.7.2. The `OneRAttributeEval` algorithm evaluates the performance of Holtes' 1R algorithm Holte [1993] as described by Holmes Holmes and Nevill-manning [1995]. The two algorithms are executed with tenfold cross validation to improve the reliability of the outcomes.

For the selection of a subset, the `CfsSubsetEval` algorithm is used. This algorithm, designed by M. A. Hall evaluates the worth of a subset of attributes by considering the individual predictive ability of each attribute along with the degree of redundancy between them Hall [1998]. The algorithm is executed with tenfold cross validation, to provide more accurate results.

The third comparison is between the colour features and the texture features. The tile size 64 and scaling factor one datasets will be split into two subsets with only colour features or only texture features and the results of the classifiers will be compared. To generate accurate results, each configuration is run twenty times with different seeds.

### 4.4.2 Results

#### Ranking of the features

The results of the two feature ranking algorithms, performed on the dataset containing only the four classes of diseases, is displayed in Table 4.4. Some of the interesting observations are listed here.

- The best ranked attribute, the average homogeneity of the value colour channel, `valueglcmhomogenityavg`, clearly outperforms the other attributes with a much higher InfoGain and also a higher score on the 1R separator.
- The features based on the colour channels hue and value, both originating from from the HSV colour space, score very good, representing twelve out of fifteen in the top list.
- The three ratios that were added all show up in the table, the *green/blue* ratio at the fifth place and the other two ratios *blue/red* and *green/red* both all the way on the bottom. Unfortunately, inspecting the raw training data revealed both values to be wrongly calculated, resulting in zero for an answer.
- The Haralick features homogeneity and contrast are present seven times in the top fifteen, making them very good features.
- With six out of the ten lowest ranked features, the saturation colour channel is not of much use.

#### Subset evaluation

The `CfsSubsetEval` algorithm was run with hundred fold cross validation, outputting the found created as a listing of the attributes with their percentage included in the subsets. Observing Table 4.5, a lot of the highest ranked attributes are represented in almost all tiles. The ones that are not included, like the fourth-ranked attribute `hue-glcm-homogenity-stddev`, likely show much similarities with previous included attributes. Adding such attribute to the subset would not improve its performance. As with the previous experiment, almost all attributes included are derived from the HSV colour space.

This experiment shows that not all attributes that are scored high by the ranking algorithms are included in the best subset of attributes. This means that there are many correlated attributes in the dataset. Removing some of the redundant attributes would not have a big impact on the performance, but would increase computational time and lower memory requirements when training classifiers.

Plotting all samples of diseases in the three dimensional space of three of the features of the subset, `value-glcm-homogenity-avg`, `value-glcm-energy-stddev` and `mean-color-hsv-hue`, a good degree of separation occurs, as can be seen in Figure 4.7.

#### Performance of only colour or texture features

The results of the third experiment, the comparison of the J48 classifier with only colour or texture features would seem to be predictable up front. Given that in the previous two experiments the texture features are more frequent than the colour features, it would be

| # | Ranking | Attribute | | Values | | |
|---|---|---|---|---|---|---|
| | | | | InfoGain | 1R | In subset |
| 1 | 1.0 | value-glcm-homogenity-avg | t | 0.85 | 70.8 | 100% |
| 2 | 2.6 | value-glcm-contrast-avg | t | 0.62 | 64.3 | 99% |
| 3 | 3.5 | hue-glcm-entropy-stddev | t | 0.62 | 63.7 | 100% |
| 4 | 3.9 | hue-glcm-homogenity-stddev | t | 0.59 | 64.4 | 0% |
| 5 | 4.0 | mean-color-ratio-green-blue | c | 0.62 | 62.1 | 100% |
| 6 | 6.0 | value-glcm-energy-stddev | t | 0.54 | 59.5 | 100% |
| 7 | 7.1 | value-glcm-contrast-stddev | t | 0.51 | 58.3 | 1% |
| 8 | 8.2 | mean-color-hsv-hue | c | 0.45 | 57.3 | 100% |
| 9 | 9.8 | gray-glcm-contrast-avg | t | 0.44 | 56.2 | 41% |
| 10 | 9.9 | hue-glcm-correlationinfo2-stddev | t | 0.44 | 55.8 | 100% |
| 11 | 11.9 | hue-glcm-contrast-stddev | t | 0.43 | 54.3 | 0% |
| 12 | 14.7 | value-glcm-clustertendency-stddev | t | 0.42 | 53.4 | 0% |
| 13 | 15.3 | hue-glcm-maxprob-stddev | t | 0.41 | 53.1 | 91% |
| 14 | 15.9 | hue-glcm-clustertendency-stddev | t | 0.39 | 54.0 | 0% |
| 15 | 16.3 | gray-glcm-homogenity-avg | t | 0.41 | 53.1 | 0% |
| | | ⋮ | | | | |
| 83 | 82.0 | saturation-glcm-maxprob-stddev | t | 0.15 | 41.5 | 0% |
| 84 | 83.1 | gray-glcm-correlation-avg | t | 0.12 | 42.0 | 0% |
| 85 | 85.4 | gray-glcm-correlationinfo1-stddev | t | 0.11 | 39.6 | 0% |
| 86 | 85.8 | saturation-glcm-info2-avg | t | 0.10 | 39.7 | 0% |
| 87 | 87.0 | saturation-glcm-info2-stddev | t | 0.09 | 38.4 | 0% |
| 88 | 89.6 | gray-glcm-entropy-stddev | t | 0.08 | 35.3 | 0% |
| 89 | 89.7 | saturation-glcm-info1-stddev | t | 0.07 | 36.0 | 0% |
| 90 | 90.7 | mean-color-ratio-green-red | c | 0.00 | 36.6 | 0% |
| 91 | 90.8 | mean-color-ratio-blue-red | c | 0.00 | 36.6 | 0% |
| 92 | 91.1 | saturation-glcm-correlation-avg | t | 0.03 | 35.1 | 0% |
| 93 | 91.3 | saturation-glcm-correlation-stddev | t | 0.03 | 35.0 | 0% |

*Table 4.4: Results of the attribute rankers* `InfogainAttributeEval` *and* `OneRAttributeEval`*. The table is ordered on average ranking and shows the top fifteen and ten lowest ranked attributes. The dataset used as input for the attribute rankers contains only samples of diseases: the skin and background classes were removed from the diseases/skin/background dataset. The column 'In subset' shows the percentage that the attribute was included in the subset generated by* `CfsSubsetEval`*. The column showing t or c indicates if the attribute is a colour feature or a texture feature.*

| Attribute | | Included in subset | Ranking |
|---|---|---|---|
| value-glcm-homogenity-avg | t | 100% | 1 |
| hue-glcm-entropy-stddev | t | 100% | 3 |
| mean-color-ratio-green-blue | c | 100% | 5 |
| value-glcm-energy-stddev | t | 100% | 6 |
| mean-color-hsv-hue | c | 100% | 8 |
| hue-glcm-correlationinfo2-stddev | t | 100% | 10 |
| saturation-glcm-homogenity-stddev | t | 100% | 16 |
| value-glcm-homogenity-stddev | t | 100% | 17 |
| mean-color-hsv-saturation | c | 100% | 25 |
| mean-color-rgb-blue | c | 100% | 30 |
| gray-glcm-homogenity-stddev | t | 100% | 59 |
| value-glcm-contrast-avg | t | 99% | 2 |
| hue-glcm-correlationinfo2-avg | t | 97% | 38 |
| hue-glcm-energy-stddev | t | 96% | 29 |
| hue-glcm-maximumprobability-stddev | t | 91% | 13 |
| saturation-glcm-homogenity-avg | t | 91% | 35 |
| gray-glcm-contrast-stddev | t | 60% | 22 |
| gray-glcm-contrast-avg | t | 41% | 9 |
| value-glcm-entropy-stddev | t | 11% | 57 |
| saturation-glcm-energy-stddev | t | 9% | 31 |
| value-glcm-correlation-stddev | t | 3% | 50 |
| value-glcm-contrast-stddev | t | 1% | 7 |
| value-glcm-correlation-avg | t | 1% | 55 |

*Table 4.5: Subset evaluation results with 100-fold cross validation, showing the percentage that each attribute was included in the subset together with the ranking of the attribute according to Table 4.4. Attributes that were zero times included are not displayed. Each subset contained 17 different attributes. The column showing t or c indicates if the attribute is a colour feature or a texture feature.*

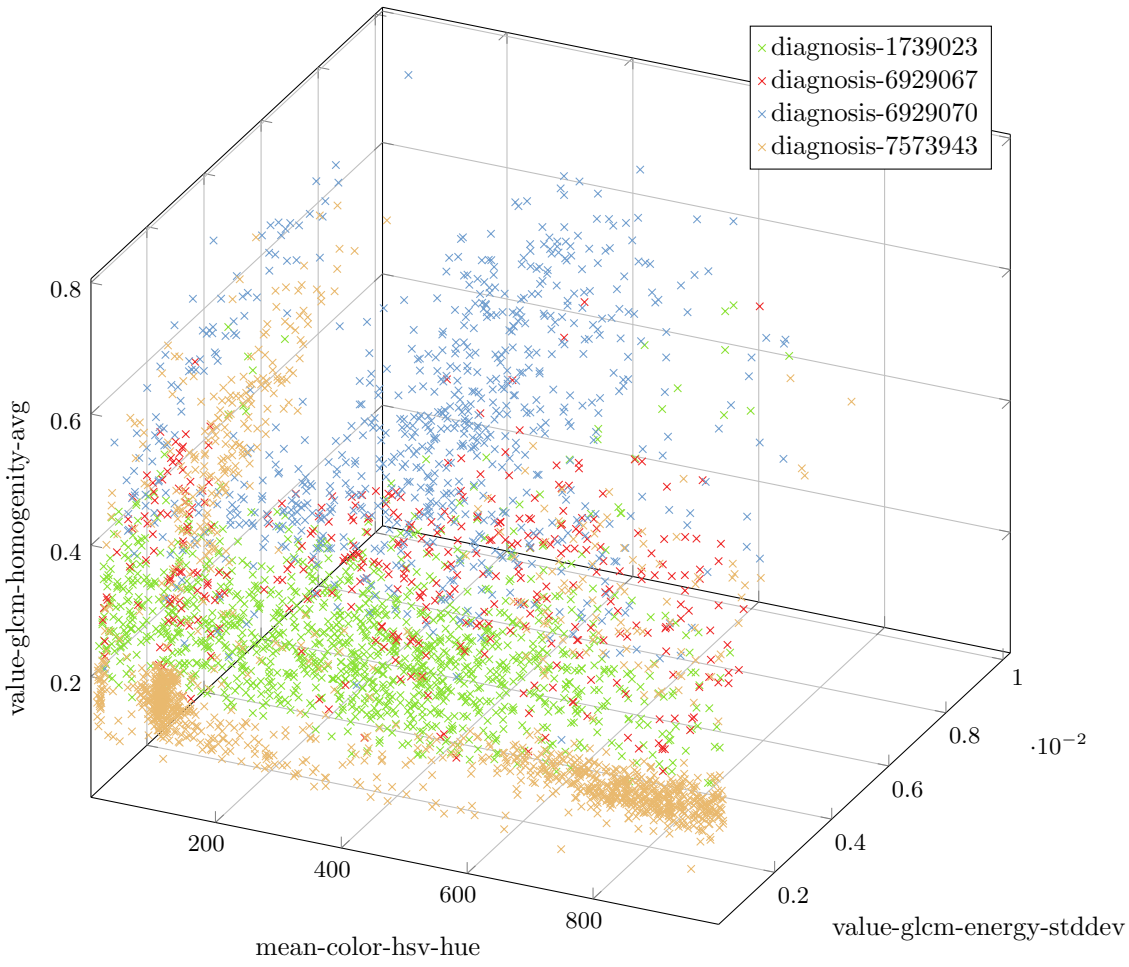



*Figure 4.7: Plot of the features* `value-glcm-homogenity-avg`*,* `value-glcm-energy-stddev` *and* `mean-color-hsv-hue`*. Except for the* `diagnosis-6929067`*, all samples are quite separable with only these three features.*

likely that the dataset of colour features would be outperformed by the dataset of texture features.

When plotting the results in Figure 4.8, this hypothesis is indeed proven correct, taking in account the 95% confidence levels that lay around 3.5, as can be seen in Table A.5 in the Appendix. Remarkable is that the texture-only dataset even performs slightly better (not significant) than the dataset containing all features. This indicates that classifiers might profit from datasets containing less features, a reason why further research should focus in experimenting with subsets of the dataset.

Another observation is that for the normal skin and background classes, both subsets perform as good as the original.

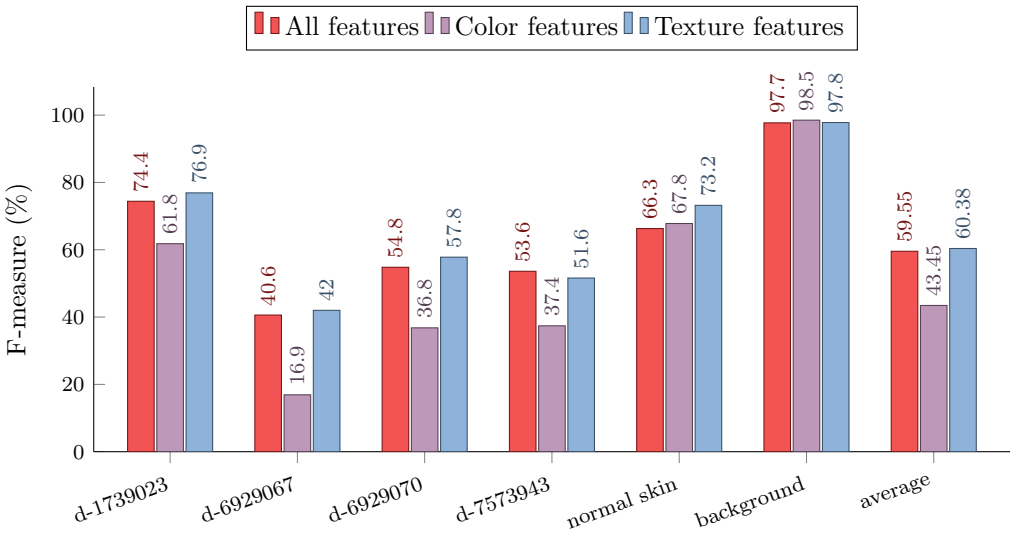



*Figure 4.8: Comparison between colour and texture features. Displayed is the f-measure averaged over the classifiers J48, NaiveBayes, SMO and MultilayerPerceptron.*

## 4.5 Classifying images

Whereas all experiments so far were based on the recognition of tiles, the final goal of this project is to be able use the classified tiles to say something about the images they are originating from. It is not possible to say that when a disease has a recall of 75%, 75% of the images with this disease are also found. There are several reasons why, but first the way images are classified must be explained. We know that all images are of affected skin, so the algorithm simply has to choose which disease it sees, not to able to discriminate between diseased and healthy skin. Therefore, the algorithm simply breaks the image into tiles and classifies them. The image is then classified according to the largest disease class that is present in the tiles.

Reasons why the recall on images won't automatically compare with the recall of the tiles would be:

- the distribution of the correctly classified tiles won't be evenly distributed amongst the images, therefore some images could contain only correctly classified tiles, while another image would not contain any correct tiles at all, still making up for a total of 75% recall;
- not all images contain the same amount of diseased tiles, in some image there is just less affected skin shown than in others, also skewing the distribution;
- although a majority of the diseased tiles would be classified correctly, when the *precision* of the classifier is not perfect, some false positives of other diseases will also show up in the images;
- even though all performance measures of the classifiers were calculated with tenfold cross validation, it can be expected that of all training images, ten percent of the tiles are left out of each fold. This means that slight differences between the images can also be 'learned' by the classifier, lowering performance on the images that were not annotated and therefore not used in the training.

### 4.5.1 Experiment setup

In order to classify the images, a new dataset was created, containing the features without classification for all 73 images in the data set. Time limited, the scope is again narrowed to only the tile sizes 64 and 32, both with scaling factor one. For each of the classifiers, the stored model is used to create predictions for the tiles. The tiles are then linked back to their originating image and totals for each class are calculated.

### 4.5.2 Results

The results are displayed in Figure 4.9. When comparing the recall with that of the tiles in Figure 4.5, the low recall for `diagnose-6929067` is visible in both histograms.

The differences between the 32 and the 64 tiles are inconclusive: for `diagnosis-1739023` the performance seems to increase with the smaller tile size, whereas for other diseases it seems to have less effect or even decreases. To get a clue why `diagnosis-6929067` is so badly recognised, a look at the original distribution of the images in Section 2.1 shows a possible reason: the number of annotated samples is only a third of all images while

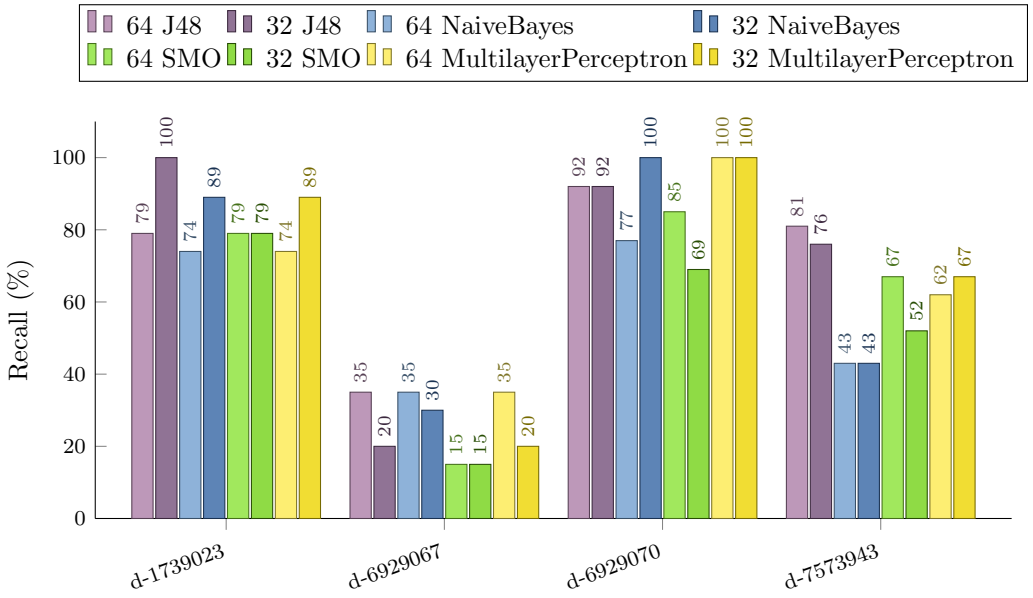



*Figure 4.9: Recall on classification of images*

`diagnosis-6929070` and `diagnosis-7573943` both have two third or more of the samples annotated, making it harder for the classifier to learn all characteristics and manifestations of the disease. However, `diagnosis-1739023` is recalled very good by all classifiers, while this class too has only a third of its images annotated.

A better explanation can be found when looking at the images themselves. `Diagnosis-6929067`, *Contact Dermatitis*, apparently has many manifestations, as can be seen in Figure 4.10. Together with a limited number of annotated image, not even containing all manifestations, it becomes virtually impossible to get an accurate prediction. A simple solution for this problem would be to add more annotated samples to the set.

Averaging the results, the best performing classifier is J48, with both tile sizes giving an equal average recall of 70%.

Figures 4.11 to 4.18 show for each disease two images that are classified. One sample of an image that was also used to train on and one image that was not included in the train set. All images were marked with the model created by the J48 algorithm, which was trained with $32 \times 32$ tiles and unscaled images. The following colours are used to mark the classes in these images:

- Pink: `diagnosis-1739023`
- Salmon: `diagnosis-6929067`
- Yellow: `diagnosis-6929070`
- Green: `diagnosis-7573943`
- Slightly gray overlay: `normal-skin`
- Slightly white overlay: `background`

*Figure 4.10: Close-ups of image diagnosed with* `diagnose-6929067`

*Figure 4.11: Image of* `diagnosis-1739023` *that is correctly classified. This image was used in training the classifier.*

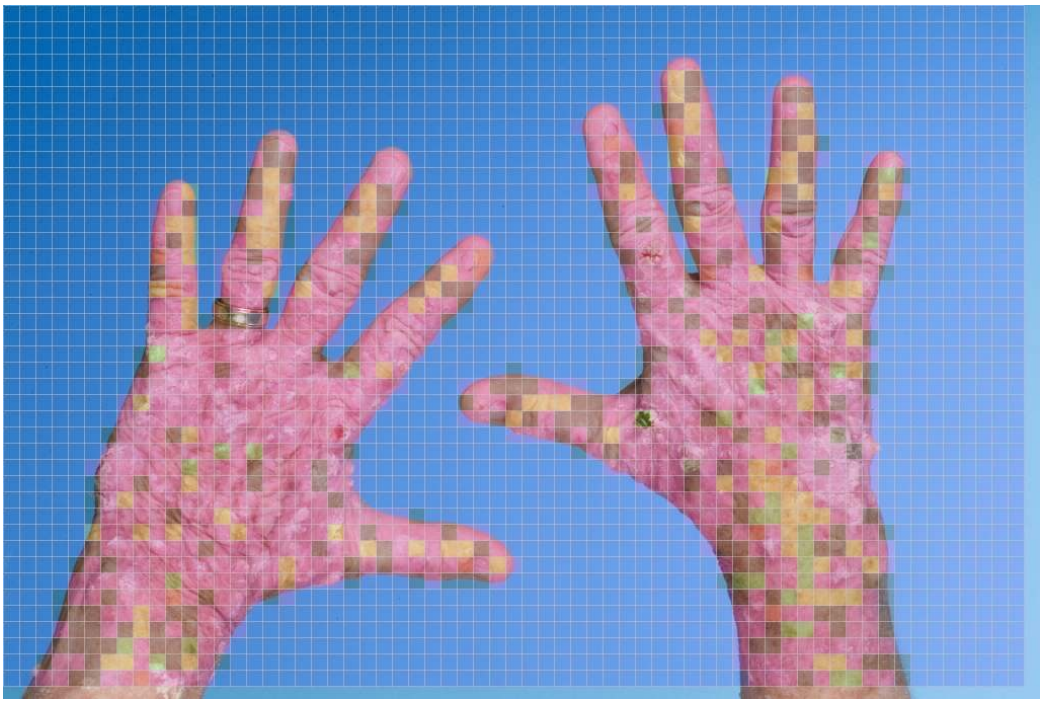


*Figure 4.12: Image of* `diagnosis-1739023` *that is correctly classified. This image was not used in training the classifier. Compared to Figure 4.11 this image contains more 'noise' from misclassified tiles, although a big majority is still correctly classified.*

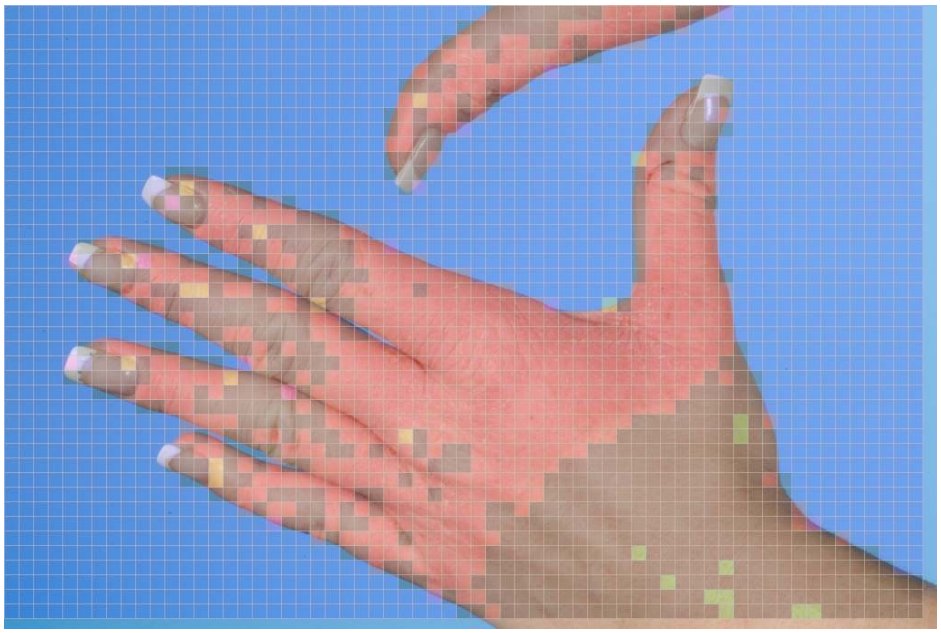


*Figure 4.13: Image of* `diagnosis-6929067` *that is correctly classified. This image was used in training the classifier.*

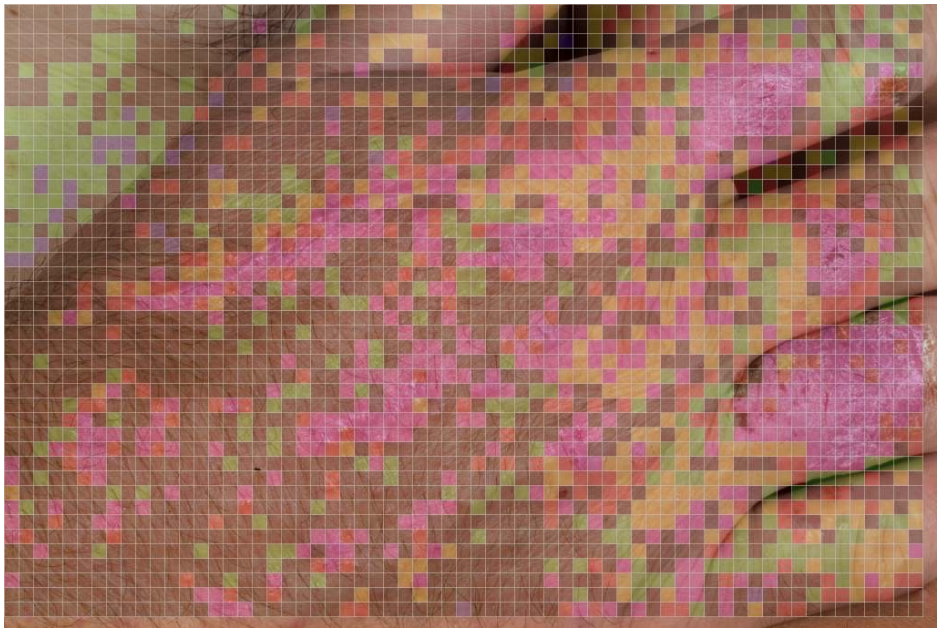


*Figure 4.14: Image of* `diagnosis-6929067` *that is incorrectly classified as* `diagnosis-1739023`*. This image was not used in training the classifier. The image clearly shows that the in the upper left corner, a lot of misclassified tiles exist. This is likely due to the fact that the texture on the torso is different than that of hands and was never seen by the classifier. All the rest of the image contains largely noisy misclassified tiles, although the fingers are as a whole misclassified rather than containing some single misclassified tiles.*

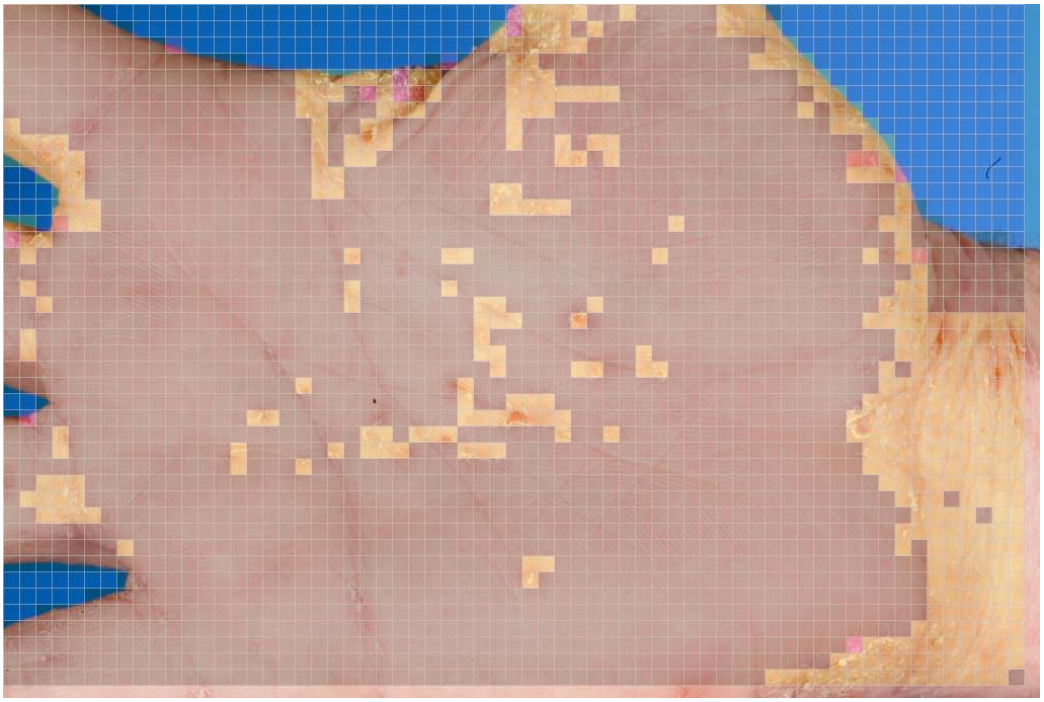


*Figure 4.15: Image of* `diagnosis-6929070` *that is correctly classified. This image was used in training the classifier.*

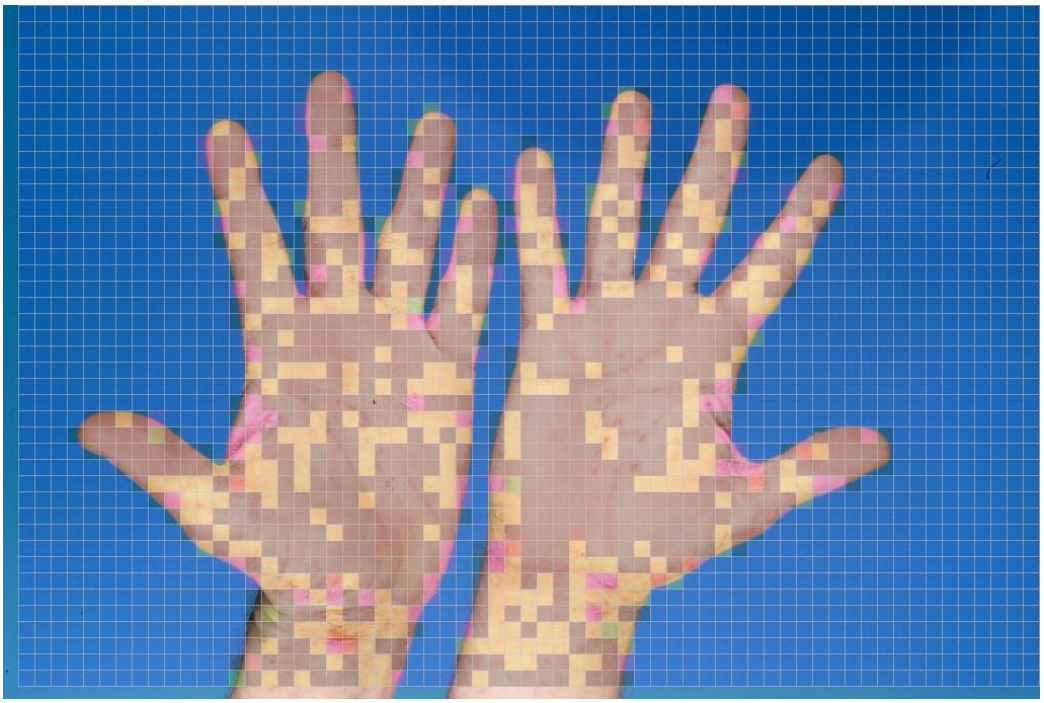


*Figure 4.16: Image of* `diagnosis-6929070` *that is correctly classified. This image was not used in training the classifier. As can be seen, the accuracy is lower and level of noise in the image is indeed higher than in the unseen Figure 4.15*

*Figure 4.17: Image of* `diagnosis-7573943` *that is correctly classified. The image was included in the set used to train the classifier. There are almost no tiles wrongly classified.*

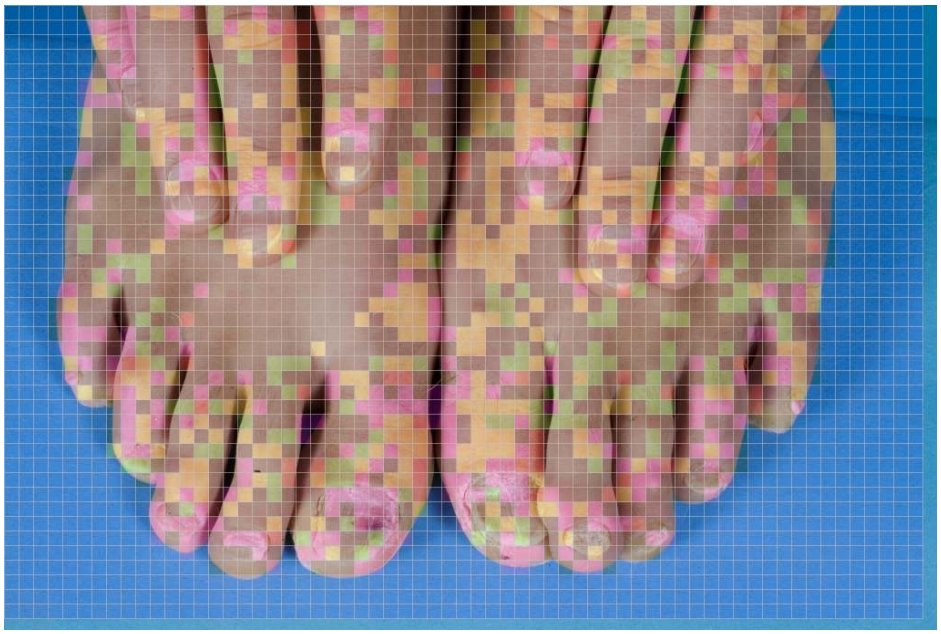


*Figure 4.18: Image of* `diagnosis-7573943` *that is incorrectly classified as* `diagnosis-7573943`*. This image was not used in training the classifier. Although this picture contains part of the fingers, the main topic are the foots of the patients. On both parts there are examples of random misclassifications and larger patches of skin that are misclassified.*

# Chapter 5

# Discussion and Conclusions

Although a great deal of discussion was already added to the Results, where it was more appropriate, a few more remarks and areas for improvement can be made on the process as a whole.

## 5.1 The road not taken

This project is a feasibility study and in the process many choices were made early on, limiting the possibilities further on. Looking back on the process, some minor and major points of improvements can be named.

### 5.1.1 The concept of tiles

While tiling and dividing the image in multiple tiles had proven a simple and effective approach on reducing the complexity of the image, a quite similar concept gives more possibilities. Instead of looking at the classification, one can look at classifying pixels, with a surrounding neighbourhood that is used to calculate features for that pixel. This way tiles can overlap and a higher 'resolution' on the image can be obtained without being limited to small tile sizes. It is also possible to reduce the number of pixels without enlarging the tile size.

Increasing the number of tiles that can be calculated from a single image not only gives more samples to train on. When enough overlapping tiles are generated, it is easy to toss out the tiles that do not contain a near majority of a single class. Also there is more chance for a small vesicle or pustule to end up entirely in a single tile, such that the texture features take into account the entire texture of the affected skin.

This approach would also have made it easier to compare different tile sizes, because they could all originate from the same image, without the need for scaling. If any further work would be done based on this project, it is recommended strongly to rewrite the feature creation algorithm to use this concept.

### 5.1.2 Resolution independence

In the current implementation, some of the example images contain close–ups of one fingernail while others contain an overview of the entire hand plus forearm. These images obviously result into different levels of detail when taking fixed size tiles. When adding multiple scalings of the same original tile size, a degree of resolution independence would be accomplished.

Just like the point-with-a-neighbourhood concept, using multiple scalings of the same image would not only increase the number of sample tiles to train with, it also makes it possible to match two images with different level of detail together.

Adding a degree of resolution independence would be a simple task: the current implementation of the tiling algorithm can be run multiple times and the results could be combined. A fast test showed promising improvements of the recall of classifiers.

## 5.2 Conclusions

This research project was a feasibility study to see if medical images that are taken with no such purpose in mind, can be used to classify dermatological diseases. As some promising results are visible in Section 4.5.2, it can safely be concluded that this is indeed feasible. Without having examined all options and certainly left room for various improvements and optimisations, a quite simple approach led to an overall recall of 70% on the images. Classification of the tiles scored much better than the baseline in Table 4.2.

A more detailed answer to the question if automated classification is feasible can be given by answering the more specific research questions listed in the introduction.

**Are all images in this database of use, or is there a clear distinction in characteristics?**

Although some diseases clearly result in a much more deviant texture than that of normal skin, results do not point out a single disease or image that is not usable. The badly recognised disease *Contact Dermatitis* has many manifestations in the images, presumably resolved by adding more images of this disease.

**What information is needed in addition to the images, for example skin colour, in order to create a classifier that can discriminate between diseases?**

First of all, some annotations were needed. For the disease affected areas, but also to mark the difference between healthy skin and background. Other data that was available was the sex and age of a patient. Adding those as features to the training samples would garble the results as the number of images is very limited and these features would push a classifier too much in a direction given an age or sex. When the size of the sample image set increases manyfold, adding those characteristics would help in ruling out conditions that occur only in childhood or old age.

**What kind of classifier can be used best for the purpose of discriminating between the four diseases.**

Figures 4.5 and 4.6 show that there is no such thing as the best classifier. "The best" is a trade-off between recall and precision. That being said, the J48 decision tree algorithm gives steady performance on both. As for classifying images, the J48 classifier scores highest on recall.

While it's not taken into any account so far, the execution time of a algorithm will be a major factor in the real world. Naive Bayes and 1R are the fastest, with Naive Bayes far more accurate. Although the recall and precision of the Naive Bayes algorithm is less than with the J48, SMO and Multilayer Perceptron classes, it comes along pretty good with the classification of images. The SMO and Multilayer Perceptron classifiers become really slow at smaller tile sizes and thus larger datasets. They would not scale well for very large datasets.

The question about the best classifier can't be answered unambiguously, for the current setup the J48 algorithm proves the best, but for larger datasets it is not possible to predict if computation time weighs up to the better performance, compared to the Naive Bayes class.

**Is it possible to locate the area of the affected skin or is it only possible to mark an entire image as containing a specific disease?**

Looking at the results in Figures 4.5 and 4.6 background tiles are recognised nearly perfect. The normal skin also scores quite good. When classifying an image, some noise might occur, but there will be a good estimate of the location of affected skin.

**How can the creation of new pictures be improved in order to increase their usefulness in automated classification?**

This research question deserves a section on its own to be answered, as this is part of a larger recommendation to the LUMC's dermatology department and their photographers.

**What kind of features are most useful for discriminating between diseases?**

The most useful features are displayed in Table 4.5. In a more general way, the results in Figure 4.8 point to the texture features as the most useful set of features. They are however also the most computational intensive features to calculate. This is why nominating features to be eliminated from the calculations is important when scaling up the amount of images in the training set.

At the start of the project, the tiling approach was not yet thought of. Therefore no research question was formulated about this. Looking at the results in Section 4.2 some important conclusions can be drawn. Adding more detail in the image by making the tiles smaller is not always going to provide better results and it certainly adds a lot of computation time. As is clearly visible in Figure 4.4, there is little to gain by decreasing the tile size lower than $64 \times 64$ pixels. Scaling down the image prior to the tiling is no good for the accuracy and should be avoided.

With all research questions being answered, some final remarks can be given as a conclusion. This bachelor project didn't provide a complete software solution for classifying dermatological images. It did however prove that it is possible to do so and hopefully leads the way to further research and a working implementation. While the scenario of an iPhone application that can be put to use in Africa or other parts of the world without adequate medical care is far away, a system like this could also be used to assign classifications to the current database of the LUMC.

## 5.3 Recommendations for the LUMC

This section contains some recommendations for the LUMC's department of dermatology and the photographers that are responsible for the image database. The most important recommendation is to collect more images of common skin diseases, although it serves no purpose for the original educational goal of the database. With more examples of the same disease, classifiers can be trained and evaluated much more accurate. How many images are needed can't be substantiated by this research, but a hundred for each disease would be a ballpark estimate. Another possibility is to store a photograph of each patient that comes in.

By storing information about the level of privacy for each image (heads, tattoos or private parts visible), obtaining 'safe' data is a much easier task. Adding the location on the body and diagnosis for each image is also essential. Without a proper classification the image is of little use for training or testing.

The third recommendation is already largely practiced: for easy recognising the skin from the surroundings, a plain background is recommended. Images should be taken with the affected part facing straight into the camera, so the texture is always the same. The rotation of the body part is of no concern, as the algorithm is fairly invariant to rotation.

If information is provided about the size of the object in an image, a better 'resolution independence' could be obtained. With information about the scale of the object, all images could be scaled into a uniform level of detail. This size information could be a manually added number, some markers that are present in the image or the camera's focal length and focus distance.

A topic that is not discussed so far, is that of complexion. Many diseases manifest themselves in different ways for different in different coloured skin. Knowing this information up front would support any classification.

When these five recommendations are followed up, a database with far greater value will arise, with the potential of being used for automated classification.

The topic for this research was originally shared with Lucas van der Meer van der Meer [2012]. This research focussed solely on the classification using the current database, his complementary project provides a general overview of how the dermatology in general can benefit from information technology. Herein a more profound advice for the department is included, along with some methods of computer supported annotation of the images.

## 5.4 Further research

Some topics worth investigating are already stated in Section 5.1: changing the way that tiling is handled and achieving a level of resolution independence by scaling the tiles in different ways. Other than things that could have been done different, some other interesting ideas came forward in this research.

The most obvious topic is a more intense study on classifying skin diseases, starting where this research ended. A larger dataset could provide better accuracy as well as more challenges concerning the computational intensive tasks of calculating the features and training the classifier.

Another idea that would be interesting is the application of the setup of this research to create predictions of new images and feed them back to the user which can check the results. By accepting or declining the proposed classification, additional images that can be used for training can be created, as well as negative examples of images *not* containing a specific disease.

# Bibliography


A. Gola Isasi, B. Garca Zapirain, and A. Mndez Zorrilla. Melanomas non-invasive diagnosis application based on the abcd rule and pattern recognition image processing algorithms. *Computers in Biology and Medicine*, 41(9):742 – 755, 2011. ISSN 0010-4825. doi: 10.1016/j.compbiomed.2011.06.010. URL http://www.sciencedirect.com/science/article/pii/S0010482511001302.

Imaging Research Group of LIACS. Tdr-3dbase: software for 3d reconstruction. URL http://bio-imaging.liacs.nl/tdr3dbase.html.

Opencv (open source computer vision). URL http://opencv.willowgarage.com/wiki/.

Weka 3: Data mining software in java. URL http://www.cs.waikato.ac.nz/ml/weka/.

Benjamin D. Zarit, Boaz J. Super, and Francis K. H. Quek. Comparison of five color models in skin pixel classification. In *In ICCV99 Intl Workshop on*, pages 58–63, 1999.

Its'hak Dinstein Robert M Haralick, K Shanmugam. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics SMC-3*, SMC-3(6):610–621, 1973. URL http://www.makseq.com/materials/lib/Articles-Books/Filters/Texture/Co-occurence/haralick73.pdf.

J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Texts in Applied Mathematics. Springer, 2002. ISBN 9780387954523. URL http://books.google.nl/books?id=1oDXWLb9qEkC.

Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–90, 1993. ISSN 0885-6125. URL http://dx.doi.org/10.1023/A:1022631118932. 10.1023/A:1022631118932.

J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

Russell en Norvig. *Artificial intelligence, A modern approach*. Prentice Hall, 2nd edition, 2003.

Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Mach. Learn.*, 29(2-3):103–130, November 1997. ISSN 0885-6125. doi: 10.1023/A:1007413511361. URL http://dx.doi.org/10.1023/A:1007413511361.

James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. 1995. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.99.6787.

Shi jin Wang, Avin D. Mathew, Yan Chen, Li feng Xi, Lin Ma, and Jay Lee. Empirical analysis of support vector machine ensemble classifiers. *Expert Systems with Applications*, 36(3 Pt2):6466–6476, April 2009. URL http://eprints.qut.edu.au/17833/.

Tristan Fletcher. Support vector machines explained. 2009. URL http://www.tristanfletcher.co.uk/SVM%20Explained.pdf.

Class smo, a. URL http://weka.sourceforge.net/doc/weka/classifiers/functions/SMO.html.

John C. Platt. Advances in kernel methods. chapter Fast training of support vector machines using sequential minimal optimization, pages 185–208. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-19416-3. URL http://fbim.fh-regensburg.de/~saj39122/Diplomarbeiten/Miklos/Papers/PLAT%20SMO-book.pdf.

Diagram of a multi-layer feedforward artificial neural network. URL http://commons.wikimedia.org/wiki/File:MultiLayerNeuralNetwork.png.

Class j48, b. URL http://weka.sourceforge.net/doc/weka/classifiers/trees/J48.html.

Class naivebayes, c. URL http://weka.sourceforge.net/doc/weka/classifiers/bayes/NaiveBayes.html.

Class multilayerperceptron, d. URL http://weka.sourceforge.net/doc/weka/classifiers/functions/MultilayerPerceptron.html.

I.H. Witten, E. Frank, and M.A. Hall. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2011.

G. Holmes and C. G. Nevill-manning. Feature selection via the discovery of simple classification rules. In *In Proceedings of the International Symposium on Intelligent Data Analysis*, 1995. URL http://www.cs.waikato.ac.nz/pubs/wp/1995/uow-cs-wp-1995-10.pdf.

Mark A. Hall. Correlation-based feature selection for machine learning, 1998. URL http://www.cs.waikato.ac.nz/~ml/publications/1999/99MH-Thesis.pdf.

L. van der Meer. Health Informatics: A vision on the future use of informatics techniques at the dermatology department of a large Dutch academic hospital. 8 2012.

# Appendix A

# Detailled tables

| Tile Size | Scale | Count | Average | Variation | StdDev | StdErr | 95% conf |
|---|---|---|---|---|---|---|---|
| 256 | 1 | 20 | 77.66 | 0.89 | 0.94 | 0.21 | 0.41 |
| 128 | 1 | 15 | 82.74 | 0.29 | 0.54 | 0.14 | 0.27 |
| 128 | 1/2 | 20 | 76.88 | 1.00 | 1.00 | 0.22 | 0.44 |
| 64 | 1 | 30 | 85.58 | 0.04 | 0.19 | 0.04 | 0.07 |
| 64 | 1/2 | 15 | 81.55 | 0.24 | 0.49 | 0.13 | 0.25 |
| 64 | 1/4 | 20 | 74.91 | 1.36 | 1.17 | 0.26 | 0.51 |
| 32 | 1 | 3 | 86.18 | 0.04 | 0.19 | 0.11 | 0.21 |
| 32 | 1/2 | 140 | 84.59 | 0.05 | 0.23 | 0.02 | 0.04 |
| 32 | 1/4 | 15 | 80.97 | 0.18 | 0.43 | 0.11 | 0.22 |
| 32 | 1/8 | 20 | 74.42 | 1.78 | 1.34 | 0.30 | 0.59 |
| 16 | 1 | 3 | 85.88 | 0.00 | 0.02 | 0.01 | 0.02 |
| 16 | 1/2 | 3 | 85.37 | 0.00 | 0.07 | 0.04 | 0.07 |
| 16 | 1/4 | 10 | 83.73 | 0.06 | 0.25 | 0.08 | 0.15 |
| 16 | 1/8 | 14 | 80.18 | 0.33 | 0.58 | 0.15 | 0.30 |
| 16 | 1/16 | 20 | 73.73 | 0.88 | 0.94 | 0.21 | 0.41 |
| 8 | 1 | 2 | 84.72 | 0.00 | 0.02 | 0.02 | 0.03 |
| 8 | 1/2 | 3 | 84.70 | 0.00 | 0.05 | 0.03 | 0.05 |
| 8 | 1/4 | 3 | 83.70 | 0.02 | 0.13 | 0.07 | 0.14 |
| 8 | 1/8 | 10 | 81.61 | 0.04 | 0.19 | 0.06 | 0.12 |
| 8 | 1/16 | 15 | 76.91 | 0.23 | 0.48 | 0.12 | 0.24 |
| 8 | 1/32 | 20 | 70.17 | 1.25 | 1.12 | 0.25 | 0.49 |

*Table A.1: Accuracy (% correctly classified tiles) of the J48 algorithm, trained with the diseases/normal skin/background dataset.*

| Tile Size | Scale | Count | Accuracy | Variation | StdDev | StdErr | 95% conf |
|---|---|---|---|---|---|---|---|
| 256 | 1 | 20 | 84.42 | 0.25 | 0.50 | 0.11 | 0.22 |
| 128 | 1 | 15 | 87.87 | 0.07 | 0.27 | 0.07 | 0.14 |
| 128 | 1/2 | 20 | 84.51 | 0.51 | 0.72 | 0.16 | 0.31 |
| 64 | 1 | 29 | 89.77 | 0.02 | 0.13 | 0.03 | 0.05 |
| 64 | 1/2 | 15 | 87.30 | 0.09 | 0.31 | 0.08 | 0.16 |
| 64 | 1/4 | 20 | 84.65 | 0.52 | 0.72 | 0.16 | 0.32 |
| 32 | 1 | 5 | 90.13 | 0.00 | 0.07 | 0.03 | 0.06 |
| 32 | 1/2 | 10 | 89.16 | 0.04 | 0.19 | 0.06 | 0.12 |
| 32 | 1/4 | 15 | 87.48 | 0.07 | 0.26 | 0.07 | 0.13 |
| 32 | 1/8 | 20 | 84.25 | 0.37 | 0.61 | 0.14 | 0.27 |
| 16 | 1 | 3 | 89.65 | 0.00 | 0.01 | 0.01 | 0.01 |
| 16 | 1/2 | 5 | 89.56 | 0.00 | 0.06 | 0.03 | 0.05 |
| 16 | 1/4 | 10 | 88.62 | 0.03 | 0.17 | 0.05 | 0.11 |
| 16 | 1/8 | 14 | 86.59 | 0.12 | 0.34 | 0.09 | 0.18 |
| 16 | 1/16 | 20 | 83.59 | 0.38 | 0.61 | 0.14 | 0.27 |
| 8 | 1 | 2 | 88.81 | 0.00 | 0.04 | 0.03 | 0.05 |
| 8 | 1/2 | 3 | 88.84 | 0.00 | 0.05 | 0.03 | 0.05 |
| 8 | 1/4 | 5 | 88.45 | 0.00 | 0.03 | 0.01 | 0.02 |
| 8 | 1/8 | 10 | 87.08 | 0.01 | 0.12 | 0.04 | 0.07 |
| 8 | 1/16 | 15 | 84.80 | 0.09 | 0.30 | 0.08 | 0.15 |
| 8 | 1/32 | 20 | 81.67 | 0.49 | 0.70 | 0.16 | 0.31 |

*Table A.2: Accuracy (% correctly classified tiles) of the J48 algorithm, trained with the diseases/no-diseases dataset.*

| | | Precision | | | | Recall | | | | F-measure | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Classifier | Class | Avg | StdDev | StdErr | 95% | Avg | StdDev | StdErr | 95% | Avg | StdDev | StdErr | 95% |
| 1R | d-1739023 | 25.5 | 1.5 | 0.4 | 0.8 | 8.3 | 0.6 | 0.2 | 0.3 | 12.6 | 0.9 | 0.2 | 0.4 |
| | d-6929067 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | d-6929070 | 32.0 | 1.8 | 0.5 | 0.9 | 13.4 | 1.2 | 0.3 | 0.6 | 18.9 | 1.4 | 0.4 | 0.7 |
| | d-7573943 | 29.7 | 1.1 | 0.3 | 0.6 | 11.1 | 0.7 | 0.2 | 0.3 | 16.2 | 0.8 | 0.2 | 0.4 |
| | normal-skin | 61.2 | 0.2 | 0.0 | 0.1 | 86.3 | 0.5 | 0.1 | 0.2 | 71.6 | 0.3 | 0.1 | 0.1 |
| | background | 95.9 | 0.1 | 0.0 | 0.1 | 97.5 | 0.1 | 0.0 | 0.0 | 96.7 | 0.1 | 0.0 | 0.0 |
| J48 | d-1739023 | 79.6 | 0.7 | 0.2 | 0.4 | 80.0 | 1.1 | 0.3 | 0.7 | 79.8 | 0.8 | 0.2 | 0.5 |
| | d-6929067 | 66.6 | 1.1 | 0.3 | 0.7 | 63.5 | 1.8 | 0.6 | 1.1 | 65.0 | 0.7 | 0.2 | 0.4 |
| | d-6929070 | 67.2 | 1.5 | 0.5 | 0.9 | 66.3 | 1.3 | 0.4 | 0.8 | 66.8 | 1.1 | 0.4 | 0.7 |
| | d-7573943 | 61.8 | 0.7 | 0.2 | 0.5 | 61.3 | 0.4 | 0.1 | 0.2 | 61.5 | 0.5 | 0.2 | 0.3 |
| | normal-skin | 83.9 | 0.1 | 0.0 | 0.1 | 84.5 | 0.4 | 0.1 | 0.3 | 84.2 | 0.2 | 0.1 | 0.1 |
| | background | 99.1 | 0.1 | 0.0 | 0.0 | 98.8 | 0.1 | 0.0 | 0.1 | 98.9 | 0.1 | 0.0 | 0.0 |
| NaiveBayes | d-1739023 | 61.5 | 0.2 | 0.1 | 0.1 | 68.6 | 0.3 | 0.1 | 0.1 | 64.9 | 0.2 | 0.1 | 0.1 |
| | d-6929067 | 14.2 | 0.1 | 0.0 | 0.1 | 67.4 | 0.5 | 0.1 | 0.3 | 23.4 | 0.2 | 0.1 | 0.1 |
| | d-6929070 | 27.0 | 0.1 | 0.0 | 0.0 | 79.4 | 0.2 | 0.1 | 0.1 | 40.3 | 0.1 | 0.0 | 0.1 |
| | d-7573943 | 37.6 | 0.1 | 0.0 | 0.1 | 64.8 | 0.2 | 0.1 | 0.1 | 47.6 | 0.1 | 0.0 | 0.1 |
| | normal-skin | 76.7 | 0.2 | 0.1 | 0.1 | 28.7 | 0.1 | 0.0 | 0.1 | 41.8 | 0.1 | 0.0 | 0.1 |
| | background | 98.8 | 0.0 | 0.0 | 0.0 | 93.3 | 0.0 | 0.0 | 0.0 | 96.0 | 0.0 | 0.0 | 0.0 |
| SMO | d-1739023 | 79.2 | 0.2 | 0.1 | 0.1 | 86.1 | 0.2 | 0.1 | 0.2 | 82.5 | 0.1 | 0.1 | 0.1 |
| | d-6929067 | 96.4 | 1.2 | 0.4 | 0.9 | 17.3 | 0.8 | 0.3 | 0.6 | 29.4 | 1.2 | 0.5 | 0.9 |
| | d-6929070 | 78.5 | 0.4 | 0.2 | 0.3 | 51.1 | 0.3 | 0.1 | 0.2 | 61.9 | 0.3 | 0.1 | 0.2 |
| | d-7573943 | 71.9 | 0.7 | 0.3 | 0.5 | 39.2 | 0.4 | 0.2 | 0.3 | 50.7 | 0.5 | 0.2 | 0.4 |
| | normal-skin | 78.1 | 0.1 | 0.0 | 0.1 | 92.6 | 0.1 | 0.0 | 0.1 | 84.7 | 0.1 | 0.0 | 0.1 |
| | background | 99.2 | 0.0 | 0.0 | 0.0 | 98.9 | 0.0 | 0.0 | 0.0 | 99.1 | 0.0 | 0.0 | 0.0 |
| MP | d-1739023 | 82.1 | 0.8 | 0.5 | 0.9 | 88.5 | 1.1 | 0.6 | 1.3 | 85.2 | 0.1 | 0.1 | 0.1 |
| | d-6929067 | 77.3 | 1.5 | 0.9 | 1.7 | 67.1 | 1.9 | 1.1 | 2.2 | 71.8 | 0.7 | 0.4 | 0.7 |
| | d-6929070 | 73.3 | 3.1 | 1.8 | 3.5 | 68.6 | 3.3 | 1.9 | 3.8 | 70.8 | 0.4 | 0.2 | 0.5 |
| | d-7573943 | 67.6 | 2.7 | 1.6 | 3.1 | 60.4 | 1.0 | 0.6 | 1.2 | 63.8 | 0.6 | 0.4 | 0.7 |
| | normal-skin | 85.5 | 0.5 | 0.3 | 0.6 | 87.8 | 1.2 | 0.7 | 1.3 | 86.6 | 0.3 | 0.2 | 0.4 |
| | background | 99.3 | 0.1 | 0.0 | 0.1 | 99.0 | 0.0 | 0.0 | 0.0 | 99.2 | 0.0 | 0.0 | 0.0 |

*Table A.3: The performance of classifiers for the diseases/skin/background dataset, averaged over 30 runs.*

| | | Precision | | | | Recall | | | | F-measure | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Classifier | Class | Avg | StdDev | StdErr | 95% | Avg | StdDev | StdErr | 95% | Avg | StdDev | StdErr | 95% |
| 1R | d-1739023 | 26.9 | 3.0 | 0.8 | 1.5 | 5.5 | 0.8 | 0.2 | 0.4 | 9.2 | 1.3 | 0.3 | 0.7 |
| | d-6929067 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | d-6929070 | 1.2 | 3.4 | 0.9 | 1.7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 |
| | d-7573943 | 32.1 | 1.8 | 0.5 | 0.9 | 11.1 | 0.9 | 0.2 | 0.4 | 16.5 | 1.1 | 0.3 | 0.6 |
| | no-disease | 85.7 | 0.0 | 0.0 | 0.0 | 97.7 | 0.1 | 0.0 | 0.1 | 91.3 | 0.1 | 0.0 | 0.0 |
| J48 | d-1739023 | 75.5 | 0.6 | 0.2 | 0.4 | 75.4 | 0.9 | 0.3 | 0.6 | 75.5 | 0.5 | 0.2 | 0.3 |
| | d-6929067 | 49.9 | 1.7 | 0.5 | 1.0 | 47.9 | 1.6 | 0.5 | 1.0 | 48.9 | 1.4 | 0.4 | 0.9 |
| | d-6929070 | 55.8 | 0.6 | 0.2 | 0.4 | 53.4 | 1.6 | 0.5 | 1.0 | 54.6 | 0.9 | 0.3 | 0.6 |
| | d-7573943 | 59.9 | 1.0 | 0.3 | 0.6 | 59.0 | 1.5 | 0.5 | 0.9 | 59.4 | 1.1 | 0.3 | 0.7 |
| | no-disease | 94.3 | 0.1 | 0.0 | 0.1 | 94.6 | 0.1 | 0.0 | 0.1 | 94.5 | 0.1 | 0.0 | 0.0 |
| NaiveBayes | d-1739023 | 45.4 | 0.2 | 0.1 | 0.1 | 69.9 | 0.3 | 0.1 | 0.1 | 55.1 | 0.2 | 0.1 | 0.1 |
| | d-6929067 | 6.8 | 0.1 | 0.0 | 0.0 | 69.4 | 0.6 | 0.2 | 0.3 | 12.4 | 0.1 | 0.0 | 0.1 |
| | d-6929070 | 13.9 | 0.1 | 0.0 | 0.0 | 79.7 | 0.3 | 0.1 | 0.2 | 23.7 | 0.1 | 0.0 | 0.1 |
| | d-7573943 | 30.6 | 0.1 | 0.0 | 0.0 | 66.8 | 0.2 | 0.0 | 0.1 | 42.0 | 0.1 | 0.0 | 0.0 |
| | no-disease | 99.1 | 0.0 | 0.0 | 0.0 | 56.3 | 0.0 | 0.0 | 0.0 | 71.8 | 0.0 | 0.0 | 0.0 |
| SMO | d-1739023 | 79.0 | 0.3 | 0.1 | 0.2 | 77.1 | 0.2 | 0.1 | 0.2 | 78.0 | 0.2 | 0.1 | 0.2 |
| | d-6929067 | 92.9 | 18.9 | 7.1 | 14.0 | 0.3 | 0.1 | 0.0 | 0.1 | 0.7 | 0.3 | 0.1 | 0.2 |
| | d-6929070 | 82.6 | 2.6 | 1.0 | 1.9 | 0.6 | 0.1 | 0.0 | 0.1 | 1.3 | 0.2 | 0.1 | 0.2 |
| | d-7573943 | 73.1 | 0.4 | 0.2 | 0.3 | 29.2 | 0.4 | 0.1 | 0.3 | 41.8 | 0.4 | 0.2 | 0.3 |
| | no-disease | 90.3 | 0.0 | 0.0 | 0.0 | 98.6 | 0.0 | 0.0 | 0.0 | 94.2 | 0.0 | 0.0 | 0.0 |
| MP | d-1739023 | 78.7 | 1.1 | 0.7 | 1.5 | 81.8 | 0.8 | 0.6 | 1.2 | 80.2 | 1.0 | 0.7 | 1.3 |
| | d-6929067 | 66.1 | 2.5 | 1.8 | 3.5 | 41.7 | 2.7 | 1.9 | 3.7 | 51.0 | 1.2 | 0.9 | 1.7 |
| | d-6929070 | 64.5 | 6.1 | 4.3 | 8.5 | 54.1 | 3.9 | 2.8 | 5.4 | 58.6 | 0.2 | 0.2 | 0.3 |
| | d-7573943 | 69.6 | 1.4 | 1.0 | 2.0 | 55.3 | 3.4 | 2.4 | 4.7 | 61.6 | 2.7 | 1.9 | 3.7 |
| | no-disease | 94.2 | 0.4 | 0.3 | 0.5 | 96.4 | 0.4 | 0.3 | 0.5 | 95.3 | 0.0 | 0.0 | 0.0 |

*Table A.4: The performance of classifiers for the diseases/no disease dataset, averaged over 30 runs.*

| | All features | | | Colour features | | | Texture features | | |
|---|---|---|---|---|---|---|---|---|---|
| Class | F-measure | StdError | 95% conf | F-measure | StdError | 95% conf | F-measure | StdError | 95% conf |
| d-1739023 | 74.4 | 1.4 | 2.8 | 61.8 | 1.1 | 2.1 | 76.9 | 0.8 | 1.5 |
| d-6929067 | 40.6 | 3.5 | 6.8 | 16.9 | 2.0 | 3.9 | 42.0 | 2.8 | 5.5 |
| d-6929070 | 54.8 | 2.2 | 4.3 | 36.8 | 2.9 | 5.6 | 57.8 | 1.2 | 2.4 |
| d-7573943 | 53.6 | 1.1 | 2.2 | 37.4 | 2.6 | 5.0 | 51.6 | 0.9 | 1.7 |
| normal-skin | 66.3 | 3.6 | 7.1 | 67.8 | 2.5 | 4.9 | 73.2 | 2.1 | 4.2 |
| background | 97.7 | 0.3 | 0.5 | 98.5 | 0.1 | 0.1 | 97.8 | 0.2 | 0.4 |

*Table A.5: Comparison between colour and texture features. Displayed is the f-measure averaged over the classifiers J48, NaiveBayes, SMO and MultilayerPerceptron together with the standard error and 95% confidence interval.*

# List of Figures

# List of Tables