NTNU

Norwegian University of
Science and Technology

Early warnings of critical diagnoses

Stig Alvestad

Master of Science in Computer Science
Submission date: February 2009
Supervisor: Øystein Nytrø, IDI
Co-supervisor: Carl-Fredrik Bassøe, INM
Ole Edsberg, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

For many medical conditions, early diagnosis is very important in order to improve prospects and prevent unnecessary damage. Sometimes, signs occur that in hindsight should have allowed a physician to suspect and diagnose the condition at an earlier stage. Since patient records in general practice contain long and relatively rich descriptions of the patient history, it should be possible to automatically discover such signs and alert the physician when there might be a reason to suspect the presence of the condition. This can be formulated as a machine learning problem. We have access to data extracted from general practice patient records, and we have identified three promising target diagnoses: Asthma, Diabetes and Hypothyroidism. The goals of this project are to develop methods that with basis in a training data set can:

* Given a partial patient history, predict whether or not the target diagnosis will occur in the near future.
* Discover rules, patterns, decision trees etc that give insight into the warning signs that precede a target diagnosis.

The student can focus on one or both of these goals. The students should develop several candidate methods and perform a throrough experimental evaluation.

The challenges of the problem include class imbalance, heterogeneity and sparsity of the data, temporal feature extraction and ensuring validity of evaluation. For a predictor to be clinically useful, it must have reasonable sensitivity and very low specificity, since clinicians will not tolerate many false alarms. We do not expect to achieve clinical-grade performance in this project, but even a weak predictor would be interesting, as a starting point for further research and a motivator for higher availability and quality of data. A negative result, if founded on systematic experiments, would also be interesting.

Assignment given: 17. September 2008
Supervisor: Øystein Nytrø, IDI

Abstract

Background: A disease which is left untreated for a longer period is more likely to cause negative consequents for the patient. Even though the general practitioner is able to discover the disease quickly in most cases, there are patients who should have been discovered earlier. Electronic patient records store time-stamped health information about patients, recorded by the health personnel treating the patient. This makes it possible to do a retrospective analysis in order to determine whether there was sufficient information to give the diagnose earlier than the general practitioner actually did. Classification algorithms from the machine learning domain can utilise large collections of electronic patient records to build models which can predict whether a patient will get the disease or not. These models could be used to get more knowledge about these diseases and in a long-term perspective they could become a support for the general practitioner in daily practice.

Purpose: The purpose of this thesis is to design and implement a software system which can predict whether a patient will get a disease in the near future or not. The system should attempt to predict the disease before the general practitioner even suspects that the patient might have the disease. Further the objective is to use this system to identify warning signs which are used to make the predictions, and to analyse the usefulness of the predictions and the warning signs. The diseases asthma, diabetes 2 and hypothyroidism have been selected to be the test cases for our methodology.

Methods: A set of suspicion-indicators which indicates that the general practitioner has suspected the disease are identified in an iterative process. These suspicion-indicators are subsequently used to limit the information available for the classification algorithms. This information is subsequently used to build prediction models, using different classification algoritms. The prediction models are evaluated in terms of various performance measures and the models themselves are analysed manually. Experiments are conducted in order to find favourable parameter values for the information extraction process. Because there are relatively few patients who have the disease test cases, the oversampling technique SMOTE is used to generate additional synthetical patients with the test cases.

Results: A set of suspicion-indicators has been identified in cooperation with domain experts. The availability of warning signs decreases as the information available for the classifier diminishes, while the performance of the classifiers is not affected to such a large degree. Applying the SMOTE oversampling technique improves the results for the prediction models. There is not much difference between the performance of the various classification algorithms.

Conclusions: The improved problem formulation results in models which are more valid than before. A number of events which are used to predict the test cases have been identified, but their real-world importance remains to be evaluated by domain experts. The performance of the prediction models can be misguiding in terms of practical usefulness. SMOTE is a promising technique for generating additional data, but the evaluation techniques used here are not good enough to make any conclusions.

Preface

This is a Master's thesis written at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU) from September 2008 to February 2009. The work is a continuation of the work started in an in-depth health informatics project [4] from autumn 2007. It is a cooperation between IDI and the Norwegian Electronic Health Record Centre (NSEP). The supervisor has been Øystein Nytrø, the main advisor was Ole Edsberg and Carl-Fredrik Bassøe has been co-advisor.

I would like to thank Øystein Nytrø for introducing me to the field of health informatics and encouraging me to write my master's within this young and interesting domain. Ole Edsberg has been a great resource for me during the whole process and I have learned much under his guidance, I am very grateful for his help. I would also like to thank Carl-Fredrik Bassøe who contributed with both data (Promed) for the experiments and good advice from a medical perspective. My thanks goes to Anders Grimsmo for providing the second data source (Profdoc) used in the experiments. I thank Arild Faxvaag and Vebjørn Remen who took time to evaluate part of the results from a medical perspective. Finally I would like to thank the people at NSEP for an inspiring environment to work in.

Contents

List of Tables viii

List of Figures x

I Project report 1

1 Introduction 2

2 Background and related work 3
2.1 Definition of relevant concepts . . . 3
2.1.1 Terms used in supervised learning . . . 3
2.1.2 Special events and periods in time . . . 4
2.1.3 Graphical introduction to how FVs are created in this experiment . . . 5
2.2 Strategies to solve main problems . . . 6
2.2.1 Solving MP1 . . . 7
2.2.2 Solving MP2 and MP3 . . . 8
2.3 The three test cases . . . 10
2.3.1 Asthma . . . 10
2.3.2 Diabetes . . . 11
2.3.3 Hypothyroidism . . . 11
2.4 Methods used in machine learning . . . 12
2.4.1 Preprocessing . . . 12
2.4.2 Classification . . . 16
2.4.3 Evaluation . . . 21
2.5 Data mining tools . . . 25
2.5.1 Intellectual property rights and software . . . 25
2.5.2 Weka classes used in this work . . . 26
2.6 Related work . . . 26

3 Materials and Methods 29
3.1 Materials . . . 29
3.1.1 Sensitivity of data . . . 29
3.1.2 Data set: PD-B . . . 30
3.1.3 Data set: PM-B . . . 31
3.2 Overview of general experiment . . . 31
3.2.1 Data flow diagrams for DAS . . . 31
3.3 Experiment overview . . . 37
3.4 E1: Determine time of suspicion . . . 38

3.4.1 Constraints . . . 39
3.4.2 Pseudocode PH selection . . . 40
3.4.3 Calculating the attribute worth . . . 40
3.4.4 Experiment settings . . . 41
3.5 E2: Varying settings when APP ≤ PP . . . 41
3.5.1 Constraints on FV creation . . . 42
3.5.2 Deciding the values of the settings . . . 42
3.5.3 Feature selection method . . . 44
3.5.4 Pseudocode PH selection . . . 44
3.5.5 Experiment settings . . . 44
3.6 E3: Increasing positives when APP ≤ PP . . . 44
3.6.1 Experiment settings . . . 46
3.7 E4: Comparing all classifiers when APP ≤ PP . . . 46
3.8 E5 and E6 . . . 46

4 Results **48**
4.1 Understanding the RIPPER model . . . 48
4.2 E1: Determine time of suspicion . . . 49
4.2.1 Examining the suspicion-indicators . . . 49
4.2.2 Time to suspicion and diagnose . . . 50
4.2.3 Model analysis . . . 59
4.3 E2: Varying settings when APP ≤ PP . . . 63
4.3.1 Effect of varying PP . . . 63
4.3.2 Effect of varying WT . . . 66
4.3.3 Effect of varying NN . . . 69
4.3.4 Effect of varying NA . . . 72
4.4 E3: Increasing positives when APP ≤ PP . . . 75
4.4.1 Increased complexity . . . 75
4.4.2 Too few attributes . . . 75
4.5 E4: Comparing all classifiers when APP ≤ PP . . . 78
4.5.1 Performance . . . 78
4.5.2 Time to build . . . 78
4.6 E5: Varying settings when APP = PP . . . 79
4.6.1 The number of training examples in E2 and E5 . . . 80
4.6.2 Positive indicators for different PP values . . . 81
4.6.3 Positive indicators for different NA values . . . 81
4.7 E6: Increasing positives when APP = PP . . . 84

5 Discussion **87**
5.1 Positive indicators . . . 87
5.1.1 Why are positive indicators interesting? . . . 87
5.1.2 Where do the positive indicators occur? . . . 88
5.1.3 Do positive indicators affect performance? . . . 89
5.1.4 Determining the validness of positive indicators . . . 90
5.2 Invalid SMOTE results because of circular reasoning . . . 90
5.2.1 Methods to ensure valid results for SMOTE . . . 90
5.3 Limitations of data sets . . . 91
5.3.1 Potential value of additional information . . . 91
5.3.2 Reasons for not including additional information . . . 91
5.4 The potential usefulness of good prediction models . . . 92
5.5 What kind of information do we retain? . . . 93

5.5.1 Extracting different information from different periods 93
5.5.2 Dominant periods 94
5.5.3 The information retained in the experiments 94
5.6 Rich FVs and few examples or sparse FVs and many examples? 95
5.7 Choosing evaluation metric for skewed class distributions 95

6 Conclusion **99**

Bibliography **99**

Appendix **102**

A Experimental settings **103**

B Additional results **105**
B.1 E5: Varying settings when APP = PP 105
B.2 E6: Increasing positives when APP = PP 114
B.3 E1: Additional scatter plots 119

C Abbreviations **122**

List of Tables

2.1 Example of inconsistent naming of events, based on PM-B. Frequency: how many events contain the name in PM-B. Original: original name. Normalised: the name after merging. . . . 13
2.2 The confusion matrix. . . . 22
2.3 Presentation of the most important classes used from Weka. . . . 27

3.1 The data sets used in the experiments. . . . 30
3.2 The event types and their elements for the data set PD-B. . . . 30
3.3 The event types and their elements for the data set PM-B. . . . 31
3.4 The experiment plan, presenting an overview of what each experiment tries to accomplish. . . . 38
3.5 Relevant experiment settings for E1. . . . 43
3.6 Each row shows the values used for the variables in one sub experiment, which is part of E2. . . . 45
3.7 Constant experiment settings for E2. . . . 46
3.8 E3 required only one run with the following settings. . . . 47

4.1 The candidates which were marked as strongest suspicion-indicators for asthma. 50
4.2 The candidates which were marked as strongest suspicion-indicators for diabetes. 51
4.3 The candidates which were marked as strongest suspicion-indicators for hypothyroidism. . . . 52
4.4 The number of patients who were suspected of TD on their first visit to the GP. 52
4.5 Positive indicators and their frequency from RIPPER models for all iterations, with respect to asthma and PD-B. . . . 59
4.6 Positive indicators and their frequency from RIPPER models, with respect to asthma and PM-B. Attributes marked with * are suspicion indicators. . . . 60
4.7 Positive indicators and their frequency from RIPPER models, with respect to diabetes and PD-B. Attributes marked with * are suspicion indicators. . . . 60
4.8 Positive indicators and their frequency from RIPPER models, with respect to diabetes and PM-B. Attributes marked with * are suspicion indicators. . . . 61
4.9 Positive indicators and their frequency from RIPPER models, with respect to hypothyroidism and PD-B. Attributes marked with * are suspicion indicators. . 62
4.10 Positive indicators and their frequency from RIPPER models, with respect to hypothyroidism and PM-B. Attributes marked with * are suspicion indicators. 62
4.11 The number of rules, with antecedents in paranthesis, for the RIPPER classifier as the number of positives is increased. . . . 75
4.12 Comparison of classifiers for asthma using PD-B. . . . 79
4.13 Comparison of classifiers for asthma using PM-B. . . . 80

4.14 Positive indicators from E5 and their frequency from RIPPER models for different dataset-disease combinations and PP-values. WT = 31, iteration 3. . . . 82
4.15 Positive indicators from E2 and their frequency from RIPPER models for different dataset-disease combinations and PP-values. WT = 31, iteration 3. . . . 82
4.16 Positive indicators from E5 and their frequency from RIPPER models for different dataset-disease combinations and NA-values. NN = 1, WT = 300, PP = 913 and iteration 3. 83
4.17 Positive indicators from E2 and their frequency from RIPPER models for different dataset-disease combinations and NA-values. NN = 1, WT = 300, PP = 913 iteration 3. 83

5.1 Comparison of RIPPER models in terms of TP, FP, TN and FN. 89
5.2 DR is dominance rank, DOB is date of birth. 94

A.1 The variables in the settings file and what they control. 104

List of Figures

2.1 t_{sus} related to the t_{TD}. 4
2.2 Illustration of how the prediction period and warning time relate to each other and t_{sus}. 4
2.3 Two scenarios which show what the possible relations between actual prediction period (APP) and prediction period (PP). 5
2.4 Simplified overview of process prior to creation of FVs. 6
2.5 Illustration of how a FV is created. 6
2.6 Comparison of the general iterative development model and how it is realised in S1. 9
2.7 The approximation of t_{sus} should improve with each iteration. 9
2.8 Two clusters of minority examples and the k=3 nearest neighbours of example *a*. 16
2.10 The maximum-margin hyperplane in the middle and the two separated hyperplanes on each side. Picture from Wikipedia. 20
2.9 Instances from two classes are separated by three different hyperplanes representing different classifiers. Picture from Wikipedia. 20
2.11 ROC plot illustrating ROC curve of random guessing and some example points. Picture from Wikipedia. 24

3.1 DFD context level of DAS (called system in figure). 32
3.2 DFD level 1, top level processes in DAS. 33
3.3 DFD, process 2, preprocess and analyse data, decomposed. 34
3.4 DFD process 3, make FVs, decomposed. 35
3.5 DFD process 5, preprocess Instances, decomposed. 36
3.6 DFD, process 6, build and evaluate model, decomposed. 37
3.7 A pair of one positive (p) and negative (n) PH. The part of the PH used to make the FV in E1 is marked with grey. 39
3.8 The grey parts of the PH are used to make FV in E2. The two subfigures show how the positive p decides the length of the period from which the features are extracted. 43

4.1 Scatter plots of how quickly GP suspects the TD, iteration 1. 54
4.2 Scatter plots of how quickly GP suspects and diagnoses patients, using data from PD-B, iteration 3. 55
4.3 Scatter plots of how quickly GP suspects and diagnoses patients, using data from PM-B, iteration 3. 56
4.4 The ratio of positives remaining as we require the period start - t_{sus} to be greater than a given number of days. X-axis: number of days, Y-axis: ratio of positives remaining. 57

4.5 The number of positives remaining as we require the period start - t_{sus} to be greater than a given number of visits. X-axis: number of visits, Y-axis: number of positives remaining. 58
4.6 Performance of classifiers for iteration 1. Classifier is evaluated using geometric mean, and it is the PP which varies in each plot. 64
4.7 Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the PP which varies in each plot. 65
4.8 Performance of classifiers for iteration 1. Classifier is evaluated using geometric mean, and it is the WT which varies in each plot. 67
4.9 Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the WT which varies in each plot. 68
4.10 Performance of classifiers for iteration 1. Classifier is evaluated using geometric mean, and it is the NN which varies in each plot. 70
4.11 Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NN which varies in each plot. 71
4.12 Performance of classifiers for iteration 1. Classifier is evaluated using geometric mean, and it is the NA which varies in each plot. 73
4.13 Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NA which varies in each plot. 74
4.14 Performance of classifiers when using SMOTE. Evaluation metric: geometric mean, NA is max, NN varies. 76
4.15 Performance of classifiers when using SMOTE. Evaluation metric: geometric mean, NA varies, NN is 8. 77

5.1 Performance of classifiers using different evaluation metrices when NN varies. All figures are for iteration 3, PM-B and asthma. 97
5.2 Performance of classifiers using different evaluation metrices when NN varies. All figures are for iteration 3, PM-B and asthma. 98

B.1 Comparing the results when APP = PP with APP $\leq$ PP. PD-B, Iteration 3, PP varies and evaluation metric is geometric mean. 106
B.2 Comparing the results when APP = PP with APP $\leq$ PP. PD-B, Iteration 3, PP varies and evaluation metric is geometric mean. 107
B.3 Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the WT which varies in each plot. 108
B.4 Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the WT which varies in each plot. 109
B.5 Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NN which varies in each plot. 110
B.6 Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NN which varies in each plot. 111
B.7 Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NA which varies in each plot. 112
B.8 Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NA which varies in each plot. 113
B.9 Comparing the results when APP = PP (E6) with APP $\leq$ PP (E3) when using SMOTE. Evaluation metric: geometric mean, NA = All, WT = 300, PP = 913, NN varies. 115
B.10 Comparing the results when APP = PP (E6) with APP $\leq$ PP (E3) when using SMOTE. Evaluation metric: geometric mean, NA varies, WT = 300, PP = 913, NN is 8. 116

B.11 Comparing the results when APP = PP (E6) with APP ≤ PP (E3) when using SMOTE. Evaluation metric: geometric mean, NA is max, NN varies. 117
B.12 Comparing the results when APP = PP (E6) with APP ≤ PP (E3) when using SMOTE. Evaluation metric: geometric mean, NA varies, NN is 8. 118
B.13 Scatter plots of how quickly GP suspects and diagnoses patients, using data from PD-B, iteration 2. 120
B.14 Scatter plots of how quickly GP suspects and diagnoses patients, using data from PM-B, iteration 2. 121

Part I

Project report

Chapter 1

Introduction

A disease which is left untreated for a longer period is more likely to cause negative consequents for the patient. Even though the general practitioner is able to discover the disease quickly in most cases, there are patients who should have been discovered earlier. Electronic patient records store time-stamped health information about patients, recorded by the health personnel treating the patient. This makes it possible to do a retrospective analysis in order to determine whether there was sufficient information to give the diagnose earlier than the general practitioner actually did. Classification algorithms from the machine learning domain can utilise large collections of electronic patient records to build models which can predict whether a patient will get the disease or not. These models could be used to get more knowledge about these diseases and in a long-term perspective they could become a support for the general practitioner in daily practice.

Trying to identify signs which a GP have missed is not an easy task, thus we do not expect very good results in terms of performance measures. Identifying some warning signs could be enough to motivate further research. This thesis is written mainly from a computer science perspective, which have affected the prioritisations in this report. This introduction finishes with a brief presentation of the main problems this project will focus on. They are further elaborated in section 2.2.

- MP1: When does a general practitioner first suspect that a patient might have a certain disease?
- MP2: Is it possible to build models which can identify the patients who later are diagnosed with a disease, by using data which precedes the time of suspicion defined in MP1?
- MP3: Which methods are most suitable for identifying these patients?

Chapter 2

Background and related work

This chapter presents background knowledge and previous work done by researchers in the field. Some of the topics which are presented in the background section are the target diseases, the classifiers used and methods which can be used to solve well-known difficulties related to this problem setting.

2.1 Definition of relevant concepts

Names used to describe different concepts used in this report are presented in this section. The terms which are defined in this section do not attempt to describe all the possible terms which occur in the report, but provides the most important ones so that it is possible to understand new terms. We also encourage the reader to use the list of abbreviations in appendix C.

This report is a continuation of the work done in a previous project, described in [4]. It is often necessary to refer to this report. Instead of citing it formally every time, we refer to it as *preceding project* or similar expressions like; *the work preceding this study.*

2.1.1 Terms used in supervised learning

Supervised learning is one field within the machine learning paradigm. Machine learning is about how machines can extract knowledge from data[1]. The task in supervised learning is to learn a *mapping function*, which maps input objects to correct output. The correct mapping is attached to each input object, so it is possible to evaluate the learned mapping function. These pairs of input objects and correct output form what is termed the *examples*, *instances* or *cases*. Depending on whether the output is a continuous value or discrete categories, supervised learning is divided into regression or classification, respectively.

In this work we perform *binary classification*, classification where there are only two *categories* or *classes*. Since it is very common that one wishes to build classifiers which separate a special group from the rest, the special group is denoted the *positive* class, while the other class is the *negative* class. The examples which belong to these two classes can be specified by the names *positives* and *negatives*, respectively. In the case where there is a class imbalance, such that

[1] The terms knowledge and data should be interpreted according to the Data Information Knowledge Wisdom (DIKW) chain, presented by Ackoff [3].

Figure 2.1: t_{sus} related to the t_{TD}.

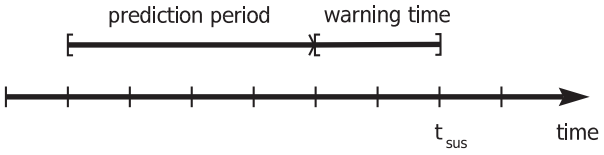


Figure 2.2: Illustration of how the prediction period and warning time relate to each other and t_{sus}.

the number of examples in one class outweights the other, we use *majority* and *minority* class to emphasise this fact.

Each case can be viewed as a structured array which consists of *feature/attribute*-value pairs. Each feature is a variable which describes the case, the value represents how this specific case evaluates with respect to the feature. Features can have values which are continuous, discrete or binary. In this project, we only have binary attributes, thus the possible values are true or false. Every case has a *class attribute*, which contains the information about which class this case belongs to. The set of all feature-value pairs which represents one case, is denoted a *feature vector*.

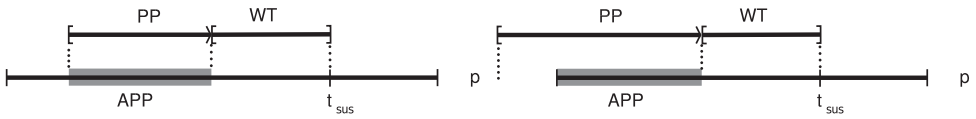
A set of feature vectors, the training examples, are given to a *learning algorithm* with the more specific name *classifier* in the classification setting. The classifier can use the examples to build a model which is the manifestation of the mapping function; it maps examples to classes.

2.1.2 Special events and periods in time

Throughout this report, there are a few concepts which need a clear definition. *Target diagnosis* or *target diseases* denote the diseases we focus on in this work and one target disease is abbreviated TD. The first time a TD is given by a general practitioner (GP) is denoted t_{TD}, *time of TD* or ToTD. The TDs are described in section 2.3. *The time of suspicion*, t_{sus}, is the time when the GP suspects the TD for the first time. Figure 2.1 illustrates the time axis for a patient history (PH) with t_{sus} and t_{TD} marked, showing their relative positions to each other. The beginning of the PH has always time = zero. A PH for a patient who has the TD is denoted a positive PH, while a patient without the TD is a negative PH.

The task of making predictions and giving warnings before the target event occurs is a general problem which must be handled regardless of the specific problem domain. It is therefore not surprising that terms have been made for this problem already. Weiss and Hirsh describe *the event prediction problem* [43] in practically the same way as we have done. They speak of a *prediction period* (PP) preceding the *target event* and a *warning time* (WT), which is the time from the end of the prediction period to the time when the target event occurs. In this case the target event is one of the TDs.

They argue that the WT is used to ensure that the prediction is made in time to be of use, while the size of the PP must be constrained in order to make correct predictions. It is necessary to



(a) APP = PP. (b) APP < PP.

Figure 2.3: Two scenarios which show what the possible relations between actual prediction period (APP) and prediction period (PP).

have a WT; there is no point in predicting something which has already happened. They also state that length of the WT and the size of the PP depend on the problem domain.

Figure 2.2 shows how the PP and WT relate to each other. The PP defines the interval of the PH which is used to build a representation/ feature vector for this patient. All events within a PP are used. The WT is defined to be the interval between the end of the data window and TD. The start and endpoints of these two periods are always located on a day, since the granularity of the PH is days. The end point for a WT is inclusive. The start point of a PP is inclusive, while the end point is exclusive. This can be seen from the figure as well.

In situations where the beginning of the PP is outside the PH, the full capacity of the PP is not used. This scenario is shown in figure 2.3b; there is only a partial overlap between the PH p and the PP. We introduce the term *actual prediction period* (APP) which is the period in the PH which is overlapped by the PP. APP is maximised when there is full overlap, APP = PP (figure 2.3a), and minimised when there is no overlap, APP = 0. APP can also be between 0 and PP as has already been shown in figure 2.3b.

We also define a term which will be used when discussing negative PHs; the *quarantine period* (QP). The QP is anchored at the end of the PH for negative PHs. The events within the QP cannot be used when building a feature vector. The motivation for this is that the patient could develop the TD in the time shortly after the PH ends, thus it should have been treated as a positive PH. By only using the part of the PH which is prior to the QP, we reduce the risk of false negatives being used as training examples.

Since the WT and the QP seem very similar, we will point out the main differences to distinguish the concepts. WT are always used with respect to positive PHs, while QP is only used for negative PHs. WT is used to make the problem of building a good model harder, while the QP reduces the risk of including patients who might get the TD in the future.

2.1.3 Graphical introduction to how FVs are created in this experiment

A feature vector is the representation of one training example, which is used by a learning algorithm. It consists of a set of attributes-value pairs, which describes properties of each unique training example. In this setting, each feature vector describes whole or part of one patient history (PH). The data sets described in section 3.1 contain many PHs, where each PH consists of one or more time-stamped events. Figure 2.4 shows that the PHs are first labelled as positive or negative in b), then the major events t_{sus} and t_{TD} are identified for the positives in c), subsequently the prediction period (grey area) is defined according to WT and PP for the positives in d). The final subfigure shows how the positives are paired up with one or more

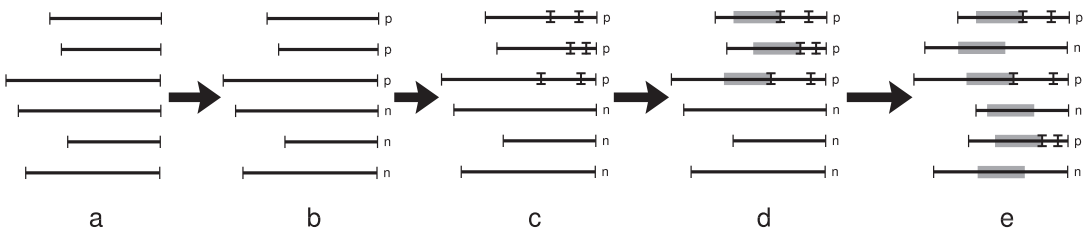


Figure 2.4: Simplified overview of process prior to creation of FVs.

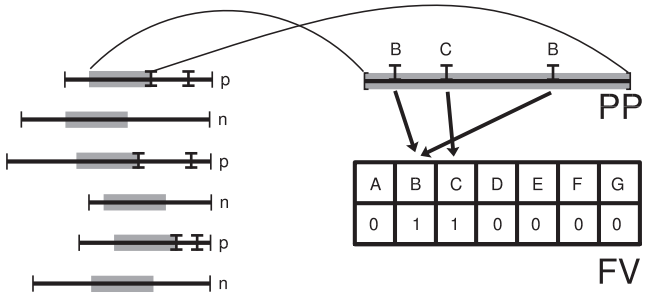


Figure 2.5: Illustration of how a FV is created.

negative examples and the prediction period is defined for the negative examples depending on which positive they are paired up with.

Figure 2.5 shows how a prediction period (PP) is used to create a feature vector (FV). Notice that event B occurs two times in the PP, but it is marked the same in the FV as event C which occurs only one time. We are only interested in knowing if the event occurs in the PP or not.

2.2 Strategies to solve main problems

This section gives a rough explanation of how we attempt to solve the main problems in this project and why it is important to solve them. Even though they were stated in chapter 1, they are repeated in the list below. The strategy to solve main problem 1 (MP1) is presented separately in section 2.2.1, while we collect the strategies for solving MP2 and MP3 in section 2.2.2. These strategies are implemented as experiments which are presented in chapter 3.

- MP1: When does a general practitioner first suspect that a patient might have a certain disease?
- MP2: Is it possible to build models which can identify the patients who later are diagnosed with a disease, by using data which precedes the time of suspicion defined in MP1?
- MP3: Which methods are most suitable for identifying these patients?

2.2.1 Solving MP1

The ambition of this strategy is to estimate t_{sus}, *the time of suspicion*. Given t_{sus}, it is possible to build more valid models based on the historical data prior to t_{sus}.

Motivation for determining the time of suspicion

The main motivation for knowing the time of the first suspicion, is that the usefulness of a predictive model is to a large degree dependent on how early it can alert the GP. A trivial example is that it is not much use in alerting the GP about a disease, if the GP already has diagnosed the patient with this disease. It would be better if a model could give an alert before the GP is certain. Perhaps it would be even better if the alert is fired before the GP even suspects the disease? Solving MP1 is therefore a prerequisite for solving MP2.

There are important distinctions between giving alerts before or after the GP has a suspicion. The project [4] preceding this study found that GPs start the treatment of symptoms which are highly correlated with the disease before they set the diagnosis. This treatment includes giving special medication and taking laboratory tests which are relevant for the disease. The models which were built to predict the disease used these medications and tests as its most predictive features. However, the model only captured knowledge which the GP already had, thus it did not contribute with new information. One could use such a model to see if it agrees with the hypothesis which the GP has. This would be useful if the GP is uncertain and ready to accept advice based on similar cases.

If a model could give the alert before the GP suspects, the alert would convey new information. The GP could then direct his attention toward the disease pointed out by the model, in order to see how the hypothesis given by the model can be confirmed or dismissed. The consequence is that the medical treatment of the patient can start earlier.

Since the ambition of the project was to give warning to the GP about a possible condition before she suspected it, these models were not valid in that context. The initial assumption about when a GP first suspects the condition must therefore be revised, in order to accommodate the fact that the GP has a suspicion prior to the time when she sets the TD. In order to formulate this assumption, we must determine a good approximation of t_{sus}.

Iterative approach to estimate the time of suspicion

We present Figure 2.6b, which shows the iterative process used in this project. t_{sus} is estimated by iteratively identifying the events which are *suspicion-indicators*; events that indicate that the GP has suspected the TD for a patient. Initially, the set of suspicion-indicators (sus-ind. in the figure) is empty. Process 1 uses the suspicion-indicators to define t^*_{sus}, which is the current estimate of t_{sus}. Recall from the concept definitions in section 2.1.2 that t_{sus} is the first day in PH when an event in suspicion-indicators occur. t^*_{sus} is found separately for each patient. t^*_{sus} is used to constrain the scope of PP in process 2, which is all events from the PH prior to t^*_{sus}. In the same process, we compute the correlation between each distinct event and the TD. The result is a ranked list of events with respect to correlation. The most correlated events, a *candidate set*, is given to a group of field experts (GoFE) in process 3. They decide to which degree the candidates are well-known suspicion-indicators based on their medical experience. The filter in process 4 only allows candidates which are graded as most certain indicators to pass through the filter and are subsequently added to the set of suspicion-indicators.

Figure 2.7 shows the expected effect on t^*_{sus} as we iterate through this cycle. Since the set of suspicion indicators is initially empty, t^*_{sus} will be at t_{TD} in the first iteration. When the first iteration is completed, we assume that some candidates were accepted into the set of suspicion-indicators. Given that a patient has at least one event which is a member of the suspicion-indicator, t^*_{sus} will be shifted to an earlier time in the PH. Eventually, no candidates will be accepted into the set of suspicion-indicators and this will terminate the iterative process. The set of suspicion-indicators will then indirectly provide the best possible estimation of t_{sus} which we can get using this method.

By comparing our specific iterative model to the more general model[2] shown in Figure 2.6a, one difference is that there is not any *requirements-*, *analysis & design-* and *implementation* processes in our specific model. Process 1 and 2 can be mapped to the *testing* process as they are controlled by the existing set of suspicion-indicators. The evaluation process corresponds to process 3; the candidates are scrutinized by a group of experts which uses their domain specific knowledge to categorise the strength of the suspicion-indicators. The planning step could be mapped to process 4, as it makes a plan for the next iteration by making a final decision as to which candidates should be included.

Section 3.4 gives a more detailed explanation of how t_{sus} is estimated.

2.2.2 Solving MP2 and MP3

MP2 and MP3 are closely related; in order to determine the best methods for identifying patients with TD (MP3), it must be possible (MP2). To determine if it is possible to identify these patients at all, more results from different methods give us a better foundation to answer the question. Therefore we use the same strategies to handle these problems.

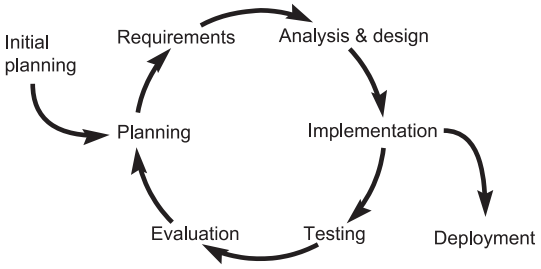
It is important to remember that any conclusions we make with respect to MP2 and MP3, depend on the correctness of our estimate of t_{sus}, which is the answer to MP1.

Motivation for MP2 and MP3

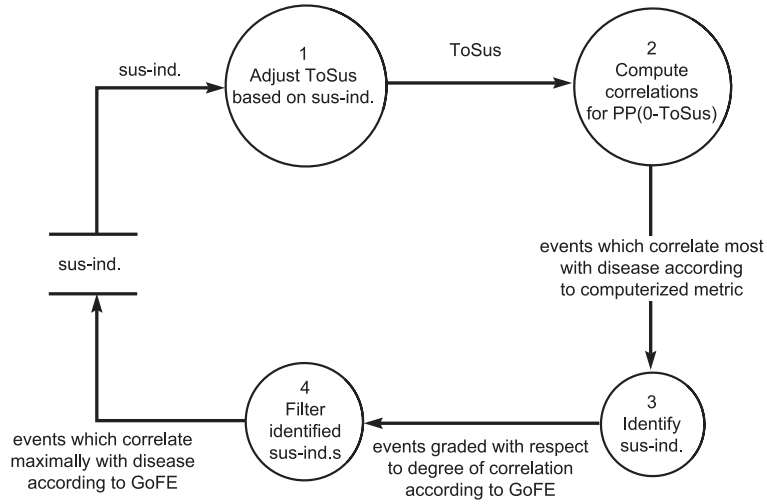
Giving an anwer to MP2 is important in determining whether one should continue research on the topic or not. If we could answer *yes, it is possible to build models which can identify the patients who will get a given disease before the GP*, it is much easier to motivate further research. There would also be better chances that one could make a decision support system which could assist the GP some time in the future. On the contrary, a negative answer would not necessarily mean that further research would be futile, but one might consider doing things differently the next time.

Determining the best methods for building prediction models in our setting is also an important problem to solve. Even though there are claims that one method is better than the other, one must always consider the context; what is best in our case? In addition, different methods perform differently depending on the data, thus the choice of methods should be data-driven [23]. In addition to evaluating different methods, we also must evaluate how to best represent the problem to the methods which build the models. This might be a far more important task than choosing one classification algorithm instead of another; if the problem representation is not representative, there might not be any information to build models on.

[2]Based on figure used in Wikipedia article about iterative and incremental development: http://en.wikipedia.org/wiki/Iterative_and_incremental_development



(a) The general model.



(b) The specific model used to estimate t_{sus}.

Figure 2.6: Comparison of the general iterative development model and how it is realised in S1.

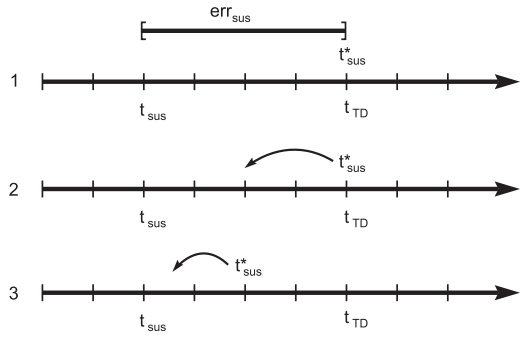


Figure 2.7: The approximation of t_{sus} should improve with each iteration.

We use different algorithms for classification, in order to see whether there are certain classifiers which are well suited for the task. It is not prioritised to optimise the performance of each classifier by tuning settings, we assume that standard settings are good enough. Variables which we will investigate are the sizes of the different periods described in section 2.1.2, in order to find out whether there are certain periods in the patient histories which are more relevant than others. It is also interesting to examine how much information we should include in the representation of each patient, therefore we will investigate the effect of varying the amount of information about each patient.

Two strategies will be used in the evaluation; evaluation based on different performance metrics and manual evaluation of the models themselves. The last strategy is more laborsome than the first, but it provides a way to see how the classifiers "think", which we cannot see from numbers. Both strategies are necessary in order to get overview and understanding of the results.

In short, the overall strategy to solve MP2 and MP3 is to analyse the problem as broadly as possible, since we do not have any grounded assumptions about what is the optimal way to do it.

2.3 The three test cases

We have chosen to use three diagnoses as test cases for our methodology. All diseases have in common that early treatment can reduce the negative effect of the disease on the patient, because the diseases themselves evolve gradually. In addition there were enough data on these patient groups in our data sources. We will give a short introduction to the characteristics of each disease below. All information is from the Norwegian Electronic Handbook for Physicians [22], unless it is stated otherwise. Part of the text about diabetes is reused from last years project [4].

2.3.1 Asthma

Asthma is defined to be a *chronical inflammatory disease* [5]. The airways are inflamed and are hypersensitive to certain substances. When these substances are inhaled, the muscles around the airways constrict, narrowing the passage where the air flows. In addition, excessive amounts of mucus line the inside of the airways, making the airway channel even narrower.

Current estimates are that 6-10% of the Norwegian population have symptoms of asthma, while the number of patients who have gotten a diagnose is 2-5%. These numbers are rising in all of Scandinavia [2]. The most important risk factors are atopic diseases in family [42], the working environment and bad air quality, for example caused by smoking [24]. Physical strain, viral infections, animals, chemicals are other factors which can provoke the disease [31]. Thus there are both genetic and environmental factors which affect the development of the disease.

The *hygiene hypothesis* claims that asthma and other allergic reactions are indirectly caused by a lack of exposure to infectious agents, microorganisms and parasites during childhood. This prevents the immune system from developing the normal defence mechanisms, thus making the person more susceptible to this class of diseases [17]. The increase in persons with asthma in Scandinavia fits nicely into this hypothesis, since focus on better hygiene has been a part of the development.

Many of the relevant medicines are drugs from the ATC-group R03: *drugs for obstructive airway diseases*. It is essential to educate the patient so that she can take proactive measures in order to prevent asthma attacks.

2.3.2 Diabetes

There are two main categories of diabetes patients; those who need regular intake of insulin (type I), and those who do not (type II). In this study, we focus on the type II patients. We will use the term *diabetes*, meaning type II diabetes unless specified otherwise. Diabetes patients suffer from chronic hyperglycemia, which means their levels of glucose in the blood is too high. The cause of this state is a combination of insufficient insulin production and insulin resistance [6]. The risk of getting diabetes increases with high age, overweight [37] and physical inactivity [32]. People with a body mass index above 35 increases the risk of getting diabetes with 93% [15]. If close family members have the disease, the risk of getting diabetes during life is 40%. But it is still mainly the life style which decides whether one gets the disease or not.

Patients with type I diabetes may experience dangerous situations with too much or too little glucose in the blood, called hyperglycemia and hypoglycemia, respectively. Type II patients are in a state of chronically hyperglycemia because of insulin resistance and reduced production of insulin.

The diagnosis is set if glucose tests show results above the given thresholds and if symptoms are present. If the symptoms are not present, two consecutive tests must give values above the threshold.

The treatment is mainly to alter the lifestyle of the patient, through more physical activity and eating healthier food. This is typically food with lower glycemic index, that is food which does not result in a fast increase in the blood glucose level. Medication should not be given immediately. Most of the drugs in the ATC-group A10A and A10B can be applied in the treatment of diabetes. Both groups aim at controlling the glucose level in the blood. A10A contains insulin- and insulin analogs drugs. Insulin analogs are insulin drugs which have been altered using genetic engineering techniques, resulting in insulin drugs which have specialized absorption rates in the body. A10B contains drugs which lower the glucose level in the body, with the exception of insulin drugs which are placed in A10A.

2.3.3 Hypothyroidism

The definition of the disease is a condition with low production of the thyroid hormones thyroxine (T4) and triiodothyronine (T3) in addition to low concentration of these hormones in the blood. There are three organs which work together in order to accomplish this; the hypothalamus produces thyrotropin-releasing hormone (TRH) which causes the pituitary gland to produce thyroid stimulating hormone (TSH), which finally stimulates the thyroid gland to produce T4 and T3. A malfunction in any of these organs could result in hypothyroidism, and the disease is divided into two variants depending on which organ failed; primary- or secondary hypothyroidism. If the thyroid gland fails it is called primary hypothyroidism, while a malfunction in the pituitary gland or the hypothalamus are labelled secondary hypothyroidism. The primary variant is far more frequent than the secondary, as it accounts for 98% of the cases.

The main causes for primary hypothyroidism include autoimmune thyroiditis[12], treatment with radioiodine[18] and external radiotherapy [7]. Autoimmune thyroiditis is an inflammation

of the thyroid gland (thyroiditis), caused by the immune system of the body itself (autoimmune). This is a property which often is inherited [16], thus genes and close family play an important role[26]. Radioiodine is used in medical therapy in order to treat cancer in the thyroid gland, but the treatment can damage the organ to the extent that it affects the production of thyroid hormones negatively. External radiotherapy could also damage the thyroid gland with the same consequences.

Important risk factors for the primary variant are other autoimmune diseases such as diabetes and coeliac disease, during and after pregnancy, Down's syndrome, multiple sclerosis and certain medicament such as Amiodaron.

The disease often develops gradually over long time. Since there are no distinct symptoms in the early phase it can be difficult to diagnose it. The levels of TSH and free levels of T4 in the blood are measured in order to set a diagnose. Primary hypothyroidism can be temporary [8], thus it can be necessary to take several measurements over time to rule out that possibility. Some patient groups are monitored more closely than others. Newborn can have the disease, most often because of iodine deficiency [21]. If left untreated, the disease causes serious damage on the central neural system, thus newborn are screened for the disease in Norway. Patients which have had radioiodine- or radio therapy are also screened after therapy because of the increased risk for hypothyroidism.

2.4 Methods used in machine learning

Machine learning can be divided into three phases; preprocessing, classification and evaluation. This section describes the most important methods used in this project.

2.4.1 Preprocessing

The preprocessing phase focuses on preparing the data for the classification. We describe a method for feature aggregation, several methods for evaluating single attributes which are used to do feature selection and a technique for generating synthetical examples called SMOTE.

Feature aggregation

During the first iteration of experiment 1 (see section 3.4 it was discovered that there was more than one name for a large amount of events. This section explains the problem and the action taken to resolve it.

Problem of inconsistent naming The data can be said to be structured as it is extracted from the data base of the PROFDOC- and PROMED journal systems. There is still a problem with inconsistencies, because the same event has different names in the same data set. Table 2.1 illustrates the problem; in the original-column there are many different ways of expressing that some kind of glucose-test has been taken. Some of these names are remarkably similar; *s GLUKOSE I.F.* and *GLUKOSE ikke.f.* are most likely denoting the same medical test. On the other hand, *s Glukose FAST* and *GLUKOSE ikke.f.* are not identical, but they are both some kind of glucose-test.

The origin of the inconsistencies is most probably caused by different persons interacting with the journal systems, using slightly different names for the same event. These inconsistencies

Frequency	Original	Normalised
23500	s Glukose 2	glucose
21561	s Glukose 1	glucose
17644	U-GLUKOSE	glucose
16100	S-GLUKOSE 2	glucose
13669	S-GLUKOSE FAST	glucose
11068	s Glukose FAST	glucose
10848	eb Glukose KL.	glucose
9931	s GLUKOSE	glucose
6036	GLUKOSE	glucose
4454	s GLUKOSE I.F.	glucose
4132	GLUKOSE ikke.f.	glucose
3636	s GLUKOSE FAST.	glucose

Table 2.1: Example of inconsistent naming of events, based on PM-B. Frequency: how many events contain the name in PM-B. Original: original name. Normalised: the name after merging.

occur in both the PROFDOC and PROMED data sources, but the problem is much larger for the PROMED data. This is not surprising, since the data is collected from 19 different GP offices. In other words, the number of people who have interacted with the PROMED system is much larger, compared to the same number with respect to the PROFDOC system.

By merging names which are believed to denote the same event, there are both positive and negative effects. Since each distinct event is transformed into an attribute, a reduction in distinct events will also reduce the number of attributes. This is advantageous because it will reduce the computational load when building the prediction models. It is also possible to argue that the merge operations will increase the quality of the models. Imagine that a real world event E is denoted E1, E2, E3 ... En by n different persons. The event is therefore split into n different attributes A1, A2, A3, ..., An which is given to a machine learning algorithm MLA. Each attribute Ai in itself is not frequent enough to be utilised by MLA, but if all the attributes had been represented by a single attribute, the probability that the MLA would use the attribute would increase.

However, this is based on the assumption that we merge events which denote the same real-world event. If this assumption does not hold, the merge operation will introduce noise to the data. This knowledge advocates that one should be conservative when merging events. Still, we argue that one can merge events which do not denote the same real-world event, as long as we adjust the granularity of the event. In table 2.1 *s Glukose 1* and *s Glukose 2* are most likely different events, but on a higher level, they are both *glucose* events. By merging them, we get an event which subsumes both events, thus it is acceptable to do it. We denote the name which results from this operation as the normalised name. In machine learning terminology, this is called *feature aggregation*.

Resolving the inconsistency problem We decided to group names which we assumed denoted the same real-world event. The process of grouping or merging the names was done semi-automatically. The original event-file was the input to the process, we call it EventFile. First of all, a script which counted the frequency of each event name was applied to EventFile.

The resulting event-frequencies were given as input to another script, which made a copy of each name and modified the copy by removing common prefixes, retaining only the first word in each event name, and finally converting the result to lowercase. An example; original: *s GLUKOSE I.F.*, removing prefix: *GLUKOSE I.F.*, retain the first word *GLUKOSE*, convert to lowercase *glucose*. The result was written to a file which was imported into a spreadsheet program (like Open Office Calc), and manual work was done to merge logical equivalent events. The result of this process was a conversion table which was custom made for EventFile. A script used the conversion table and EventFile as inputs and replaced all names in EventFile according to the conversion table. The automatic modification operations done prior to the manual part, made sure many events were already merged. However, the manual part was necessary since the automatic process was not perfect.

Experiments affected by inconsistency problem The issue of inconsistent naming described in section 2.4.1 was discovered based on feedback from the field experts who evaluated iteration 1 in experiment 1: determine time of suspicion. The data set was subsequently modified in order to reduce the problem. Thus the issue with inconsistent naming did not affect any other experiments.

Single-attribute evaluation

The typical machine learning approach is to build a predictive model based on the available training data and then evaluate this model with qualitative and quantitative measures. Knowledge is extracted from the model. However, there are other ways to extract knowledge from training data, without constructing a model first. One can evaluate features with respect to the class feature directly by using *attribute subset evaluators*. A subset consists of one or more attributes of the entire set of attributes used in the feature vectors. The special case where the subset only has one single attribute is termed *single-attribute* evaluation.

These evaluators can give information about which attributes are relevant with respect to the class attribute. In settings where the number of attributes is high relative to the number of training examples, it can be advantageous to reduce the number of attributes. Each attribute increases the dimensionality of the space containing possible training examples, making it harder for the classification algorithm to find the common properties of each class. Irrelevant attributes results in longer time to build the predictive model and greater risk of missing important similarities which should have been in the models.

Thus a single-attribute evaluator can aid us in the task of removing irrelevant attributes prior to the process of building classification models. In this study, it is necessary to remove some of the attributes which are highly correlated with the class feature. The methods used to calculate the worth of a single attribute with respect to the class attribute are presented below.

The chi-squared statistic The chi-squared statistic tests whether a null hypothesis holds or not. The null hypothesis in our setting is that the occurrence of an attribute is independent of the class attribute. Equation 2.1 shows how the chi-squared statistic is calculated. The number n denotes the number of possible outcomes for the class variable, which in the binary classification setting is 2. Thus the equation results in a sum of two terms. Each term determines how closely related the value of an attribute is to each of the possible outcomes of the class variable. A_i, is is the observed frequency of a given attribute and E_i is the expected

frequency of this attribute, given by the null hypothesis.

$$X^2 = \sum_{i=1}^{n} \frac{(A_i - E_i)^2}{E_i} \tag{2.1}$$

Information gain Computes the difference in entropy between the class attribute itself and the class attribute, given the occurrence of the attribute A. Equation 2.2 shows how information gain is computed using the entropy metric.

$$InformationGain(C, A) = Entropy(C) - Entropy(C\|A) \tag{2.2}$$

Symmetrical uncertainty Computes the correlation between the class attribute and a given attribute. It is very similar to *Information gain*, but it normalises the result, so it is always between 0 and 1. Equation 2.3 presents how the Symmetrical uncertainty is computed.

$$SymmetricalUncertainty(C, A) = 2 \cdot \frac{Entropy(C) - Entropy(C\|A)}{Entropy(C) + Entropy(A)} \tag{2.3}$$

SVM classifier This method uses a linear support vector machine to rank attributes based on their weight coefficients [25]. It iteratively produces the list of the best features, picking the worst attributes first, and finishes with the best attribute. For each iteration, it builds a model using the linear SVM and the remaining attributes. It then removes the k attributes which are ranked lowest.

SMOTE - generating synthetic positives

Synthetic Minority Over Sampling Technique (SMOTE) is an algorithm which is used to generate new synthetic examples in order to increase the number of examples belonging to the minority class. It was first presented in a paper by Chawla et al. [20] in 2002. The synthetic examples are generated from existing examples, originals, which are not synthetic themselves. Depending on the desired amount of oversampling, the algorithm determines how many originals it should use in the process and randomly selects these. The next step is to compute the k nearest neighbours for each of the originals. In order to create a synthetic example, SMOTE computes the difference between the original and one of its nearest neighbours with respect to each feature, then multiplies this difference with a random number between 0 and 1 and adds the result to the original. Equation 2.4 shows the equation used to compute the new feature value for the synthetic example.

$$f_{new} = f_{original} + rand(0.0 - 1.0) \cdot (f_{neighbour} - f_{original}) \tag{2.4}$$

Imagining that the examples only have two features, the synthetic example will be placed somewhere along the straight line connecting the original and its neighbour. With n features, the synthetic example will end up somewhere between the original and the neighbour in the (n-1)th dimension. Since SMOTE always adds new synthetic examples between originals, the clusters with minority examples will become denser. However, if one allows the algorithm to use more neighbours than there are originals in one cluster, there is a risk that SMOTE will

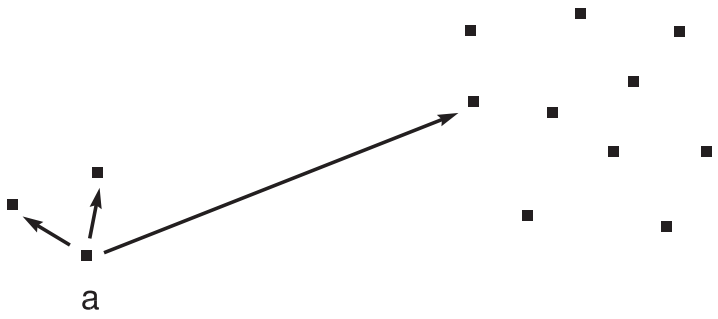


Figure 2.8: Two clusters of minority examples and the k=3 nearest neighbours of example *a*.

create synthetic examples which lie between clusters. Figure 2.8 illustrates this situation; there are two clusters of minority examples (majority class examples omitted for clarity), but the k-value is greater than the number of minority examples in the smallest cluster. Thus the 3rd nearest neighbour for the example *a* belongs to the other cluster. Since there is equal probability of picking any one of the neighbours, there is 33% risk that an example will be added between the two clusters. This would have a negative effect on the classification. Thus it is important to not choose too high k-value in order to avoid the problem.

2.4.2 Classification

This section presents the classifiers which were used in the experiments. The text describing RIPPER, HyperPipes, C4.5, NaiveBayes and SVM was reused from the project [4], since it was written for the same purpose.

RIPPER - a rule learner

Repeated Incremental Pruning to Produce Error Reduction (RIPPER) was developed by Cohen [14]. Tan et. al. [40] describes it is a rule-based inductive learner which is very well suited for use in situations with skewed class distributions. In the binary classification setting, it sets the default class to be the majority class and tries to learn rules for identifying the minority class.

Rules are added to the rule set as long as none of the stopping conditions occur. One of these is that adding the new rule, must not increase the total description length of the rule set by more than d bits. Another is that the error of the rule added, must not exceed 50% when evaluated on RIPPER's own validation set.

The rule-growing process itself uses a general-to-specific strategy. It adds conjuncts to the rule antecedent by selecting the conjunct which maximises FOIL's information gain. When the process of adding conjuncts stop, the rule is pruned if it improves a metric related to the accuracy of the rule.

After the rule set has been grown, k optimisation iterations are performed on the rule set. Each rule is optimised in turn. Two new rules, a *replacement* rule and a *revision* rule are made from the current rule to be optimised. The *replacement* rule is grown and then pruned so as to minimise the total error of the entire rule set with the replacement rule instead of the rule to be

optimised. The *revision* uses the current rule as a starting point, and extends it greedily with more conditions. Using the MDL heuristics, a decision is made whether to keep the original rule or to use the replacement or revision rule instead.

The if-then-rules extracted by the RIPPER algorithm does not always define a consistent and precise model. This can largely be explained by the incremental way the model is built, adding rules iteratively. The extracted rules are highly similar to rules-of-thumb, simple generalisations which often are correct. These rules may represent much experience in a short and comprehensible way, and domain experts often represent their knowledge similarly. As a consequence, most people do not have any difficulties in understanding one rule or small sets of rules. As the number of rules grows, this task grows more difficult, since rule i assumes the negation of all *i-1* rules before it. This dependence on the context can make it difficult to comprehend large rule-sets.

The algorithm scales linearly with the number of training instances, and is able to show its internal knowledge similarly to other rule-based systems. Together with its ability to handle the class imbalance problem well, it is a well suited classifier in this project. JRip is a WEKA implementation of the RIPPER algorithm, which is used in this project.

HyperPipes - extremely fast and simple

The Hyperpipes classifier [44] is a very simple classifier. For each class in the training set, it makes a Hyperpipe which contains each attribute and the bounds for the values of the attribute. As an example, picture a nominal attribute with four possible values (red, green, blue, black). If we have a class RedAndBlueFlowers where all the training examples have values red or blue for this attribute, the RedAndBlueFlowers-Hyperpipe's boundaries for this attribute will be (red, blue). When determining which class a new example should be classified into, it takes the class that most contains the example. Definition of *most contains* is the class which has the highest value for the expression $\frac{count}{nofAttributes}$, where *count* is the number of times the value of the example's attribute was within the class-specific Hyperpipe's boundaries for this attribute. *nofAttributes* is the number of attributes used to represent each example.

The reason to include the Hyperpipes classifier in the experiments is two-fold. Firstly, it is a very fast classifier, using very little time to build a model. The cost of including it is therefore not high in terms of time spent on building models, and it gives a picture of how fast it is possible to build a model when comparing to the more sophisticated classifiers. Secondly, it is a classifier which is able to get rather high sensitivity values, even though the data is imbalanced. Its transparency is limited to the possibility of displaying the attribute boundaries for each class Hyperpipe.

C4.5 - a decision tree

J48 is a decision tree algorithm which is a WEKA implementation of the C4.5 algorithm developed by Quinlan [35]. The C4.5 is an extension of his earlier algorithm ID3. It is biased to make the smallest decision trees possible. The algorithm works by iteratively selecting the attribute which maximises the information gain as the next decision node. The information gain is the difference in entropy before and after a split. Entropy is an impurity measure, which tells us something about the class distribution at a specific node. Low entropy indicates that there are very many of one class compared to the other class at a specific node, the instances are more ordered.

One of the improvements in C4.5 compared to ID3 is post-pruning. After the tree is built, the algorithm replaces less useful branches with leafs. This is a good way of making the tree more general, reducing the risk of overfitting the model.

A tree can be converted to if-then-rules, making its transparency the same as RIPPER. But the graph structure of trees is also easily comprehensible for most people, resulting in two good alternatives for presenting the knowledge to a human. The specific output from WEKA is a bit difficult to read, but this can easily be fixed by attaching a module which can draw more visually appealing trees.

The choice between the two alternatives will often depend on the graphical output devices available at the terminal. Rules are more compact and does not require so much space as trees. However, most terminals today have screens with high resolution, so this is a less important problem today.

K-star - an instance-based classifier

While all the other classifiers which are used in this project spend most of their time building a model in the training phase, K-star [11] is a lazy classifier which does little work until it is time to classify a new example/instance, thus it is categorised *lazy*. Non-lazy classifiers build models which are based on all training examples, a lazy instance-based classifier uses the training examples which are *most similar* to the instance it is going to classify. The general idea is that examples which are similar to the new instance will belong to the same class. The quality of the *similarity metric* which determines how similar two instances are, is therefore very important.

K-star uses information theory to determine the distance between two examples a and b. Example a can be transformed into b by applying a finite number of transformations. A transformation is a change of one value for one of the features. Unless the two examples differ at only one feature, there will be more than one way to transform a to b, thus we have several *transformation paths* $\bar{t}$ which give the same result. K-star defines the difference between a and b to be the sum of probabilities for all transformation paths which lead from a to b. Equation 2.5 presents this sum of probabilities $P^*(b|a)$. When the algorithm is to decide what class label to put on a new instance a, it computes the probability that the instance belongs to each class C, $P^(C|a)$. In order to find the probability $P^(C|a)$, the sum of all $P^*(b|a)$ is calculated for all instances b which belong to C. This is shown formally in equation 2.6. In order to get a deeper understanding of K-star, see Cleary & Trigg [11].

$$P^*(b|a) = \sum_{\bar{t} \in P : \bar{t}(a) = b} p(\bar{t}) \tag{2.5}$$

$$P^*(C|a) = \sum_{b \in C} P^*(b|a) \tag{2.6}$$

NaiveBayes - the benchmark classifier

The Naive Bayes classifier has been used successfully in medical diagnostics for a long time, and is considered a benchmark algorithm which should be tested before attempting to solve the problem with more complex algorithms [30]. It is a statistical method, which makes the

assumption that all attributes are conditionally independent, given the class label. This is written formally in equation 2.7, where Y is the class and X is an attribute vector.

$$P(X|Y = y) = \prod_{i=1}^{d} P(X_i|Y), whereallX = X_1, X_2, ..., X_d \tag{2.7}$$

The algorithm calculates the posterior probability for each class, given the example to classify. The class which has the highest posterior probability is chosen as the class label for the example, see equation 2.8.

$$P(Y|X) = \frac{P(Y)\prod_{i=1}^{d} P(X_i|Y)}{P(X)} \tag{2.8}$$

Since P(X) is a constant term, it is the numerator which determines which class will be chosen. To compute the value $\prod_{i=1}^{d} P(X_i|Y)$, one must estimate the values $P(X_i|Y)$. For categorical values, this is estimated as the fraction of training examples in class $Y = y$ that has the value X_i. Thus the learning process estimates all these conditional probabilities. Estimating continuous attributes is not so relevant in this project, since there are currently no continuous attributes at all. To learn about estimating continuous attributes, see [28].

The independence assumption is often not true, but the algorithm performs very well despite this false assumption. It is also extremely fast, and scales well as the number of attributes increases.

The textual representation of Naive Bayes which is output from WEKA, gives all the information needed to compute the most probable class. But to do so you need knowledge of the algorithm itself, the mathematical skills to do so and lots of time because of the high number of attributes. As a result, the physician will not be able to understand the knowledge as is without the help of expertise.

However, Kononenko has previously shown simple techniques that make the knowledge of Naive Bayes comprehensible [29]. It is possible to show how much each feature contribute to the decision of classifying a new sample to a specific class. Using this method, one can show the influence of individual features for and against the decision of classifying a new sample into a specific class. This way of arguing was similar to the way physicians reasoned when they worked. Thus one can visualise the knowledge in the Naive Bayes model.

SVM -

The advantage with SVMs compared to most classifiers is that they try to minimise an upper bound on the generalisation error instead of focusing too much on the training error. The theory behind this technique is a statistical learning theory called *structural risk minimisation*.

SVM is a classifier which has proven its usefulness in handling imbalanced classes and this is the main reason that it is included in the baseline approach. The success of the SVM depends on a number of settings which should be chosen with care in order to give the best results. In the baseline approach, SVM is applied with the standard settings of WEKA, without any tuning phase. Thus there is no guarantee that the SVM will perform optimally in the baseline approach.

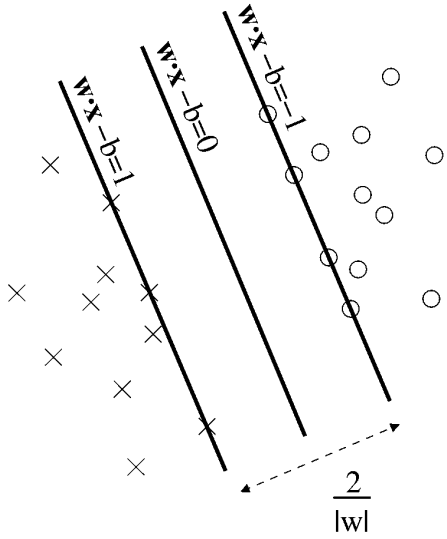


Figure 2.10: The maximum-margin hyperplane in the middle and the two separated hyperplanes on each side. Picture from Wikipedia.

SVMs were first introduced at the COLT conference in 1992 by Vapnik et al and is therefore a relatively young classifier. But most of the principles behind it had existed since the 1960s and it was therefore the idea of putting them all together which was new [38]. Below is an explanation of the basics behind SVMs, starting with SVM for the separable case and ending with the asymmetrical margin SVMs.

Separable data points Figure 2.9 shows a scenario where the instances from the classes are separable and three different hyperplanes which all performs well on the instances shown in the figure. But if the instances shown are only training examples, the three hyperplanes will most likely perform differently on the set of all instances. A SVM classifier constructs a separating hyperplane which tries to maximise the distance to the closest instance from each class, also called the *maximum-margin hyperplane*. As a part of this process, two parallel hyperplanes are constructed along the frontier of each class, see Figure 2.10. Training examples lying on these separated hyperplanes are called *support vectors*.

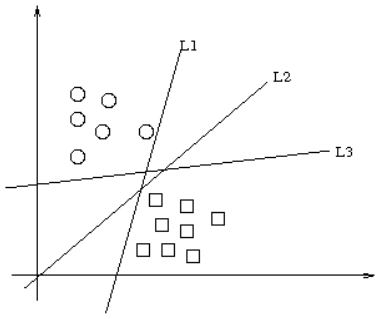


Figure 2.9: Instances from two classes are separated by three different hyperplanes representing different classifiers. Picture from Wikipedia.

Inseparable data points In many cases, the data is almost linearly separable. It is still possible to use much of the same strategy as above, with the use of *slack variables*. These variables allow some instances to be misclassified and measure the degree of misclassification. There is one slack variable ξ_i for each class i. In addition, there are two penalty weights C_i, which regulates the trade-off between maximising the margin and minimising the

error for each class i.

A clean cut between the two classes is impossible, but a hyperplane that separates most of the instances in the two classes can still be built. However, it is now subject to two constraints. The first one is the same as before; maximise the margin to the two separated hyperplanes, the second is to minimise the weighted sum of the slack variables which denote the degree of error connected to misclassified instances. The resulting hyperplane is not the maximum-margin hyperplane, since the hyperplane is adjusted to accommodate the instances which were misclassified.

When SVM the error-weights used with respect to the two classes are equal, $C_{minority} = C_{majority}$, the approach uses *symmetrical margins*. Both classes are penalised equally for misclassified instances. If the classes are unbalanced, one might want to penalise errors of one class more than the other, in order to increase the sensitivity with respect to the minority class. The solution is to increase the error-weight for the minority class, which will cause the decision boundary to be closer to the majority class, allowing more examples from the minority class to be classified correctly. When the error-weights are unequal, $C_{minority} \neq C_{majority}$, it is called *asymmetrical margin support vector classification*.

The mathematics required to find the classifier decision function includes solving the Wolfe dual problem and finding Lagrange multipliers for each training example. For details, see Cohen et al. for a compact presentation [13] or the book authored by Shawe-Taylor and Nello Cristianini [38] for a more elaborated approach.

2.4.3 Evaluation

Evaluation is a key in understanding the quality of the models which are built by the classifiers. Section 2.4.3 presents the methods used for evaluation, while Section 2.4.3 explains the four categories where we put the examples used in testing. Finally, different performance metrics are presented in section 2.4.3. The text in section 2.4.3 is from the project [4] last year.

Evaluation methods

Since most classifiers try to build models which minimises the error over the training examples, it is a requirement to use examples it has not used in the training process when doing the evaluation. The aim of the evaluation is to determine whether the classifier has been able to generalise or whether it has overfitted the model to the training data. Two evaluation strategies which fulfil the requirement of not reusing training examples in the testing are presented below; the *holdout method* and *cross-validation*.

The holdout method divides the data in two disjoint sets; a training set and a test set. The classifier builds its model using the examples in the training set and is evaluated on the examples in the test set. The test set could consist of 50 - 10 % of all examples and these are not available in the training process. This could have a negative effect on the model quality, since the performance of a classifier usually increases with the number of training examples. Another problem is that there can be large variations in the test result, depending on which examples are in the test set and vice versa. One can circumvent this problem by using random subsampling and average the results. The holdout method is then repeated several times, and each time examples are assigned randomly to the two sets.

Cross-validation remedies the weakness of the previous strategy; it is able to use all examples for both training and testing. The method randomly partitions the data into k subsets in the

		Actual value	
		P	N
Prediction outcome	P	TP	FP
	N	FN	TN

Table 2.2: The confusion matrix.

first step. One subset is used for testing and the examples in the remaining k-1 subsets are used for training. The results are collected and the process is repeated, this time with a new subset as the test set. In turn, all subsets are used exactly once for testing and k-1 times for training. By summing the results of all k tests, we have an approximation of the quality of a classifier which had used all data for training.

A challenge with cross-validation is that we cannot view the representation of the model we have evaluated, because it is actually k models. In the experiments conducted in this work, each classifier was evaluated with cross-validation, but the representation of the corresponding model was derived from a classifier which used all data in the training. For practical purposes, we assume that the model which is presented is the average of the k models which were used in the evaluation.

TP, TN, FP and FN

A test case which is used to evaluate the quality of a trained model can be classified as either belonging to the target class (positive) or not belonging (negative), this is the *prediction outcome*. Since we perform supervised learning, the *actual value* of the case is known, thus it is straightforward to determine if the decision of the model was correct (true) or not (false). Each test case is therefore labelled as one of the following; true positive (TP), true negative (TN), false positive (FP) and false negative (FN). The count of all cases which are true positive is also denoted TP and similarly for the other categories.

The *confusion matrix* shown in table 2.2 shows how the combination of prediction outcome and actual value decides which category a test case belongs to. As an example, if the prediction outcome is positive (P) and the actual value is negative (N), the case is a false positive. When evaluating the quality of a model, it is counted how many test cases fall into each of these four categories and these sums are presented in the confusion matrix. These categories form the building blocks when defining other evaluation metrics such as accuracy and sensitivity.

Performance metrics

The *sensitivity* and *specificity* of a classifier is how good it is at identifying positives and negatives, respectively. If the specificity is 1.0, it means that all negatives were correctly classified as such.

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.9}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2.10}$$

Positive predictive value (PPV) gives the proportion of correctly classified positive cases, while the Negative Predictive Value (NPV) is the proportion of correctly classified negative cases.

$$PPV = \frac{TP}{TP + FP} \tag{2.11}$$

$$NPV = \frac{TN}{TN + FN} \tag{2.12}$$

Equation 2.16 presents the evaluation metrics used in this project. Accuracy is a metric which is the ration of correctly classified cases. When the number of test cases from each class is quite equal, this metric can give a fairly good picture of the performance of the classifier. However, it is not well suited for situations where there are far more cases from one class with respect to the other. If there were 10 000 test cases and the number of positives and negatives was 100 and 9 900 respectively, a simple classifier could say that all cases are negative. In terms of accuracy, it would be correct in 99% of the cases. The classifier would still be useless, since it is not able to find a single positive example.

The metric geometric mean uses both sensitivity and specificity to calculate the quality. It gives them equal importance and as such it rewards models which are quite balanced in their performance on both positives and negatives. A concrete example; two models A and B has the same sum of sensitivity and specificity, model A has a sensitivity which is much greater than its specificity, while these two values are equal for model B. Then model B will get a higher geometric mean value than A.

F-measure makes it possible to give higher priority to the positive cases, which is useful when the positive class is greatly outnumbered. But it does not take the negative class into account in the evaluation, which makes it less suited to give a overall performance metric for the classifier. α is a number between 0 and 1 used to decide the relative importance between PPV and sensitivity. If $\alpha = 0$, F-measure simplifies to sensitivity, while if $\alpha = 1$ it becomes PPV.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.13}$$

$$Geometric_Mean = \sqrt{sensitivity \times specificity} \tag{2.14}$$

$$F - measure_\alpha = \frac{PPV \times sensitivity}{\alpha sensitivity + (1 - \alpha)PPV} \tag{2.15}$$

$$CWA = weight \times sensitivity + (1 - weight) \times specificity \tag{2.16}$$

Cohen et al. proposed the mean class-weighted accuracy (CWA) [13] as a metric which combined the best of the geometric mean accuracy and F-measure. It uses sensitivity and specificity like geometric mean, and adds weights which are used to regulate the importance of each class. The formulas used for F-measure and CWA in Equation 2.16 are simplifications of the more general formula (not presented), made for the binary classification problem. This approach allows us to control the relative influence of sensitivity and specificity on the evaluation metric, making it possible to customise the weights to the problem at hand.

ROC curves and ROC AUC This section about ROC AUC is from last years project [4]. Receiver operating characteristic (ROC) curve has become a popular visualisation of a classifier's quality. It is a graphical plot of the classifier's sensitivity along the y-axis versus (1

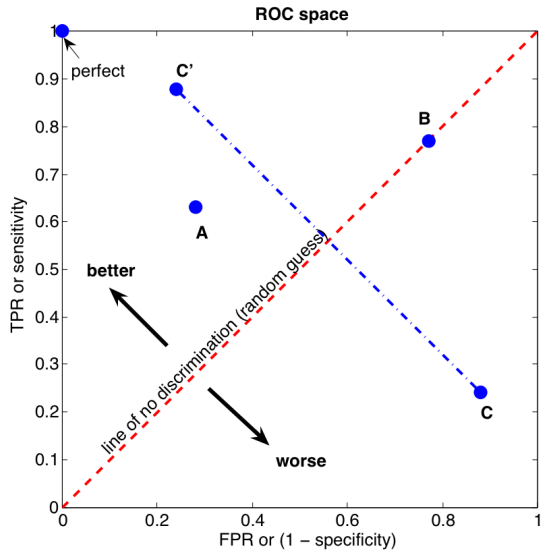


Figure 2.11: ROC plot illustrating ROC curve of random guessing and some example points. Picture from Wikipedia.

- specificity) as the x-axis. Integrating the area under the curve (AUC) gives the ROC AUC which is often used to evaluate the performance of a classifier.

One trained model of a classifier results in a single point in the ROC space. One way of obtaining a ROC curve, is to vary one of the parameters of the classifier. A typical parameter which is used, is the discrimination threshold. This threshold determines when a case should be classified as positive or negative, in the setting where we perform binary classification. A classifier like Naive Bayes calculates the probability that a case belongs to a certain class, then it uses a discrimination threshold to determine if this class should be classified as a positive or negative example. Varying such a parameter makes it possible to draw a ROC-curve using only one model. However, not all classifiers use probability values to determine the class. Then it is necessary to vary other parameters in order to get a ROC curve.

Figure 2.11 shows a ROC plot which can be used to explain some of its basic properties. The y-axis is labelled *TPR or sensitivity*, while the x-axis is labelled *FPR or (1 - specificity)*. True Positive Rate (TPR) is a different name for sensitivity, while False Positive Rate is the number of False Positives divided by the number of True Negatives, which is equivalent to (1 - specificity). A perfect classifier with 100% accuracy, that is 1.0 in sensitivity and specificity, would have coordinate (0,1), which is shown by a blue dot in the upper left corner. The diagonal line through the plot is the ROC curve of a classifier which is doing classification through guessing, the line of no discrimination. Classifiers above this diagonal are better than random guessers, while classifiers below are worse. It should be noted that by taking the negation of a classifier's decisions, its ROC coordinate is mirrored across the diagonal. The four blue dots labelled A, B, C and C' denote the coordinates of four classifiers, where the classifier C' is the negation of classifier C.

Making a ROC curve by varying a parameter, is a good way of visualising which is the best value for it, with respect to the costs (specificity) and benefits (sensitivity). In the case of medical diagnosis, the physicians might require that the specificity of the classifier be above 95% and then one can use the ROC curve to find the model which gives the highest sensitivity

given this constraint.

2.5 Data mining tools

There exists several learning environments which provide algorithms from the ML domain, which eases the task of programming the code needed for the experiments. Weka [44] and RapidMiner (formerly known as YALE [33]) are two of the most popular ones, both implemented in Java. They are very similar in many aspects, but have different aims and thus they have different approaches to managing their intellectual property rights (IPR) as described below. In the end of this section we explain why Weka was chosen in this work and the most important tools which were used from the Weka toolkit.

2.5.1 Intellectual property rights and software

IPR is about protecting intangible assets such as creations of the mind, both artistic and commercial, providing legal rights exclusively to the creators. Copyrights, trademarks, patents and trade secrets are all different types of IPR, giving the creator alternative ways of protecting their creation. These rights have different applications; copyright is typically used for artistic creations, while patents and trademarks are more common with respect to commercial creations, thus they are called *industrial properties*. A patent gives the patent holder the right *to prevent or exclude others from making, using, selling, offering to sell or importing the invention*[3].

This paragraph gives a short version of how software has been protected, gathered from [1]. When software was defined as separate from hardware in USA in the 1970s, it was defined to be mathematical algorithms, which cannot be patented. Copyright was seen as the best protection for software, which was formalised in 1980 when software was included in copyright law. In 1981, the *Diamond v. Diehr* case opened up for patenting software, since the software was part of a physical process which was judged patentable. But the software in itself was still not patentable. But in the *Alappat* case in 1994, a physical machine with software was stated to be different from a machine without the software. This ruling together with the *State Street Bank v. Signature Financial Group* decision (1998), which decided that business methods can be patented, finally opened up for patenting software. During the same period, copyright protection had lost most of its power since it only protects the exact code as it is originally. A rewrite of the program would in many cases circumvent the copyright protection.

Anyone who writes a software program gets the copyright to it[4]. The owner can grant a user the license to a copy of the program if the user accepts the terms of the license agreement. Software licenses are divided into proprietary and open/free software licenses. The proprietary licenses only grant a few well defined rights to the end user, most of the rights remain with the owner. Free licenses give the user more rights; access to the source code and the right to modify it. But in order to distribute work built on freely licensed work, one must attach the source code and use the same license conditions as the original work. GNU General Public License (GPL)[5] is a popular standard license agreement which is often used with open source code.

[3] Definition from Wikipedia: Patent, accessed 20th of January 2009.
[4] Norwegian law on copyright; Åndsverkloven, Â§1
[5] Current version of GPL is 3; http://www.gnu.org/licenses/gpl-3.0.txt

Comparison of license terms for Weka and RapidMiner

Weka has been developed by the University of Waikato in New Zealand since 1993 and is freeware released under the GPL. The source code is open and free to change by anyone who wants to, but the license requires you to make derived works of Weka available to others under the GPL.

The work on YALE (Yet Another Learning Environment) was begun by the Artificial Intelligence Unit of University of Dortmund in 2001 and has now evolved into RapidMiner which is a commercial company which provides both software and additional services to its customers. There are two versions of RapidMiner; the Enterprise edition which is for the commercial market and the Community edition which is free to use in contrary to the other version. Both editions are available as open source code under the GNU Affero General Public License (AGPL). The AGPL is similar to the GPL in the way that any developer can obtain RapidMiner and modify the code, as long as they make the full source code of their own application available for users to modify and redistribute under the AGPL. In addition RapidMiner provides a proprietary license for those who wish to embed RapidMiner in their work without making the source code for their work publicly available. Thus they have a dual licensing system.

2.5.2 Weka classes used in this work

Weka was chosen as the learning environment for this project. It had the desired methods available, in addition to the fact that the code from last year's project also was accommodated to Weka. Since RapidMiner offers most of the methods found in Weka, it is most likely that it has more to offer than Weka. There was, however, no need for this extra functionality. This project uses Weka 3.6 and the most important classes which are shown in Table 2.3.

2.6 Related work

This section presents some articles documenting work which can be related to this project. We have not been able to find articles describing exactly the same setting as our project. Thus it could be that our iterative approach to define t_{sus} ,and require that we only use information prior to this time, is a novel approach.

There are numerous ways to predict a disease using data mining techniques. Certain problems reappear in several of the articles which is about predicting diagnoses, one of them is how to select the best subset of features. There are two good reasons for reducing the number of features; increasing the predictive performance and reducing the computational load [45]. 11 different methods for feature selection were examined in [23], among them were 3 based on mutual correlation and 7 methods previously used to select genes. Based on their results from predicting women with breast cancer, the best feature selection algorithm was a mutual correlation based filter made by Tsai and Chiu [41]. But even though this algorithm performed best in this case, it was concluded that the best combination of feature selection and modeling method depends on the data. This project uses a simple feature selection scheme, which is described in section 3.5.3. Since our feature selection method is considerably more simple than the ones used in [23], our project would most probably benefit from using more sophisticated feature selection methods.

A strategy for countering imbalanced data sets is to oversample the minority class, and possibly downsample the negative class at the same time. Even though these methods are simple, they

Class	Type	Comment
HyperPipes	Classifier	Extremely fast classifier, no documentation on origin.
JRip	Classifier	The RIPPER algorithm [14].
J48	Classifier	The C4.5 algorithm [35].
NaiveBayes	Classifier	The NaiveBayes algorithm [28].
LibSVM	Classifier	A wrapper class [19] for the libsvm tools [9] .
KStar	Classifier	The K* algorithm [11].
Randomize	Filter	Randomizes the training examples.
Remove	Filter	Removes a range of attributes from the data set.
SMOTE	Filter	The SMOTE algorithm [20].
ChiSquaredAttributeEval	Att. eval.	see section 2.4.1.
InfoGainAttributeEval	Att. eval.	see section 2.4.1.
SVMAttributeEval	Att. eval.	see section 2.4.1.
SymmetricalUncertAttributeEval	Att. eval.	Calculates the symmetrical uncertainty, see section 2.4.1.

Table 2.3: Presentation of the most important classes used from Weka.

can improve the situation considerably. In the preceding project it was shown that resampling the distribution of positives and negatives improved the overall performance as the distribution became more balanced. However, when oversampling the positive training examples, there is a risk that the models are overfitted to the training data.

One approach which increases the number of positives by creating synthetic cases, SMOTE (see section 2.4.1 is presented by Chawla et al. [20]. An article which presents the application of SMOTE is Taft et al. [39]. Their task was to improve models which predicted events of adverse drug reactions (ADEs) in an emergency care unit at a hospital. The problem was that there were only 0.348 examples with ADE per 100 examples without it, thus they had a typical case of class imbalance. They tried other sampling techniques such as oversampling using AdaBoost, but these attempts did not improve the results of the base classifier. By using SMOTE to increase the number of positives, they got results for C4.5 which showed an improvement in the true positive rate from 0.32 without SMOTE to 0.62 when adding 200% extra synthetical positives generated using SMOTE. The effect on the decision tree models was a greater granularity than before, with more branches and leaves. This made it possible to find other risk factors for ADE, which could be verified by domain experts. An important contribution in the study was that they showed that using SMOTE avoids the overfitting problem which is typically encountered in oversampling techniques, by showing that the distribution of features was unchanged by SMOTE.

Park et al. claims that case-based reasoning (CBR) has been less used in medical research than other AI methods [34]. Case-based reasoning does not try to generalise over all training examples (cases) by building a model; it is an instance-based learning paradigm which tries to find similar cases when determining the class of a new case. One of the simplest algorithms

within this paradigm is k-nearest neighbours. Given an unknown case to classify, it uses a distance metric to find the k cases which are most similar, then the class which is most frequent among its neighbours is given to the new case. Even though CBR has not been used much in the medical domain, it should be well suited for the area since clinical problem solving is often case-specific and there are many examples available [36].

A study in Estonia by Remm & Remm [36] showed the usefulness of a case-based machine learning and prediction system for estimating the risk of enterobiasis. Estimating the risk for belonging to a class is closely related to predicting the class, as most classification algorithms calculates the probability that a case belongs to the different classes. The other methods in the study were quite different from ours, but we include it because it helps illustrate the alternatives we did not or could not use. The data for the study was collected using questionnairies, an approach which involves more work than using data which is already collected which is the case in our project. An advantage of collecting the data yourself is that it is possible to gather the information which is believed to be relevant. We had no say in how and what data should be collected and as a result there is more irrelevant data and the data is not so fine-grained. Their approach of using a case-based learner is very different from the learning algorithms we use, except K-star. While our classifiers builds a model which tries to incorporate the information from all examples, case-based learning and the K-star algorithm do not make a model to cover all examples. K-star is a very simple case-based learning method

[27] has made the MUTARC algorithm, mining unexpected temporal association rules (UTARs). They are trying to identify unexpected, infrequent episodes (UIE) which could cause adverse drug reactions (ADE) of the simplified form UIE -> ADE. They use a brand new interistingness measure for events called *residual-leverage* and use case-based exclusion to eliminate frequent events which are assumed irrelevant for the task. They get a ranked list based on this interstingness-measure, showing which events are the most important with respect to special events. This is related to thir project, since one could picture the diagnose we're trying to predict as an ADE, and then we could try to find UIE which could cause ADE. One problem is that this assumes that the diagnose is caused by single events, while we don't know if several events act together. We also do not know if the causes for diseases are infrequent and unexpected.

Chapter 3

Materials and Methods

This study has utilised information from different sources and these are presented in the beginning of this chapter. The following sections are used to describe data sets, data mining tools and the experiments themselves.

3.1 Materials

The data for this study originates from the electronic journal-systems used at different GP offices in Norway. More specifically, the data originates from two different commercial journal-systems; PROFDOC and PROMED, thus the two main *data sources* are named PROFDOC- and PROMED-data. The PROFDOC data originates from a single GP office, while the PROMED data is collected from 19 different offices. It is therefore not unexpected that there is roughly five times as many events in PROMED as in PROFDOC, see table 3.1. However, the extra data comes with a cost; the inconsistency problem is much greater for the PROMED data than PROFDOC.

Since the data is from real patients, the sensitivity of the data is described in section 3.1.1, while the two data sets are presented in section 3.1.2 and 3.1.3. In section 2.4.1 the inconsistency problem is presented and it is explained how it was handled.

3.1.1 Sensitivity of data

Access to the data is granted through The Norwegian EHR Research Centre (NSEP), where this study is conducted. The data which is considered sensitive is kept on servers which are only connected to a secure, isolated LAN (SIL). Isolated means that it is not in any way connected to the Internet, thus one can only access the data when physically connected to the SIL.

The list below presents a categorisation of data based on the sensitivity, the definitions were copied from the internal guideline on NSEP. According to these guidelines, only SG0 data can be copied or stored outside SIL. As table 3.1 shows, the data sets used in this study were classified as SG1 data, thus all data handling was done while connected to the SIL at NSEP.

1. SG0 Anonymous: data cannot be used to indirectly identify patients.
2. SG1 De-identified: data can be used to identify patients indirectly.

ID	Source	SG	# data rows	# patients	AST	DIA	HYP
PD-B	PROFDOC	SG1	1,225,385	11,853	645	434	254
PM-B	PROMED	SG1	6,889,655	112,420	3,165	1,793	1,217

Table 3.1: The data sets used in the experiments.

		Event types		
El. no.	El. name	Diagnosis	Prescription	Test
1	PatientID	x	x	x
2	Time of event	x	x	x
3	Type of event	x	x	x
4	Type-specific el.	ICPC2-code	ATC-code	Test name
5	Test result			1.0 or 0.0

Table 3.2: The event types and their elements for the data set PD-B.

3. SG2 (unnamed): data contains free-text fields which have been subject to an imperfect, automatic process of removing direct identifiers.
4. SG3 (unnamed): data contains free-text fields which have *not* been subject to any automatic deidentification process.

There are mainly two laws in Norway which are relevant when using data extracted from EHR; The Personal Data Act (Personopplysningsloven) and The Personal Health Data Filing System Act (Helseregisterloven). Personopplysningsloven regulates the use of personal data while Helseregisterloven states how personal health data should be stored and dealt with.

One *data set* is extracted from each data source. Table 3.1 shows basic information about each data set. The extracted data originates from well-defined fields in the data sources, thus there is no free text data. The table also shows how many rows are in each data set and the number of distinct patients which are described. There is one difference between the two sets, which is that PD-B provides information whether the test result was abnormal or non-abnormal, PM-B does not.

3.1.2 Data set: PD-B

This is the same data set which was used in the preceding project [4]. It is presented again, in order to make this report more self-contained. It is a collection of events, where each event is one of three types; diagnosis, prescription or test. Each event has a number of fields, some are common and some are specific for each event. Table 3.2 shows the structure of the events. The name of the data set is an acronym for ProfDoc Basic.

The first element, *patientID*, is a unique identifier for each patient in the data set. It is generated for the data set and is not an identifier which originates from the EPR. *Time of event* describes the time this event occurred, relative to the first event this patient has in the data set. The first event for every patient occurs at time 0. *Type of event* is one of the three event types; *diagnosis*, *prescription* or *test*. Element four is context-sensitive, meaning that

		Event types		
El. no.	El. name	Diagnosis	Prescription	Test
1	PatientID	x	x	x
2	Time of event	x	x	x
3	Type of event	x	x	x
4	Type-specific el.	ICPC2-code	ATC-code	Test name

Table 3.3: The event types and their elements for the data set PM-B.

the content depends on the type of the event. It is essentially a name for the event. Diagnosis-events are named using ICPC2-codes, prescription-events use ATC-codes, while the test names are not standardised. Only test events have a fifth element, and that is the test result which is either 1.0 (abnormal result) or 0.0 (non-abnormal result).

3.1.3 Data set: PM-B

The name of the data set is an acronym for ProMed Basic. The fields describing each event are identical to PD-B, with the following exceptions. The patient ID of PM-B is the result concatenating the institution with a patient number. The patient number is not unique across institutions, thus these two must be concatenated to give a unique patient ID. In addition there is no field describing the test results for PM-B, in contrast to PD-B.

3.2 Overview of general experiment

All experiments are conducted using the Disease Analyser System (DAS) which is programmed in Java, using the machine learning environment Weka (see section 2.5.2 for more information). This section describes DAS using data flow diagrams (DFDs) to describe the data flows and the processes which modify these in DAS. DAS is only used as term in this section, other sections only refer to DAS as system or similarly.

There exists different notations for making DFDs, we have chosen to follow the Yourdon and Coad notation. There are only four symbols which are used; external entities are marked with rectangular boxes, a file or a database is marked with two horizontal lines, processes that takes data as input, modifies it and outputs data are marked with circles and the data flows themselves are arrows indicating the direction of the data flow. We interpret objects which hold frequently used data as databases in the following figures.

3.2.1 Data flow diagrams for DAS

Figure 3.1 shows the context of DAS; inputs are the data set(s) and the configuration which control the experiment. The evaluation results, the actual models and properties of the patient histories (PHs) is written to files which are put in the results folder. The information in the configuration file controls the dynamic properties of the experiment such as the size of the PP, WT and QP, how many attributes should be used to describe each patient, whether to generate additional positive examples synthetically and many other parameters. All of these

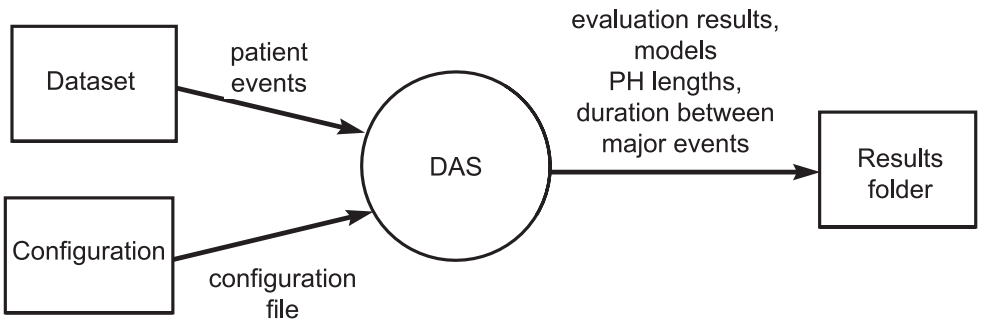


Figure 3.1: DFD context level of DAS (called system in figure).

settings are described in section A, while the data sets which are used are presented in section 3.1.

By decomposing the *system*-process in figure 3.1, the top level processes of DAS emerge. Each process has a unique number which indicates the order of the processes in time. Processes 1, 2 and 3 are performed without the use of Weka, while 4, 5 and 6 are carried out with assistance from Weka. DAS creates all the arff-files first, then it processes them one by one.

First of all the configuration file is read and the parsed settings are stored in memory. The events from the data set are read in process 2, preprocessed and analysed. The preprocessing collects the events for each patient into a patient history object (PH) and all PHs are stored in the *Patient data* object. In the analysis step, the PHs are fetched from Patient data and metadata about each PH is collected, then the metadata is written to the PH and put back in the store. Process 3 uses the PHs and metadata in order to make the feature vectors (FVs), according to the constraints from the experiment settings. The created FVs are stored as arff-files, a file format defined by Weka.

Since arff-files can be imported into the Weka environment, it is now possible to use the tools made available through Weka. Process 4 imports an arff-file into an Instances-object, which holds the collection of all training/test examples. It is necessary with a second preprocessing step in 5; settings from the experiment settings controls the actions of this step. Finally, the preprocessed Instances are used to train and evaluate classifiers in process 6. The results from this process, together with results extracted in processes 2 and 3, are put into the results folder.

In order to get a better view of the more complex processes 2, 3, 5 and 6, they are decomposed into sub processes which are shown in figures 3.3, 3.4, 3.5 and 3.6 respectively.

Sub process 2.1 in figure 3.3 transforms the patient events into patient histories, while 2.2 identifies and analyses all patients with respect to which diagnoses, tests and prescriptions they have taken, whether they have had one of the target diagnoses and if so; when did it occur the first time and when did the first event indicating suspicion occur? This information is denoted metadata and is coupled to the PHs and put back into the patient data repository. It is reused in process 3 and in sub process 2.3, which finds the lengths between the major events for each patient. The periods which are measured are between the following events; PH beginning and end, in addition to t_{sus} and t_{TD} for each of the TDs the patient has. These periods are measured in days and the number of visits and the results are written to file and stored in the results folder.

The decomposition of process 3 in figure 3.4 shows how the PHs are filtered in sub process 3.1.

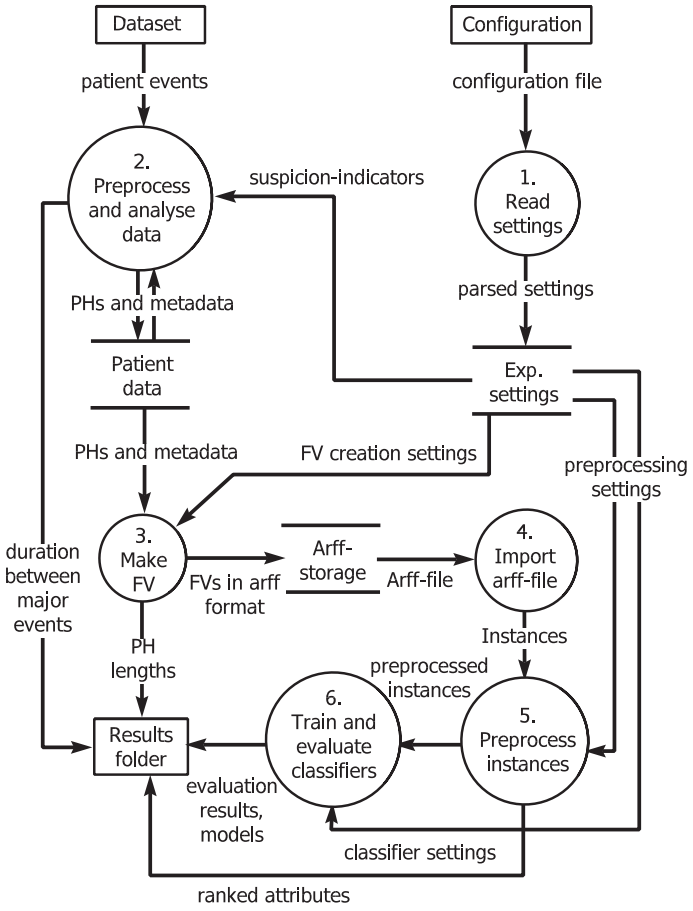


Figure 3.2: DFD level 1, top level processes in DAS.

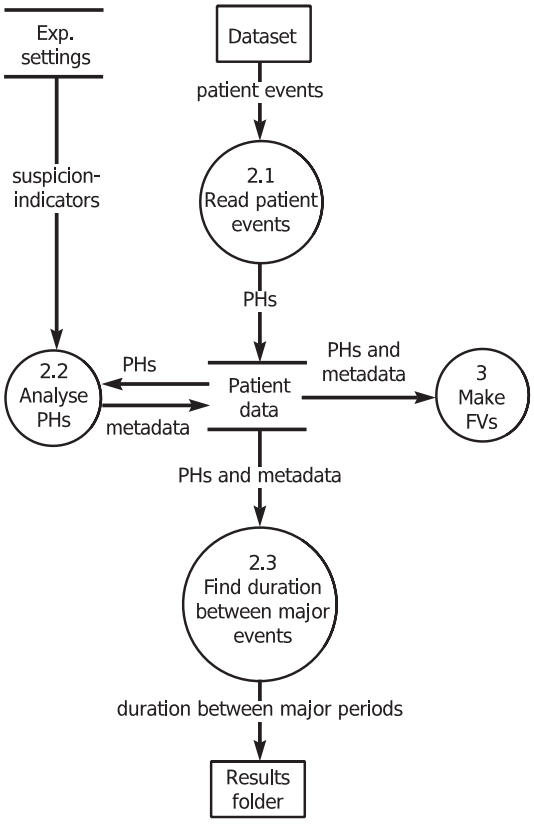


Figure 3.3: DFD, process 2, preprocess and analyse data, decomposed.

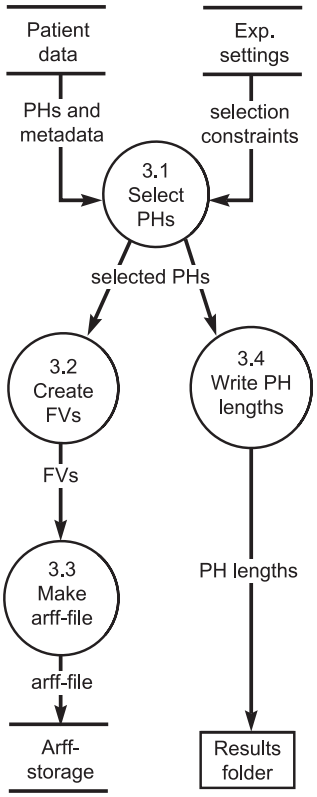


Figure 3.4: DFD process 3, make FVs, decomposed.

The selection constraints regulate which PHs are selected and which part of each PH which should be used. More details about the selection process is given in section 3.3 under each experiment description. The selected PHs are input to sub processes 3.2, where the feature vectors (FVs) are created. A FV template with all possible events (diagnoses, prescriptions and tests) is compared with the events occurring in the PP of the PH; if an event occurs in the PP, it is checked off in the template. The resulting FV holds the information about which events this patient has had in the PP. These FVs are not in a format which can be read by Weka, so it is necessary to convert them into the arff-format. This task is performed in sub process 3.3 and the arff-file is written to a separate folder for arff-files. Sub process 3.4 calculates the length of the PHs and writes the information to the results folder.

The details of the preprocessing which is done after the Instances have been illustrated in figure 3.5. A number of actions are listed; applying SMOTE, randomising, attribute ranking and selection. Not every experiment performs all these preprocessing tasks. The *SMOTE flag* can turn SMOTE functionality on or off, the *pipe flag* decides whether the experiment should terminate by storing the ranked list of attributes in the results folder, or pipe instances and the ranked list of attribute to 5.4. The number of attributes which are retained in 5.4 is also controlled. The randomisation step is the only preprocessing step which is conducted in every possible scenario.

The final part of the experiment is the process where the models are built and evaluated,

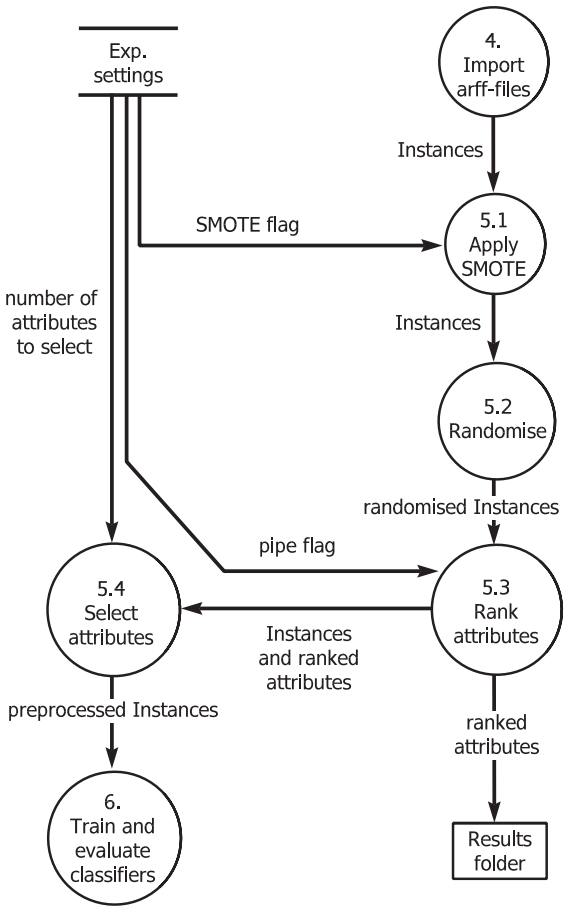


Figure 3.5: DFD process 5, preprocess Instances, decomposed.

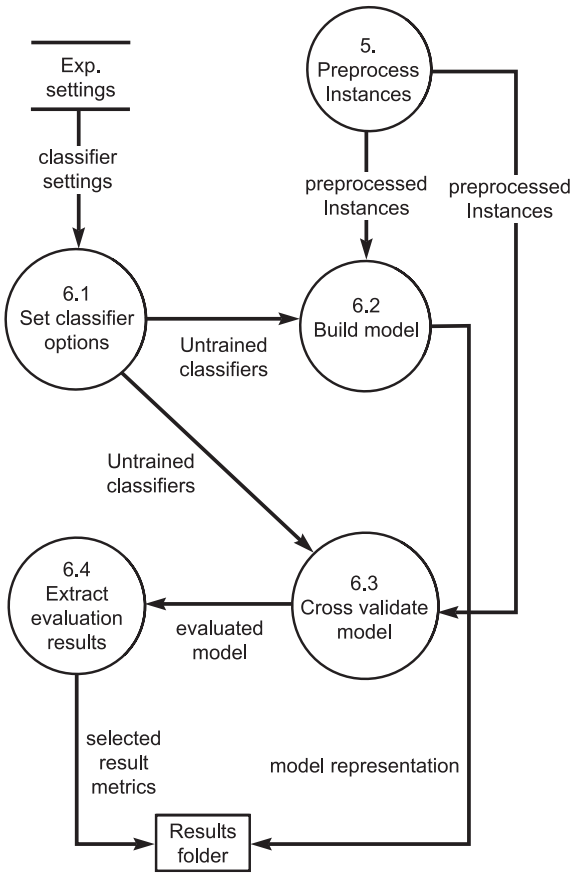


Figure 3.6: DFD, process 6, build and evaluate model, decomposed.

which is presented in detail in figure 3.6. The experimental settings are used to control which classifiers should be initialised in sub process 6.1 and these classifiers are input to sub process 6.2 together with the preprocessed Instances from process 5. Each classifier builds one model on all Instances available, and the representation of this model is stored in the results folder. The next step is to use cross-validation to evaluate the quality of the model in sub process 6.3. We use cross-validation with 5 folds. It is not the model which is built in 6.2 which is evaluated, but a set of models which together are approximately the same. See section 2.4.3 for a more detailed explanation of the cross-validation procedure. The different evaluation metrics are extracted in sub process 6.4 and these are structured and stored in the results folder.

3.3 Experiment overview

This section presents the experiments and their details, a rough overview of all experiments is given in table 3.4. All experiments are given an identifier *E(xperiment) + number*, and these identifiers are used when referring to the experiments in the rest of the report. Each experiment has a purpose which is linked to the main problems (MPs) presented in the introduction 1. E1 is conducted in order to estimate t_{sus} while the other experiments focus on how the most optimal model can be built.

The experiments are also divided with respect to the actual prediction period (APP) used. In E1, E5 and E6, we require that APP must be equal to PP. This is the only difference between E2 and E5, and E3 and E6. APP was described in section 2.1.2.

All experiments except E4 use the classifiers Hyperpipes, NaiveBayes and Ripper to build models. This standard set of classifiers are used because the time needed to build them is relatively fast compared to the other classifiers presented in section 2.4.2. The Ripper algorithm makes rules which enables us to easily inspect how it classifies examples. The model of NaiveBayes is a bit more tedious to analyse, while Weka does not output a representation of the Hyperpipes model. In E4 we use both the standard classifiers and the remaining ones, in order to see if there are great differences.

3.4 E1: Determine time of suspicion

As described in section 2.2, the aim is to determine t^*_{sus}, which is an approximation of t_{sus}. The list below gives a formal description of how the entire experiment is conducted (ToSus-star is t^*_{sus}, ToSus is t_{sus}). Figure 2.6b illustrates this process while figure 2.7 shows the expected result of this iterative approach.

1. **Initial condition:** The set of *suspicion-indicators* is empty, thus the initial approximation of ToSUS, ToSUS-star, is equal to ToTD.
2. **Iteration *i* begins:** Adjust ToSus-star based on the current suspicion-indicators; ToSus-star is the time when the first event from *suspicion-indicators* occurs.
3. Calculate the worth of each attribute with respect to the class attribute, using a set of attribute evaluation metrics.
4. Compute the average ranking of the attributes across the results from the different evaluation metrics.
5. Select the X attributes which have the highest average ranking.
6. Sort these attributes firstly by type of event, then name. Remove diagnosis-events.
7. Present these X attributes for a GoFE. For each attribute, the GoFE must answer the following question; *How probable is it that the presence of this event (test/prescription) is an indication that the GP has suspected the target disease?*. The alternatives are; very unlikely, unlikely, neutral, likely, very likely.

ID	MP	APP	Description
E1	MP1	$= PP$	Estimate t_{sus}.
E2	MP2, MP3	$\leq PP$	Observe effect of varying parameters.
E3	MP2, MP3	$\leq PP$	Observe effect of adding positives using SMOTE.
E4	MP3	$\leq PP$	Compare different classifiers.
E5	MP2, MP3	$= PP$	Observe effect of varying parameters.
E6	MP2, MP3	$= PP$	Observe effect of adding positives using SMOTE.

Table 3.4: The experiment plan, presenting an overview of what each experiment tries to accomplish.

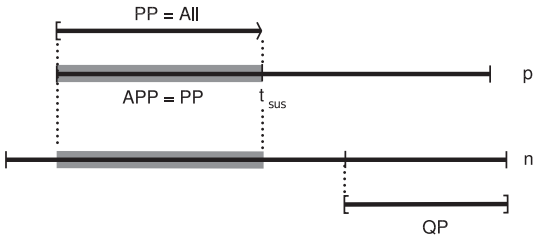


Figure 3.7: A pair of one positive (p) and negative (n) PH. The part of the PH used to make the FV in E1 is marked with grey.

8. If the GoFE answered *very likely* to one or more features, proceed with task 10.
9. If the GoFE did not answer *very likely* for any features, this is a strong indication that we have a set of suspicion indicators which gives an accurate estimate of ToSus, thus the loop can be terminated.
10. Put the features where the GoFE answered *very likely* in the set called *suspicion-indicators*.
11. **Iteration *i* ends:** Go to step 2.

Steps 2 - 4 are carried out using the automated experimental process described in section 3.2, the remaining steps are manual tasks. We will continue this experiment description with explaining how the automatic part of the process is done.

Process 1 in figure 3.2 imports the settings which control the experiment, these settings are described and defined in section 3.4.4. The second process illustrated in figure 3.3 reads the patient histories (PHs) and analyses the PHs in order to determine t^*_{sus} for each patient. This process corresponds to step 2 in the list. A subtask in the next process, making the FVs (see figure 3.4), is to select the subset of PHs which are to be represented as FVs. Section 3.4.1 describes the constraints under which the PHs are selected. Pseudocode for the PH selection process itself is described in section 3.4.2. The PHs are now selected and one FV is created for each selected PH, the set of FVs is stored as an arff-file. Step 3 - 4 computes the average correlation between each attribute and the class, using the method described in 3.4.3. These steps correspond to process 5.3 in figure 3.5. SMOTE is not applied and the ranked attributes are written to file and put in the results folder, thus no models are built.

3.4.1 Constraints

In this experiment, we use all information available in the positives, which is prior to t_{sus}. This is shown in figure 3.7, where the prediction period covers the positive PH p from beginning to t_{sus}. The PP does not have a fixed value and its length will depend on each patient. Note that there is no WT between the end of PP and t_{sus}, thus WT = 0. The PP cannot be empty, which is specified in constraint 3.4.1. The positives which fulfill this constraint are added to the set $\vec{P}^*_{E1}$. We could have removed those positives which have very short PH prior to the t_{TD}, but they might include information which could identify suspicious events, so we keep them.

Constraint 3.4.1 *There must be at least one event in each positive PH prior to t_{sus}.*

In order to control the balance between the number of positives and negatives which are selected, we find one negative PH n for each positive PH p in $\vec{P}^*_{E1}$, see Constraint 3.4.4. Furthermore, to make sure that the average length of the PP for the positives does not differ from the average length for negatives, we pose Constraint 3.4.2 on n. Figure 3.7 shows the n which has been selected with respect to p. If n fulfils Constraint 3.4.2, it must also fulfill Constraint 3.4.3.

Constraint 3.4.2 *Each negative which is paired with a positive must adjust its PP so it has the same length as the PP of the positive.*

Constraint 3.4.3 *Length of negative PH must be greater than or equal to length of corresponding positive PH plus the size of the QP.*

Negatives which have been successfully paired with a positive are added to the set of selected negatives; $\vec{N}^*_{E1}$. NN = 1 of the selected negatives is duplicated, in order to make sure there are more negatives than positives. The reason is that the Ripper algorithm makes rules for the class with least examples, thus we make it to focus on the positive class.

Constraint 3.4.4 *NofNegPHs = NN * (NofPosPHs + 1), where NN = 1 in E1.*

3.4.2 Pseudocode PH selection

The task of selecting PHs is carried out in processes 2 and 3 in figure 3.2. Algorithm 1 is the high-level pseudocode for this task. Reading section 3.4.1 is a prerequisite in order to get the definitions of the sets used in the algorithm. The functions used in the algorithm are described below.

findAllPositive and *findAllNegative* returns the collection of all positive and negative PHs in the data set, respectively. Functions *remove* and *add* removes and adds the desired PH to the specified COLLECTION. *maskEventsAfter(EVENTS, COLLECTION)* iterates through each PH in the COLLECTION and determines t^*_{sus} using the EVENTS as suspicion-indicators. The events after t^*_{sus} in the PH are masked, thus they are treated as if they were non-existant in the remainder of the experiment. *lengthDays(PH)* returns the length of the unmasked part of the PH, measured in days. *findRandom(COLLECTION)* picks a PH at random from the COLLECTION. The probability of selecting a PH depends on the length in days; the probability of selecting a PH, A, is twice as much as selecting another PH, B, if B is half the length of A. *randomCrop(PH, DISTANCE)* randomly chooses a time interval within PH with length DISTANCE (in days) and masks all events outside this interval.

3.4.3 Calculating the attribute worth

The attribute worth is calculated in order to identify which attributes are most closely related to the class attribute. A list of the most related attributes is then given to a GoFE for them to evaluate which events indicate that the GP suspects the TD. This section describes how the ranked list of attributes with respect to attribute worth is made.

We have decided to use compute three single-attribute evaluation metrics, and make the ranked list based on an average of the results. The evaluation metrics used are *the chi-squared statistic*, *information gain* and *symmetrical uncertainty*. They have already been presented in section 2.4.1. Each of these produce a ranking of the attributes. The final rank of each attribute is

Algorithm 1 Pseudocode for PH selection in E1.

```
{Find all positive PHs and remove part of PH after ToTD}
P⃗ = findAllPositivePH()
P⃗*_E1 = maskEventsAfter(sus − ind, P⃗)
{Find all negative PHs and remove those that are too short}
N⃗ = findAllNegativePH()
for all n in N⃗ do
  if lengthDays(n) − Quarantine < 1 then
    remove(n, N⃗)
  end if
end for
{Match each positive PH with a random negative one}
N⃗*_E1 = newCollection()
for all p in P⃗* do
  {Picks a negative PH at random}
  n = findRandom(N⃗) {Make sure negative PH is at least as long as positive one}
  while lengthDays(n) < lengthDays(p) do
    n = findRandom(N⃗)
  end while{Crop negative PH so it has same length as positive one}
  n* = randomCrop(n, lengthDays(p))
  add(n*, N⃗*_E1)
end for
```

calculated using Equation 3.1. Since we use three different metrics, there is less risk that the bias of one metric will favour special types of attributes.

$$Rank(A) = \frac{Rank_{SymU}(A) + Rank_{X^2}(A) + Rank_{InfG}(A)}{3} \tag{3.1}$$

$Rank_{SymU}(A)$ is the rank of attribute A using the metric symmetrical uncertainty. If this rank is 2, there is only one other attribute which is more closely related to the class attribute than A.

3.4.4 Experiment settings

The relevant settings which were used to control E1 are shown in table 3.5. The entire PH prior to t_{sus} is used since WT is 0 and PP is not constrained. The suspicion-indicators for the three diseases varies through the iterations. Iteration 1 is with no suspicion-indicators, iteration 2 uses the suspicion-indicators found in iteration 1, while iteration 3 uses the suspicion-indicators found in iteration 1 and 2. For a more detailed description of each setting, see table A.1.

3.5 E2: Varying settings when APP ≤ PP

In the experimental setup described in section 3.2, a core of settings together decide which information is given to the classifiers. The aim of this experiment is to see the effect each of these have on the classifiers. The settings we are going to analyse are the iteration (It.), PP,

WT, NN and NA. PP and WT have previously been described in section 2.1.2; they control which part of the PH we include events, given t_{sus}. t_{sus} is controlled by the suspicion-indicators and each iteration has a single set of suspicion-indicators per TD associated with it. Therefore It. represents the different sets of suspicion-indicators. NN is short for the number of negatives per positive example, as defined in Constraint 3.4.4.

The remaining settings which control the experiment are presented in table 3.7. It might seem trivial to include which diseases and data sets to make models for, but they are included for completeness.

3.5.1 Constraints on FV creation

Where E1 consequently used all information prior to t_{sus} when making the FVs, E2 constrains the length of both PP and WT. The difference is evident if one compare figure 3.7 with 3.8a. In the latter case, the WT forces the PP further back in time, this affects how long the positive PH should be prior to t_{sus}. Constraint 3.5.1 states this formally.

Constraint 3.5.1 *Length of positive PH prior to t_{sus} must be greater than WT.*

Note that the PP does not necessarily cover the remainder of the positive PH as it always did in E1, thus APP $\leq$ PP. Figure 3.8b shows an example this situation. Constraint 3.5.2 makes sure that each negative which is coupled to a positive, has the same APP as the positive. Thus it is the APP which controls the size of the period from which we extract data from the negative.

Constraint 3.5.2 *Each negative is to have the same length of APP as its corresponding positive.*

From Constraint 3.5.2 we can derive Constraint 3.5.3; given that APP = PP for the positive, the negative must at least be of length QP + PP in order satisfy Constraint 3.5.2.

Constraint 3.5.3 *Length of negative PH must be greater than or equal to PP + QP*

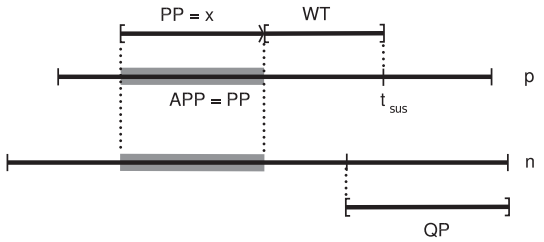
E2 controls the balance between positives and negatives in the same way as E1, thus Constraint 3.4.4 also applies to E2. Since there are few positives available, this experiment tries to use as many positives as possible. Constraint 3.5.1 is the only requirement which positives must fulfill in order to be included as training examples. This constraint follows from the definition of WT, which is a period from which we are not allowed to extract information. As a consequence, the APP (actual prediction period, see section 2.1.2) can vary from 1 day to the maximum value which is PP. The positives not fulfilling this constraint are excluded from the experiment, while the rest are included in the set named $\vec{P}^*_{E1}$.

3.5.2 Deciding the values of the settings

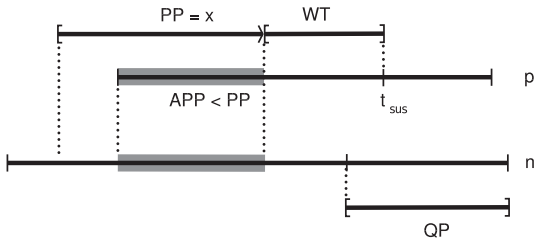
Each row in table 3.6 shows the values for each setting used for one sub experiment and all rows together collect the information we assume is needed to see the effect of each setting. An alternative to this set of sub experiments is to do only one experiment where all values of each variable are included. The problem with the latter approach, is that it generates a larger number of models, thus it takes longer time to complete. Equation 3.2 computes the number of models which is generated in each sub experiment. The function l(X) returns the number

Setting	Value
PP	All
WT	0
QP	730
NN	1
NA	All
SUS_IND_AST	<dynamic>
SUS_IND_DIA	<dynamic>
SUS_IND_HYP	<dynamic>
USE_SMOTE	False
PIPE_CORR_BM	False

Table 3.5: Relevant experiment settings for E1.



(a) A pair of one positive (p) and negative (n) PH. The part of the PH used to make the FV in E2 is marked with grey.



(b) The same amount of data is extracted from both the positive and negative example, even when APP is less than PP for the positive.

Figure 3.8: The grey parts of the PH are used to make FV in E2. The two subfigures show how the positive p decides the length of the period from which the features are extracted.

of values X has in the experiment, *l(Dis)*, *l(Set)* and *l(C)* are the numbers of TDs, data sets and classifiers respectively.

$$NumModels = l(It)l(PP)l(WT)l(QP)l(NN)l(NA)l(Dis)l(Set)l(C) \tag{3.2}$$

We apply Equation 3.2 to the case where all values are to be computed in one single experiment (using the values from table 3.6); $NumModels = 3 \cdot 6 \cdot 4 \cdot 1 \cdot 4 \cdot 5 \cdot 3 \cdot 2 \cdot 3 = 25920$. If we compare this with the sub experiments approach, the number of models for each row is 360, 432, 360, 432, 360, 432, summing to 2 376 models. This is approximately 10% as many models as if we were to compute models for all combinations of the variables. A potential problem is that we might have rejected combinations of settings which would produce better models than our chosen set of combinations. However, this experiment will hopefully give results which can aid us in finding the optimal combination which gives the best classifier with respect to performance.

3.5.3 Feature selection method

A very simple method is used to select a set of features with a desired size. The first step is to generate a list of ranked features using the procedure described in E1, section 3.4.3. The final step is to retain the number of features we need from the top of the list.

3.5.4 Pseudocode PH selection

Algorithm 2 presents pseudocode for the PH selection in E2. It is similar to that of E1, but the different constraints cause some differences. The conditions in the if-statements have been altered, and there is an extra for-loop with a function *maskEventsOutside(PH, LENGTH-PP, LENGTH-WT)*, which masks the events in the PH which are outside the PP, resulting in a PH where all unmasked events can be used to create a FV. The other functions are the same as the ones described in section 3.4.2. There is also one difference in the final for-loop, where each positive can be matched with more than one negative example.

3.5.5 Experiment settings

We divide the settings into two tables; table 3.7 shows the settings which are constant throughout the sub experiments, while table 3.6 shows the values for the setting which vary throughout each sub experiment.

3.6 E3: Increasing positives when APP $\leq$ PP

SMOTE was presented in section 2.4.1 as a sampling technique which generates synthetical positives. This experiment will apply SMOTE in order to see if it has a positive effect on the classifier performance. This experiment is the same as E2, the only difference is the experiment settings.

Recall that SMOTE is used to generate new synthetical positives as described in section 2.4.1. In order to maintain balance between the positives and negatives, we use SMOTE to add synthetical positives until the two classes have the same number of examples. Applying SMOTE

Algorithm 2 Pseudocode for PH selection in E2.

{Find all positive PHs and remove those that are too short}
$\vec{P} = findAllPositivePH()$
$\vec{P^*_{E1}} = maskEventsAfter(sus - ind, \vec{P})$
for all p in $\vec{P^*}$ **do**
  if $lengthDays(p) < WT$ **then**
    $remove(p, \vec{P^*_{E1}})$
  end if
end for
{Mask events outside PP}
for all p in $\vec{P^*}$ **do**
  $maskEventsOutside(p, PP, WT)$
end for
{Find all negative PHs and remove those that are too short}
$\vec{N} = findAllNegativePH()$
for all n in $\vec{N}$ **do**
  if $lengthDays(n) - Quarantine < PP$ **then**
    $remove(n, \vec{N})$
  end if
end for
{Match each positive PH with NN random negative ones}
$\vec{N^*_{E1}} = newCollection()$
for all p in $\vec{P^*}$ **do**
  for $i = 0$ to NN **do**
    {Picks a negative PH at random}
    $n = findRandom(\vec{N})$ {Crop negative PH so it has same length as positive one}
    $n^* = randomCrop(n, lengthDays(p))$
    $add(n^*, \vec{N^*_{E1}})$
  end for
end for

It.	PP	WT	NN	NA
1	913	300	1, 2, 4, 8	All, 1000, 100, 50, 10
1	31, 183, 365, 913, 1825, All	0, 31, 100, 400	1	All
2	913	300	1, 2, 4, 8	All, 1000, 100, 50, 10
2	31, 183, 365, 913, 1825, All	0, 31, 100, 400	1	All
3	913	300	1, 2, 4, 8	All, 1000, 100, 50, 10
3	31, 183, 365, 913, 1825, All	0, 31, 100, 400	1	All

Table 3.6: Each row shows the values used for the variables in one sub experiment, which is part of E2.

in a situation where the classes are balanced have no effect in this system. Thus we must create the class imbalance before we apply SMOTE. Class imbalance is achieved by using NN > 1 when selecting the PHs should be used to make FVs. The FVs are made in process 3 in figure 3.2, while SMOTE is applied after this in process 5. As a result, we increase the number of positives and negatives synchronously.

3.6.1 Experiment settings

In order to observe the effect of SMOTE, it is enough to vary NN. We also choose to vary NA, in order to see if the classifiers perform differently with an increased amount of training examples and varying NA. The other variables in table 3.8 are constant. In addition we reuse the settings for E2 which were presented in table 3.7, except that USE_SMOTE = True.

3.7 E4: Comparing all classifiers when APP $\leq$ PP

This experiment compares the performance of all classifiers presented in section 2.4.2, using the same procedure as E2. The experimental settings of E4 are very similar to E3, the differences are listed below.

- USE_SMOTE = False.
- NA values are 1000, 100, 50, 10.
- Classifiers used are Ripper, Hyperpipes, NaiveBayes, C4.5, SVM and K-star.

The maximum number of attributes used in this experiment is 1000. The reason we do not use all attributes is to speed up the time it takes to build these classifiers. We assume that the attributes which are not used are irrelevant.

3.8 E5 and E6

The remaining experiments descriptions of E5 and E6 can be defined shortly, since they build on the previous experiments to a large extent. E5 is nearly identical to E2 and E6 is nearly identical to E3. The difference is that E5 and E6 require that APP must equal PP, while E2 and E3 only state that APP must be greater than 0. By replacing Constraint 3.5.1 with the

Setting	Value
CLASSES_TO_PREDICT	AST, DIA, HYP
CLASSIFIERS	HyperPipes, NaiveBayes, Ripper
DATA_SET	PD-B, PM-B
MIN_NUM_EVENTS	1
QP	730
USE_SMOTE	False
PIPE_CORR_BM	True

Table 3.7: Constant experiment settings for E2.

more strict Constraint 3.8.1, E5 is identical to E2 and E6 is identical to E3. A consequence is that each positive must at least be of length WT + PP in the period which is prior to t_{sus}.

Constraint 3.8.1 *APP must equal PP for all selected positives.*

It.	PP	WT	NN	NA
3	913	300	1, 2, 4, 8	All, 1000, 100, 50, 10

Table 3.8: E3 required only one run with the following settings.

Chapter 4

Results

The experiments conducted have generated a large amount of results which can be presented in many different ways. It has therefore been a laborsome and difficult task to determine which results to present and how. Experiment E5 was closely related to E2, thus section 4.6 compares the results of E2 and E5, while E2 focuses on how the different parameters affected the quality of the models. The same approach is chosen for experiment E3 and E6 which also are closely related.

We use RIPPER models to investigate what kind of information this classifier uses. In order to have a more precise language, we begin this chapter with a description of how to understand RIPPER models in section 4.1. A last remark is that we did not have time to explain what all the different attributes which occur in the models mean.

4.1 Understanding the RIPPER model

Because RIPPER is one of the models which is most easily understood, it will be used to investigate what information the model utilises. Model 4.1.1 is an example of a RIPPER model. Each line is one rule and each rule consists of 0 or more antecedents. (ALAT = f) is the first antecedent in rule 1. It requires that the ALAT attribute must be false in order to be satisfied, while (NONE = t) requires that the NONE-attribute must be true. All antecedents in a rule must be satisfied, in order to trigger the conclusion. If all antecedents are satisfied in the first rule, the rule concludes that the patient has diabetes. The quality of each rule is presented at the end of each line with two numbers inside parantheses. The first number is how frequently the rule concludes if it is applied to the examples used to build the model, while the second number is how many times the conclusion is wrong. Rule 1 concludes 402 times and it is wrong in 105 cases, thus it is correct in 297 cases.

Model 4.1.1 *Example:*
(ALAT = f) and (NONE = t) => Diabetes=t (402.0/105.0)
(ALAT = f) and (FERRITIN = f) => Diabetes=t (646.0/294.0)
(HEMOGLOBIN = f) and (URAT = f) and (HB = f) => Diabetes=t (21.0/5.0)
(FOSFOLIPIDER = t) => Diabetes=t (8.0/3.0)
=> Diabetes=f (293.0/14.0)

When the model is used to classify a new example, it tests each rule in top-down sequence. Rule 3 is tested after rule 1 and 2. All rules but the last one conlude that the example has diabetes, while the last rule states that if none of the rules above matched, the example is labeled with the default class. The default class is that the example does not have diabetes.

We introduce some terminology for speaking of RIPPER models. A condition which require that an attribute should be true, is called a *positive indicator*, while the opposite is a *negative indicator*. If a model only has negative indicators, we call it a *negation*-model. That is, it has not any conditions of the form (Attribute = t). Model 4.1.2 is an example of a negation-model.

Model 4.1.2 *Example:*
(ALAT = f) and (FERRITIN = f) => Diabetes=t (646.0/294.0)
(HEMOGLOBIN = f) and (URAT = f) and (HB = f) => Diabetes=t (21.0/5.0)
=> Diabetes=f (293.0/14.0)

4.2 E1: Determine time of suspicion

This section focuses on the events which were identified as the strongest suspicion-indicators, since they indirectly define t_{sus}. Tables 4.1, 4.2 and 4.3 presents the candidates which were identified as the strongest suspicion-indicators by the group of field experts (GoFE). The tables are sorted by iteration, then data set. Recall from Section 2.4.1 that there was an issue with inconsistent naming of events which affected iteration 1 of E1. As a consequence, the suspicion-indicators found in iteration 1 had to be converted to normalised versions. For the remaining iterations, the data set had been modified, thus the suspicion-indicators were already normalised.

4.2.1 Examining the suspicion-indicators

Even though the names of the suspicion-indicators from the two data sets PD-B and PM-B appear different in Table 4.1, one should note that the prescriptions from PD-B are named using *the name of the drug* in the medication, while PM-B uses *the name of the medication*. By looking up[1] the drugs used in the medications for asthma, we found that Bricanyl, Ventoline and Efedrin all have drugs from the R03-category in the ATC-register; *drugs for obstructive airway diseases*. More exactly, Bricanyl contains R03AC03, Ventoline contains R03AC02 and R03CC02 while Efedring contains R03CA02. 5 of the 6 suspicion-indicators from the R03-category are adrenergics, more specifically they are beta-2- or beta-adrenoreceptor agonists; a class of drugs which is used to treat asthma [5]. Tuxi is a medicine used to treat cough and since asthma irritates the airways, there is a possible connection.

The suspicion-indicators found for diabetes are mostly tests, especially PM-B which has only tests as its suspicion-indicators. Four drugs from the A10-category are found in the PD-B data set, the name of this category is simply *drugs used in diabetes*. The connection to the disease is therefore obvious. The test HBA1C measures the amount of glucose bound to hemoglobin proteins. Increased amounts of bound glucose indicates a longer period of elevated blood sugar levels, which could be related to diabetes [5]. Measurements of glucose in the blood is another test which is typical in relation with diabetes [5], and Table 4.2 shows that glucose is a suspicion-indicator. Creatinine and diabetes strips do also occurr in the table and they are also related to diabetes.

[1] From information found on Felleskatalogen; www.felleskatalogen.no

Set	It.	Name	Normalised	Type
PD-B	1	R03AC02	r03ac02	pres.
PD-B	1	R03AC03	r03ac03	pres.
PD-B	1	R03BA02	r03ba02	pres.
PM-B	1	BRICANYL-TURBOHALER	bricanyl	pres.
PM-B	1	VENTOLINE	ventoline	pres.
PM-B	2	-	efedrin	pres.
PM-B	2	-	tuxi	pres.

Table 4.1: The candidates which were marked as strongest suspicion-indicators for asthma.

Table 4.3 shows that it is mostly tests for different variants of thyroid hormones which are suspicion-indicators for hypothyroidism. T3 and T4 are both thyroid hormones as described in Section 2.3.3, while TSH is the hormone which stimulates production of T3 and T4. These hormones are bound or free in the body, FT4 thus means free T4. H03AA01 is a kind of thyroid hormone, and since it is a prescription it is likely that it is used in the treatment of the disease. The C-reactive protein (CRP) test is taken when bacterial infections are suspected.

4.2.2 Time to suspicion and diagnose

As part of E1 it was noted when t_{sus} and t_{TD} occurred for each patient who had a TD. We counted the number of (NOF) days from the start of PH to t_{sus}, and from t_{sus} to t_{TD}. In addition we also counted the NOF visits for the same periods. For each patient in each period, we have a pair (NOF days, NOF visits) which contains the information that tells us the lengths of these periods using two perspectives on time. All pairs for the same TD, iteration and data set were plotted together, giving an overview of the lengths of these periods.

Figure 4.1 shows the plots for iteration 1, using data from PD-B and PM-B. They only have points in the period *start* - t_{sus}, the plots for t_{sus} - t_{TD} are not shown since they are empty. This is understandable, since $t_{sus} - t_{TD} = 0$ because t_{sus} was defined to equal t_{TD} at this iteration. One can clearly see the difference between the two data sets with respect to the number of positives, PM-B contains far more data than PD-B. The subfigures provide useful information about when t_{TD} is, relative to the first time each patient visited the GP. Very few positives in PD-B are diagnosed more than 5000 days after their first visit to the GP, while this number is considerably larger for PM-B.

It is more interesting to see the situation after t_{sus} has been adjusted in iteration 2 and 3; the figures for iteration 3 are 4.2 and 4.3 (figures B.13 and B.14 for iteration 2 are in Appendix). We have included the period t_{sus} - t_{TD} which shows how long time the GP uses from suspicion till the diagnosis is set. There are some differences between asthma on one side and diabetes and hypothyroidism on the other; the GP suspects the presence of diabetes and hypothyroidism earlier than asthma, but the time from suspicion to diagnosis is shorter for asthma patients compared to the other two diseases. This is most easily seen in Figure 4.2 which shows data from PD-B.

Table 4.4 strengthens this observation, as it presents the number of patients who were suspected of having a TD on their first visit to the GP. For PM-B, almost half of the patients with diabetes or hypothyroidism were suspected of having the TD on their first visit, while the number is

Set	It.	Name	Normalised	Type
PD-B	1	A10AB01	a10ab01	pres.
PD-B	1	A10AC01	a10ac01	pres.
PD-B	1	A10BB07	a10bb07	pres.
PD-B	1	A10BB12	a10bb12	pres.
PD-B	1	B-GLYKO	b-glyko	test
PD-B	1	GLUCOSE	glucose	test
PD-B	1	HBA1C	hba1c	test
PM-B	1	eb HbA1c	hba1c	test
PM-B	1	s Glukose 1	glucose	test
PM-B	1	s Glukose 2	glucose	test
PM-B	1	s Glukose FAST	glucose	test
PM-B	1	S-GLUKOSE FAST	glucose	test
PM-B	1	s Kreatinin	kreatinin	test
PM-B	1	s Natrium	natrium	test
PM-B	1	Testtape	testtape	test
PM-B	1	u Strimmel Glukose	strimlerGlukose	test
PD-B	2	-	Ustix	test
PM-B	2	-	Diabetes-str.	test
PM-B	2	-	Strimler	test
PM-B	2	-	Ketoner	test

Table 4.2: The candidates which were marked as strongest suspicion-indicators for diabetes.

Set	It.	Name	Normalised	Type
PD-B	1	H03AA01	h03aa01	pres.
PD-B	1	R05CB01	r05cb01	pres.
PD-B	1	HB	hb	test
PD-B	1	CRP	crp	test
PD-B	1	S-CRP	crp	test
PD-B	1	S-FT4	ft4	test
PD-B	1	S-T3	t3	test
PD-B	1	S-TSH	tsh	test
PD-B	1	T4	t4	test
PM-B	1	THYROXIN-NATRIUM	thyroxin-natrium	test
PM-B	1	s Thyroxin, fritt T4	ft4	test
PM-B	1	S-FRITT T4	ft4	test
PM-B	1	s Thyroxin, total T4	t4	test
PM-B	1	S-T4	t4	test
PM-B	1	s TSH	tsh	test
PM-B	1	S-TSH	tsh	test
PM-B	1	s Trijodothyronin T3	t3	test
PM-B	2	-	Thyroxin	test

Table 4.3: The candidates which were marked as strongest suspicion-indicators for hypothyroidism.

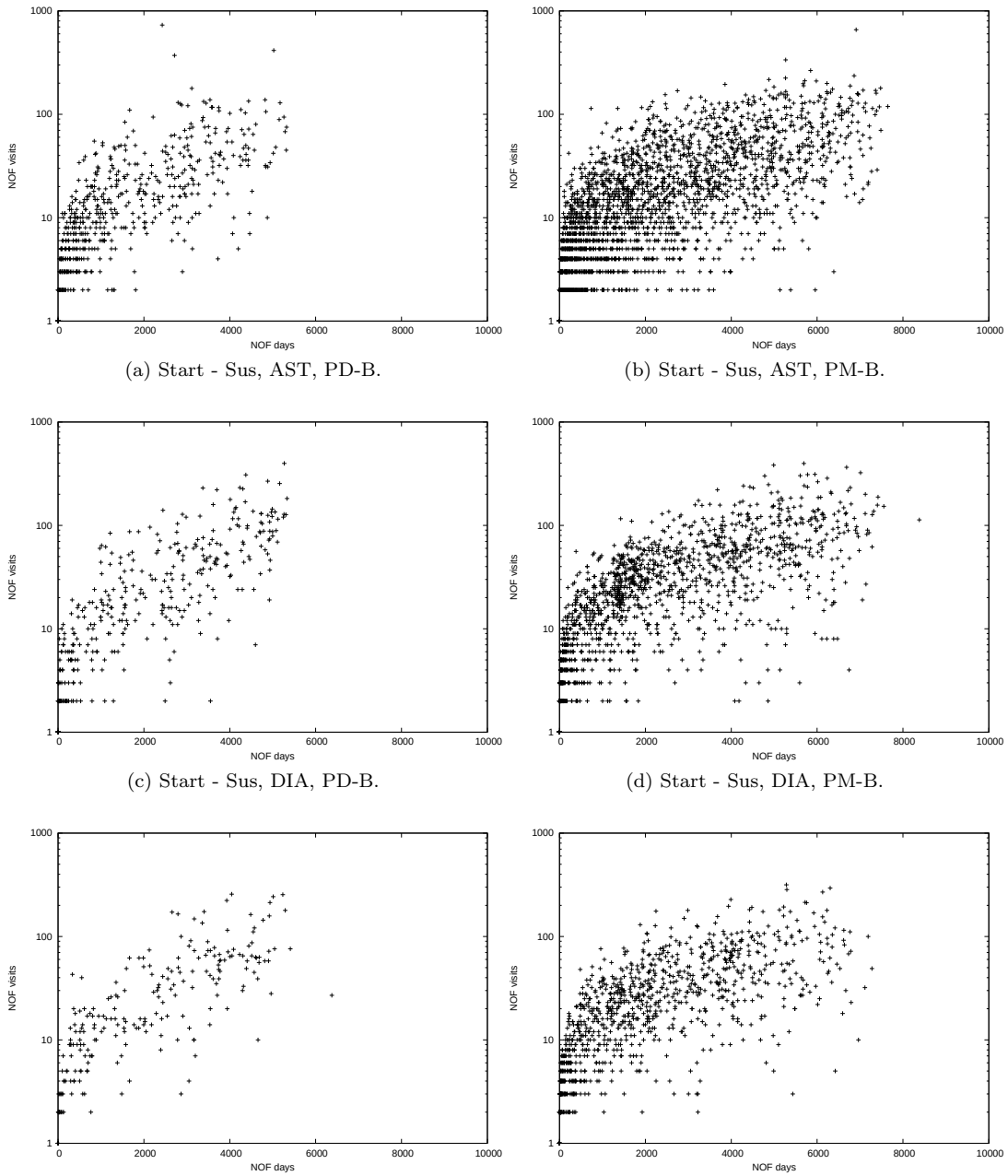
	PD-B		PM-B	
TD	Freq.	%	Freq.	%
AST	98	15	797	26
DIA	173	41	810	46
HYP	97	38	567	47

Table 4.4: The number of patients who were suspected of TD on their first visit to the GP.

only 26% for asthma.

Since the overall task is to give warning prior to the suspicion, we constrain ourselves to only use data from the start of PH until t_{sus} (we use the best approximation of t_{sus} which was found in iteration 3 of E1). Thus we depend on the events which occurred in this period to make a prediction. If there are no events or the events which exist are irrelevant, the task of making predictions based on this data will be impossible. In order to get a better picture of how much information is available in this period, we have documented how many positives would remain if we required that the period *start* - t_{sus} was at least X days. This is shown in Figure 4.4 for the two data sets PD-B and PM-B. Figure 4.5 is similar, but its x-axis is the number of visits instead of days.

The curves for diabetes and hypothyroidism are very close in these figures, always lying below the curve for asthma. From Figure 4.4a we can see that if we required that start - t_{sus} was at least 500 days, approximately 60% of the patients with asthma would fulfill this requirement, while the corresponding number for the patients with diabetes and hypothyroidism would be only 30%.



(a) Start - Sus, AST, PD-B.

(b) Start - Sus, AST, PM-B.

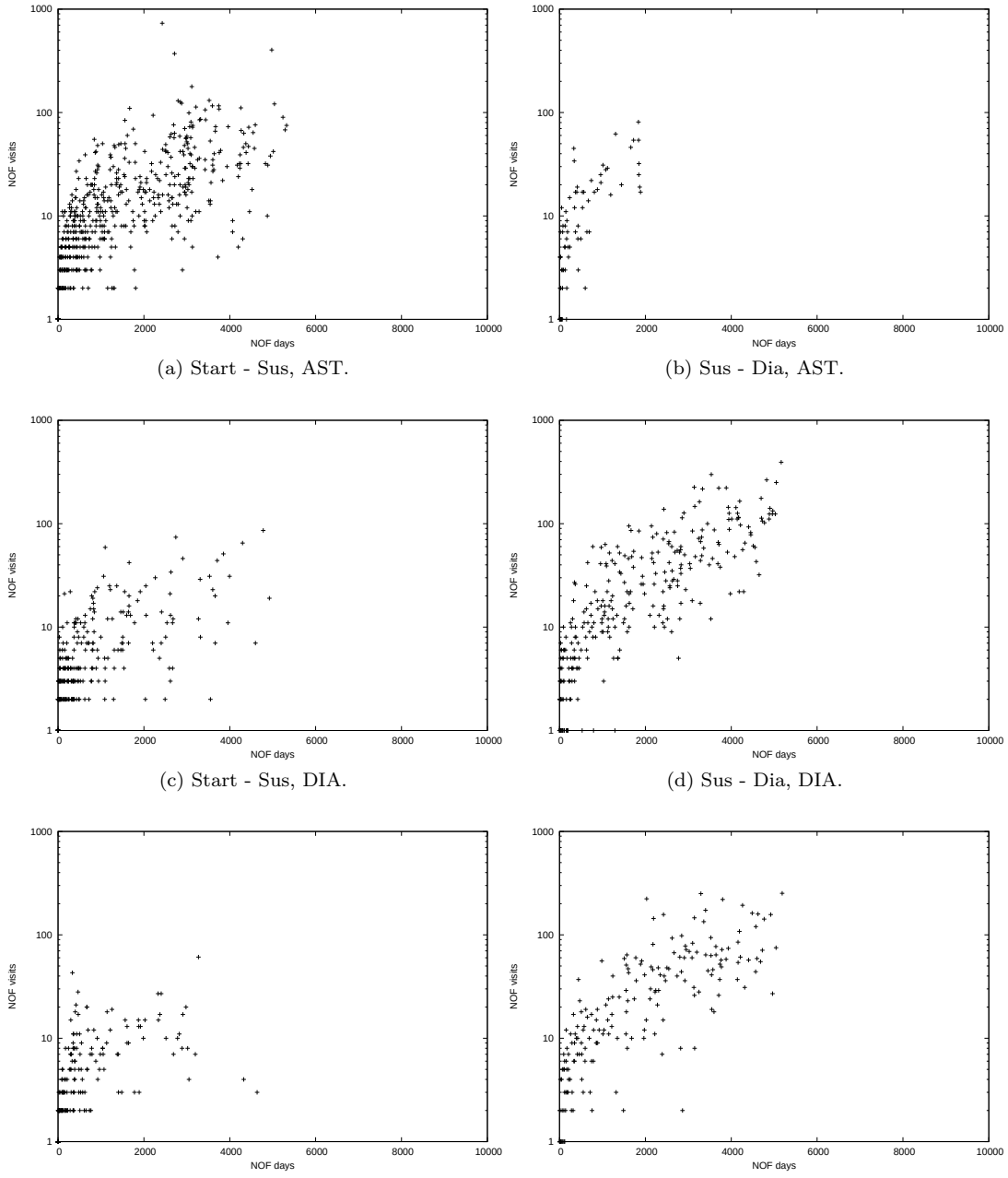
(c) Start - Sus, DIA, PD-B.

(d) Start - Sus, DIA, PM-B.

(e) Start - Sus, HYP, PD-B.

(f) Start - Sus, HYP, PM-B.

Figure 4.1: Scatter plots of how quickly GP suspects the TD, iteration 1.



(a) Start - Sus, AST.

(b) Sus - Dia, AST.

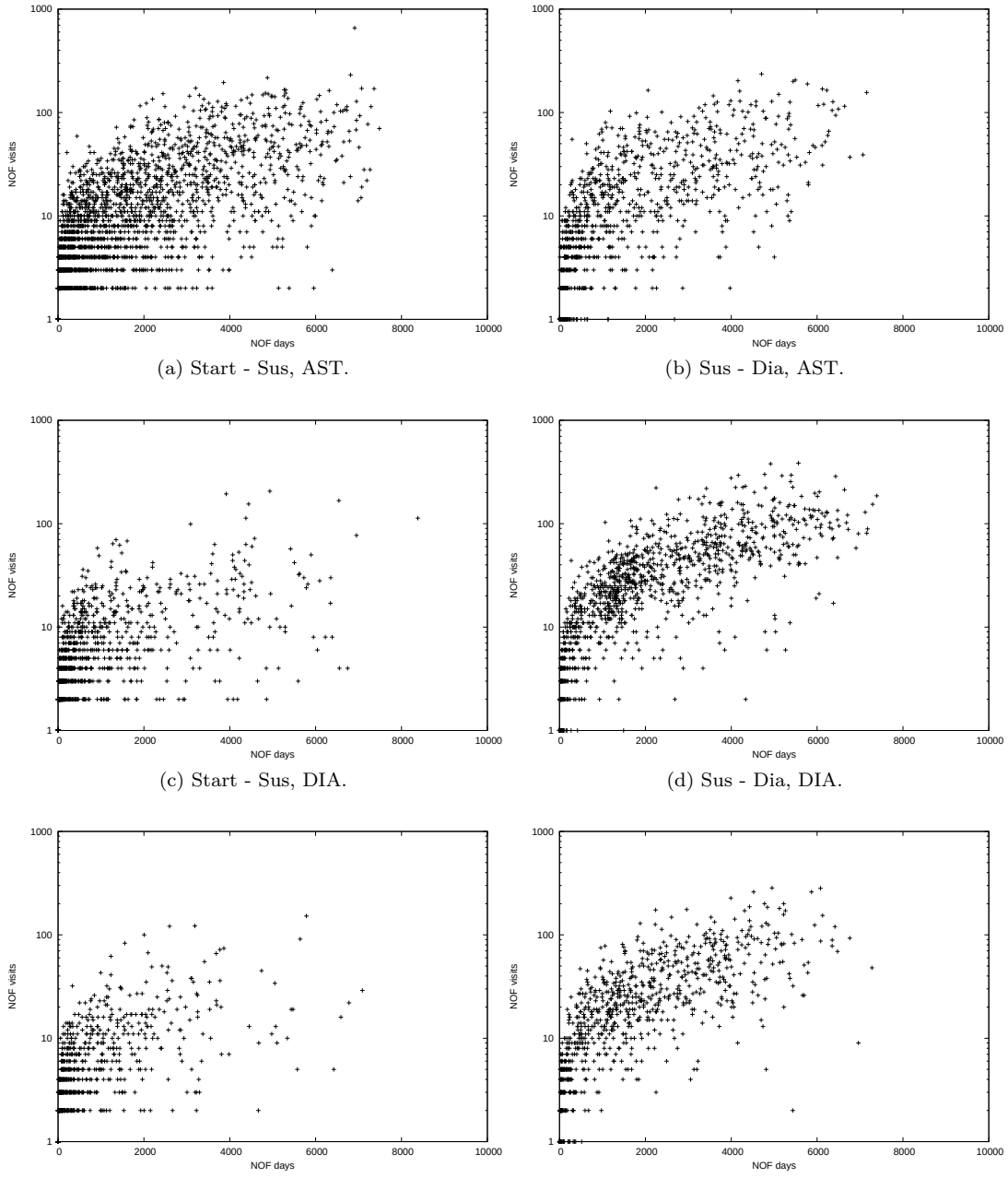
(c) Start - Sus, DIA.

(d) Sus - Dia, DIA.

(e) Start - Sus, HYP.

(f) Sus - Dia, HYP.

Figure 4.2: Scatter plots of how quickly GP suspects and diagnoses patients, using data from PD-B, iteration 3.



(a) Start - Sus, AST.

(b) Sus - Dia, AST.

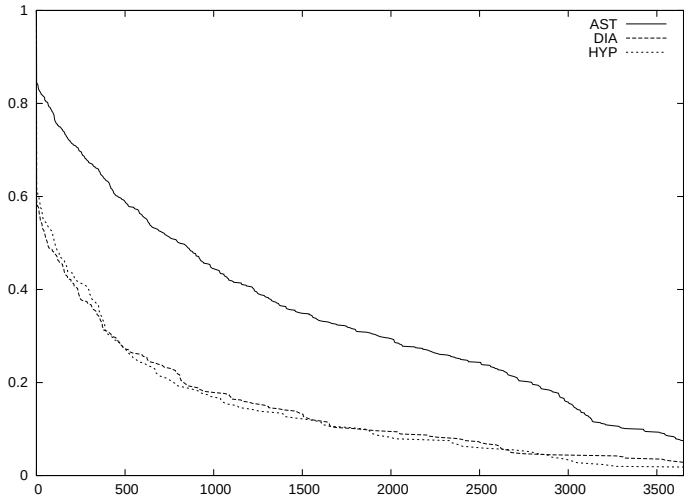
(c) Start - Sus, DIA.

(d) Sus - Dia, DIA.

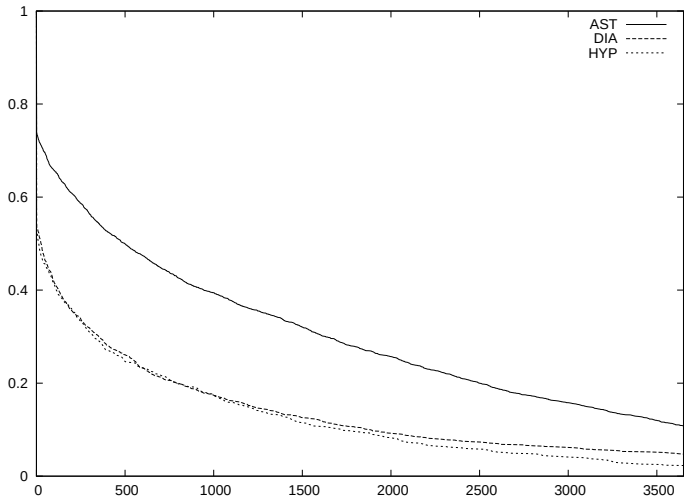
(e) Start - Sus, HYP.

(f) Sus - Dia, HYP.

Figure 4.3: Scatter plots of how quickly GP suspects and diagnoses patients, using data from PM-B, iteration 3.

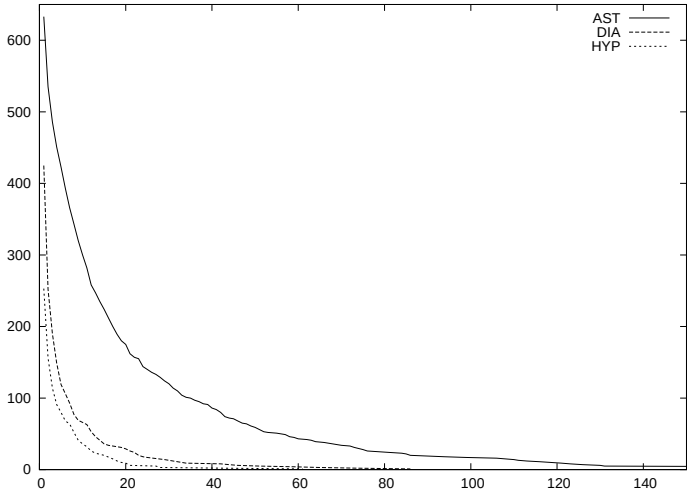


(a) PD-B, iteration 3.

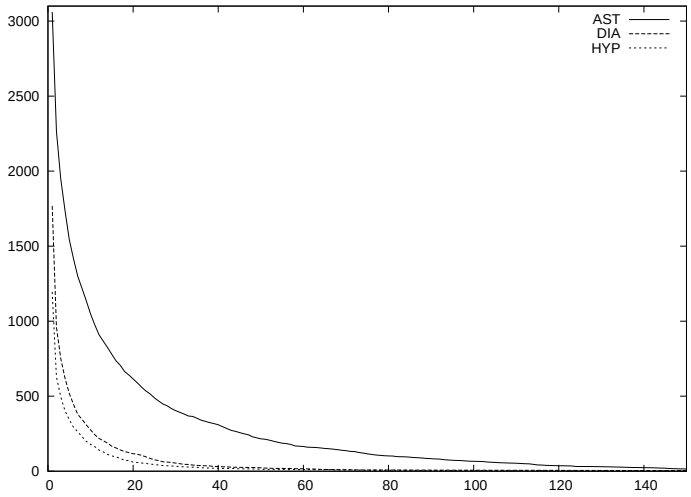


(b) PM-B, iteration 3.

Figure 4.4: The ratio of positives remaining as we require the period start - t_{sus} to be greater than a given number of days. X-axis: number of days, Y-axis: ratio of positives remaining.



(a) PD-B, iteration 3.



(b) PM-B, iteration 3.

Figure 4.5: The number of positives remaining as we require the period start - t_{sus} to be greater than a given number of visits. X-axis: number of visits, Y-axis: number of positives remaining.

It. 1		It. 2		It. 3	
1	R81	1	R78	1	R83
1	R78	1	R05	1	R78
1	R74	1	R02	1	R74
1	R02	1	H71	1	R02
1	H71			1	H71

Table 4.5: Positive indicators and their frequency from RIPPER models for all iterations, with respect to asthma and PD-B.

4.2.3 Model analysis

The previous section gave an overview of the results in E1 from a performance-oriented perspective. This section investigates how the RIPPER models are altered for each iteration. We choose to focus our attention on the *positive indicators* which occur in the models for different iterations. If an antecedent in a rule requires an attribute to be true, the attribute is a positive indicator (described in section 4.1). Tables 4.5 to 4.10 present the positive indicators for each disease-dataset combination. The tables present the positive indicators and their frequency for each iteration.

One common trend for all these tables is that the number of positive indicators diminish for each iteration. This is not unexpected, since information is retained further away from the diagnosis is set for each iteration.

Positive indicators for asthma

The positive indicators for asthma are shown in table 4.5 and 4.6. For the PD-B data set, it is mainly diagnosis codes from the R-group which are used. Note that none of the suspicion-indicators are positive indicators. H71, R02 and R78 are used in all iterations. The corresponding diagnoses titles for these codes are *acute otitis media/myringitis*, *shortness of breath/dyspnoea* and *acute bronchitis/bronchiolitis*. These are all diseases which are related to the airways, thus it seems likely that they could have a connection to asthma.

There are no positive indicators remaining in iteration 3 when using the PM-B data set, which indicates that the model did not find anything particularly useful.

Positive indicators for diabetes

The models which are built to predict diabetes, use the positive indicators presented in tables 4.7 and 4.8. In the first table there are three positive indicators which remain in iteration 3; HSTAT, GLYKO and EVF. These are medical tests, the GLYKO test is probably some kind of glycogen test.

In the PM-B data set, the positive indicators in iteration 3 (see table 4.8) include tests for hemoglobin, FT4 and kalium. The diagnosis K86, *hypertension uncomplicated* is a condition of chronically elevated blood pressure which is especially dangerous for diabetes patients [10]. We ignore the positive indicator NONE, as we believe it is irrelevant based on its name.

It. 1		It. 2		It. 3
2	*VENTOLINE	2	CRP	
2	*BRICANYL	1	NONE	
1	T4	1	KATT	
1	R81	1	IGE	
1	R74			
1	PULMICORT			
1	NONE			
1	IGE			
1	G6			
1	D2			
1	CRP			

Table 4.6: Positive indicators and their frequency from RIPPER models, with respect to asthma and PM-B. Attributes marked with * are suspicion indicators.

It. 1		It. 2		It. 3	
3	GLUCOSE*	1	K86	1	HSTAT
1	R78	1	HB	1	GLYKO
1	MCV	1	GLYKO	1	EVF
1	L13	1	FRUKT		
1	K86	1	ERYTROCYTTER		
1	K07	1	ALAT		
1	HBA1C*				
1	GLYKO				
1	FRUKT				
1	CL				

Table 4.7: Positive indicators and their frequency from RIPPER models, with respect to diabetes and PD-B. Attributes marked with * are suspicion indicators.

It. 1		It. 2		It. 3	
10	*GLUCOSE	4	NONE	1	NONE
4	NONE	2	HEMOGLOBIN	1	KALIUM
4	*HBA1C	1	SYSTOLISK	1	K86
4	ALBUMIN	1	KALIUM	1	HEMOGLOBIN
2	TRIJODOTHYRONIN	1	K86	1	HB
2	*TESTTAPE	1	HB	1	FT4
2	FERRITIN	1	FT4	1	ALAT
2	ASAT	1	FOSFOLIPIDER		
1	URINSYRE	1	DIASTOLISK		
1	TROMBOCYTTER	1	BLODTRYKK		
1	PH	1	ALP		
1	KOLESTEROL	1	ALAT		
1	K86				
1	HB				
1	GLYKOS.HEMOGLO.				
1	GLUKOSEBELASTNING				
1	GLUCOPHAGE				
1	GAMMA				
1	CRP				
1	CK				
1	AMYLASE				

Table 4.8: Positive indicators and their frequency from RIPPER models, with respect to diabetes and PM-B. Attributes marked with * are suspicion indicators.

It. 1		It. 2		It. 3	
1	*FT4	1	ERYTROCYTTER	1	EVF
		1	ALAT	1	ALAT

Table 4.9: Positive indicators and their frequency from RIPPER models, with respect to hypothyroidism and PD-B. Attributes marked with * are suspicion indicators.

It. 1		It. 2		It. 3	
7	*FT4	2	NONE	1	NONE
2	*THYROXIN	1	*THYROXIN	1	GLUCOSE
1	TRIJODOTHYRONIN	1	TESTTAPE	1	ALP
1	THYROGLOBULIN	1	HEMOGLOBIN	1	ALAT
1	SR	1	GLUCOSE		
1	NONE	1	FERRITIN		
1	*HB	1	DIASTOLISK		
		1	ALP		
		1	ALAT		

Table 4.10: Positive indicators and their frequency from RIPPER models, with respect to hypothyroidism and PM-B. Attributes marked with * are suspicion indicators.

Positive indicators for hypothyroidism

The average of positive indicators for the PD-B dataset is low compared to the other diseases. Table 4.9 show that EVF and ALAT (alanine aminotransferase) remain in iteration 3. ALAT can also be found in iteration 3 for the PM-B data set as well, as can be seen in table 4.10. In addition we find the attributes NONE, GLUCOSE and ALP. NONE could indicate a missing value in the journal system, ALP (alkaline phosphatase) is a medical test which measures the level of a protein in the blood. It is done to diagnose liver or bone disease, or to see if treatments for those diseases are working.

4.3 E2: Varying settings when APP ≤ PP

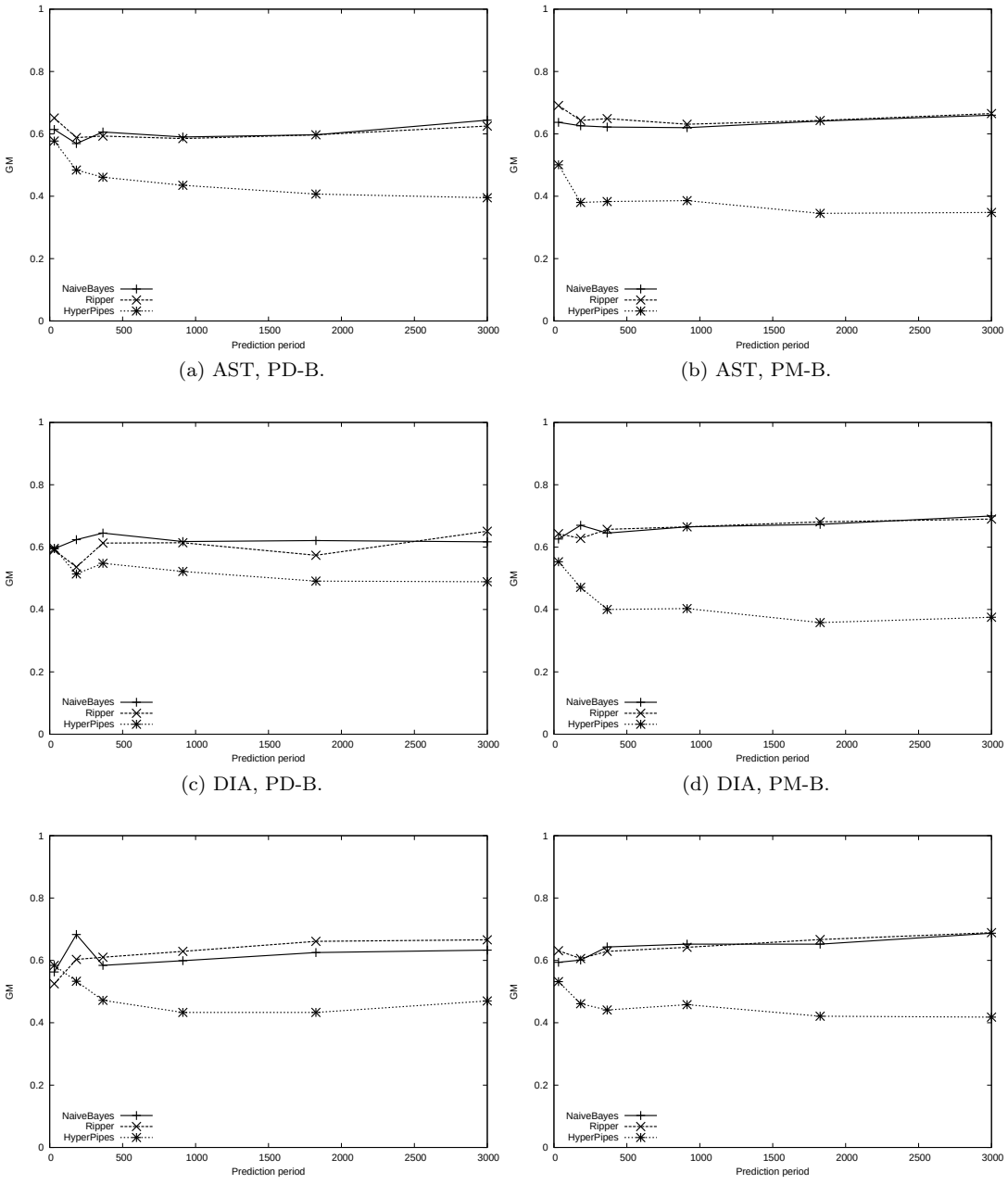
This section presents the results from experiment 2, which aims at documenting how the different parameters such as prediction period (PP), warning time (WT), number of negatives (NN) and number of attributes (NA) affect the classifiers which build the predictive models. We present the variation of each setting in the context of the the three different TDs, the two data sets and the three iterations in order to capture any patterns if they exist. We have chosen to use geometric mean as the evaluation measure, the rationale behind this choice is described in Section 5.7. In short, it gives a reasonably good picture of the classifiers, even in situations with class imbalance.

We denote maximum PP (PP = All) with the value 3000 in the figures. In fact, the value of PP will vary depending on each patient, but since the models are built using many patients, we must assign a static value to PP in the figures.

4.3.1 Effect of varying PP

Figure 4.6 and 4.7 show the quality of the models with respect to accuracy, when we vary the size of the PP. As the PP grows in size, events are collected from a longer period, increasing the chance that more events will be included in the FVs. Constant values for this experiment were; WT = 100, NN = 1, NA = All.

The sub figures from iteration 1 shows that Ripper and Naive Bayes has a gentle improvement from PP = 100 to PP = All. This improvement is absent in iteration 3. By comparing the classifiers, Naive Bayes and Ripper outperforms HyperPipes in practically all figures for iteration 1, while the situation is not so clear-cut in iteration 3. Hyperpipes is still worse for asthma, but for the other two diseases it is just as good or better. For PP = [31, 365], the figures do not show much similarity. Since it is a quite short period to collect data from, the classifiers are more susceptible to noise, which could explain these variations.



(a) AST, PD-B.

(b) AST, PM-B.

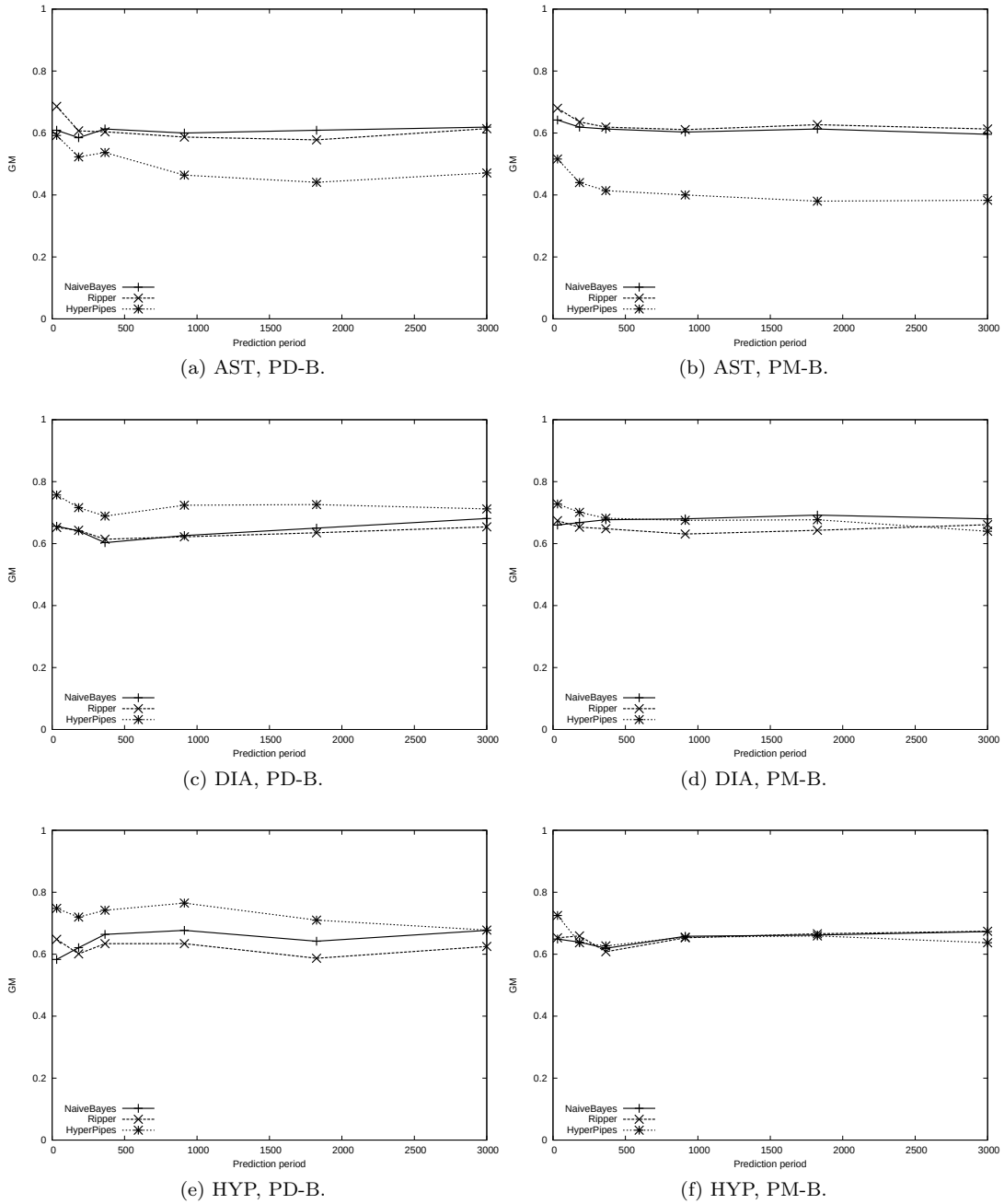
(c) DIA, PD-B.

(d) DIA, PM-B.

(e) HYP, PD-B.

(f) HYP, PM-B.

Figure 4.6: Performance of classifiers for iteration 1. Classifier is evaluated using geometric mean, and it is the PP which varies in each plot.



(a) AST, PD-B.

(b) AST, PM-B.

(c) DIA, PD-B.

(d) DIA, PM-B.

(e) HYP, PD-B.

(f) HYP, PM-B.

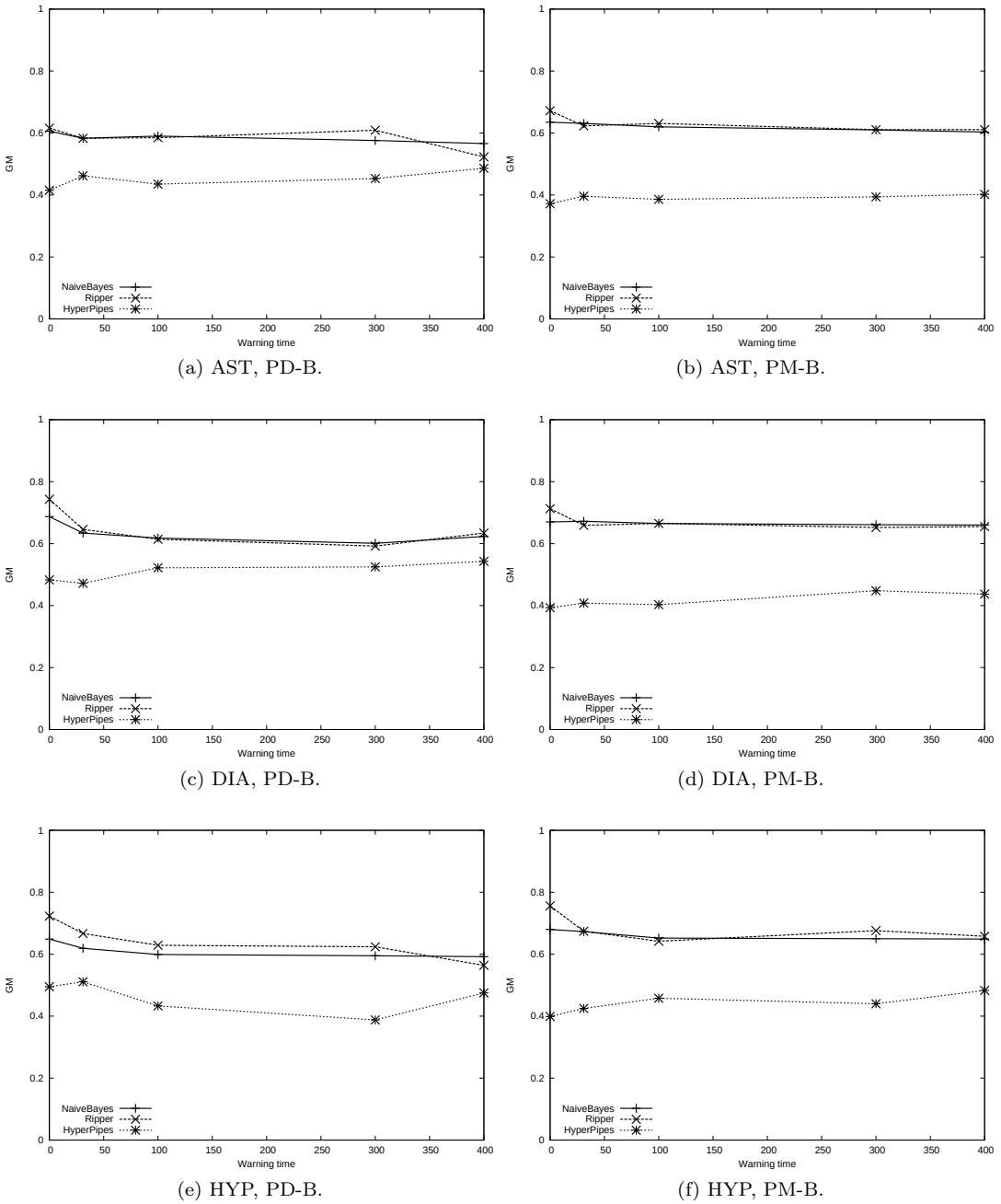
Figure 4.7: Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the PP which varies in each plot.

4.3.2 Effect of varying WT

Figure 4.8 and 4.9 show the quality of the models with respect to geometric mean, when we vary the size of the WT. As the size of the WT increases, it makes the learning problem harder, since we cannot collect the events which are immideately prior to t_{sus}. Constant values for this experiment were; PP = 913, NN = 1, NA = All.

The Ripper classifier performs very well in iteration 1 when WT = 0, compared to the cases where WT > 0. This could mean that it was able to use information which occurred immideately prior to t_{sus}, which is actually t_{TD} for iteration 1. The other two classifiers do not seem to capture this information to the same extent. Whether WT = 31 or greater does not have much effect on the classifiers in iteration 1. One possible explanation would be that the information which is particularly useful in classification is less than 31 days prior to t_{TD}.

One would expect that the results of iteration 3 could not match those of iteration 1, since the t_{sus} used in iteration 3 is much earlier than t_{sus} in iteration 1. However, the Ripper algorithm gives a score which is comparable to the best from iteration 1 for hypothyroidism (PD-B) and diabetes (PM-B).



(a) AST, PD-B.

(b) AST, PM-B.

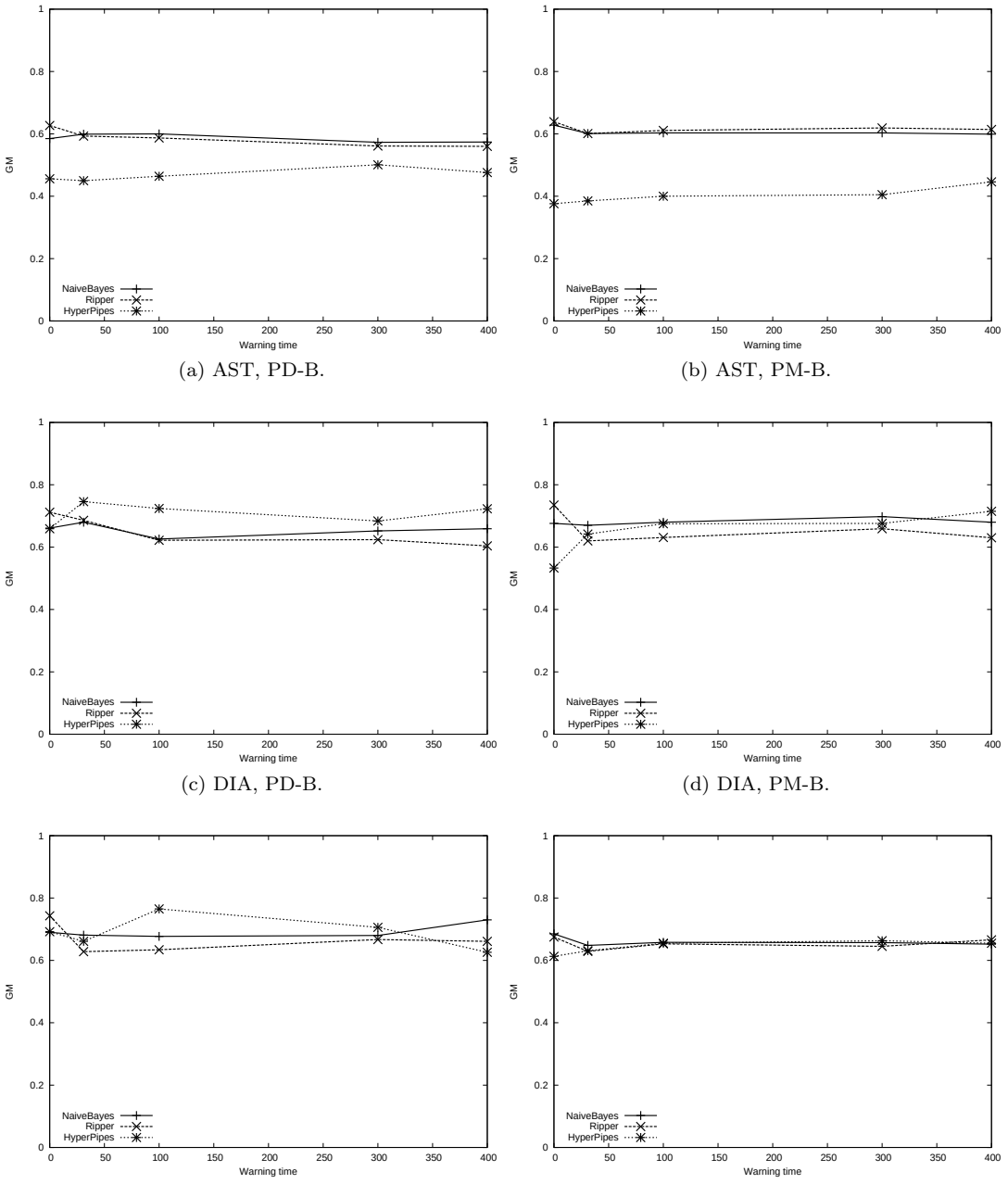
(c) DIA, PD-B.

(d) DIA, PM-B.

(e) HYP, PD-B.

(f) HYP, PM-B.

Figure 4.8: Performance of classifiers for iteration 1. Classifier is evaluated using geometric mean, and it is the WT which varies in each plot.



(a) AST, PD-B.

(b) AST, PM-B.

(c) DIA, PD-B.

(d) DIA, PM-B.

(e) HYP, PD-B.

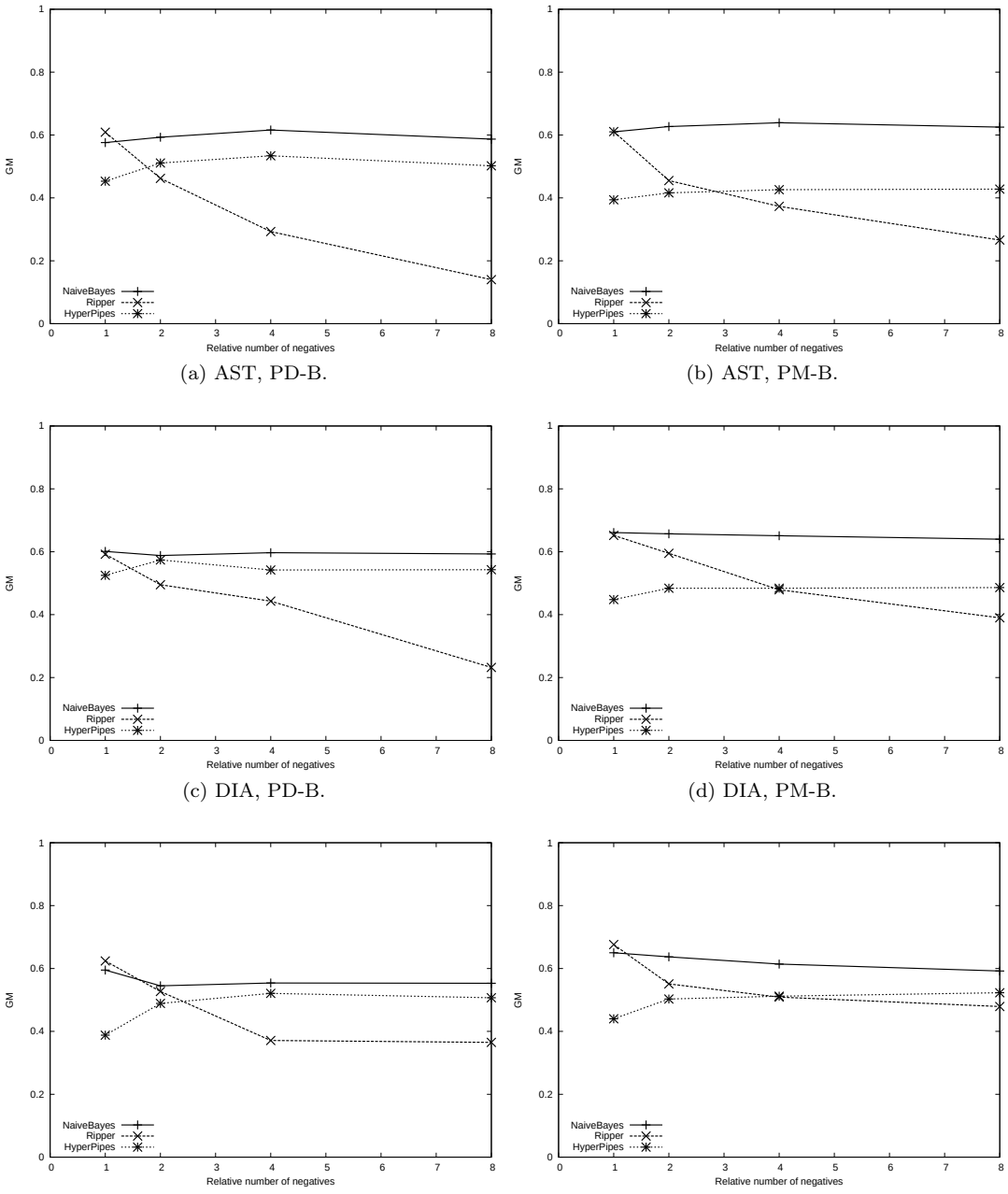
(f) HYP, PM-B.

Figure 4.9: Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the WT which varies in each plot.

4.3.3 Effect of varying NN

In this experiment, the class distribution is altered from having practically equal numbers of positives and negatives at NN = 1, to the situation where there is approximately 8 times as many negatives as positives at NN = 8. Thus the distribution becomes more and more imbalanced with respect to the two classes. Figures 4.10 and 4.11 show the quality of the models with respect to GM, when we vary NN. NN decides how many negative examples are added to the training set for each positive. Increasing NN will affect the distribution of positives and negatives in the training set, by increasing the number of negatives compared to the positives. This makes it harder to learn a good model for both classes, because the class imbalance induced by larger NN often have a negative effect on the ability of the classifier to make a model which generalises over both classes. Constant values for this experiment were; PP = 913, WT = 300, NA = All.

The Ripper classifier shows poor performance as NN increases, Naive Bayes and Hyperpipes are not affected particularly much. The last two classifiers show markedly improved performance in iteration 3 compared to iteration 1.



(a) AST, PD-B.

(b) AST, PM-B.

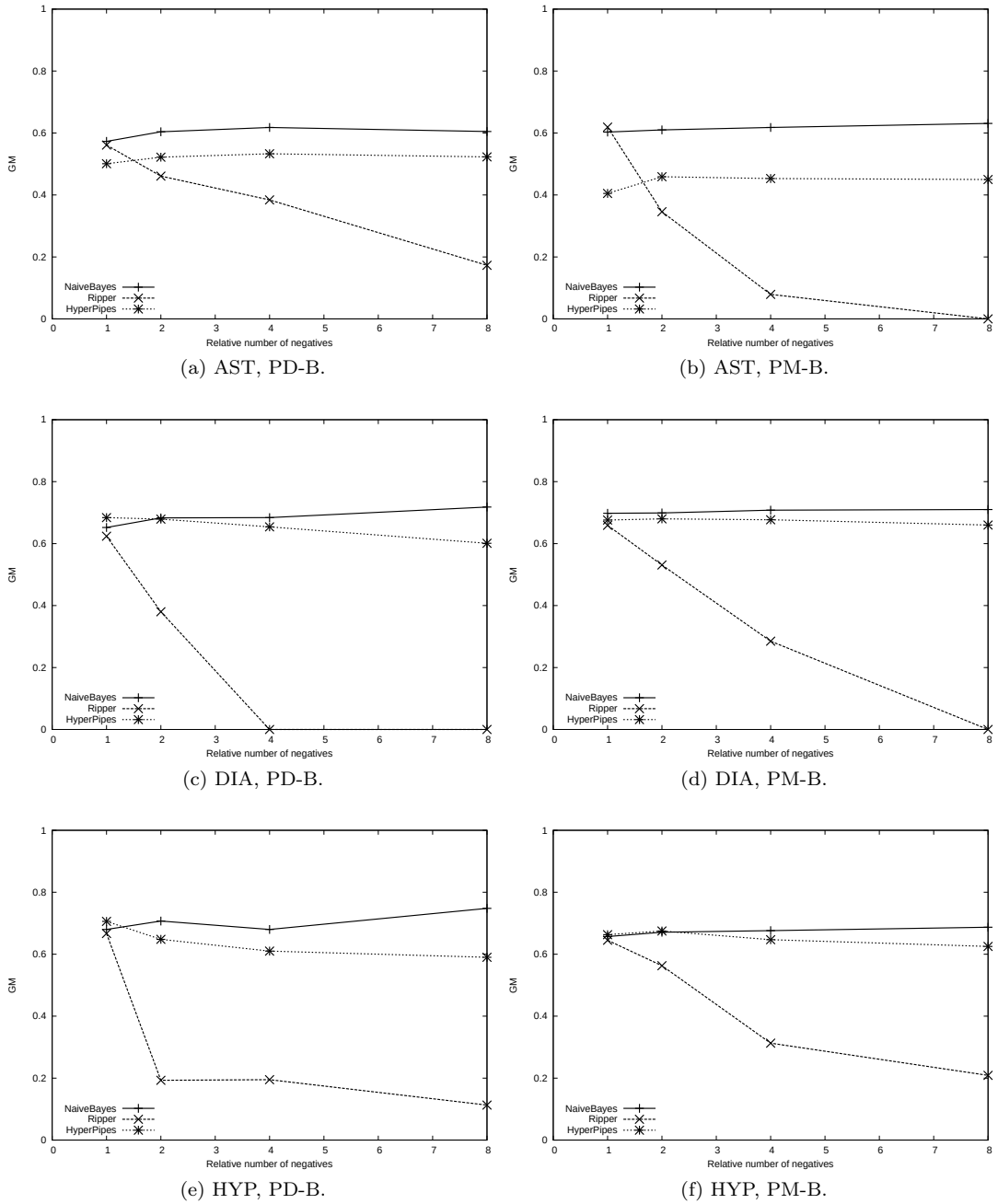
(c) DIA, PD-B.

(d) DIA, PM-B.

(e) HYP, PD-B.

(f) HYP, PM-B.

Figure 4.10: Performance of classifiers for iteration 1. Classifier is evaluated using geometric mean, and it is the NN which varies in each plot.



(a) AST, PD-B.

(b) AST, PM-B.

(c) DIA, PD-B.

(d) DIA, PM-B.

(e) HYP, PD-B.

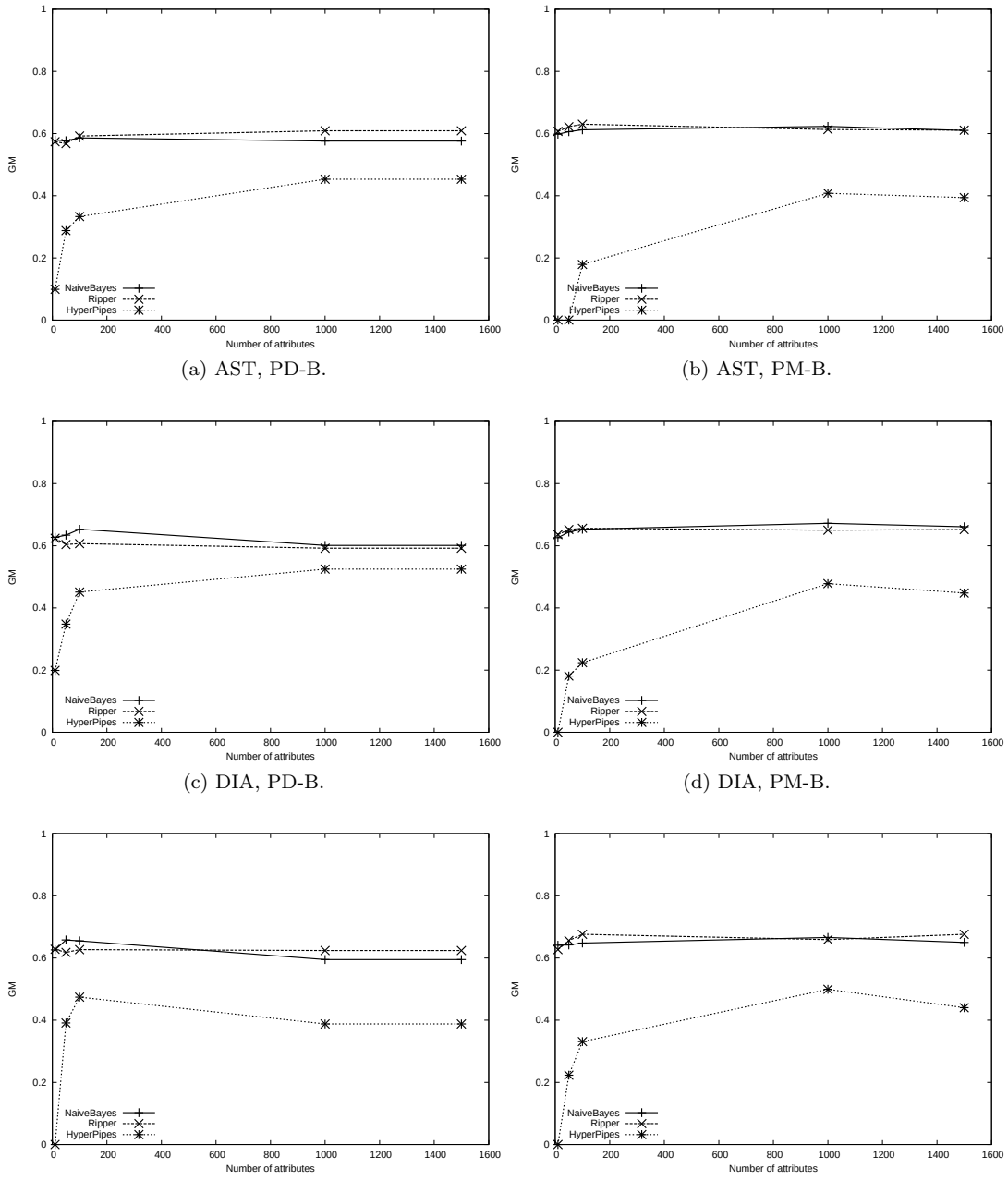
(f) HYP, PM-B.

Figure 4.11: Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NN which varies in each plot.

4.3.4 Effect of varying NA

Figures 4.12 and 4.13 show the quality of the models with respect to GM, when we vary NA. NA controls the number of attributes which are used in each FV. They contain information which can be used by the classifiers. The question is how much of this information is relevant with respect to the TDs. Information which is not relevant makes the task of predicting harder, becase of the *curse of dimensionality* which is part of the discussion in Section 5.6. Constant values for this experiment were; PP = 913, WT = 300, NN = 1.

The Hyperpipes classifier shows a sharp decline for the smallest values of NA in 10 of 12 sub figures. One possible explanation is that the remaining features does not discriminate between the two classes any longer. Naive Bayes and Ripper are basically unaffected by the variation in NA. One difference between the iterations 1 and 3 is that the Naive Bayes and Ripper have slightly better performance in iteration 3 than in 1.



(a) AST, PD-B.

(b) AST, PM-B.

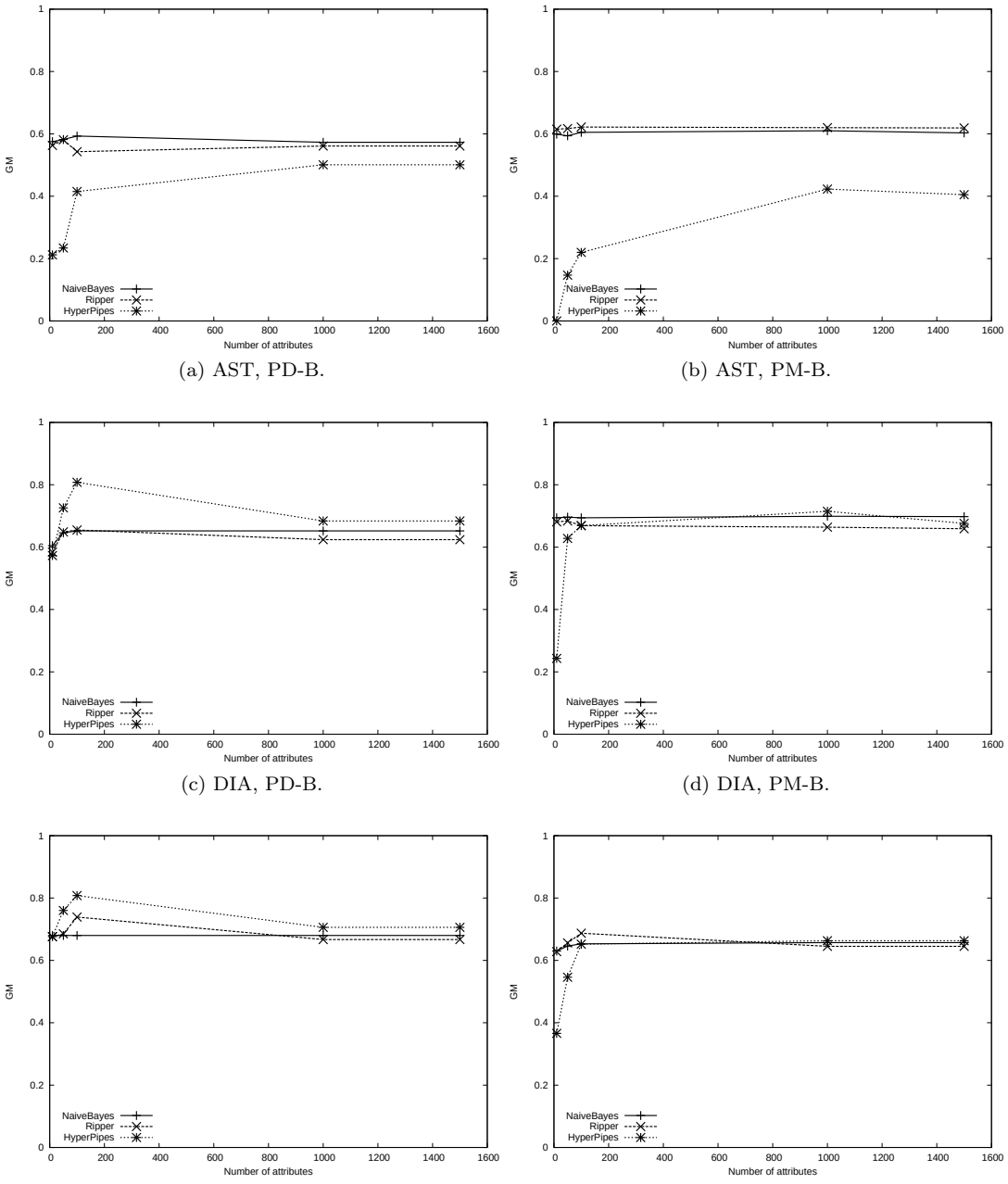
(c) DIA, PD-B.

(d) DIA, PM-B.

(e) HYP, PD-B.

(f) HYP, PM-B.

Figure 4.12: Performance of classifiers for iteration 1. Classifier is evaluated using geometric mean, and it is the NA which varies in each plot.



(a) AST, PD-B.

(b) AST, PM-B.

(c) DIA, PD-B.

(d) DIA, PM-B.

(e) HYP, PD-B.

(f) HYP, PM-B.

Figure 4.13: Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NA which varies in each plot.

	NN			
Model	1	2	4	8
AST, PD-B	4 (8)	8 (24)	12 (81)	13 (94)
AST, PM-B	2 (3)	2 (4)	8 (43)	49 (518)
DIA, PD-B	4 (6)	4 (10)	3 (10)	13 (146)
DIA, PM-B	3 (7)	7 (31)	8 (46)	4 (11)
HYP, PD-B	2 (3)	3 (31)	3 (34)	7 (93)
HYP, PM-B	3 (7)	5 (27)	8 (22)	13 (111)

Table 4.11: The number of rules, with antecedents in paranthesis, for the RIPPER classifier as the number of positives is increased.

4.4 E3: Increasing positives when APP ≤ PP

Figure 4.14 shows what happens when we add extra positives which are generated using SMOTE. The x-axis denotes NN and increasing NN will increase the number of negative examples per non-synthetic positive. SMOTE generates enough synthetic positives so that there are equal numbers of positives and negatives. Thus if NN = 4, there are four times as many negatives as non-synthetic positives, but SMOTE adds extra synthetic positives so the total number of positives is the same as negatives. 75% of the positives are synthetic in that case.

We do not have to create synthetical negatives, because there is an abundancy of negative patient histories available in both PD-B and PM-B.

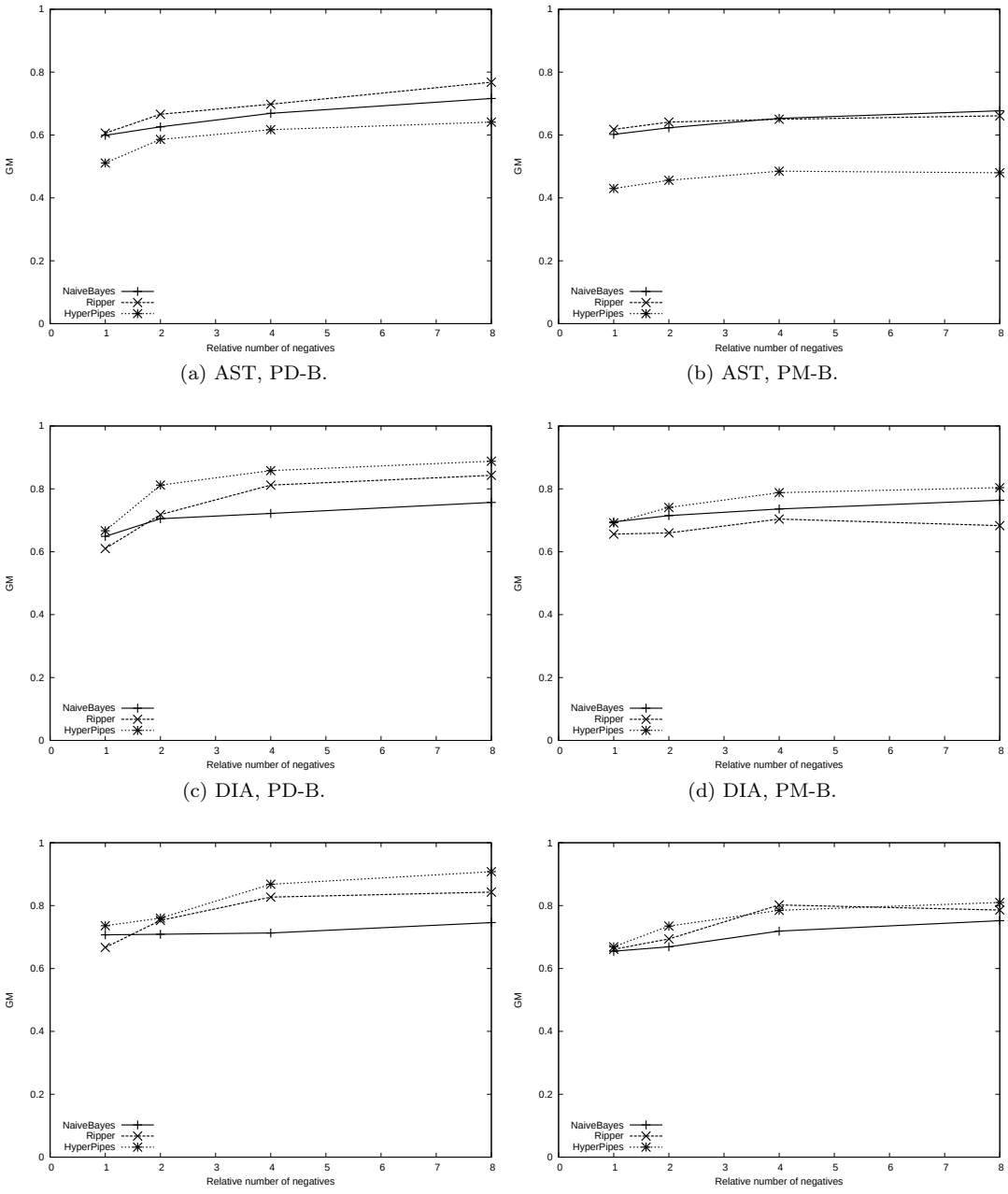
4.4.1 Increased complexity

In all the subfigures shown in figure 4.14, the performance of the classifiers increase as we add more positives. Table 4.11 shows the complexity of the models for the subfigures, using the models of RIPPER. The trend is that with increasing number of examples, the model has more rules and especially antecedents. Thus the models become more fine-grained and complex. The same trend was seen in the decision trees models presented by Taft et al. [39], as a consequence of increasing the number of positives with SMOTE.

4.4.2 Too few attributes

Figure 4.15 shows the effects of varying the number of attributes after we increase the number of negatives with NN = 8 and add the corresponding number of synthetical positives. The results show that there is not much difference between using 1000 instead of all attributes. The performance drops as NA diminishes, but there are great differences among the classifiers. 100 attributes is apparently not enough for Hyperpipes, as its performance drops drastically from 1000 to 100 attributes compared to the other two classifiers. As NA drops from 100 to 50 and 10 attributes, all classifiers decrease in performance.

From these results, it seems like the optimal number of attributes is between 100 and 1000 attributes.



(a) AST, PD-B.

(b) AST, PM-B.

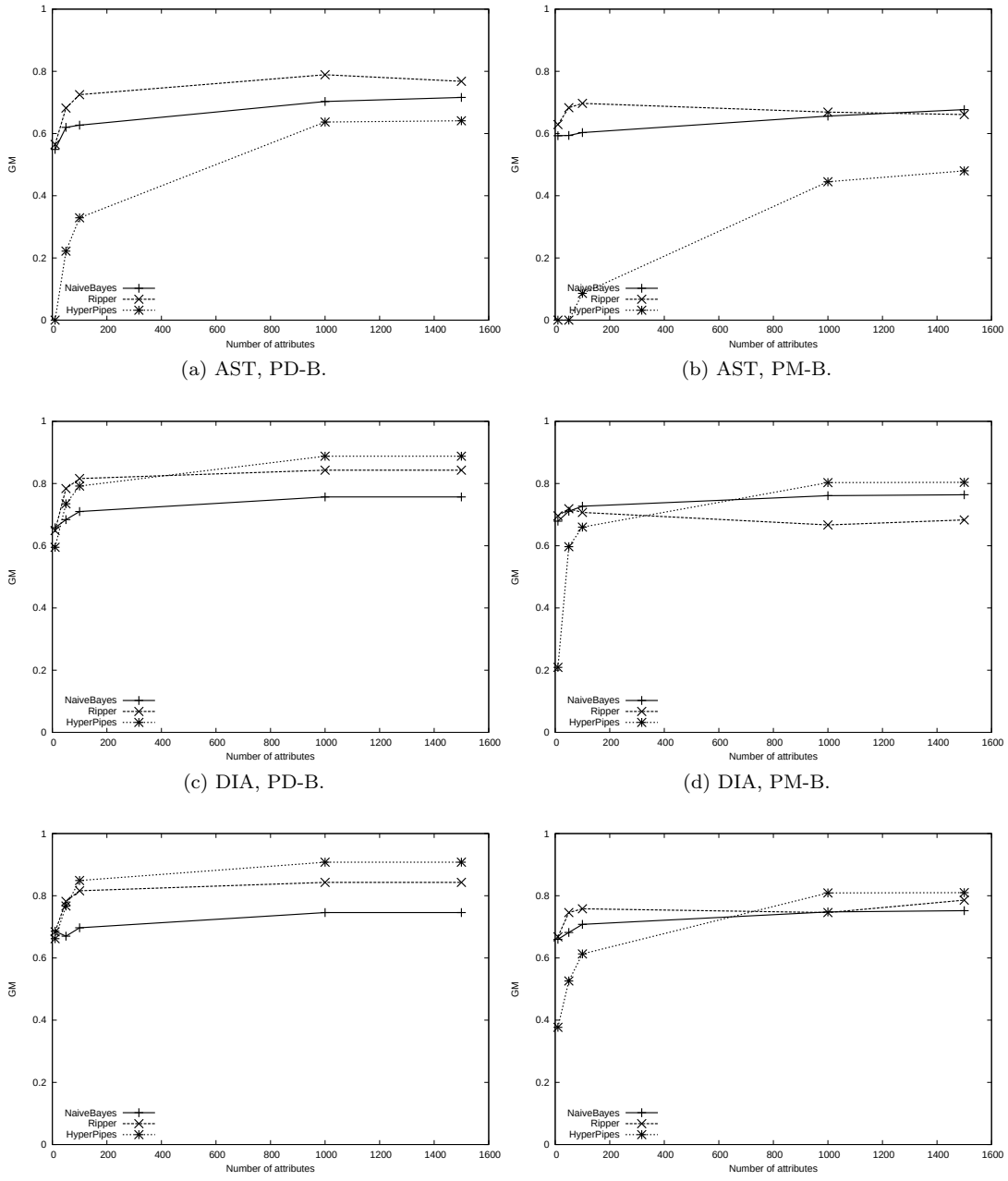
(c) DIA, PD-B.

(d) DIA, PM-B.

(e) HYP, PD-B.

(f) HYP, PM-B.

Figure 4.14: Performance of classifiers when using SMOTE. Evaluation metric: geometric mean, NA is max, NN varies.



(a) AST, PD-B.

(b) AST, PM-B.

(c) DIA, PD-B.

(d) DIA, PM-B.

(e) HYP, PD-B.

(f) HYP, PM-B.

Figure 4.15: Performance of classifiers when using SMOTE. Evaluation metric: geometric mean, NA varies, NN is 8.

4.5 E4: Comparing all classifiers when APP $\leq$ PP

Tables 4.12 and 4.13 compares the results of the six different classifiers used in this experiment. The results are only for asthma, using the two data sets. Since we focus on the classifiers and not the disease, we believe that important information is not lost by ignoring the results for diabetes and hypothyroidism.

The classifiers are sorted firstly by the number of attributes used (NA), then by their geometric mean (GM). We also present additional information for each classifier; the time necessary to build the model in seconds (time), the area under the curve (AUC) for the receiver operating charachteristics, sensitivity and specificity. Recall from the experiment description in section 3.7 that the results were obtained using iteration 3, NN = 1, PP = 913 and WT = 300.

We have not inspected the models for any of these classifiers in this experiment, as it is only RIPPER, C4.5 and to a certain extent NaiveBayes, which has transparent models. It is also considered too time consuming. Even though, it should definitely be done in order to verify that the models are useful. As it is, we can only present the results with respect to evaluation metrices.

4.5.1 Performance

The results for the PD-B data given in table 4.12, do not show that one classifier is definitely better than the others. SVM and C4.5 share the top position when NA = 100 with GM = 0.64, while K-Star and C4.5 are the best classifiers measured in GM when NA = 1000. We can say that C4.5 is the best classifier on average when the number of attributes varies. The algorithm has relatively higher sensitivity (0.78) than specificity (0.53), making it better at identifying positives than negatives. In terms of AUC, NaiveBayes is the best classifier, regardless of NA value. It is not much difference between C4.5 and NaiveBayes in terms of AUC when NA = 100, but NaiveBayes is judged considerably better when NA = 1000. HyperPipes is the classifier with highest sensitivity, but it has very poor performance in terms of specificity, thus it is ranked as the worst classifier. It is worth noting that SVM has the opposite charachteristics of HyperPipes; it is much better at identifying negatives than positives.

If we shift our attention to table 4.13, HyperPipes is still the worst classifier. It is apparent that 100 attributes is not enough to distinguish the classes, as HyperPipes has a GM of 0.18 because of low specificity. There is not much difference between the other classifiers in terms of GM, while AUC favours NaiveBayes as the best classifier. Finally, we note that there is not any consistent difference between classifiers if we compare the two data sets.

4.5.2 Time to build

From table 4.12 one can see that the cost of building each model with respect to time is highly dependent on the number of attributes. SVM uses 1.93 seconds to build its model when NA = 100, a number which increases to 18.07 when NA = 1000. The time increases with 836%, when the NA increases with 1000%, thus the time scales almost linearly with the number of attributes. The C4.5 algorithm scales even worse than SVM, the time increases with exactly 2000%, while RIPPER increases with 844%. Thus it is obvious that there is much time to save if one can use a subet of the features instead of the full set.

NaiveBayes and HyperPipes are extremely fast compared to the other classifiers. We do not have enough decimal places in order to see how long time they use when NA = 100, so we

Classifier	Time (sec.)	AUC	Sens	Spec	GM	NA
SVM	1.93	0.64	0.60	0.69	0.64	100
C4.5	0.42	0.67	0.78	0.53	0.64	100
NaiveBayes	0.00	0.69	0.73	0.49	0.60	100
K-star	0.00	0.62	0.73	0.49	0.59	100
RIPPER	0.09	0.56	0.69	0.45	0.55	100
HyperPipes	0.00	0.62	0.98	0.20	0.45	100
K-star	0.00	0.61	0.48	0.71	0.58	1000
C4.5	8.82	0.59	0.66	0.50	0.58	1000
NaiveBayes	0.03	0.67	0.71	0.46	0.57	1000
RIPPER	0.85	0.57	0.54	0.57	0.56	1000
HyperPipes	0.01	0.61	0.81	0.34	0.52	1000
SVM	18.07	0.57	0.24	0.91	0.46	1000

Table 4.12: Comparison of classifiers for asthma using PD-B.

cannot say anything about how well they scale with the number of attributes. K-star is the classifier which uses least time building a model. The reason is that it does not build any model at all. It is first when it is going to classify a new case that it does any work, see section 2.4.2.

However, if we look at how long time it takes to cross-validate each classifier, the picture is quite different for K-star. Cross-validation is used to evaluate the performance of a classifier (see section 2.4.3), by training and testing the classifier five times. The time it takes to cross-validate each classifier combines the time to build models and the time spent classifying, thus we get the total picture of how fast the classifier is. It takes 1151.94 seconds to cross-validate K-star when NA = 1000, the corresponding times for SVM, NaiveBayes, C4.5, RIPPER and HyperPipes are 902.12, 0.45, 1102.82, 18.49 and 0.12 seconds. Thus we see that K-star needs most time, because it takes much longer time to classify examples than the other classifiers.

A brief check on table 4.13 show that it takes considerably longer time to build a model, when using PM-B instead of PD-B. The C4.5 algorithm needs 520.26 seconds to build a model when NA = 1000, the corresponding time in table 4.12 was 8.82. The algorithm uses 5800% more time on building the model for PM-B than PD-B. Since the number of attributes is the same in both cases, the difference here is because PM-B provides more training examples than PD-B. PM-B has 3216 examples, while PD-B has 802 examples, thus PM-B has 301% more examples than PD-B. The time needed to build a C4.5 model grows exponentially.

4.6 E5: Varying settings when APP = PP

The only difference between E2 and this experiment is that APP $\leq$ PP for E2, while APP = PP in E5. It is therefore interesting to compare the results of E5 with E2. Comparing corresponding figures from the two experiments did not show any distinctive trends, it proved to be more interesting to compare the models themselves. For this reason, the figures which compare results from E2 and E5 have been placed in Appendix B.1.

Classifier	Time (sec.)	AUC	Sens	Spec	GM	NA
SVM	29.24	0.63	0.60	0.67	0.63	100
RIPPER	0.78	0.62	0.68	0.56	0.62	100
K-star	0.00	0.64	0.77	0.50	0.62	100
C4.5	1.17	0.61	0.76	0.50	0.62	100
NaiveBayes	0.01	0.66	0.73	0.51	0.61	100
HyperPipes	0.00	0.52	1.00	0.03	0.18	100
C4.5	520.26	0.63	0.69	0.56	0.62	1000
RIPPER	3.89	0.61	0.65	0.59	0.62	1000
NaiveBayes	0.05	0.69	0.74	0.51	0.61	1000
K-star	0.00	0.63	0.69	0.55	0.61	1000
SVM	294.40	0.60	0.39	0.81	0.56	1000
HyperPipes	0.02	0.62	0.97	0.20	0.44	1000

Table 4.13: Comparison of classifiers for asthma using PM-B.

4.6.1 The number of training examples in E2 and E5

Figure B.1 shows results from the PD-B data set. There are only small variations when comparing E5 results with E2, there are no distinct trends which repeat themselves. Figure B.2 which has the results for PM-B shows even less variation. There are occasional deviations, for example sub figure B.2c shows that at PP = 1825, both RIPPER and NaiveBayes had a distinct improvement compared to the same situation in E2. The models for RIPPER in E2 and E5 for this case are presented in Model 4.6.1 and 4.6.2 respectively.

Model 4.6.1 *E2:*
(ALAT = f) and (NONE = t) => Diabetes=t (402.0/105.0)
(ALAT = f) and (FERRITIN = f) => Diabetes=t (646.0/294.0)
(HEMOGLOBIN = f) and (URAT = f) and (HB = f) => Diabetes=t (21.0/5.0)
(FOSFOLIPIDER = t) => Diabetes=t (8.0/3.0)
=> Diabetes=f (293.0/14.0)

Model 4.6.2 *E5:*
(HEMOGLOBIN = f) and (ASAT = f) => Diabetes=t (157.0/37.0)
=> Diabetes=f (99.0/7.0)

From these models, we can see that in E2, there were 686 negatives and 684 positives, while in E5 the corresponding numbers were 129 and 127. The E2 model has more rules than E5. The great difference in the number of training examples can explain why the E5 model is simpler, fewer examples give less information to generalise over. Even though E5 has higher score than E2 for this case, the E5 model is too simple to be of any real use. The only rule which is used to identify diabetes-patients requires that the patient must not have taken the hemoglobin and asat-tests. This might make sense in the training set, but not in a real-world situation.

We compare these models with similar models at PP = 183 instead of 1825 in Model 4.6.3 and 4.6.4. Counting the number of examples show that there are 684 positives and 686 negatives in E2, while E5 has 421 positives and 423 negatives. Thus the number of examples does not change for E2 from PP = 1825 to PP = 183, but there are more than three times as many positives when PP = 183 compared to PP = 1825. This is expected, since the requirement

that APP = PP will cause fewer positives to be accepted as training examples.

Model 4.6.3 *E2:*
(ALAT = f) and (FERRITIN = f) and (NONE = t) and (SYSTOLISK = t) => Diabetes=t (69.0/13.0)
(SR = f) and (HEMOGLOBIN = f) and (CRP = f) and (DIASTOLISK = t) => Diabetes=t (51.0/11.0)
(FERRITIN = f) and (ALAT = f) and (KOLESTEROL = f) and (D01 = f) and (CRP = f) and (TRIMETOPRIM = f) => Diabetes=t (797.0/290.0)
=> Diabetes=f (453.0/81.0)

Model 4.6.4 *E5:*
(HEMOGLOBIN = f) and (HB = f) and (CRP = f) => Diabetes=t (534.0/184.0)
=> Diabetes=f (310.0/71.0)

By examining the models, we see that both E5 models are negation-models, they do not require that any of the attributes should be true. The E2 models both use the positive indicator NONE, while FOSFOLIPIDER, SYSTOLISK and DIASTOLISK occur in one of the models, but not both.

4.6.2 Positive indicators for different PP values

The results presented in this section shows how different sizes of the prediction period affects the positive indicators used in the RIPPER models for E2 and E5. We choose to extract the results from the situation where WT = 31, since there is an indication that E5 gives slightly better results when WT is 31 and 100 days long, as can be seen in figures B.3c, B.3e and B.4d.

Table 4.14 summarises the positive indicators for all combinations of dataset and diseases with respect to different PP-sizes. The event R78 (acute bronchitis/bronchiolitis) is a positive indicator for asthma in PD-B in all sizes of PP. From this we can say that it is an indicator which is present within 1 - 2 months prior to t_{sus}. Its presense when PP = 1825 means that patients who have this event also have a patient history which is at least 1856 days (PP + WT) prior to t_{sus}. With the exception of NONE and R78, there are no other positive indicators which occur in more than one of the PP-sizes. Also note that there are many PP-sizes which do not have any positive indicators at all. In those cases the RIPPER model only use negative indicators to determine if the patient has the disease or not.

If we compare the results from E5 in table 4.14 with the E2 results in table 4.15, one big difference is that there are more positive indicators in E2 than in E5 when PP = 1825. If we exclude the NONE attribute, there are 9 positive indicators for E2 and 4 for E5.

4.6.3 Positive indicators for different NA values

We present the positive indicators for different NA values in table 4.16 for E5 and table 4.17 for E2. WT = 300 for these results, which is 10 times as long WT as for the results showed in section 4.6.2. It would have been an advantage to use the same value for WT in both cases, but we did not have results for the combination WT = 31 and varying values for NA.

From table 4.16 one can see that there are no positive indicators for the diseases when the data was extracted from PD-B. This indicates that WT = 300 hides the positive indicators which were found in section 4.6.2, or that it hides the majority of them. Another remark is that for *PM-B, AST*, there are positive indicators when NA = 100 and 1000, but not when all

	PP = 31		PP = 365		PP = 1825	
PD-B, AST	1	R78	1	R78	1	USTIX
					1	R78
PM-B, AST			1	FOSFAT	1	SERETIDE
					1	IGE
PD-B, DIA						
PM-B, DIA	1	SYSTOLISK	1	ALKALISK		
	1	NONE				
PD-B, HYP			1	CYTOLOGI		
PM-B, HYP			1	NONE		

Table 4.14: Positive indicators from E5 and their frequency from RIPPER models for different dataset-disease combinations and PP-values. WT = 31, iteration 3.

	PP = 31		PP = 365		PP = 1825	
PD-B, AST	1	R78	1	R78	1	R81
	1	EVF	1	R74	1	R78
					1	R74
					1	R05
PM-B, AST						
PD-B, DIA					1	HSTAT
PM-B, DIA			1	NONE	1	NONE
			1	K86	1	K86
			1	DIASTOLISK		
PD-B, HYP						
PM-B, HYP	1	NONE	1	TESTTAPE	1	TESTTAPE
			1	NONE	1	NONE
					1	KOMBINERT VAG-CERV
					1	DIASTOLISK

Table 4.15: Positive indicators from E2 and their frequency from RIPPER models for different dataset-disease combinations and PP-values. WT = 31, iteration 3.

	NA = 100		NA = 1000		NA = All	
PD-B, AST						
PM-B, AST	1	FE	1	FE		
	1	A29	1	A29		
	1	HB				
PD-B, DIA						
PM-B, DIA	1	-30	1	SYSTOLISK	1	SYSTOLISK
PD-B, HYP						
PM-B, HYP	1	NONE	1	NONE	1	NONE
	1	KOLESTEROL				

Table 4.16: Positive indicators from E5 and their frequency from RIPPER models for different dataset-disease combinations and NA-values. NN = 1, WT = 300, PP = 913 and iteration 3.

	NA = 100		NA = 1000		NA = All	
PD-B, AST						
PM-B, AST						
PD-B, DIA	1	SR				
PM-B, DIA	1	NONE	1	NONE	1	NONE
			1	DIASTOLISK		
PD-B, HYP						
PM-B, HYP	1	NONE	1	NONE	1	NONE

Table 4.17: Positive indicators from E2 and their frequency from RIPPER models for different dataset-disease combinations and NA-values. NN = 1, WT = 300, PP = 913 iteration 3.

attributes are included. The reason could be that the extra number of attributes in NA = All is noise which masks these suspicion indicators. A29 (general symptom/complaint, other) is a very general disease description which could be used for very many conditions. The second diagnosis code -30 (medical examin/health eval complete) is at best a weak connection to diabetes, thus neither of these two are of any great value alone.

Table 4.17 consists of many models without any positive indicators and there are only two of them in total if we ignore NONE. As in the previous table, there is reason to believe that the WT is too large.

4.7 E6: Increasing positives when APP = PP

The figures which compare the performance of classifiers from E6 with E3 have been placed in appendix B.2. This section presents 12 lists which contain positive indicators for E6 (List 1-6) and E3 (List 7-12) for different NN-values. We use lists instead of tables because there were too many indicators. Recall that we use SMOTE in both E3 and E6, thus there are equal numbers of positives and negatives. If NN = 2, 50% of the positives are synthetical examples, NN = 4, 75% are synthetical examples and so on.

The tendency is that we find more positive indicators in E3 than E6.

List 1: Asthma, PD-B. Positive indicators from E6 and their frequency from RIPPER models for asthma and PD-B, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1:
- NN = 2: S87 (1), R91, R81, R78, MCV, CRP, ALP.
- NN = 3: GLUCOSE (2). R95 (1), R91, R78, NATRIUM, MCV, HB, H71, FE, CRP, ALBUMIN, ALAT.
- NN = 4: MCV (5). FT4 (3). R74 (2), L79, HB, E-MCV, ALP. X05 (1), URAT, S87, S08, R95, R91, R81, R78, N01, LD, K, H71, GLUCOSE, G-GT, FEMET, F71, F13, EVF, D82, CYTOLOGI, ASAT, AMYLASE, ALAT, ACHYM, A04.

List 7: Asthma, PD-B. Positive indicators from E3 and their frequency from RIPPER models, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1: S87 (1), R78.
- NN = 2: R78 (2), MCV. S87 (1), R81, GLUCOSE, CRP, ALBUMIN.
- NN = 3: MCV (4), CRP. USTIX (3). R78 (2), R74. TSH (1), SR, S87, R91, R77, PSA, NATRIUM, MCH, H71, FT4, F71.
- NN = 4: USTIX (3), R78, CRP. R74 (2), MCV, MCH, E-MCV. URIN/TRANSPORTAGAR (1), TOTPR, S87, S18, R91, R81, R77, R75, NATRIUM, KREATININ, FT4, AMYLASE, A12.

List 2: Asthma, PM-B. Positive indicators from E6 and their frequency from RIPPER models, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1:
- NN = 2: SR (1), R74, HB, GT, G6, FE.
- NN = 3: GT (2), FOSFAT, A29. STRIMLER (1), R74, LEUCOCYTTER, KATT, K+, HB, FT4, BILIRUBIN.
- NN = 4: X27 (1), KOBALAMINER, FOSFAT, DIASTOLISK.

List 8: Asthma, PM-B. Positive indicators from E3 and their frequency from RIPPER models, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1:
- NN = 2: NONE (2). SYSTOLISK (1), SR, R05, LIVOSTIN, BILIRUBIN.

- NN = 3: NONE (4). HB (2), FT4, FOSFAT, BILIRUBIN. SYSTOLISK (1), SR, NATRIUM, KOBALAMINER, KALSIUM, HEMOGLOBIN, GLUCOSE, DIASTOLISK, BLOD, ANTI-TPO.
- NN = 4: CRP (4). SR (3), KOLESTEROL, FT4. TRIGLYCERIDER (2), NONE, KOBALAMINER, FOSFAT, ASAT, ALBUMIN. X27 (1), TRIJODOTHYRONIN, SYSTOLISK, SOLBÆRSIRUP/EFEDRIN, R74, R05, M2, LEUCOCYTTER, KREATININ, KETONER, KALSIUM, IGE, HEMOGLOBIN, GLUCOSE, FERRITIN, DIASTOLISK, ALP, ALAT, A29, A03.

List 3: Diabetes, PD-B. Positive indicators from E6 and their frequency from RIPPER models, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1:
- NN = 2:
- NN = 3: L76 (1).
- NN = 4:

List 9: Diabetes, PD-B. Positive indicators from E3 and their frequency from RIPPER models, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1: L81 (1), HSTAT.
- NN = 2: L81 (1), HSTAT.
- NN = 3: HSTAT (1).
- NN = 4: SR (2), L81, L79, HB. R78 (1), P76, L84, L76, H71, ERYTROCYTTER, CYTOLOGI.

List 4: Diabetes, PM-B. Positive indicators from E6 and their frequency from RIPPER models, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1: SYSTOLISK (1).
- NN = 2: NONE (2). SYSTOLISK (1).
- NN = 3:
- NN = 4: NONE (3). HEMOGLOBIN (2), CRP. TSH (1), SYSTOLISK, SR, -30.

List 10: Diabetes, PM-B. Positive indicators from E3 and their frequency from RIPPER models, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1: SYSTOLISK (1), NONE.
- NN = 2: SR (2), NONE. SYSTOLISK (1).
- NN = 3: KOLESTEROL (1).
- NN = 4: SYSTOLISK (1), HDL.

List 5: Hypothyroidism, PD-B. Positive indicators from E6 and their frequency from RIPPER models, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1:
- NN = 2:
- NN = 3:

- NN = 4:

List 11: Hypothyroidism, PD-B. Positive indicators from E3 and their frequency from RIPPER models, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1:
- NN = 2: UD (1).
- NN = 3: UM (1).
- NN = 4: X87 (1), USTIX, UM, UD, G01AF02.

List 6: Hypothyroidism, PM-B. Positive indicators from E6 and their frequency from RIPPER models, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1: NONE (1).
- NN = 2: TESTTAPE (1), NONE.
- NN = 3: NONE (2). TESTTAPE (1), LD, DIASTOLISK, CERVIX, NONE, TESTTAPE, LD, DIASTOLISK, CERVIX.
- NN = 4: NONE (3). HEMOGLOBIN (2). TESTTAPE (1), SOPP, KOLESTEROL, DIASTOLISK.

List 12: Hypothyroidism, PM-B. Positive indicators from E3 and their frequency from RIPPER models, with varying NN and SMOTE. PP = 913, WT = 300, NA = 1000, iteration 3.

- NN = 1: TESTTAPE (1), SYSTOLISK, NONE.
- NN = 2: NONE (3). SYSTOLISK (2). SR (1), KOMBINERT VAG-CERV, HEMOGLOBIN, ALP.
- NN = 3: NONE (6). TESTTAPE (1), SYSTOLISK, LOBAC, KOMBINERT VAG-CERV, KOLESTEROL, KALIUM, HEMOGLOBIN, GLUCOSE, FLUANXOL, DIASTOLISK.
- NN = 4: NONE (7). HEMOGLOBIN (6). SYSTOLISK (4). URAT (2), TESTTAPE, GLUCOSE. URINMIKROSKOPI (1), SR, LD, KALIUM, FERRITIN, CERVIX, BLODTRYKK, ALP.

Chapter 5

Discussion

This chapter presents seven sections, which discuss positive indicators found in the models (section 5.1), problems concerning the way SMOTE is evaluated (section 5.2), the limitations imposed by the data sets (section 5.3), the potential usefulness of the prediction models (section 5.4), the information we can retain from the models (section 5.5), the choice between rich FVs or many examples (section 5.6) and a discussion of how to select the best evaluation metric for a skewed class distribution (section 5.7).

5.1 Positive indicators

In the results we introduced the term *positive indicators* as an attribute which is required to be true in order to conclude that the patient has the target disease (TD). These positive indicators were extracted from the RIPPER models, since such models consist of rules which are easily understood by humans. Section 5.1.1 will argue why positive indicators are interesting, section 5.1.2 discuss where they occur in the patient history and finally we look at how one can determine the validness of positive indicators.

5.1.1 Why are positive indicators interesting?

We begin by repeating part of the information about RIPPER models given in 4.1 as part of the discussion. In this project, a RIPPER model consists of rules where all but the last rule conclude that the patient has the TD. In the example model 5.1.1, the TD is diabetes and there are three rules which conclude that the patient has diabetes. If none of these three rules match the patient we are going to classify, the model concludes that the patient does not have diabetes. There are two positive indicators in this model, ALKALISK and FOSFOLIPIDER, while the remaining attributes are negative indicators.

Model 5.1.1 *Example:*
(ALAT = f) and (FERRITIN = f) => Diabetes=t
(ALAT = f) and (ALKALISK = t) => Diabetes=t
(FOSFOLIPIDER = t) => Diabetes=t
=> Diabetes=f

A model which has no positive indicators was termed a negation-model in 4.1. Assume that ALKALISK and FOSFOLIPIDER were negative indicators in model 5.1.1, it would then be a negation-model. If a negation-model was applied in a real-world situation, all patients who visited the GP for the first time would be classified as diabetes patients. Their patient history is without events, thus all attributes in their feature vector will be false and then all rules consisting of only negative indicators will match the patient. Negation-models are strongly biased to say that perfectly healthy people are sick, thus they are illogical and we regard them as invalid models.

The class of invalid models can be extended further. Since rules in a RIPPER model are applied in a top-down fashion, it is actually enough to look at the first rule in order to determine if the model will be invalid. If the first rule only consists of negative indicators, it will have the same effect on patients without events in their patient history as described in the previous paragraph. If we apply this new definition of an invalid RIPPER model to example model 5.1.1, it is now considered invalid, whereas the old definition judged it as valid since it had two positive indicators.

As a consequence, positive indicators which appear in the first rule of a model are deemed most interesting, because they occurred in a model which is valid. Positive indicators which occur in an invalid model may still represent useful knowledge, thus they are also interesting. We claim that all RIPPER models must have at least one or more positive indicators in order to be considered interesting at all. For this reason, we believe that positive indicators are more important than negative indicators in general. Negative indicators can be used to refine a group of patients which share a positive indicator, so the negative indicators are not necessarily irrelevant, but less important.

From now on, we term positive indicators which appear in a valid model to be *strong positive indicators*, while positive indicators occurring in invalid models are *weak positive indicators*.

5.1.2 Where do the positive indicators occur?

The results from the experiments were analysed with respect to which positive indicators occurred in the RIPPER models in a variety of situations. The positive indicators presented in the results did not distinguish between strong and weak positive indicators. In this section, we present the most prominent trends which show in which situations the positive indicators occur.

A trend which was shown in E1 was that the positive indicators decreased in number when analysing the models from iteration 1 - 3 (see section 4.2.3). In addition, the positive indicators changed from one iteration to the next. The decrease is understandable since the task of identifying patients with the TD becomes more challenging as the estimate of t_{sus} is moved further and further away from t_{TD}. It is also natural that the classifiers must use other positive indicators in the different iterations. A good example is that some of the positive indicators in iteration 1 and 2 are also suspicion indicators and since we have decided to use information from the time before they appear, it is obvious that they do not appear in iteration 3.

In section 4.6.2 we examined which positive indicators were found when varying the size of PP. The results were not conclusive, but there was an indication that longer PP resulted in more positive indicators. This trend could be seen both when APP = PP and APP $\leq$ PP. The main difference between the two variants was that APP $\leq$ PP resulted in models with more positive indicators. The advantages and disadvantages of these two variants are discussed further in section 5.6. Even though longer PP results in more positive indicators, the size of the PP

NA	TP	FP	TN	FN	Sens	Spec	GM
100	619	343	601	323	0.66	0.64	0.65
1000	635	360	584	307	0.67	0.62	0.64
All	690	455	489	252	0.73	0.52	0.61

Table 5.1: Comparison of RIPPER models in terms of TP, FP, TN and FN.

should be limited, since we assume that the probability that two events are related decreases with increasing distance between the events. Even though there might exist events which are early positive indicators for the TD, a PP long enough to include this event will also include more irrelevant events. The irrelevant events are noise which makes it harder to identify the indicators which are truly relevant.

The tables presented in section 4.6.3 showed which positive indicators were found when analysing models with 100, 1000 and all attributes. Given that the attributes selected for these models were the most relevant attributes, one might expect the models to be more valid, thus containing more positive indicators. The number of positive indicators was at a minimum when all attributes were used, in most cases no positive indicators were present at all. Reducing the number of attributes to 1000 and 100 yielded a few more positive indicators. From these results it seems that the feature selection process has removed irrelevant attributes, thus it has made the task of finding positive indicators easier for the RIPPER algorithm.

5.1.3 Do positive indicators affect performance?

-even though there is little change in performance, models change, and contain fewer or none positive indicators)RIPPER

In section 5.1.1 we argued why positive indicators are more useful than negative indicators, but we have not discussed how positive indicators affect performance. Table 4.16 shows that the models for asthma, built from PM-B, have 3, 2 and 0 positive indicators for the NA-values 100, 1000 and all. Figure B.8a shows that the performance for these three models is practically the same with respect to geometric mean. The rules of the models are presented below.

Model 5.1.2 *NA = 100:*
(HEMOGLOBIN = f) and (TRIGLYCERIDER = f) and (A29 = t) => Asthma=t (114.0/29.0)
(HEMOGLOBIN = f) and (KOLESTEROL = f) and (PINEX = f) => Asthma=t (780.0/296.0)
(HB = t) and (FE = t) => Asthma=t (68.0/18.0)
=> Asthma=f (924.0/323.0)

Model 5.1.3 *NA = 1000:*
(HEMOGLOBIN = f) and (A29 = t) => Asthma=t (141.0/39.0)
(HEMOGLOBIN = f) and (FE = t) => Asthma=t (84.0/21.0)
(KOLESTEROL = f) and (HEMOGLOBIN = f) => Asthma=t (770.0/300.0)
=> Asthma=f (891.0/307.0)

Model 5.1.4 *NA = All:*
(HEMOGLOBIN = f) => Asthma=t (1145.0/455.0)
=> Asthma=f (741.0/252.0)

Model 5.1.2 and 5.1.3 both has A29 as a strong positive indicator, while model 5.1.4 has no positive indicators, thus it is an invalid model according to our definition in section 5.1.1. It

is also a less complex model, with only one rule and one antecedent. Despite the differences, these models are deemed practically even by the performance metric geometric mean as can be seen in table 5.1.

This example shows that it is not enough to use only the evaluation metric when evaluating models with respect to practical use. It is necessary to analyse the models in order to validate that they do not make decisions based on nonsense. In situations where the models cannot be manually inspected, for example support vector machines (SVM) or K-star, the models must be subjected to a more thorough test procedure.

5.1.4 Determining the validness of positive indicators

In order to determine the validity of the positive indicators found in the results section, it is necessary to use domain experts which are knowledgable with respect to the three TDs. This project has identified a number of positive indicators, but it is outside the scope of this project to determine if they are relevant from a medical point of view.

5.2 Invalid SMOTE results because of circular reasoning

It is currently problematic to evaluate the validitiy of the results from the experiments using SMOTE. The reason is that SMOTE is applied to the examples before the cross validation procedure begins. This means that all folds will contain synthetic positives. Since each fold is used as a test set once, all test sets will contain synthetic positives. We do not know how succesful SMOTE is in generating true positive examples, they could in fact be examples which belong to the negative class. This is something a test should reveal. Since the test set contains synthetic positives, it is not an objective test set, because we are assuming that the synthetic positives are correct by using them in the test set. This is called *circular reasoning*; SMOTE is used to prove itself. Thus we cannot draw any conclusions from these results.

This problem was not realised early enough to alter the experimental setup, thus we do not have any valid results in the experiments where SMOTE was used. It is difficult to determine whether the experiments conducted in [39] did the same mistake, they also used cross validation to evaluate their suite of classifiers. The SMOTE results were not excluded from this report, because the results were promising and we want to put focus on this sampling method as we believe it can be useful in this setting. We describe two alternative methods which can be applied to get valid results in the subsection below.

5.2.1 Methods to ensure valid results for SMOTE

SMOTE might do a good job of creating true positives, but in order to have valid test results we must use a test set with non-synthetic examples. One way to do it, is to postpone the generation of synthetic positives until the step before the classifier is built for each fold in cross validation. This procedure is described in more detail in the list below.

1. original examples are partitioned into n folds
2. one fold x is selected as test set, the remaining *n-1* folds is the training set
3. SMOTE uses the positives from the training set to create synthetical positives which are added to the training set

4. a classifier is built using the training set, and tested with the test set

An alternative way is to partition the original positives into a training and test set, use SMOTE to increase the number of positives in the training set, build classifiers and evaluate them with the test set. This approach applies SMOTE only once, compared to the n times it is applied in the procedure described in the list. But the last approach suffers from the same problems as standard use of only one training and test set, as described in section 2.4.3; the test result is too dependant on which examples end up in the training and test sets.

5.3 Limitations of data sets

There is a lot of potentially useful information which is not in the data sets. General information about patients such as age and gender is not included in the data sets. The information detail with respect to medical tests is also very rough; a test event from PD-B provides information whether it was abnormal or not, while PM-B does not even provide the result. It only states that a test has been taken. The journal notes which the GP writes at each visit could also have been used. In the rest of this section we discuss what how additional information could have been useful and why it was not included.

5.3.1 Potential value of additional information

Grouping patients based on age and gender is useful, because there are often large variations between groups. It would be possible to determine if there are differences between groups with respect to frequency and difficulty in diagnosing them correctly. Another aspect is that there might be different reasons which causes a disease for different groups, diabetes was traditionally known as a disease which was typical for old people; today the disease is an increasing problem for people under 40 years. When there is no way to separate these two groups, we cannot investigate these more specific questions.

More fine-grained information about the tests would also give more possibilities. Laboratory tests are ordered for different reasons and in different situations, presumably most of them show that everything is in order. These tests are not as interesting as the tests which return with an abnormal result, because abnormal results indicates that some part of the body is not as it should be. Especially tests which are used for screening purposes have little value if we do not remove the tests with normal results. Abnormal results are more rare, thus they are more likely to be of use when building a predictive model. One could go one step further and use the values from the test itself, then let the classifiers determine which values are abnormal themselves. This means more data and processing, thus there are negative consequences.

The journal notes is another important source of information, it contains the GP's comments from the patient visits as unstructured text. The text might contain symptoms which are too vague or for other reasons are not written into the EHR as separate events. The GP can also have hypotheses with respect to what the problem is; information which is less formal, but still with potential value.

5.3.2 Reasons for not including additional information

The additional information could have been useful, but it was not included, even though it existed in the data sources. A general issue is that with increasing amount of patient

information, it becomes less hard to identify the true identity of the patient. This was certainly one of the most important reasons for not including age and gender. Even though the data does not contain any direct identifiers such as name or address, each patient history is unique. It is therefore theoretically possible to find the patient history of a real person, given that one knows some of the diseases this person has had and that the person is in the data set. With age and gender this is even easier. However, researchers who have access to such data are bound by moral and legal laws, which forbids them to use data sets for such purpose.

The reason that the PM-B data set did not contain test results, was that the cost of extracting the data was too high. The structures of the two data sources were not the same, and these technical barriers blocked that information from reaching the data set. This was not the case with the journal notes, they were available but it was not enough time to utilise them in the experiments.

Determining how much information should be available for a research project has no correct answer. A golden guideline is to include as much information which is necessary in order to accomplish the research task. One could then iteratively use more and more information until the goal is met. The claim that additional information will always lead to improvement is not nuanced enough. One cannot say for sure until one has evaluated the results with and without the information. New kinds of information must be incorporated into the model, adding to the complexity of the system, thus increasing the time needed to implement it.

5.4 The potential usefulness of good prediction models

The main purpose of E1 was to identify the events which were used to estimate t_{sus} for each patient. In addition it served another purpose, which was to give an overview of how quickly the patients were suspected and diagnosed by the GP. This overview is vital when evaluating the assumption that *there are patients who should have been treated earlier*. If this assumption is false, a system which predicts diseases earlier than the GP would have no chance of success, because the GP treated all patients at the earliest time possible. As a result there would be no data in the EHR prior to t_{sus} which would be relevant in making predictions.

The results presented in Table 4.4, Section 4.2.2, showed that the GP is able to correctly suspect the TD at the first visit in 47% of the cases where a patient had hypothyroidism (PM-B data set). We do not know what has happened prior to this event since it is the first event recorded in the EHR for this patient in this GP office. A patient could have gotten the diagnose at her previous GP (GP A), then the patient started seeing a new GP (GP B) and told about her diagnoses on the first visit to GP B. Even though patients change GPs, one can assume that it does not explain all the patients who were discovered on their first visit.

If 47% of the hypothyroidism patients were suspected on their first visit, 53% were not suspected. This does not necessarily imply that the GP should have suspected them. Since hypothyroidism often evolves over a period of time, it is most likely that a large part of these 53% did not have the disease at the time of their first visit. They might visit the GP for many years because of other diseases, but at some point in time the patient would go from a state which does not fulfill the definition of hypothyroidism to a state where these requirements are fulfilled. We denote this time t_{dis} and give a definition; t_{dis} is the earliest time when it is possible to identify the disease with present medical knowledge.

Knowing t_{dis} would be useful since that would allow us to give an exact measure of how quickly the GP suspects the condition. Unfortunately we do not know t_{dis} and the best estimates we

can make are quite rough. Therefore we will use the estimate that t_{dis} is at the start of the PH.

With this estimate of t_{dis}, figure 4.5 shows that within 10 visits to the GP, only 16% of the patients with hypothyroidism have not been suspected by the GP (with respect to the PD-B data set). The corresponding numbers for asthma and diabetes are 52% and 14% respectively. The same tendency is seen for the PM-B dataset. The GP is much faster at finding patients with diabetes and hypothyroidism than asthma. It also means that there is less data available for diabetes and hypothyroidism patients, because most of them are discovered so quickly that their patient histories are practically empty. This makes it relatively harder to build good prediction models for diabetes and hypothyroidism, compared to asthma. It also weakens the potential usefulness of a prediction model for diabetes and hypothyroidism, because there are not many patients who are not discovered by the general practitioner.

5.5 What kind of information do we retain?

This section has a theoretical perspective, discussing what kind of information we can get from the different periods in the patient history/ electronic patient record, how certain periods are more dominant than others and the type of information we have extracted in our experiments.

5.5.1 Extracting different information from different periods

Section 2.1.2 introduced the terms t_{sus} and t_{TD} and have been used throughout the report. From the discussion in Section 5.4 a new term emerged; t_{dis} is the earliest time when it is possible to identify the disease with present medical knowledge. In this section we wish to say something about what a learning algorithm (LA) can learn from the different periods we get by splitting the timeline at these three events. In the rest of this section, the LA is trying to learn how to predict the TD using data from different periods. We begin with the period after t_{TD} and continue backwards in time.

When the diagnosis has been set at t_{TD}, the GP is certain that the patient has the TD. The GP must then decide how to treat the disease by making a treatment plan. Typical elements in such a plan which can be tracked in the EHR include prescriptions for medication, tests which can monitor the development of the disease together with regular visits to the GP. A LA which is applied on data from the period of EHR which is after t_{TD}, with the aim of learning which patients have the TD, will most probably use the disease specific medication and tests (assuming that we ignore the diagnosis event which states that the patient has TD).

In the period from t_{sus} to t_{TD} the GP suspects that the patient might have TD. As Project07 found out, the GP gives the patient medication for symptoms, which strongly implies certain diseases if they have a positive effect on the patient. Laboratory tests which can narrow down which disease the patient has are also taken. Thus, if the LA uses data solely from t_{sus} to t_{TD}, it would capture these tests and prescriptions for medication which is relevant for the disease. For many diseases, these events are the same as in the period after t_{TD}.

Prior to suspicion is the period from t_{dis} to t_{sus}. The disease is present in this period, but the GP has not suspected anything yet. If the patient visits the GP in this period, it is most likely that the events which are most correlated with the disease, are signs observed by the GP. Unless the sign can be classified as a disease itself, the sign might be recorded as a part of

DR	Period	Information used
1	t_{TD} -	Prescriptions and medical tests
2	t_{sus} - t_{TD}	Symptoms, prescriptions and medical tests
3	t_{dis} - t_{sus}	Symptoms
4	- t_{dis}	Causes

Table 5.2: DR is dominance rank, DOB is date of birth.

the journal note which the GP writes about the visit. But there is a risk that the sign is not recorded in the EHR at all.

The earliest period is the time prior to t_{dis}. If the research question is why the TD developed, data should be collected from this period. A disease can develop for a number of reasons; acute damage, inheritance, environment. Some of these reasons can lead to other diseases, which might develop earlier than the TD. The genome of the patient is not available through the EHR, thus a LA would not be able to use such data. There might be information about the environment of the patient in the journal notes, but not in the structured EHR. If the LA is able to find any connection between positives using data from this period, it will most probably be other diseases which occur because of the same underlying reason.

5.5.2 Dominant periods

In the previous description of the different periods, we ignored the fact that information which is found to be useful in one period, might occur in a later period. A disease D which is found to correlate with TD in the time prior to t_{dis} might show up in a later period as well, for example the period t_{sus} to t_{TD}. However, D will most likely be ignored by LA, because the other events in the suspicion period are more likely to have stronger correlation with the TD.

One reason later events correlate stronger with the TD is that there is more data available the closer we get to the TD, because it varies how quickly patients go to the GP with their problems. Since many patients are suspected on their first visit to the GP (see Table 4.4), they must have had the disease before they came on the visit. Thus we do not get any events from these patients for the periods preceding t_{sus}.

The claim is therefore that if data is collected from two periods, it is most likely that a LA will build its model based on information from the period which is most recent.

5.5.3 The information retained in the experiments

Based on the theoretical discussion in the two previous sections 5.5.2 and 5.5.1, what kind of information should we expect from the models made in the experiments? Assuming that we managed to find a good estimation of t_{sus} in E1, the models which were built using this estimate should collect their events from the period prior to suspicion.

siden jeg bruker en miks av hendelser fra flere perioder, er det mest sannsynlig at jeg sitter igjen med hendelser fra den seneste perioden, siden disse dominerer over eldre

From the machine learning perspective t_{dis} would enable us to choose more specific learning tasks. One could use the data from t_{dis} to t_{sus} in order to build models, knowing that we have

events from the period where the disease is present. This learning situation would be extra focused on symptoms that indicate the presence of the disease. This is different from trying to identify the causes which say why the disease occurred. In order to focus solely on causality, one needs to focus on the period prior to t_{dis}. Since we do not know t_{dis}, there are good chances that we use events before and after t_{dis} for many patients. Thus the models might discover both symptoms and causes for the disease.

5.6 Rich FVs and few examples or sparse FVs and many examples?

One of the most difficult tasks in this work, has been to balance the average number of non-zero features in FVs against the number of positives used for training. Recall that the problem definition constrains us to only use data from the period prior to t_{sus}. In order to use all positives, the WT must be 0 and we include all positives regardless of their length in this period. It is problematic to use the shortest positives for training, because they have very few events, resulting in only a handfull of non-zero features. Comparing to the number of features which are zero, the difference can be as large as three orders of magnitude in favor of the zero-valued features. Furthermore, we constrain the lengths of the negatives to equal the length of a positive (see Constraint 3.4.3 in Section 3.4.1), thus the FVs of the negatives also become very sparse with respect to non-zero features.

These sparse FVs make the task of learning harder, a phenomenon termed *the curse of dimensionality*[40]. Since each feature represents one dimension, each training example can be visualised as a data point in the multidimensional space of features. With an increasing number of dimensions, the distance between the data points increases and it becomes harder for the classification algorithm to build a good model.

An alternative approach which seeks to increase the number of non-zero features, is to require that the period prior to t_{sus} must be greater than or equal to the size of the PP. Positives which do not fulfill this requirement are not used. Thus we get richer FVs but fewer positives. Figure 4.4 shows how many positives will be left for training as we increase the PP. Note that if we have a non-zero WT, we must add the sizes of WT and PP in order to get the minimum length of the period start - t_{sus}. If WT = 400 and PP = 600, the period must be at least 1000 days long. Looking up how many positives fulfill this requirement for the three TDs, we are left with 15% of the patients with diabetes and hypothyroidism, and 45% of the patients with asthma.

Both these approaches have been tested in the experiments. Experiment 1, 2, 3 and 4 used the first method which prioritises many examples and sparse FVs, while experiments 5 and 6 prioritised few examples and rich FVs.

5.7 Choosing evaluation metric for skewed class distributions

Figures 5.1 and 5.2 presents nine different evaluation metrices which evaluate the same classifiers. One can immideately state that these figures give very different views of the results. According to the view of accuracy in Figure 5.1a, Ripper performs very well with increasing NN, while Naive Bayes is relatively unaffected and Hyperpipes is negatively affected.

We continue by looking at the two sub figures at the bottom of Figure 5.2; sensitivity and specificity. They measure how well the classifiers are at classifying the positives and negatives respectively. They show huge differences among the classifiers; Hyperpipes is absolutely best at finding most positives, with Naive Bayes being quite good as well. The Ripper algorithm begins well for NN = 1, but subsequently drops drastically and for NN = 4 and 8, it is not able to identify a single positive. The reason it still performs so well in terms of accuracy, is its high specificity; for NN = 4 and 8 it correctly labels all negatives. Hyperpipes shares the problem Ripper has; it only does well on one class when class distributions are imbalanced. It is very poor at classifying negatives correctly, but not as bad as Ripper is at finding positives. Naive Bayes is again second in terms of specificity and climbs from 0.5 to 0.6 with increasing NN.

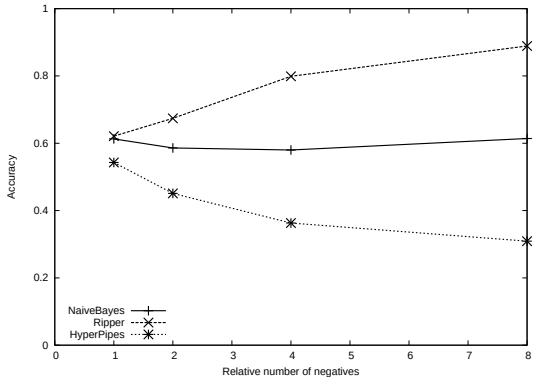
Having described the views of sensitivity and specificity, we present the views of *positive predictive value* (PPV), *negative predictive value* (NPV), *area under the curve* (AUC), *class-weighted mean accuracy* (CWA), *F-measure* and *Geometric mean* (GM). These metrices have already been described in Section 2.4.3, page 63. F-measure is the most pessimistic view in this situation, since the performance of all classifiers degrade with increasing NN.

F-measure depends on the PPV, sensitivity and a α value which decided the relative importance of these two. It does not take the negative class into account. The figure in this plot was made with an α value of 0.8, meaning that it emphasises sensitivity more heavily than PPV. This view would not improve much by shifting the weight over on PPV, because it is actually the PPV which contributes most to the low values for F-measure as seen in Figure 5.1b.

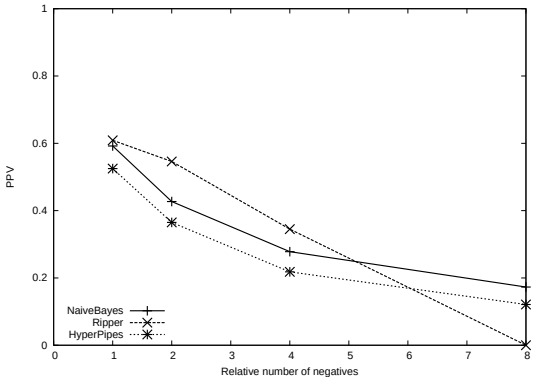
The views of PPV and NPV show opposite trends; PPV decreases with increasing NN, while NPV increases. This means that FP grows faster than TP and TN grows faster than FN. This is natural becuse we add extra negatives without increasing the number of positives. These are highly class-specific metrices which do not capture the whole picture.

Geometric mean and CWA are closely related since both use sensitivity and specificity, and one can see from their figures that they resemble each other. GM punishes Ripper and Hyperpipes harder than CWA for having so poor ability on the positives and negatives respectively. CWA is useful in those situations where the cost of misclassifying the two classes differ. In this case we have weighted the relative importancy of the positive class with 0.7, giving the negative class a weight of 0.3. This weighting-scheme will favour classifiers who are good at correctly identifying positives.

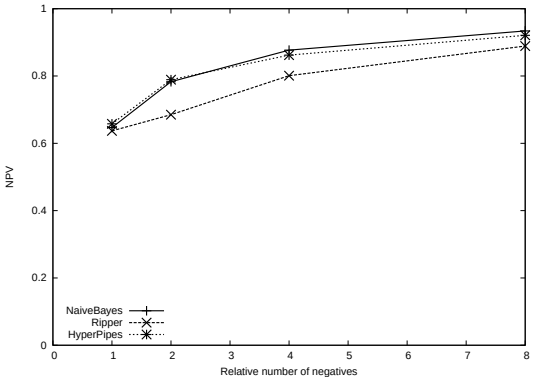
In this experiment, we have chosen to weigh the two classes with equal importance. The accuracy metric we have used up till now does not discriminate with respect to class. The natural choice is therefore geometric mean, which we will use whenever the class distributions are not equal.



(a) Accuracy.

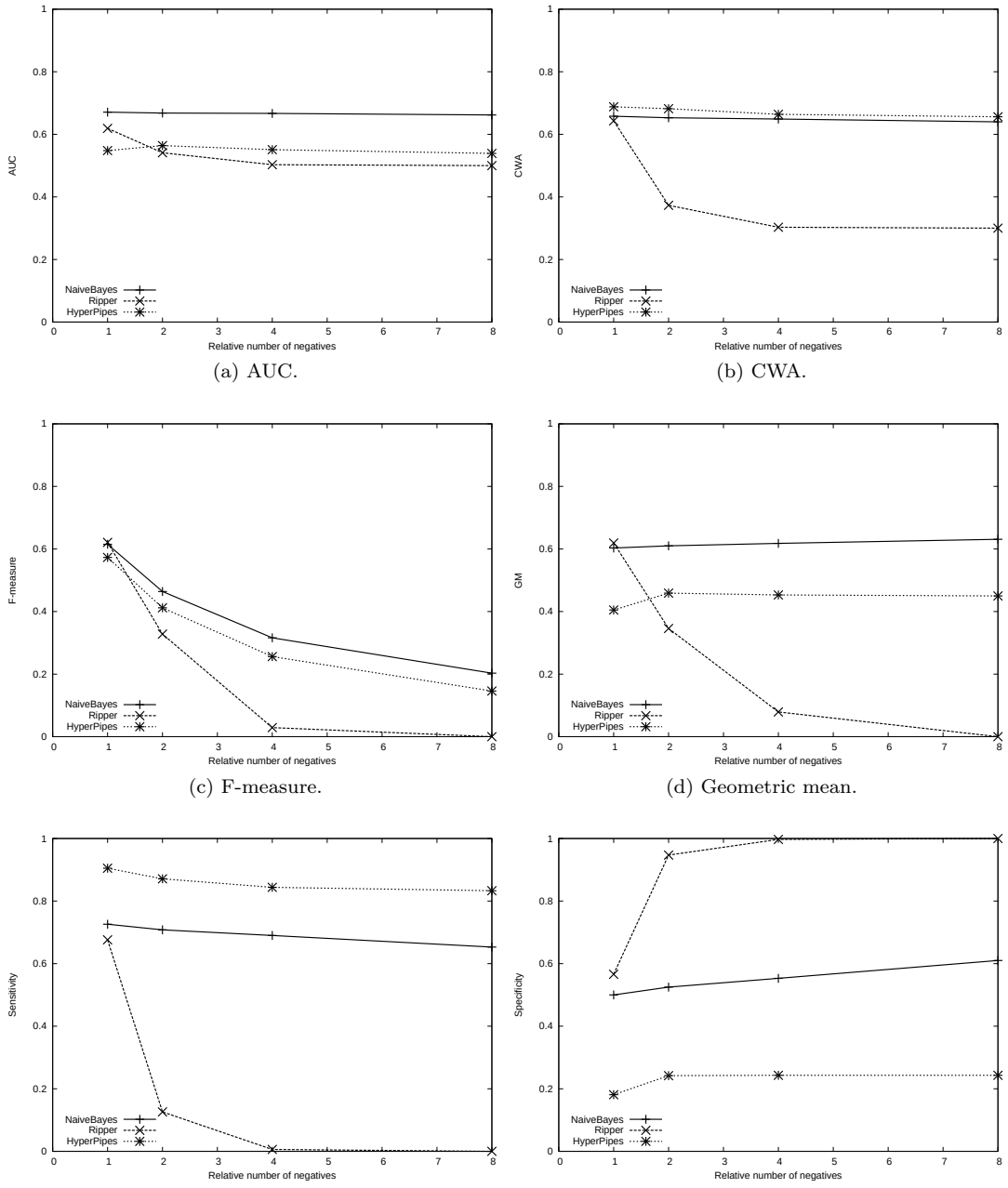


(b) PPV.



(c) NPV.

Figure 5.1: Performance of classifiers using different evaluation metrices when NN varies. All figures are for iteration 3, PM-B and asthma.



(a) AUC.

(b) CWA.

(c) F-measure.

(d) Geometric mean.

(e) Sensitivity.

(f) Specificity.

Figure 5.2: Performance of classifiers using different evaluation metrices when NN varies. All figures are for iteration 3, PM-B and asthma.

Chapter 6

Conclusion

The aim of this project was to find out if it is possible to warn the general practitioner about patients who might get a disease in the future. In order to improve the usefulness of our method, we have accounted for the situation where the general practitioner suspects the disease before the diagnosis is set. This has resulted in an improved formulation of the learning task, which leads to more useful models than before.

A set of variables has controlled the information which has represented each patient history. Predictive models have been built for different values of each variable, in order to find the more favourable values. The results are not conclusive, but they indicate that

- the warning time should be less than 300 days
- the number of attributes should be between 100 and 1000
- the class distribution should be balanced

We have manually analysed a subset of the models made by the RIPPER algorithm, and have identified the events we believe could be used to predict the target diseases. These should be scrutinised by domain experts in order to determine whether they are of any real value or not. Our manual analysis revealed that many of the models are practically useless, but this does not show when evaluating the models in terms of geometric mean. This advocates that models should always be manually inspected if possible.

Our comparison of six different classifier algorithms showed that there were no major difference among the five best classifiers.

Experiments which added extra positive examples using SMOTE show promising results, but the current evaluation method is deemed invalid, thus a more reliable evaluation methodology is required to make conclusions about the SMOTE method in this problem setting.

Bibliography

[1] *Intellectual Property Rights in Frontier Industries: Software and Biotechnology.* AEI Press, 2005.

[2] Linneberg A. Forekomst af allergisk luftvejssygdom i danmark. *Ugeskr LÃ¦ger*, 2004.

[3] Russell Ackoff. From data to wisdom. *Journal of Applies Systems Analysis*, 1989.

[4] Stig Alvestad, Øystein Nytrø, and Ole Edsberg. Early warnings of critical diagnosis, 2007.

[5] Norsk Helseinformatikk AS. Astma - norsk elektronisk legehåndbok. `http://www.legehandboka.no/`, Version of 12th of December 2008.

[6] Thorsby P et al Birkeland KI, Kilhovd B. Heterogeneity of non-insulin-dependent diabetes expressed as variability in insulin sensitivity, beta-cell function and cardiovascular risk profile. *Diabet Med*, 2003.

[7] Sklar C, Whitton J, and Mertens A et al. Abnormalities of the thyroid in survivors of hodgkin's disease: data from the childhood cancer survivor study. *J Clin Endocrinol Metab*, 2000.

[8] Roberts CGP and Ladenson PW. Hypothyroidism. *Lancet*, 2004.

[9] Chih-Chung Chang and Chih-Jen Lin. Libsvm - a library for support vector machines, 2001. The Weka classifier works with version 2.82 of LIBSVM.

[10] Aram V. Chobanian, George L. Bakris, and Henry R. Black et al. Seventh report of the joint national committee on prevention, detection, evaluation, and treatment of high blood pressure. *Journal of Applies Systems Analysis*, 2003.

[11] John G. Cleary and Leonard E. Trigg. K*: An instance-based learner using an entropic distance measure. In *12th International Conference on Machine Learning*, pages 108–114, 1995.

[12] Dayan CM and Daniels GH. Chronic autoimmune thyroiditis. *N Engl J Med*, 1996.

[13] Gilles Cohen, Melanie Hilario, Hugo Sax, Stephane Hugonnet, and Antoine Geissbuhler. Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine- 2006 May (Vol. 37, Issue 1)*, 2006.

[14] William W. Cohen. Fast effective rule induction. In *Twelfth International Conference on Machine Learning*, pages 115–123, 1995.

[15] Stampfer MJ Manson JE Hennekens CH Arky RA et al Colditz GA, Willett WC. Weight as a risk factor for clinical diabetes in women. *Am J Epidemiol*, 2000.

[16] Phillips D, McLachlan S, and Stephenson A et al. Autosomal dominant transmission of autoantibodies to thyroglobulin and thyroid peroxidase. *J Clin Endocrinol Metab*, 1990.

[17] Strachan DP. Family size, infection and atopy: the first decade of the "hygiene hypothesis". *Thorax*, 2000.

[18] Becker DV and Hurley JR. Complications of radioiodine treatment of hyperthyroidism. *Semin Nucl Med*, 1971.

[19] Yasser EL-Manzalawy. Wlsvm, 2005. You don't need to include the WLSVM package in the CLASSPATH.

[20] Nitesh V. Chawla et. al. Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[21] Delange F, de Benoist B, Pretell E, and Dunn JT. Iodine deficiency in the world: where do we stand at the turn of the century? *Thyroid*, 2001.

[22] Foreningen for utgivelse av Norsk legemiddelhåndbok. Norsk legemiddelhåndbok. `http://www.legemiddelhandboka.no/xml/`, Date: 9th of January 2009.

[23] Sean N. Ghazavi and Thunsun W. Liao. Medical data mining by fuzzy modeling with selected features. *Artificial intelligence in medicine*, 2008.

[24] Wright R Gold DR. Population disparities in asthma. *Annu Rev Public Health*, 2005.

[25] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[26] Tamai H, Ohsako N, and Takeno K et al. Changes in thyroid function in euthyroid subjects with a family history of graves disease: a follow-up study of 69 patients. *J Clin Endocrinol Metab*, 1980.

[27] Huidong Jin, Jie Chen, Hongxing He, Graham J. Williams, Chris Kelman, and Christine M. O'Keefe. Mining unexpected temporal associations: Applications in detecting adverse drug reactions. *IEEE Transactions on information tech. in biom.*, 12(4), 2008.

[28] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Mateo, 1995. Morgan Kaufmann.

[29] Igor Kononenko. Inductive and bayesian learning in medical diagnosis. *Applied Artificial Intelligence*, 1993.

[30] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in Medicine*, 23:89–109, 2001.

[31] Mintz M. Asthma update: Part i. diagnosis, monitoring, and prevention of disease progression. *Am Fam Physician*, 2004.

[32] Stampfer MJ Colditz GA Willett WC Krolewski AS et al Manson JE, Rimm EB. Physical activity and incidence of non-insulin-dependent diabetes mellitus in women. *Lancet*, 1991.

[33] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In Lyle Ungar, Mark Craven, Dimitrios Gunopulos, and Tina Eliassi-Rad, editors, *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA, August 2006. ACM.

[34] Yoon-Joo Park, Byung-Chun Kim, and Se-Hak Chun. New knowledge extraction technique using probability for case-based reasoning: application to medical diagnosis. *Expert Systems*, 2006.

[35] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[36] Mare Remm and Kalle Remm. Case-based estimation of the risk of enterobiasis. *Artificial Intelligence in Medicine*, 2008.

[37] Inzucchi SE. Oral antihyperglycemic therapy for type 2 diabetes. *JAMA*, 2002.

[38] John Shawe-Taylor and Nello Cristianini. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[39] L.M. Taft, R.S. Evans, C.R. Shyu, M.J. Egger, N. Chawla, J.A. Mitchell, S.N. Thornton, B. Bray, and M. Varner. Countering imbalanced datasets to improve adverse drug event predictive models in labor and delivery. *Journal of Biomedical Informatics*, In Press, Corrected Proof:–, 2008.

[40] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Addison-Wesley, 2006.

[41] Chieh-Yuan Tsai and Chuang-Cheng Chiu. A case-based reasoning system for pcb principal process parameter identification. *Expert Systems with Applications*, 32(4):1183 – 1193, 2007.

[42] McSharry C Hart CL et al. Upton MN, McConnachie A. Intergenerational 20 year trends in the prevalence of asthma and hay fever in adults: the midspan family study surveys of parents and offspring. *BMJ*, 2000.

[43] Gary M. Weiss and Haym Hirsh. Learning to predict rare events in event sequences. In *Proceedings of the 4 International Conference on Knowledge Discovery and Data Mining*.

[44] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd edition, 2005.

[45] G.P. Zhang. Neural networks for classification: a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 30(4):451–462, Nov 2000.

Appendix A

Experimental settings

The experimental settings which controls the behaviour of the system are given to the system as a configuration-file. This file should set the variables presented in Table A.1. The syntax of the configuration file must follow the rules below.

1. One variable per line
2. Each variable must be stated: <Name>#<Value>
3. List variables must be stated: <Name>#<Value> [, <Value>]
4. The right hand part may be empty, but it must contain one or more space-charachters
5. It is possible to add a comment after a variable, then it must be stated after a %-symbol, like this; <Name>#<Value>%<Comment>
6. The order of the variables are irrelevant
7. All variables in Table A.1 must be included
8. Spaces are allowed between separators like # and %.

One example: IN_ENCODING#ASCII %OK: UTF-8, Unicode, ASCII IKKE OK: ANSI.

Name	Type	Description
CLASSES_TO_PREDICT	boolean[]	Which classes should be predicted.
DATA_SOURCE	String[]	Names of data sets to be used.
DEBUG_MODE	boolean	Whether to display extra information to screen.
DEL_CENTRAL	boolean	Whether to delete the central object or not.
DEL_FVS	boolean	Whether to delete the FVset object or not.
ITERATION	int	Which iteration this is.
FVS_TO_CREATE	String[]	Which FV-types to create.
IN_ENCODING	String	The encoding of the data which is read in.
MIN_NUM_EVENTS	int	The minimum number of events a FV must contain.
MAX_NOF_EVENTS_TO_READ	int	The maximum number of events to read from the data set. Can be empty.
NOF_INTERVALS_FREQ	int	
NUM_NEG	int[]	The number of negatives per positive example.
NUM_ATTS	int[]	The number of attributes to use.
OUTPUT_FREQUENCY	int	How often output should be written to screen.
OUTPUT_PH_GRAP	boolean	If true, a PH representation of randomly selected PHs is output.
PERIOD_SIZE	int	The granlurity of the PH representation in days.
PIPE_CORR_BM	boolean	Must be true in order to control NUM_ATTS.
PREDICTION_PERIOD	int[]	The size of PP in days.
QUARANTINE_PERIOD	int[]	The size of QP in days.
REQUIRE_PATHOLOGICAL	boolean	Require that the test result was abnormal.
SUS_IND_AST	String[]	Suspicion-indicators for asthma.
SUS_IND_DIA	String[]	Suspicion-indicators for diabetes.
SUS_IND_HYP	String[]	Suspicion-indicators for hypothyroidism.
USE_ALL_EXAMPLES	boolean	Include all examples in FV. Overrides NUM_NEG.
USE_SMOTE	boolean	Use SMOTE or not.
WARNING_TIME	int[]	The size of WT in days.
WRITE_ARFF_BATCH	boolean	If true, all Instances are written in batch to arff-file, else they are written one by one.

Table A.1: The variables in the settings file and what they control.

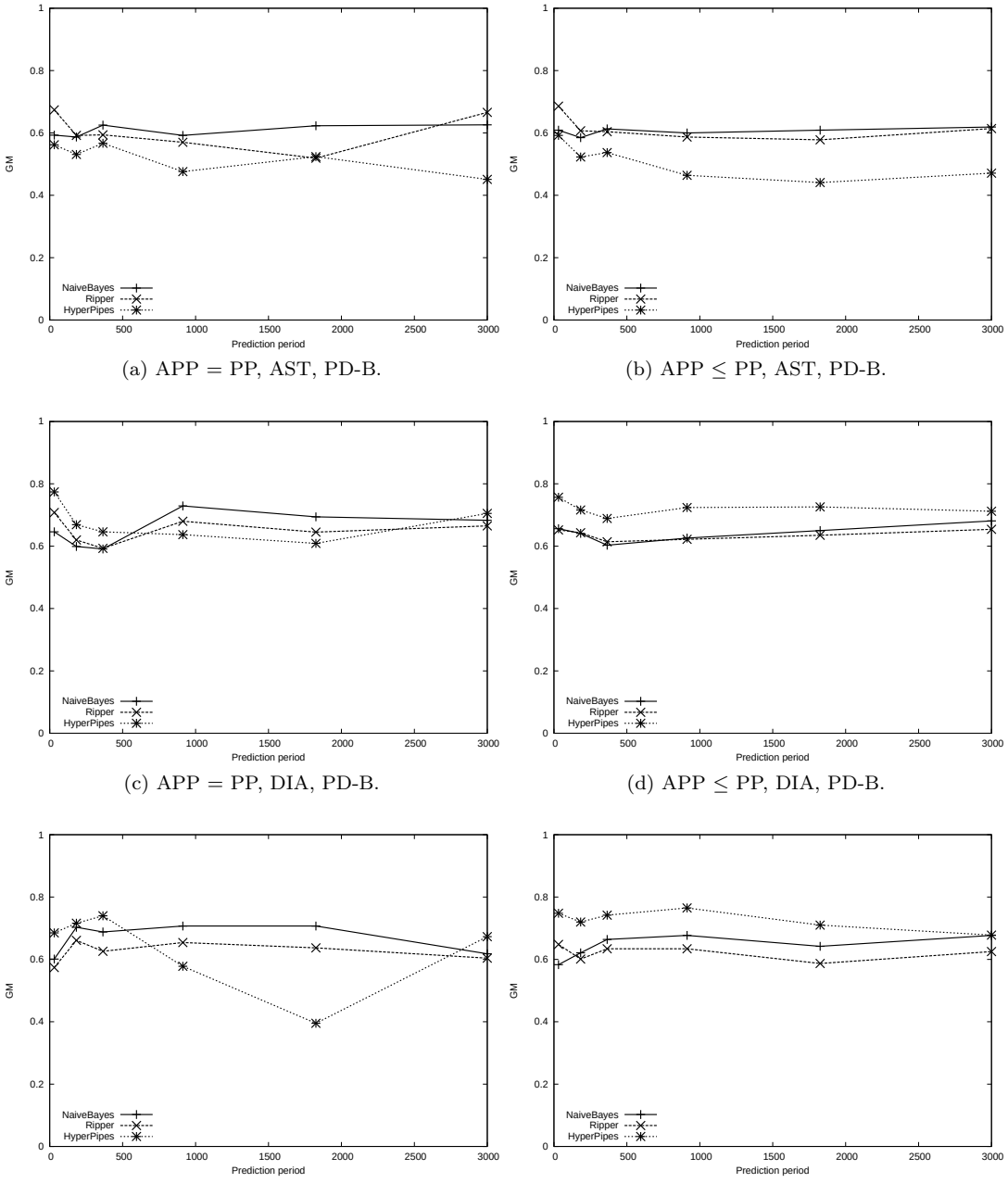
Appendix B

Additional results

This chapter contains additional results which were omitted in the main report, because the results were considered less important or took to much space in the report.

B.1 E5: Varying settings when APP = PP

The only difference between E2 and this experiment is that APP $\leq$ PP for E2, while APP = PP in E5. It is therefore interesting to compare the results of E5 with E2. We compare them by aligning figures from E5 in the first column, while the second column has the corresponding figures from E2.



(a) APP = PP, AST, PD-B.

(b) APP $\leq$ PP, AST, PD-B.

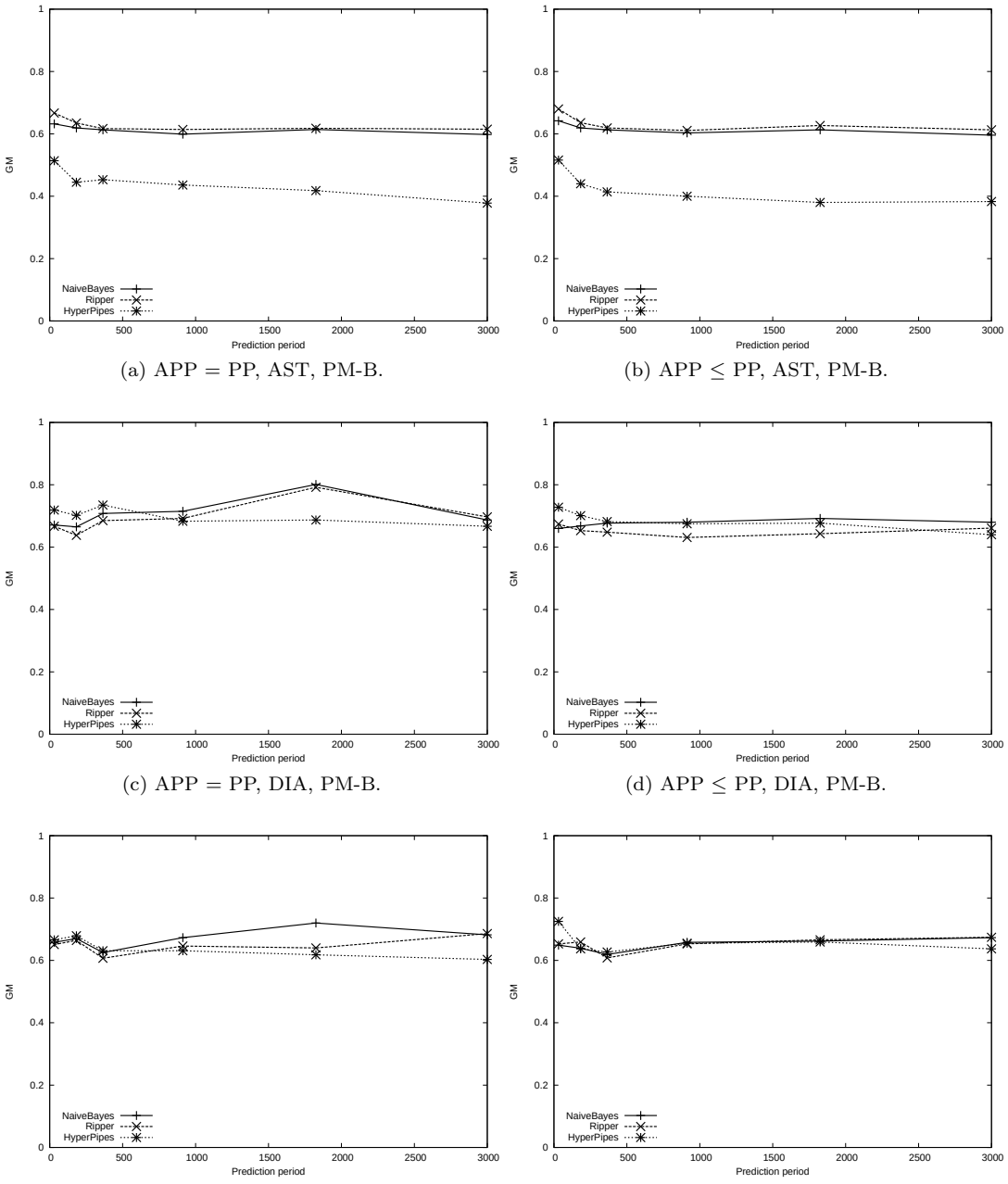
(c) APP = PP, DIA, PD-B.

(d) APP $\leq$ PP, DIA, PD-B.

(e) APP = PP, HYP, PD-B.

(f) APP $\leq$ PP, HYP, PD-B.

Figure B.1: Comparing the results when APP = PP with APP $\leq$ PP. PD-B, Iteration 3, PP varies and evaluation metric is geometric mean.



(a) APP = PP, AST, PM-B.

(b) APP ≤ PP, AST, PM-B.

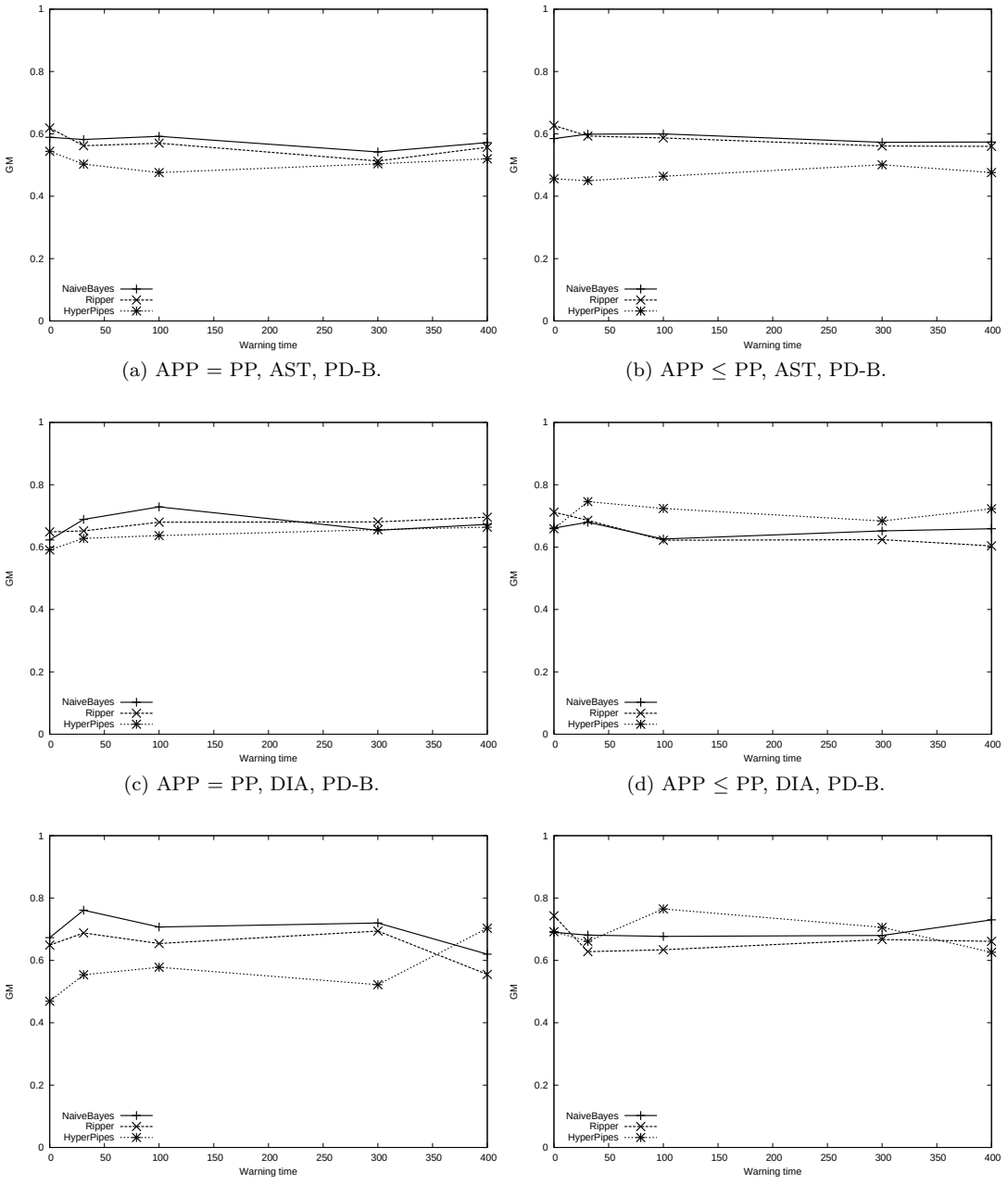
(c) APP = PP, DIA, PM-B.

(d) APP ≤ PP, DIA, PM-B.

(e) APP = PP, HYP, PM-B.

(f) APP ≤ PP, HYP, PM-B.

Figure B.2: Comparing the results when APP = PP with APP ≤ PP. PD-B, Iteration 3, PP varies and evaluation metric is geometric mean.



(a) APP = PP, AST, PD-B.

(b) APP ≤ PP, AST, PD-B.

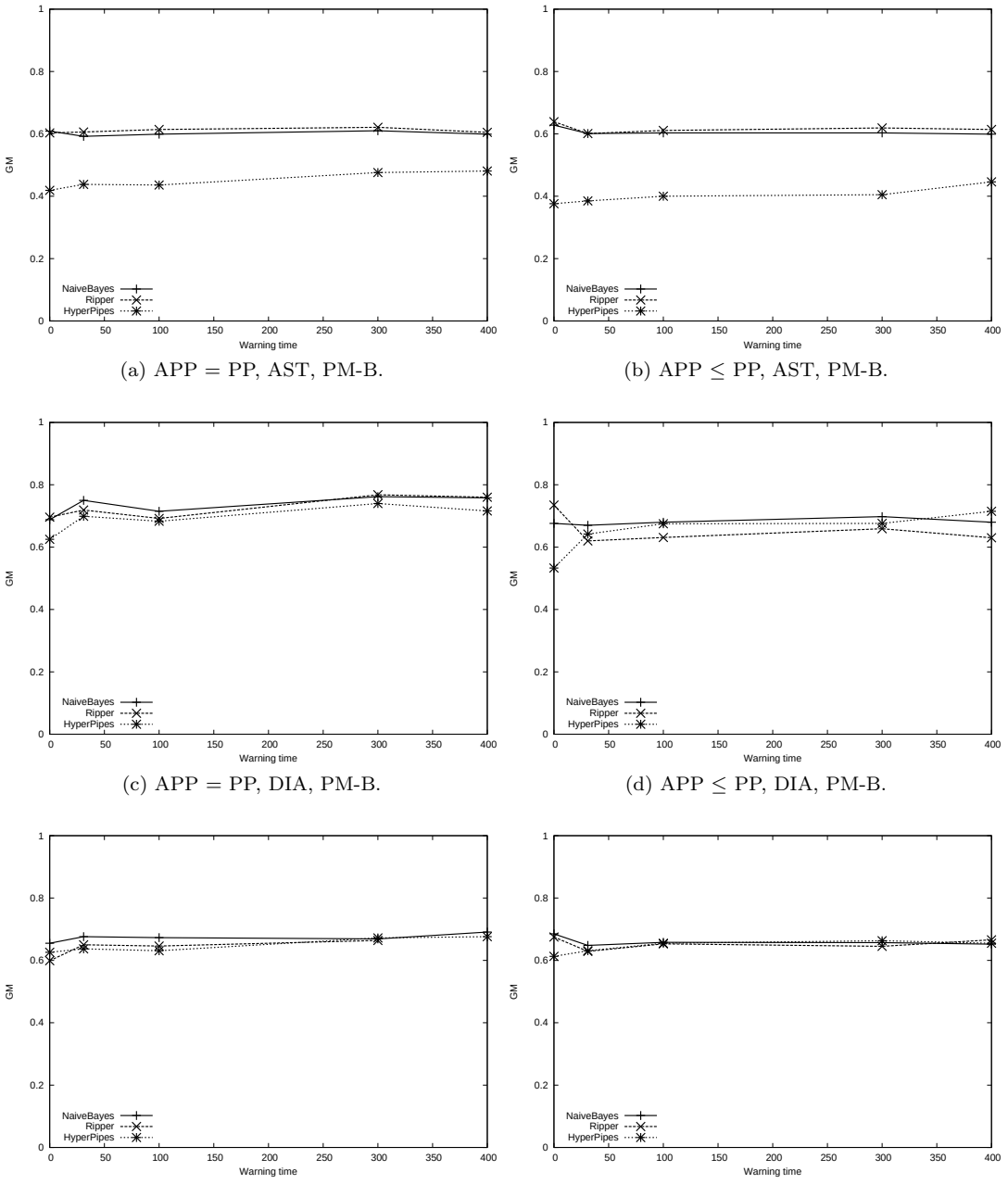
(c) APP = PP, DIA, PD-B.

(d) APP ≤ PP, DIA, PD-B.

(e) APP = PP, HYP, PD-B.

(f) APP ≤ PP, HYP, PD-B.

Figure B.3: Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the WT which varies in each plot.



(a) APP = PP, AST, PM-B.

(b) APP $\leq$ PP, AST, PM-B.

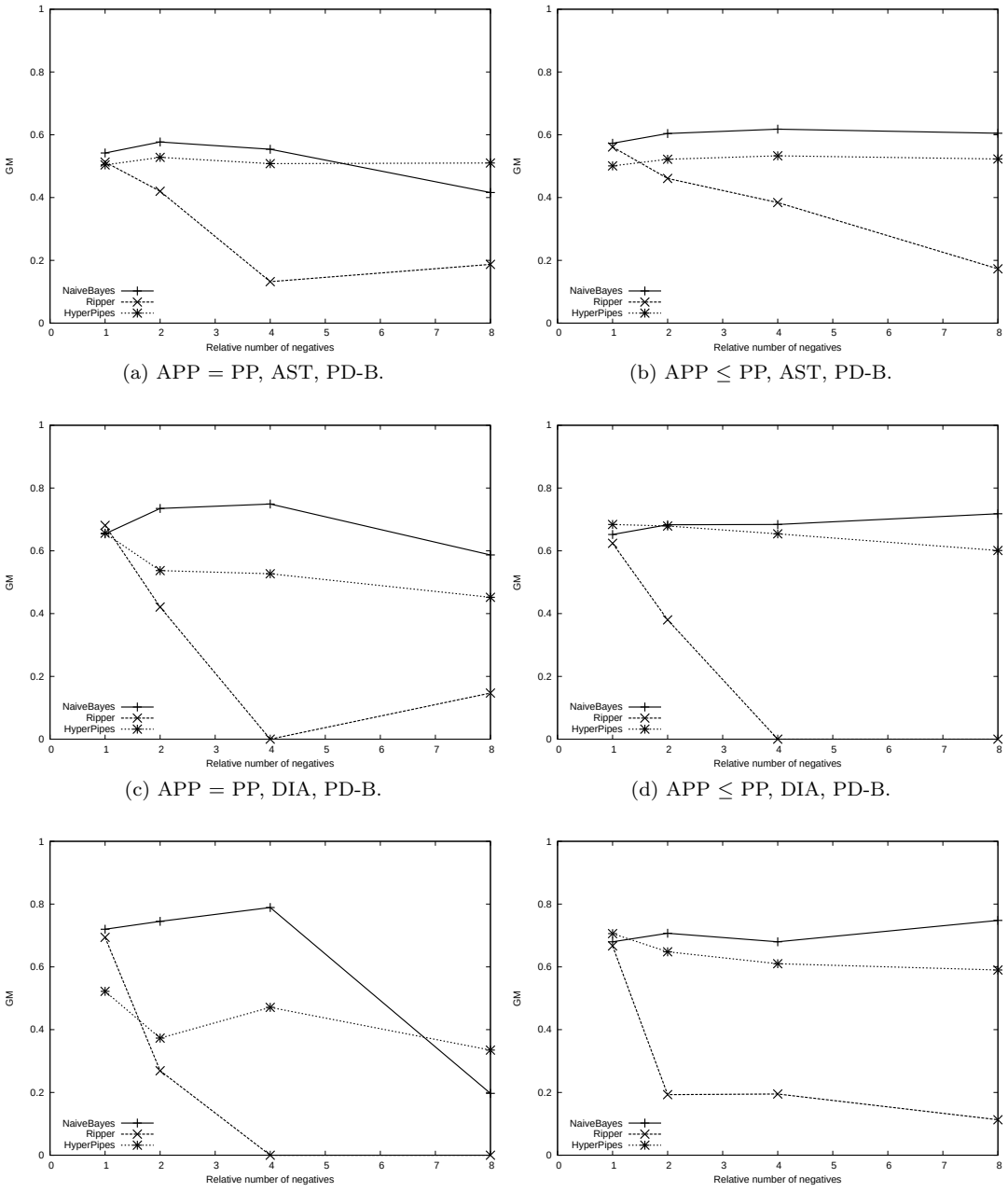
(c) APP = PP, DIA, PM-B.

(d) APP $\leq$ PP, DIA, PM-B.

(e) APP = PP, HYP, PM-B.

(f) APP $\leq$ PP, HYP, PM-B.

Figure B.4: Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the WT which varies in each plot.



(a) APP = PP, AST, PD-B.

(b) APP $\leq$ PP, AST, PD-B.

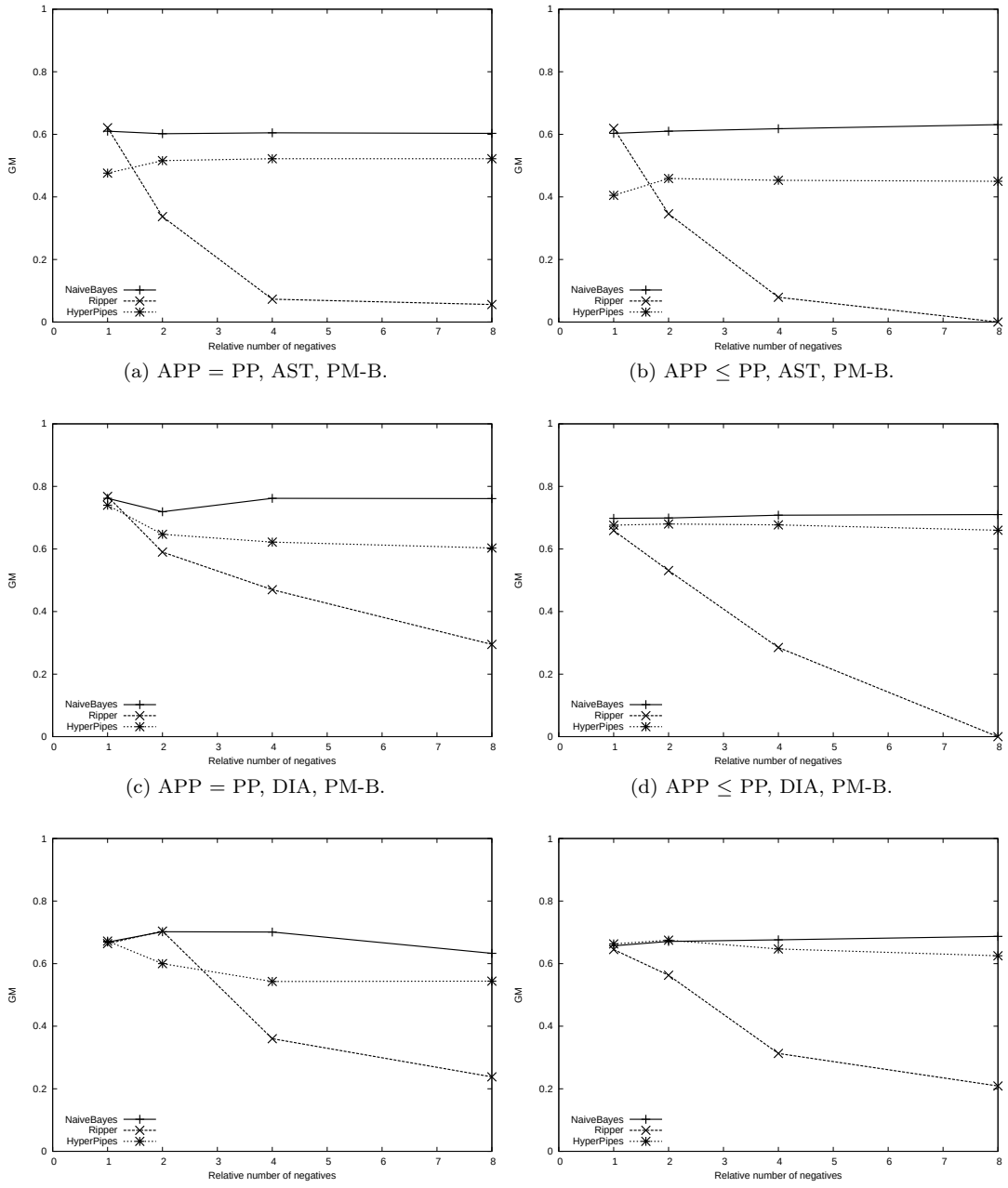
(c) APP = PP, DIA, PD-B.

(d) APP $\leq$ PP, DIA, PD-B.

(e) APP = PP, HYP, PD-B.

(f) APP $\leq$ PP, HYP, PD-B.

Figure B.5: Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NN which varies in each plot.



(a) APP = PP, AST, PM-B.

(b) APP $\leq$ PP, AST, PM-B.

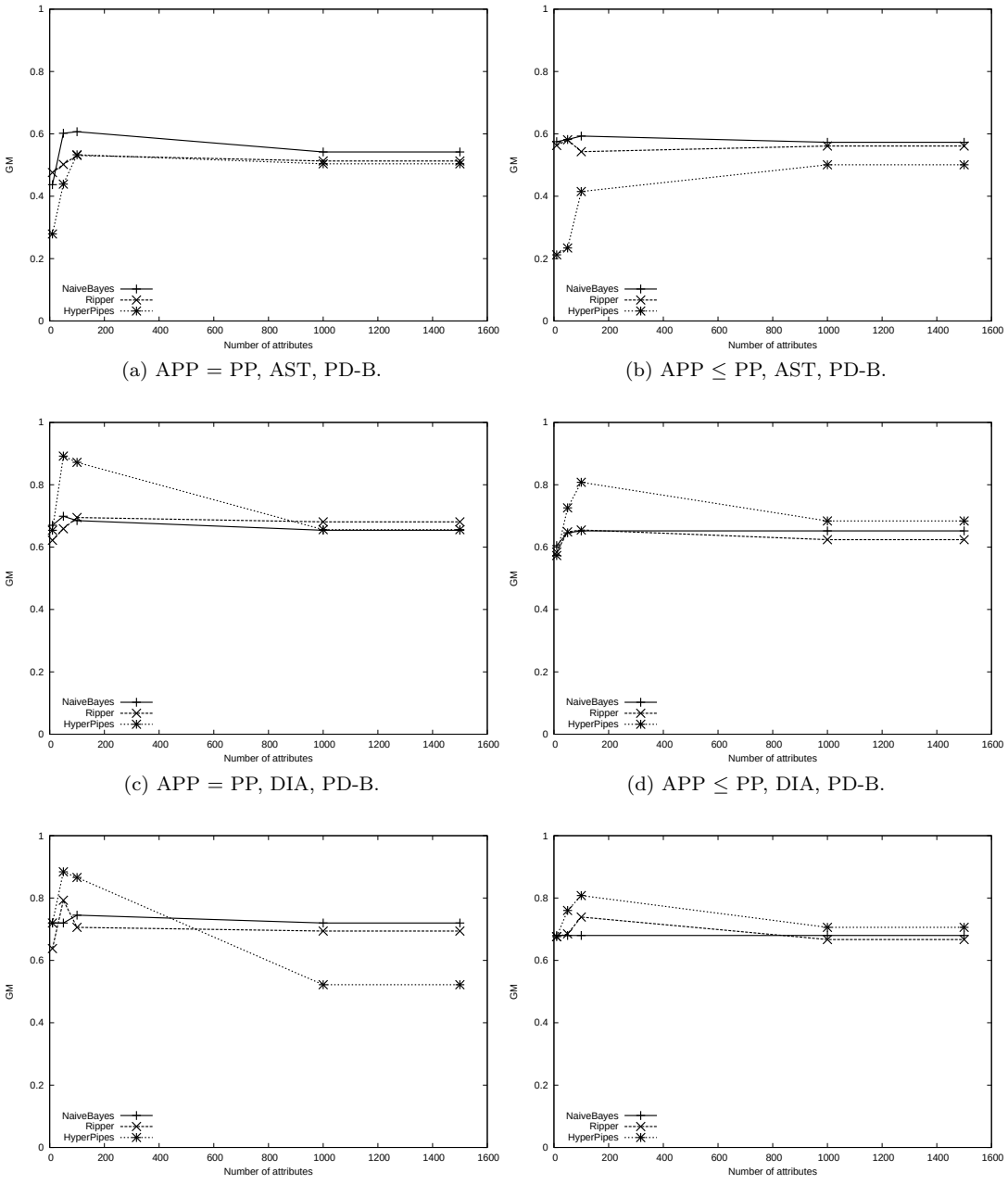
(c) APP = PP, DIA, PM-B.

(d) APP $\leq$ PP, DIA, PM-B.

(e) APP = PP, HYP, PM-B.

(f) APP $\leq$ PP, HYP, PM-B.

Figure B.6: Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NN which varies in each plot.



(a) APP = PP, AST, PD-B.

(b) APP $\leq$ PP, AST, PD-B.

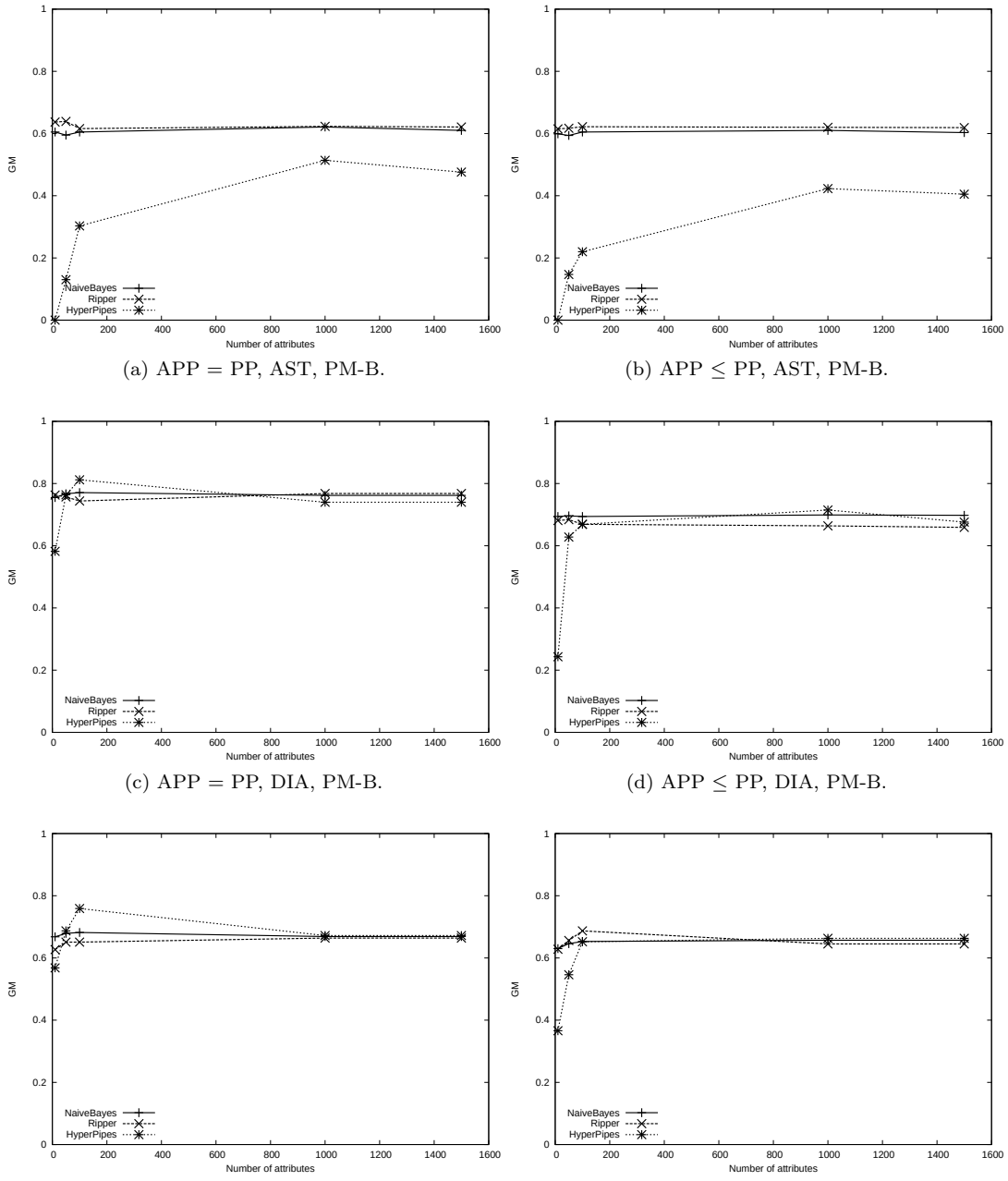
(c) APP = PP, DIA, PD-B.

(d) APP $\leq$ PP, DIA, PD-B.

(e) APP = PP, HYP, PD-B.

(f) APP $\leq$ PP, HYP, PD-B.

Figure B.7: Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NA which varies in each plot.



(a) APP = PP, AST, PM-B.

(b) APP $\leq$ PP, AST, PM-B.

(c) APP = PP, DIA, PM-B.

(d) APP $\leq$ PP, DIA, PM-B.

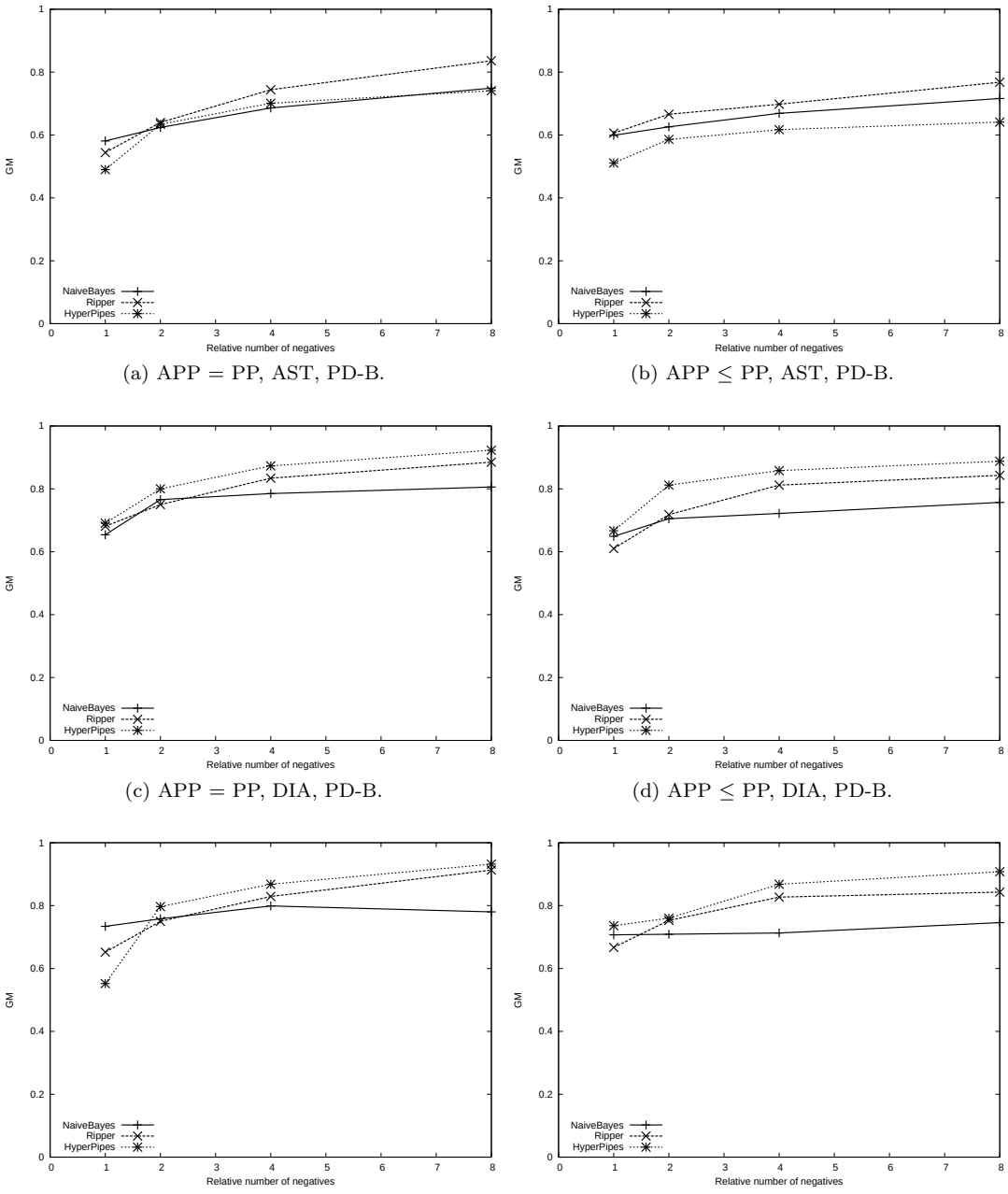
(e) APP = PP, HYP, PM-B.

(f) APP $\leq$ PP, HYP, PM-B.

Figure B.8: Performance of classifiers for iteration 3. Classifier is evaluated using geometric mean, and it is the NA which varies in each plot.

B.2 E6: Increasing positives when APP = PP

This section provides extra figures which compares the performance of E3 and E6. APP $\leq$ PP in E3, while E6 required APP = PP.



(a) APP = PP, AST, PD-B.

(b) APP ≤ PP, AST, PD-B.

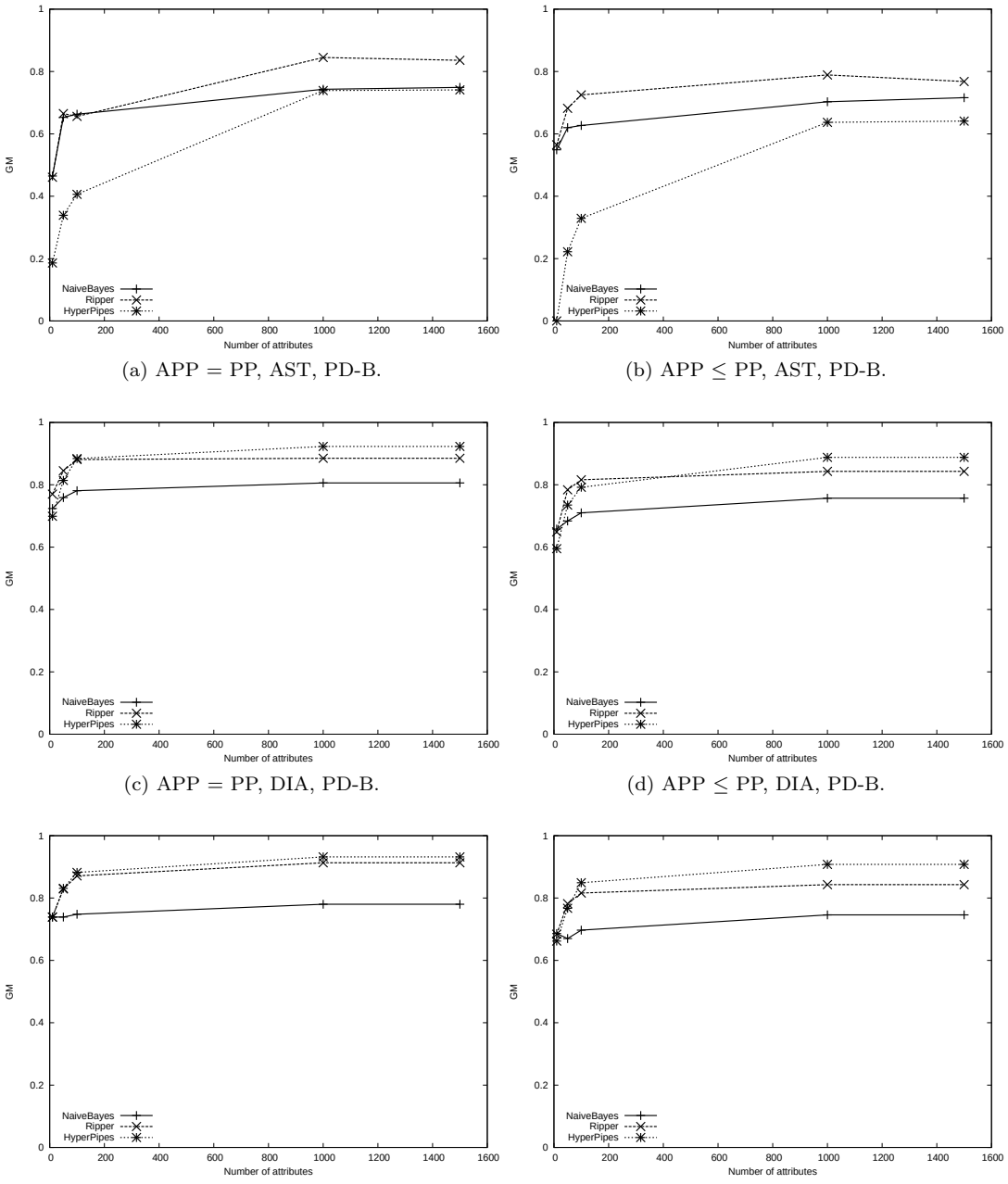
(c) APP = PP, DIA, PD-B.

(d) APP ≤ PP, DIA, PD-B.

(e) APP = PP, HYP, PD-B.

(f) APP ≤ PP, HYP, PD-B.

Figure B.9: Comparing the results when APP = PP (E6) with APP ≤ PP (E3) when using SMOTE. Evaluation metric: geometric mean, NA = All, WT = 300, PP = 913, NN varies.



(a) APP = PP, AST, PD-B.

(b) APP ≤ PP, AST, PD-B.

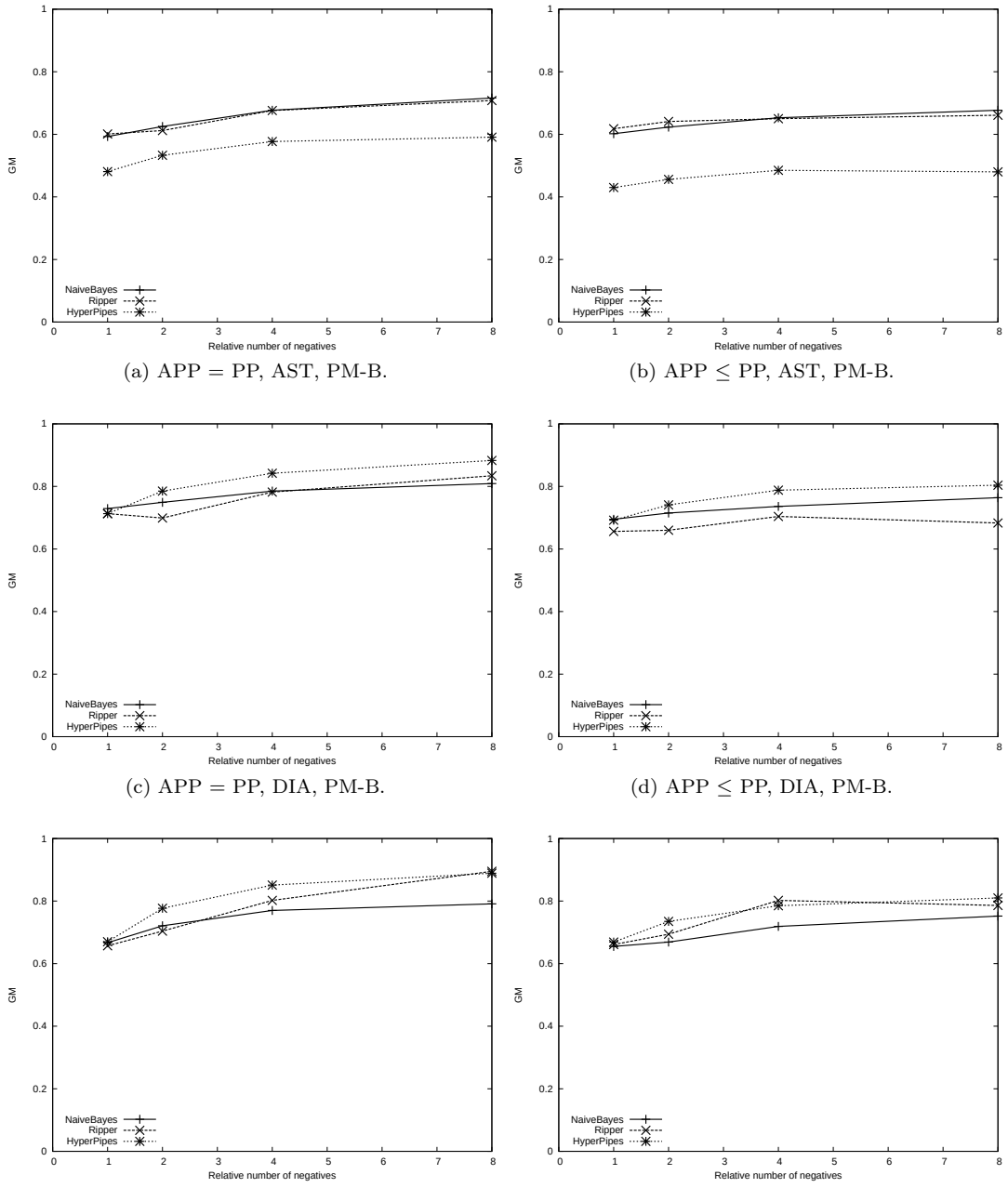
(c) APP = PP, DIA, PD-B.

(d) APP ≤ PP, DIA, PD-B.

(e) APP = PP, HYP, PD-B.

(f) APP ≤ PP, HYP, PD-B.

Figure B.10: Comparing the results when APP = PP (E6) with APP ≤ PP (E3) when using SMOTE. Evaluation metric: geometric mean, NA varies, WT = 300, PP = 913, NN is 8.



(a) APP = PP, AST, PM-B.

(b) APP $\leq$ PP, AST, PM-B.

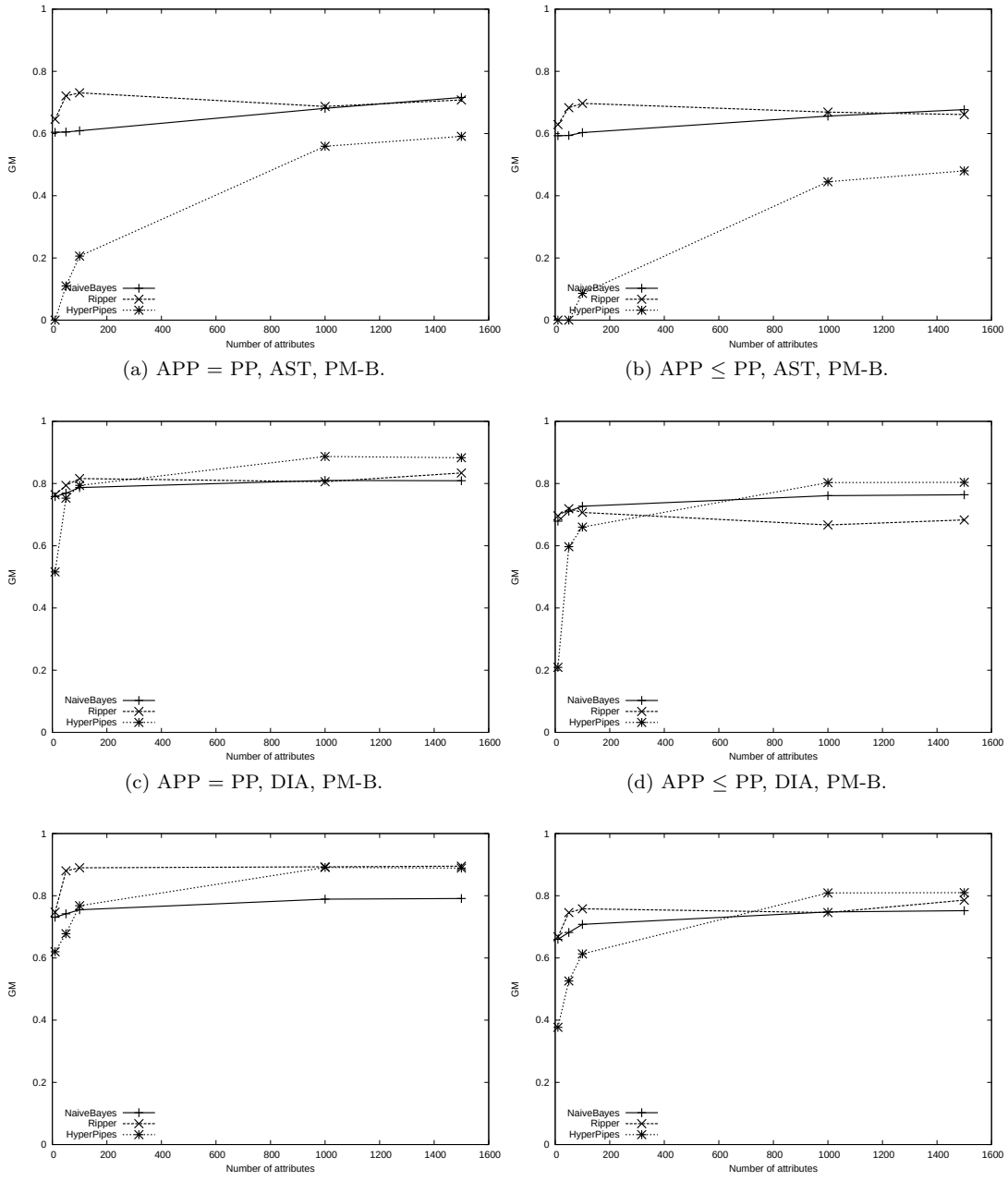
(c) APP = PP, DIA, PM-B.

(d) APP $\leq$ PP, DIA, PM-B.

(e) APP = PP, HYP, PM-B.

(f) APP $\leq$ PP, HYP, PM-B.

Figure B.11: Comparing the results when APP = PP (E6) with APP $\leq$ PP (E3) when using SMOTE. Evaluation metric: geometric mean, NA is max, NN varies.



(a) APP = PP, AST, PM-B.

(b) APP $\leq$ PP, AST, PM-B.

(c) APP = PP, DIA, PM-B.

(d) APP $\leq$ PP, DIA, PM-B.

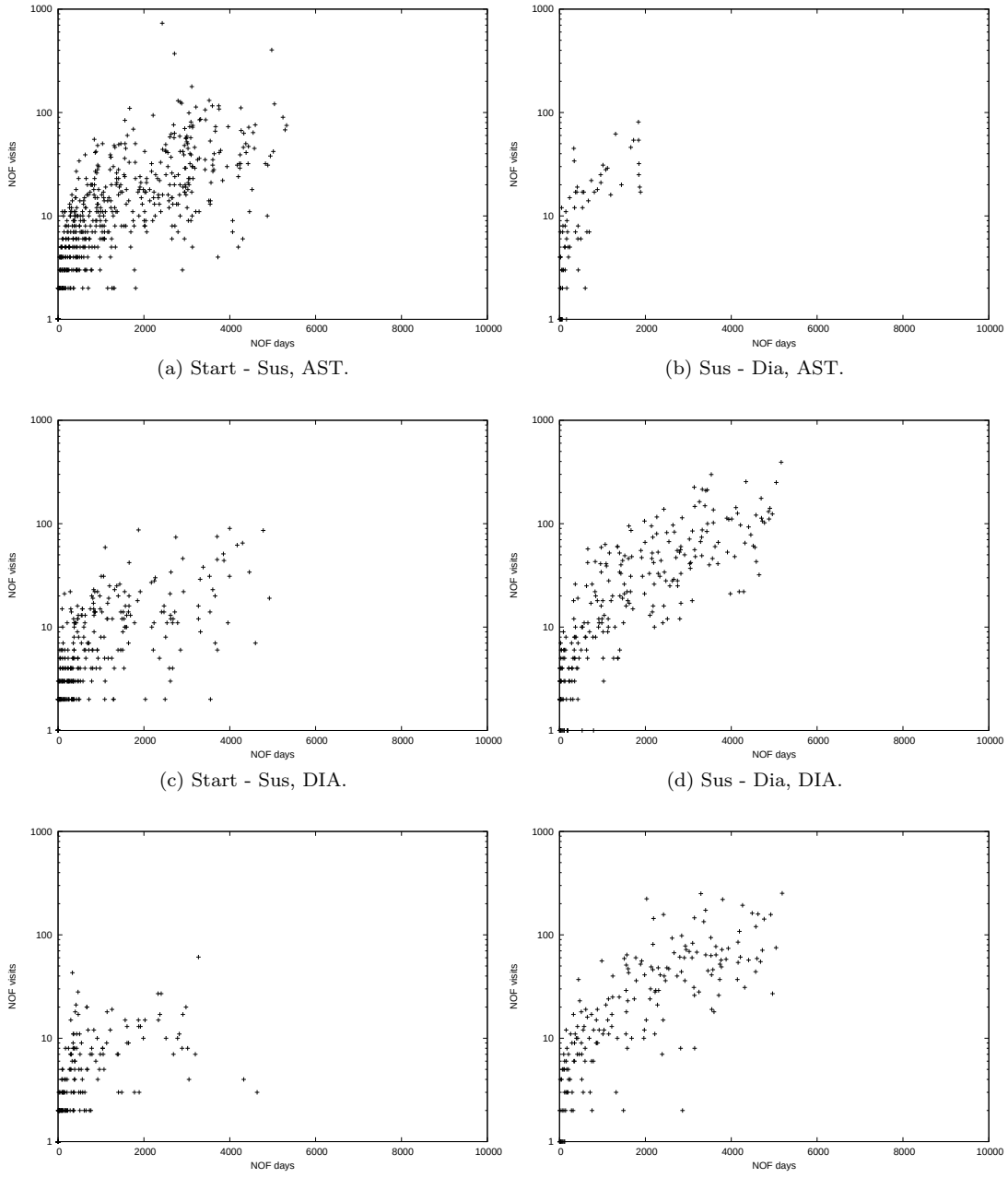
(e) APP = PP, HYP, PM-B.

(f) APP $\leq$ PP, HYP, PM-B.

Figure B.12: Comparing the results when APP = PP (E6) with APP $\leq$ PP (E3) when using SMOTE. Evaluation metric: geometric mean, NA varies, NN is 8.

B.3 E1: Additional scatter plots

This section has the scatter plots for iteration 2 in E1.



(a) Start - Sus, AST.

(b) Sus - Dia, AST.

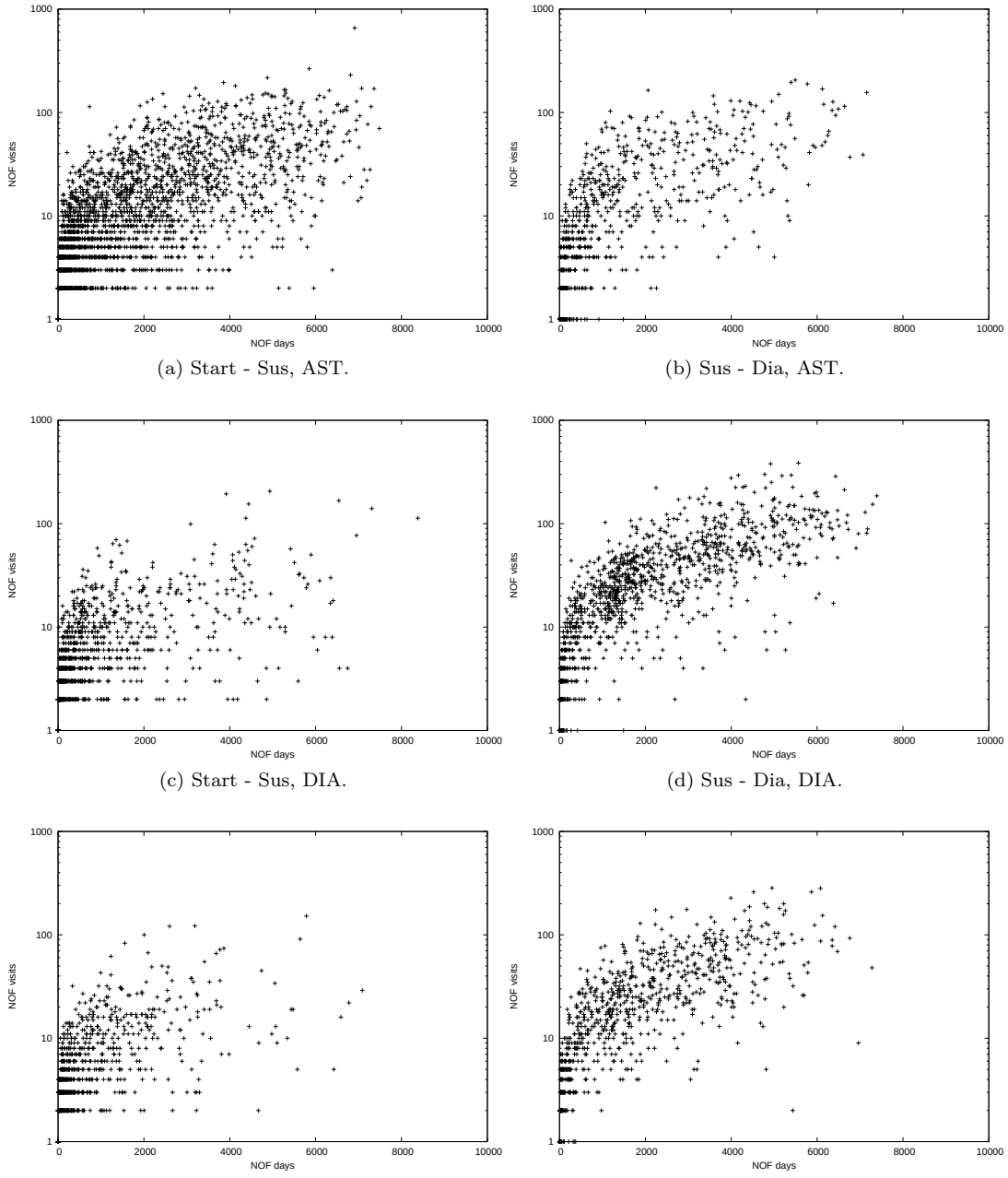
(c) Start - Sus, DIA.

(d) Sus - Dia, DIA.

(e) Start - Sus, HYP.

(f) Sus - Dia, HYP.

Figure B.13: Scatter plots of how quickly GP suspects and diagnoses patients, using data from PD-B, iteration 2.



(a) Start - Sus, AST.

(b) Sus - Dia, AST.

(c) Start - Sus, DIA.

(d) Sus - Dia, DIA.

(e) Start - Sus, HYP.

(f) Sus - Dia, HYP.

Figure B.14: Scatter plots of how quickly GP suspects and diagnoses patients, using data from PM-B, iteration 2.

Appendix C

Abbreviations

APP, Actual Prediction Period
AST, Asthma
ATC, Anatomical Therapeutic Chemical, A WHO drug classification system
C45, Tree learning algorithm
CWA, Mean Class-Weighted Accuracy, see section 2.4.3
DIA, Diabetes 2
DAS, Disease Analyser System, Name of the system which is used to conduct the experiments in this project
FV, Feature Vector
GM, Geometric Mean, see section 2.4.3
GP, General Practitioner
HYP, Hypothyroidism
ICPC , International Classification of Primary Care, C classification method for primary care encounter classification
MPX, Main problem X
NA, Number of attributes
NN, Number of negatives
NPV, Negative Prediction Value, see section 2.4.3
PH, Patient History
PP, Prediction Period
PPV, Positive Predictive Value, see section 2.4.3
QP, Quarantine Period
RIPPER, Repeated Incremental Pruning to Produce Error Reduction, see section 2.4.2
SMOTE, Synthetic Minority Over Sampling Technique
SVM, Support Vector Machine, see section 2.4.2
t^*_{sus}, Estimate of t_{sus}
t_{sus}, Time of suspicion
t_{TD}, Time when target diagnosis is set
ToSus*see* t_{sus}
ToTD*see* t_{TD}
WT, Warning Time