Master's Degree in Informatics Engineering

Dissertation

# Finding the Critical Sampling Size of Big Datasets

July, 2017

**José Miguel Parreira e Silva**
*jmpsilva@student.dei.uc.pt*

**Advisor**
Prof. Dr. Bernardete Ribeiro

**Co-Advisor**
Prof. Dr. Andrew H. Sung

**Co-Advisor**
Prof. Dr. César Teixeira

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

University of Coimbra
Department of Informatic Engineering

Master's Degree in Informatics Engineering
Dissertation
July, 2017

# Finding the Critical Sampling Size of Big Datasets

***Author***
José Miguel Parreira e Silva
University of Coimbra
Department of Informatic Engineering
jmpsilva@student.dei.uc.pt

***Supervisor***
Prof. Dr. Bernardete Ribeiro
University of Coimbra
Department of Informatic Engineering
bribeiro@dei.uc.pt

***Jury***

| | |
|---|---|
| Prof. Dr. António Correia | Prof. Dr. Carlos Fonseca |
| University of Coimbra | University of Coimbra |
| Department of Informatic Engineering | Department of Informatic Engineering |
| dourado@dei.uc.pt | cmfonsec@dei.uc.pt |

*“Size matters.”*

# *Acknowledgements*

First of all, I want to thank my supervisors, professor Bernardete Ribeiro, professor Andrew H. Sung and professor César Teixeira. This was a joint work and without your help this work would not be the same. I learned a lot this year, thanks to their guidance. It was a unique journey that really made me travel. Once again, I thank my supervisors who have done everything to make this possible.

I must also thank another great friend and contributor to my success, professor Nuno Lourenço. Thank you for all the enlightening conversations.

To my friends, particularly, João Macedo, Nelso Oliveira (not a typo) and Pedro Magalhães a big thank you.

A very special thank you to my partner in crime.

Finally, to those who have supported me the longest, my family. And for the one like me, my best friend, this is for you.

José Miguel

# *Abstract*

Big Data allied to the Internet of Things nowadays provides a powerful resource that various organizations are increasingly exploiting for applications ranging from decision support, predictive and prescriptive analytics, to knowledge and intelligence discovery. In analytics and data mining processes, it is usually desirable to have as much data as possible, though it is often more important that the data is of high quality thereby raising two of the most important problems when handling large datasets: sample and feature selection. This work addresses the sampling problem and presents a heuristic method to find the "critical sampling" of big datasets.

The concept of the critical sampling size of a dataset is defined as the minimum number of examples that are required for a given data analytic task to achieve a satisfactory performance. The problem is very important in data mining, since the size of data sets directly relates to the cost of executing the data mining task.
Since the problem of determining the optimal solution for the Critical Sampling Size problem is intractable, in this work a heuristic method is tested, in order to infer its capability to find practical solutions.

Results have shown an apparent Critical Sampling Size for all the tested datasets, which is rather smaller than the their original sizes. Further, the proposed heuristic method provides a practical solution to find a useful critical sample for data mining tasks.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

This work was developed in the scope of the Dissertation in Intelligent Systems for the Master's Degree in Informatics Engineering at the University of Coimbra, during the academic year 2016/2017.

An increasing amount of data is becoming available on the Internet, in 2013 it was stated by IBM that "a full 90 percent of all the data in the world has been generated over the last two years"[1] and in 2016, CISCO predicted that by the end of the year the annual global IP traffic will pass the zettabyte threshold[2]. Every day, almost everything that we do is monitored and recorded, data is generated at an enormous and increasing speed by all sorts of sources. Two of the easiest examples to think about are Google, when we search about a topic, and Facebook, when we post something. All these huge volumes of data that are being gathered by companies such as these two, or from others from different areas, is now what it is called Big Data.

Big Data allied to the Internet of Things [3] provides a powerful resource that various organizations are increasingly exploiting for applications ranging from decision support, predictive and prescriptive analytics, to knowledge extraction and intelligence discovery. In addition to these uses we can list many others and, the methods for working with large amounts of data are defined as Data Mining. In analytics and data mining processes, it is usually desirable to have as much data as possible, though it is often more important that the data is of high quality thereby two of the most important problems are raised when handling large datasets: sampling and feature selection. As said before, when organizations use the data for knowledge extraction, they resort to machine learning. These machines are trained with a "desirable" good quality data which they will then construct a model from. The big question is how much data we need to get satisfactory results and at the same time to the built model not resulting in overfitting, memorize the training data instead of learning from it, or generalization, not being able to perform well on new unseen data.

## 1.2 Goals

Previous work, to see if it is possible to find a critical feature dimension for big datasets has been conducted, and it concluded to be feasible. Also for sampling, a sampling heuristic has been proposed.

This dissertation addresses the problem of sampling large datasets and how to find their Critical Sampling Size. This problem can be described as discovering the minimum number of examples that need to be used, in order to achieve a performance threshold $T$. For that, the heuristic method proposed in [4] is used because, as the authors also proved, it is impossible to determine the optimal solution to this problem, since it is both NP-hard and coNP-hard. That being said, the goals of this work are as follows:

- Implement the heuristic for determining the Critical Sampling Size of datasets and design possible alternatives.
- Conduct an empirical study on data sets to verify if they possess, or do not possess, a Critical Sampling Size.
- Study the effect of different machine learning algorithms on the Critical Sampling Size.
- Compare the performance of data mining using the complete dataset and the reduced dataset, that is, the Critical Sampling.

## 1.3 Contributions

In the era of Big Data a huge amount of data, from a variety of sources and types, is generated at ever increasing rates. This data can be used for various purposes, and increasing numbers of industries and organizations are recognizing its utility, e.g. using machine learning they can perform data mining tasks to extract insights from the data in order to make strategic decisions. These tasks of knowledge discovery could be very expensive, such as resource-heavy and time consuming, therefore, it is highly desirable to have techniques that can help select only the necessary data instead of using entire datasets. This is because much of the data may be redundant, duplicated, irrelevant, or even useless.
As the authors in [5] said, "Scalability is a key requirement for any knowledge discovery and data mining algorithm. It has been previously observed that many well known machine learning algorithms do not scale well". With this proposed sampling method, the size of the used datasets may decrease drastically, thus allowing better scalability for the algorithms as well as making the dataset size impact on performance of data mining tasks less significant. This alied with methods of finding the critical feature dimension of big datasets could be an added value. Furthermore, this can be applied to any kind of data mining task.

## 1.4 Document Structure

This document is structured in 5 chapters. The next chapter elucidates the work done in the scope of the Critical Feature Dimension and Critical Sampling Size problems. In Chapter 3 the goals of this dissertation are described and the proposed heuristic is presented.Chapter 4 details the experimental configuration aspects, such as datasets used, sampling methods studied, and considerations on various heuristic parameters. Next, in Chapter 5, results are presented and discussed. Finally some conclusions about this work are exposed.

# Chapter 2

# State of the Art

Although the problem being addressed is the Critical Sampling Size (CSS), the Critical Feature Dimension (CFD) is equally important since both are strongly related, and are two of the most relevant issues in the Big Data field.
In this chapter, previously developed works that relate to the subjects mentioned above, are described. In Section 2.1 the work done on the CFD problem is presented, then in Section 2.2 the one carried out about the CSS problem.

## 2.1 Feature Dimension Problem

When speaking about CFD, it comes to feature reduction or feature selection. These are methods that reduce the dimension of an example and consequently of the whole dataset. By reducing the number of features to a small subset, it is possible to observe a corresponding reduction in the sample size that the algorithm needs to guarantee a good generalization capability [6][7].

### 2.1.1 Feature Relevance

The problem is how to define which features are relevant and which are not, and consequently how to combine them, thus, the concept of relevance has to be defined. As the authors stated in [7], the concept of relevance can differ depending on what is being considered. The simplest and more intuitive notion of relevance is the concept of relevant to the target class. It is described as having two examples, from the dataset, that differ only in one feature which causes the assigned classes to be different. The second definition presented by the authors is very similar to the first one except that both examples now have to be present in the training set. With this definition, the feature is considered strongly relevant to the sample. Thereafter a weakly relevant feature for the sample is defined as if it is possible to remove a set of features to the examples so that it becomes strongly relevant.

Although a feature may be considered relevant, it may not be useful for a given algorithm and because of this, another way to define the utility of a feature is given. It takes into account the learning algorithm and a set of features which is incremented by adding one feature at the time. Each time this is done, if the performance obtained by the algorithm is better, then the added feature is considered useful.

### 2.1.2 Feature Selection Methods

There are plenty of feature selection methods. In [8] three categories of feature selection methods are described, while presenting some examples for each. Those categories are, filter, wrapper and embedded methods.

Filter methods are commonly used in the pre-processing phase. They mainly rely on ranking algorithms to select features which then are applied to a learning machine. Pearson correlation coefficient and Mutual Information are two examples of filter methods that are described in the paper. As stated by the authors, ranking techniques are simple and effective thus making filter methods computationally light. In addition, filter methods are independent to the learning machine.

Wrapper methods use an algorithm performance to evaluate the feature subset quality. This feature selection criterion requires the learning machine to be trained and tested each time that a feature subset is evaluated. This can be the main drawback of wrapper methods, if the size of the dataset is considerably big. Another disadvantage of using machine learning performance as an objective function is the propensity to overfitting. In this category are included, sequential selection algorithms and heuristic search algorithms, such as the Sequential Feature Selection (SFS) and Genetic algorithms, respectively.

The last one, Embedded methods try to combine the advantages of both Filter and Wrapper method. The filter phase is incorporated in the training phase and then the results outputted by the model help to redefine the subset of features to use in future iterations of the algorithm. Support Vector Machines (SVM) and Artificial Neural Networks (ANN) are examples of this, as during their training phases they update the weights associated to features. And in specific cases, they can even remove features using methods such as the SVM-RFE, for SVM [9][10], and the Network Pruning for the ANN [11][12].

### 2.1.3 Critical Feature Dimension

Authors in [13] study one of the problems of mining Big Data, the Critical Feature Dimension which is a feature selection problem. CFD is defined, in an intuitive way, as being the absolute minimal number of features, of a specific dataset, that are needed to ensure that the performance of a given learning machine meets a performance threshold.

They propose a heuristic that first ranks the features, with Chi-Square ranking method, and then sorts them by descent order. After that, in an iterated way, the least ranked feature is removed and checked if occurs a decrease in the performance, and if that so, then the CFD was found. The possibility of performance rising again was also considered, and therefore if that happened, it was not really the CFD.
In the results, each one of the used data sets have shown an apparent CFD, and also that the feature size has significantly decreased, while maintaining a desirable performance.

## 2.2 Sample Size Problem

In this section works done in the scope of sample reduction are presented. Like the feature selection problem, these works also aim to reduce the size of the dataset, but instead of performing feature reduction, by doing sample reduction.
Some of them are specific to one type of algorithm and others are generic.

### 2.2.1 Support Vector Machines optimization

In [14], a new algorithm to speed up the training time of SVM is presented. The major drawback of SVM is their training time and the proposed method tries to select a small and representative amount of data from the dataset to improve that. The new method showed up to be very efficient for with big datasets.

It starts by training a SVM with a small amount of the dataset examples in order to obtain a sketch of the optimal separating hyperplane. The amount of examples is defined by a technique that considers the unbalance of the dataset. And then, a decision tree is used to detect and, only retain, examples from the dataset that have similar characteristics to the computed Support Vectors, which are the ones that contribute to define the separating hyperplane.
According to the results, the new algorithm was able to be faster than the current SVM implementations and generate results with similar performance.

### 2.2.2 Adaptative Sampling

Regarding to the scope of this dissertation, some studies have been conducted in order to find ways of sampling big datasets, like in [5], where the authors propose an adaptive sampling method.

The proposed algorithm sequentially obtains examples and then it determines whether it has already seen a large enough number of examples. Thus, sample size is not fixed in the beginning of execution, instead, it depends on the problem and the dataset. When working with big datasets one approach is to reduce their size by applying some kind of random sampling. In many cases this method could be applicated, since sometimes an approximate answer to the problem being addressed is sufficient. But this is not recommendable due to the difficulty of determining the appropriate sample size [15]. For instance, the training set size is one of the factors that must be taken in consideration.

The goal of applying data mining in big datasets is to infer structures, rules that describe well as much as possible their data. So the objective is, from the possible set of rules, to find the one that better defines the dataset. This problem, that authors tries to solve with random sampling, they called it *General Rule Selection*. As the proposed solution to solve this problem does not define an a priori sampling size, the Hoeffding's bound was used. Although this limit allows to calculate the sampling it usually sets very high values, so it was used to estimate the error when calculating the sampling size.

The proposed method was called *AdaSelect* and it is based on preliminary work developed by the authors[16][17]. Instead of sampling in batches, it is done by sequentially obtaining examples, one by one. And at each step it is checked whether enough have been collected, to issue with high confidence, the best rule. With the results obtained the proposed method showed to be accurate enough and to greatly reduce the running time for each dataset. As the authors stated, one of the key points when performing sequential sampling is to determine a good stopping condition.

### 2.2.3 Predicting Sample Size

In real cases of data mining, the access to large amounts of labelled data is often scarce. Following this premise, the authors in [18] present a new method to predict the performance that a classifier will achieve with a relatively big sample size.

With this method, the classifier is tested several times, having been trained with training sets of different dimensions. With the obtained performances associated with each training set size, it models a function of a curve that will be used to predict classifier's future results. The authors also postulated that the classifier performance at a larger training set size is more indicative of the classifier's future performance.

### 2.2.4 Critical Sampling Size

The concept of the Critical Sampling Size presented in [4] is very similar to the one presented for the CFD [13]. Like in the CFD problem, it has been proved that CSS is a NP-hard and coNP-hard problem and that is why it was also proposed a simple heuristic to solve it. It also follows an iterative approach, however, having some different aspects.

The papar states that for a specific dataset, it may exist a sample of it that is the minimal quantity of data needed to apply to a learning machine, such that its performance exceeds a given threshold. Other issue addressed in this paper is the quality of data. Nowadays with the exponential growth of data, and its use in many fields of knowledge, a method to calculate the quality of datasets can be very useful. It was designed a metric such that the optimal case is when all data is essential for the data mining task, meaning that the CSS is equal to the dataset size and on the other hand, that when the heuristic fails to find a CSS it is considered the worst case. To calculate the quality of the datasets, this metric covers both problems, CFD and CSS.

# Chapter 3

# Critical Sampling Size

In this section the concept of Critical Sampling is defined, and the heuristic method proposed in [4] described. An intuitive way to define the concept of CSS is the absolute minimal number of examples, of a given dataset, that are needed for a specific learning machine to achieve some performance threshold.

## 3.1 Problem Formulation

To formally define this problem, a given dataset is represented by $D_n$, where $n$ denotes the number of examples at hand. The learning machine is designated by $M$ and the performance threshold by $T$. With respect to the notation, $v$ is used to denote the CSS where $v \leq n$.
This way, using these notations it is possible to define it with the following two conditions. In order for $v$ to actually represent the Critical Sampling Size of a specific dataset, the two conditions must hold.

There exists a $D_v$ which lets $M$ achieve a performance of at least $T$, where $D_v$ is a sample of $D_n$ with $v$ examples and $P_M(D_v)$ is the performance of $M$ when trained with $D_v$

$$(\exists D_v \subset D_n)[P_M(D_v) \geq T] \tag{3.1}$$

For all $j < v$, a sample of $D_n$ with $j$ examples fails to let $M$ achieve a performance of at least $T$

$$(\forall D_j \subset D_n)[j < v \Rightarrow P_M(D_j) < T] \tag{3.2}$$

Thus, $v$ is the critical (absolute minimal) number of data examples required in any sampling to ensure that the performance of $M$ meets the given performance threshold $T$. The problem of deciding if a given number $v$ is the CSS for the dataset $D_n$, with respect to a given learning machine $M$ and performance threshold $T$ is both NP-hard and coNP-hard; a sketch of the proof is given in [4]. For this reason, heuristic methods are used, which can also obtain satisfactory results.

## 3.2 Complexity

This section presents a short explanation of the proof presented in [4] showing that the CSS problem is both NP-hard and coNP-hard. In order to prove its complexity, a known NP-hard and coNP-hard was transformed into the CSS problem. The used problem was the Exact Maximal Independent Set (EMIS), or Maximum Independent Set, which is defined as the maximum number of vertices in a graph where none is adjacent to another. Figure 3.1 presents an example of a Maximum Independent Set.

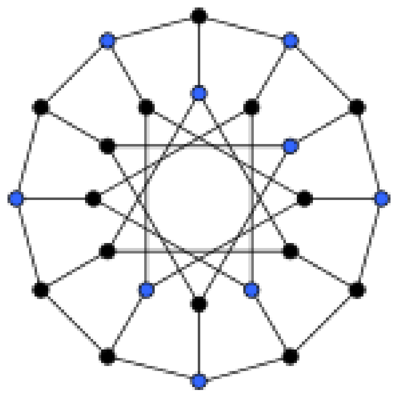


FIGURE 3.1: Maximum Independent Set

Source: https://en.wikipedia.org/wiki/Independent_set_(graph_theory)

Given a graph with $n$ nodes, finding the EMIS can be further translated into deciding if there is a Maximum Independent Set of size exactly $k$, where $k \leq n$.
To transform the CSS problem into the EMIS problem, a dataset $D_n$ with $n$ examples is represented by a graph $G$ with $n$ nodes and $v$, the critical number of data examples, corresponds to $k$. After that, if the solution to an instance of the EMIS problem is true, the solution to the constructed instance of the CSS problem is also true.

## 3.3 Heuristic Method

In this section, the proposed heuristic method to find a critical sampling is presented. The heuristic is a sequential example selection method composed of five steps. It is in fact a meta-heuristic algorithm since it is a higher-level procedure that resorts to other algorithms (learning machines) in order to solve the CSS problem.

This heuristic fits into the Wrapper methods group, since it treats the learning machines as black boxes and uses their performance as the objective function to evaluate the critical sample being created.

1. Apply a clustering algorithm like k-means to partition $D_n$ into $k$ clusters. ($k$ can be determined, for example, by the number of classes in the dataset)
2. Select, say randomly, $m$ examples from each cluster to form a sample with $m * k$ examples. (The value $m$ is set to be fairly small)
3. Supplement the sample with additional $d * k$ (for some d) examples, selected randomly from the whole dataset $D_n$, to form a sample $D_v$.
4. Apply learning machine $M$ on the sample, then measure the performance $P_M(D_v)$.
5. If $P_M(D_v) \geq T$, then $D_v$ is a critical sampling, and its size $v$ is the Critical Sampling Size for $(D_n,\ M)$. Otherwise enlarge $D_v$ by repeating step 2 and step 3 until a critical sampling is found, or until the whole $D_n$ is exhausted and the procedure fails to find $v$.

# Chapter 4

# Experimental Setup

This chapter starts by clarifying the objectives of this investigation work.
It then follows by referring the used development tools and describing the datasets used to conduct this research. After that, the proposed heuristic to find the CSS is described in more detail. Finally the adopted approach method is exposed. In addition, some questions regarding the heuristic parameters are also presented.

## 4.1 Research objectives

Recent researches have been conducted with big datasets in order to find a CFD. This concept can be described as the minimum number of features required, for a specific machine learning algorithm executed on a given dataset, to achieve a satisfactory performance. The results were positive and interesting because, in some cases, the dataset size was significantly reduced.
Following these studies and its satisfactory results led to the issue of the CSS, which is the goal of this research project. For this, a heuristic has been proposed, because as has been proved, finding the optimal solution to this problem is intractable. Using various learning machine algorithms and four datasets, this project goal is to verify if a dataset does possess a CSS. After that, the performance of data mining using the complete and reduced datasets can be analyzed, and consequently, it could contribute to infer the quality of the data.

## 4.2 Tools and Datasets

All the developed code was written using Python 3 coupled with the Scikit-learn package which is built on NumPy, SciPy, and Matplotlib. This package provides simple and efficient tools for use in machine learning, like many algorithms and functions to handle data. Also, for loading datasets, the Pandas and Arff packages were used.

The datasets used in this work were downloaded from the UCI Machine Learning Repository[19]. Their dimensions vary both in the number of examples and in the number of features. This way, it is possible to better visualize the quality of the heuristic dealing with datasets of different characteristics.

Table 4.1: Datasets characteristics

| | **Ads** | **Credit** | **Hapt** | **Isolet** |
|---|---|---|---|---|
| **# Features** | 1558 | 23 | 561 | 617 |
| **# Classes** | 2 | 2 | 12 | 26 |
| **# Examples** | 3279 | 30000 | 10929 | 7797 |

**Ads dataset**

The full name of this dataset is "Internet Advertisements Dataset" and it represents a set of possible ads that may appear on Internet pages. The features encode the geometry of the image and all the text associated to it, such as URLs and words occurring near them. All its features are binary except for the three that hold continuous values. In addition, in 28% of the samples, one or more of the three continuous features are missing. This is a binary class dataset, with 3279 samples, in which 458 represent ads and the remaining 2821, non-ads.

**Credit dataset**

This dataset is binary. However, it has a higher number of samples than the Ads dataset although with a reduced number of features. It also has no missing values and the features can assume integer and real values. The dataset was built by researchers from Taiwan which aimed to predict the customers default payments in a financial risk problem. To predict whether a costumer is credible or not, information such has gender, marital status, education, history of past payments and their amounts to name a few, are used. Among the total 30000 samples, 5529 represent the customers that entered in default. In the description that comes along, artificial neural networks are referred to be a good data mining technique that can accurately estimate the probability of default.

**Hapt dataset**

Hapt is an acronym for "Human Activities and Postural Transitions". A group of 30 volunteers performed a total of 12 postural activities while wearing a smartphone on their waist. The 561 features were obtained by applying some pos-processing on the data captured with the smartphone's embedded accelerometer, used to capture 3-axial linear acceleration, and gyroscope, to capture 3-axial angular velocity. The features assume real values and the samples consist in fixed-width sliding windows of 2.56 sec and 50% overlap. This dataset has no missing values.

**Isolet dataset**

The Isolet dataset refers to "Isolated Letter Speech Recognition". The goal is to predict which letter was spoken by an individual and therefore, there are a total of 26 classes. For the creation of this dataset, 150 subjects spoke the name of each letter of the alphabet twice, but 3 records were discarded, thus it comes up with 7797 samples. The whole samples have no missing values and the features are real values.

## 4.3 Approach Method

In this section, how the experiments were conducted is described in more detail. First, an analysis of several clustering algorithms is presented. Next, the sampling method and how it will be analyzed is defined. After that, a study is made on some parameters of the heuristic. Finally, it is explained how the performance of the CSS is measured and which metrics are used.

### 4.3.1 Clustering

The first step of the proposed heuristic is to cluster the data. There are several clustering algorithms, each with different behaviors. Below, some of the existing algorithms and their behavior for four different test cases are shown. After that, a comparative analysis to help determine which one to use, is presented.

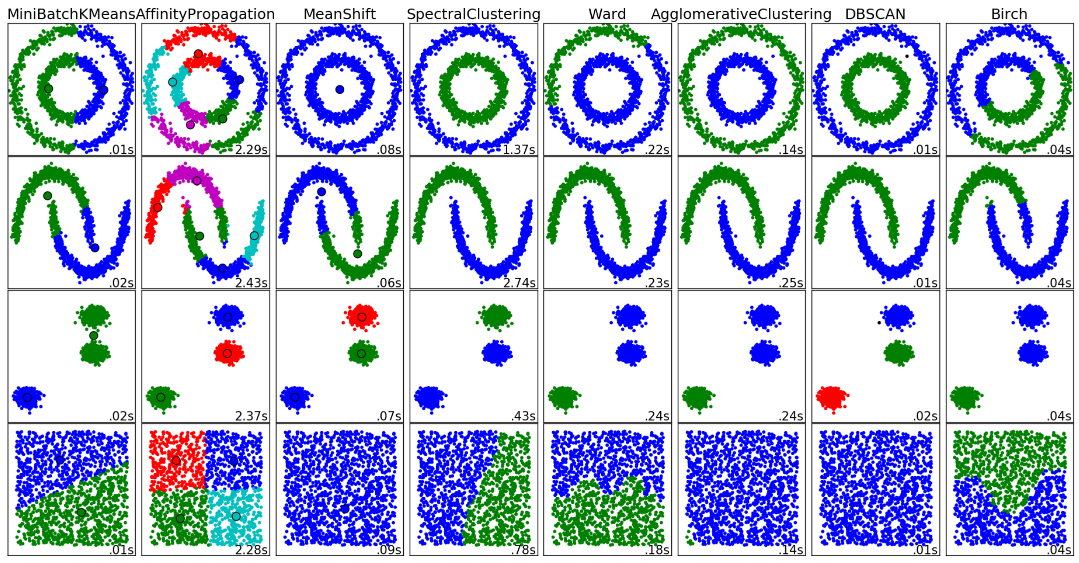


FIGURE 4.1: A comparison of clustering algorithms

Source: http://scikit-learn.org/stable/modules/clustering.html

K-means is one of the most widely used clustering algorithms. It is fast and simple to implement. The algorithm divides the data into k clusters while trying to minimize the inertia, the squared distance of each example to its closest centroid [20]. The test cases in Figure 4.1 show one of the best execution times among the different algorithms. However, there are cases where clustering results are better, such as DBSCAN, which can better identify the different patterns.

Looking at the Affinity Propagation algorithm it is possible to see that it produces many clusters for all the test cases, managing to obtain a good partitioning of the input space. A good aspect about this algorithm is that the number of clusters is defined based on the data and not by a parameter, as happens with k-means. However, due its complexity, the execution time is quite high showing that it does not scale well and so it is not a good choice for this work.

Like k-means, the Mean Shift is a centroid based algorithm, but instead of relying in a parameter to set the number of clusters it takes a bandwidth parameter. This parameter defines the neighborhood of examples that are used to compute the mean shift vector for each centroid. The algorithm does not scale well, as it requires to search multiple times the centroids nearest neighbors during its execution [21].

Spectral clustering does a dimensional reduction before applying a clustering algorithm. As can be seen in Figure 4.1, the execution times of the algorithm are quite high, demonstrating this way that it not scale so well,thus making it not one of the best options.

The agglomerative hierarchical clustering algorithms begin by considering each data example as a cluster. Then join them by following a linkage criterion, usually, pairwise distance. Ward is a specific linkage criterion to merge clusters that follows the same approach than k-means, by trying to minimize the sum of the squared distances of points. Due to the number of comparisons that are made, these algorithms have a high complexity which may limit the size of the datasets that can be processed [20].

DBSCAN is a density based clustering algorithm. It can detect outliers and because of that it may not produce a complete clustering. The number of clusters is determined by the algorithm. They are formed by locating areas of high density separated by areas of low density [20]. This approach shows good results for the first three test cases of Figure 4.1, except for the last. For datasets with a distribution similar to the last case, the algorithm may not produce a high number of clusters. In addition, it can be computational expensive to define the densinty of clusters when dealing with high dimensional data [22].

Birch has a linear scalable running time [23]. However it does not scale very well to high dimensional data [24]. It is local meaning that clustering decisions are not made scanning all the data. It can also detect sparse data examples as outliers and in contrast it treats dense areas of data as being a single cluster. This algorithm outputs similar clusters produced by k-means.

The following table summarizes more clearly the complexity of the algorithms described above. The data in the table were obtained using the following articles [25] [22].

Table 4.2: Clustering Algorithms Time Complexity

| **Algorithm** | K-means | Affinity Propagation | Mean Shift | Spectral | Ward | Agglomerative | DBSCAN | Birch |
|---|---|---|---|---|---|---|---|---|
| **Time Complexity** | O(KN) | O(KN$^2$) | O(N$^2$) | O(N$^3$) | O(N$^2$) | O(N$^2$) | O(N log(N)) | O(N) |

By analyzing Table 4.2, the k-means was the chosen algorithm, due to its speed and simplicity. DBSCAN and Birch, despite having good complexity and producing good results, gave a low number of clusters when tested with the datasets being studied.
K-means can be used for a variety of data types, however, as it calculates the distance and average of the examples, it makes it only suitable for datasets with continuous data. In addition, the algorithm does not deal with outliers, which can significantly influence the results. It also does not consider the size of clusters, which can lead to clusters of very different sizes [22].

### 4.3.2 Sampling Method

The size of both the training and test sets can influence the performance, that a learning machine achieves. This is an important aspect thus, when comparing the performance of a data mining task using the whole dataset and the sampled data, this must be taken in consideration. Because of this, three ways to split the datasets were considered, and therefore, for each of these methods, a respective CSS will be heuristically determined. Each method consists in splitting the dataset in different ratios, which are:

- 30% Train / 70% Test
- 50% Train / 50% Test
- 70% Train / 30% Test

If eventually, the CSS for the 3 splitting ratios ends up being the same, or very close to each other (the concept of closeness may differ depending on the nature of the problem), this may indicate that the CSS of the dataset is independent of the used splitting ratio.

As stated in Section 3.3, the CSS is obtained iteratively. First the data is clustered using k-means and then, from each cluster, $m$ examples are selected to form the sample $Dv$. To analyze the usefulness of this method, another two approaches were proposed.

- $mk$+$r$: Initial approach, where the sample $D_v$ is composed by selecting $m$ examples from each one of the $k$ clusters and then complemented with more $d * r$ random examples.
- $mk$: Same as above except that the sample $D_v$ is not complemented with $d*$ random examples.
- $r$: Random sampling. The construction of $D_v$ is made by randomly selected examples. In order to maintain some consistency, here $r$ can be calculated with $m * k$.

### 4.3.3 Parameters Setup

The proposed heuristic to find the CSS has four parameters (see 3.3). Their values should be decided by considering (*i*) the nature of the problem, (*ii*) the size of the dataset, (*iii*) the data mining task that will be applied and (*iv*) the amount of available resources. These four parameters are, respectively, $k$, $m$, $d$ and $T$.

**Parameter k**

This parameter represents the number of clusters in which the data will be partitioned. Initially, the number of classes of the dataset was the suggested value to assume. Note that when the dataset has only two classes the number of clusters will be diminutive. This will make the sampling process very similar to the $r$ approach, even if the $mk$, or $mk + r$, is being used. Therefore, if this method is used to define the value of $k$, increasing its value, when the number of classes is small, may be a good procedure. This may enable the creation of a more descriptive sampling of the input space.
In the experiments, several values for $k$ will be tested in order to study the effect that the number of clusters has on the results.

**Parameter m**

The value of this parameter must be small, so that with each iteration of the heuristic, as the sample $D_v$ grows, it is possible to observe how the performance of the classifiers evolves. The value of $m$ was defined as being 1% of the size of the dataset divided by the number of clusters.
Initially, it was suggested to randomly select the $m$ points. But the arrangement of examples in a cluster can be an alternative. Therefore, two more methods were considered to select the $m$ examples. The first one selects them by ascending order, relatively to the cluster centroid, the second one, by decreasing order. Later on the results section, these three methods, namely, random, increasing and decreasing, are referred to as *rand*, *asc* and *decr*, respectively.

**Parameter d**

In the $mk + r$ approach, the sample is composed by complementing it with more $d * k$ samples. These samples are randomly chosen from the remaining dataset. In order for this addition not to be as significant as step $m * k$, the value assumed by $d$ takes this into account and consequently has been defined as the value of $m$ divided by 4.
Certainly other values can be used. An interesting aspect about parameter $d$ is that it allows to check other CSS values for the same dataset. This would not be possible for the $mk$ and $r$ approaches. Therefore, it may find lower values of the CSS for a specific dataset.

**Parameter T**

The value of threshold $T$ represents a reasonable performance requirement or expectation of the specific learning machine $M$.

It is desirable to the critical sample to achieve the same performance as the one obtained when using the entire dataset, that is, when trained with 30%, 50% or 70% of the whole dataset. Hence, those values can be used to define $T$. Note that this is an empirical study in which the primary goal is to verify if datasets possesses, or do not possess, a CSS. With the datasets being used it is possible to infer, in useful time, the expected value of $T$ when using 30%, 50% or 70% of the whole dataset for training. One reason for this is to show the usefulness of our proposed heuristic. In real cases, when dealing with much larger datasets, it will have to be defined by the user.

### 4.3.4 Performance of machine M

Another aspect that deserves special attention is the method to calculate the performance of a specific machine $M$. There is more than one way proposed by the authors in [4]. The performance of machine $M$ trained with $D_v$ is denoted by $P_M(D_v)$, and can be calculated using ($D_n$ - $D_v$) as a test set or using a test set with fixed size. The second approach is more consistent, so it was the one adopted.

Another important aspect to point out is the size of the test set being used. For instance, when using the 30% for training and 70% for testing, the critical sample should also be tested with 70%. That is, for the results using the entire dataset, the model is created using 30% of the data and tested with 70%. To obtain the CSS results, the model is then trained with the sample $D_v$ and tested with 70% of the data. Next, the two performances are compared. Therefore there is no point in $v$ getting bigger than the size of the train set, in this case, more than 30% of the dataset size. So this can be considered as a stopping condition for the heuristic.

For different types of problems, different types of performance measures are required, in this case the default task of the used datasets is classification, therefore some classification metrics are needed. Next, two of the most common used metrics are described.

**Accuracy**

Accuracy function is very intuitive and easy to interpret. Basically is the sum of correctly classified examples divided by the total number of examples.

Assuming $\widehat{y}$ as the predicted value of the $i$-th example, $y_i$ as the corresponding true value, and $\mathbb{1}(x)$ as the indicator function, then the formula can be defined as

$$accuracy(y, \widehat{y}) = \frac{1}{n_{examples}} \sum_{i=0}^{n_{examples}-1} \mathbb{1}(y_i = \widehat{y}_i) \tag{4.1}$$

Indicator function is represented as

$$\mathbb{1}(expression) = \begin{cases} 1 & if\ expression\ is\ true \\ 0 & if\ expression\ is\ false \end{cases} \tag{4.2}$$

**F-score**

In many cases the *Accuracy* is not sufficient, for instance, in a dataset that is not balanced, it can give erroneous information about the prediction quality of the learning machine. Assuming that the dataset has only two classes and that the positive class has a much lower number of examples than the negative one, what could happen, depending on the algorithm and how the model was built, is that the machine could correctly classify all the negative cases and wrongly all the positive ones, resulting in a good *Accuracy* value but in the point of view of the aim of the problem, a catastrophic result. Because of this, it is necessary to resort to another metric, the *F-score*. It is a metric that allows to better evaluate the quality of a model. Firstly, *precision* and *recall*, have to be defined. *Precision* is the ability of the model to not classify a negative example as being positive and *recall* is its capacity to find all the positive examples. If *tp* is the true positives, positive examples well classified, $fp$ the negative examples wrongly classified as positive and $fn$ the positive examples wrongly classified as negatives then *precision* and *recall* are defined by

$$precision = \frac{tp}{tp + fp} \tag{4.3}$$

$$recall = \frac{tp}{tp + fn} \tag{4.4}$$

With these two functions it is possible to compute the *F-score*. The following function is the general function for *F-score* where $\beta$ can take different values. The traditional *F-score* uses $\beta$ value equals to 1

$$F_\beta = (1+\beta^2)\frac{precision \times recall}{\beta^2 * precision + recall} \tag{4.5}$$

The previous formulas are designed for binary class problems. To extend them to multiclass problems, some aspects must be taken into account. In these cases, there are different ways to calculate the *F-score*. Below, these methods are listed and their differences explained.

- Obtain the *F-score* for each class and then calculate the mean. This method is called “macro” and gives equal weight to each class.
- The “weighted” method takes into account the fact that there could exist class imbalance in the dataset. In addition to what is done in the “macro” method, each class is weighted according to its presence in the dataset.
- Calculate metrics globally by counting the total true positives, false negatives and false positives, thus giving an equal contribution to each example. This method is called “micro”.
- Calculating the sum of all the correct prediction and dividing by the total number of examples. This method is only applicable to multiclass datasets, because otherwise, it behaves the same way as *Accuracy*.

The assumption that all classes are equally important is often untrue, but in this case this may be a method of circumventing the effect of unbalanced datasets on the results. Thus, to calculate the *F-score*, the “macro” method was used, since it attributes equal weight to all classes. It can be defined by the following formula

$|L|$ is the set of classes, or labels, present in the dataset.

$$\frac{1}{|L|}\sum_{l\in L} F_\beta(y_l, \widehat{y_l}) \tag{4.6}$$

## 4.4 Experimental Setup Overview

Table 4.3 shows all the test cases that will be tested by combining the value of $k$, the sampling method, the cluster sampling method, and the training set size with which the results will be compared.
For each value of $k$, the 3 sampling methods will be tested. Both $mk$+$r$ and $mk$ methods, since they are the only ones that perform clustering, will be tested with 3 different ways of selecting examples from the clusters. These results will then be compared to the results obtained when using 30%, 50% and 70% of the dataset for training. Finally, all these combinations (test cases) will run 30 times each.

TABLE 4.3: Experimental Setup

| **K** | 2, 5, 10, 20, 30, 50 | | |
|---|---|---|---|
| **Sampling Method** | mk+r | mk | r |
| **Cluster Sampling Method** | rand<br>asc<br>decr | rand<br>asc<br>decr | - |
| **Training Set Size** | 30%, 50%, 70% | | |

# Chapter 5

# Results

This chapter presents the results. In Section 5.1, the choice of the algorithms used to conduct the experiments is explained. A description of its settings is given below. In Section 5.2 the experimental results are exposed and discussed.

## 5.1 Algorithms Settings

Initially, 6 learning machines were tested for use in the experiments. After being tested with all datasets, each divided into 70% for training and 30% for testing, it was found that for binary problems (Ads and Credit datasets) the best results were AdaBoost, and that for the remaining ones, the MLP was the best choice. Evidences of these results are depicted in Table 5.1. It can be seen that for the HAPT dataset, the KNN algorithm obtained better results than the MLP, but looking at the Isolet dataset, it is possible to see the superiority of the MLP. Therefore, AdaBoost was the chosen algorithm. With these conclusions, these two learning machines were used to carry out the experiments.

Note that the results for each algorithm were obtained without doing any kind of pre-processing in the datasets, since the goal of this dissertation is not to optimize the classifiers. The choice of algorithms to use was mainly based on performance, knowing that although they behave well for these datasets, as it is stated by the “No Free Lunch Theorem”, that may not be true for other datasets or other types of problems [26].

Table 5.1: Algorithms Tested. Results associated to each algorithm are the achieved F-score when using 70% of the dataset for training and 30% for testing.

| | | **Datasets** | | | |
|---|---|---|---|---|---|
| **Algorithm** | **Learning Machine** | **Ads** | **Credit** | **Hapt** | **Isolet** |
| Lazy | KNN | 90.25% | 54.71% | **84.18%** | 86.12% |
| Meta | AdaBoost | **92.58%** | **66.35%** | 24.89% | 12.09% |
| Artificial Neural Network | MLP | 91.09% | 52.28% | 80.26% | **95.20%** |
| Tree | Decision Tree | 92.33% | 65.78% | 48.10% | 44.74% |
| | Random Forest | 46.34% | 55.36% | 39.13% | 55.89% |
| Bayes | Naive Bayes | 69.79% | 38.17% | 61.35% | 82.76% |

**Adaptative Boost**

The Adaptative Boost (AdaBoost) classifier is a meta-estimator that fits other classifiers on the original dataset to improve their performance [27]. The implementation provided by the scikit-learn was used. It uses 50 estimators, which are Decision Tree Classifiers, at a learning rate of 1. This parameter defines the contribution of each estimator for the performance outputted by AdaBoost. It uses the SAMME.R real boosting algorithm which converges more quickly than SAMME [28].

**Artificail Neural Net - Multi Layer Perceptron**

Multi Layer Perceptron is a fully connected feedforward artificial neural network. It was also used the implementation provided by the scikit-learn, which has 3 layers. The hidden layer is composed of 100 neurons and the activation function is the rectified linear unit function. The used solver for weight optimization is the stochastic gradient descent with a constant learning rate of 0.001 and a maximum of 200 iterations.

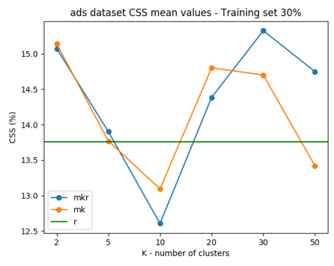
## 5.2 Experimental Results

Several values for $k$ were used in order to study the effect of the number of clusters in the results. Figs. 5.1 to 5.4 show the CSS that were needed to achieve the same performance when 30%, 50% or 70% of the datasets were used as a training set. Each dot of the lines represents the averaged CSS value, of 30 runs, for each value of $k$. For the $mk + r$ and $mk$ sampling methods, the value that is represented is the one that obtained the best results among the *rand*, *asc* and *decr* way to select the examples of the clusters. As for the $r$ method, random sampling, the CSS is represented by a straight line, which value is also the averaged value of all runs. These results can be seen in more detail in Appendix A, Tables A.1 to A.8.

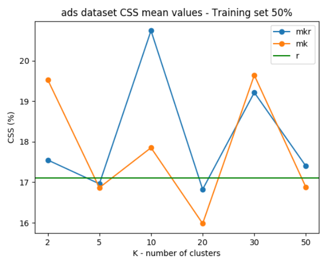
In Figs. 5.1a to 5.1c, it is not so visible the improvement of the heuristic ($mk + r$ and $mk$ methods) over the random sampling. Moreover, in the majority of cases it is the one that gets the best results. However, as can be seen in Figure 5.1a, when $k$ is 10, it gets surpassed for both $mk + r$ and $mk$ methods and when the number of clusters is 50, by $mk$. Both $mk + r$ and $mk$ methods presented the best results with the number of clusters being 10.

Looking at Figure 5.1b, which shows the results of the heuristic compared to those obtained when using 50% of dataset for training, $mk + r$ was the worst method. This method, and $mk$, had the best results with 20 clusters, and most of the times, they had worst results than the random sampling, $r$ method. In Figure 5.1c, the $mk + r$ method presented the best CSS value, with 5 clusters, and right after, the $mk$, with 20 clusters. The best CSS values, for all the training set sizes, are quite similar ranging from 12.61% to 15.69%.

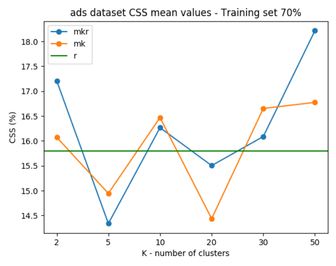
For this dataset, it is possible to observe the effect that $k$ has on the results. The results of the methods that perform clustering, $mk + r$ and $mk$, present a large variation depending on the value of $k$. This shows that for this dataset, to obtain the best results with these two methods, the number of clusters is an important parameter.



(A) Training set - 30%



(B) Training set - 50%

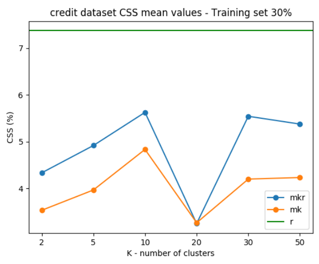


(C) Training set - 70%

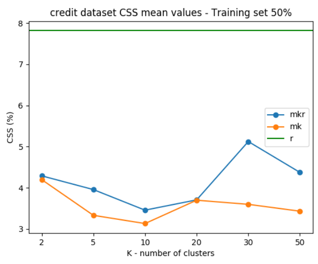
Figure 5.1: Results of Ads dataset

The results for the Credit dataset are noticeable, with Figs. 5.2a to 5.2c showing really low values for the CSS. This dataset is a binary problem and this may be one of the reasons for such. Another interesting aspect is that both $mk + r$ and $mk$ sampling methods, for all values of $k$, always obtains better results than $r$, random sampling. It shows that clustering was able to discover a good data structure.

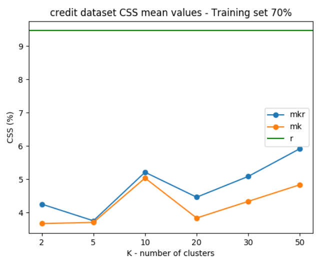
Figs. 5.2a to 5.2c are quite conclusive, clearly showing that the $mk+r$ and $mk$ methods are a better alternative to the $r$ method, with $mk$ being the best. Further, results for these two sampling methods are consistent for all values of $k$. The greatest variation occurred in the results is visible in Figure 5.2a. These results demonstrate the positive effect of using clustering, and as can be seen, the value of $k$ does not cause big differences in the CSS value. Once again, the best CSS values, for all the training set sizes, are very close ranging from 3.13% to 3.67%.



(A) Training set - 30%



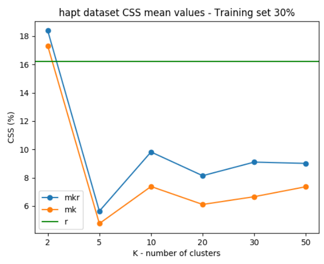
(B) Training set - 50%



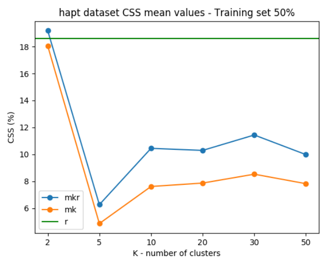
(C) Training set - 70%

FIGURE 5.2: Results of Credit dataset

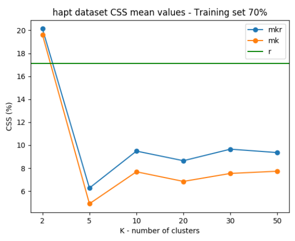
In Figs. 5.3a to 5.3c, the results of $mk + r$ and $mk$ sampling methods are impressive. Comparing them to random sampling, they can achieve CSS values, 10% smaller. This reinforces the positive aspect of clustering. In addition it is clear that the best results for CSS are obtained with 5 clusters. And the only value of $k$ in which the methods $mk+r$ and $mk$ are worse than random sampling is 2. Once more, $mk$ proved to be the best sampling approach.
The best CSS values for the Hapt dataset are comprised between 4.76% and 4.90%.



(A) Training set - 30%



(B) Training set - 50%

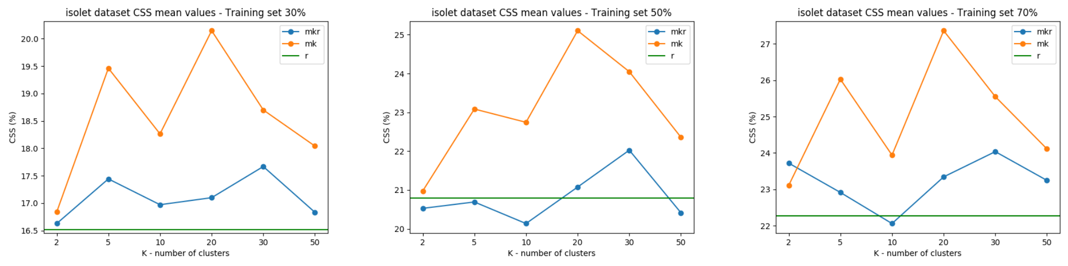


(C) Training set - 70%

FIGURE 5.3: Results of Hapt dataset

For the Isolet dataset, $mk$ is clearly the worst method. As can be seen in Figs. 5.4a to 5.4c, most of the times as the value of $k$ increases, so does the CSS. In addition, it is the method that has the greatest variation for the CSS value. The worst results always occurs when the number of clusters is 5 and 20, and the best when it is 2.

The results obtained by the $mk+$ method are closer to those obtained by random sampling. This is due to the additional step that supplements the sample with random examples. Looking at the results, randomness is a good approach for the Isolet dataset. This dataset has a high number of classes, 26, and for each one has 300 examples. Because of this, randomness is able to easily select examples from all classes. This shows that for datasets with few samples per class, random sampling may be the best option. Despite all this, in Figs. 5.4b and 5.4c, $mk+r$ still managed to overcome random sampling, accomplishing the best result when the number of clusters was 10.



(A) Training set - 30% (B) Training set - 50% (C) Training set - 70%

FIGURE 5.4: Results of Isolet dataset

For the $mk + r$ and $mk$ sampling methods, the best results were always obtained with 20 clusters or less. The number of cluster seems to do not need to grow larger than that to get the best results. However, this number may increase for even larger datasets.

By inspecting the best results of each dataset, their number of classes proved not to be the best approach to define $k$. As seen in the results, this parameter is quite important. Its value varies for each dataset and therefore an analysis should always be made to determine the best value.

Table 5.2 summarizes the best values that $mk + r$ and $mk$ obtained for each dataset. In addition, the most effective method for selecting examples from clusters is presented.

The best method was always the *rand* except for the Credit dataset, which was the *decr*. But even in this dataset, the difference between *decr* and *rand* is minimal, so *rand* can be considered the best approach. Looking at Tables A.10 to A.13 it is possible to see that *rand* was always the one producing the best results. Furthermore, its statistical difference with the other two methods is high. The only exception, as said before, was for the Credit dataset, where *rand* and *decr* had no statistical difference.

Another important aspect to point out is the fact that *asc* often failed to discover a CSS, or when it was able to, it was the worst value comparing to the other methods. The only exception for this, was in the Hapt dataset when using more than 5 clusters. For those cases, *decr* was the worst.

Table 5.2: Best results for each test case.

| | **Training Set Size - 30%** | | | **Training Set Size - 50%** | | | **Training Set Size - 70%** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Sampling Method** | **Best Results** | **k** | **Sampling Method** | **Best Results** | **k** | **Sampling Method** | **Best Results** | **k** |
| **Ads** | mk+r (rand) | 12.61 ± (3.40) | 10 | mk (rand) | 15.98 ± (5.68) | 20 | mk+r (rand) | 14.34 ± (6.25) | 20 |
| | mk (rand) | 13.09 ± (3.60) | 10 | mk+r (rand) | 16.82 ± (6.18) | 20 | mk (rand) | 14.44 ± (5.46) | 20 |
| **Credit** | mk+r (decr) | 3.25 ± (2.47) | 20 | mk (decr) | 3.13 ± (2.03) | 10 | mk (decr) | 3.67 ± (4.30) | 2 |
| | mk (decr) | 3.27 ± (3.04) | 20 | mk+r (decr) | 3.46 ± (2.24) | 10 | mk+r (decr) | 3.75 ± (1.80) | 5 |
| **Hapt** | mk (rand) | 4.76 ± (0.87) | 5 | mk (rand) | 4.86 ± (0.99) | 5 | mk (rand) | 4.90 ± (1.11) | 5 |
| | mk+r (rand) | 5.64 ± (1.03) | 5 | mk+r (rand) | 6.27 ± (1.61) | 5 | mk+r (rand) | 6.27 ± (1.74) | 5 |
| **Isolet** | mk+r (rand) | 16.63 ± (1.62) | 2 | mk+r (rand) | 20.14 ± (2.39) | 10 | mk+r (rand) | 22.06 ± (2.57) | 10 |
| | mk (rand) | 16.84 ± (1.78) | 2 | mk (rand) | 20.97 ± (2.37) | 2 | mk (rand) | 23.11 ± (3.01) | 2 |

## 5.3 Concluding remarks

In the results, the effect of $k$ value was studied and it has a notorious impact on the CSS, mainly on the Credit and Hapt datasets. It is possible to see the improvement of $mk + r$ and $mk$ sampling methods compared to the random sampling, the $r$ method. However, this is not so visible for the Ads and Isolet dataset. In the latter case, this can be due to the fact, as previous explained, that the dataset is perfectly balanced, that is, each class has the same number of examples, which allows the random sampling to obtain good results. The Hapt dataset is also well balanced, however, it has a higher number of examples per class. As said before, random sampling seems to be a good choice when dealing with datasets that contains low examples per class.

As the heuristic starts by having very small training sets, $D_v$, the training process is fast. Using this incremental approach, it is easier and faster to find out the CSS of a specific dataset. And by looking at results it is possible to conclude that the $mk+r$ and $mk$ sampling methods are a good alternative to the random sampling. As for those two sampling methods, the best shows up to be $mk$ since it presents more consistent results in the first 3 datasets.

Despite all this, the reduction for all datasets is noticeable, demonstrating that when the learning machines were trained with 30%, 50% or 70% of the dataset it was unnecessary. This shows that most of the data present in these datasets may be redundant, or even irrelevant. And by analyzing the results for the 3 training set sizes of all datasets, they show that the CSS value does not change significantly. Where it is noted a greater difference is in the Isolet dataset.

As it was intended to show, all the datasets present a CSS, and it was possible to discover it by means of a simple heuristic. In addition, the CSS was shown to be independent of the way the dataset was divided. Therefore, for future work, this heuristic can be a good approach for reducing datasets before using them for data mining tasks.

# Chapter 6

# Conclusions

In this dissertation an empirical study on four datasets, to assess if datasets possess a Critical Sampling Size, was conducted. Its existence could contribute to data mining tasks, by improving their speed while maintaining the desirable performance. In data mining, big datasets are used, and building good models for knowledge discovery could be very expensive (i.e. resources and time consuming) because the size of the used training sets. Finding the CSS of big datasets may reduce the impact of that issue on data mining.

It has been proved that this problem is of the NP-hard type and therefore, discovering its optimal solution is intractable. Therefore, a heuristic approach was proposed and more than one sampling method was used to analyze their effect on the results.
The experimental results show the existence of an apparent CSS for each dataset; It is "apparent" since finding the exact CSS is highly intractable, hence whatever sampling generated by the heuristic methods would be an approximated CSS at best. In addition, the CSS of the datasets are demonstrated to be considerably smaller than their initial size. These results seem to validate our assumption that since it is possible to find the (apparent) CFD of a dataset using simple heuristic methods, it is likely that such methods will prove practicable in finding the dataset's (apparent) CSS.

Rather than training a model in the traditional way, by randomly selecting a percentage of the dataset, the proposed heuristic method produced satisfactory results showing that it can significantly reduce the size of the training set, while maintaining the same performance. With these results it proved to be a good alternative to random sampling and a simple and efficient method to discover the CSS of a dataset.

## 6.1 Future work

There is always room for improvement. Many aspects of the proposed heuristic deserve further study. Studying more values for the different parameters can provide better conclusions, and the same applies to datasets. Future work can begin by studying more datasets, with even larger dimensions. After that, combining both heuristics to find CFD and CSS, can lead to results that can further reduce datasets.
Finally, other studies can be done in order to compare this heuristic with state-of-the-art methods, such as SVM optimization.

# Appendix A

# Result tables

## A.1 Experimental Results

The following tables present the results for each dataset. Each pair of tables refers to a dataset. The tables show the averaged CSS values plus standard deviation for each test case. Blue values indicate the best sampling method for a specific $k$ value, and training set size. If $mk + r$ or $mk$ is the best method then, the blue value indicates the best way to select the examples from clusters.

There are some occasions where the standard deviation is 0. Those cases happened when the heuristic failed to find the CSS and the maximum sample size was reached. This is visible for th Ads, Hapt and Isolet datasets.

TABLE A.1: Ads dataset. CSS results for k = 2, 5, 10. Each cell display the average ± (standard deviation). Values are in percentage.

| K:2 | | | | |
|---|---|---|---|---|
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 15.07 ± (6.02) | 15.14 ± (5.79) | 13.93 ± (4.91) |
| | asc | 27.67 ± (2.62) | 28.10 ± (0.83) | |
| | decr | 17.85 ± (6.29) | 18.01 ± (5.58) | |
| Training Set: 50% | rand | 17.55 ± (7.44) | 19.53 ± (7.07) | 17.87 ± (5.93) |
| | asc | 28.48 ± (3.38) | 33.46 ± (7.70) | |
| | decr | 31.89 ± (14.50) | 28.20 ± (12.73) | |
| Training Set: 70% | rand | 17.21 ± (5.54) | 16.07 ± (5.14) | 15.69 ± (7.54) |
| | asc | 29.50 ± (6.69) | 32.90 ± (7.84) | |
| | decr | 20.62 ± (11.65) | 21.77 ± (12.63) | |
| K:5 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 13.90 ± (6.10) | 13.77 ± (6.43) | 13.77 ± (4.28) |
| | asc | 30.16 ± (0.00) | 30.95 ± (0.00) | |
| | decr | 20.24 ± (6.11) | 18.82 ± (5.13) | |
| Training Set: 50% | rand | 16.96 ± (5.68) | 16.86 ± (5.77) | 15.69 ± (5.41) |
| | asc | 47.38 ± (4.69) | 44.33 ± (4.46) | |
| | decr | 24.13 ± (7.53) | 22.95 ± (6.20) | |
| Training Set: 70% | rand | 14.34 ± (6.25) | 14.94 ± (7.73) | 14.80 ± (5.65) |
| | asc | 46.20 ± (7.26) | 42.09 ± (4.70) | |
| | decr | 21.86 ± (7.65) | 20.96 ± (7.34) | |
| K:10 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 12.61 ± (3.40) | 13.09 ± (3.60) | 13.46 ± (3.94) |
| | asc | 24.75 ± (4.52) | 23.50 ± (5.72) | |
| | decr | 20.59 ± (5.70) | 18.62 ± (6.72) | |
| Training Set: 50% | rand | 20.74 ± (6.81) | 17.85 ± (5.93) | 16.35 ± (4.82) |
| | asc | 32.12 ± (10.29) | 28.95 ± (6.78) | |
| | decr | 24.04 ± (8.87) | 22.20 ± (7.21) | |
| Training Set: 70% | rand | 16.27 ± (7.78) | 16.47 ± (7.47) | 15.17 ± (6.52) |
| | asc | 28.41 ± (10.65) | 25.74 ± (9.71) | |
| | decr | 22.87 ± (9.11) | 22.08 ± (8.52) | |

TABLE A.2: Ads dataset. CSS results for k = 20, 30, 50. Each cell display the average ± (standard deviation). Values are in percentage.

| K:20 | | | | |
|---|---|---|---|---|
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 14.38 ± (3.69) | 14.80 ± (4.47) | 13.46 ± (3.53) |
| | asc | 27.24 ± (5.08) | 24.89 ± (5.79) | |
| | decr | 19.87 ± (6.44) | 17.08 ± (5.70) | |
| Training Set: 50% | rand | 16.82 ± (6.18) | 15.98 ± (5.68) | 17.36 ± (5.13) |
| | asc | 34.46 ± (9.25) | 32.73 ± (7.08) | |
| | decr | 23.38 ± (9.55) | 22.53 ± (8.54) | |
| Training Set: 70% | rand | 15.50 ± (5.80) | 14.44 ± (5.46) | 14.96 ± (5.33) |
| | asc | 30.34 ± (9.84) | 26.72 ± (10.40) | |
| | decr | 20.03 ± (10.24) | 20.33 ± (9.55) | |
| K:30 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 15.32 ± (6.45) | 14.70 ± (6.49) | 14.21 ± (3.80) |
| | asc | 25.08 ± (6.62) | 22.02 ± (7.41) | |
| | decr | 18.07 ± (6.05) | 18.30 ± (5.35) | |
| Training Set: 50% | rand | 19.21 ± (7.78) | 19.64 ± (8.61) | 18.60 ± (5.02) |
| | asc | 35.61 ± (9.66) | 34.03 ± (10.08) | |
| | decr | 24.32 ± (8.11) | 21.23 ± (6.63) | |
| Training Set: 70% | rand | 16.09 ± (5.52) | 16.65 ± (8.19) | 16.59 ± (5.31) |
| | asc | 29.96 ± (10.77) | 29.83 ± (10.95) | |
| | decr | 22.72 ± (6.03) | 21.53 ± (7.06) | |
| K:50 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 14.75 ± (4.59) | 13.42 ± (4.91) | 13.88 ± (4.54) |
| | asc | 22.06 ± (6.03) | 23.53 ± (5.96) | |
| | decr | 18.85 ± (5.24) | 19.21 ± (5.28) | |
| Training Set: 50% | rand | 17.40 ± (8.55) | 16.88 ± (6.05) | 16.88 ± (7.14) |
| | asc | 29.75 ± (9.65) | 30.85 ± (10.33) | |
| | decr | 23.45 ± (8.75) | 20.28 ± (6.11) | |
| Training Set: 70% | rand | 18.21 ± (7.41) | 16.77 ± (5.98) | 17.59 ± (6.52) |
| | asc | 28.11 ± (10.61) | 29.07 ± (12.17) | |
| | decr | 22.44 ± (9.55) | 22.01 ± (10.06) | |

TABLE A.3: Credit dataset. CSS results for k = 2, 5, 10. Each cell display the average ± (standard deviation). Values are in percentage.

| K:2 | | | | |
|---|---|---|---|---|
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 4.33 ± (2.60) | 4.20 ± (2.80) | 8.03 ± (6.40) |
| | asc | 19.25 ± (5.29) | 16.03 ± (6.04) | |
| | decr | 4.67 ± (3.50) | 3.53 ± (3.27) | |
| Training Set: 50% | rand | 4.29 ± (2.22) | 4.33 ± (2.47) | 8.10 ± (9.33) |
| | asc | 18.42 ± (3.01) | 16.73 ± (5.34) | |
| | decr | 5.12 ± (8.82) | 4.20 ± (7.17) | |
| Training Set: 70% | rand | 4.25 ± (2.78) | 3.73 ± (2.33) | 8.03 ± (13.25) |
| | asc | 17.46 ± (6.19) | 14.53 ± (6.57) | |
| | decr | 4.67 ± (6.72) | 3.67 ± (4.30) | |
| K:5 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 6.08 ± (5.37) | 5.23 ± (2.73) | 7.97 ± (5.32) |
| | asc | 11.25 ± (2.15) | 9.90 ± (0.40) | |
| | decr | 4.92 ± (3.95) | 3.97 ± (2.40) | |
| Training Set: 50% | rand | 4.62 ± (1.92) | 4.37 ± (3.26) | 6.50 ± (7.34) |
| | asc | 9.79 ± (4.03) | 8.97 ± (1.19) | |
| | decr | 3.96 ± (3.20) | 3.33 ± (2.47) | |
| Training Set: 70% | rand | 4.00 ± (2.16) | 3.97 ± (2.33) | 15.13 ± (22.26) |
| | asc | 9.42 ± (4.73) | 9.77 ± (3.63) | |
| | decr | 3.75 ± (1.80) | 3.70 ± (2.10) | |
| K:10 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 5.62 ± (3.06) | 5.13 ± (3.00) | 7.73 ± (4.49) |
| | asc | 7.71 ± (2.10) | 7.17 ± (2.23) | |
| | decr | 6.67 ± (6.62) | 4.83 ± (3.54) | |
| Training Set: 50% | rand | 5.21 ± (3.06) | 4.57 ± (2.50) | 11.67 ± (12.61) |
| | asc | 7.25 ± (2.73) | 6.73 ± (1.87) | |
| | decr | 3.46 ± (2.24) | 3.13 ± (2.03) | |
| Training Set: 70% | rand | 7.50 ± (12.26) | 5.03 ± (5.11) | 6.83 ± (9.97) |
| | asc | 8.46 ± (3.42) | 7.17 ± (1.86) | |
| | decr | 5.21 ± (5.24) | 6.43 ± (12.46) | |

TABLE A.4: Credit dataset. CSS results for k = 20, 30, 50. Each cell display the average ± (standard deviation). Values are in percentage.

| K:20 | | | | |
|---|---|---|---|---|
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 5.62 ± (5.92) | 3.67 ± (1.86) | 8.63 ± (5.58) |
| | asc | 5.08 ± (1.99) | 4.97 ± (1.71) | |
| | decr | 3.25 ± (2.47) | 3.27 ± (3.04) | |
| Training Set: 50% | rand | 4.04 ± (2.09) | 4.17 ± (2.15) | 7.30 ± (8.12) |
| | asc | 4.62 ± (2.11) | 4.57 ± (1.91) | |
| | decr | 3.71 ± (3.42) | 3.70 ± (4.15) | |
| Training Set: 70% | rand | 5.58 ± (6.03) | 5.57 ± (6.78) | 6.43 ± (4.95) |
| | asc | 6.83 ± (3.98) | 5.67 ± (3.30) | |
| | decr | 4.46 ± (4.04) | 3.83 ± (3.26) | |
| K:30 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 5.54 ± (3.99) | 4.20 ± (2.46) | 6.00 ± (3.22) |
| | asc | 5.79 ± (2.66) | 4.97 ± (2.34) | |
| | decr | 5.88 ± (5.24) | 5.70 ± (5.42) | |
| Training Set: 50% | rand | 5.46 ± (4.59) | 4.50 ± (3.55) | 7.87 ± (9.23) |
| | asc | 5.50 ± (2.93) | 4.97 ± (2.71) | |
| | decr | 5.12 ± (4.35) | 3.60 ± (2.76) | |
| Training Set: 70% | rand | 5.96 ± (4.75) | 4.97 ± (4.84) | 11.53 ± (17.70) |
| | asc | 5.08 ± (3.18) | 4.33 ± (2.60) | |
| | decr | 5.08 ± (5.62) | 5.43 ± (7.77) | |
| K:50 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 5.38 ± (3.72) | 5.13 ± (3.96) | 5.87 ± (5.59) |
| | asc | 5.50 ± (1.90) | 4.43 ± (1.19) | |
| | decr | 6.12 ± (5.88) | 4.23 ± (3.92) | |
| Training Set: 50% | rand | 5.17 ± (4.78) | 4.43 ± (4.05) | 5.47 ± (2.99) |
| | asc | 6.04 ± (2.39) | 5.43 ± (3.31) | |
| | decr | 4.38 ± (4.73) | 3.43 ± (2.51) | |
| Training Set: 70% | rand | 7.08 ± (8.03) | 5.80 ± (8.29) | 8.80 ± (11.32) |
| | asc | 5.92 ± (2.99) | 4.83 ± (2.23) | |
| | decr | 8.21 ± (13.20) | 5.37 ± (6.27) | |

TABLE A.5: Hapt dataset. CSS results for k = 2, 5, 10. Each cell display the average ± (standard deviation). Values are in percentage.

| K:2 | | | | |
|---|---|---|---|---|
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 18.39 ± (6.64) | 17.31 ± (6.51) | 15.84 ± (6.22) |
| | asc | 30.09 ± (0.00) | 30.19 ± (0.00) | |
| | decr | 24.57 ± (4.90) | 25.20 ± (2.89) | |
| Training Set: 50% | rand | 19.18 ± (5.66) | 18.05 ± (5.18) | 18.12 ± (5.92) |
| | asc | 50.14 ± (0.00) | 50.32 ± (0.00) | |
| | decr | 30.25 ± (6.70) | 27.65 ± (4.44) | |
| Training Set: 70% | rand | 20.14 ± (7.56) | 19.63 ± (8.46) | 15.90 ± (5.82) |
| | asc | 70.20 ± (0.00) | 70.45 ± (0.00) | |
| | decr | 30.13 ± (7.52) | 27.91 ± (5.03) | |
| K:5 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 5.64 ± (1.03) | 4.76 ± (0.87) | 17.21 ± (5.84) |
| | asc | 8.65 ± (1.56) | 7.41 ± (0.81) | |
| | decr | 24.40 ± (4.33) | 27.68 ± (1.71) | |
| Training Set: 50% | rand | 6.27 ± (1.61) | 4.86 ± (0.99) | 19.39 ± (6.61) |
| | asc | 8.77 ± (2.30) | 8.42 ± (0.81) | |
| | decr | 28.33 ± (7.42) | 30.70 ± (5.18) | |
| Training Set: 70% | rand | 6.27 ± (1.74) | 4.90 ± (1.11) | 16.10 ± (4.89) |
| | asc | 9.03 ± (2.49) | 7.72 ± (1.36) | |
| | decr | 25.61 ± (5.21) | 28.82 ± (3.18) | |
| K:10 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 9.82 ± (3.11) | 7.38 ± (2.34) | 15.40 ± (4.89) |
| | asc | 16.42 ± (5.18) | 15.23 ± (1.11) | |
| | decr | 23.52 ± (4.62) | 26.20 ± (3.24) | |
| Training Set: 50% | rand | 10.45 ± (2.14) | 7.62 ± (1.92) | 19.79 ± (7.03) |
| | asc | 19.30 ± (3.20) | 16.47 ± (1.53) | |
| | decr | 25.66 ± (6.22) | 28.45 ± (5.35) | |
| Training Set: 70% | rand | 9.49 ± (2.61) | 7.68 ± (1.70) | 17.14 ± (4.42) |
| | asc | 17.63 ± (4.51) | 15.80 ± (1.40) | |
| | decr | 25.95 ± (6.63) | 27.98 ± (5.11) | |

TABLE A.6: Hapt dataset. CSS results for k = 20, 30, 50. Each cell display the average ± (standard deviation). Values are in percentage.

| K:20 | | | | |
|---|---|---|---|---|
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 8.14 ± (2.67) | 6.11 ± (1.60) | 14.64 ± (4.71) |
| | asc | 23.06 ± (5.20) | 19.40 ± (1.36) | |
| | decr | 24.75 ± (4.62) | 26.94 ± (3.62) | |
| Training Set: 50% | rand | 10.29 ± (2.76) | 7.87 ± (2.18) | 17.86 ± (7.90) |
| | asc | 24.16 ± (5.14) | 18.74 ± (2.87) | |
| | decr | 29.60 ± (6.33) | 29.21 ± (5.20) | |
| Training Set: 70% | rand | 8.65 ± (2.70) | 6.84 ± (2.28) | 17.64 ± (7.36) |
| | asc | 25.53 ± (6.72) | 20.42 ± (3.64) | |
| | decr | 26.95 ± (5.89) | 28.18 ± (5.12) | |
| K:30 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 9.10 ± (3.43) | 6.66 ± (1.87) | 16.98 ± (4.53) |
| | asc | 16.42 ± (10.48) | 15.77 ± (10.91) | |
| | decr | 25.57 ± (3.99) | 26.50 ± (3.53) | |
| Training Set: 50% | rand | 11.44 ± (6.38) | 8.53 ± (3.61) | 18.85 ± (6.71) |
| | asc | 20.08 ± (15.16) | 16.03 ± (10.71) | |
| | decr | 29.42 ± (7.67) | 27.93 ± (5.12) | |
| Training Set: 70% | rand | 9.65 ± (3.63) | 7.54 ± (2.76) | 16.40 ± (5.74) |
| | asc | 19.67 ± (14.20) | 15.70 ± (11.02) | |
| | decr | 26.21 ± (7.97) | 28.51 ± (5.87) | |
| K:50 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 9.30 ± (2.23) | 7.37 ± (1.92) | 17.11 ± (4.37) |
| | asc | 9.01 ± (3.89) | 8.28 ± (2.32) | |
| | decr | 28.23 ± (3.01) | 26.90 ± (3.07) | |
| Training Set: 50% | rand | 9.98 ± (2.51) | 7.82 ± (2.01) | 17.57 ± (6.59) |
| | asc | 10.78 ± (3.37) | 8.74 ± (2.24) | |
| | decr | 30.11 ± (5.99) | 28.78 ± (4.57) | |
| Training Set: 70% | rand | 9.75 ± (2.51) | 7.73 ± (2.09) | 19.49 ± (8.09) |
| | asc | 9.35 ± (4.44) | 8.19 ± (3.37) | |
| | decr | 28.00 ± (4.93) | 26.72 ± (5.19) | |

TABLE A.7: Isolet dataset. CSS results for k = 2, 5, 10. Each cell display the average ± (standard deviation). Values are in percentage.

| K:2 | | | | |
|---|---|---|---|---|
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 16.63 ± (1.62) | 16.84 ± (1.78) | 16.67 ± (2.04) |
| | asc | 31.10 ± (0.00) | 30.01 ± (0.00) | |
| | decr | 31.10 ± (0.00) | 30.01 ± (0.00) | |
| Training Set: 50% | rand | 20.53 ± (2.71) | 20.97 ± (2.37) | 20.57 ± (2.03) |
| | asc | 51.01 ± (0.00) | 50.02 ± (0.00) | |
| | decr | 44.41 ± (1.43) | 44.48 ± (1.57) | |
| Training Set: 70% | rand | 23.72 ± (3.53) | 23.11 ± (3.01) | 23.04 ± (3.46) |
| | asc | 67.39 ± (4.62) | 67.86 ± (1.29) | |
| | decr | 45.37 ± (1.90) | 45.25 ± (1.41) | |
| K:5 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 17.44 ± (1.49) | 19.46 ± (1.67) | 16.76 ± (2.02) |
| | asc | 30.78 ± (0.00) | 30.78 ± (0.00) | |
| | decr | 28.56 ± (3.09) | 30.71 ± (0.26) | |
| Training Set: 50% | rand | 20.69 ± (2.73) | 23.09 ± (2.68) | 20.25 ± (2.54) |
| | asc | 50.02 ± (0.00) | 50.28 ± (0.00) | |
| | decr | 34.97 ± (5.95) | 34.82 ± (1.68) | |
| Training Set: 70% | rand | 22.91 ± (3.03) | 26.03 ± (3.24) | 22.47 ± (3.03) |
| | asc | 67.97 ± (2.76) | 66.90 ± (3.80) | |
| | decr | 33.13 ± (4.62) | 35.84 ± (2.09) | |
| K:10 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 16.97 ± (1.95) | 18.26 ± (2.23) | 16.83 ± (1.47) |
| | asc | 30.78 ± (0.00) | 30.78 ± (0.00) | |
| | decr | 28.00 ± (3.11) | 30.78 ± (0.00) | |
| Training Set: 50% | rand | 20.14 ± (2.39) | 22.74 ± (2.47) | 20.55 ± (2.46) |
| | asc | 49.72 ± (1.05) | 50.28 ± (0.00) | |
| | decr | 32.06 ± (2.63) | 39.13 ± (1.80) | |
| Training Set: 70% | rand | 22.06 ± (2.57) | 23.94 ± (3.26) | 22.09 ± (2.69) |
| | asc | 60.92 ± (5.06) | 69.43 ± (2.98) | |
| | decr | 32.88 ± (3.41) | 39.50 ± (2.87) | |

TABLE A.8: Isolet dataset. CSS results for k = 20, 30, 50. Each cell display the average ± (standard deviation). Values are in percentage.

| K:20 | | | | |
|---|---|---|---|---|
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 17.10 ± (2.11) | 20.14 ± (2.40) | 16.31 ± (1.85) |
| | asc | 30.78 ± (0.00) | 30.78 ± (0.00) | |
| | decr | 26.98 ± (2.62) | 30.68 ± (0.31) | |
| Training Set: 50% | rand | 21.08 ± (2.56) | 25.10 ± (3.33) | 20.55 ± (2.43) |
| | asc | 49.76 ± (1.19) | 50.28 ± (0.00) | |
| | decr | 30.23 ± (2.54) | 33.72 ± (2.08) | |
| Training Set: 70% | rand | 23.34 ± (2.32) | 27.36 ± (3.25) | 21.89 ± (2.40) |
| | asc | 56.86 ± (6.07) | 62.90 ± (4.24) | |
| | decr | 32.23 ± (3.19) | 35.30 ± (2.61) | |
| K:30 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 17.67 ± (1.31) | 18.70 ± (2.34) | 15.89 ± (1.71) |
| | asc | 30.17 ± (0.00) | 30.01 ± (0.00) | |
| | decr | 27.24 ± (2.63) | 30.01 ± (0.00) | |
| Training Set: 50% | rand | 22.03 ± (3.00) | 24.05 ± (3.55) | 21.62 ± (2.31) |
| | asc | 48.98 ± (2.04) | 50.52 ± (0.72) | |
| | decr | 30.79 ± (1.95) | 36.44 ± (2.28) | |
| Training Set: 70% | rand | 24.04 ± (3.52) | 25.55 ± (3.25) | 22.16 ± (2.85) |
| | asc | 53.24 ± (5.44) | 56.44 ± (4.80) | |
| | decr | 32.08 ± (2.60) | 37.17 ± (1.95) | |
| K:50 | | | | |
| Sampling Method | | mk+r | mk | r |
| Training Set: 30% | rand | 16.83 ± (1.82) | 18.04 ± (1.93) | 16.67 ± (1.62) |
| | asc | 30.46 ± (0.00) | 30.70 ± (0.47) | |
| | decr | 26.35 ± (2.17) | 30.61 ± (0.56) | |
| Training Set: 50% | rand | 20.41 ± (2.69) | 22.36 ± (2.40) | 21.20 ± (2.45) |
| | asc | 43.93 ± (5.13) | 48.27 ± (2.83) | |
| | decr | 29.98 ± (3.10) | 34.80 ± (2.56) | |
| Training Set: 70% | rand | 23.25 ± (3.41) | 24.11 ± (3.36) | 22.06 ± (3.00) |
| | asc | 47.72 ± (7.60) | 51.39 ± (6.38) | |
| | decr | 30.99 ± (3.30) | 36.17 ± (2.41) | |

## A.2 Statistical Analysis

To make the statistical analysis of the results, all the test cases were executed 30 times. To infer the existence of statistical difference between the methods that select the examples from the clusters (*rand*, *asc*, *decr*) it was made the Friedman non-parametric test, at a 0.05 level of significance. If that happened, then the Wilcoxon non-parametric test was done, to assess the difference for each pair of those methods. And finally, it was calculated the effect size of those differences [29].
Tables A.10 to A.13 present this information in more detail.

To compare the methods referred above when performing the statistical analysis, the hypothesis tested was:

- $H_0$: $m1 = m2$ - the means of the two methods were equal.
- $H_1$: $m1 \neq m2$ - the means of the the two methods were different.

To test these hypothesis, when conducting the statistical tests, looking at the $p$-value, level of significance, if it was smaller then 0.05 then the null hypothesis, $H_0$, could be rejected and therefore accept $H_1$.

The statistical results are reported using a graphical notation, +, - and $\sim$. The symbols indicate the quality of the method on the left compared to the one on the right. The amount of + or - symbols depends on the effect size between the two methods. The $\sim$ indicates that no statistical differences between the methods were found. The + sign means that the left method was better that the one on the right, and - the opposite. The effect sizes were defined as shown in Table A.9.

Table A.9: Effect Size definition

| **Effect Size** | ***r*** |
|---|---|
| Small | 0.1 |
| Medium | 0.3 |
| Large | 0.5 |

Table A.10: Ads dataset statistical results. Each cell display the statistical difference between the two methods being compared.

| K:2 | | | | | K:20 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | ∼ | — | 30% | mk+r | +++ | +++ | — |
| | mk | +++ | ∼ | — | | mk | +++ | ∼ | — |
| 50% | mk+r | +++ | +++ | ∼ | 50% | mk+r | +++ | +++ | — |
| | mk | +++ | ∼ | ∼ | | mk | +++ | +++ | — |
| 70% | mk+r | +++ | ∼ | — | 70% | mk+r | +++ | ∼ | — |
| | mk | +++ | ∼ | — | | mk | +++ | ++ | – |
| K:5 | | | | | K:30 | | | | |
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | +++ | — | 30% | mk+r | +++ | ∼ | — |
| | mk | +++ | +++ | — | | mk | +++ | +++ | ∼ |
| 50% | mk+r | +++ | +++ | — | 50% | mk+r | +++ | ∼ | — |
| | mk | +++ | +++ | — | | mk | +++ | ∼ | — |
| 70% | mk+r | +++ | +++ | — | 70% | mk+r | +++ | +++ | — |
| | mk | +++ | +++ | — | | mk | +++ | +++ | — |
| K:10 | | | | | K:50 | | | | |
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | +++ | – | 30% | mk+r | +++ | +++ | ∼ |
| | mk | +++ | +++ | — | | mk | +++ | +++ | – |
| 50% | mk+r | +++ | ∼ | — | 50% | mk+r | +++ | ++ | – |
| | mk | +++ | ++ | — | | mk | +++ | ∼ | — |
| 70% | mk+r | +++ | +++ | ∼ | 70% | mk+r | +++ | ∼ | ∼ |
| | mk | +++ | ++ | ∼ | | mk | +++ | ∼ | ∼ |

Table A.11: Credit dataset statistical results. Each cell display the statistical difference between the two methods being compared.

| K:2 | | | | | K:20 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | ~ | — | 30% | mk+r | ~ | – | — |
| | mk | +++ | ~ | — | | mk | +++ | ~ | — |
| 50% | mk+r | +++ | ~ | — | 50% | mk+r | ~ | ~ | – |
| | mk | +++ | — | — | | mk | ~ | – | — |
| 70% | mk+r | +++ | ~ | — | 70% | mk+r | ++ | ~ | — |
| | mk | +++ | ~ | — | | mk | ~ | ~ | — |
| K:5 | | | | | K:30 | | | | |
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | ~ | — | 30% | mk+r | ~ | ~ | ~ |
| | mk | +++ | ~ | — | | mk | ~ | ~ | ~ |
| 50% | mk+r | +++ | ~ | — | 50% | mk+r | ~ | ~ | ~ |
| | mk | +++ | ~ | — | | mk | ~ | ~ | — |
| 70% | mk+r | +++ | ~ | — | 70% | mk+r | ~ | ~ | ~ |
| | mk | +++ | ~ | — | | mk | ~ | ~ | ~ |
| K:10 | | | | | K:50 | | | | |
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | ~ | — | 30% | mk+r | ~ | ~ | ~ |
| | mk | +++ | ~ | — | | mk | ~ | ~ | ~ |
| 50% | mk+r | ++ | – | — | 50% | mk+r | ~ | ~ | — |
| | mk | +++ | ~ | — | | mk | ~ | ~ | — |
| 70% | mk+r | +++ | ~ | — | 70% | mk+r | ~ | ~ | ~ |
| | mk | +++ | ~ | — | | mk | ~ | ~ | ~ |

TABLE A.12: Hapt dataset statistical results. Each cell display the statistical difference between the two methods being compared.

| K:2 | | | | | K:20 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | +++ | — | 30% | mk+r | +++ | +++ | ∼ |
| | mk | +++ | +++ | — | | mk | +++ | +++ | +++ |
| 50% | mk+r | +++ | +++ | — | 50% | mk+r | +++ | +++ | +++ |
| | mk | +++ | +++ | — | | mk | +++ | +++ | +++ |
| 70% | mk+r | +++ | +++ | — | 70% | mk+r | +++ | +++ | ∼ |
| | mk | +++ | +++ | — | | mk | +++ | +++ | +++ |
| K:5 | | | | | K:30 | | | | |
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | +++ | +++ | 30% | mk+r | ++ | +++ | ++ |
| | mk | +++ | +++ | +++ | | mk | +++ | +++ | +++ |
| 50% | mk+r | +++ | +++ | +++ | 50% | mk+r | ∼ | +++ | ++ |
| | mk | +++ | +++ | +++ | | mk | +++ | +++ | +++ |
| 70% | mk+r | +++ | +++ | +++ | 70% | mk+r | ++ | +++ | ++ |
| | mk | +++ | +++ | +++ | | mk | +++ | +++ | +++ |
| K:10 | | | | | K:50 | | | | |
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | +++ | +++ | 30% | mk+r | ∼ | +++ | +++ |
| | mk | +++ | +++ | +++ | | mk | ∼ | +++ | +++ |
| 50% | mk+r | +++ | +++ | +++ | 50% | mk+r | ∼ | +++ | +++ |
| | mk | +++ | +++ | +++ | | mk | ∼ | +++ | +++ |
| 70% | mk+r | +++ | +++ | +++ | 70% | mk+r | ∼ | +++ | +++ |
| | mk | +++ | +++ | +++ | | mk | ∼ | +++ | +++ |

Table A.13: Isolet dataset statistical results. Each cell display the statistical difference between the two methods being compared.

| K:2 | | | | | K:20 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | +++ | ∼ | 30% | mk+r | +++ | +++ | — |
| | mk | ∼ | +++ | +++ | | mk | ∼ | +++ | +++ |
| 50% | mk+r | +++ | +++ | — | 50% | mk+r | +++ | +++ | — |
| | mk | ∼ | +++ | +++ | | mk | ∼ | +++ | +++ |
| 70% | mk+r | +++ | +++ | — | 70% | mk+r | +++ | +++ | — |
| | mk | ∼ | +++ | +++ | | mk | ∼ | +++ | +++ |
| K:5 | | | | | K:30 | | | | |
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | +++ | – | 30% | mk+r | +++ | +++ | — |
| | mk | ∼ | +++ | +++ | | mk | ∼ | +++ | +++ |
| 50% | mk+r | +++ | +++ | — | 50% | mk+r | +++ | +++ | — |
| | mk | ∼ | +++ | +++ | | mk | ∼ | +++ | +++ |
| 70% | mk+r | +++ | +++ | — | 70% | mk+r | +++ | +++ | — |
| | mk | ∼ | +++ | +++ | | mk | ∼ | +++ | +++ |
| K:10 | | | | | K:50 | | | | |
| Comparison | | rand - asc | rand - decr | asc - decr | Comparison | | rand - asc | rand - decr | asc - decr |
| 30% | mk+r | +++ | +++ | — | 30% | mk+r | +++ | +++ | — |
| | mk | ∼ | +++ | +++ | | mk | ∼ | +++ | +++ |
| 50% | mk+r | +++ | +++ | — | 50% | mk+r | +++ | +++ | — |
| | mk | ∼ | +++ | +++ | | mk | ∼ | +++ | +++ |
| 70% | mk+r | +++ | +++ | — | 70% | mk+r | +++ | +++ | — |
| | mk | ∼ | +++ | +++ | | mk | ∼ | +++ | +++ |

# Appendix B

# Paper published in ACM International Conference on Computing Frontiers 2017

# Finding the Critical Sampling of Big Datasets

José Silva
Department of Informatics
Engineering, University of Coimbra
Coimbra, Portugal 3030-290
jmpsilva@student.dei.uc.pt

Bernardete Ribeiro
Department of Informatics
Engineering, University of Coimbra
Coimbra, Portugal 3030-290
bribeiro@dei.uc.pt

Andrew H. Sung
School of Computing, The University
of Southern Mississippi
Hattiesburg, Mississippi 39406
andrew.sung@usm.edu

## ABSTRACT

Big Data allied to the Internet of Things nowadays provides a powerful resource that various organizations are increasingly exploiting for applications ranging from decision support, predictive and prescriptive analytics, to knowledge extraction and intelligence discovery. In analytics and data mining processes, it is usually desirable to have as much data as possible, though it is often more important that the data is of high quality thereby two of the most important problems are raised when handling large datasets: sampling and feature selection. This paper addresses the sampling problem and presents a heuristic method to find the "critical sampling" of big datasets.

The concept of the critical sampling size of a dataset $D$ is that there is a minimum number of samples of $D$ that is required for a given data analytic task to achieve satisfactory performance. The problem is very important in data mining, as the size of data sets directly relates to the cost of executing the data mining task. Since the problem of determining the critical sampling size is intractable, in this paper we study heuristic methods to find the critical sampling.

Several datasets were used to conduct experiments using three versions of the heuristic sampling method for evaluation. Preliminary results obtained have shown the existence of an apparent critical sampling size for all the datasets being tested, which is generally much smaller than the size of the whole dataset. Further, the proposed heuristic method provides a practical solution to find a useful critical sampling for data mining tasks.

## KEYWORDS

Big Data, Critical Sampling Size, Data Mining, Sampling Methods

**ACM Reference format:**
José Silva, Bernardete Ribeiro, and Andrew H. Sung. 2017. Finding the Critical Sampling of Big Datasets. In *Proceedings of CF'17, Siena, Italy, May 15-17, 2017,* 6 pages.
https://doi.org/http://dx.doi.org/10.1145/3075564.3078886

## 1 INTRODUCTION

In the era of Big Data a huge amount of data, from a variety of sources and types, is generated at ever increasing rates. This data can be used for various purposes, and increasing numbers of industries and organizations are recognizing its utility, e.g. using machine learning they can perform data mining tasks to extract insights from the data to make strategic decisions. These tasks of knowledge discovery could be very expensive, such as resources and time consumption, and therefore, it is highly desirable to have techniques that can help selecting only the necessary data instead of using entire datasets. This is because much of the data may be redundant, duplicate, irrelevant, or useless.

We focus on two key issues: the problem of selecting good, and the problem of sampling relevant data from big datasets, which are addressed in this paper. We propose a heuristic method to find their Critical Sampling Size (CSS), that is, the absolute minimal number of data samples required to ensure that a learning machine meets a desirable performance. Taking into account that a heuristic method was able to find the Critical Feature Dimension (CFD) of big datasets [4], and given the great similarity of the CSS and CFD problems, we believe that a similar approach will also attain satisfactory and practical results for the CSS problem.

Previous research relevant to our current work includes the adaptive sampling methods for knowledge discovery that adaptively determines the sampling size [2]. Furthermore, a method for speeding up the training time of Support Vector Machines (SVM), which is one of its drawbacks, was proposed in [1], which relies on performing a random sampling of the training set in order to obtain a sketch of the optimal separating hyperplane. In the last case, our heuristic can be an added value improving the way the sampling is done.

In the next section the concept of Critical Sampling Size (CSS) is defined while the proposed heuristic method to implement is presented. In section 3 the experimental setup is described and details of how this work was conducted are given. In section 4, the experimental results on selected datasets to demonstrate the existence of the CSS are detailed. Finally, section 5 provides conclusions and addresses some lines of future work.

## 2 CRITICAL SAMPLING

In this section the concept of Critical Sampling is defined, and the heuristic method proposed in [3] is described. It will be implemented for the experimental design. An intuitive way to define the concept of CSS is the absolute minimal number of samples, of a given dataset, that are needed in order to a specific learning machine achieve some performance threshold.

To formally define this problem, a given dataset is represented by $D_n$, where $n$ denotes the number of samples at hand. The learning machine is designated by $M$ and the performance threshold by $T$. With respect to the notation we use $v$ to denote the CSS where $v \leq n$.
In order to $v$ actually represent the Critical Sampling Size of a specific dataset, the two following conditions must hold.

There exists a $D_v$ which lets $M$ achieve a performance of at least $T$, where $D_v$ is a sampling of $D_n$ with $v$ samples and $P_M(D_v)$ is the performance of $M$ when trained with sampling $D_v$:

$$(\exists D_v \subset D_n)[P_M(D_v) \geq T] \tag{1}$$

For all $j < v$, a sampling of $D_v$ with $j$ samples fails to let $M$ achieve a performance of at least $T$

$$(\forall D_j \subset D_n)[j < v \Rightarrow P_M(D_j) < T] \tag{2}$$

Thus, $v$ is the critical (absolute minimal) number of data samples required in any sampling to ensure that the performance of $M$ meets the given performance threshold $T$. The problem of deciding if a given number $v$ is the CSS for the dataset $D_n$ with respect to given learning machine $M$ and performance threshold $T$ is both NP-hard and coNP-hard; a sketch of the proof is given in [3]. For this reason, we resort to heuristic methods that can also obtain satisfactory results.

## 2.1 Heuristic Method

The heuristic method is composed by five steps. It is in fact a meta-heuristic algorithm since it is a higher-level procedure that resorts to another algorithms (learning machines) in order to solve the CSS problem. In [3], where this heuristic methodology was proposed, it is possible to find a more detailed description about it.

(1) Apply a clustering algorithm like k-means to partition $D_n$ into $k$ clusters. ($k$ can be determined, for example, by the number of classes in the dataset)
(2) Select, say randomly, $m$ samples from each cluster to form a sampling with $m * k$ samples. (The value $m$ is set to be fairly small)
(3) Supplement the sampling with additional $d * k$ (for some d) samples, selected randomly from the whole dataset $D_n$, to form a sampling $D_v$.
(4) Apply learning machine $M$ on the sampling, then measure performance $P_M(D_v)$.
(5) If $P_M(D_v) \geq T$, then $D_v$ is a critical sampling, and its size $v$ is the Critical Sampling Size for $(D_n, M)$. Otherwise enlarge $D_v$ by repeating step 2 and step 3 until a critical sampling is found, or until the whole $D_n$ is exhausted and procedure fails to find $v$.

## 2.2 Performance of machine M

One aspect that deserves special attention is the method to calculate the performance of a specific machine $M$ when trained with the data sampling $D_v$. There is more than one way proposed by the authors in [3].
The performance of machine $M$ trained with $D_v$ is denoted by $P_M(D_v)$, and can be calculated using ($D_n$-$D_v$) as testing set or using a testing set with fixed size. The second approach is more consistent, then it will be the one used.

Another important aspect to point out is the size of the test set being used. For example, when using the 30% for training and 70% for testing, the critical sampling should also be tested with 70%. That is, for the results using the entire dataset, the model is created using 30% of the data and tested with 70%. To obtain the CSS results, the model is then trained with critical sampling $D_v$ and tested with 70% of the data. Next, the two performances are compared. Therefore there is no point in $v$ getting bigger than the size of the train set, in this case, more than 30% of the dataset size. So this could be considered a stopping condition.

## 3 EXPERIMENTAL SETUP

In this section some aspects regarding the used datasets are presented, followed by how the sampling method is conducted when constructing the Critical Set $D_v$. In the following stage, some relevant aspects regarding the parameters of the heuristic method are discussed. Lastly, the learning machines used in the experiments are depicted.

## 3.1 Datasets

The datasets used in this work were downloaded from the UCI Machine Learning Repository[1]. Their dimensions vary both in the number of samples and in the number of features. In this way, we are able to visualize the quality of the heuristic for different datasets. Due to space limitations, no detailed information about the datasets are given as they are found on the repository.
Table 1 shows the characteristics of the used datasets.

**Table 1: Datasets properties**

| | Ads | Amazon | Credit | Hapt | Isolet |
|---|---|---|---|---|---|
| **# Features** | 1558 | 10000 | 23 | 561 | 617 |
| **# Classes** | 2 | 50 | 2 | 12 | 26 |
| **# Samples** | 3279 | 1500 | 30000 | 10929 | 7797 |

## 3.2 Sampling method

The size of the train and test sets can affect the performance that a learning machine achieves, for example, if the model is built with a small number of training samples it may not behave well with future data, thus getting a bad performance. So in the results, when comparing the performance of a data mining task using the entire dataset and the sampled data, $D_v$, this has to be considered. This said, three ways to split the datasets are considered, and therefore, for each of these methods, a respective CSS will be heuristically determined. Each method consists in splitting the dataset in different ratios, which are

- 30% Train / 70% Test
- 50% Train / 50% Test
- 70% Train / 30% Test

In addition to eventually ending up with a distinct CSS for each splitting ratio, this will also allow to observe their influence on the results. If eventually, the CSS for the 3 splitting ratios ends up being the same, or very close to each other (the concept of closeness may

[1]http://archive.ics.uci.edu/ml

differ depending on the nature of the problem), this may indicate that the CSS of the dataset is independent of the used splitting ratios.

As stated before in Section 2.1, the CSS is obtained iteratively. First the data is clustered using k-means and then, from each cluster, $m$ samples are selected to form the critical sampling $Dv$. To analyze the usefulness of this method, two additional approaches are proposed.

- $mk + r$: Initial approach, the sampling $D_v$ is composed by selecting $m$ samples from each one of the $k$ clusters and then complemented with more $d$ random samples
- mk: Same as above except that the sampling $D_v$ is not complemented with $d$ random samples
- r: The construction of $D_v$ is made by randomly selected samples. In order to maintain some consistency, here R can be calculated with $m * k$

## 3.3 Parameters Setup

There are four parameters in the definition of the heuristic to find the CSS (see Section 2.1) that are not settled and their values should be decided by considering (i) the nature of the problem, (ii) the size of the dataset, (iii) the data mining task that will be applied and (iv) the amount of available resources. These four parameters are, respectively, $k$, $m$, $d$ and $T$.

*3.3.1 Parameter $k$.* This parameter represents the number of clusters in which the data will be partitioned. In a first stage, its value will be the number of classes of the dataset. Note that when the dataset has only two classes the number of clusters will be diminutive. This will make the sampling process very similar to the $r$ approach, even if the $mk$, or $mk+r$, is being used. The way that the $m$ samples are selected from each cluster is always randomly but, because the number of clusters, in this case, only two, the random process of selection is being made in half of the dataset. That is like applying the $r$ approach but only on half of the samples.

Hence, in these situations, it is advisable to increase the value of $k$ (i.e. $k = 10$) in order to form more clusters and consequently make it possible to create a more descriptive sampling of the input space.

*3.3.2 Parameter $m$.* This parameter should be fairly small so that in each iteration of the heuristic implementation, it may be possible to observe the evolution of the classifiers' performance, when trained with the sampling $D_v$. For the results obtained, to define $m$, the value of 1% of the size of the generated clusters in step 1 of the heuristic (see 2.1), was used.

*3.3.3 Parameter $d$.* In the $mk + r$ approach, the sampling is composed by complementing it with more $d * k$ samples. These samples are randomly chosen from the remaining dataset. For this addition to be not as significant as the $m * k$ step, the parameter $d$ is useful to take it into account, and is defined by:

$$d = \frac{m}{4} \tag{3}$$

Other values can certainly be used. An interesting point that clarifies $d$ parameter is that it allows to check other CSS values for the same dataset. This would not be not be possible with the previous two approaches. As a consequence it can help tuning the value of the CSS for a specific dataset.

*3.3.4 Parameter $T$.* The value of threshold $T$ represents a reasonable performance requirement or expectation of the specific learning machine $M$, when it is selected to solve a specific problem and given available dataset $D$.

It is desirable to the critical sampling to achieve the same performance as the one obtained using the entire dataset, that is, with training set 30%, 50% or 70% of the whole data. Hence, threshold $T$ can be defined as the expected performance value but, we can relax the heuristic and propose it to be $T = P_M(D_n) - 1\%$. As with the critical sampling performance approaches $P_M(D_n)$, also herein the increasing of performance per iteration should begin to decrease, thus making unnecessarily enlarge $D_v$ without getting visible improvements. That is why the slack variable of 1% was introduced in the threshold, but in some cases, depending on the nature of the problem, it may be considerably high and therefore it should be properly adjusted.
Note that this is an empirical study which primary goal is to verify if datasets possesses, or does not possess, a CSS. We are using datasets with dimensions that allows to infer the expected value of $T$ when using 30%, 50% or 70% of the whole dataset for training, in useful time. One reason for this is to show the usefulness of our proposed heuristic. In real cases, when dealing with much larger datasets, then it will have to be defined by the user.

## 3.4 Learning Machines

In the experimentation the datasets are first classified by building a model based on six different algorithms depicted in Table 2. We choose the machine with the best prediction accuracy as the classifier that will be used to find the CSS for that dataset. Due to space limitations, descriptions of the learning machines and references are not included as they can be found easily elsewhere.

**Table 2: Used algorithms**

| Algorithm | Learning Machine |
|---|---|
| Lazy | KNN |
| Meta | AdaBoost |
| Neural Network | Neural Network |
| Tree | Decision Tree<br>Random Forest |
| Bayes | Naive Bayes |

After all six algorithms have been tested, we have concluded that for binary datasets the best one to use is AdaBoost and for the remaining datasets, Artificial Neural Networks.

## 4 RESULTS AND DISCUSSION

The datasets were split into train and test sets according to the three proposed ratios. Then for each ratio, the used algorithms were executed 30 times. Note that we are trying to find the existence of an CSS of a specific dataset and since we are not optimizing the performance of the algorithms, no pre-processing whatsoever is made in the datasets. The results were obtained using a machine running Linux with 64GB of ram, and an i7 processor with 12 cores at 3.4GHz.

The main goal of Tables 3 to 7 is to report the values of the minimum and maximum size, in percentage, of the Critical Set relative to the total size of each dataset. These tables do not show the frequency of the CSS values obtained over the 30 runs, their purpose is to demonstrate the reduction that each dataset suffered for each combination of sampling method, performance measure and splitting ratio. Therefore, in the first cell of Table 3, where the values are "2.74%-11.28%", it means that in 30 runs the minimum and maximum size obtained for the Critical Set was 2.74% and 11.28%, respectively.

As it can be seen, when the size of the testing set decreases, the maximum value of the CSS increases, in order to attain the performance threshold $T$. This occurs for all datasets, regardless of the sampling method or performance measure used. However, the minimum value remains the same, or slightly changes, except for the Amazon dataset. A possible justification is due to the fact that this dataset has a high number of classes, 50. When the size of the training set increases (smaller testing set), the built model has a better performance and because of this, the CSS needed to also achieve that performance, will become larger.

The results for all datasets, except Amazon, shows that what was stated in Section 3.2 may be true for most cases, i.e. the CSS for a specific dataset is independent on the used splitting ratio. Although sometimes the difference between the minimum and maximum CSS values is large (Ads and Hapt dataset), the values obtained in the 30 runs follow a seemingly normal distribution and so, it is possible to determine with more certainty the expected value for the CSS. Moreover, this normal distribution is more evident when using the $mk(+r)$ method while in the random sampling method, this is not so noticeable. In the Ads dataset, the big difference of minimum and maximum values may be due to a great unbalance of the classes that it presents. Still on this dataset, despite the big difference, 9 times, that is, almost a third of the 30 runs, the CSS value is between 250 and 300 (see Figure 1).

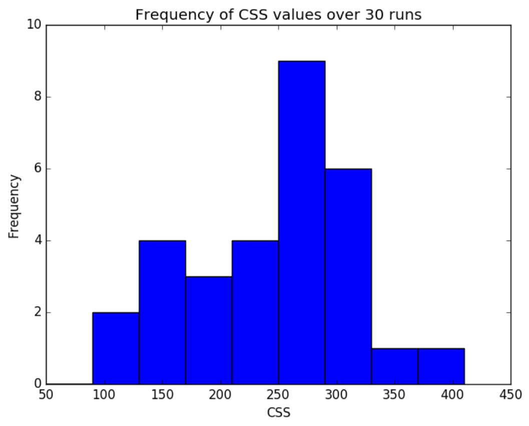


**Figure 1: Dataset: Ads, Sampling Method: $mk + r$, Perf. Measure: Accuracy, Test Set: 70%**

Figure 2 shows how the accuracy improves as the CSS increases. Each vertical bar represents the mean accuracy (point) plus or minus the standard deviation (whiskers) obtained, when the learning machine was trained with the corresponding CSS. The "baseline" indicates the obtained accuracy, averaged over 30 runs, when the learning machine was trained with 70% of the dataset and the "stop condition" line corresponds to the previously defined threshold $T$.

The stop condition line is first crossed when the CSS is 170 but as it increases, the mean value (point) becomes closer to this line. Despite the small improvement of the accuracy when the CSS increases from 170 up to 290, the overall performance of the machine is increased tremendously. When the value of the CSS reaches 330, an improvement in accuracy is more noticeable. This reinforces what were previously stated that the most frequent values for the CSS was between 250 and 300.

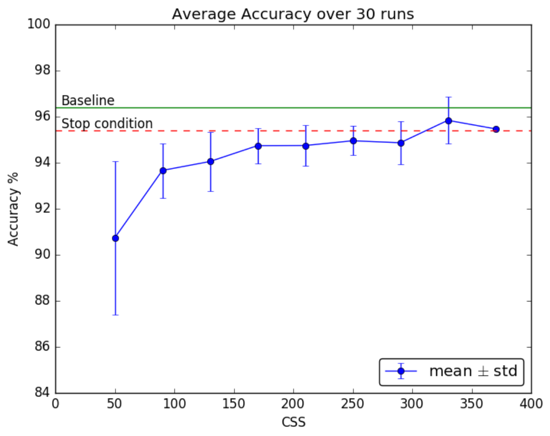


**Figure 2: Dataset: Ads, Sampling Method: $mk + r$, Perf. Measure: Accuracy, Test Set: 70%**

For the Amazon dataset and Isolet dataset, the sampling method that obtained better results (i.e. minimum value for the CSS) for most of the times was $r$. This may be due the fact that these data sets have few samples per class, thus making the probability of selecting samples from various class, higher. Because of this, the number of samples needed to build a good model will be smaller, and so the $r$ method can be sufficient to obtain good results. Also for the Amazon dataset the results using the $mk$ method were very similar to the ones using the $r$ method. The remaining datasets obtained better results, most of the times, when using the $mk$ method, demonstrating this way that can be more efficient method than $r$. Furthermore, as can be seen, the $mk + r$ method always presents values higher than the $mk$, due to the $d$ samples that complemented it. Since the performance achieved by both methods are quite close, once again, the $mk$ method appears to be the best option.

This suggests that the $mk$ method could be used in the pre-processing step of the method proposed in [1], instead of using the random sampling that is made, in order to better select samples that will define the decision boundary of SVM. Also, it can be applied to any other type of problems, as it is independent of the used learning

**Table 3: Minimum-Maximum Critical Sample Size percentages relative to the total size of the Ads dataset**

| **Sampling Method** | *mk* + *r* | | *mk* | | *r* | |
|---|---|---|---|---|---|---|
| **Perf. Measure** | **Accuracy** | **F-score** | **Accuracy** | **F-score** | **Accuracy** | **F-score** |
| **Test Set: 70%** | 2.74%-11.28% | 2.74%-18.60% | **2.44%-12.20%** | **2.44%-15.86%** | 3.66%-13.42% | 4.88%-15.86% |
| **Test Set: 50%** | 3.96%-13.72% | 5.18%-23.48% | **3.66%-12.20%** | **4.88%-21.96%** | 3.66%-17.08% | 4.88%-23.18% |
| **Test Set: 30%** | 2.74%-17.38% | 5.18%-24.70% | **2.44%-12.20%** | **3.66%-24.40%** | 3.66%-13.42% | 4.88%-30.50% |

**Table 4: Minimum-Maximum Critical Sample Size percentages relative to the total size of the Amazon dataset**

| **Sampling Method** | *mk* + *r* | | *mk* | | *r* | |
|---|---|---|---|---|---|---|
| **Perf. Measure** | **Accuracy** | **F-score** | **Accuracy** | **F-score** | **Accuracy** | **F-score** |
| **Test Set: 70%** | 17.47%-20.80% | 17.47%-20.80% | 16.67%-23.33% | **16.67%-23.33%** | **16.67%-20.00%** | **16.67%-23.33%** |
| **Test Set: 50%** | 24.13%-30.80% | 24.13%-30.80% | **23.33%-30.00%** | **23.33%-30.00%** | **23.33%-30.00%** | **23.33%-30.00%** |
| **Test Set: 30%** | 30.80%-37.47% | 30.80%-44.13% | 26.67%-40.00% | **30.00%-40.00%** | **26.67%-33.33%** | **30.00%-40.00%** |

**Table 5: Minimum-Maximum Critical Sample Size percentages relative to the total size of the Credit dataset**

| **Sampling Method** | *mk* + *r* | | *mk* | | *r* | |
|---|---|---|---|---|---|---|
| **Perf. Measure** | **Accuracy** | **F-score** | **Accuracy** | **F-score** | **Accuracy** | **F-score** |
| **Test Set: 70%** | 2.25%-7.25% | 1.25%-6.25% | **2.00%-6.00%** | **1.00%-7.00%** | 2.00%-8.00% | 2.00%-6.00% |
| **Test Set: 50%** | 2.25%-6.25% | 1.25%-6.25% | 2.00%-7.00% | **1.00%-7.00%** | **2.00%-6.00%** | 1.00%-9.00% |
| **Test Set: 30%** | 2.25%-7.25% | 1.25%-6.25% | **2.00%-11.00%** | **1.00%-8.00%** | 3.00%-7.00% | 2.00%-11.00% |

**Table 6: Minimum-Maximum Critical Sample Size percentages relative to the total size of the Hapt dataset**

| **Sampling Method** | *mk* + *r* | | *mk* | | *r* | |
|---|---|---|---|---|---|---|
| **Perf. Measure** | **Accuracy** | **F-score** | **Accuracy** | **F-score** | **Accuracy** | **F-score** |
| **Test Set: 70%** | 6.86%-12.35% | 3.57%-14.55% | **5.49%-9.88%** | **4.39%-10.98%** | 5.49%-12.08% | 7.69%-23.06% |
| **Test Set: 50%** | 6.86%-12.35% | 4.67%-12.35% | **6.59%-17.57%** | **4.39%-14.27%** | **6.59%-17.57%** | 6.59%-29.65% |
| **Test Set: 30%** | 6.86%-17.84% | 4.67%-12.35% | 6.59%-16.47% | **4.39%-16.47%** | **6.59%-15.37%** | 5.49%-25.25% |

**Table 7: Minimum-Maximum Critical Sample Size percentages relative to the total size of the Isolet dataset**

| **Sampling Method** | *mk* + *r* | | *mk* | | *r* | |
|---|---|---|---|---|---|---|
| **Perf. Measure** | **Accuracy** | **F-score** | **Accuracy** | **F-score** | **Accuracy** | **F-score** |
| **Test Set: 70%** | 11.25%-18.25% | 12.25%-15.25% | 12.00%-20.01% | **10.00%-17.01%** | **10.00%-15.01%** | 11.00%-14.01% |
| **Test Set: 50%** | 12.25%-20.25% | **11.25%-19.25%** | 12.00%-22.01% | 13.01%-20.01% | **12.00%-18.01%** | 12.00%-18.01% |
| **Test Set: 30%** | 12.25%-22.25% | 12.25%-23.25% | 13.01%-23.01% | 13.01%-24.01% | **11.00%-20.01%** | **11.00%-21.01%** |

machine. However, it is important to note that the threshold value $T$ will need to be defined on the basis of prior knowledge of the problem under study, since for big datasets it will be intractable to compute the baseline.

In addition, there was a noticeable reduction in size for each dataset, reinforcing the hypothesis of this investigation which states that heuristic methods can find practically useful CSS for datasets.

## 5 CONCLUSIONS

In this research an empirical study on big data sets was conducted to assess if each dataset possesses a Critical Sampling Size. In data mining, big datasets are often used, and building good models for knowledge discovery could be very expensive (i.e. resources and time consumption) because the sheer size of the used datasets. Finding the CSS, therefore may significantly reduce the cost on data mining while maintaining the desirable performance.

In the proposed approach several classifiers as well as three sampling methods for the heuristic were used, in order to analyze their effects on the results. The importance of the heuristic parameters has also been described and that their values must be carefully considered to better adapt to the nature of the problem.

The preliminary experiments completed so far have shown the existence of an apparent CSS for each dataset; note that we use the term "apparent" here since finding the exact CSS is highly intractable, hence whatever sampling generated by the heuristic methods would be an approximated CSS at best. In addition, the CSS of the datasets are demonstrated to be considerably smaller than their initial size. These results seem to validate our assumption that since it is possible to find the (apparent) CFD of a dataset using simple heuristic methods, it is likely that such methods will prove practicable in finding the dataset's (apparent) CSS.

In order to fully validate these results, more experiments will need to be conducted. These experiments will focus on the parameters of the implemented heuristic as well as the use of new and bigger datasets. Furthermore, the way that the selection of samples from each cluster is made, when using $mk(+r)$ method, may try to be improved. For instance, try to get more samples from each class, instead the picking being done randomly; consider the density or the distance from the centroid of the clusters. As for the way in which $k$ is defined, in the first step of the heuristic 2.1, it can be improved by using techniques such as the "elbow point" technique or the silhouette method. In the end, these modifications to the $mk(+r)$ method may show more significant improvements over random sampling.

As cloud computing is evolving into a mature technology, the rapidly proliferating IoT devices, on top of the existing IT infrastructure, are generating big data at unprecedented rates and driving up data-related costs for organizations and enterprises, prompting managers to demand higher ROI for analytics [5]. With datasets getting increasingly larger, and the emerging "data on demand" services, the sampling problem will become more and more important, and effective heuristic methods for composing high-quality or "critical" samples will prove increasingly useful in practical developments that involve analytics or data mining. It is hoped that the heuristic method studied in this paper, once refined and fully validated, will provide a cost-effective tool for data sampling in analytics and data mining tasks.

## REFERENCES

[1] Jair Cervantes, Farid García Lamont, Asdrúbal López-Chau, Lisbeth Rodríguez Mazahua, and J Sergio Ruíz. 2015. Data selection based on decision tree for SVM classification on large data sets. *Applied Soft Computing* 37 (2015), 787–798.

[2] Carlos Domingo, Ricard Gavalda, and Osamu Watanabe. 2002. Adaptive Sampling Methods for Scaling Up Knowledge Discovery Algorithms. *Data Mining and Knowledge Discovery, Volume 6, Issue 2, pp.131-152.* (2002).

[3] Andrew H. Sung, Bernardete Ribeiro, and Qingzhong Liu. 2016. Sampling and Evaluating the Big Data for Knowledge Discovery. (2016). International Conference on Internet of Things and Big Data.

[4] Qingzhong Liu, Andrew H. Sung, Bernardete Ribeiro, and Divya Suryakumar. 2014. Mining the Big Data: The Critical Feature Dimension Problem. (2014). Proceedings of eKNOW 2014, The Sixth International Conference on Information, Process, and Knowledge Management.

[5] Alex Woodie. 2016. Companies struggle to find an ROI on analytics. (2016). https://www.datanami.com/2016/07/06/companies-struggle-find-roi-analytics/

# Bibliography

[1] Big data in telco and banking analytics, 2013. URL `http://www.ibm.com/ru/events/presentations/rstl/IBM%20STC%20Innovation%20Day%20Big%20Data%20For%20Banks%20and%20Telco%20Benjamin%20Sznajder.pdf`.

[2] The zettabyte era—trends and analysis, 2016. URL `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html`.

[3] Hermann Kopetz. Internet of things. In *Real-time systems*, pages 307–323. Springer, 2011.

[4] Andrew H. Sung, Bernardete Ribeiro, and Qingzhong Liu. Sampling and evaluating the big data for knowledge discovery. In *Proceedings of the International Conference on Internet of Things and Big Data*, pages 378–382, 2016.

[5] Carlos Domingo, Ricard Gavaldà, and Osamu Watanabe. Adaptive sampling methods for scaling up knowledge discovery algorithms. *Data Mining and Knowledge Discovery*, 6(2):131–152, 2002.

[6] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Occam's razor. *Information processing letters*, 24(6):377–380, 1987.

[7] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1):245–271, 1997.

[8] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[9] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422, 2002.

[10] Olivier Chapelle and S Sathiya Keerthi. Multi-class feature selection with support vector machines. In *Proceedings of the American statistical association*, 2008.

[11] Rudy Setiono and Huan Liu. Neural-network feature selector. *IEEE transactions on neural networks*, 8(3):654–662, 1997.

[12] Enrique Romero and Josep María Sopena. Performing feature selection with multilayer perceptrons. *IEEE Transactions on Neural Networks*, 19(3):431–441, 2008.

[13] Qingzhong Liu, Bernardete Ribeiro, Andrew H Sung, and Divya Suryakumar. Mining the big data: The critical feature dimension problem. In *Advanced Applied Informatics (IIAIAAI), 2014 IIAI 3rd International Conference on*, pages 499–504. IEEE, 2014.

[14] Jair Cervantes, Farid García Lamont, Asdrúbal López-Chau, Lisbeth Rodríguez Mazahua, and J Sergio Ruíz. Data selection based on decision tree for svm classification on large data sets. *Applied Soft Computing*, 37:787–798, 2015.

[15] Min Wang, Bala Iyer, and Jeffrey Scott Vitter. Scalable mining for classification rules in relational databases. In *Database Engineering and Applications Symposium, 1998. Proceedings. IDEAS'98. International*, pages 58–67. IEEE, 1998.

[16] Carlos Domingo, Ricard Gavalda, and Osamu Watanabe. Practical algorithms for on-line sampling. In *International Conference on Discovery Science*, pages 150–161. Springer, 1998.

[17] Carlos Domingo Soriano, Ricard Gavaldà Mestre, and Osamu Watanabe. On-line sampling methods for discovering association rules. 1999.

[18] Rosa L Figueroa, Qing Zeng-Treitler, Sasikiran Kandula, and Long H Ngo. Predicting sample size required for classification performance. *BMC medical informatics and decision making*, 12(1):8, 2012.

[19] M. Lichman. Uci machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[20] Pang-Ning Tan et al. *Introduction to data mining.* Pearson Education India, 2006.

[21] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[22] Sabhia Firdaus and Md Ashraf Uddin. A survey on clustering algorithms and complexity analysis. *International Journal of Computer Science Issues (IJCSI)*, 12 (2):62, 2015.

[23] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996.

[24] Birch clustering algorithm. `http://scikit-learn.org/stable/modules/clustering.html#birch` note = Accessed: 2017-04-13.

[25] Toon Van Craenendonck and Hendrik Blockeel. Using internal validity measures to compare clustering algorithms. In *Benelearn 2015 Poster presentations (online)*, pages 1–8, 2015.

[26] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[27] Adaboost classifier. URL `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html`. Accessed: 2016-10-10.

[28] Ji Zhu, Hui Zou, Saharon Rosset, Trevor Hastie, et al. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[29] Andy Field. *Discovering statistics using SPSS*. Sage publications, 2009.