Vrije Universiteit Amsterdam

Research paper

Predicting the most common autoimmune diseases

An experimental comparison of methods for multi-label learning using three well known classification algorithms

Author:
Laurèl Joannes

Supervisor:
Mark Hoogendoorn

Predicting the most common autoimmune diseases

Laurèl Joannes

Research Paper

Vrije Universiteit Amsterdam
Faculty of Sciences
The Netherlands

November 11, 2018

Abstract

Autoimmune diseases are in the top ten leading causes of all deaths among U.S. women under 65 years[39] and is the fourth-largest cause of disability for these women.[25] Patients spent over 4.5 years with 5 different doctors before getting the correct diagnosis[2].
This research gives an experimental comparison of methods for multi-label learners is done based on three well known classification algorithms. Learners are made in order to predict the most common autoimmune diseases based on patients' symptoms and conditions. The aim is to assist medical professionals in diagnosing the patient by analyzing his symptoms, conditions and health record and deduce the time until diagnosis.
Two methods are used to handle patients having multiple autoimmune diseases. The first is the Problem Transformation method, where multi-label classification problems are converted into single-label problems. This is split into the Label Powerset method and the Binary Relevance method. In the Lapel Powerset method a new class is defined for each unique combination of labels attested in the training set. The Binary Relevance method on the other hand a separate binary problem is created, one for every possible label. The second is the Algorithm Adaptation method where conventional classifiers are implemented in a way that allows for multi-label classification.
Based on these methods, three well known algorithms have been used namely, the k-Nearest Neighbors, the Random Forest and the Support Vector Machine. For every combination of algorithm and method models are developed and optimized for the F-measure of this model, while tuning the parameters. The two best parameter settings are chosen, which results in 18 developed models, that are evaluated and compared. Finally, a conclusion can be drawn that an ensemble chain of Binary Relevance methods, overall, works best, especially in combination with the Support Vector Machine. This method uses a Radial kernel, a γ-parameter of 2^{-10}, a cost parameter of 10 and a threshold of 0.3. The F-measure has a 95% confidence interval of [14.83, 21.03] which is not high. But as a result, in almost 50% of the cases at least one autoimmune disease of a patient can be predicted correctly, and the average time before receiving a diagnosis can be deduced significantly.

Table of Contents

1 Introduction

Informatics and machine learning is widely applied in health care nowadays, sensors are available that constantly monitor blood glucose levels, analyze readings, recognize patterns in data, as well as in images or text, testing in silico is made possible, using simulations to avoid in vitro testing.[23][41]
The progress in biotechnology and health science have led to the enormous production of electronic health records and data. As a result, hundreds of petabytes of health records available in digital dossiers.[41]

An autoimmune disease (aw-to-ĭ-mūn dĭ-zēz) is "any disease characterized by tissue injury caused by an apparent immunologic reaction of the host with his own tissues; distinguished from autoimmune response, with which it may or may not be associated."[14] There are over 100 different autoimmune diseases, that cross different medical specialties, such as rheumatology, endocrinology, hematology, neurology, cardiology, gastroenterology, and dermatology, and result in different clinical pictures and symptoms for every patient. Due to this, diagnosing an autoimmune disease is very difficult. Such specialties usually focus on one autoimmune disease and virtually no research has been don on autoimmunity as the underlying cause.

This is were bioinformatics comes in, making it possible to spot similarities in how patients bodies respond to autoimmune diseases, in stead of focusing on one medical specialty or one disease. This especially comes in handy since it has been shown that having co-occurring autoimmune diseases is not infrequent, and how they might relate to each other. This points to multi-label learnings, where labels can coexist and are not exclusive.

This research gives an experimental comparison of methods for multi-label learning and is done based on three well known classification algorithms. Learners are made in order to predict the most common autoimmune diseases based on patients' symptoms and conditions.

It is said that it takes on average 4.6 years, 5 doctors and $50,000 for a patient to receive the correct diagnosis.[2] Furthermore, it has been estimated that health care costs in the U.S. for autoimmune diseases is over $100 billion dollars a year. Furthermore, autoimmune diseases are in the top ten leading causes of all deaths among U.S. women under 65 years[39] and is the fourth-largest cause of disability for these women.[25]
For these reasons the aim of this research is to assist medical professionals in diagnosing a patient with the correct autoimmune disease faster by analyzing his health records.

First, section 2 describes the data that is used, explains the cleaning, transformation and selection of the data and gives an analysis of the patients and health records. Section 3describes the different methods in dealing with the multi-label nature of diagnosing autoimmune diseases and the three classifiers that are used. Furthermore, it describes the evaluation measures to decide what model works best. Next, section 4 gives the experimental setup and expands on feature engineering and selection and describes the process of parameter selection. Section 5 gives the results and evaluates them. It gives the optimized parameter settings, describes the performance and the reliability of the models. Next a comparison between models is done. Finally, section 6 and section 7 give a conclusion and discussion on the research, respectively.

2 Data

The data used for this research is based on *Flaredown - The world's most advanced symptom tracker*. Flaredown describes itself as "a free web and mobile app that helps patients track and visualize their illness, treatments, and symptom triggers so that they can understand how their choices affect their health".[15] The reasoning behind the choice of this data is that some patients deteriorate quickly, while for others the most severe symptoms arrive after years. This makes predicting an autoimmune disease very difficult and in order to reduce the influence of time, data is used from a period aggregated over 2 years.

This section focuses on the data itself, first a short description of the data is given. Next, an explanation of the cleaning of the data is given, which is described in subsection 2.1. Third, an exploratory data analysis is done in subsection 2.2, then the data is transformed and subsets are made, to shape the data in a way that fits the aim of this research in subsection 2.3. Finally, the adjusted data is analyzed in subsection 2.4.

The data contains the following columns

- The unique user_id
- The age
- The gender
- The country of residence
- A checkin_date
- A trackable_id
- A trackable_type
- A trackable_name
- A trackable_value

The trackable_type consist of six different possibilities, namely:

- Condition; an illness/syndrome a patient suffers from
- Food; anything they consumed in order to find causes of a flare
- Symptom; a physical or mental feature that might indicate a condition
- Tag; a miscellaneous category, containing mostly activities
- Treatment; any medication or care someone uses
- Weather; the weather circumstances of that day, in order to find causes of a flare

Based on the trackable_type a specification is given in the trackable_name. The trackable_value gives the severity of the symptom or condition, the dose of treatment or other extra information.

The trackable_id is disregarded in this research.
Table 1 shows a summary of the data divided over the nine columns. It shows that 973,583 observations are logged, and that the application was used by 6,307 unique users. At first sight it becomes clear that data is entered incorrectly. As shown in table 1 the minimal age in the data is equal to $-196,694$. Furthermore, the unique sexes are split into NA, "male", "female", "other" and "doesn't say".

Table 1: Summary of the observations in the original data

	User id	Sex	Country
Length	973,583	973,583	973,583
NA's	0	13,622	17,369
Unique	6,307	5	96

	Trackable type	Trackable name	Trackable value
Length	973,583	973,583	973,583
NA's	0	0	112,377
Unique	6	22,525	3,909

	Age	Check-in date	Trackable id
Length	973,583	973,583	973,583
Min	-196,694	2015-05-24	1
1st Qu.	24	2016-02-25	189
Median	30	2016-10-23	553
Mean	31.93	2016-09-01	2,588
3rd Qu.	39	2017-03-29	3.383
Max	2.016	2017-06-23	14.923
NA's	43.244	0	0
Unique	89	762	14.640

2.1 Data cleaning

Since the data is hand entered by ordinary people the data contains many mistakes and is unstructured, since no rules exist about logging information on the application. Not only typing errors occur frequently, but it also contains many synonyms for the same term, or people logging in their own language. A lot of time went into cleaning the data from noisy and inconsistent data to data structured and fit for the aim of this research.
First the characters for user_ids are transformed into unique numeric ids, ranging from 1 to 6,307. As stated in the last subsection the ages entered in the application were not credible. So for the variable age it is assumed ages equal to or below zero and above 90 are entered wrong. The reasoning behind this is the average life expectancy of 72 years according to the World Health Organization.[42] To account for outliers in the data, the average age per country is considered. The maximum equals a life expectancy of 87.1 years in Japan, and since this is an average, this is rounded up. These have to be changed into Not Available (NA). Furthermore, the recorded names for sex were incorrect as well. All values for "other" and "doesn't say" are set to NA. Making all entered trackable_names and trackable_values lowercase already reduced the number of unique trackable_names from 22,525 to 21,253 and the number of unique trackable_values from 3,909 to 3,827.

Next, all trackable names were cleaned per trackable type. The following six revisions are done for every trackable type:
– Abbreviations are written out.

- If a trackable name contains multiple names in one observation, they are split and logged in the dataset as a new observation.
- Names are merged when synonyms exist.
- All names are written in singular form
- Adjectives are removed
- Trackable names that belong to other trackable types are moved to the correct type. (e.g. observations that contain the words "syndrome" or disease are changed into type Condition.)

The most important part was cleaning the trackable_type Condition. Most of the entered conditions are not autoimmune diseases, but symptoms, immune deficiencies, other illnesses or diseases that go hand in hand with an autoimmune disease or may even be caused by treatments for the autoimmune disease. So all entered conditions were compared to two lists of autoimmune diseases. The first is from the Autoimmune Registry, Inc. (ARI)[3], Ari calls itself "a hub for research, statistics, and patient data on all autoimmune illnesses." According to this list there are 156 different autoimmune diseases. The second list includes 144 different autoimmune diseases and was compiled by the American Autoimmune Related Disease Association, Inc. (AARDA) [1]. AARDA says it "is the only national nonprofit health agency dedicated to bringing a national focus to autoimmunity, the major cause of serious chronic disease." If the condition entered by the user is included within one of the two lists mentioned above, the trackable_type of that observation is changed into the trackable_type Autoimmune Disease (AD).
Text mining, text analysis and frequently used words and patterns are used to clean the condition names. If a condition contains any word, the plural, the conjugation, the superlative or any other form of the word mentioned in the list in Appendix A, the trackable_type is changed into Symptom. These are usually symptoms caused by a condition or side effects caused by medication. Finally, there is a group called "Miscellaneous", that represent conditions that are measured in less than 10 observations. This group aggregates 859 unique names and combines to 2,338 observations. This is done because it is assumed that having a condition that is this rare, will not influence a trained model, or make it extra complex. What is left in the trackable_type Condition, are mostly non-autoimmune diseases and syndromes.

Next, within the trackable_type Symptom the observations need cleaning as well. Some examples for synonyms that were aggregated into one form for trackable_type Symptom are:

- Combine pain, ache, sore into pain.
- Combine anxiety and panic and worry into anxiety
- Combine abdominal, stomach, belly and tummy into abdominal.
- Impaired functioning combines terms like can't get up, working is more difficult, need help with things, etc.
- Impaired walking combines terms like problems walking, need to use a walking stick, etc.
- Impaired sight combines terms like blurry vision, depth perception issues, etc.
- Impaired memory combines terms like memory issues, poor memory, trouble remembering, etc.
- Impaired speech combines terms like word swapping, talk slurry, etc.

This is done since too many unique names exist for the same term, but using synonyms. The same reasoning holds as before. Symptoms that are left are clustered for keywords

such as limb, joint, muscle, facial, etc. If a specific form is a frequently observed name, it is kept separate. An example is burning feet, this is a trackable_name that occurs relatively often as a symptom, so this is not categorized as a burning extremity. Finally, there is a group called "Miscellaneous", which represent symptoms that are very rare and are observed less than 25 times. This group aggregates 622 unique symptoms and combines to a total of 3,499 observations.
The trackable_types Tag, Treatments and Weather are disregarded, since they are beyond the scope of this research.
Eventually, the aggregating of the names results in a decrease of unique names from 5,739 to 476. It is assumed that grouping symptoms and conditions that are very similar will make the model less complex and make it more feasible to train a well predicting model.

2.2 Exploratory data analysis

After cleaning all the data an altered dataset is made. The structure of the data stays the same. But in stead of six unique trackable_types, the set now contains seven. A newly added type AD exists. Table 2 shows the number of records of the original data and the, cleaned, altered data. It shows that almost 3,000 extra observations were added, because many conditions and symptoms were split into two or even more observations. The mean number of entries per unique user increased for the same reason, and the number of unique users remain the same. It also includes the mean age per unique user, the unique number of home countries of users and the number of unique check-in dates.

Table 2: Number of records original data vs. cleaned data

	Original data	Cleaned data
Observations	973,583	976,533
Unique users	6,307	6,307
Mean entries per unique user	154.36	154.83
Median entries per unique user	17	17
Mean age per unique user	-5.96	31.43
Median age per unique user	29	30
Number of unique home countries	96	96
Number of unique check-in dates	762	762

As stated in table 2 the mean number of entries per unique user equals 154.8 entries, whereas the median only equals 17 entries. Figure 1, A illustrates this in bins of size 20. It shows that, by far most people only have between 0 and 20 entries in the dataset. It also shows that the bin with over 200 entries is quite large, and the maximum number of entries for a patient equals 22,749 entries, which increases the mean number of entries fast.
Figure 2, A shows that check-in dates range between 24-05-2015 and 23-06-2017. It shows that the number of entries starting from January 2017 are generally higher than the other years. This might be because the application received more attention. Furthermore, it seems as if the months April, May or June show a peak as well. Figure

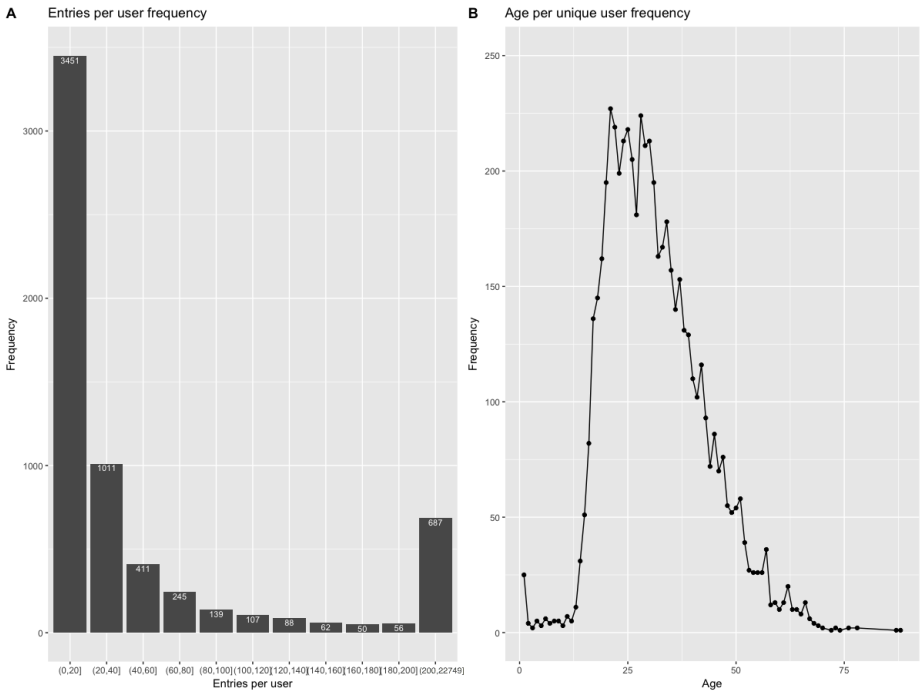

Fig. 1: A: Entries per unique user frequency. B: Age per unique user frequency

2, C show the same premonition. Unfortunately, the time window within the data is not long enough to justify this. Since users of the application are from countries all over the world, the month can not be linked to a season and the trackable_type Weather might be more interesting than a time line itself. Figure 2, B shows the mean number of entries per day. It shows that the number of entries from Fridays until Mondays are relatively lower than the rest of the week. People tend to enter negative feelings over positive ones. This might imply that people have more rest during the weekend and start feeling worse after working on Mondays.

Figure 1, B shows the distribution of the age per unique user. It show that especially many patients between the ages of 23 and 30 use the application. One cause is the fact that younger people tend to use mobile applications more. But literature shows that mostly women in their childbearing years are affected by autoimmune diseases.[2]
So the ages in the dataset clearly correspond to the literature, it has also been cited that women have a higher chance of having an autoimmune disease. The literature states that overall, women are three times more likely to have an autoimmune disease and it is estimated that 75% of American patients suffering from autoimmune diseases are women.[2]. Some specific autoimmune diseases have an even higher incidence among women. These statements are clearly represented in the data as well, which is illustrated in figure 3. It shows that almost 80% of the users of the application is female, where only 10% is male. The other 10% did not enter information on their gender.

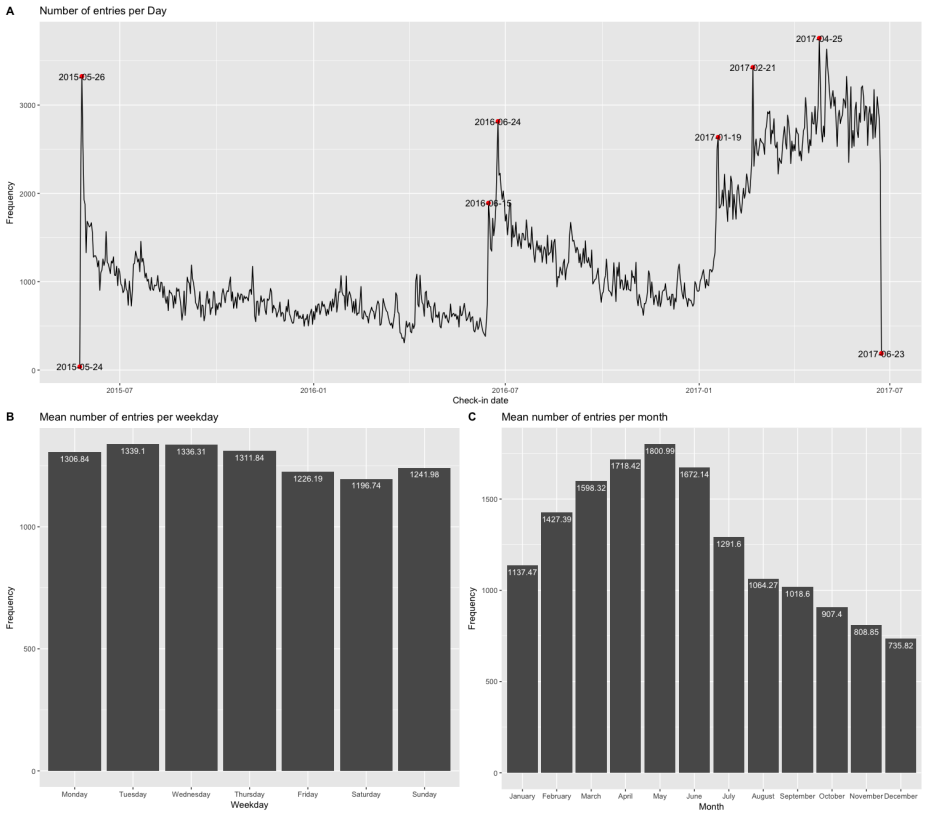

Fig. 2: A: Mean umber of entries per day. B: Mean number of entries per weekday. C: Number of entries per month.

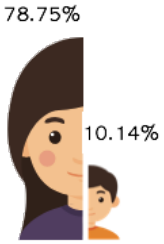

Fig. 3: Distribution of gender per unique user within the dataset

Finally, literature also states that one of the probable causes of autoimmune disease are caused by environmental factors, they are said to have a stronger influence than genetics.[20] An example of a suspected trigger is a "western diet" consisting of high-

fat, high-sugar and highly processed foods.[22] The last theory is called the hygiene hypothesis, because of vaccines and antiseptics in western countries, the lack of exposure to germs make immune systems overreact to otherwise harmless substances.[33] This corresponds to the findings from the data, figure 4 shows a world map, that shows that most users are from the United States, United Kingdom and Australia. This might also point to another cause of the differences, maybe the language barrier plays a part in using the application.

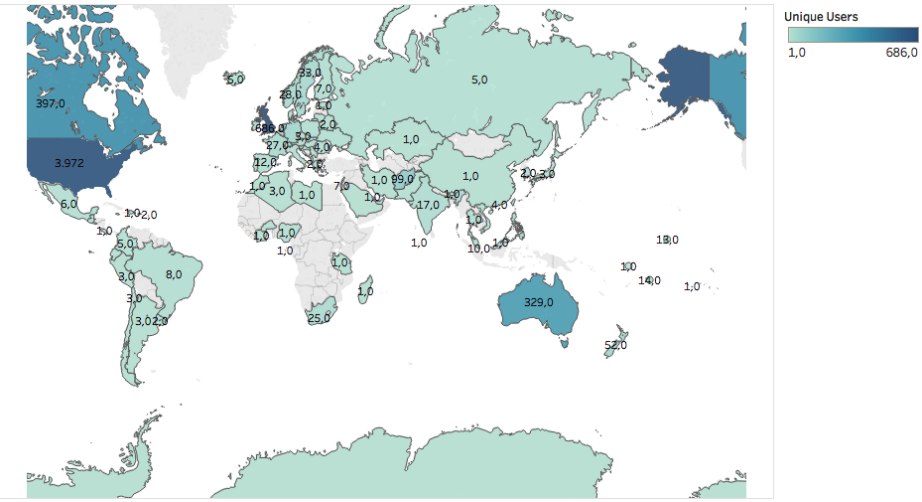

Fig. 4: Unique users per Country

Figure 5 shows the distribution between the trackable_types for both the original dataset and the altered one. In both cases the trackable_type Symptom includes by far the highest number of observations. The type AD is a newly added trackable_type and covers around 4% of the observations. Symptoms, Weather and Food are types that grew in size, this is mainly because trackable_names were split if multiple names were entered as one observation. The trackable_type Condition decreased in size the most, since most autoimmune diseases were filtered from this type. Types Treatment and Tag decreased as well, this is in particular due to observations that were entered under the wrong trackable_type.

Figure 6 give the top 200 entered names for Autoimmune diseases, Conditions and Symptoms. Figures 7, A and B give the top 10 most frequent names for Conditions and Symptoms as well. It shows that the most common condition is the irritable bowel syndrome (IBS), it is shown that IBS is closely linked to the immune system and autoimmune diseases like celiac disease, but the link itself is not well understood.[19][37] Many fundamental pathogenic mechanisms link autoimmune diseases and cause the symptoms to be similar. The most common symptoms include fatigue, and joint pain, which are found for all autoimmune diseases. These are examples of Figure 8, A shows the top ten autoimmune diseases. A total of 74 different autoimmune diseases were distinguished within the dataset. Oddly enough, five out of ten, including the two most

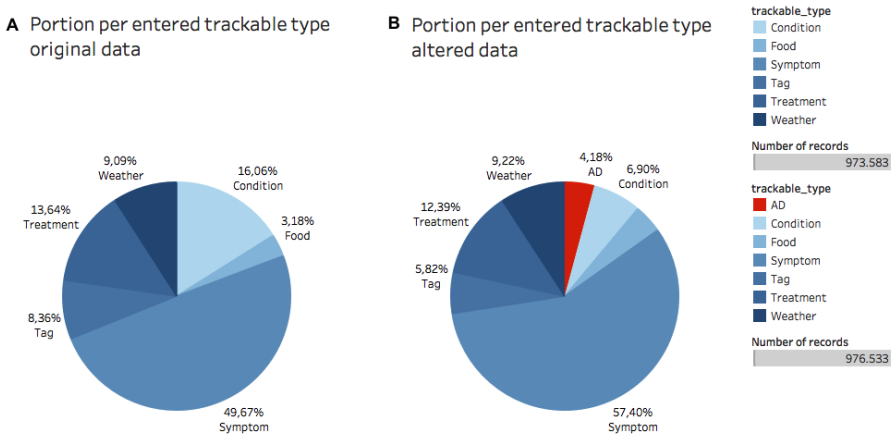

Fig. 5: A: The original distribution between trackable_types. B: The altered distribution between trackable_types.

frequently occurring autoimmune diseases in the dataset, are not mentioned in the 14 most common autoimmune diseases according to Healthline.[40]

Some people have multiple autoimmune diseases. The mean amount of autoimmune diseases for a user is equal to 1.538 and the maximum is equal to eight diseases. This distribution is shown in figure 8, B. It is said that around 25% of patients suffering from an autoimmune disease have multiple autoimmune diseases.[11] The number of patients with multiple co-occurring autoimmune diseases within the dataset is over 35%.

Some AD's rarely occur in the dataset as described in table 3. It shows that there are 13 autoimmune diseases that only have one patient and only two autoimmune diseases that have over 500 patients.

Table 3: Number of autoimmune diseases with x number of patients diagnosed

# patients	# AD's	# patients	# AD's	# patients	# AD's
1	13	6	2	(10,20]	7
2	7	7	1	(20,50]	7
3	5	8	0	(50,100]	9
4	2	9	1	(100,500]	13
5	3	10	2	(500,1232]	2

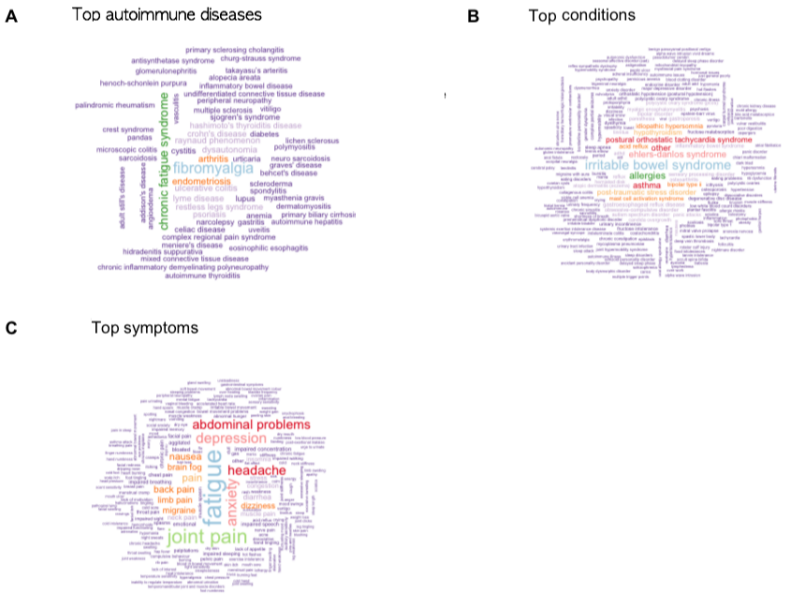

Fig. 6: A: Top 200 Autoimmune Disease frequency . B: Top 200 conditions frequency. C: Top 200 symptom frequency.

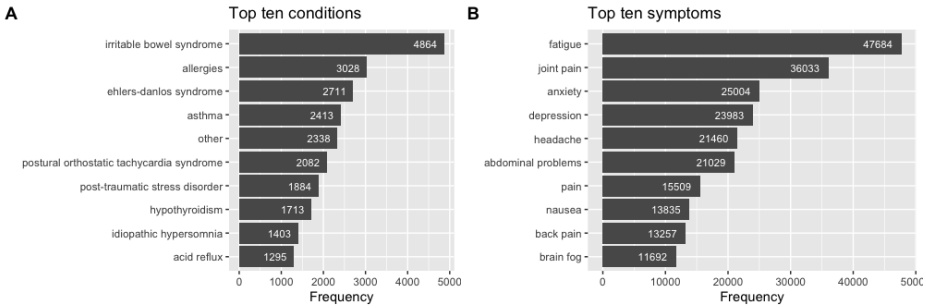

Fig. 7: A: Top 10 most frequent conditions. B: Top 10 most frequent symptoms.

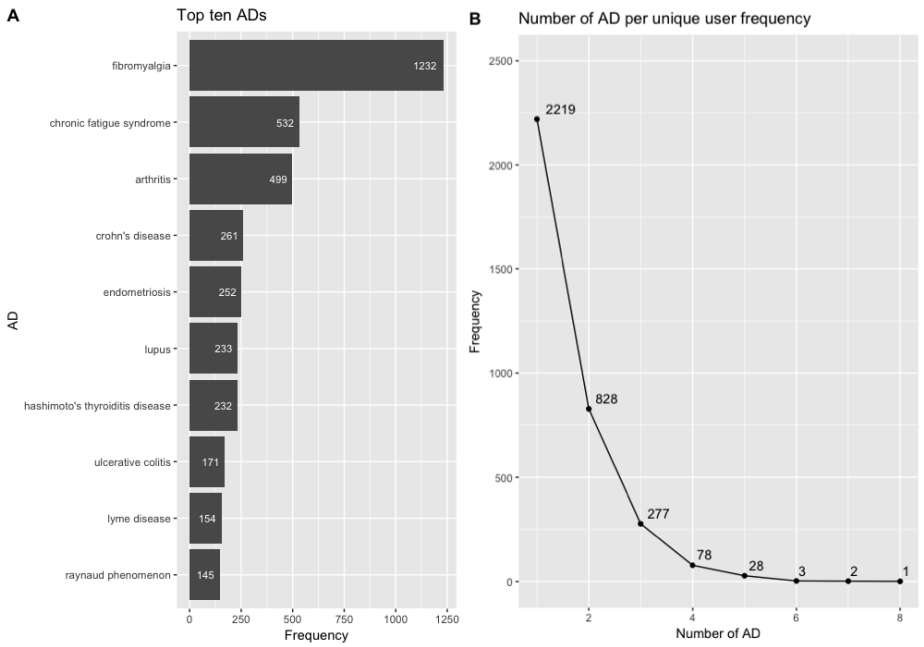

Fig. 8: A: Top ten Autoimmune diseases, B: Number of Autoimmune diseases per unique user

2.3 Data transformation and selection

As described in table 3 in the last subsection, some autoimmune diseases are very rare in the dataset. There are 13 autoimmune diseases that only have one patient, predicting these 13 disease will be impossible for a model.
For this reason the diseases that have at least 100 patients are used. This leaves 16 of the 74 autoimmune diseases. Within the dataset a total of 3,066 patients suffer from (at least) one of these diseases. Important to note is that the entire dataset is tested, even people that suffer from a different autoimmune disease, these should be predicted as "Miscellaneous". This means that doctors who suspect their patient to have an autoimmune disease can predict if they have one of the 16 autoimmune disease, based on their symptoms. The 2,744 users that have not entered an autoimmune disease are disregarded.
Next, patients that did not enter any conditions and symptoms are disregarded as well, which removes 127 patients from the dataset.

Since no correlation is proven in literature between symptoms/ conditions and the age, sex or country of residence, this information is ignored. The final dataset is made using one row of data for every unique user_id. For all conditions and symptoms that occur in the data a column is made. Every column contains a different unique symptom or condition with a boolean value of 1 if the patient with that unique user_id suffers from this symptom or condition, and a 0 otherwise. It is assumed that if a patient logged this condition or symptom, anywhere between 24-05-2015 and 23-06-2017 they suffer from this symptom or condition. If a patient entered this symptom or condition multiple times, it is treated the same as when it would have been logged only ones. The same goes for the label set for every user_id for the possible autoimmune diseases to predict. This aggregates a total of 499,794 observations into 3,436 observations, one row per unique patient.

2.4 Data analysis

The final dataset used for this research results in a total of 476 explanatory variables. And a set of 17 response variables with a boolean value for every autoimmune disease for every unique user_id.

Figure 9 shows that an average patient in the dataset has 10.28 symptoms, 2.54 conditions and 1.54 autoimmune diseases.
Figure 10 states the distribution of the autoimmune diseases to be predicted for the patients included in the final dataset.

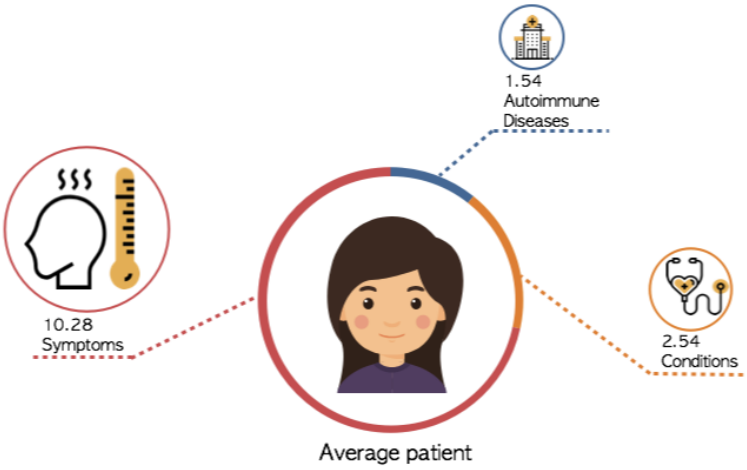

Fig. 9: The information that is known about an average patient

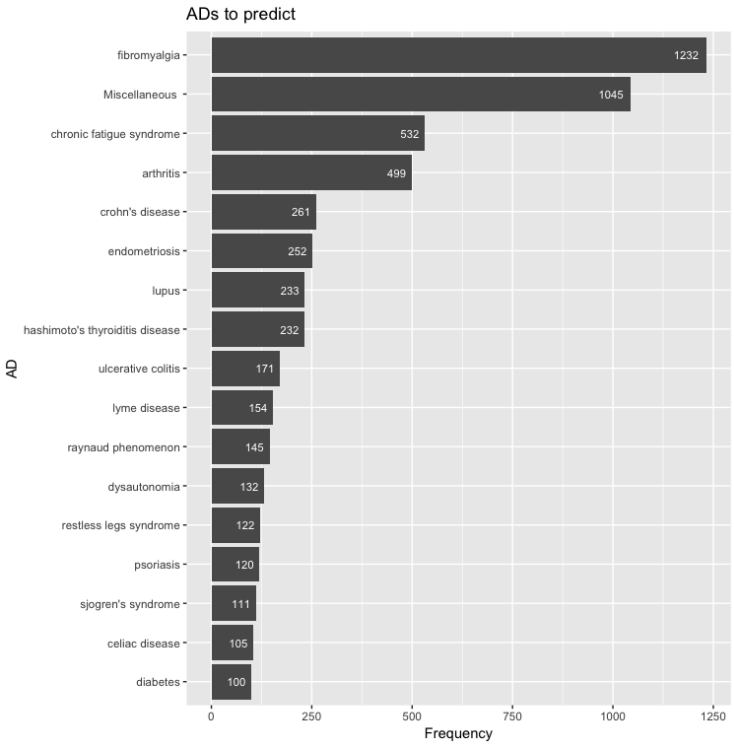

Fig. 10: The 16 autoimmune diseases that are predicted.

3 Methodology

This section will discuss the different methods used, outlines the different algorithms used and describes how the performance is measured. Subsection 3.1 will discuss the k-Nearest Neighbors, Random Forest and Support Vector Machine algorithms. These three algorithms are chosen since it are well known, easy to implement algorithms. All three have implementations in various fields, including bioinformatics. Furthermore, all three have multi-label implementations. Subsection 3.2 describes how scores and performances measures are calculated to determine the quality between the implementations.

3.1 Methods for dealing with multi-label classification

This section describes the methods that are used to handle patients having co-occurring autoimmune diseases. Table 4 shows an example of a single-label classification whereas table 5 shows a multi-label problem. Next, table 6 shows how table 5 can be restructured into a categorical multi-label classification problem.
Multi-label classification has received much attention in literature, including the field of bioinformatics.[10][38][43][9]

Three well known algorithms are implemented using tree different methods to adapt to the multi-label classification problem at hand. Literature states two main categories in methods for handling multi-label classification, namely the Problem Transformation method and the Algorithm Adaptation method. [35]

Table 4: Single-Label Classification

X_1	X_2	X_3	X_4	X_5	Y
1	0	0	1	1	1
0	0	0	1	1	0
1	1	0	0	1	2
1	1	1	0	1	1
0	1	1	1	1	??

Table 5: Multi-Label Classification

X_1	X_2	X_3	X_4	X_5	Y
1	0	0	1	1	$\{Y_1, Y_2\}$
0	0	0	1	1	$\{Y_3\}$
1	1	0	0	1	$\{Y_2, Y_3\}$
1	1	1	0	1	$\{Y_1, Y_4\}$
0	1	1	1	1	??

Table 6: Multi-label Classification

X_1	X_2	X_3	X_4	X_5	Y_1	Y_2	Y_3	Y_4
1	0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	1	0
1	1	0	0	1	0	1	1	0
1	1	1	0	1	1	0	0	1
0	1	1	1	1	??	??	??	??

– Problem Transformation methods
Multi-label classification problems are converted into single-label classifications problems. Here the data is adapted to fit the algorithm. This is done using the following two methods:
  - Label Powerset method[30]
    In the Label Powerset method a new class is defined for each unique combination of multi-labels. It makes a single multi-class problem with 2^L possible class values, with L the number of possible labels. Tables 8 and 9 illustrate this. Table 7 gives the advantages and disadvantages of this method.
  - Binary Relevance method
    For the Binary Relevance method L separate binary problems are created, one for each possible label. Again, these are trained with any off-the-shelf binary classifier. An illustration of this process is given in tables 10 and 11, for the first, second and fifth label in the label set. A disadvantage of this method is that in many cases no positive decision function exists, so an observation remains unclassifiable. As an heuristic any unclassifiable observation will be classified as having "no autoimmune disease / miscellaneous autoimmune disease". Another big disadvantage is that it does not take label dependency into account, which can be solved using a Chain Classifier.[31][32] It makes L binary problems, but in this case includes previous predictions as explanatory variables. This is illustrated in table 12, for the first, second and fifth label in the label set. An ensemble of Classifier Chains are implemented here, which is composed by a set of Classifier Chains. Support Vector Machines are often used for implementing the Binary Relevance Method due to their high generalizable nature.[6][44] Table 7 gives the advantages and disadvantages of this method. The Binary Relevance method using an ensemble of Classifier Chain will be used, but simply named the Binary Relevance method for the remainder of this research.
– Algorithm Adaptation method
As opposed to the Problem Transformation methods, in the Algorithm Adaptation method the algorithm is adapted to fit the data. Conventional classifiers are implemented in a way that allows for multi-label classification. Table 7 gives the advantages and disadvantages of this method.

Table 7: Advantages and disadvantages of the three methods for handling multi-label classification

	Advantage	Disadvantage
Label Powerset	- Any classifier can be used - Easy to implement	- Complexity due to the number of classes - Imbalance of the classes - Overfitting due to unknown combination of labels
Binary Relevance	- Any classifier can be used - Number of classes to predict does not increase - Easy to implement	- Many observations are unclassifiable - Computationally expensive
Algorithm Adaptation	- Data does not need to be altered	- Classifier needs to be reimplemented

In the next sections the three implemented algorithms are described as well as the Algorithm Adaptation method for that specific algorithm. Since the Problem Transfor-

mation methods behave in the same way for all tree algorithms, no further explanation is given.

Table 8: Multi-label Classification using Label Powerset method

$\bar{X}$	Y_1	Y_2	Y_3	Y_4	Y_5
$\bar{x}^{(1)}$	0	0	1	1	1
$\bar{x}^{(2)}$	0	0	1	1	0
$\bar{x}^{(3)}$	1	0	0	1	1
$\bar{x}^{(4)}$	1	1	0	0	1
$\bar{x}^{(5)}$	0	1	0	0	0

Table 9: Multi-class Classification due to Label Powerset method

$\bar{X}$	$Y \in 2^L$
$\bar{x}^{(1)}$	00111
$\bar{x}^{(2)}$	00110
$\bar{x}^{(3)}$	10011
$\bar{x}^{(4)}$	11001
$\bar{x}^{(5)}$	01000

Table 10: Multi-label Classification using Binary Relevance method

$\bar{X}$	Y_1	Y_2	Y_3	Y_4	Y_5
$\bar{x}^{(1)}$	0	0	1	1	1
$\bar{x}^{(2)}$	0	0	1	1	0
$\bar{x}^{(3)}$	1	0	0	1	1
$\bar{x}^{(4)}$	1	1	0	0	1
$\bar{x}^{(5)}$	0	1	0	0	0

Table 11: Single-class Classification due to Binary Relevance method

$\bar{X}$	Y_1
$\bar{x}^{(1)}$	0
$\bar{x}^{(2)}$	0
$\bar{x}^{(3)}$	1
$\bar{x}^{(4)}$	1
$\bar{x}^{(5)}$	0

$\bar{X}$	Y_2
$\bar{x}^{(1)}$	0
$\bar{x}^{(2)}$	0
$\bar{x}^{(3)}$	0
$\bar{x}^{(4)}$	1
$\bar{x}^{(5)}$	1

$\bar{X}$	Y_5
$\bar{x}^{(1)}$	1
$\bar{x}^{(2)}$	0
$\bar{x}^{(3)}$	1
$\bar{x}^{(4)}$	1
$\bar{x}^{(5)}$	0

Table 12: Using the Chain Classifier to improve the Binary Relevance method

$\bar{X}$	Y_1
$\bar{x}^{(1)}$	0
$\bar{x}^{(2)}$	0
$\bar{x}^{(3)}$	1
$\bar{x}^{(4)}$	1
$\bar{x}^{(5)}$	0

$\bar{X}$	Y_1	Y_2
$\bar{x}^{(1)}$	0	0
$\bar{x}^{(2)}$	0	0
$\bar{x}^{(3)}$	1	0
$\bar{x}^{(4)}$	1	1
$\bar{x}^{(5)}$	0	1

$\bar{X}$	Y_1	Y_2	Y_3	Y_4	Y_5
$\bar{x}^{(1)}$	0	0	1	1	1
$\bar{x}^{(2)}$	0	0	1	1	0
$\bar{x}^{(3)}$	1	0	0	1	1
$\bar{x}^{(4)}$	1	1	0	0	1
$\bar{x}^{(5)}$	0	1	0	0	0

3.1.1 k-Nearest Neighbors

k-Nearest Neighbors (kNN) is a widely disposable algorithm and finds applications in various fields, as well as bioinformatics, such as the breast cancer diagnosis problem.[34] It does not make any assumptions about the distribution of underlying data and is easy to implement for both single-label as multi-label classifications.

kNN is an algorithm that classifies n-dimensional objects (i.e. an observations with n features) based on their similarity (e.g., distance functions) to other n-dimensional objects. Here an observation is classified as the class most common among the k most

similar observations. With k as a positive integer.[27]

Distance Functions

The distance between observations can be calculated in multiple ways. Table 13 shows N observations with n explanatory variables. The distance between two observations i and j is defined as $d(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = ||\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j||$

Table 13: N observations with n explanatroy variables

Observation	Explanatory Variables					
$\bar{\mathbf{x}}_1$	x_{11}	x_{12}	$\cdots$	x_{1k}	$\cdots$	x_{1n}
$\bar{\mathbf{x}}_2$	x_{21}	x_{22}	$\cdots$	x_{2k}	$\cdots$	x_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$\bar{\mathbf{x}}_i$	x_{i1}	x_{i2}	$\cdots$	x_{ik}	$\cdots$	x_{in}
$\bar{\mathbf{x}}_j$	x_{j1}	x_{j2}	$\cdots$	x_{jk}	$\cdots$	x_{jn}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$\bar{\mathbf{x}}_N$	x_{N1}	x_{N2}	$\cdots$	x_{Nk}	$\cdots$	x_{Nn}

Since the data used in this research only contains boolean explanatory variables $\in [0, 1]$ the Euclidean-, Manhattan-, Canberra and Squared Euclidean- and Minkowski distance all end up with the same result. Furthermore, the Chebyshev distance always calculates the distance as being either 0 or 1, which does not make this a good distance measure either.

The decision was made to work with the Euclidean distance, since this is the most commonly used measure, the Pearson distance and the Jaccard distance. These are defined as follows:[27]

Euclidean distance:

$$d(\bar{x}_i, \bar{x}_j) = ||\bar{x}_i - \bar{x}_j|| = \sqrt{\sum_{\forall k} (x_{ik} - x_{jk})^2} \qquad i, j, k \in \mathbb{Z}^+, \quad i \neq j$$

Pearson distance:

$$d(\bar{x}_i, \bar{x}_j) = ||\bar{x}_i - \bar{x}_j|| = \sum_{\forall k} \frac{(x_{ik} - x_{jk})^2}{x_{jk}} \qquad i, j, k \in \mathbb{Z}^+, \quad i \neq j$$

Jaccard distance:

$$d(\bar{x}_i, \bar{x}_j) = ||\bar{x}_i - \bar{x}_j|| = 1 - \frac{|\bar{x}_i \cap \bar{x}_j|}{\bar{x}_i \cup \bar{x}_j} = 1 - \frac{\bar{x}_i \cap \bar{x}_j}{|\bar{x}_i| + |\bar{x}_j| - |\bar{x}_i \cap \bar{x}_j|}$$
$$= \frac{|\bar{x}_i \cup \bar{x}_j| - |\bar{x}_i \cap \bar{x}_j|}{|\bar{x}_i \cup \bar{x}_j|}$$

This means it calculates the distance by dividing the difference of the sizes of the union and the intersection by the size of the union. In other words, per explanatory variable:

$$d(\bar{x}_i, \bar{x}_j) = ||\bar{x}_i - \bar{x}_j|| = 1 - \frac{\sum_{\forall k} (x_{ik} \cdot x_{jk})}{\sum_{\forall k} {x_{ik}}^2 + \sum_{\forall k} {x_{jk}}^2 - \sum_{\forall k} (x_{ik} \cdot x_{jk})} \qquad i, j, k \in \mathbb{Z}^+, \quad i \neq j$$

Note: for the division by zero the following is done:

- case: $^0/_0 \rightarrow 0$
- case: $^n/_0 \rightarrow {}^n/_\epsilon, \quad \epsilon = 0.00001.$

Informal kNN algorithm
The intuition behind the kNN algorithm implemented here can be explained with the following steps:

1. Determine k.
2. Calculate the distance between the queried observation and all observations in the training data.
3. Sort the distances in a decreasing order and based on the k observations with the minimum distance, determine the k nearest neighbors.
4. Gather the response variables of the k nearest neighbors
5. Classify the queried observation based on a set threshold function.

Section 4 gives the experimental setup for different values of k, the distance function, and the threshold function.

Formal kNN algorithm
Now the formal kNN algorithm can be defined given:
A train set $\bar{T}$, with observation $\bar{t}$, $\bar{t} \subseteq \bar{T}$.
A dataset $\bar{Q}$ to classify, with queried observation $\bar{q}$, $\bar{q} \subseteq \bar{Q}$.
And an integer k.
The set of k nearest neighbors of $\bar{q}$ from $\bar{T} := kNN(\bar{q}, \bar{T})$
now are a set of k observations from $\bar{T}$ such that:[27]

$$\forall_{\bar{o}} \in kNN(\bar{q}, \bar{T}), \quad \forall_{\bar{t}} \in \{\bar{T} - kNN(\bar{q}, \bar{T})\}, \quad d(\bar{o}, \bar{q}) \leq d(\bar{t}, \bar{q}) \tag{1}$$

Combining each observation $\bar{q} \in \bar{Q}$ with its k nearest neighbors from T, is defined as:[27]

$$kNNjoin(\bar{Q}, \bar{T}) = \{(\bar{q}, \bar{t}) \mid \forall_{\bar{q}} \in \bar{Q}, \forall_{\bar{t}} \in kNN(\bar{q}, \bar{T})\} \tag{2}$$

3.1.1.1 Algorithm Adaptation method

To adapt the algorithm to work with multi-label data, the ML-kNN classifier is used. The k nearest neighbors are identified based on the traditional kNN algorithm as described above. To classify the label for the queried observation the maximum a posteriori principle is used, based on statistical information of the label sets of the k nearest neighbors for the queried observation.

Informal ML-kNN algorithm
The normal kNN algorithm is followed to determine the k nearest neighbors and their respective label sets. The difference lays in the decision function. The intuition behind the final prediction depends on the most common labels among the k nearest neighbors, combined Bayesian inference. Meaning the probability for a label increases as more evidence or information becomes available.

Formal ML-kNN algorithm
Now the formal ML-kNN algorithm can be defined given:[45]
A train set $\bar{T}$, with observation $\bar{t}$, $\bar{t} \subseteq \bar{T}$.
A dataset $\bar{Q}$ to classify, with queried observation $\bar{q}$, $\bar{q} \subseteq \bar{Q}$.
And an integer k.

The set of k nearest neighbors of $\bar{q}$ from $\bar{T} := kNN(\bar{q}, \bar{T})$.

Given observation $\bar{t}$ and its associated label set $Y \subseteq \mathcal{Y}$,
Let $\bar{y}_{\bar{t}}$ be the classification vector for $\bar{t}$, where its l-th component,

$$y_{\bar{t}}(l) \quad , l \in \mathcal{Y} = \quad \begin{cases} 1 & if\ l \in Y \\ 0 & otherwise \end{cases}$$

Now, a membership counting vector can be defined, counting the number of neighbors of $\bar{q}$ belonging to the l-th class as:

$$C_{\bar{q}}(l) = \sum_{\bar{t} \in kNN(\bar{q}, \bar{T})} y_{\bar{t}}(l) \qquad , l \in \mathcal{Y} \tag{3}$$

For each queried observation $\bar{q} \subseteq \bar{Q}$, first identify $kNN(\bar{q}, \bar{T})$ and let:
$H_1^l :=$ the event that $\bar{q}$ has label l,
$H_0^l :=$ the event that $\bar{q}$ doest not have label l,
$E_j^l :=$ the event that, among the k nearest neighbors of $\bar{q}$, there are exactly j instances which have label l. $\quad , j \in \{0, 1, ..., k\}$

Now it follows that:

$$\begin{aligned} y_{\bar{q}}(l) &= \arg \max_{b \in \{0,1\}} P(H_b^l \mid E_{C_{\bar{q}}(l)}) \qquad , l \in \mathcal{Y} \\ &= \arg \max_{b \in \{0,1\}} \frac{P(H_b^l) P(E_{C_{\bar{q}}(l)} \mid H_b^l)}{P(E_{C_{\bar{q}}(l)})} \\ &= arg \max_{b \in \{0,1\}} P(H_b^l) P(E_{C_{\bar{q}}(l)} \mid H_b^l) \end{aligned} \tag{4}$$

Now the prior probabilities $P(H_b^l), \quad l \in \mathcal{Y}, b \in \{0, 1\}$ and posterior probabilities $P(E_j^l \mid H_b^l), \quad j \in \{0, 1, ..., k\}$ are needed to determine the category vector $\bar{y}_{\bar{q}}(l)$. These probabilities can directly be estimated from the training data based on frequency counting.

3.1.2 Random Forest

Like the kNN algorithm, the Random Forest is an algorithm that is used for many purposes and fields. An example of an implementation of the Random Forest in the bioinformatics field is the detection and localization of multiple organs multi-channel magnetic resonance scans.[28]
Advantages of Random Forest include the fast and easy implementation, high accuracies in prediction and the ability to handle a large number of explanatory variables without overfitting.[4] Furthermore, they are conceptually easy to understand for anyone. This fact makes it a good algorithm for the medical field. A drawback of the random forest is the inability to handle noisy and incomplete data.

Informal Decision Tree algorithm

A decision tree is a tree such that each decision node corresponds to an explanatory variable whereas the leafs correspond to a classifying label. The problem is to decide which concepts in the hypothesis space will be best in classifying and so, determining the set of rules to follow.

Note that every split in this research deals with Boolean explanatory variables, either having a certain symptom/ condition or not.
In other words, when a set of questions about a patient are asked, one follows the steps and eventually ends up with a certain autoimmune disease; the prediction.

The intuition behind the Decision Tree algorithm implemented here can be explained with the following steps:[16]

1. Measure how well every explanatory variable separates observations into targeted classes, using the entire training set.
2. Choose the best split, so the dataset is split into subsets using the explanatory variable that minimizes the impurities, and increases the homogeneity. This leads to subsets in the daughter nodes belonging to similar classes. This favors splits that lead to bigger partitions and so, ultimately minimizes the errors in prediction.
3. Make a decision node of this explanatory variable.
4. Iterate the steps above, in a recursive matter by making subsets using the remaining explanatory variables until a leaf node is reached at every branch, and a classification can be made by computing the most frequent class.

Informal Random Forest algorithm
The Random Forest algorithm is an ensemble learning method that constructs a number of decision trees based on random parts of the training set and random subsets of the explanatory variables. Combining many of these weak learners tackles the disadvantage of overfitting, and makes the final classifier stronger.
In other words, a prediction is done based on different trees. And the final prediction is the label that has the majority vote among the trees.[16]

The intuition behind the Random Forest algorithm implemented here can be explained given:[16]
$ntree$:= the total number of trees to grow,
$mtry$:= the total number of explanatory variables randomly sampled as candidates at each split.
n := the total number of observations in the training set.
m := the total number of explanatory variables in the training set.

1. Sample n cases with replacement from the training set.
2. At each decision node:
   (a) Sample $mtry$ explanatory variables out of m, where $mtry < m$
   (b) Split the node into daughter nodes using the best split
   (c) Repeat steps (a) and (b), until the tree is grown to the largest extent possible.
3. Recurse steps 1. and 2. until $ntree$ number of trees are grown.

Using the $ntree$ grown trees to make a prediction for the queried observation, the next steps should be followed:

1. Use the explanatory variables of queried observation as input and use the rules of every tree in the forest
2. Store the classification of every tree for the queried observation
3. Calculate the number of votes for each predicted class
4. The final prediction is the classification having the highest number of votes.

Section 4 gives the experimental setup for different values of *ntree* and *mtry*.

Formal Random Forest algorithm
Now the formal Random Forest algorithm can be defined:[7],[8]
Let a train set $\bar{T}$, with observation $\bar{t}$, $\bar{t} \subseteq \bar{T}$.
A dataset $\bar{Q}$ to classify, with queried observation $\bar{q}$, $\bar{q} \subseteq \bar{Q}$.
And the possible label set $\mathcal{Y} = 1, ..., L$
Let $\bar{T} = \{(\bar{t}_i, y_1), ..., (\bar{t}_n, y_n)\}$ the train set with n observations.
Where $\bar{t}_i = (t_{i,1}, ..., t_{i,m})^T$. Meaning the train set has n observations, where each observation consists of m explanatory variables, and a response variable y.

1. Recurse the following steps until *ntree* trees are grown:
   (a) Take a bootstrap sample $\bar{T}_j$ of size n from $\bar{T}$
      i. Use all observations $\{(\bar{t}_i, y_1), ..., (\bar{t}_n, y_n)\}$ from $\bar{T}_j$ in a single node.
      ii. Repeat the following steps using the remaining explanatory variables until a leaf node is found for every branch:
         A. Sample *mtry* explanatory variables out of m, where $mtry < m$
         B. Among these *mtry* explanatory variables, find the split s possible, minimizing the Gini index and so optimizing the node purity.
            The best split is found by minimizing:

$$\theta(\bar{t}, s) = \left(1 - \sum_{l=1}^{L} \left(\frac{n_{l,1}}{n_1}\right)^2\right) + \left(1 - \sum_{l=1}^{L} \left(\frac{n_{l,2}}{n_2}\right)^2\right) \tag{5}$$

            Where the subscripts 1 and 2 represent the daughter nodes of the node. And $n_{l,1}$ and $n_{l,2}$ are defined as the number of cases of class l in the left and right daughter, respectively, such that $n_l = n_{l,1} + n_{l,2}$
         C. Split this node into daughter nodes.
      iii. Let k denote the leaf nodes and let $y_{k1}, ..., y_{kn}$ denote the response variables of $\bar{T}_j$
         The predictions are given by:

$$\hat{h}(\bar{t}) = arg\max_y \sum_{i=1}^{n} I(y_{ki} = y) \qquad , where \begin{cases} 1 & if\ y_{ki} = y \\ 0 & otherwise \end{cases} \tag{6}$$

2. To make a prediction for a new observation $\bar{q}$ calculate

$$\hat{f}(\bar{q}) = arg\max_y \sum_{j=1}^{ntree} I(\hat{h}_j(\bar{q}) = y) \tag{7}$$

   where $\hat{h}_j(\bar{q})$ is the predicted classification for $\bar{q}$ using the j-th tree.

3.1.2.1 Algorithm Adaptation method

The split in the Random Forest algorithm ensures that labels within a node are similar to each other and different from labels in the other daughter node. For multi-label classification the label sets need to be considered, rather than the labels themselves. For this reason a different type of splitting need to be used. This finally result in an algorithm that allows for multiple labels at the leaves.
Let $\bar{T} = \{(\bar{t}_i, Y), ..., (\bar{t}_n, Y\}$ the train set with n observations.

Where $\bar{x}_i = (t_{i,1}, ..., t_{i,m})^T$ and the associated label set $Y \subseteq \mathcal{Y}$ for every observation. Meaning the train set has n observations, where each observation consists of m explanatory variables, and a set of response variables Y.
The difference between the Algorithm Adaptation Method and a standard Random Forest is that a composite normalized Gini index splitting rule is used, which is an average over the entire set of classification labels.[29]

$$\theta(\bar{x}, s) = \frac{n_1}{n}\left(1 - \sum_{l=1}^{L}\left(\frac{n_{l,1}}{n_1}\right)^2\right) + \frac{n_2}{n}\left(1 - \sum_{l=1}^{L}\left(\frac{n_{l,2}}{n_2}\right)^2\right) \tag{8}$$

3.1.3 Support Vector Machine

The Support Vector Machine (SVM) is another well known classifier, that finds many implementations. One implemntation in bioinformatics using a combination of a SVM and a genetic algorithm were used for heart disease classification.[5]

A SVM essentially looks for the optimal separating hyperplane between observations with the same label. If observations of two different classes can be illustrated on a two dimensional graph, a line can be drawn that is able to separate classes, and so divides the graph into two areas. This line is called the hyperplane. The SVM looks for the optimal separating hyperplane by maximizing the margin between the closest points between the classes. The points lying on the boundaries of the margin are called support vectors, these are the critical elements of the training set. The support vectors namely specify the decision function for the prediction.

Informal Support Vector Machine algorithm

For the linear case the following holds:
The maximal margin hyperplanes are two lines at equal distance from the optimal hyperplane. These lines give the boundaries of the margin that separates the two classes. Since the margin between two parallel lines needs to be maximized, one can calculate the distance between the two lines, and maximize this distance. This leads to a quadratic, constrained optimization problem that can be solved using the Lagrangian multiplier method. A graphical interpretation is given in figure 11. Where the arrows show the distance to be maximized, the pink line represents the optimal hyperplane, the dotted lines maximal margin hyperplanes and the classes are represented by circles.

In the case of multi-class classification the SVM uses the one-against-one technique by fitting all cases of having a certain label against all others, and finding the correct classification by a voting mechanism.
In many cases classes are not linearly separable, so the Kernel Trick can be used, in which the data is mapped to a higher dimensionality in order to gain linearly separation. This trick is illustrated in figure 12, where the left image in two dimensions is not linearly separable, but as shown in the right figure, it turns out to be linearly separable in three dimensions.

Kernel Functions

When a situation like described in figure 12 occurs, a kernel function $k(\bar{x}_i, \bar{x}_j)$, k:$\mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ implicitly computes the similarity between $\bar{x}_i$ and $\bar{x}_j$ in $\mathbb{R}^M$.
Two Kernel functions are used in this research, namely the Linear Kernel function and the Radial Basis function (RBF).

$$k(\bar{x}_i, \bar{x}_j) = \begin{cases} \bar{x}_i \cdot \bar{x}_j & if\ Linear \\ \exp(-\gamma \mid \bar{x}_i - \bar{x}_j \mid^2) & if\ RBF \end{cases} \tag{9}$$

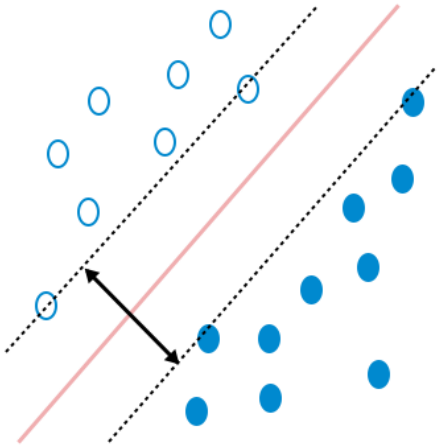

Fig. 11: The Maximum margin hyperplane illustrated

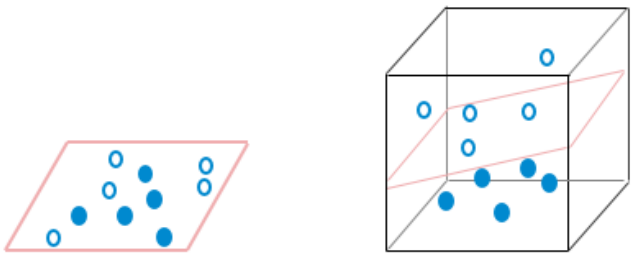

Fig. 12: The Kernel Trick illustrated

Formal Support Vector Machine algorithm

The formal Support Vector Machine algorithm can be defined:[12] The equation defining the decision surface separating the classes is a hyperplane of the form: $\bar{w}^T\bar{x}+b$, with $\bar{w}$ a weight vector, $\bar{x}$ an input vector and b the bias. Furthermore, the total distance between the max margin hyperplanes equals $\frac{2}{||\bar{w}||}$

In order to prevent observations of falling in the margin and observations having to lie

on the correct side of the margin, the following constraints have to hold:

$$\begin{aligned} \bar{w} \cdot \bar{x}_i - b \leq 1 \qquad & if\ y_i = 1 \\ \bar{w} \cdot \bar{x}_i - b \neq -1 \qquad & if\ y_i = -1 \end{aligned}$$

Meaning anything above the boundary is of one class, with label 1, and any observations below it belong to the other class with label -1.
This can be combined into the optimization problem

$$\begin{aligned} & \min \mid\mid \bar{w} \mid\mid \\ s.t. \qquad & y_i(\bar{w} \cdot \bar{x}_i - b) \leq 1 \qquad , for\ i \in (1, ..., n) \end{aligned} \tag{10}$$

By adding cost $C > 0$ per misclassification the influence of misclassified observations is reduced, this allows trading-off margin size against training error of misclassified observations.
A slack variable $\epsilon_i \leq 0$, is added as well. In order to get the optimal hyperplane, equation 11 eventually can be written as:

$$\begin{aligned} & \min \frac{1}{2} \mid\mid \bar{w} \mid\mid^2 + C \sum_{i=1}^{n} \epsilon_i \\ s.t. \qquad & y_i(\bar{w} \cdot \bar{x}_i - b) \geq 1 - \epsilon_i \qquad , for\ i \in (1, ..., n) \\ & \epsilon_i \geq 0 \qquad , for\ i \in (1, ..., n) \end{aligned} \tag{11}$$

The Support Vector Machine algorithm uses tree parameters, namely the kernel function, the γ parameter and the cost parameter.
The γ parameter define a Gaussian function, that is used as similarity measure between two points. This means high values for γ define a Gaussian function with small variance and only points that are close to each other can be considered similar. Vice versa, with low values for γ, two points can be considered similar even if they are far apart.
The cost parameter decides the cost of constraint violation, in other words it trades off misclassification against model complexity. High cost parameters will result in a very complex model, whereas low cost parameter make decisions surface smooth and will probably result in more training errors.
The threshold function used in the Binary Relevance Method and the Algorithm Adaptation Method decides which diseases to include in the prediction. For every label, the algorithm decides the probability of it being fitting to the model. A disease is included in the prediction if the probability is at least equal to x, where x equals the threshold. A set of different settings for the experimental setup are discussed in section 4.

3.1.3.1 Algorithm Adaptation method

The Algorithm Adaptation Method makes use of SVMRank[21]. It reduces a ranking problem into a classification problem over pairs of observations.
Given pairwise observations and the corresponding explanatory variables, a ranking of labels is made to give a prediction. In this way it finds regularities between similar observations in order to find the most fitting labels. Implicitly, the classification label are used to generate pairwise preference constraints for two given observations. Meaning,

for every observation a constraint is made where an ordering of importance is given. This will eventually result in the following model, that looks similar to the SVM:

$$\begin{aligned} & \min \frac{1}{2} \mid\mid \bar{w} \mid\mid^2 + C \sum_{i=1}^{n} \epsilon_{i,j,k} \\ s.t. \quad & \forall (l_i, l_j) \in \bar{r}_1^* : \bar{w}\phi(\bar{x}_1, l_i) \geq \bar{w}\phi(\bar{x}_1, l_j) + 1 - \epsilon_{i,j,k} \\ & \dots \\ & \forall (l_i, l_j) \in \bar{r}_n^* : \bar{w}\phi(\bar{x}_n, l_i) \geq \bar{w}\phi(\bar{x}_n, l_j) + 1 - \epsilon_{i,j,k} \\ & \forall i \forall j \forall k \epsilon_{i,j,k} \geq 0 \end{aligned} \tag{12}$$

With $\bar{w}$ a weight vector, $\bar{l} = \{l_1, ..., l_m\}$ the label set $\phi(l, \bar{x}_1)$ a mapping onto features describing the match between the label and observation, slack variables $\epsilon_{i,j,k}$ and target rankings $\bar{r}^*$ for the n observations: $(\bar{x}_1, \bar{r}_1^*), ..., (\bar{x}_n, \bar{r}_n^*)$
Note that in this case the target is not a class label but a binary ordering relation and the goal is not to minimize the training error.

3.2 Evaluation Measures

Given queried dataset $\bar{Q}$ and its queried observations $\bar{q}$, with $\bar{q} \subseteq \bar{Q}$, having n observations.
Furthermore, given $\bar{q}$ with its associated diagnosed label set $Y \subseteq \mathcal{Y}$
and the length of the possible label set $\mathcal{Y} := m$ the following equation can be defined, given its l-th label:

$$y_{\bar{q}}(l) \quad , l \in \mathcal{Y} = \quad \begin{cases} 1 & if\ l \in Y \\ 0 & otherwise \end{cases}$$

This represents whether a certain patient is diagnosed with a certain autoimmune disease.
In the same way the predicted label for the queried observation can be defined, given its l-th label, and $\hat{Y} \subseteq \mathcal{Y}$ the predicted label set. This, on the other hand, represents whether a certain autoimmune disease is predicted for a certain patient.

$$\hat{y}_{\bar{q}}(l) \quad , l \in \mathcal{Y} = \quad \begin{cases} 1 & if\ l \in \hat{Y} \\ 0 & otherwise \end{cases}$$

Next the number of diagnosed diseases for a given patient can be defined as:

$$C_{\bar{q}} = \sum_{\forall l \in \mathcal{Y}} y_{\bar{q}}(l)$$

Finally four functions, $tp_{\bar{q}}(l)$, $tn_{\bar{q}}(l)$ $fp_{\bar{q}}(l)$ and $fn_{\bar{q}}(l)$, can be defined that describe whether a prediction for a certain label is correct or not:

$$tp_{\bar{q}}(l) = \begin{cases} 1, & \text{if } \hat{y}_{\bar{q}}(l) = y_{\bar{q}}(l) = 1 \\ 0, & otherwise \end{cases} \quad \forall \bar{q} \in \bar{Q}, \forall l \in \mathcal{Y}$$

$$tn_{\bar{q}}(l) = \begin{cases} 1, & \text{if } \hat{y}_{\bar{q}}(l) = y_{\bar{q}}(l) = 0 \\ 0, & otherwise \end{cases} \quad \forall \bar{q} \in \bar{Q}, \forall l \in \mathcal{Y}$$

$$fp_{\bar{q}}(l) = \begin{cases} 1, & \text{if } \hat{y}_{\bar{q}}(l) = 1 \neq y_{\bar{q}}(l) \\ 0, & otherwise \end{cases} \quad \forall \bar{q} \in \bar{Q}, \forall l \in \mathcal{Y}$$

$$fn_{\bar{q}}(l) = \begin{cases} 1, & \text{if } \hat{y}_{\bar{q}}(l) = 0 \neq y_{\bar{q}}(l) \\ 0, & otherwise \end{cases} \quad \forall \bar{q} \in \bar{Q}, \forall l \in \mathcal{Y}$$

The following performance measures are widely used to evaluate how well a model works. These are based on a confusion matrix for every disease separately, an example of such a confusion matrix is depicted in figure 14. The following four cases can be distinguished:

- True Positives (TP): True positives are the cases where the actual class of the data point was True and the predicted is also True.
  E.g.: A person is diagnosed with lupus and the model classifies his case as having lupus.
- True Negatives (TN): True negatives are the cases where the actual class of the data point was False and the predicted is also False.
  E.g.: A person doest not have lupus and the model classifies his case as having no lupus.
- False Positives (FP): False positives are the cases where the actual class of the data point was False and the predicted is True.
  E.g.: A person does not have lupus and the model classifies his case as having lupus.
- False Negatives (FN): False negatives are the cases when the actual class of the data point was True and the predicted is False.
  E.g.: A person is diagnosed with lupus and the model classifies his case as having no lupus.

Table 14: Confusion matrix

		Actual value	
		Positive	**Negative**
Prediction outcome	**Positive**	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Accuracy, precision and recall are the most common performance measures. In this case accuracy should not be used, since the target variable classes are unbalanced.
E.g.: In our lupus detection example with 100 people, only 5 people have lupus. Lets say our model is very bad and predicts every case as No lupus. In doing so, it has classified those 95 non-lupus patients correctly and 5 lupus patients as Non-lupus. Now even though the model is terrible at predicting lupus, The accuracy of such a bad model is 95%.

The precision represents how well a model predicts a certain disease to be True. A

high precision is desired.
E.g.: out of all the people that were predicted to have lupus, how many actually had it? High precision makes sure not many people are told to have lupus, when actually they do not.

$$Precision(l) = \frac{TP(l)}{TP(l) + FP(l)} = \frac{\sum_{\bar{q}\in\bar{Q}} tp_{\bar{q}}(l)}{\sum_{\bar{q}\in\bar{Q}} tp_{\bar{q}}(l) + fp_{\bar{q}}(l)} \qquad , l \in \mathcal{Y}$$

The recall on the other hand gives the fraction of cases where a disease is predicted correctly divided by the times that prediction is missed for a certain disease.
E.g.: Out of all the people who actually have lupus, how many did the model identify? High recall makes sure the model does not fail to spot many people who have lupus.

$$Recall(l) = \frac{TP(l)}{TP(l) + FN(l)} = \frac{\sum_{\bar{q}\in\bar{Q}} tp_{\bar{q}}(l)}{\sum_{\bar{q}\in\bar{Q}} tp_{\bar{q}}(l) + fn_{\bar{q}}(l)} \qquad , l \in \mathcal{Y}$$

Since both are found very important, the F-measure is calculated, which is a weighted harmonic mean of the precision and recall.

$$F - measure(l) = 2 \cdot \frac{precision(l) \cdot recall(l)}{precision(l) + recall(l)} \qquad , l \in \mathcal{Y}$$

Since these performance measures have to be calculated for every class, a macro-average precision and recall can be calculated as a combination for all classes. The macro-average is used for computational reasons and the fact that macro-average gives equal importance to each class, whereas micro-average give equal importance to each sample. This means that a micro-average performance measure will favor majority classes, like the accuracy when regarding the F-measure. These can be calculated as follows:

$$Precision = \frac{\sum_{l\in\mathcal{Y}} precision(l)}{m} \qquad \in [0,1] \tag{13}$$

$$Recall = \frac{\sum_{l\in\mathcal{Y}} recall(l)}{m} \qquad \in [0,1] \tag{14}$$

$$F - measure = \frac{precision \cdot recall}{precision + recall} \qquad \in [0,1] \tag{15}$$

Finally, the percentages of cases in which at least one disease is predicted correctly is calculated:

$$Score = \frac{\sum_{\bar{q}\in\bar{Q}} \frac{\sum_{l\in\mathcal{Y}} tp_{\bar{q}}(l)\cdot 100}{\sum_{l\in\mathcal{Y}} tp_{\bar{q}}(l)}}{n} \qquad \in [0,100] \tag{16}$$

All four performance measures need to be maximized in order to optimize the model. During training of the model and tuning of the parameters the focus lays on finding the maximum F-measure.

4 Experimental Setup

To do a righteous evaluation and prevent overfitting, the dataset is split into a train-, validation- and test set of 60%, 20% and 20% respectively. Each user id is sampled randomly, with the distribution of having "no autoimmune disease / miscellaneous autoimmune disease" equally distributed among the sets. Table 15 describes the distribution of patients within the tree sets. Figure 13 gives the distribution between the tree different sets for every autoimmune disease that has to be predicted. It shows that the sets are evenly distributed for the response variables.

Table 15: The distribution of patients between the train-, validation- and test set.

	Training set	Test set	Validation set	Total
%	60	20	20	100
Patients	2.062	687	687	3.436
From within 16ADs dataset	1.777	592	593	2.962
From other ADs dataset	285	95	94	474

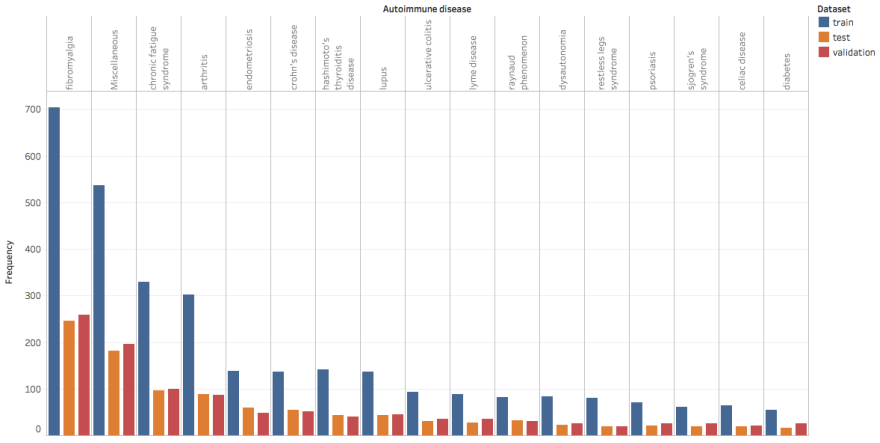

Fig. 13: The distribution of ADs between the train-, validation- and test set.

In order to use the limited data in the best way possible, K-Fold cross validation is used to detect and prevent overfitting. The train set is split into K, non overlapping, folds. Now the $K-1$ chunks of data are used to train, and the last, remaining part is used to test on. The performance measures are calculated and then averaged over all K different configurations. Here, 5-Fold cross validation is used, for computational reasons, during the training of each of the models. Parameter tuning is done for the

5-Fold cross validation set, in order to optimize the performance measures.

The optimal parameter setting is selected based on the two maximal F-measures for all nine combinations of methods and algorithms. Based on this parameter setting, the model is trained and then tested on the validation set. Finally, to make best use of the limited data, the train and validation set are combined in the end to train the model that is tested on the test set. All evaluation measures for the cross validation set, the validation- and test set are compared.
For the average performance measures of the 5-Fold cross validation, validation- and test set performances should not differ more than 10% from each other. If this were to be the case, the parameter setting or the dataset itself could be biased which leads to a model that might be over- or underfitting.

No current published existing methods exist that predict this problem. So a set of four benchmarks are made, which indicate the performance measures for naive solutions. The following benchmarks are made with 100 simulations, and the performance measures are averaged. The first benchmark is most naive, and the successors take more knowledge of the data into account. The results of these benchmarks are depicted in section 5.

1. The number of diseases for a certain patient are randomly sampled between the minimum and maximum number of labels a patients has; $sample_1 \in [1, 8]$, as shown in figure 8, B. Next, a bootstrap sample of all 17 different labels is done of size $sample_1$
2. From the set that is used to train the model, the most frequently occurring disease is used as a prediction for every patient.
3. The number of diseases for a certain patient are randomly sampled between the minimum and maximum number of labels a patients has; $sample_1 \in [1, 8]$. From the set that is used to train the model, the distribution of autoimmune diseases is deducted; $distribution_{trainset}$. Finally, a bootstrap sample of size $sample_1$ is done, following $distribution_{trainset}$.
4. From the set that is used to train the model, the average number of disease patients have is deducted; $Av\_diseaseNumber_{trainset}$, as well as the distribution of diseases occurring; $distribution_{trainset}$. Since $Av\_diseaseNumber_{trainset}$ is an averaged number, it will not be an integer. With equal probability it is decided whether to use the smallest or largest following integer from $Av\_diseaseNumber_{trainset}$; $sample_2$. Finally, a bootstrap sample of size $sample_2$ is done, following $distribution_{trainset}$.

4.1 Feature engineering and selection

All explanatory variables are boolean values. They indicate the presence or absence of some categorical variable. So feature engineering is deemed unnecessary for this research. Feature selection is not used either, because no assumptions can be made on the importance of symptoms or conditions, since the possible label set is this big.

4.2 Parameter selection

Parameter tuning is necessary to optimize the model and get the highest performance measures. For each of the nine models, a set of parameters is tested according to a grid

search. Below the different parameter values per algorithm that are tested for each possible combination of parameters are described.
For every model the computational cost is weighed against having enough models to compare.

4.2.0.1 k-Nearest Neighbors

The kNN algorithm uses tree variable parameters. Namely, the distance function to determine similar patients, the K-parameter that determine how many similar patients are regarded and the threshold function that decide how to predict which diseases to adopt from the similar patients.
The threshold function parameters are decided by trial and error for every method.
Problem Transformation method
Label Powerset method

$$\begin{aligned} Distance\ function &\in [Euclidean, Jaccard, Pearson] \\ k\ parameter &\in [1, 3, 5, 7, 10] \\ Threshold\ function &\in [1, 2] \end{aligned}$$

A rule of thumb for choosing the value of k is said to be evaluated starting from one to the square root of the number of observations.[17] Furthermore, keeping the value of k odd, prevents having a tie between classes when classifying. This is too computationally expensive, and the label power set will become too big, so using trial and error it is decided to have a maximum k of ten.
The threshold function used in the Label Powerset method decides which diseases to use as prediction. The label sets for the k neighbors are combined to one large set, and the diseases included in the prediction are the ones that occur at least x times in the combined set, where x equals the threshold. If the K-parameter equals 1, the entire label set of the nearest neighbor is used. If the threshold function equals 1, the unique combination of the k nearest neighbors' diseases is used as the prediction.

Binary Relevance method

$$\begin{aligned} Distance\ function &\in [Euclidean] \\ k\ parameter &\in [1, 3, 5, 7, 10, 12, 15, 20, 25, 30, 35, 40] \\ Threshold\ function &\in [0.3, 0.4, 0.5] \end{aligned}$$

The distance function available in the used package is limited to the Euclidean distance. Because of this, more k parameters are added as opposed to the Label Powerset method, making a grid search according to the rule of thumb.[17]
The threshold function used in the Binary Relevance Method decides which diseases to include in the prediction. For every label, the algorithm decides the probability of it being fitting to the model. A disease is included in the prediction if the probability is at least equal to x, where x equals the threshold.

Algorithm Adaptation method

$$\begin{aligned} Distance\ function &\in [Euclidean] \\ k\ parameter &\in [1,3,5,7,10,12,15,20,25,30,35,40] \\ Threshold\ function &\in [0.1,0.2,0.25,0.3,0.35] \end{aligned}$$

Like in the Binary Relevance method, only the Euclidean distance is available. The Binary Relevance method is a computational expensive method, since a chain of classifiers is used. The Algorithm Adaptation method on the other hand is less computational costly, and so an extra number of threshold parameters are tested.
Next to the parameters mentioned above, there is a parameter s, which controls the strength of the uniform prior. This smoothing parameter s is set to 1, which yields the Laplace smoothing.
The threshold function used in the Algorithm Adaptation Method decides which diseases to include in the prediction. For every label, the algorithm decides the probability of it being fitting to the model. A disease is included in the prediction if the probability is at least equal to x, where x equals the threshold.

4.2.0.2 Random Forest

The Random Forest algorithm uses two parameters. Namely, the number of trees to grow in every forest; the $Ntree$ parameter and the number of explanatory variables to sample as a possible split; the $Mtry$ parameter.
In general the rule of thumb for the $Ntree$ parameters is that the more trees are grown, the better the performances. However, this is computationally expensive, and after a certain number of trees grown, improvement is negligible.[26] Both the Ntree and Mtry parameters are chosen based on trial and error. The package used to implement the Random Forest has a default of $Mtry = \sqrt{\#of\ eplanatory\ variables}$. The other Mtry parameters surrounding it are explored.

Problem Transformation method

Label Powerset method

$$\begin{aligned} Ntree\ parameter &\in [250,500,1000] \\ Mtry\ parameter &\in [15,\sqrt{\#explanatory\ variables},30,60,100] \end{aligned}$$

Binary Relevance method

$$\begin{aligned} Ntree\ parameter &\in [250] \\ Mtry\ parameter &\in [15,\sqrt{\#explanatory\ variables},30,60,100] \\ Treshold\ function &\in [0.3,0.4,0.5] \end{aligned}$$

The Ntree parameter is set only to 250 for computational reasons and the fact that the Binary Relevance method uses a chain of classifiers. This means 250 trees are grown for every possible label.
The threshold function used in the Algorithm Adaptation Method decides which diseases to include in the prediction. For every label, the algorithm decides the probability of it being fitting to the model. A disease is included in the prediction

if the probability is at least equal to x, where x equals the threshold. The possibilities for this parameters are based on trial and error.

Algorithm Adaptation method

$$\begin{aligned} Ntree\ parameter &\in [100, 250, 500, 1000] \\ Mtry\ parameter &\in [15, \sqrt{\#explanatory\ variables}, 30, 60, 100, 250, 500] \end{aligned}$$

The Algorithm Adaptation method is not computational expensive and so an extra number of parameters to test have been added.

4.2.0.3 Support Vector Machine

The threshold function parameters are decided by trial and error for every method. In case of the Linear Kernel function, no γ parameter is needed.

Problem Transformation method

Label Powerset method

$$\begin{aligned} Kernel\ function &\in [Linear, Radial] \\ \gamma\ parameter &\in [2^{-\#explanatory\ variables}, \\ &\quad 2^{-\frac{\#explanatory\ variables}{2}}, \ , 2^{-100}, 2^{-10}, \\ &\quad \frac{1}{\#explanatory\ variables}, \frac{1}{\#explanatory\ variables/_2}, \ 1] \\ Cost\ parameter &\in [0.1, 1, 10, 100, 500, 1000, 10000] \end{aligned}$$

It is found that trying exponentially growing sequences for the Cost and γ parameter is a practical method to identify good parameters.[18]

Binary Relevance method

$$\begin{aligned} Kernel\ function &\in [Linear, Radial] \\ \gamma\ parameter &\in 2^{-10}, \frac{1}{\#explanatory\ variables}, \\ &\quad \frac{1}{\#explanatory\ variables/_2}, \ 1] \\ Cost\ parameter &\in [0.1, 1, 10, 100, 500, 1000, 10000] \\ Threshold\ function &\in [0.3, 0.4, 0.5] \end{aligned}$$

The Binary Relevance method is very computationally expensive, for this reason, less possibilities are tested for the γ parameter.

Algorithm Adaptation method

$$
\begin{aligned}
Kernel\ function &\in [Linear, Radial] \\
\gamma\ parameter &\in 2^{-10}, \frac{1}{\#explanatory\ variables}, \\
&\quad \frac{1}{\#explanatory\ variables/_2},\ 1] \\
Cost\ parameter &\in [0.1, 1, 10, 25, 50, 100, 500, 1000, 10000] \\
Threshold\ function &\in [0.1, 0.2, 0.3]
\end{aligned}
$$

Since not all γ parameters used in the Label Powerset method are eligible in the implementation of the Algorithm Adaptation method, only the valid ones are used. For this reason, an extra number of Cost parameter possibilities are added to review.

5 Results and evaluation

5.1 Parameter settings

All results resulting from the 5-fold cross validation are stated in appendix B. For the optimized F-measures, the two best parameter settings per algorithm and per method are distinguished. The model with these settings was trained on the training set and validated on the validation set. Finally, the combined train and validation set were used to train, and this model was tested on the test set. All results are shown in appendix C. The chosen parameter settings result in 18 different models, that are stated in table 16. It shows that the parameter pairs differ enormously depending on the model.

Table 16: Final model parameter settings

	Parameter setting			
	Distance function	k		Threshold
KNN LP 1	Euclidean	1		
KNN LP 2	Jaccard	1		
KNN BR 1	Euclidean	7		0.4
KNN BR 2	Euclidean	12		0.4
KNN AA 1	Euclidean	15		0.25
KNN AA 2	Euclidean	25		0.25
	Ntree	Mtry		Threshold
RF LP 1	500	60		
RF LP 2	1000	30		
RF BR 1	250	$\sqrt{\#explanatory\ variables}$		0.3
RF BR 2	250	60		0.4
RF AA 1	100	500		
RF AA 2	500	500		
	Kernel	γ	Cost	Threshold
SVM LP 1	Radial	2^{-10}	100	
SVM LP 2	Radial	$1/\#\ explanatory\ variables$	100	
SVM BR 1	Radial	$1/\#\ explanatory\ variables/2$	10	0.3
SVM BR 2	Radial	2^{-10}	100	0.3
SVM AA 1	Radial	2^{-10}	10	0.2
SVM AA 2	Radial	$1/\#\ explanatory\ variables/2$	10	0.2

Table 17: Final F-measures performances

	5-fold CV set	Validation set	Test set	Percentage
Benchmark 1	5.87	5.9	5.91	23.2
Benchmark 2	3.32	3.37	3.37	64.19
Benchmark 3	5.87	5.84	5.94	22.97
Benchmark 4	5.83	5.9	5.82	8.58
KNN LP 1	11.53	9.57	9.48	32.6
KNN LP 2	13.84	10.87	11.12	37.12
KNN BR 1	14.44	12.83	12.89	35.37
KNN BR 2	14.18	14.78	13.58	36.83
KNN AA 1	11.77	11.53	11.37	40.9
KNN AA 2	11.79	11.29	10.46	42.21
RF LP 1	17.44	13.96	13.48	42.63
RF LP 2	16.53	16.14	14.22	45.27
RF BR 1	19.64	16.19	16.73	44.69
RF BR 2	19.66	16.99	16.71	46.29
RF AA 1	14.57	9.49	11.06	36.1
RF AA 2	15.11	9.66	11.7	36.24
SVM LP 1	15.61	13.13	16.86	43.23
SVM LP 2	15.34	14.74	17.89	43.67
SVM BR 1	19.27	19.85	17.7	47.16
SVM BR 2	19.41	20.65	18.67	47.6
SVM AA 1	18.44	17.96	15.32	42.07
SVM AA 2	18.41	18.48	15.72	41.63

5.2 Performance

Table 17 shows the F-measures of all chosen models. Furthermore, it shows the percentages of cases in which at least one disease is predicted correctly for the test set. It shows that there are quite obvious differences between performances in different models, even though they have all been optimized.
Figure 14 illustrates the F-measures of these models and the benchmark, sorted by algorithm. The blue shapes represent the 5-Fold cross validation set, red are the performances for the validation set and orange represents the F-measures for the test set of a given model. The square shapes represent a model using the Label Powerset method, the circles have a Binary Relevance method as underlying model method and the crosses represent that Algorithm Adaptation methods are used.

Figure 15 shows the same measures, but sorted by method. The colors represent the datasets in the same way as in figure 14. But now circles represent a model using the k-Nearest Neighbors algorithm, squares represent the Random Forest and crosses imply

a Support Vector Machine being used.

When comparing the F-measures between the different sets in table 17, and figures 14 and 15 it seems that all models of the Random Forest are over- or under- fitting, as do all Label Powerset methods, and so are not reliable.

From both table 17 and figures 14 and 15, it is clear that all models outperform the four benchmarks, where all Benchmarks outperform Benchmark 2. Furthermore, when comparing the algorithms, from figure 14 it seems that the k-Nearest Neighbor is outperformed by the Random Forest, and the Random Forest on his turn is outperformed by the Support Vector Machine.
From figure 15 a comparison between the methods can be deducted. This comparison is less clear, but it seems that the Binary Relevance method on average performs best, followed by the Algorithm Adaptation method and finally the Label Powerset Method.

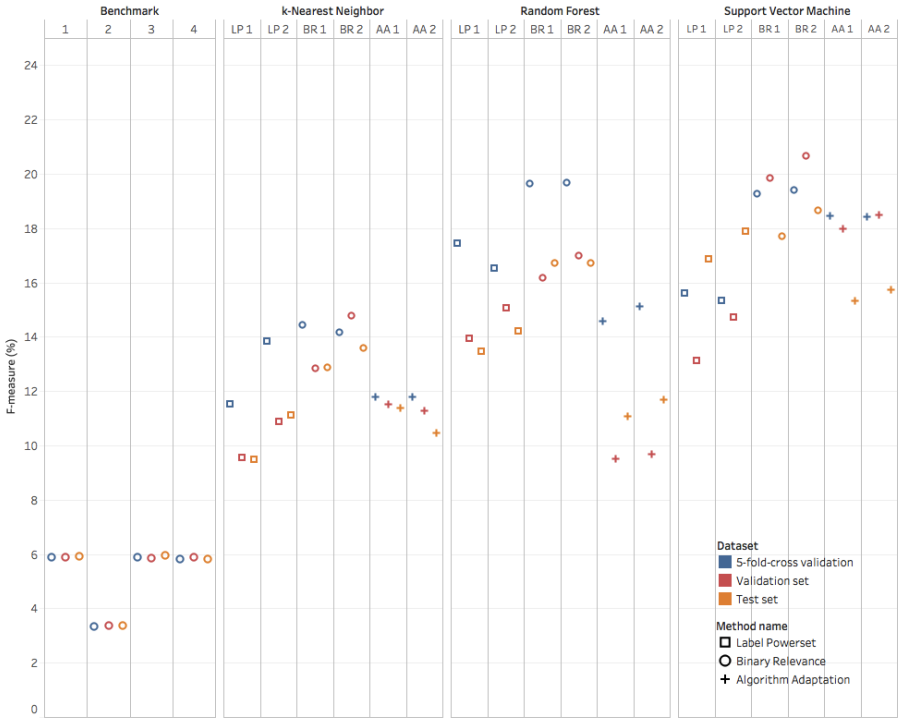

Fig. 14: Final F-measure performances sorted by algorithm

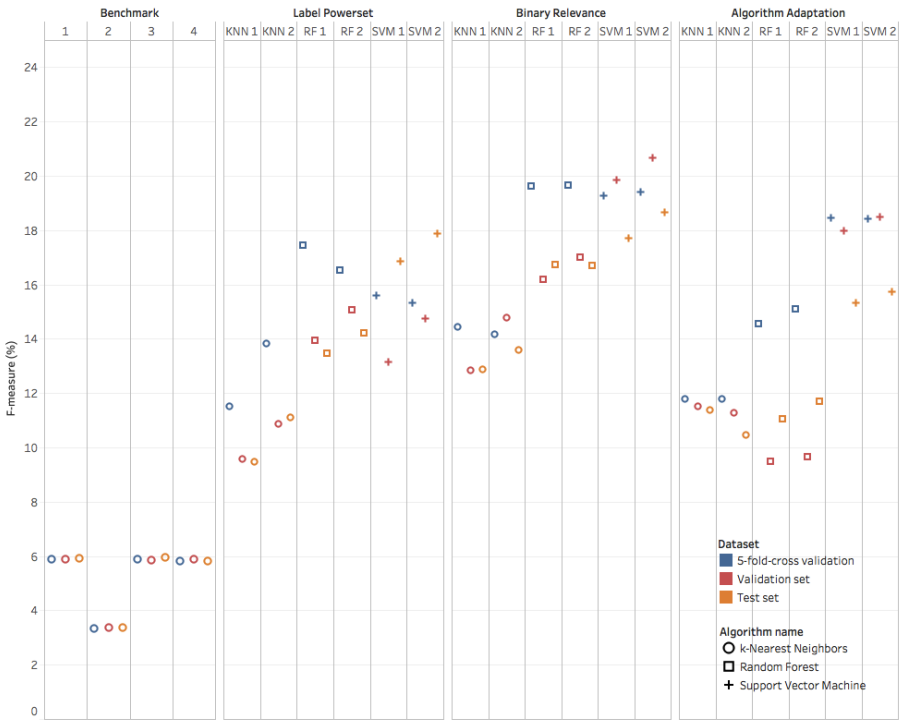

Fig. 15: Final F-measure performances sorted by method

5.3 Reliability of the models

To test the hypothesis of reliability of the model performances, confidence intervals can be made. The confidence interval shows what the performance measure is in 95% of the cases. Now, using bootstrapping, a sample of 500 F-measures of patients are drawn and the mean F-measure is calculated. This process is repeated 1,000 times. This process results in 1,000 mean F-measure values for which the 2.5% and 97.5% quantiles result in the confidence intervals described in table 18

Table 18: Confidence intervals of the models

	95% Confidence interval
KNN LP 1	[7.71, 11.49]
KNN LP 2	[9.08, 13.17]
KNN BR 1	[10.43, 15.79]
KNN BR 2	[10.37, 16.95]
KNN AA 1	[6.98, 12.32]
KNN AA 2	[8.24, 12.76]

	95% Confidence interval
RF LP 1	[11.28, 15.75]
RF LP 2	[11.5, 16.85]
RF BR 1	[12.33, 18.86]
RF BR 2	[12.6,19.36]
RF AA 1	[8.68, 13.19]
RF AA 2	[9.25, 13.98]

	95% Confidence interval
SVM LP 1	[12.91, 19.35]
SVM LP 2	[13.45, 20.67]
SVM BR 1	[14.83, 21.03]
SVM BR 2	[14.4, 20.78]
SVM AA 1	[12.42, 18.53]
SVM AA 2	[12.79, 19.0]

The confidence intervals mentioned in table 18 support the expectations based on the visualizations above. The k-Nearest Neigbor models based on the Label Powerset method are not deemed reliable, as well as all Random Forest Models, except for the Label Powerset model 2. So even though the parameters are optimized based on the 5-Fold cross validation, the models still overfit. The five models will be disregarded, because not only are the performance measures found, deemed unreliable, they do not outperform other models either.

The fact that the Label Powerset is over- or under-fits is not surprising. Many of the label sets necessary in the classification of both the validation and test set, are unknown to the model at the moment of training. The fact that the Random Forest over- or under- fits is surprising, although every machine learning algorithm with high complexity can over- or under-fit. But in general an ensemble method such as the Random Forest will deduce the likelihood of this happening.

5.4 Model Comparison

After the reliability check in subsection 5.3, 11 models are left to compare to each other as well as to the four benchmarks. Since the same group of patients is used, for all models, the paired T-test can be used. From this test p-values can be retrieved, that indicate whether there is indeed a significant difference between the F-measure of the different models. If the p-value is smaller than 0.05, the null hypothesis is rejected and thus the models are found to be significantly different. The p-values for every combination are found in appendix C.
As expected, there is indeed a significant difference between the models and the benchmarks. All models outperform the benchmarks, and as expected, benchmark 2 is significantly worse than the other benchmarks. But no significant difference was found between benchmarks 1, 3 and 4.
The SVM Binary Relevance models significantly outperform all other models, except

for the Support Vector Machine models with Algorithm Adaption underlying methods. The SVM BR, model 1 does outperform the SVM AA, model 2. There is no significant difference between the SVM Binary Relevance models themselves.
The Algorithm Adaptation method in combination with the Support Vector Machine performs better than the k-Nearest Neighbors Algorithm Adaptation method and in most cases with the Binary Relevance method as well. The only one that is not significantly better is the SVM AA, model 1 in comparison to the KNN BR, model 2. They do however, not outperform the Random Forest with Label Powerset method significantly.
There is no significant difference between the performance measures of the Label Powerset methods, regardless of the underlying algorithm. There are significant difference for both the Algorithm Adaptation and Binary Relevance Methods, depending on the algorithm.
The Binary Relevance Method, on average, outperforms the other methods significantly, as expected and deducted from the figures.

Overall, the F-measure values found are not high, and so none of the models seems to work very well.

6 Conclusion

As a conclusion there is not one model that outperforms all other significantly. In fact, all F-measures are low, and none perform well. But the Support Vector Machine with the Binary Relevance method; model 1, overall performs best. As parameter settings this model uses a Radial kernel, a γ-parameter of 2^{-10}, a cost parameter of 10 and a threshold of 0.3.

Although, looking at the F-measures of the model, this model does not perform well with a 95% confidence interval of the F-measure of [14.83, 21.03]. For the test set it does predict at least one of the autoimmune diseases correctly in 47.16% of the cases. This means that for almost half of the patients at least one disease can be diagnosed immediately, in stead of having to wait multiple years and visit several doctors before getting a diagnosis. Benchmark 2 does retain an even higher percentage of cases of 64.19% in which at least one autoimmune disease is predicted correctly. But the F-measure is a significantly lower. This is because in Benchmark 2, only one disease is predicted, the most common autoimmune disease.

7 Discussion

The final performance measures found in this research were not high. This could be due to a number of reasons.
The first is the fact that many autoimmune diseases have similar characteristics, which makes diagnosing very difficult, and so predicting based on machine learning as well. Furthermore, As mentioned before the severity and the and the clinical picture strongly depend on the patient, which makes an autoimmune disease different for every person.

Other reasons might have to do with design decisions taken in this research. For example, including the Miscellaneous category in the trackable_type AD, might make the model unnecessarily complex. Furthermore, combining different names of trackable_type Condition and Symptom into one, might have removed important specifics for autoimmune diseases. Lastly, the decision was made to not take regular suffering from certain symptoms and conditions into account. This means, if a user logged having a headache in the application ones, it is handled the same as when a user has headaches every day.

For further research one could try and deviate from these design choices. Improvements are possible in a number of different ways as well.
Gathering and saving the data could be done in a more structured way. In stead of users being allowed to freely type their names, there could be a drop down window. In this case, the data would have been formatted in the same, clear, structured and concise way.
Furthermore, medical professionals could be involved, filtering out the side effects from treatments and other conditions. This means symptoms due to treatments or other conditions are removed and only symptoms due to the autoimmune disease remain.

Feature engineering or adding more features could also help boost performances. It was found that the gender, age and country of residence were found to be beyond the scope of this research, but they could add value.
If a patient has already been diagnosed with one autoimmune disease, the co-occurrence of other diseases is warranted. The dataset shows that the combination of fibromyalgia and chronic fatigue syndrome occurs very often, as well as fibromyalgia in combination with arthritis.
It has also been shown that some autoimmune diseases are more common in certain ethnic groups.[2] This could be added as an explanatory variable.
The same reasoning goes for an explanatory variable of having someone in a patients family with an autoimmune disease. Some autoimmune diseases are stated to run in families. Not every family member will necessarily have the disease, but they inherit a susceptibility to an Autoimmune condition.[2]

The kNN might not have been a good choice because of the dimensionality. With so many dimensions any data point is actually closer to the border of the feature space than to any other point. If a larger dataset is available, this problem would be overcome. Another possible solution would be to use feature selection, and so reduce the dimensionality. As shown in figure 7, B symptoms occur across all autoimmune diseases and so a medical background would be useful in feature selection, to disregard a set of

symptoms.

One could also look at other classifier algorithms. In this research three well known, and often implemented algorithms have been used. But the use of Bayesian Networks could also boost performances.[13] It has been said that they are one of the most most successful tools for medical diagnostics.[24]

Finally, a last possible method could be the use of ensemble methods. Each classifier in an ensemble method is based on a Problem Transformation- or Algorithm Adaptation method or a combination of both.[36][32] They could also combine algorithms, and use either boosting or bagging to improve on performances.

References

1. American Autoimmune Related Disease Association, Inc. *Autoimmune Disease List* . AARDA website. Consulted on July 5, 2018 on `https://www.aarda.org/diseaselist/`
2. American Autoimmune Related Disease Association, Inc. *Autoimmune Disease List* . AARDA website. Consulted on Oktober 18, 2018 on `https://www.aarda.org/who-we-help/patients/women-and-autoimmunity/#1481567809996-dfed32c6-2f25`
3. Autoimmune Registry, Inc. *List of Autoimmune Diseases.* ARI website. Consulted on July 5, 2018 on `http://www.autoimmuneregistry.org/the-list-1/`
4. Biau, G. (2012). *Analysis of a Random Forests Model* Journal of Machine Learning Research 13 (2012) 1063-1095
5. Bhatia, S., Prakash, P., & Pillai, G.N. (2008). *SVM based Decision Support System for Heart Disease Classification with Integercoded Genetic Algorithm to select critical features* Proceedings of the World Congress on Engineering and Computer Science, San Francisco, USA, pp.34-38, 2008.
6. Boutell, M.R., Luo, J., Shen, X., & Brown, C.M. (2004). *Learning multi-label scene classification.* Pattern Recognition, 37(9):17571771.
7. Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984).*Classification and regression trees.* Wadsworth Books, 358.
8. Breiman, L. (2001), Random Forests, Machine Learning 45(1), 5-32.
9. Chou, K.C. (2013). *Some remarks on predicting multi-label attributes in molecular biosystems* Molecular Biosystems, vol. 9, no. 6, pp. 10921100, 2013.
10. Clare, A., & King, R.D. (2001). *Knowledge discovery in multi-label phenotype data.* In Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery, pages 4253, London, UK, 2001. Springer-Verlag
11. Cojocaru, M. Cojocaru, I.M., & Silosi, I. (2010). *Multiple autoimmune syndrome* Maedica (Buchar). 2010 Apr; 5(2): 132134.
12. Cortes, C., & Vapnik, V. (1995). *Support-Vector Networks*, Machine Learning, 20, pp.273-297
13. Domingos, P. (2015). *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*
14. Dox, I.G., Melloni, B.J., Eisner, G.M., Melloni, J.L. (2002). *Melloni's illustrated medical dictionary* (4th ed.), The Parthenon Publishing group, ISBN: 1-85070-094-X.
15. Featured Kaggle Dataset (2017). *Flaredown Autoimmune Symptom Tracker, version 3. Flaredown user data as of June 23, 2017.* Kaggle website. Consulted on September 8, 2017 on `https://www.kaggle.com/flaredown/flaredown-autoimmune-symptom-tracker/data`
16. Flach, P. (2012). *Machine Learning - The Art and Science of Algorithms that Make Sense of Data*
17. Hassanat, A.B. Abbadi, M.A. & Altarawneh , G.A. (2014). *Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach* (IJCSIS) International Journal of Computer Science and Information Security, Vol. 12, No. 8, August 2014
18. Hsu, C., Chang, C., & Lin, C. (2013). *A Practical Guide to Support Vector Classification*
19. Hughes, P.A., Zola, H., Penttila, I.A., et al. (2013). *Immune activation in irritable bowel syndrome: can neuroimmune interactions explain symptoms?* Am J Gastroenterol. 2013;108:1066-1074.

20. Lerner, A., Jeremias, P., & Matthias, T. (2015). *The World Incidence and Prevalence of Autoimmune Diseases is Increasing* International Journal of Celiac Disease. 2015, 3(4), 151-155 doi:10.12691/ijcd-3-4-8
21. Joachims, T. (2002). *Optimizing Search Engines Using Clickthrough Data.* Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM.
22. Manzel, A., Muller, D.N., Hafler, D.A., Erdman, S.E., Linker, R.A., & Kleinewietfeld, M. (2014). *Role of "Western diet" in inflammatory autoimmune diseases.* Curr Allergy Asthma Rep. 2014 Jan;14(1):404. doi: 10.1007/s11882-013-0404-6.
23. Medtronic MiniMed. (2014) *Continuous glucose monitoring* Consulted on October 15, 2018 at `http://www.medtronicdiabetes.com/treatments/continuous-glucose-monitoring`
24. Nikovski, D. (2000). *Constructing Bayesian Networks for Medical Diagnosis from Incomplete and Partially Correct Statistics* IEEE Transactions on Knowledge and Data Engineering (Volume: 12 , Issue: 4 , Jul/Aug 2000)
25. Office on Womens Health, U.S. Department of Health and Human Services. (2001). *Womens Health Issues: An Overview* Consulted on October 27, 2018 on `http://www.commed.vcu.edu/IntroPH/MCH/Office_on_Women's_Health.pdf`
26. Oshiro, T.M., Perez, P.S., & Baranauskas, J.A. (2012). *How Many Trees in a Random Forest?*Conference: 8th International Conference on Machine Learning and Data Mining in Pattern Recognition, MLDM 2012, Lecture Notes in Computer ScienceVolume: 7376
27. Parsian, M. (2015). *Data algorithms: Recipes for scaling up with Hadoop and Spark.* 309.
28. Pauly, O. (2012). *Random Forests for Medical Applications* Ph.D. Thesis for the Technical University München.
29. Probst, P., Au, Q., Casalicchio, G., Stachl, C., & Bischl, B. (2017). *Multilabel Classification with R Package mlr.* arXiv preprint arXiv:1703.08991.
30. Read, J. (2008). *A Pruned Problem Transformation Method for Multi-label classification.* In Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008), pages 143150, 2008.
31. Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2009). *Classifier Chains for Multi-label Classification.* Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, 5782, 254-269.
32. Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). *Classifier chains for multi-label classification.* Machine Learning, 85(3), 333-359.
33. Rook, G.A.W. (2012). *Hygiene Hypothesis and Autoimmune Diseases* Clinical Reviews in Allergy & Immunology February 2012, Volume 42, Issue 1, pp 515
34. Sarkar, M., & Leong, T.Y. (2000). *Application of K-Nearest Neighbors Algorithm on Breast Cancer Diagnosis Problem.* Proceedings of the AMIA Symposium: 759763.
35. Tsoumakas, G., Katakis, I., & Vlahavas, I. (2009). *Mining multi-label data.* Data Mining and Knowledge Discovery Handbook, pages 119, 2009.
36. Tsoumakas, G., & Vlahavas, I. (2007). *An ensemble method for multilabel classification.* Random k-labelsets: In J. N. Kok, J. Koronacki, R. L. Mantaras, S. Matwin, D. Mladenič, and A. Skowron, editors, Machine Learning: ECML 2007, volume 4701 of Lecture Notes in Computer Sci- ence, pages 406417. Springer Berlin Heidelberg, 2007.
37. Villani, A.C., Lemire, M., Thabane, M,, et al. (2010). *Genetic risk factors for post-infectious irritable bowel syndrome following a waterborne outbreak of gastroenteritis.* Gastroenterology. 2010;138:1502-1513.

38. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., & Blockeel, H. (2008). *Decision trees for hierarchical multi-label classification.* Machine Learning, 73(2):185214, 2008.
39. Walsh, SJ, LM. (2000). *Autoimmune Diseases: A Leading Cause of Death among Young and Middle-Aged Women in the United States.* American Journal of Public Health. 2000;90:1463-1465
40. Watson, S. Medically reviewed by Daniel Murrell,D. MD. (2017). *Autoimmune Diseases: Types, Symptoms, Causes and More.* Consulted on October 18, 2018 on `https://www.healthline.com/health/autoimmune-disorders`
41. Willemse E. (2015). *Beter?! Toekomstbeelden van technologie in de zorg* Stichting toekomstbeeld der techniek. isbn 978 94 913 97 110.
42. World Health Organization. (2018). *Life expectancy and Healthy life expectancy. Data by WHO region* Consulted on October 27, 2018 on `http://apps.who.int/gho/data/view.main.SDG2016LEXREGv?lang=en`
43. Xiao, X., Wang, P., Lin, W.Z., Jia, J.H., & Chou K.C. (2013). *iamp-2l: a two-level multi-label classifier for identifying antimicrobial peptides and their functional types* Analytical biochemistry, vol. 436, no. 2, pp. 168 177, 2013.
44. Xu, J. (2011). *An extended one-versus-rest support vector machine for multi- label classification.* Neurocomputing, 74(17):31143124.
45. Zhang, M.L. L., & Zhou, Z.H. H. (2007). *ML-KNN: A lazy learning approach to multi-label learning.* Pattern Recognition, 40(7), 2038-2048.

A Appendix

Conditions are changed in symptoms if the trackable_name contains any of the following words in any form:

- Ache
- Acne
- Activity
- Ankle
- Anxiety
- Arm
- Back
- Bloating
- Burn
- Cough
- Cramp
- Deficiency
- Depression
- Diarrhea
- Dizziness
- Ear
- Eye
- Face
- Fatigue
- Fear
- Fever
- Finger
- Fog
- Foot
- Hair
- Hand
- Head
- Hunger
- Insomnia
- Itch
- Joint
- Knee
- Leg
- Migraine
- Mood
- Nausea
- Neck
- Numb
- Pain
- Rash
- Sensitivity
- Skin
- Sore
- Spasm
- Stiffness
- Stress
- Sweat
- Swelling
- Tooth
- Ulcer
- Upset
- Urinating
- Weak

B Appendix

Table 19: KNN Label Powerset, performance measures 5-fold-cross validation

			5-fold-cross validation			
			Performance measures			
Distance function	k	Min Occurrence	Precision	Recall	F-measure	Percentage
Euclidean	1		0.1121	0.1198	0.1153	33.1241
Euclidean	3	1	0.0063	0.0101	0.0077	22.2633
Euclidean	3	2	6e-04	0.0062	0.0012	4.0746
Euclidean	5	1	0.0416	0.0417	0.0415	42.0984
Euclidean	5	2	0.0032	0.008	0.0046	18.1884
Euclidean	7	1	0.0989	0.0904	0.0941	54.8999
Euclidean	7	2	0.012	0.0161	0.0137	32.5426
Euclidean	10	1	0.1022	0.0954	0.0986	66.4406
Euclidean	10	2	0.0473	0.0468	0.0469	47.2862
Jaccard	1		0.1388	0.1382	0.1384	39.4754
Jaccard	3	1	0.0116	0.0152	0.0131	29.436
Jaccard	3	2	0.0011	0.007	0.0019	7.1289
Jaccard	5	1	0.066	0.057	0.0611	49.5647
Jaccard	5	2	0.0063	0.0106	0.0079	24.9756
Jaccard	7	1	0.1248	0.1081	0.1158	61.2997
Jaccard	7	2	0.0263	0.0234	0.0246	39.2843
Jaccard	10	1	0.1204	0.1079	0.1137	71.8705
Jaccard	10	2	0.0841	0.0662	0.0739	54.9477
Pearson	1		0.0965	0.1	0.0976	29.973
Pearson	3	1	0.0044	0.0089	0.0058	18.0425
Pearson	3	2	3e-04	0.0043	6e-04	2.9105
Pearson	5	1	0.0283	0.0344	0.031	37.9771
Pearson	5	2	0.0019	0.0068	0.003	13.096
Pearson	7	1	0.0746	0.0815	0.0775	50.3422
Pearson	7	2	0.0075	0.0125	0.0093	25.2685
Pearson	10	1	0.0876	0.0862	0.0856	63.4834
Pearson	10	2	0.0321	0.0405	0.0358	41.0778

Table 20: KNN Binary Relevance, performance measures 5-fold-cross validation

			5-fold-cross validation			
			Performance measures			
Distance function	k	Threshold	Precision	Recall	F-measure	Percentage
Euclidean	1	0.3	0.1265	0.1309	0.1285	35.1588
Euclidean	1	0.4	0.1331	0.1319	0.1322	32.3451
Euclidean	1	0.5	0.1513	0.1247	0.1364	31.6184
Euclidean	3	0.3	0.1266	0.1309	0.1285	35.3527
Euclidean	3	0.4	0.1316	0.1306	0.1308	32.248
Euclidean	3	0.5	0.1515	0.1252	0.1368	31.6669
Euclidean	5	0.3	0.1271	0.1317	0.1292	35.3529
Euclidean	5	0.4	0.1336	0.1321	0.1325	32.6361
Euclidean	5	0.5	0.1522	0.1256	0.1373	31.7638
Euclidean	7	0.3	0.1475	0.1335	0.1394	34.9653
Euclidean	7	0.4	0.1694	0.1297	0.1444	32.7832
Euclidean	7	0.5	0.1641	0.1184	0.1363	32.1036
Euclidean	10	0.3	0.1589	0.1299	0.1415	35.7896
Euclidean	10	0.4	0.1607	0.1202	0.1356	33.8014
Euclidean	10	0.5	0.1553	0.1093	0.1278	32.9773
Euclidean	12	0.3	0.1628	0.1274	0.1416	36.2263
Euclidean	12	0.4	0.1735	0.1225	0.1418	35.0138
Euclidean	12	0.5	0.1485	0.1091	0.125	33.5603
Euclidean	15	0.3	0.1583	0.1264	0.1383	36.5159
Euclidean	15	0.4	0.1474	0.12	0.1307	35.5466
Euclidean	15	0.5	0.1485	0.1077	0.1235	34.0931
Euclidean	20	0.3	0.1341	0.1238	0.1283	37.2923
Euclidean	20	0.4	0.1464	0.1185	0.1296	36.2746
Euclidean	20	0.5	0.1637	0.1093	0.1296	35.2568
Euclidean	25	0.3	0.1434	0.1253	0.1333	38.6518
Euclidean	25	0.4	0.1652	0.12	0.1381	37.1484
Euclidean	25	0.5	0.1704	0.1093	0.1315	36.033
Euclidean	30	0.3	0.1503	0.1229	0.135	38.7974
Euclidean	30	0.4	0.1641	0.1163	0.1344	37.0513
Euclidean	30	0.5	0.1544	0.1058	0.1246	35.9854
Euclidean	35	0.3	0.1558	0.1187	0.1342	38.6523
Euclidean	35	0.4	0.1757	0.1157	0.1387	37.4891
Euclidean	35	0.5	0.1515	0.1053	0.1232	35.9859
Euclidean	40	0.3	0.1683	0.1164	0.1369	38.6518
Euclidean	40	0.4	0.1848	0.113	0.1393	37.441
Euclidean	40	0.5	0.1497	0.1047	0.1224	36.665

Table 21: KNN Algorithm Adaptation, performance measures 5-fold-cross validation

			5-fold-cross validation			
			Performance measures			
Distance function	k	Threshold	Precision	Recall	F-measure	Percentage
Euclidean	1	0.1	0.0867	0.0756	0.0807	58.9698
Euclidean	1	0.2	0.06	0.0834	0.0696	41.9009
Euclidean	1	0.25	0.0296	0.0687	0.0407	36.0834
Euclidean	1	0.3	0.0263	0.0656	0.0372	32.3948
Euclidean	1	0.35	0.0277	0.0656	0.0389	32.4443
Euclidean	3	0.1	0.0712	0.0771	0.0735	62.5107
Euclidean	3	0.2	0.1073	0.0785	0.0893	40.4934
Euclidean	3	0.25	0.1223	0.0864	0.1001	37.1022
Euclidean	3	0.3	0.1167	0.0849	0.097	37.0534
Euclidean	3	0.35	0.1024	0.0802	0.0877	34.6799
Euclidean	5	0.1	0.0924	0.0812	0.0857	59.3134
Euclidean	5	0.2	0.1163	0.0999	0.1072	42.2443
Euclidean	5	0.25	0.1128	0.0922	0.1011	38.7512
Euclidean	5	0.3	0.1175	0.0861	0.0991	36.7609
Euclidean	5	0.35	0.1187	0.081	0.0949	35.1135
Euclidean	7	0.1	0.0866	0.084	0.085	59.8452
Euclidean	7	0.2	0.119	0.0985	0.1069	42.8235
Euclidean	7	0.25	0.116	0.0974	0.1051	39.428
Euclidean	7	0.3	0.1162	0.0965	0.1036	36.5178
Euclidean	7	0.35	0.109	0.0845	0.0926	35.1135
Euclidean	10	0.1	0.0862	0.0825	0.084	60.2793
Euclidean	10	0.2	0.1107	0.0963	0.1014	44.4745
Euclidean	10	0.25	0.1249	0.0916	0.1035	38.4095
Euclidean	10	0.3	0.1251	0.0905	0.1028	37.0055
Euclidean	10	0.35	0.1192	0.0858	0.0973	35.9378
Euclidean	12	0.1	0.0886	0.0868	0.0876	59.7472
Euclidean	12	0.2	0.1215	0.0994	0.1071	45.0546
Euclidean	12	0.25	0.1269	0.0959	0.1077	39.5746
Euclidean	12	0.3	0.1449	0.093	0.1117	38.2653
Euclidean	12	0.35	0.1341	0.0861	0.1037	34.0953
Euclidean	15	0.1	0.083	0.0862	0.0842	58.8256
Euclidean	15	0.2	0.1362	0.0947	0.1108	42.7768
Euclidean	15	0.25	0.1439	0.1009	0.1177	41.2736
Euclidean	15	0.3	0.1497	0.0931	0.1137	37.5369
Euclidean	15	0.35	0.1247	0.0853	0.0989	36.1332
Euclidean	20	0.1	0.088	0.09	0.0887	59.0195
Euclidean	20	0.2	0.1326	0.1041	0.1152	44.6182
Euclidean	20	0.25	0.1424	0.0995	0.1165	41.03
Euclidean	20	0.3	0.1379	0.0907	0.1077	38.5556
Euclidean	20	0.35	0.1206	0.0837	0.0955	35.162
Euclidean	25	0.1	0.0876	0.0888	0.088	58.7278
Euclidean	25	0.2	0.1317	0.0987	0.1121	45.7828
Euclidean	25	0.25	0.142	0.1024	0.1179	41.3691
Euclidean	25	0.3	0.1525	0.0907	0.1132	38.1664
Euclidean	25	0.35	0.149	0.0884	0.1099	37.1519
Euclidean	30	0.1	0.0943	0.0896	0.0913	59.1145
Euclidean	30	0.2	0.1423	0.0961	0.1138	44.4754
Euclidean	30	0.25	0.1397	0.0922	0.1105	40.2071
Euclidean	30	0.3	0.1523	0.0906	0.1122	37.9736
Euclidean	30	0.35	0.1453	0.0844	0.1043	35.6925
Euclidean	35	0.1	0.0921	0.0909	0.0913	57.6601
Euclidean	35	0.2	0.143	0.0963	0.1136	43.8441
Euclidean	35	0.25	0.1572	0.0947	0.1167	39.7216
Euclidean	35	0.3	0.1446	0.0879	0.1078	37.2473
Euclidean	35	0.35	0.1414	0.0871	0.1065	36.1782
Euclidean	40	0.1	0.0944	0.0903	0.0917	57.6139
Euclidean	40	0.2	0.1319	0.0973	0.1108	43.9424
Euclidean	40	0.25	0.1461	0.0977	0.1161	41.2722
Euclidean	40	0.3	0.1341	0.0897	0.1041	38.0214
Euclidean	40	0.35	0.1217	0.0855	0.097	35.9856

Table 22: RF Label Powerset, performance measures 5-fold-cross validation

		5-fold-cross validation			
		Performance measures			
Ntree	Mtry	Precision	Recall	F-measure	Percentage
250	15	0.1877	0.126	0.1505	41.9992
250	$\sqrt{\#\ explanatory\ variables}$	0.2076	0.1333	0.1607	42.7965
250	30	0.2119	0.138	0.1657	42.4757
250	60	0.1895	0.1457	0.1647	43.1547
250	100	0.1815	0.1383	0.1567	42.1079
500	15	0.1841	0.1248	0.1484	41.6116
500	$\sqrt{\#\ explanatory\ variables}$	0.2072	0.1314	0.1598	42.457
500	30	0.2005	0.14	0.1647	42.3794
500	60	0.2151	0.1471	0.1744	43.0064
500	100	0.1926	0.1414	0.1623	42.561
1000	15	0.167	0.1254	0.1415	41.9992
1000	$\sqrt{\#\ explanatory\ variables}$	0.1953	0.1335	0.1578	42.6516
1000	30	0.205	0.1388	0.1653	42.8176
1000	60	0.1881	0.1433	0.1625	42.6957
1000	100	0.1834	0.1411	0.1591	42.601

Table 23: RF Binary Relevance, performance measures 5-fold-cross validation

			5-fold-cross validation			
			Performance measures			
Ntree	Mtry	Threshold	Precision	Recall	F-measure	Percentage
250	15	0.3	0.2746	0.1476	0.19	45.976
250	15	0.4	0.2827	0.1359	0.182	44.569
250	15	0.5	0.2573	0.129	0.1709	43.6466
250	$\sqrt{\#\ explanatory\ variables}$	0.3	0.2811	0.1518	0.1964	46.0243
250	$\sqrt{\#\ explanatory\ variables}$	0.4	0.2989	0.1475	0.1962	44.9099
250	$\sqrt{\#\ explanatory\ variables}$	0.5	0.2752	0.1362	0.1816	43.9874
250	30	0.3	0.2464	0.1583	0.1915	46.1221
250	30	0.4	0.2655	0.152	0.1922	45.3459
250	30	0.5	0.2847	0.145	0.1911	45.2491
250	60	0.3	0.2305	0.1611	0.1881	46.6549
250	60	0.4	0.2516	0.1629	0.1966	45.5891
250	60	0.5	0.2357	0.148	0.181	44.7153
250	100	0.3	0.2305	0.1667	0.1928	46.8991
250	100	0.4	0.2422	0.1626	0.1931	45.2981
250	100	0.5	0.242	0.152	0.1852	43.7468

Table 24: RF Algorithm Adaptation, performances measures 5-fold-cross validation

		5-fold-cross validation			
		Performance measures			
Ntree	Mtry	Precision	Recall	F-measure	Percentage
100	15	0.1526	0.0791	0.1032	34.0943
100	$\sqrt{\#\ explanatory\ variables}$	0.1634	0.0911	0.1162	35.3072
100	30	0.1579	0.0954	0.1179	36.4727
100	60	0.1948	0.1026	0.1341	37.0046
100	100	0.2268	0.1062	0.1441	37.1484
100	250	0.1849	0.1105	0.1371	37.8779
100	500	0.2102	0.1116	0.1457	37.1983
250	15	0.1553	0.0802	0.1049	34.095
250	$\sqrt{\#\ explanatory\ variables}$	0.1498	0.0902	0.1124	35.5985
250	30	0.1467	0.0933	0.1137	36.083
250	60	0.1855	0.1006	0.1294	36.7635
250	100	0.2011	0.1043	0.1371	37.3922
250	250	0.2132	0.1084	0.1433	37.5365
250	500	0.2273	0.1115	0.1492	37.5867
500	15	0.1429	0.0787	0.1002	33.8525
500	$\sqrt{\#\ explanatory\ variables}$	0.1575	0.0912	0.1148	35.6958
500	30	0.18	0.0973	0.1256	36.7621
500	60	0.1968	0.101	0.1327	37.5867
500	100	0.2074	0.1052	0.1391	37.5379
500	250	0.2054	0.1087	0.1419	37.2956
500	500	0.2327	0.1125	0.1511	38.0228
1000	15	0.1374	0.0789	0.0987	34.0458
1000	$\sqrt{\#\ explanatory\ variables}$	0.1545	0.0895	0.1128	35.5987
1000	30	0.1572	0.0962	0.119	36.7619
1000	60	0.1871	0.0996	0.129	37.2466
1000	100	0.1928	0.1028	0.1339	37.1007
1000	250	0.2063	0.1102	0.1435	37.8282
1000	500	0.2166	0.1095	0.145	37.5862

Table 25: SVM Label Powerset, performance measures 5-fold-cross validation

			5-fold-cross validation			
			Performance measures			
Kernel	γ	Cost	Precision	Recall	F-measure	Percentage
Linear		0.1	0.2087	0.1187	0.1511	40.3051
Linear		1	0.1601	0.1427	0.1509	40.9791
Linear		10	0.1309	0.1248	0.1278	38.1202
Linear		100	0.1251	0.123	0.124	37.4394
Linear		500	0.1283	0.125	0.1266	37.8275
Linear		1000	0.1295	0.1262	0.1278	38.4095
Linear		10000	0.128	0.1248	0.1264	38.0704
Radial	$2^{-\#\ explanatory\ variables}$	0.1	0.0134	0.0588	0.0218	34.1424
Radial	$2^{\#\ explanatory\ variables}/_2$	0.1	0.0134	0.0588	0.0218	34.1424
Radial	2^{-100}	0.1	0.0134	0.0588	0.0218	34.1424
Radial	2^{-10}	0.1	0.0134	0.0588	0.0218	34.1424
Radial	$1/_{\#\ explanatory\ variables}$	0.1	0.0134	0.0588	0.0218	34.1424
Radial	$1/_{\#\ explanatory\ variables/_2}$	0.1	0.0134	0.0588	0.0218	34.1424
Radial	1	0.1	0.0134	0.0588	0.0218	34.1424
Radial	$2^{-\#\ explanatory\ variables}$	1	0.0134	0.0588	0.0218	34.1424
Radial	$2^{\#\ explanatory\ variables}/_2$	1	0.0134	0.0588	0.0218	34.1424
Radial	2^{-100}	1	0.0134	0.0588	0.0218	34.1424
Radial	2^{-10}	1	0.0134	0.0588	0.0218	34.1424
Radial	$1/_{\#\ explanatory\ variables}$	1	0.0161	0.0602	0.0251	34.288
Radial	$1/_{\#\ explanatory\ variables/_2}$	1	0.0256	0.0669	0.0365	35.7422
Radial	1	1	0.1144	0.0709	0.0839	35.5495
Radial	$2^{-\#\ explanatory\ variables}$	10	0.0134	0.0588	0.0218	34.1424
Radial	$2^{\#\ explanatory\ variables}/_2$	10	0.0134	0.0588	0.0218	34.1424
Radial	2^{-100}	10	0.0134	0.0588	0.0218	34.1424
Radial	2^{-10}	10	0.0292	0.0726	0.0417	37.1995
Radial	$1/_{\#\ explanatory\ variables}$	10	0.0804	0.0844	0.0796	38.1701
Radial	$1/_{\#\ explanatory\ variables/_2}$	10	0.1834	0.1121	0.139	39.8208
Radial	1	10	0.1462	0.0774	0.0997	36.5206
Radial	$2^{-\#\ explanatory\ variables}$	100	0.0134	0.0588	0.0218	34.1424
Radial	$2^{\#\ explanatory\ variables}/_2$	100	0.0134	0.0588	0.0218	34.1424
Radial	2^{-100}	100	0.0134	0.0588	0.0218	34.1424
Radial	2^{-10}	100	0.1832	0.1367	0.1561	41.1761
Radial	$1/_{\#\ explanatory\ variables}$	100	0.1719	0.1386	0.1534	40.7368
Radial	$1/_{\#\ explanatory\ variables/_2}$	100	0.1657	0.1422	0.1529	40.9788
Radial	1	100	0.1462	0.0774	0.0997	36.5206
Radial	$2^{-\#\ explanatory\ variables}$	500	0.0134	0.0588	0.0218	34.1424
Radial	$2^{\#\ explanatory\ variables}/_2$	500	0.0134	0.0588	0.0218	34.1424
Radial	2^{-100}	500	0.0134	0.0588	0.0218	34.1424
Radial	2^{-10}	500	0.1559	0.1408	0.1479	40.7851
Radial	$1/_{\#\ explanatory\ variables}$	500	0.1503	0.141	0.1455	40.0129
Radial	$1/_{\#\ explanatory\ variables/_2}$	500	0.1395	0.1334	0.1364	39.3319
Radial	1	500	0.1462	0.0774	0.0997	36.5206
Radial	$2^{-\#\ explanatory\ variables}$	1000	0.0134	0.0588	0.0218	34.1424
Radial	$2^{\#\ explanatory\ variables}/_2$	1000	0.0134	0.0588	0.0218	34.1424
Radial	2^{-100}	1000	0.0134	0.0588	0.0218	34.1424
Radial	2^{-10}	1000	0.1512	0.1415	0.1461	40.4005
Radial	$1/_{\#\ explanatory\ variables}$	1000	0.1403	0.1342	0.1372	39.4297
Radial	$1/_{\#\ explanatory\ variables/_2}$	1000	0.1346	0.1277	0.131	38.1202
Radial	1	1000	0.1462	0.0774	0.0997	36.5206
Radial	$2^{-\#\ explanatory\ variables}$	10000	0.0134	0.0588	0.0218	34.1424
Radial	$2^{\#\ explanatory\ variables}/_2$	10000	0.0134	0.0588	0.0218	34.1424
Radial	2^{-100}	10000	0.0134	0.0588	0.0218	34.1424
Radial	2^{-10}	10000	0.1268	0.1243	0.1255	38.2173
Radial	$1/_{\#\ explanatory\ variables}$	10000	0.1357	0.127	0.1309	37.9734
Radial	$1/_{\#\ explanatory\ variables/_2}$	10000	0.1307	0.1272	0.1288	38.4093
Radial	1	10000	0.1462	0.0774	0.0997	36.5206

Table 26: SVM Binary Relevance 1/2, performance measures 5-fold-cross validation

				5-fold-cross validation			
				Performance measures			
Kernel	γ	Cost	Threshold	Precision	Recall	F-measure	Percentage
Linear		0.1	0.3	0.2263	0.1633	0.1892	47.1894
Linear		0.1	0.4	0.2413	0.1606	0.1922	46.0269
Linear		0.1	0.5	0.2473	0.1527	0.1874	44.765
Linear		1	0.3	0.2119	0.1568	0.18	45.7352
Linear		1	0.4	0.2252	0.1513	0.1803	44.3286
Linear		1	0.5	0.237	0.1428	0.1777	42.9701
Linear		10	0.3	0.1628	0.1308	0.1449	42.1493
Linear		10	0.4	0.185	0.1224	0.1465	40.7898
Linear		10	0.5	0.1983	0.1123	0.1422	39.5741
Linear		100	0.3	0.1563	0.1124	0.1299	40.886
Linear		100	0.4	0.1677	0.1042	0.1284	39.9153
Linear		100	0.5	0.1801	0.0955	0.1234	38.4604
Linear		500	0.3	0.1681	0.103	0.125	38.5078
Linear		500	0.4	0.1549	0.0974	0.1186	38.5087
Linear		500	0.5	0.1609	0.0933	0.1164	37.9265
Linear		1000	0.3	0.1737	0.104	0.1275	38.653
Linear		1000	0.4	0.1427	0.0966	0.114	38.6051
Linear		1000	0.5	0.1461	0.0901	0.1086	37.8303
Radial	2^{-10}	0.1	0.3	0.2338	0.1628	0.191	47.2867
Radial	2^{-10}	0.1	0.4	0.2316	0.1541	0.1839	46.1231
Radial	2^{-10}	0.1	0.5	0.246	0.1457	0.1802	44.6677
Radial	$1/_{\#\ explanatory\ variables}$	0.1	0.3	0.2252	0.1626	0.1882	47.3836
Radial	$1/_{\#\ explanatory\ variables}$	0.1	0.4	0.2383	0.158	0.1889	46.2192
Radial	$1/_{\#\ explanatory\ variables}$	0.1	0.5	0.2399	0.1491	0.1824	45.5408
Radial	$1/_{\#\ explanatory\ variables/_2}$	0.1	0.3	0.2235	0.1673	0.1909	47.9173
Radial	$1/_{\#\ explanatory\ variables/_2}$	0.1	0.4	0.2339	0.159	0.1886	46.5586
Radial	$1/_{\#\ explanatory\ variables/_2}$	0.1	0.5	0.2503	0.1518	0.1877	45.5413
Radial	1	0.1	0.3	0.1285	0.0824	0.1002	37.101
Radial	1	0.1	0.4	0.1405	0.0813	0.1023	37.0529
Radial	1	0.1	0.5	0.1643	0.0782	0.1053	36.5686
Radial	2^{-10}	1	0.3	0.2206	0.1635	0.1873	47.4326
Radial	2^{-10}	1	0.4	0.231	0.1601	0.1883	46.7527
Radial	2^{-10}	1	0.5	0.2412	0.1502	0.1836	45.2012
Radial	$1/_{\#\ explanatory\ variables}$	1	0.3	0.2241	0.1651	0.1895	47.6746
Radial	$1/_{\#\ explanatory\ variables}$	1	0.4	0.2377	0.16	0.1904	46.753
Radial	$1/_{\#\ explanatory\ variables}$	1	0.5	0.2463	0.1531	0.1876	45.6376
Radial	$1/_{\#\ explanatory\ variables/_2}$	1	0.3	0.2227	0.1672	0.1906	48.0144
Radial	$1/_{\#\ explanatory\ variables/_2}$	1	0.4	0.23	0.159	0.1874	46.4132
Radial	$1/_{\#\ explanatory\ variables/_2}$	1	0.5	0.2531	0.1551	0.1913	45.7835
Radial	1	1	0.3	0.1492	0.0833	0.1061	37.2952
Radial	1	1	0.4	0.1422	0.08	0.1017	36.908
Radial	1	1	0.5	0.1624	0.0787	0.105	36.5689

Table 27: SVM Binary Relevance 2/2, performance measures 5-fold-cross validation

				5-fold-cross validation			
				Performance measures			
Kernel	γ	Cost	Threshold	Precision	Recall	F-measure	Percentage
Radial	2^{-10}	10	0.3	0.2235	0.1662	0.1902	47.6258
Radial	2^{-10}	10	0.4	0.2345	0.1606	0.1897	46.8498
Radial	2^{-10}	10	0.5	0.248	0.1525	0.1872	45.2988
Radial	$1/_{\#\ explanatory\ variables}$	10	0.3	0.2166	0.1678	0.1888	47.8685
Radial	$1/_{\#\ explanatory\ variables}$	10	0.4	0.2401	0.1618	0.1922	46.5586
Radial	$1/_{\#\ explanatory\ variables}$	10	0.5	0.2509	0.155	0.1902	45.2997
Radial	$1/_{\#\ explanatory\ variables/_2}$	10	0.3	0.2234	0.17	0.1927	48.1113
Radial	$1/_{\#\ explanatory\ variables/_2}$	10	0.4	0.2326	0.1639	0.1919	46.2197
Radial	$1/_{\#\ explanatory\ variables/_2}$	10	0.5	0.2505	0.157	0.1922	45.2505
Radial	1	10	0.3	0.1242	0.0784	0.0948	36.5677
Radial	1	10	0.4	0.1359	0.076	0.0969	36.4225
Radial	1	10	0.5	0.1156	0.0739	0.0896	36.2293
Radial	2^{-10}	100	0.3	0.2371	0.1647	0.1941	47.3831
Radial	2^{-10}	100	0.4	0.2446	0.1581	0.191	46.0743
Radial	2^{-10}	100	0.5	0.242	0.1516	0.1854	44.3767
Radial	$1/_{\#\ explanatory\ variables}$	100	0.3	0.2179	0.1659	0.188	47.7213
Radial	$1/_{\#\ explanatory\ variables}$	100	0.4	0.2421	0.1568	0.1898	45.1531
Radial	$1/_{\#\ explanatory\ variables}$	100	0.5	0.2512	0.149	0.1861	43.9402
Radial	$1/_{\#\ explanatory\ variables/_2}$	100	0.3	0.2123	0.1594	0.1818	46.4601
Radial	$1/_{\#\ explanatory\ variables/_2}$	100	0.4	0.2291	0.1579	0.1866	45.7814
Radial	$1/_{\#\ explanatory\ variables/_2}$	100	0.5	0.238	0.1498	0.183	44.2775
Radial	1	100	0.3	0.1242	0.0784	0.0948	36.5677
Radial	1	100	0.4	0.1359	0.076	0.0969	36.4225
Radial	1	100	0.5	0.1156	0.0739	0.0896	36.2293
Radial	2^{-10}	500	0.3	0.2128	0.1581	0.1812	46.0257
Radial	2^{-10}	500	0.4	0.2228	0.151	0.1795	44.4254
Radial	2^{-10}	500	0.5	0.2306	0.1435	0.1766	43.0182
Radial	$1/_{\#\ explanatory\ variables}$	500	0.3	0.1991	0.1575	0.1755	45.6859
Radial	$1/_{\#\ explanatory\ variables}$	500	0.4	0.2086	0.1485	0.1723	43.8917
Radial	$1/_{\#\ explanatory\ variables}$	500	0.5	0.2127	0.1343	0.1636	42.0965
Radial	$1/_{\#\ explanatory\ variables/_2}$	500	0.3	0.181	0.1487	0.163	44.4236
Radial	$1/_{\#\ explanatory\ variables/_2}$	500	0.4	0.1947	0.1426	0.164	43.164
Radial	$1/_{\#\ explanatory\ variables/_2}$	500	0.5	0.2041	0.1331	0.1597	42.3873
Radial	1	500	0.3	0.1242	0.0784	0.0948	36.5677
Radial	1	500	0.4	0.1359	0.076	0.0969	36.4225
Radial	1	500	0.5	0.1156	0.0739	0.0896	36.2293
Radial	2^{-10}	1000	0.3	0.2018	0.1548	0.1748	45.2509
Radial	2^{-10}	1000	0.4	0.2039	0.148	0.1713	43.99
Radial	2^{-10}	1000	0.5	0.2282	0.1355	0.169	42.6786
Radial	$1/_{\#\ explanatory\ variables}$	1000	0.3	0.1832	0.1469	0.1625	44.1825
Radial	$1/_{\#\ explanatory\ variables}$	1000	0.4	0.1935	0.1408	0.1621	43.0667
Radial	$1/_{\#\ explanatory\ variables}$	1000	0.5	0.1955	0.1272	0.1528	41.1756
Radial	$1/_{\#\ explanatory\ variables/_2}$	1000	0.3	0.1792	0.1456	0.1603	44.6185
Radial	$1/_{\#\ explanatory\ variables/_2}$	1000	0.4	0.1955	0.1412	0.1633	43.2119
Radial	$1/_{\#\ explanatory\ variables/_2}$	1000	0.5	0.2013	0.1268	0.1549	40.9826
Radial	1	1000	0.3	0.1242	0.0784	0.0948	36.5677
Radial	1	1000	0.4	0.1359	0.076	0.0969	36.4225
Radial	1	1000	0.5	0.1156	0.0739	0.0896	36.2293

Table 28: SVM Algorithm Adaptation 1/2, performance measures 5-fold-cross validation

				5-fold-cross validation			
				Performance measures			
Kernel	γ	Cost	Threshold	Precision	Recall	F-measure	Percentage
Linear		0.1	0.1	0.0102	0.0588	0.0173	26.0445
Linear		0.1	0.2	0.0102	0.0588	0.0173	26.0445
Linear		0.1	0.3	0.0102	0.0588	0.0173	26.0445
Linear		1	0.1	0.0563	0.0685	0.0606	26.7237
Linear		1	0.2	0.0102	0.0588	0.0173	26.0445
Linear		1	0.3	0.0102	0.0588	0.0173	26.0445
Linear		10	0.1	0.1691	0.1402	0.1529	53.8781
Linear		10	0.2	0.223	0.154	0.1815	44.4257
Linear		10	0.3	0.2051	0.117	0.1475	35.5506
Linear		25	0.1	0.1331	0.1281	0.13	55.8184
Linear		25	0.2	0.1832	0.153	0.1665	38.1666
Linear		25	0.3	0.1928	0.1234	0.1504	30.5567
Linear		50	0.1	0.1261	0.1239	0.1238	54.8471
Linear		50	0.2	0.1506	0.1474	0.1489	33.7048
Linear		50	0.3	0.1553	0.1183	0.1342	28.6638
Linear		100	0.1	0.1186	0.1256	0.1209	53.3437
Linear		100	0.2	0.1432	0.1433	0.1432	31.6711
Linear		100	0.3	0.1564	0.1188	0.135	27.9349
Linear		500	0.1	0.1148	0.1256	0.1198	49.7563
Linear		500	0.2	0.1409	0.1409	0.1409	30.8456
Linear		500	0.3	0.1526	0.1184	0.1332	27.7403
Linear		1000	0.1	0.1148	0.1256	0.1198	49.7563
Linear		1000	0.2	0.1409	0.1409	0.1409	30.8456
Linear		1000	0.3	0.1526	0.1184	0.1332	27.7403
Linear		10000	0.1	0.1148	0.1256	0.1198	49.7563
Linear		10000	0.2	0.1409	0.1409	0.1409	30.8456
Linear		10000	0.3	0.1526	0.1184	0.1332	27.7403
Radial	2^{-10}	0.1	0.1	0.0102	0.0588	0.0173	26.0445
Radial	2^{-10}	0.1	0.2	0.0102	0.0588	0.0173	26.0445
Radial	2^{-10}	0.1	0.3	0.0102	0.0588	0.0173	26.0445
Radial	$1/_{\#\ explanatory\ variables}$	0.1	0.1	0.0102	0.0588	0.0173	26.0445
Radial	$1/_{\#\ explanatory\ variables}$	0.1	0.2	0.0102	0.0588	0.0173	26.0445
Radial	$1/_{\#\ explanatory\ variables}$	0.1	0.3	0.0102	0.0588	0.0173	26.0445
Radial	$1/_{\#\ explanatory\ variables/_2}$	0.1	0.1	0.0102	0.0588	0.0173	26.0445
Radial	$1/_{\#\ explanatory\ variables/_2}$	0.1	0.2	0.0102	0.0588	0.0173	26.0445
Radial	$1/_{\#\ explanatory\ variables/_2}$	0.1	0.3	0.0102	0.0588	0.0173	26.0445
Radial	1	0.1	0.1	0.0102	0.0588	0.0173	26.0445
Radial	1	0.1	0.2	0.0102	0.0588	0.0173	26.0445
Radial	1	0.1	0.3	0.0102	0.0588	0.0173	26.0445
Radial	2^{-10}	1	0.1	0.068	0.0678	0.0655	26.6751
Radial	2^{-10}	1	0.2	0.0102	0.0588	0.0173	26.0445
Radial	2^{-10}	1	0.3	0.0102	0.0588	0.0173	26.0445
Radial	$1/_{\#\ explanatory\ variables}$	1	0.1	0.0621	0.0678	0.0619	26.6751
Radial	$1/_{\#\ explanatory\ variables}$	1	0.2	0.0102	0.0588	0.0173	26.0445
Radial	$1/_{\#\ explanatory\ variables}$	1	0.3	0.0102	0.0588	0.0173	26.0445
Radial	$1/_{\#\ explanatory\ variables/_2}$	1	0.1	0.0664	0.0687	0.0638	26.7717
Radial	$1/_{\#\ explanatory\ variables/_2}$	1	0.2	0.0102	0.0588	0.0173	26.0445
Radial	$1/_{\#\ explanatory\ variables/_2}$	1	0.3	0.0102	0.0588	0.0173	26.0445
Radial	1	1	0.1	0.0484	0.0609	0.0499	26.287
Radial	1	1	0.2	0.0102	0.0588	0.0173	26.0445
Radial	1	1	0.3	0.0102	0.0588	0.0173	26.0445
Radial	2^{-10}	10	0.1	0.1667	0.1408	0.1523	53.5388
Radial	2^{-10}	10	0.2	0.2263	0.1565	0.1844	44.6682
Radial	2^{-10}	10	0.3	0.2146	0.1193	0.1518	35.9859
Radial	$1/_{\#\ explanatory\ variables}$	10	0.1	0.1665	0.1426	0.1533	53.8293
Radial	$1/_{\#\ explanatory\ variables}$	10	0.2	0.2245	0.1565	0.1839	44.4742
Radial	$1/_{\#\ explanatory\ variables}$	10	0.3	0.2227	0.1194	0.1541	35.9854
Radial	$1/_{\#\ explanatory\ variables/_2}$	10	0.1	0.1699	0.1447	0.1559	53.8293
Radial	$1/_{\#\ explanatory\ variables/_2}$	10	0.2	0.2235	0.1575	0.1841	44.5239
Radial	$1/_{\#\ explanatory\ variables/_2}$	10	0.3	0.2313	0.1179	0.1548	35.8883
Radial	1	10	0.1	0.1095	0.0767	0.0897	36.9082
Radial	1	10	0.2	0.0835	0.0652	0.0713	27.0151
Radial	1	10	0.3	0.0494	0.0612	0.0482	26.2872

Table 29: SVM Algorithm Adaptation 2/2, performance measures 5-fold-cross validation

				5-fold-cross validation			
				Performance measures			
Kernel	γ	Cost	Threshold	Precision	Recall	F-measure	Percentage
Radial	2^{-10}	25	0.1	0.1257	0.1277	0.1262	55.9158
Radial	2^{-10}	25	0.2	0.1867	0.1578	0.1707	38.0693
Radial	2^{-10}	25	0.3	0.1927	0.1207	0.1484	30.6538
Radial	$1/_{\#\ explanatory\ variables}$	25	0.1	0.1276	0.1294	0.128	56.4007
Radial	$1/_{\#\ explanatory\ variables}$	25	0.2	0.1911	0.1611	0.1745	38.7477
Radial	$1/_{\#\ explanatory\ variables}$	25	0.3	0.1898	0.1206	0.1474	30.9451
Radial	$1/_{\#\ explanatory\ variables/_2}$	25	0.1	0.1273	0.1308	0.1286	56.9828
Radial	$1/_{\#\ explanatory\ variables/_2}$	25	0.2	0.1868	0.1604	0.1724	39.2336
Radial	$1/_{\#\ explanatory\ variables/_2}$	25	0.3	0.1938	0.1231	0.1505	30.9465
Radial	1	25	0.1	0.0995	0.076	0.0859	37.3451
Radial	1	25	0.2	0.0841	0.0648	0.072	26.9181
Radial	1	25	0.3	0.0455	0.0611	0.0438	26.2387
Radial	2^{-10}	50	0.1	0.1207	0.1253	0.1219	55.1871
Radial	2^{-10}	50	0.2	0.1518	0.1479	0.1497	34.4806
Radial	2^{-10}	50	0.3	0.1573	0.1172	0.1342	28.5191
Radial	$1/_{\#\ explanatory\ variables}$	50	0.1	0.1283	0.1274	0.1267	56.1083
Radial	$1/_{\#\ explanatory\ variables}$	50	0.2	0.1561	0.1505	0.1532	35.0141
Radial	$1/_{\#\ explanatory\ variables}$	50	0.3	0.1674	0.1182	0.1383	28.8575
Radial	$1/_{\#\ explanatory\ variables/_2}$	50	0.1	0.1174	0.1287	0.1223	56.4003
Radial	$1/_{\#\ explanatory\ variables/_2}$	50	0.2	0.1595	0.1522	0.1557	35.3053
Radial	$1/_{\#\ explanatory\ variables/_2}$	50	0.3	0.177	0.1191	0.1421	29.6827
Radial	1	50	0.1	0.1081	0.076	0.089	37.3446
Radial	1	50	0.2	0.0836	0.0647	0.0718	26.8695
Radial	1	50	0.3	0.0553	0.0616	0.0499	26.287
Radial	2^{-10}	100	0.1	0.1187	0.1266	0.1214	53.5857
Radial	2^{-10}	100	0.2	0.1451	0.1442	0.1447	32.203
Radial	2^{-10}	100	0.3	0.1586	0.1177	0.135	28.2269
Radial	$1/_{\#\ explanatory\ variables}$	100	0.1	0.1058	0.127	0.1154	54.1196
Radial	$1/_{\#\ explanatory\ variables}$	100	0.2	0.1495	0.1458	0.1476	32.8817
Radial	$1/_{\#\ explanatory\ variables}$	100	0.3	0.1541	0.1175	0.1332	28.664
Radial	$1/_{\#\ explanatory\ variables/_2}$	100	0.1	0.107	0.1279	0.1165	55.2357
Radial	$1/_{\#\ explanatory\ variables/_2}$	100	0.2	0.1579	0.1506	0.1541	34.3858
Radial	$1/_{\#\ explanatory\ variables/_2}$	100	0.3	0.1609	0.1184	0.1362	29.7798
Radial	1	100	0.1	0.1081	0.076	0.089	37.3446
Radial	1	100	0.2	0.0836	0.0647	0.0718	26.8695
Radial	1	100	0.3	0.0553	0.0616	0.0499	26.287
Radial	2^{-10}	500	0.1	0.1163	0.1262	0.1209	50.7256
Radial	2^{-10}	500	0.2	0.1429	0.1423	0.1426	31.427
Radial	2^{-10}	500	0.3	0.1561	0.118	0.1343	28.2259
Radial	$1/_{\#\ explanatory\ variables}$	500	0.1	0.1169	0.126	0.1211	51.5501
Radial	$1/_{\#\ explanatory\ variables}$	500	0.2	0.1466	0.1445	0.1455	32.0098
Radial	$1/_{\#\ explanatory\ variables}$	500	0.3	0.1611	0.118	0.1361	28.6642
Radial	$1/_{\#\ explanatory\ variables/_2}$	500	0.1	0.1181	0.1276	0.1226	52.6164
Radial	$1/_{\#\ explanatory\ variables/_2}$	500	0.2	0.1515	0.1459	0.1486	33.3683
Radial	$1/_{\#\ explanatory\ variables/_2}$	500	0.3	0.1639	0.1209	0.139	29.3921
Radial	1	500	0.1	0.1081	0.076	0.089	37.3446
Radial	1	500	0.2	0.0836	0.0647	0.0718	26.8695
Radial	1	500	0.3	0.0553	0.0616	0.0499	26.287
Radial	2^{-10}	1000	0.1	0.1163	0.1262	0.1209	50.7256
Radial	2^{-10}	1000	0.2	0.1429	0.1423	0.1426	31.427
Radial	2^{-10}	1000	0.3	0.1561	0.118	0.1343	28.2259
Radial	$1/_{\#\ explanatory\ variables}$	1000	0.1	0.1169	0.126	0.1211	51.5501
Radial	$1/_{\#\ explanatory\ variables}$	1000	0.2	0.1466	0.1445	0.1455	32.0098
Radial	$1/_{\#\ explanatory\ variables}$	1000	0.3	0.1611	0.118	0.1361	28.6642
Radial	$1/_{\#\ explanatory\ variables/_2}$	1000	0.1	0.1181	0.1276	0.1226	52.6164
Radial	$1/_{\#\ explanatory\ variables/_2}$	1000	0.2	0.1515	0.1459	0.1486	33.3683
Radial	$1/_{\#\ explanatory\ variables/_2}$	1000	0.3	0.1639	0.1209	0.139	29.3921
Radial	1	1000	0.1	0.1081	0.076	0.089	37.3446
Radial	1	1000	0.2	0.0836	0.0647	0.0718	26.8695
Radial	1	1000	0.3	0.0553	0.0616	0.0499	26.287
Radial	2^{-10}	10000	0.1	0.1163	0.1262	0.1209	50.7256
Radial	2^{-10}	10000	0.2	0.1429	0.1423	0.1426	31.427
Radial	2^{-10}	10000	0.3	0.1561	0.118	0.1343	28.2259
Radial	$1/_{\#\ explanatory\ variables}$	10000	0.1	0.1169	0.126	0.1211	51.5501
Radial	$1/_{\#\ explanatory\ variables}$	10000	0.2	0.1466	0.1445	0.1455	32.0098
Radial	$1/_{\#\ explanatory\ variables}$	10000	0.3	0.1611	0.118	0.1361	28.6642
Radial	$1/_{\#\ explanatory\ variables/_2}$	10000	0.1	0.1181	0.1276	0.1226	52.6164
Radial	$1/_{\#\ explanatory\ variables/_2}$	10000	0.2	0.1515	0.1459	0.1486	33.3683
Radial	$1/_{\#\ explanatory\ variables/_2}$	10000	0.3	0.1639	0.1209	0.139	29.3921
Radial	1	10000	0.1	0.1081	0.076	0.089	37.3446
Radial	1	10000	0.2	0.0836	0.0647	0.0718	26.8695
Radial	1	10000	0.3	0.0553	0.0616	0.0499	26.287

C Appendix

Table 30: Benchmark performances

	5-fold-cross validation			Validation set			Test set			
	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Percentage
Benchmark 1	0.0588	0.0588	0.0587	0.0588	0.0593	0.059	0.0589	0.0594	0.0591	23.2
Benchmark 2	0.0232	0.0588	0.0332	0.0236	0.0588	0.0337	0.0236	0.0588	0.0337	64.19
Benchmark 3	0.0587	0.0589	0.0587	0.0587	0.0582	0.0584	0.0592	0.0598	0.0594	22.97
Benchmark 4	0.0589	0.0583	0.0583	0.0591	0.0591	0.059	0.0589	0.0577	0.0582	8.58

Table 31: KNN Label Powerset performances

			5-fold-cross validation			Validation set			Test set			
Distance function	k	Min Occurrence	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Percentage
Euclidean	1		0.1121	0.1198	0.1153	0.0944	0.0971	0.0957	0.0917	0.098	0.0948	32.6
Jaccard	1		0.1388	0.1382	0.1384	0.1067	0.1108	0.1087	0.1117	0.1107	0.1112	37.12

Table 32: KNN Binary Relevance performances measures

			5-fold-cross validation			Validation set			Test set			
Distance function	k	Threshold	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Percentage
Euclidean	7	0.4	0.1694	0.1297	0.1444	0.1442	0.1155	0.1283	0.1377	0.1212	0.1289	35.37
Euclidean	12	0.4	0.1735	0.1225	0.1418	0.1837	0.1237	0.1478	0.1574	0.1195	0.1358	36.83

Table 33: KNN Algorithm Adaptation performance measures

			5-fold-cross validation			Validation set			Test set			
Distance function	k	Threshold	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Percentage
Euclidean	15	0.25	0.1439	0.1009	0.1177	0.1459	0.0953	0.1153	0.1751	0.0841	0.1137	40.90
Euclidean	25	0.25	0.142	0.1024	0.1179	0.1347	0.0972	0.1129	0.1144	0.0963	0.1046	42.21

Table 34: RF Label Powerset performance measures

		5-fold-cross validation			Validation set			Test set			
Ntree	Mtry	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Percentage
500	60	0.2151	0.1471	0.1744	0.1586	0.1247	0.1396	0.1485	0.1234	0.1348	42.63
1000	30	0.205	0.1388	0.1653	0.2011	0.1206	0.1508	0.1614	0.1271	0.1422	45.27

Table 35: RF Binary Relevance performance measures

			5-fold-cross validation			Validation set			Test set			
Ntree	Mtry	Threshold	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Percentage
250	$\sqrt{\#\ explanatory\ variables}$	0.3	0.2811	0.1518	0.1964	0.2173	0.129	0.1619	0.25	0.1257	0.1673	44.69
250	60	0.4	0.2516	0.1629	0.1966	0.2147	0.1406	0.1699	0.2152	0.1366	0.1671	46.29

Table 36: RF Algorithm Adaptation performance measures

		5-fold-cross validation			Validation set			Test set			
Ntree	Mtry	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Percentage
100	500	0.2102	0.1116	0.1457	0.1084	0.0844	0.0949	0.1537	0.0863	0.1106	36.1
500	500	0.2327	0.1125	0.1511	0.1137	0.084	0.0966	0.179	0.0869	0.117	36.24

Table 37: SVM Label Powerset performance measures

			5-fold-cross validation			Validation set			Test set			
Kernel	γ	Cost	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Percentage
Radial	2^{-10}	100	0.1832	0.1367	0.1561	0.1467	0.1188	0.1313	0.2125	0.1397	0.1686	43.23
Radial	$1/_{\#\ explanatory\ variables}$	100	0.1719	0.1386	0.1534	0.1652	0.133	0.1474	0.2557	0.1376	0.1789	43.67

Table 38: SVM Binary Relevance performance measures

				5-fold-cross validation			Validation set			Test set			
Kernel	γ	Cost	Threshold	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Percentage
Radial	$1/_{\#\ explanatory\ variables/_2}$	10	0.3	0.2234	0.17	0.1927	0.3251	0.1428	0.1985	0.2131	0.1513	0.177	47.16
Radial	2^{-10}	100	0.3	0.2371	0.1647	0.1941	0.3717	0.143	0.2065	0.2295	0.1574	0.1867	47.6

Table 39: SVM Algorithm Adaptation Method performance measures

				5-fold-cross validation			Validation set			Test set			
Kernel	γ	Cost	Threshold	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Percentage
Radial	2^{-10}	10	0.2	0.2263	0.1565	0.1844	0.2792	0.1324	0.1796	0.1811	0.1327	0.1532	42.07
Radial	$1/_{\#\ explanatory\ variables/_2}$	10	0.2	0.2235	0.1575	0.1841	0.2952	0.1345	0.1848	0.1925	0.1329	0.1572	41.63

Table 40: P-values of paired T-Tests

	Benchmark 2	Benchmark 3	Benchmark 4	KNN BR 1	KNN BR 2	KNN AA 1	KNN AA 2	RF LP 2	SVM LP 1	SVM LP 2	SVM BR 1	SVM BR 2	SVM AA 1	SVM AA 2
Benchmark 1	0	0.7418	0.2379	0.0051	0.0018	5e-04	0.0056	0.0053	0.0136	0.0089	0.0025	0.0018	0.0074	0.0061
Benchmark 2		1e-04	1e-04	0.0029	0.001	3e-04	0.0026	0.0033	0.0085	0.0057	0.0017	0.0013	0.0049	0.0042
Benchmark 3			0.5876	0.005	0.002	6e-04	0.006	0.0054	0.0129	0.0085	0.0026	0.0019	0.0076	0.0064
Benchmark 4				0.0051	0.0015	4e-04	0.0051	0.0051	0.0141	0.0093	0.0023	0.0016	0.007	0.0058
KNN BR 1					0.3409	0.0499	0.0228	0.0218	0.2431	0.1692	0.017	0.0183	0.0389	0.0368
KNN BR 2						0.0144	0.0114	0.226	0.5516	0.2976	0.0045	0.002	0.0525	0.0332
KNN AA 1							0.3069	0.0218	0.0839	0.0516	0.0061	0.0048	0.0226	0.0183
KNN AA 2								0.0061	0.093	0.0669	0.0027	0.0037	0.0097	0.008
RF LP 2									0.961	0.681	0.0252	0.0314	0.0623	0.0601
SVM LP 1										0.2911	0.1585	0.1204	0.3914	0.3414
SVM LP 2											0.2077	0.1393	0.5801	0.4948
SVM BR 1												0.1283	0.0654	0.0493
SVM BR 2													0.0811	0.0692
SVM AA 1														0.2176

Table 41: P-values of paired T-Tests

	Combined Binary Relevance methods
Combined Algorithm Adaptation methods	$2.923615e-06$