EXAMENSARBETE INOM TEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, *SVERIGE 2017*

The Sparse Data Problem Within Classification Algorithms

The Effect of Sparse Data on the Naïve Bayes Algorithm

OSCAR WÄRNLING

JOHAN BISSMARK

KTH
SKOLAN FÖR DATAVETENSKAP OCH KOMMUNIKATION

KTH Datavetenskap
och kommunikation

The Sparse Data Problem Within Classification Algorithms

The Effect of Sparse Data on the Naïve Bayes Algorithm

Oscar Wärnling Johan Bissmark

June 5, 2017

Bachelor's Thesis in Computer Science at CSC, DD142x

Supervisor: Kevin Smith
Examiner: Örjan Ekeberg

Abstract

In today's society, software and apps based on machine learning and predictive analysis are of the essence. Machine learning has provided us with the possibility of predicting likely future outcomes based on previously collected data in order to save time and resources.

A common problem in machine learning is sparse data, which alters the performance of machine learning algorithms and their ability to calculate accurate predictions. Data is considered sparse when certain expected values in a dataset are missing, which is a common phenomenon in general large scaled data analysis.

This report will mainly focus on the Naïve Bayes classification algorithm and how it is affected by sparse data in comparison to other widely used classification algorithms. The significance of the performance loss associated with sparse data is studied and analyzed, in order to measure the effect sparsity has on the ability to compute accurate predictions.

In conclusion, the results of this report lay a solid argument for the conclusion that the Naïve Bayes algorithm is far less affected by sparse data compared to other common classification algorithms. A conclusion that is in line with what previous research suggests.

Sammanfattning

I dagens samhälle är maskininlärningsbaserade applikationer och mjukvara, tillsammans med förutsägelser, högst aktuellt. Maskininlärning har gett oss möjligheten att förutsäga troliga utfall baserat på tidigare insamlad data och därigenom spara tid och resurser.

Ett vanligt förekommande problem inom maskininlärning är gles data, eftersom det påverkar prestationen hos algoritmer för maskininlärning och deras förmåga att kunna beräkna precisa förutsägelser. Data anses vara gles när vissa förväntade värden i ett dataset saknas, vilket generellt är vanligt förekommande i storskaliga dataset.

I den här rapporten ligger fokus huvudsakligen på klassificeringsalgoritmen Naïve Bayes och hur den påverkas av gles data jämfört med andra frekvent använda klassifikationsalgoritmer. Omfattningen av prestationssänkningen som resultat av gles data studeras och analyseras för att mäta hur stor effekt gles data har på förmågan att kunna beräkna precisa förutsägelser.

Avslutningsvis lägger resultaten i den här rapporten grund för slutsatsen att algoritmen Naïve Bayes påverkas mindre av gles data jämfört med andra vanligt förekommande klassificeringsalgoritmer. Den här rapportens slutsats stöds även av vad tidigare forskning har visat.

Contents

1. Introduction

Over the past few years society has become increasingly dependent on services based on machine learning and different machine learning techniques. There is a considerable amount of applications for machine learning, with the most significant being in the field of data mining. As people are highly susceptible to mistakes in data analysis, machine learning is used as an alternative and more accurate way of solving data mining problems [1].

On a deeper level, different data mining techniques are used for discovering patterns in data, in order to attain accurate predictions. Classification is a common approach used for tackling such tasks, and is widely used within the field of supervised machine learning to produce predictive calculations and solve supervised learning problems [1].

A classifier can be used for predictions in a broad range of different areas, such as classification of flowers, determining if an individual is likely to develop a disease based on their medical history or if a user is likely to purchase a subscription based on demographic information. The main purpose of a classifier is to predict the class of an object, based on the attributes of objects of the same dataset. It can, therefore, be used on practically any type of data to distinguish and differentiate data to obtain solid predictions [18].

A general problem regarding public datasets is sparse data, which is a recognized challenge when working with machine learning within classifiers. Sparse data refers to data that is incomplete and can have an immense effect on the ability to train the classifier into producing accurate predictions [6][7].

In this thesis, the purpose is to test the performance of the Naïve Bayes classifier for datasets of different sparsity levels. For comparative reasons, the same tests are conducted with two other types of algorithms, namely J48 and SVM. Naïve Bayes classifiers use an algorithm based on a mathematical theorem of probability formulated by Thomas Bayes in the 18th century [2]. The algorithm is appealing as it is simple to implement, easy to interpret, generally effective when handling datasets with missing data and considered to provide competitive predictions [7].

1.1 Problem Definition

We will investigate the Naïve Bayes algorithm, used for predictive modelling. Research will be conducted regarding the performance of the algorithm when it encounters sparse data rather than datasets containing dense data. Investigations will be performed about whether sparse data has a negative effect on the predictive accuracy of the implemented algorithm in question compared to other popular classification algorithms, hence the question:

What is the performance loss associated with sparse data when implementing Naïve Bayes for classification compared to other common classification algorithms?

1.2 Scope and Constraints

This thesis is delimited to the analysis of how the predictive accuracy of the Naïve Bayes algorithm is affected by different levels of missing data. The primary focus is on the sparse data problem and how the algorithm of choice handles different sparsity levels. Therefore, the main focus will be on the Naïve Bayes algorithm. However, several algorithms are tested to allow for comparisons between Naïve Bayes and the other alternative classification algorithms. In order to avoid variances in performance associated with a certain dataset or types of data, tests are conducted on several datasets to ensure the accuracy of the test results.

1.3 Thesis Overview

Section 2 of this report consists of an extensive background study of classifiers, the algorithms of choice as well as the datasets and the potential obstacles that might occur when handling large amounts of data. Section 3 addresses the course of action and more specifically the adopted methodology for resolving the problem statement. In the fourth section, the results of the testing and research work is covered. To conclude, the fifth and last section includes the analysis of the results produced in Section 4 as well as a thorough conclusion about the subject.

2. Background

This section introduces the concept of classification and classification of sparse data, as well as a more extensive background analysis of the Naïve Bayes classifier. Introductions to the J48 and SVM classifiers follow, preceding a summary of relevant data problems that classification models encounter. Lastly, an introduction of the several datasets used to benchmark the algorithm concludes the background study.

2.1 Classification

Classification is a machine learning and data mining task, used to assign a class to specific data, in order to distinguish and differentiate items within a dataset. It is used for predictive modeling where the aim is to predict a categorical target, thus, a classification model is trained to be able to classify which pre-defined category new observations belong to. The primary goal is to achieve highly accurate predictions [13].

Classification is common for solving a large amount of different predictive and analytical problems. More concretely, it is extensively used within areas, such as text categorization, fraud detection, natural-language processing, market segmentation and recommender systems [18].

2.1.1 Machine Learning Techniques

For a predictive model to produce predictions, it needs to undergo some type of learning. Machine learning techniques can be categorized as supervised learning and unsupervised learning. The concept of the learning processes is for the algorithms to be trained on a set of fixed data to produce predictions on the outcome. Within supervised machine learning, all the data, including the pre-determined outcome is provided. The training process lets the algorithm find relations between the data and the corresponding outcomes of the dataset, making future predictions possible. Supervised learning is usually done through independent training and test data sets, used in classification and regression tasks [17].

Unsupervised learning, on the other hand, is in a way rather similar to the supervised learning method. A fixed train dataset is acquired, used to train the algorithm to make predictions on the test data. In contrast to supervised learning, unsupervised learning does not train the algorithm with the fixed outcomes. The actual outcomes are missing from the training data, which results in the unsupervised learning algorithms having to

find and discover patterns based solely on the input data. A common case being when engaging in clustering problems [17].

2.1.2 Data Limitations

Lacking data is a crucial problem when working with classification. The amount of data in a dataset relative to the amount of users is known as the density of the dataset. By collecting the data and inserting it into a matrix, one can identify how dense or sparse a dataset is. The more information there is relative to the amount of users, the denser the dataset. The density of the dataset used has a huge impact on the final prediction accuracy. A higher density generally results in more accurate predictions, since there is more data to learn and train from [5].

$$M_{Dense} = \begin{vmatrix} M_{11} & M_{21} & M_{31} & M_{41} & M_{51} \\ M_{12} & M_{22} & M_{32} & M_{42} & M_{42} \\ M_{13} & M_{23} & M_{33} & M_{43} & M_{53} \\ M_{14} & M_{24} & M_{34} & M_{44} & M_{54} \\ M_{15} & M_{25} & M_{35} & M_{45} & M_{55} \end{vmatrix} \quad M_{Sparse} = \begin{vmatrix} M_{11} & 0 & M_{31} & M_{41} & 0 \\ 0 & 0 & 0 & M_{42} & 0 \\ 0 & 0 & 0 & 0 & M_{53} \\ M_{14} & M_{24} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{55} \end{vmatrix}$$

Figure 1 - Matrix representation of a dense dataset. *Figure 2 - Matrix representation of a sparse dataset.*

With a larger amount of missing values, the dataset is considered sparser, making accurate predictions more difficult to acquire. Data sparsity is widely considered as a key cause for unsatisfactory classification accuracy. Sparsity can for example occur when a user forgets to enter certain explicit data when signing up to a website, leaving the matrix of collected data sparse. For public datasets, it is common to encounter datasets with sparsity levels of above 90% [6]. Some classification algorithms perform better than others when dealing with sparse datasets. Naïve Bayes classifiers are considered high performers when dealing with sparse data [7]. When encountering a missing value, the algorithm omits that property in its calculations, which is only applicable due to the assumption that all attributes are independent [9].

When collecting data, missing data can occur in different ways. For our research, the focus is on data that is missing completely at random (MCAR), which implies that data is missing independently of observed and unobserved data. There is in this case no relation between the missingness of the data and variable values in the dataset. The contrary being missing not at random (MNAR), where there is an existing relation

between the missing values and the current variable values of the dataset. The values are evidently not randomly missing in this case [15].

2.1.3 Handling Missing Values

Most classification algorithms have been developed for handling data without any missing attributes and, therefore, require completely dense datasets in order to compute predictions. Since sparse datasets are very common and most algorithms cannot handle missing values natively, techniques for solving the missing data issues are widely used. The two most common approaches for handling sparsity are imputation or deletion. These two pre-processing methods are frequently used to eliminate the dataset sparsity when using algorithms without an internal way of handling the missing values. Not all techniques are presented in this report and alterations to the techniques are common. On the other hand, certain algorithms actually have a native way of handling sparsity and do not require any type of pre-processing of data in order to work [8].

2.1.3.1 Mean/Mode Imputation

Mean imputation is the process of calculating the mean value of all the values of the attribute, allowing it to replace the missing value. If the attribute is not numerical, mode imputation is used, which is letting the most frequent value of the attribute replace the missing value. Estimating missing values through imputation is considered viable when handling small amounts of sparse data. By using imputation on data of low sparsity, the likelihood of large deviation in results is less likely. Mean/mode imputation is the most commonly used method for handling sparse data [23].

2.1.3.2 Litwise Deletion

The litwise deletion method involves removing or ignoring entire observations from the dataset if the observation in question contains a missing attribute value. The obvious drawback of this method is, however, that potentially viable and important data may be excluded in the calculations and may severely bias the results [23].

2.1.3.3 Internal sparsity handling

As aforementioned, most machine learning algorithms require a completely dense dataset without missing values in order to work. There are, however, exceptions, such as in the case of the Naïve Bayes algorithm and some decision tree algorithms. Such classification algorithms are able to handle missing values natively, meaning they can compute predictions on sparse data without using any type of prior pre-processing. The

algorithmic calculations include the missing values, making the sparsity a part of the final predictions [8].

2.2 The Classification Algorithms

This section starts with an introduction to the Naïve Bayes classifier algorithm. Thereafter, brief introductions to the J48 and SVM classifiers follow

2.2.1 Naïve Bayes Classifier

The Naïve Bayes classifier bases its calculations on a theorem formulated by the Englishman Thomas Bayes (1702–1761) [2]. The theorem provides the following formula:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{P(A) \cdot P(B|A)}{P(B)}, \qquad (1)$$

where $P(A|B)$ is the probability of A, given B, and $P(A)$ and $P(B)$ is the probability of A and B respectively, independently of each other [3]. Naïve Bayes classifiers use this theorem by calculating the highest probability for the class of A, given an attribute B. Since the exact probability is irrelevant in the case of classifiers, dividing by a constant $P(B)$ is unnecessary. If there are several attributes, Naïve Bayes classifiers assume these attributes to be conditionally independent. The formula for calculating the likelihood then looks as follows:

$$P(B|A) = P(B_1|A) \cdot P(B_2|A) \cdot \ldots \cdot P(B_n|A), \qquad (2)$$

where n is the number of attributes. The classifiers calculate this for all the classes A and predict this object to belong to the class that generates the greatest value of $P(B|A)$ [4].

The Naïve Bayes classifier is generally considered to perform particularly well when encountering cases of missing data. The algorithm takes advantage of the assumption that the attributes are independent and ignores the missing attributes when calculating the probability, making use of other available feature values in order to make predictions [9]. For Equation (2), this means that the specific terms associated with the missing values are removed from the calculation. While more common types of missing value handling, such as imputation, works well for data with small amounts of missing values, the Naïve Bayesian sparse data handling method is considered a high performer as sparsity increases. In several previous studies, the Naïve Bayes algorithm

has been proven to perform exceptionally well when handling large amounts of missing values [16][22][25].

2.2.2 J48

The J48 classifier is an implementation of an algorithm called Iterative Dichotomiser 3 (ID3), which uses decision trees to make its predictions. The training on the dataset consists of several iterations of calculating which attribute currently provides the most valuable information about the object being classified. This is called the information gain. For every time the algorithm calculates the information gain, it creates a new node in the decision tree. In the testing phase, this tree is the basis of the predictions [11].

The J48 algorithm uses the C4.5 strategy for handling sparsity, which is common within decision tree learning algorithms. When the classifier encounters a missing value and, therefore, cannot go further down the decision tree, the probability distribution of the possible decisions is calculated to find the most frequent outcome. The J48 classifier follows the branch in the decision tree in order to produce its prediction for the observation containing the missing value [21].

2.2.3 SVM

When an SVM (Support Vector Machine) trains on a dataset, it plots the objects of the dataset in an n-dimensional space, where n is the number of attributes in the dataset. It then tries to find hyperplanes that divide the objects into their separate classes. Thereafter, it can make its predictions based on in which division the objects appear [12].

When handling sparse data, the SVM algorithm uses imputation and replaces all missing values by global means/modes [24].

2.3 The Datasets

The following subsections provide information about the datasets used in this research. For each dataset, there will be a short introduction, and for some datasets two tables follow the introduction: one table explaining the attributes of the dataset; and one table explaining the class.

2.3.1 Car Evaluation Dataset

For the Car Evaluation dataset, the goal is to predict the customer acceptability of a car, given information about the price, maintenance costs, number of doors, number of people the car can carry, size of the luggage boot and the safety of the car. The dataset contains data for 1,728 cars and has a density of 100 % [14].

LABEL	**TYPE**	**DESCRIPTION**
Buying	nominal	The price to buy the car; *vhigh*, *high*, *med* or *low*.
Maint	nominal	The costs for the maintenance; *vhigh*, *high*, *med* or *low*.
Doors	nominal	The number of doors of the car; *2*, *3*, *4* or *more..*
Persons	nominal	The number of persons the car can carry; *2*, *4* or *more*..
Lug_boot	nominal	The size of the luggage boot; *small*, *med* or *big*.
Safety	nominal	The estimated safety; *low*, *med* or *high*.

Table 1 - The attributes of the Car Evaluation dataset [14].

LABEL	**TYPE**	**DESCRIPTION**
class	nominal	The customer acceptability of the car. Can either be *unacc*, *acc*, *good* or *vgood*.

Table 2 - The class of the Car Evaluation dataset [14].

2.3.2 Pen-Based Recognition Dataset

The Pen-Based Recognition dataset contains 11,992 digits, written by 44 writers on a tablet. For each digit, there is a series of ratio type integers in the range of 0 to 100, indicating the input from the users. These integers represent where the pen was located on the tablet, with an interval of 100 milliseconds. By using this data, the classifiers aim to predict the specific digit produced by the writer [17].

2.3.3 Mushroom Dataset

Given information about the shape and properties of a mushroom, the aim is to predict whether it is edible or not. The dataset contains 8,124 observations on 23 different species of mushroom. More information about the attributes and classes is found in Table 3 and Table 4.

LABEL	TYPE	DESCRIPTION
cap-shape	nominal	The shape of the cap can either be *bell*, *conical*, *convex*, *flat*, *knobbed* or *smooth*.
cap-surface	nominal	The surface of the cap can either be *fibrous*, *grooves*, *scaly* or *smooth*.
cap-color	nominal	The color of the cap can either be *brown*, *buff*, *cinnamon*, *gray*, *green*, *pink*, *purple*, *red*, *white* or *yellow*.
bruises?	nominal	Indicates if the mushroom has bruises or not.
odor	nominal	The odor can either be *almond*, *anise*, *creosote*, *fishy*, *foul*, *musty*, *pungent*, *spicy* or *none*.
gill-attachment	nominal	The look of how the gills are attached is either *attached*, *descending*, *free* or *notched*.
gill-spacing	nominal	The spacing between the gills is either *close*, *crowded* or *distant*.
gill-size	nominal	The sizes of the gills are divided into *broad* and *narrow*.
gill-color	nominal	The color of the gill is either *black*, *brown*, *buff*, *chocolate*, *gray*, *green*, *orange*, *pink*, *purple*, *red*, *white* or *yellow*.
stalk-shape	nominal	The shape of the stalk is either *enlarging* or *tapering*.
stalk-root	nominal	The root is either *bulbous*, *club*, *cup*, *equal*, *rhizomorphs*, *rooted* or *missing*.

stalk-surface-above-ring	nominal	The stalk surface above the ring is either *fibrous*, *scaly*, *silky* or *smooth*.
stalk-surface-below-ring	nominal	The stalk surface below the ring is either *fibrous*, *scaly*, *silky* or *smooth*.
stalk-color-above-ring	nominal	The color of the stalk above the ring is either *brown*, *buff*, *cinnamon*, *gray*, *orange*, *pink*, *purple*, *red*, *white* or *yellow*.
stalk-color-below-ring	nominal	The color of the stalk below the ring is either *brown*, *buff*, *cinnamon*, *gray*, *orange*, *pink*, *purple*, *red*, *white* or *yellow*.
veil-type	nominal	The types of veil are *partial* or *universal*.
veil-color	nominal	The color of the veil is either *brown*, *orange*, *white* or *yellow*.
ring-number	nominal	The number of rings the mushroom can have is either have *one*, *two* or *none*.
ring-type	nominal	The type of the rings is either *cobwebby*, *evanescent*, *flaring*, *large*, *pendant*, *sheathing*, *zone* or *none*.
spore-print-color	nominal	The color of the spore print is either *black*, *brown*, *buff*, *chocolate*, *green*, *orange*, *purple*, *white* or *yellow*.
population	nominal	The population can be *abundant*, *clustered*, *numerous*, *scattered*, *several* or *solitary*.
habitat	nominal	Its habitat is either *grasses*, *leaves*, *meadows*, *paths*, *urban*, *waste* or *woods*.

Table 3 - The attributes of the Mushroom dataset [19].

LABEL	**TYPE**	**DESCRIPTION**
class	nominal	A mushroom is either *definitely edible*, *not recommended* or *definitely poisonous*. The two latter have been combined, giving the final

		classes of *edible* or *poisonous*.

Table 4 - The attributes of the Mushroom dataset [19].

2.3.4 Australian Credit Approval Statlog Dataset

The Australian Credit Approval Statlog dataset contains information about credit card applications. Due to confidentiality, very little information about the originally collected data is provided. What is known about the dataset is that it consists of six numerical attributes and eight nominal attributes. The dataset is used to predict whether the application is approved or not [20].

3. Method

This section describes the approach of the research in detail, beginning with the test procedure and how the results are calculated. Secondly, the data filtering algorithm used to generate sparsity is described. A short discussion regarding the algorithm implementations will then follow to conclude the method section.

3.1 Test Procedure

In this study, sparse data is tested for the Naïve Bayes algorithm. The algorithm is compared to two highly popular classification algorithms, J48 and SVM.

Performance tests for the three classifiers are conducted on datasets of different density levels, scaling from 100 %, 90 %, 80 %, 70 % etc. down to 10 %, to accurately measure the performance loss associated with sparse data. The tests measure the different algorithms' predictive accuracy, with the aim to investigate the impact of sparse data on their predictive performance. Calculations of the baselines of the datasets are performed using Weka's Zero Rule algorithm. This algorithm calculates which class is the most frequently observed and what percentage of the dataset that it constitutes. The result is then set as the baseline for that specific dataset. The predictive accuracy is compared in relation to the baseline of each dataset to analyze the performance of the algorithm. All the results are graphed for comparisons.

To achieve the most reliable results possible, as well as counteracting overfitting and underfitting, Weka's pre-implemented version of 10-fold cross-validation is used on all tests. Since the removal of data is done randomly, the predictions vary slightly depending on how the data was removed. Therefore, ten tests are conducted for each sparsity level and the prediction accuracy is calculated for each test. The mean of the ten results is calculated to make sure that the test results are as reliable as possible for each sparsity level.

3.2 Data Filtering Algorithm

To remove data and to make the datasets sparse, a self-implemented data filtering algorithm is used. The program transforms the dataset into a sparse version of the original dataset, randomly removing data based on a pre-specified sparsity. Class values are never removed in the filtering process. The program also takes a CSV file and transforms it into an ARFF file, which is the preferred dataset file type by Weka.

The missing entries are represented by ‘?’ in an ARFF-formatted dataset, which Weka interprets as a missing value.

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
```

```
5.1,?,1.4,?,Iris-setosa
4.9,3.0,1.4,?,Iris-setosa
?,3.2,?,?,Iris-setosa
4.6,?,?,?,Iris-setosa
5.0,3.6,1.4,?,Iris-setosa
5.4,?,1.7,?,Iris-setosa
4.6,3.4,?,?,Iris-setosa
?,3.4,1.5,?,Iris-setosa
4.4,?,1.4,0.2,Iris-setosa
?,?,1.5,0.1,Iris-setosa
?,3.7,?,0.2,Iris-setosa
?,3.4,?,?,Iris-setosa
?,3.0,1.4,?,Iris-setosa
?,?,1.1,?,Iris-setosa
?,?,?,0.2,Iris-setosa
5.7,4.4,1.5,?,Iris-setosa
?,?,?,0.4,Iris-setosa
5.1,?,?,?,Iris-setosa
5.7,?,?,0.3,Iris-setosa
```

(Figure 3. A dataset before and after undergoing a 50% sparsity data filtering)

3.3 Weka

Weka is used in order to implement and test the Naïve Bayes as well as the J48 and SVM algorithms, both used for a comparative purpose. Weka is a collection of Java based machine learning algorithms used for data mining tasks. There are several pre-implemented algorithms that are easily accessed and used for different problems [10].

Complications regarding the conversion of textual data into numerical data and for the algorithm to handle sparse data, required further own coding to suit the Weka parameters. The default way to represent data in Weka is through ARFF files, instead of the more common CSV dataset file types, which also is an issue that is handled by the data filtering algorithm, explained in 3.2.

4. Results

In this section, the results from the performance tests of the Naïve Bayes classifier on the Car evaluation, Pen-based recognition and Mushroom datasets are presented. For each dataset, J48 and SVM were also tested to allow for comparisons. For more detailed information about the test results, see Appendix A.

4.1 Test Results

4.1.1 Car Evaluation

For the Car Evaluation dataset, the performance difference between the three algorithms was relatively small. For a completely dense dataset, the J48 and SVM algorithms performed better than Naïve Bayes, but they have a higher decline as soon as the dataset gets sparse. Naïve Bayes does not perform as well as the others when the dataset is very dense, but at 75 % density or less, it performs superior. When the dataset is very sparse, from about 30 % density and down, the algorithms perform approximately the same as they have approached the baseline.

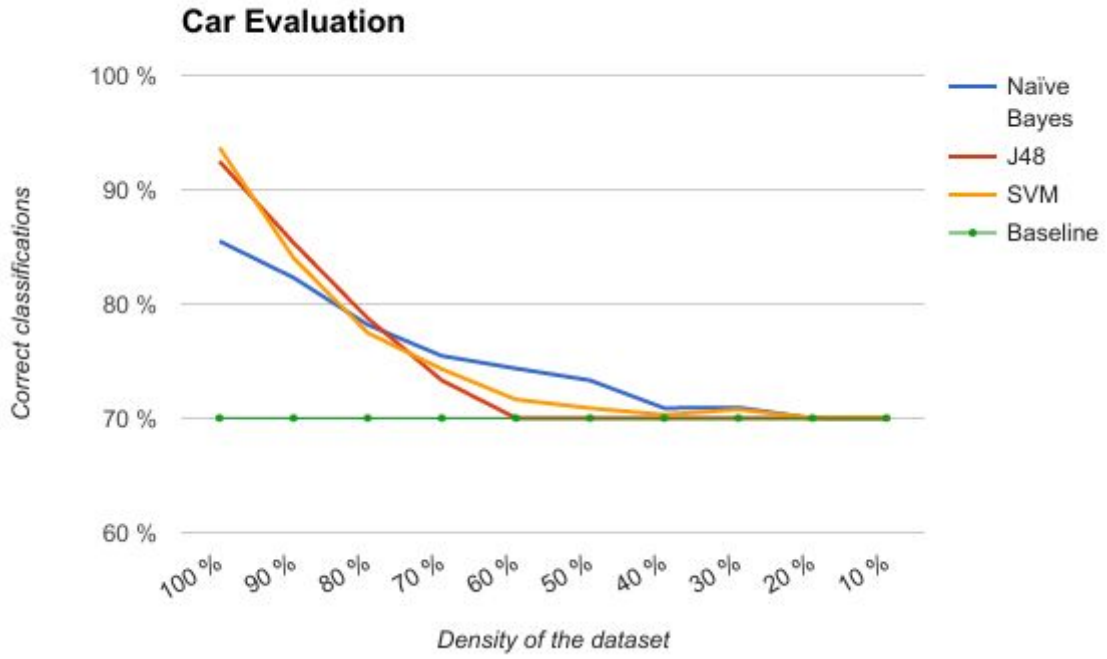

Figure 4 - Performance of the algorithms for the Car Evaluation dataset.

4.1.2 Pen-Based Recognition

The test on the Pen-Based Recognition dataset showed that Naïve Bayes performs the worst when making predictions on a 100 % dense dataset. It is, however, rather unaffected by the missing values of the dataset until it reaches sparsity levels of about 40 %. Both the J48 and SVM classifiers, on the other hand, perform better than Naïve Bayes initially, when the dataset is still very dense. Their performance does, however, decrease at a quicker rate as the sparsity increases.

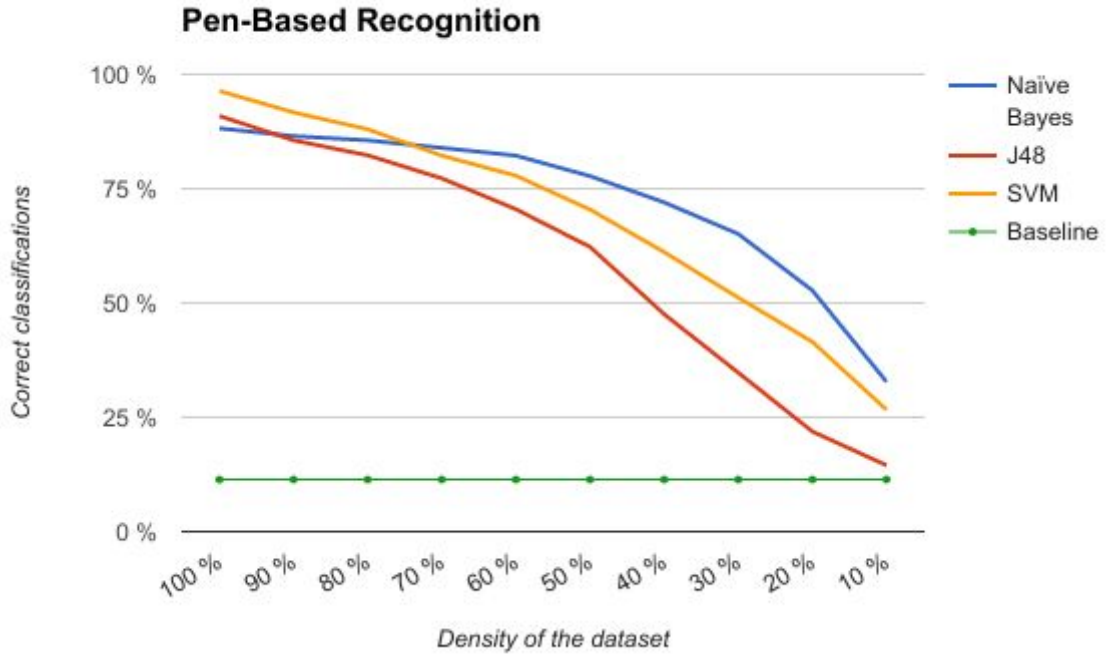

Figure 5 - Performance of the algorithms for the Pen-Based Recognition dataset.

4.1.3 Mushroom Dataset

For the mushroom dataset, results were as follows: The Naïve Bayes algorithm initially performs the worst of the three. However, the Naïve Bayes algorithm's performance decreases gradually, but at a slower rate than J48 and SVM. The Naïve Bayes algorithm has comfortably surpassed both the other algorithms when the dataset density reaches slightly below 50 %.

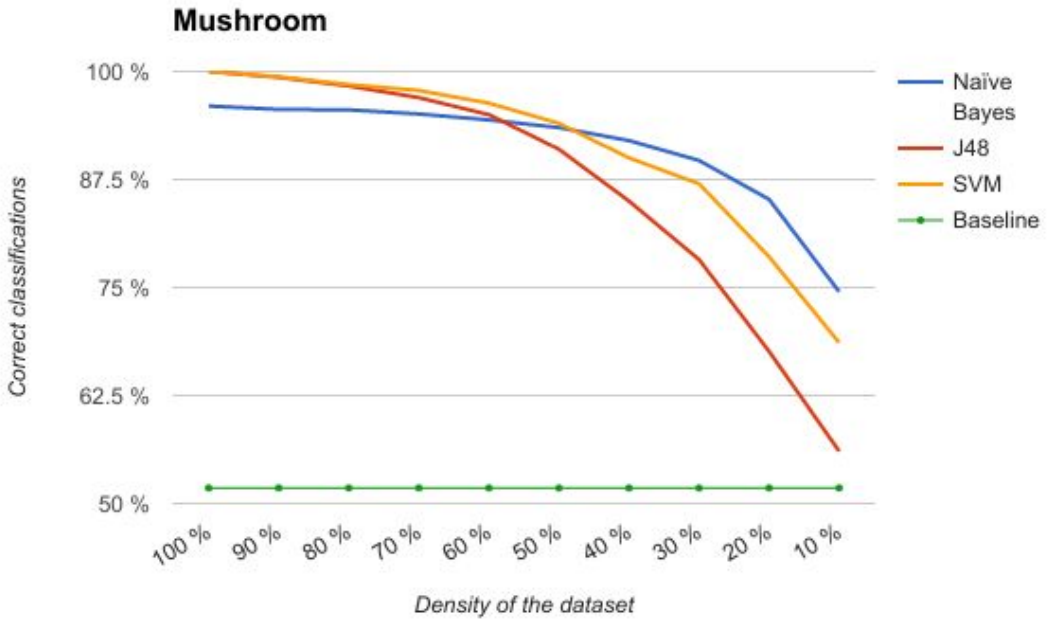

Figure 6 - Performance of the algorithms for the Mushroom dataset.

4.1.4 Australian Credit Approval Statlog

The Australian Credit Approval Statlog dataset has the SVM and J48 algorithms outperforming the Naïve Bayes algorithm at the higher densities. At a density of 60 % the prediction accuracies for the three algorithms collide. The Naïve Bayes algorithm is, however, far superior below density levels of 60 %.

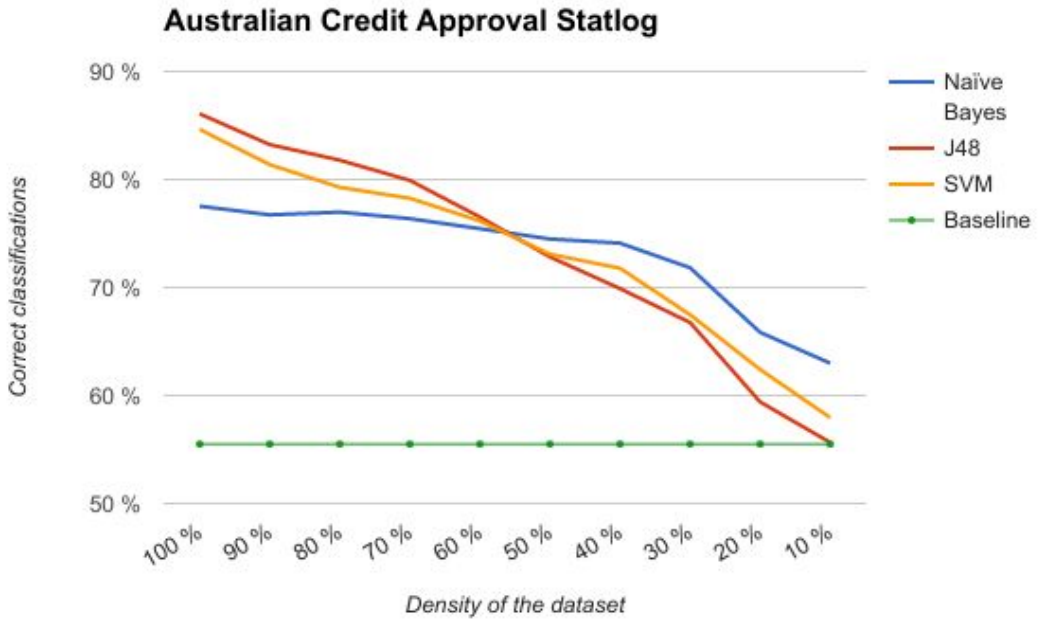

Figure 7 - Performance of the algorithms for the Australian Credit Approval Statlog dataset.

4.2 Compiled Results

4.2.1 Naïve Bayes Performance

For all datasets, the predictive performance of the Naïve Bayes algorithm deteriorates at a slow pace until about 50-60 %, see Figure 8. From that point forward, towards higher sparsity levels, the performance increasingly worsens for all datasets except the Car Evaluation dataset. For this dataset, the performance starts to stabilize, as it starts to reach the baseline of the dataset at a sparsity of around 60 %.

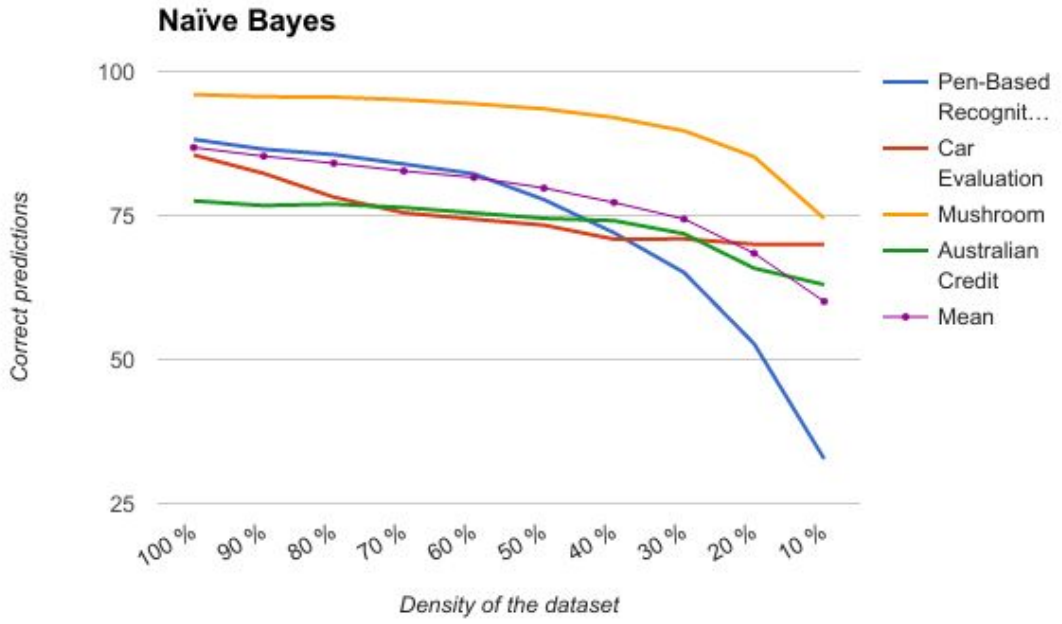

Figure 8 - Performance of the Naïve Bayes algorithm for each dataset.

4.2.2 Algorithm Performance Comparison

4.2.2.1 Crossover Points

According to the tests performed on the three algorithms and the three datasets, the initial performance of the J48 and SVM algorithms is slightly superior to the performance of Naïve Bayes. As the test results presented in Figure 9 suggest, the crossover points between the Naïve Bayes algorithm and the comparing algorithms are on average located between 65 % and 75 % density. In this interval, Naïve Bayes surpasses both of the other algorithms and continues to provide superior predictions for the remainder of the lower density levels, as shown in Figure 9.

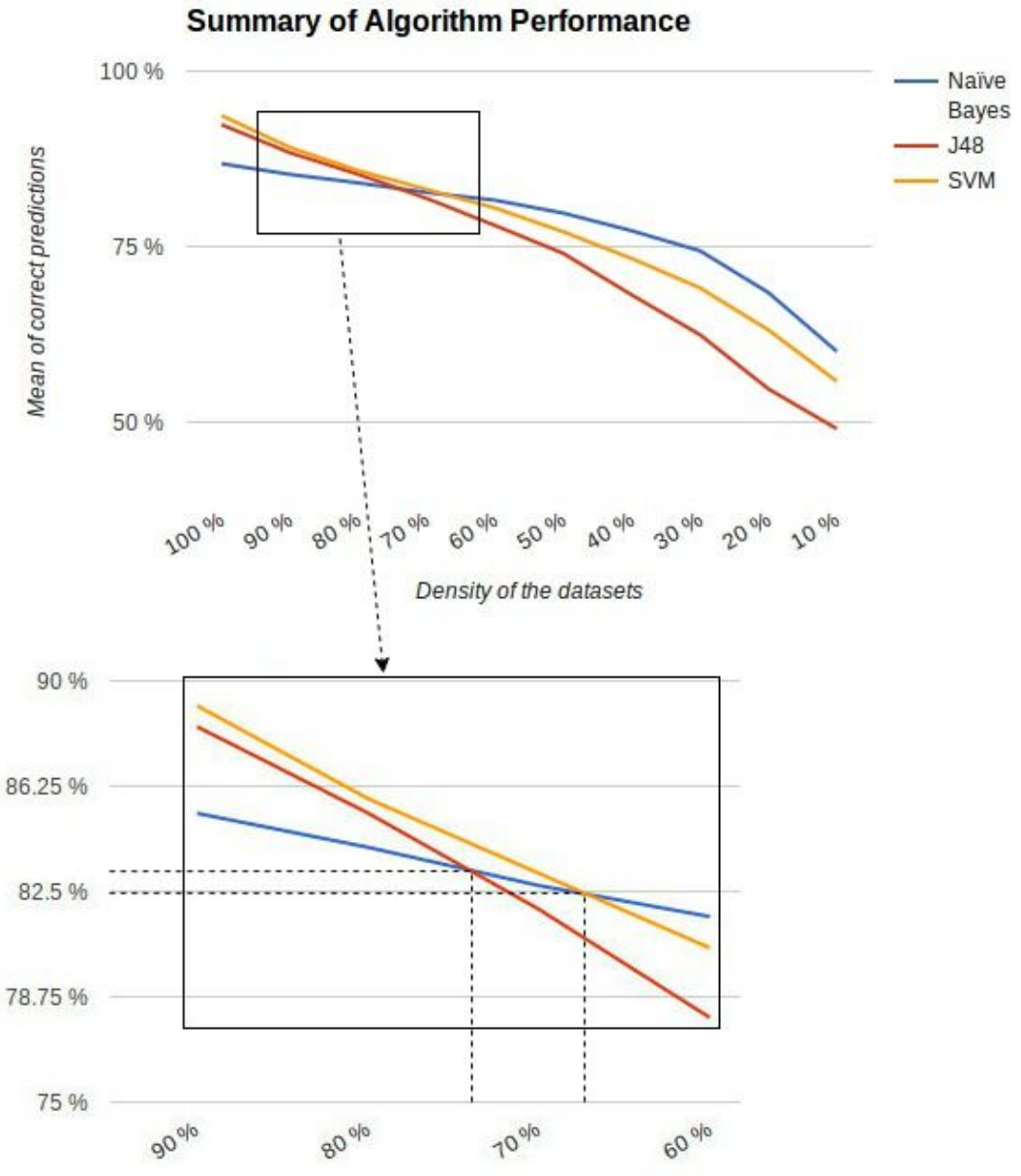

Figure 9 - Mean performance of the algorithms for the four datasets.

4.2.2.2 Percental Performance Loss

The average percental performance loss for each algorithm at specific sparsity levels are presented in Figure 10. The Naïve Bayes algorithm has the least performance loss for all sparsity levels. At 50 % sparsity, Naïve Bayes has a performance loss of 8.1 %, while the SVM and J48 algorithms are at 17.7 % and 19.9 % respectively. For further and more specific details, see Appendix A.

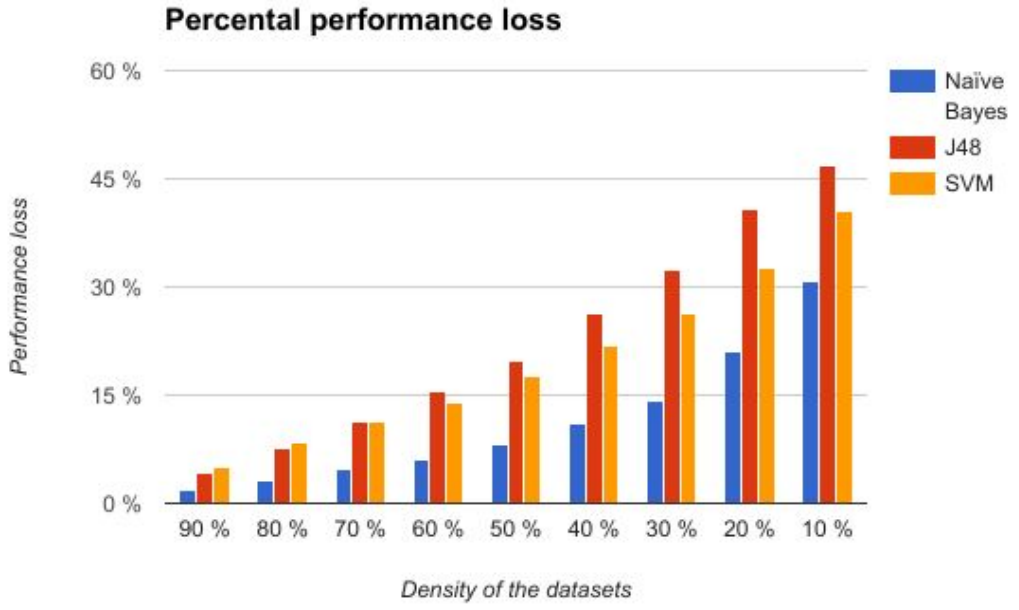

Figure 10 - The average percental performance loss for the algorithms.

5. Discussion & Conclusion

In this section the results produced and presented in Section 4 are discussed and analyzed. A critical analysis regarding the methods used in our research, followed by a final conclusion of our work and our results conclude the report.

5.1 Discussion

For all datasets, the Naïve Bayes algorithm performs slightly worse than the J48 and SVM algorithm implementations when the datasets are completely dense. The prediction accuracy for the J48 and SVM algorithms does, however, deteriorate at a higher pace than the Naïve Bayes algorithm. These results, therefore, prove that Naïve Bayes is the least affected by the increase in dataset sparsity.

As anticipated, the comparative performance loss associated with the increase in sparse data is less significant for the Naïve Bayes algorithm. However, the difference in comparative performance loss is far greater than expected. The average loss in performance for the Naïve Bayes algorithm at 50 percent sparsity is approximately 8 percent. At the same sparsity level, the performance of the J48 and SVM algorithms declines an average of approximately 19 percent. Since the performance disparity is substantial, it further emphasizes the proficiency of the Naïve Bayes algorithm when handling sparse data.

Analysis of the crossover points for the datasets, suggests that the Naïve Bayes algorithm is generally superior when sparsity levels reach a sweet spot of somewhere between 20 and 50 percent. The results of this research, therefore, further support the claim that classification tasks handling highly sparse datasets could greatly benefit from using a Naïve Bayes classifier.

From the results, further conclusions can be drawn regarding the Naïve Bayes algorithm contra other common classification algorithms. Since the J48 and SVM implementations seem to provide more accurate predictions while data is completely dense, it is arguable that the Naïve Bayes algorithm may not be suited for classification of data containing no, or small amounts of missing values.

5.2 Methodology Discussion

Firstly, in order to produce more accurate results, a larger number of unique datasets could have been used in testing. As different classification algorithms behave and perform differently on different data, more scientifically accurate results could have been achieved by testing a wider range of different datasets. The number of comparative algorithms could also have been increased to get a more accurate picture of how the Naïve Bayes algorithm performs in comparison to other classification algorithms when encountering sparse data. Further testing of additional datasets as well as algorithms was, however, prevented due to time restraints.

The study also is also limited to exclusively using the algorithm implementations provided by Weka. Our results and conclusions are, therefore, solely based on their implementations of the three algorithms and may not apply for other alternative implementations.

Furthermore, data removal is carried out manually and at random. Therefore, the research gives no indication of how the Naïve Bayes algorithm behaves if data is not missing at random, which is a frequent occurrence in data analysis.

5.3 Conclusion

As previous research suggests, the Naïve Bayes algorithm provides extremely competitive and potent predictions when handling sparse data in comparison to other classification algorithms. The predictive accuracy of the Naïve Bayes algorithm is comparatively far less affected by increased sparsity in data. Classification tasks with datasets containing large amounts of missing values could, thereby, strongly benefit from using an implementation of the Naïve Bayes classifier.

Even though there may be some deviation in results between datasets, especially in analysis of the crossover points, the general results are conclusive. It is, however, evident that the dataset shape and characteristics play a key role in the algorithm’s predictive accuracy concerning the sparse data problem.

6. References

1. Kotsiantis, S. “Supervised Machine Learning: A Review of Classification Techniques.” *Informatica.si*, University of Peloponnese, 2007, http://www.informatica.si/index.php/informatica/article/viewFile/148/140. Accessed 12 May. 2017.
2. Gale, Thomas. “Bayes, Thomas.” *International Encyclopedia of the Social Sciences*, Encyclopedia.com, 2008, www.encyclopedia.com/people/science-and-technology/mathematics-biographies/thomas-bayes. Accessed 27 Feb. 2017.
3. Staff, Investopedia. “Bayes' Theorem.” *Investopedia*, 2 Oct. 2009, www.investopedia.com/terms/b/bayes-theorem.asp. Accessed 27 Feb. 2017.
4. Keogh, Eamonn. “Naïve Bayes Classifier.” *University of California, Riverside*, www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf. Accessed 27 Feb. 2017.
5. Najafi, Safir, and Ziad Salam. “Evaluating Prediction Accuracy for Collaborative Filtering Algorithms in Recommender Systems.” *Diva-Portal.org*, Royal Institute of Technology, 2016, kth.diva-portal.org/smash/get/diva2:927356/FULLTEXT01.pdf. Accessed 27 Feb. 2017.
6. Adomavicius, Gediminas, and Jingjing Zhang. “Stability of Collaborative Filtering Recommendation Algorithms.” *Uiowa.edu*, University of Iowa, 2010, dollar.biz.uiowa.edu/~street/adomavicius11.pdf. Accessed 27 Feb. 2017.
7. Li, Xiang, et al. “The Convergence Behavior of Naive Bayes on Large Sparse Datasets - IEEE Xplore Document.” *IEEE Xplore*, 7 Jan. 2016, ieeexplore.ieee.org/document/7373401/. Accessed 27 Feb. 2017.
8. Alasalmi, Tuomo, Heli Koskimaki, Jaakko Suutala, and Juha Roning. "Classification Uncertainty of Multiple Imputed Data." 2015 IEEE Symposium Series on Computational Intelligence (2015): n. pag. Web. 5 June 2017.
9. Ricci, Francesco, Lior Rokach, and Bracha Shapira. *Recommender systems handbook*. New York, N.Y.: Springer, 2015. Print.
10. "Weka 3: Data Mining Software in Java." *Weka 3 - Data Mining with Open Source Machine Learning Software in Java*. N.p., n.d. Web. 24 Apr. 2017
11. "J48 decision tree." *Mining at UOC*. N.p., n.d. Web. 24 Apr. 2017
12. Ray, Sunil, Faizan Shaikh, Tavish Srivastava, and Swati Kashyap. "Understanding Support Vector Machine algorithm from examples (along with code)." *Analytics Vidhya*. N.p., 13 Sept. 2016. Web. 24 Apr. 2017
13. "Data Mining Concepts." *Classification*. N.p., 01 July 2008. Web. 29 Apr. 2017

14. "Sign In." *RPubs - Multinomial classification project - Car Evaluation data set*. N.p., 18 Oct. 2015. Web. 29 Apr. 2017
15. Little, Roderick J.A., and Donald B. Rubin. *Statistical analysis with missing data*. New York, N.Y.: Wiley, 1987. Print
16. Priya, S., and Antony Selvadoss Thanamani. "Multiple Imputation Of Missing Data Using Naïve Bayes ian Classifier ." International Journal of Innovative Research in Computer and Communication Engineering, 2 Apr. 17. Web. 30 May 2017.
17. Aragon, Nyssa, William Lane, and Fan Zhang. "Classification of Hand-Written Numeric Digits." *Stanford.edu*. N.p., 12 Dec. 2013. Web. 29 Apr. 2017
18. Shapire, Rob. "Machine Learning Algorithms for Classification." Cs.princeton.edu. Department of Computer Science at Princeton University, n.d. Web. 11 May 2017. <http://www.cs.princeton.edu/~schapire/talks/picasso-minicourse.pdf>
19. "Mushroom Data Set." UCI Machine Learning Repository: Mushroom Data Set. N.p., n.d. Web. 11 May 2017.
20. "Statlog (Australian Credit Approval) Data Set." *UCI Machine Learning Repository: Statlog (Australian Credit Approval) Data Set*. N.p., n.d. Web. 23 May 2017.
21. Witten, Ian H., and Eibe Frank. *Data mining practical machine learning tools and techniques*. Amsterdam: Morgan Kaufmann, 2005. Print.
22. Li, Xiang, Charles X. Ling, and Huaimin Wang. "The Convergence Behavior of Naive Bayes on Large Sparse Datasets." *2015 IEEE International Conference on Data Mining* (2015): n. pag. Web. 30 May 2017.
23. Sharma, Anjana, Naina Mehta, and Iti Sharma. "Reasoning with Missing Values in Multi Attribute Datasets." N.p., 5 May 2013. Web. 30 May 2017.
24. "Class SMO." *SMO*. N.p., 19 Dec. 2016. Web. 30 May 2017.
25. Hand, David J., and Keming Yu. "Idiots Bayes?Not So Stupid After All?" *International Statistical Review* 69.3 (2001): 385-98. Web. 30 May 2017.

Appendix A

Detailed Test Results

The mean values for the ten tests conducted for each sparsity level are presented for each dataset and algorithm in the following sections, as well as the baseline for each dataset. There is one algorithm per row and one sparsity level (in percent) per column in the tables.

Car Evaluation

Baseline: 70.02 %

	100 %	90 %	80 %	70 %	60 %	50 %	40 %	30 %	20 %	10 %
NB	85.5	82.3	78.18	75.46	74.36	73.32	70.89	70.94	70	70.02
J48	92.48	85.35	78.76	73.32	70.02	70.02	70.02	70.02	70.02	70.02
SVM	93.69	84.02	77.48	74.3	71.64	70.89	70.31	70.77	70.02	70.03

Pen-Based Recognition

Baseline: 11.37 %

	100 %	90 %	80 %	70 %	60 %	50 %	40 %	30 %	20 %	10 %
NB	88.2	86.5	85.57	83.93	82.23	77.75	72	65.1	52.73	32.76
J48	90.9	85.57	82.3	77.26	70.5	62.26	47.55	34.75	21.89	14.49
SVM	96.37	91.68	87.99	82.23	77.83	70.43	61.12	51.17	41.5	26.65

Mushroom

Baseline: 51.8 %

	100 %	90 %	80 %	70 %	60 %	50 %	40 %	30 %	20 %	10 %
NB	96	95.61	95.56	95.08	94.39	93.5	92	89.7	85.2	74.5
J48	100	99.323	98.32	96.97	95	91	85	78.23	67.6	56.05
SVM	100	99.38	98.48	97.8	96.33	94	90	87	78.55	68.62

Australian Credit Approval Statlog

Baseline: 55.5 %

	100 %	90 %	80 %	70 %	60 %	50 %	40 %	30 %	20 %	10 %
NB	77.53	76.723	76.976	76.373	75.443	74.49	74.103	71.83	65.843	62.99

J48	86.08	83.23	81.786	79.916	76.52	72.83	69.896	66.76	59.406	55.646
SVM	84.64	81.373	79.273	78.256	76.163	73.083	71.783	67.483	62.406	57.956

Mean performance of the Algorithms over all Datasets

	100 %	90 %	80 %	70 %	60 %	50 %	40 %	30 %	20 %	10 %
NB	86.808	85.283	84.072	82.711	81.606	79.765	77.248	74.393	68.443	60.068
J48	92.365	88.368	85.292	81.867	78.010	74.028	68.117	62.440	54.729	49.052
SVM	93.675	89.113	85.806	83.147	80.491	77.101	73.303	69.106	63.119	55.814

Percental Performance Loss over all Datasets

	90 %	80 %	70 %	60 %	50 %	40 %	30 %	20 %	10 %
NB	1.756	3.152	4.719	5.992	8.113	11.012	14.302	21.155	30.804
J48	4.327	7.658	11.366	15.542	19.853	26.253	32.399	40.747	46.894
SVM	4.870	8.400	11.239	14.074	17.693	21.747	26.228	32.619	40.417