Cahsai, A., Anagnostopoulos, C. and Triantafillou, P. (2015) Scalable data quality for big data: the Pythia framework for handling missing values. *Big Data*, 3(3), pp. 159-172. (doi:10.1089/big.2015.0002)

This is the author's final accepted version.

There may be differences between this version and the published version. You are advised to consult the publisher's version if you wish to cite from it.

http://eprints.gla.ac.uk/109175/

Deposited on: 20 August 2015

Enlighten – Research publications by members of the University of Glasgow
http://eprints.gla.ac.uk

# Scalable Data Quality for Big Data: The Pythia Framework for Handling Missing Values

A. Cahsai, C. Anagnostopoulos, P. Triantafillou
School of Computing Science
University of Glasgow, G12 8QQ, Glasgow, UK
a.cahsai.1@research.gla.ac.uk; christos.anagnostopoulos@glasgow.ac.uk;
peter.triantafillou@glasgow.ac.uk

## ABSTRACT

Solving the missing-value (MV) problem with small estimation errors in large-scale data environments is a notoriously resource-demanding task. The most widely used MV imputation approaches are computationally expensive because they explicitly depend on the volume and the dimension of the data. Moreover, as datasets and their user community continuously grow, the problem can only be exacerbated. In an attempt to deal with such problem, in our previous work [1], we introduced a novel framework coined Pythia, which employs a number of distributed data nodes (cohorts), each of which contains a partition of the original dataset. To perform MV imputation, the Pythia, based on specific machine and statistical learning structures (signatures), selects the most appropriate subset of cohorts to perform locally a Missing Value substitution Algorithm (MVA). This selection relies on the principle that that particular subset of cohorts maintains the most relevant partition of the dataset. In addition to this, as Pythia uses only part of the dataset for imputation and accesses different cohorts in parallel, it improves efficiency, scalability and accuracy comparing against a single machine (coined Godzilla), which uses the entire massive dataset to compute imputation requests. Although this paper is an extension to our previous work, we particularly investigate the robustness of the Pythia framework and show that the Pythia is independent from any MVA and signatures construction algorithms. In order to facilitate our research, we considered two well-known MVAs (namely K-nearest neighbor and expectation-maximization imputation algorithms) as well as two machine and neural computational leaning signature construction algorithms based on adaptive vector quantization and competitive learning. We prove comprehensive experiments to assess the performance of the Pythia against Godzilla and showcase the benefits stemmed from this framework.

**Categories and Subject Descriptors:** H. Information Systems; I.5.3 Clustering.

**Keywords:** Big data; Scalability; Missing values; Imputation; Adaptive vector quantization; Self organizing maps; Adaptive resonance theory.

## 1. INTRODUCTION

Data quality is a major concern in big data processing and knowledge management systems. One relevant problem in data quality is the presence of missing values (MVs). The MV problem should be carefully addressed, otherwise bias might be introduced into the induced knowledge. Common solutions to the MV problem either fill-in the MVs (*imputation*) or ignore / exclude them. Imputation entails a *MV substitution algorithm* (MVA) that replaces MVs in a dataset with some plausible values.

On the one hand, most computational intelligence and machine learning (ML) techniques (such as neural networks and support vector machines) fail if one or more inputs contains MVs and thus cannot be used for decision-making purposes [2]. Furthermore, the choice of different MVAs affects the performance of ML techniques that are subsequently used with imputed data [3]. On the other hand, the MV problem abounds: it can be found, for instance, in results from medical experimentation and chemical analysis, in datasets from domains such as meteorology and microarray gene monitoring technology [6], and in survey databases [7]. MVs can occur e.g., due to wireless sensor faults, not reacting experiments, or participants skipping survey questions. Industrial and research databases include MVs [8], e.g., maintenance databases have up to 50% of their entries missing [9]. Patient records in medical databases lack some values; interestingly, a database of patients with cystic fibrosis missing more than 60% of its entries was analyzed in [10]. Moreover, gene expression microarray data sets contain MVs, making the need for robust MVAs apparent, since algorithms for gene expression analysis require complete gene array data [11].

**Motivations.** Given the significance of MVAs, three notes are in order: Firstly, MVAs which can ensure low estimation errors are computationally expensive and typically their performance is largely dependent on dataset sizes and on the data dimension. Secondly, nowadays, datasets can be massive. Even worse, existing datasets grow significantly with time; it is not surprising that most MVAs in the literature are typically tested over small-to medium sized datasets. Lastly, as if the scalability limitations imposed by dataset sizes were not enough, in many applications the user community (e.g., in shared scientific datasets in data centers accessed by scientists from all over the world) can be very large and thus the MV imputation input arrival rates can

become high as well. These facts pose a scalability nightmare.

The scalability gospel (as established by the seminal work from Google researchers producing the Map-Reduce (MR) [12] data-access paradigm and systems such as the Google File System [13]) rests on the notion of *scaling out*: that is, (i) employ a large number of commodity (off-the-shelf and thus inexpensive) machines, each storing a much smaller partition of the original dataset, and (ii) access them in parallel.

However, MR is not a panacea, for the following reasons. First, not all complex problems are 'embarrassingly parallelizable' and amenable to MR techniques. In particular, there exist MVAs coming with small imputation errors, which are not MR-able [14]. The MVAs are basically (complex) statistical and machine learning algorithms like the expectation maximization imputation algorithm [20] and the sequential multivariate regression imputation [19] algorithm. Such algorithms are based on iterative processes, i.e., the MR scheme needs to process data again and again. In such MVAs, the intermediate processes need to communicate to each other and possibly processing requires lot of data to be shuffled over the network of distributed data nodes. In addition, when the Map phase generates too many keys, then sorting takes for *ever*. Moreover, it's not always straight forward to implement any potential ML-based imputation algorithm as a MR program [14]. Further, MR does not efficiently handle streaming MV imputation requests. Since the Map output stream it is not kept into the memory, this will be inefficient especially when dealing with a high rate of MV imputation requests. In the context of MVAs, even if they were 'embarrassingly parallelizable', not all partitions may be relevant. Specifically, given the fact that a number of machines are involved for locally executing the MVAs, the entire massive dataset is partitioned into smaller datasets distributed onto the machines. This dataset partitioning is simple in the sense that each data node contains a unique, non-overlapping subset of the entire dataset. It may very well be the case that a number of the machines hold data that cannot help (or even hurt) in the MV imputation process. And, obviously, engaging only a fraction of all machines will introduce large benefits: First with respect to performance. MV imputation will be shorter, as these times typically depend on the worst performing machine and with increasing machine numbers the probability of a mall-performing machine increases. Further, overall MV imputation throughput will be higher, as each imputation will be taxing fewer overall system resources (processors, communication bandwidth and disks). Second, with respect to MV estimation errors. In fact, as we shall formally show later, engaging all machines and their dataset partitions may actually introduce large additional MV estimation errors.

**Goals.** In this work, we will consider a stream of MV imputation requests, hereinafter referred to as inputs. An input is a multi-dimensional vector with some MVs in certain dimensions, arriving at a data system. Typically, the system is presented with a batch of data items with MVs, which must be added to the system after MVs have been estimated. It is worth noting in this case that, in order to trace back imputed MVs in the system, each *imputed* input is accompanied with an imputation meta-data vector containing the dimension index of each estimated/imputed value. Through this meta-data flagging technique, the system makes clear what values are *real* values or *imputed* ones. From a data management viewpoint, this enables the system to ensure that the stored data maintain integrity by clearly flagging MVs as an important operation and legal matter.

There are two system alternatives to impute the MVs. The first is based on employing a single machine which stores the whole of the dataset. We affectionately call this machine *Godzilla*. Godzilla can employ any MVA to perform the MV imputations. As motivated earlier, this approach suffers from several disadvantages. The second alternative employs a (potentially large) number of machines, referred to as *cohorts*, each storing a partition of Godzilla's dataset. Each cohort stores a unique and non-overlapping subset of the massive Godzilla's dataset. Imputation execution engages cohorts in parallel, whereby each cohort runs an MVA on a much smaller local dataset. This can introduce dramatic performance improvements. As an illustration, let us assume 50 cohorts and an MVA operating on a dataset of size $n$ with asymptotic complexity $O(n^2)$, or $O(n^3)$ [4], [6]. A scale-out execution is expected to speedup input processing by a factor of $50^2 = 2,500$ (or $50^3 = 125,000$) as such MVA runs in parallel on a dataset of size $\frac{1}{50}n$. Moreover, this alternative affords the possibility of accessing only a subset of all cohorts for a given input.

The formidable challenges here entail: (i) for data accuracy (estimation-error) reasons, we should ensure that the subset of cohorts contacted achieve similar, if not smaller estimation errors, compared to the errors that Godzilla would yield; (ii) *swiftly* determine cohort to engage per imputation, achieving large efficiency/scalability gains.

# 2. BACKGROUND & RELATED WORK

## 2.1 Missing data

Assume a data set $\mathcal{X}$ of $d$-dimensional data points with some MVs on a certain dimension $X_i$. Data on $X_i$ are said to be *missing completely at random* (MCAR) if the probability of MV on $X_i$, $q$, is unrelated to the value of $X_i$ itself or to the values of any other dimensions. If data are MCAR, a reduced sample of $\mathcal{X}$ will be a random sub-sample of $\mathcal{X}$; MCAR assumes that the distributions of MVs and complete data are the same. Data on $X_i$ are said to be *missing at random* (MAR) if $q$ depends on the observed data, but does not depend on the MV itself. In MAR, the dimension associated with MVs has a relation to other dimensions, i.e., MVs can be estimated by using the complete data of other dimensions. Data on $X_i$ are *missing not at random* (MNAR) if $q$ depends on the MVs and, thus, imputation is not permissible in this case.

## 2.2 Related work

Missing data hinder the application of many statistical analysis and ML techniques available in off-the-shelf software. To analyze $\mathcal{X}$ with MVs, certain MVAs have been proposed [15]. The simplest method is discarding the data points with MVs or removing the corresponding dimensions. Both removals of such points and dimensions result in decreasing the information content of $\mathcal{X}$ and are applicable only when (i) $\mathcal{X}$ contains a small amount of MVs, and (ii) the analysis of the remaining complete points will not be biased by the removal. There are many MVAs varying from naïve methods, e.g., mean imputation, to some more robust methods based on relationships among dimensions. In the *dummy variable adjustment*, MVs are set to some arbitrary

value. The *mean / mode imputation* replaces MVs of a dimension by the sample mean / mode of all observed values of that dimension. In *hot deck* MVA [16], a MV is filled in with a value from an estimated distribution w.r.t. $\mathcal{X}$. In the K-nearest neighbors MVA [17], the MVs of a point are imputed considering the K most similar (observed) points from $\mathcal{X}$. The regression- and likelihood-based MVAs are introduced in [18]. In *regression-based imputation* [19], the MVs of a point are estimated by regression of the dimensions corresponding to MVs on the dimensions associated to the observed values of that point. This approach argues that dimensions have relationships among themselves; if no relationships exist among dimensions in $\mathcal{X}$ and the dimensions corresponding to MVs, such MVA will not be precise for imputation. *Likelihood-based imputation* [18] is based on parameter estimation in the presence of MVs, i.e., $\mathcal{X}$'s parameters are estimated by maximum likelihood or maximum a posteriori procedures relying on variants of the Expectation-Maximization algorithm. The *multiple imputation* MVA [20], instead of filling in a single value for each MV, replaces each MV with a set of plausible values that represent the uncertainty about the actual value to impute. These multiply-imputed datasets are then analyzed by using standard procedures for complete data and combining the results from these analyses. In case of MVs in time series, the models in [21] (using dynamic Bayesian networks), [22] (using matrix completion), and [23] (using Gaussian mixtures clustering) recover MVs in motion capture sequences, vital signs, and micro-array gene expression streams, respectively. Furthermore, ML-based MVAs, e.g., decision-trees and rule-based methods, generate a model from $\mathcal{X}$ that contain MVs, which is used to perform classification that imputes the MVs (see [3] and the references therein). Finally, the imputation framework [8] applies most existing MVAs (base methods) to improve their accuracy of imputation while preserving the asymptotic computational complexity of the base methods. The interested reader could also refer to [8], [11] and [24] (and the references therein) for a comprehensive survey of the most recent MVAs.

# 3. DEFINITION

## 3.1 Definitions & Notations

*Definition 1.* Given a set $\mathcal{X}$ of $d$-dimensional data points, $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{X}|}\}$, for each $\mathbf{x}_i$ we define the imputation meta-data vector $\mathbf{w}_i = [w_{ik}]^\top$ with $w_{ik} = 0$ whenever $\mathbf{x}_i$'s $k$-th dimensional value is missing; otherwise $w_{ik} = 1$. We express $\mathbf{x}_i$ as $(\mathbf{z}_i, \mathbf{z}_i^{\mathbf{m}})$, where $\mathbf{z}_i \in \mathbb{R}^{d'}$ denotes observed values and $\mathbf{z}_i^{\mathbf{m}} \in \mathbb{R}^{(d-d')}$ denotes MVs, with $d' = \sum_{k=1}^{d} w_{ik}$.

*Definition 2.* Given a finite integer $m > 0$, $\mathcal{X}_i$ is a partition of $\mathcal{X}$ such that $\mathcal{X} \equiv \cup_{i=1}^{m} \mathcal{X}_i$ and $\mathcal{X}_i \neq \mathcal{X}_j, i \neq j$. $S_i$ denotes the machine (*cohort*), which maintains $\mathcal{X}_i$, performs a MVA over $\mathcal{X}_i$, and is indexed by $i$, $i = 1, \ldots, m$. $\mathcal{S} = \{S_i\}_{i=1}^{m}$ is the set of all cohorts. The (imaginary) *Godzilla* $S_0$ assembles all $\mathcal{X}_i$ and is capable of performing a MVA over $\mathcal{X}$.

*Definition 3.* A single MV input on MVA is $\mathbf{i} = (\mathbf{x}, \mathbf{w})$ and output is $\hat{\mathbf{x}}$ expressed by $(\mathbf{z}, \hat{\mathbf{z}}^{\mathbf{m}})$. $\hat{\mathbf{x}} \in \mathbb{R}^d$ is referred to as *estimate* containing $\hat{\mathbf{z}}^{\mathbf{m}} \in \mathbb{R}^{(d-d')}$ of imputed MVs by MVA. If $\mathbf{x}_a$ is the actual vector, the absolute reconstruction error is $e = \| \hat{\mathbf{x}} - \mathbf{x}_a \|$; $\| \mathbf{x} \|$ denotes the Euclidean norm.

## 3.2 MVAs in our framework

As our approach is independent of any particular MVA, we overview and experiment with two popular and representative MVAs in our framework. Note, in our previous work we also experiment with the REG [19] MVA algorithm. To exemplify our framework and methods, we employ the weighted K-nearest neighbors (KNN) [17] and Expectation Maximization imputation method (EM) [20]. These MVAs are widely used for multivariate imputation in many scientific areas.

### 3.2.1 Weighted $K$-nearest neighbors imputation

KNN is widely used [24] since it has many attractive characteristics: it is a non-parametric method, which does not require the creation of a predictive model for each dimension with MV and takes into account the correlation structure of the data. KNN is based on the assumption that points close in distance are potentially similar. For given input $(\mathbf{x}_i, \mathbf{w}_i)$ with $\mathbf{x}_i = (\mathbf{z}_i, \mathbf{z}_i^{\mathbf{m}})$, KNN calculates a weighted Euclidean distance $D_{ij}$ between $\mathbf{x}_i$ and $\mathbf{x}_j \in \mathcal{X}$ such that

$$D_{ij} = \left( \frac{\sum_{k=1}^{d} w_{ik} w_{jk} (x_{ik} - x_{jk})^2}{\sum_{k=1}^{d} w_{ik} w_{jk}} \right)^{1/2}.$$

The MV of the $k$-th dimension of $\mathbf{x}_i$ (i.e., $z_{ik}^{\mathbf{m}}$ of $\mathbf{z}_i^{\mathbf{m}}$) is estimated by the weighted average of non-MVs of the K most similar $\mathbf{x}_j$ to $\mathbf{x}_i$, i.e., $\hat{z}_{ik}^{\mathbf{m}} = \sum_{j=1}^{K} \frac{D_{ij}^{-1}}{\sum_{v=1}^{K} D_{iv}^{-1}} x_{jk}$. KNN is typically used with K=10,15,20; theses values have been favored in previous studies [24], [25]. (In our experiments we will use K=10).

*Remark 1.* A naive approach for searching the closest $d$-dimensional data point with respect to a given point $\mathbf{x}$ over a dataset of size $n = |\mathcal{X}|$ requires $O(nd)$ time. This implies also $O(ndK)$ time for retrieving the $K$ nearest data points. Nonetheless, we could built a $d$-dimensional tree over the points of $\mathcal{X}$ to allow to efficiently perform $K$ nearest neighbors search. Such structure is 'good' for searches in low-dimensional spaces. However, its efficiency decreases as dimensionality grows, and in high-dimensional spaces this structure gives no performance over naive $O(ndK)$ linear search. Overall, the KNN imputation algorithm can build a $d$-dimensional tree with $O(n \log n)$ time complexity and achieves imputation time complexity close to $O(Kd \log n)$.

### 3.2.2 Expectation Maximization imputation

The EM algorithm is an iterative algorithm for estimating MVs by maximizing the likelihood function [18]. Assume that $\mathcal{X}$ is generated by a probability density function $f(\mathcal{X}|\theta)$, where $\theta$ is a parameter of the model. The likelihood function $L(\theta|\mathcal{X})$ is a function of the parameter $\theta$ for fixed $\mathcal{X}$. For mathematical convenience, likelihood function is represented by its log-likelihood function $l(\theta|\mathcal{X}) = \ln(L(\theta|\mathcal{X}))$. Without loss of generality, consider for fixed $\mathcal{X}$ a set of parameters $\theta = \{\theta_1, \ldots, \theta_t\}$ with $t > 0$. For every $\theta_i \in \theta$, we calculate $l(\theta_{i \leq t}|\mathcal{X})$. The obtained outcome shows that how likely $\mathcal{X}$ is observed under $\theta_i$. The highest the outcome is the most likely $\mathcal{X}$ is observed under that parameter. In general, $L$ is used to identify the value of $\theta$, which is best supported by $\mathcal{X}$. For a set $\mathcal{X}$, which contains observed values, $\mathcal{X}_{obs}$ and MVs $\mathcal{X}_{miss}$, the log maximum likelihood of $\mathcal{X}$ is $l(\theta|\mathcal{X})$ $= l(\theta|\mathcal{X}_{obs}, \mathcal{X}_{miss}) = l(\theta|\mathcal{X}_{obs}) + \ln f(\mathcal{X}_{miss}|\mathcal{X}_{obs}, \theta)$. The main concept of EM is to maximize the maximum likelihood

estimation (MLE) of $\theta$ from $l(\theta|\mathcal{X}_{obs})$ in order to maximize the MLE of $l(\theta|\mathcal{X})$. The EM algorithm consists four steps: Step (i) replace MVs by estimated values, Step (ii) estimate $\theta$ (also known as E-step), Step (iii) re-estimate MVs using the new $\theta$ (referred as M-step ), Step (iv) re-estimate $\theta$, iterate until convergence [18].

To illustrate how the EM algorithm is used for imputation, consider the $d$-dimensional mean vector $\mathbf{u} = [u_1, \ldots, u_d]^\top$ and covariance matrix $\Sigma = [\sigma_{jk}]$ with $j, k = 1, \ldots, d$. Both $\mathbf{u}$ and $\Sigma$ refer to learning parameter $\theta$, i.e., $\theta = (\mathbf{u}, \Sigma)$. Initially, $\mu$ and $\Sigma$ are calculated considering only the non missing values, i.e., from the $\mathcal{X}_{obs}$ set. Then, the EM imputation algorithm calculates each step as follows:

- Step 1: For each $\mathbf{x}_i \in \mathcal{X}$, if $w_{ik} = 0$ then we estimate $\hat{z}_{ik}^{\mathbf{m}} = u_k$. Note that $\mathbf{w}_i$ remains unchanged in order to help us identify which dimensions are observed or missed in the original data set $\mathcal{X}$.
- Step 2: Estimation of parameter $\theta_t$ at iteration $t \geq 1$. For each $k, j = 1, \ldots, d$ we calculate:

$$E\left(\sum_{i=1}^{|\mathcal{X}|} x_{ik}|\mathcal{X}_{obs}, \theta_t\right) = \sum_{i=1}^{|\mathcal{X}|} x_{ik}^t$$

and

$$E\left(\sum_{i=1}^{|\mathcal{X}|} x_{ik}x_{ij}|\mathcal{X}_{obs}, \theta_t\right) = \sum_{i=1}^{|\mathcal{X}|} x_{ik}^t x_{ij}^t + c_{jki}^t$$

with

$$\hat{z}_{ik}^{\mathbf{m}} = \begin{cases} x_{ik}, & \text{if } w_{ik} = 1 \\ E\left(\sum_{i=1}^{n} x_{ik}|\mathcal{X}_{obs}, \theta_t\right) & \text{if } w_{ik} = 0 \end{cases}$$

and

$$c_{jki}^t = \begin{cases} 0, & \text{if } w_{ik} = 1 \text{ or } w_{ij} = 1 \\ x_{ik}x_{ij}|\mathcal{X}_{obs}, \theta_t & \text{if } w_{ik} = 0 \text{ and } w_{ij} = 0 \end{cases}$$

At the end of this step, the purpose is to estimate the sufficient statistics, i.e., mean, variance, and covariance so that the following step can update the parameter $\theta_t$. Specifically, this step estimates $\mathbf{u}$ and $\Sigma$, and uses them to build a set of regression equations that predict the missing values from the $\mathcal{X}_{miss}$ set. This is achieved by the *sweep* regression operator [18] realizing the conditional expectations. Such operator combines the mean vector and the covariance matrix into a single augmented matrix and applies a series of transformations that produce the desired regression coefficients and residual variances.

- Step 3: re-estimate MVs using the new $\theta_t$ parameter. This step becomes a straightforward estimation problem that uses the filled-in sufficient statistics from the previous step to impute the missing values. Then, for each $k, j = 1, \ldots, d$ we calculate:

$$u_k^{(t+1)} = (|\mathcal{X}| - 1)^{-1} \sum_{i=1}^{|\mathcal{X}|} \hat{z}_{ik}^{\mathbf{m}}$$

and

$$\sigma_{jk}^{(t+1)} = (|\mathcal{X}| - 1)^{-1} \sum_{i=1}^{|\mathcal{X}|} [(x_{ij} - \mu_j)(x_{ik} - \mu_k) + c_{jki}]$$

- Step 4: If $|l(\theta_{t+1}|\mathcal{X}) - l(\theta_t|\mathcal{X})| \leq \epsilon$ then terminate (converge); otherwise, go to Step 2. We set $\epsilon = 10^{-3}$ for convergence.

*Remark 2.* Each iteration takes $O(nd)$ computations given that $n = |\mathcal{X}|$. However, the termination behavior of EM is not easy and guaranteed. Theoretically speaking, without any stopping threshold (or, setting a stopping threshold $\epsilon = 0$), EM would infinitely converge up to an infinite precision, i.e., $\epsilon = 0$. Hence, the theoretical runtime of EM is infinite. Any small and non-negative threshold $\epsilon > 0$ and $\epsilon \to 0$ forces EM to terminate earlier. But it will be hard to get a theoretical limit here different than $O(ndt_\epsilon)$ where $t_\epsilon$ is the number of iterations up to achieving precision close to $\epsilon$.

# 4. THE PYTHIA FRAMEWORK

The Pythia framework in [1] employs a potentially large number of cohorts, $\mathcal{S}$. Each cohort, $\mathcal{S}_i \in \mathcal{S}$, stores a unique and non-overlapping subset of a massive dataset $\mathcal{X}$. Imputation execution engages cohorts in parallel, whereby each cohort runs a MVA on a much smaller local dataset, $\mathcal{X}_i$. Accordingly, for some input, Pythia must swiftly predict the appropriate cohort or cohorts, $\mathcal{S}'$, in which the MVA is going to be executed. Pythia predicts $\mathcal{S}' \subseteq \mathcal{S}$ for each input based on per-cohort signatures [1]. Each cohort $S_i$ constructs a signature $P_i$ from $\mathcal{X}_i$. $P_i$ reflects the current structure of data points in $\mathcal{X}_i$. The idea behind a signature is that $S_i$ is engaged for a given $\mathbf{i}$ once $\mathbf{x}$ can be 'explained' through $P_i$. $S_i$ provides its (locally) created $P_i$ to Pythia, which stores all signatures forming $\mathcal{P} = \{P_i\}_{i=1}^m$. The operation of the Pythia framework is as follows: Given in imputation request (input) $\mathbf{i}$,

- Step 1: Pythia predicts the subset of cohorts $\mathcal{S}' \subseteq \mathcal{S}$ with respect to $\mathcal{P}$
- Step 2: Pythia engages only the cohorts from $S'$ sending the input $\mathbf{i}$ to them.
- Step 3: Each cohort $S_i \in \mathcal{S}'$
  - Step 3.1: $S_i$ invokes locally a MVA and
  - Step 3.2: $S_u$ provides its estimate $\hat{\mathbf{x}}_i$ to Pythia.
- Step 4: Pythia constructs the aggregate estimate $\hat{\mathbf{x}}$ that is sent to the cohorts from $\mathcal{S}'$.
- Step 5: Each $S_i \in \mathcal{S}'$ can exploit $\hat{\mathbf{x}}$ for updating its $P_i$.
- Step 6: Pythia uses $\hat{\mathbf{x}}$ for updating the signatures set $\mathcal{P}$.
- Step 7: Pythia stores the imputation meta-data vector $\mathbf{w}$ to trace back imputed MVs in the system.

## 4.1 Signatures

The general idea of a signature $P_i$ is to represent knowledge on the probability density function (or distribution) of the $\mathcal{X}_i$ of a cohort. Through $P_i$ we optimally quantize the $\mathcal{X}_i$ space of each cohort $S_i$ to estimate the distribution of the underlying $\mathcal{X}_i$. Through adaptive vector quantization, which is achieved by unsupervised competitive learning, we obtain a set of 'representatives' over $\mathcal{X}_i$. The information conveyed by these representatives and their topological neighboring

representatives drives the decision on whether a cohort $S_i$ is eligible for being engaged in a given imputation request $\mathbf{i}$. In this section, we propose two methods for signatures creation based on two adaptive vector quantization algorithms, namely the Adaptive Resonance Theory [26] and the Self Organizing Maps [5].

### 4.1.1 Adaptive Resonance Theory Signature

Each cohort $S_i \in \mathcal{S}$ employs the ART [26], an unsupervised learning model from the competitive learning paradigm, in order to locally construct $P_i$ over $\mathcal{X}_i$. In ART, whose algorithm is shown in Algorithm 1, each $\mathbf{x}_k \in \mathcal{X}_i$ is processed by finding the nearest representative $\mathbf{c}^* \in \mathbb{R}^d$ to $\mathbf{x}_k$, i.e., $\mathbf{c}^* = \arg\min_{\mathbf{c}\in\mathcal{C}_i} \parallel \mathbf{c} - \mathbf{x}_k \parallel$, where $\mathcal{C}_i$ is the set of representatives. Then, it is allowed $\mathbf{x}_k$ to modify/update $\mathbf{c}^*$ only if $\mathbf{c}^*$ is sufficiently close to $\mathbf{x}_k$ ($\mathbf{c}^*$ is said to 'resonate' with $\mathbf{x}_k$) i.e., if $\parallel \mathbf{c}^* - \mathbf{x}_k \parallel \leq \rho_i$ for some *vigilance* $\rho_i > 0$. In this case, $\mathbf{c}^*$ is updated through the rule $\mathbf{c}^* \leftarrow \mathbf{c}^* + \eta_i(\mathbf{x}_k - \mathbf{c}^*)$, where $\eta_i \in (0, 1)$ is a learning rate, which gradually decreases. Otherwise, i.e., $\parallel \mathbf{c}^* - \mathbf{x}_k \parallel > \rho_i$, a new representative $\mathbf{c}$ is formed handling $\mathbf{x}_k$ such that $\mathbf{c} = \mathbf{x}_k$ and $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \{\mathbf{c}\}$.

*Definition 4.* The ART signature $P_i$ of cohort $S_i$ over $\mathcal{X}_i$ is the triple

$$P_i = \langle \mathcal{C}_i, \rho_i, \eta_i \rangle. \quad (1)$$

**ALGORITHM 1:** ART signature creation algorithm at cohort $S_i$

**Input**: $\mathcal{X}_i, \eta_i, \rho_i$
**Output**: $\mathcal{C}_i$
$\mathcal{C}_i = \{\mathbf{x}_1\}$;
**for** $1 < k \leq |\mathcal{X}_i|$ **do**
  $b^* = \parallel \mathbf{c}^* - \mathbf{x}_k \parallel = \min_{\mathbf{c}\in\mathcal{C}_i} \parallel \mathbf{c} - \mathbf{x}_k \parallel$;
  **if** $b^* > \rho_i$ **then**
    $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \{\mathbf{x}_k\}$;
  **else**
    $\mathbf{c}^* \leftarrow \mathbf{c}^* + \eta_i(\mathbf{x}_k - \mathbf{c}^*)$;
  **end**
**end**

*Definition 5.* We say that $\mathbf{x}$ is a *member* of an ART $P_i$ signature, notated by $\mathbf{x} \in P_i$, iff $\min_{\mathbf{c}\in\mathcal{C}_i} \parallel \mathbf{c} - \mathbf{x} \parallel \leq \rho_i$; otherwise, $\mathbf{x} \notin P_i$.

The statement '$\mathbf{x} \in P_i$' denotes that there is at least one $\mathbf{c} \in \mathcal{C}_i$ such that $\mathbf{x}$ is placed close to $\mathbf{c}$ with distance less than $\rho_i$, for instance, the closest representative $\mathbf{c}^*$ to $\mathbf{x}$. The more representatives $\mathbf{c} \in \mathcal{C}_i$ satisfy the criterion $\parallel \mathbf{c} - \mathbf{x} \parallel \leq \rho_i$, the more appropriate $\mathcal{C}_i$ is for $\mathbf{x}$. In this sense, if $\mathbf{x} \in P_i$ then $\mathbf{x}$ can be represented by at least one representative from $\mathcal{X}_i$. Based on this intuition, if $\mathbf{x} \in P_i$, cohort $S_i$ provides a rather good estimate for some missing parts of $\mathbf{x}$ compared to a cohort $S_j$ associated with a $P_j$ for which it holds true that $\mathbf{x} \notin P_j$. The latter case indicates that no representative from $\mathcal{C}_j$ can be a representative point for $\mathbf{x}$.

Since $\rho_i$ represents a threshold of similarity between points and representatives, thus, guiding the ART algorithm in determining when a new representative should be formed, it should depend on $\mathcal{X}_i$. In order to give a physical meaning to $\rho_i$, it is expressed through a set of percentages $\alpha_k \in (0, 1)$ of the ranges between the lowest $x_k^{\min}$ and highest $x_k^{\max}$ values of each dimension $k$ of points in $\mathcal{X}_i$, $k = 1, \ldots, d$. Let $\mathbf{r}_i = [(x_1^{\max} - x_1^{\min}), \ldots, (x_d^{\max} - x_d^{\min})]^\top$ and the diagonal $d \times d$ matrix $\mathbf{A}$ with $\mathbf{A}[k, k] = \alpha_k$. Then $\rho_i = \parallel \mathbf{A}\mathbf{r}_i \parallel$. High $\alpha_k$ values result to a low number of representatives and vice versa. Each $S_i$ determines a $\rho_i$ over $\mathcal{X}_i$, creates $P_i$ through Algorithm 1, and sends $P_i$ to Pythia.

*Remark 3.* When dealing with mixed-type data points, e.g., consisting of categorical, binary, and continuous attributes, we can adopt appropriate distance metrics [29] for the distance between $\mathbf{x}_k$ and $\mathbf{x}_l$ instead of using the Euclidean distance $\parallel \mathbf{x}_k - \mathbf{x}_l \parallel$; this does not spoil the generality of signature creation.

### 4.1.2 Self-Organizing Map Signature

The basic SOM [5], whose algorithm is shown in Algorithm 2, is formally a nonlinear, ordered, smooth mapping of high dimensional vectorial data manifolds, input vector ($\mathbf{x}$), onto the vectorial elements (representatives) of a regular, low dimensional lattice $\mathcal{L}$. SOM implicitly captures the structure of $\mathbf{x}$ and, in particular, identifies the representatives of $\mathbf{x}$ that have similar statistical characteristics in the high-dimensional vector space. The most important characteristic of the SOM is the capability of producing a structured *ordering* of the vectors, i.e., similar vectors in the input space are mapped to neighboring representatives of the map. The incrementally formated representatives estimate the distribution of a dataset $\mathcal{X}$.

Consider the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{X}_i|}$ from the dataset $\mathcal{X}_i$ of cohort $S_i$. The on-line SOM algorithm maps incrementally these vectors into a lattice $\mathcal{L}$ composed of $\ell_i \times \ell_i$ representatives, $\ell_i > 0$, i.e., the number of representatives in $S_i$ is $\ell_i^2$. Hereinafter the parameter $\ell$ is referred to as lattice width. The representatives are linked together by a neighborhood relationship $h(j, j')$ over the indicies $j, j' \in \mathcal{L}$, $j, j' = 1, \ldots, \ell_i^2$. With each representative on $\mathcal{L}$, we associate a representative $\mathbf{c}_j$ of lattice $\mathcal{L}$ with the same dimension as $\mathbf{x}$. The $\ell_i^2$ representatives of $\mathcal{L}$ are initialized randomly among the input vectors. By assuming a general distance metric between $\mathbf{x}$ and $\mathbf{c}_j$, $\mathcal{D}(\mathbf{x}, \mathbf{c}_j)$, the image of $\mathbf{x}$ onto the lattice $\mathcal{L}$ is defined by the winning representative $\mathbf{c}_j^*$ that matches best with $\mathbf{x}$, i.e.,

$$j^* = \arg\min_{j\in\mathcal{L}} \mathcal{D}(\mathbf{x}, \mathbf{c}_j). \quad (2)$$

In the Euclidean space where $\mathcal{D}(\mathbf{x}, \mathbf{c}_j) = \parallel \mathbf{x} - \mathbf{c}_j \parallel_2$, i.e., the 2-norm, the on-line SOM algorithm, at the $k^{th}$ input $\mathbf{x}_k$, $k = 1, \ldots, |\mathcal{X}_i|$, consists of two steps:

- Step 1: (Assignment) Vector $\mathbf{x}_k$ is assigned to a winning representative $\mathbf{c}_j^*$, i.e., $\parallel \mathbf{x}_k - \mathbf{c}_j^* \parallel = \min_{j\in\mathcal{L}} \parallel \mathbf{x}_k - \mathbf{c}_j \parallel$
- Step 2: (Update) All representatives in $\mathcal{L}$ are updated as

$$\mathbf{c}_j = \mathbf{c}_j + \eta(k) h(j, j^*; k) \left(\mathbf{x}_k - \mathbf{c}_j\right). \quad (3)$$

The parameter $\eta(k) \in (0, 1)$ called learning rate is a non-increasing function of $k$. A good choice of $\eta(k)$ improves significantly the convergence of SOM [5]; usually $\eta(k) = \frac{\eta(k-1)}{1+\eta(k-1)}$ with $\eta(0) = 1$. A discussion

about $\eta(k)$ and the choice of an 'optimal' learning rate can be found in [5]. The $h(j, j^*; k)$ is a smoothing Kernel function defined over indicies $j, j^* \in \mathcal{L}$, usually given by the Gaussian neighborhood function:

$$h(j, j^*; k) = \exp\left(-\frac{\| \mathbf{r}_j - \mathbf{r}_j^* \|^2}{2\beta^2(k)}\right).$$

Vectors $\mathbf{r}_j$ and $\mathbf{r}_j^*$ are, respectively, the locations of representatives $\mathbf{c}_j$ and $\mathbf{c}_j^*$ on $\mathcal{L}$. The topological neighborhood is symmetric around the winning representative, which has the maximum value. Parameter $\beta(k)$ is the width of the neighborhood with initial value $\beta_0$ defined as $\beta(k) = \beta_0 \exp(-\frac{k}{T_\beta})$, where $T_\beta$ is a constant. The boundaries of neighborhood $h(j, j^*; k)$ depends on $\beta(k)$. A small width value corresponds to narrow boundaries, while with high width, the boundaries contains more neighbors.

*Remark 4.* Since the size of $|\mathcal{X}_i| = \frac{1}{m}|\mathcal{X}|$ is significantly large thus we can assume that the on-line algorithm of SOM converges. However, if the algorithm has not converged then an additional iteration (or, iterations) is performed until a termination criterion holds true. This criterion, which is compared to a percentage convergence threshold $\epsilon > 0$, refers to the 1-norm between successive estimates of the representatives, i.e., the algorithm converges if $\sum_{j\in\mathcal{L}} \|\mathbf{w}_j(k) - \mathbf{w}_j(k-1)\|_1 < \epsilon \cdot \sum_{j\in\mathcal{L}} \|\mathbf{w}_j(k-1)\|_1$ with $\|\mathbf{w}_j\|_1 = \sum_i^d |w_{ji}|$ and $k > 0$.

Let $\mathcal{C}_i$ be the set of representatives $\{\mathbf{c}_j\}_{j=1}^{\ell_i^2}$ belonging to lattice $\mathcal{L}$.

*Definition 6.* The SOM signature $P_i$ of cohort $S_i$ over $\mathcal{X}_i$ is the tuple

$$P_i = \langle \mathcal{C}_i, \ell_i \rangle. \tag{4}$$

Each cohort $S_i \in \mathcal{S}$ can locally set the number of representatives $\ell_i^2$ in its lattice thus giving it the flexibility to, independently of the other cohorts, determine the 'resolution' (quality) of data space quantization. Evidently, the higher the value of $\ell_i$ the more fine grained the resolution of $\mathcal{X}_i$ quantization gets; however at the expense of higher space requirements. On the other hand, a low value of $\ell_i$ might not be enough to represent the diversity of the data in $\mathcal{X}_i$.

The distance between $\mathbf{x}$ and its winner representative $\mathbf{c}_{j^*}$ plays a significant role on determining how appropriately $\mathbf{x}$ is represented by $P_i$. The fact that $\mathbf{c}_{j^*}$ is the closest representative to $\mathbf{x}$ does not covey any information about how qualitatively the topologically close data space area of $\mathbf{c}_{j^*}$ represents the topologically close data space are to $\mathbf{x}$. Given a smooth distance metric between $\mathbf{x}$ and $\mathbf{c}_{j^*}$ we define as *degree of membership* of $\mathbf{x}$ to $P_i$ the function $\mu_i : \mathbb{R}^d \to [0, 1]$ such that

$$\mu_i(\mathbf{x}) = \exp(- \| \mathbf{x} - \mathbf{c}_{j^*} \|_2^2). \tag{5}$$

A $\mu_i$ value close to unity indicates that $\mathbf{x}$ is topologically very close to its winning representative $\mathbf{c}_{j^*}$ thus $\mathbf{x}$ is believed to be a member of $P_i$ with a high degree. A $\mu_i$ value close to zero indicates that $\mathbf{x}$ is topologically very distant from its winning representative $\mathbf{c}_{j^*}$. Hence, in this case $\mathbf{x}$ is not a member of $P_i$.

*Definition 7.* We say that $\mathbf{x}$ is a *member* of a SOM $P_i$, notated by $\mathbf{x} \in_\mu P_i$, with a degree of $\mu_i(\mathbf{x}) > \epsilon$, $\epsilon \to 0$.

**ALGORITHM 2:** SOM signature creation algorithm at cohort $S_i$

**Input**: $\mathcal{X}_i$, $\ell_i$, $\beta_0$, $T_\beta$
**Output**: $\mathcal{C}_i$
**Initialize** $\mathbf{c}_j$, $j \in \mathcal{L}$;
$\mathcal{C}_i = \{\mathbf{c}_1, \ldots, \mathbf{c}_{\ell_i^2}\}$;
**for** $(1 \le k \le |\mathcal{X}_i|)$ **do**
  $j^* = \arg\min_{j\in\mathcal{L}} \| \mathbf{x}_k - \mathbf{c}_j \|_2$;
  $\mathbf{c}_j \leftarrow \mathbf{c}_j + \eta(k) h(j, j^*; k)(\mathbf{x}_k - \mathbf{c}_j)$, $j \in \mathcal{L}$;
**end**

*Remark 5.* Once Pythia has produced the estimate $\hat{\mathbf{x}}$ given an input $\mathbf{i} = ((\mathbf{z}, \hat{\mathbf{z}}^{\mathbf{m}}), \mathbf{w})$, it updates locally the signatures of those cohorts which were engaged in the imputation process. In case of ART signatures, the reader could refer to [1] which reports on the expected magnitude of change of the representatives in an ART signature due to the estimate. In the case of SOM signatures, the updates are the same with that of the ART signature, provided that the winner representative $\mathbf{c}_{j^*}$ of input $\mathbf{z}$ get updated with a small constant rate $\eta$; the same rate is adopted in ART signatures.

## 4.2 Cohorts prediction

Up to this point, we have shown how to use signatures as a guiding light to select appropriate cohorts for MV imputations. Now, our concern is twofold: MV imputations must be (i) low cost and (ii) high accuracy. Low cost (once signature processing is performed) refers to the communication cost between Pythia and cohorts and to the cost of running MVAs at cohorts. High accuracy refers to low RMSE. Therefore, in our previous work we presented algorithms with these in mind. Now, we further propose two cohorts prediction algorithms corresponding to ART- and SOM-signatures, which engage the top-$\mathcal{K}$ relevant cohorts for an imputation request, $1 \le \mathcal{K} \le m$. Under this class of algorithms, Pythia is not involved in producing the (final) estimate $\hat{\mathbf{x}}$, instead, only the top-$\mathcal{K}$ best cohorts are engaged for doing this locally. Pythia communicates only with these cohorts, which run the MVA in parallel, thus, this optimizes our cost metric. Note, the reader could refer also to the accuracy-aware class of algorithms in our previous work [1], in which Pythia is (merely) engaged in the final estimate.

### 4.2.1 ART signature cohort prediction

For simplicity consider the top-1 (best cohort) scheme, i.e., $\mathcal{K} = 1$. Given imputation request $\mathbf{i}$ and a set of ART signatures, Pythia determines the best cohort $S^* \in \mathcal{S}$ with $P^* = \langle \mathcal{C}^*, \rho^*, \eta^* \rangle$ such that the following criteria hold true:

- Criterion C1: $\mathbf{c}^* = \arg\min_{\mathbf{c}\in\cup_{i=1}^m \mathcal{C}_i} \| \mathbf{c} - \mathbf{z} \|$ and $\mathbf{c}^* = \arg\min_{\mathbf{c}\in\mathcal{C}^*} \| \mathbf{c} - \mathbf{z} \|$, i.e., $\mathbf{c}^* \in \mathcal{C}^*$ is the closest representative to $\mathbf{z}$ among all representatives from all signatures, and
- Criterion C2: $\mathbf{z} \in P^*$, i.e., the vector $\mathbf{z}$ is member of the ART signature $P^*$.

Note that $\mathbf{z} \in \mathbb{R}^{d'}$ with $0 < d' = \sum_{k=1}^d w_k < d$ provided that $\mathbf{x}$ contains $d - d'$ MVs. In order to evaluate '$\mathbf{z} \in P^*$'

Pythia calculates $\rho^{*(d')} \leq \rho^*$ associated with the $n$ dimensions of $\mathbf{q}^*$ corresponding to the $n$ non-MVs. Then, it checks if $\| \mathbf{c}^* - \mathbf{z} \| \leq \rho^{*(d')}$ dealing only with the $d'$ dimensions of $\mathbf{c}^*$. Pythia engages only the best cohort $S^*$, which produces the final $\hat{\mathbf{x}}$. If there is no cohort that satisfies criteria C1 and C2, then Pythia engages the cohort that satisfies only criterion C1. If $\mathcal{K} > 1$ one can repeat the above criteria for the top $\mathcal{K}$ cohorts ranked with the distance between the corresponding $\mathbf{c}_j^*$ and $\mathbf{z}$, $1 \leq j \leq \mathcal{K} < m$. In this case the final $\hat{\mathbf{x}}$ is produced by aggregating all $\hat{\mathbf{x}}_j$ with $1 \leq j \leq \mathcal{K}$.

#### 4.2.2 SOM signature cohort prediction

Consider again for simplicity the top-1 (best cohort) scheme. Given imputation request $\mathbf{i}$ and a set of SOM signatures, Pythia determines the best cohort $S^* \in \mathcal{S}$ as follows:

- Step 1: Find the winner prototype $\mathbf{c}_i^* = \arg\min_{\mathbf{c} \in \mathcal{C}_i} \| \mathbf{z} - \mathbf{c} \|$ from signature $S_i$, $\forall i$.
- Step 2: Define a membership indicator $I_i(\mathbf{z}) = 1$ if $\mu_i(\mathbf{z}) > \epsilon$; otherwise 0, $\forall i$.
- Step 3: Calculate the normalized membership degree

$$\tilde{\mu}_i(\mathbf{z}) = \frac{\mu_i(\mathbf{z}) I_i(\mathbf{z})}{\sum_{k=1}^{m} \mu_k(\mathbf{z}) I_k(\mathbf{z})}$$

and select the cohort with the maximum value of $\tilde{\mu}$.

If for all cohorts $S_i \in \mathcal{S}$, it holds true that $I_i(\mathbf{z}) = 0$, i.e., $\mathbf{z}$ cannot be represented by any winner representative from all signatures, then Pythia engages the cohort whose winner representative is the closest to input $\mathbf{z}$ among all winner representatives (from all signatures). If $\mathcal{K} > 1$, then Pythia engages (at most) the top $\mathcal{K}$ cohorts ranked with respect to the $\tilde{\mu}$ value. In this case the final $\hat{\mathbf{x}}$ is produced by aggregating all $\hat{\mathbf{x}}_j$ with $1 \leq j \leq \mathcal{K}$.

### 4.3 Pythia asymptotic complexity

Let $\xi$ be the average number of representatives per ART signature. In a SOM signature we have $\ell^2$ representatives, assuming that the latices from all signatures have the same number of representatives. For the top-$\mathcal{K}$ class of algorithms for cohorts prediction, we adopt a $d$-dimensional tree structure over all representatives from all ART and SOM signatures in $\mathcal{P}$. Given imputation input $\mathbf{i}$, Pythia performs a 1NN search with $O(d \log(m\xi))$ and $O(d \log(m\ell))$ time since it searches over all representatives in ART and SOM signatures, respectively, from all signatures $\cup_{i=1}^{m} \mathcal{C}_i$ given that $\mathcal{K} = 1$. Pythia requires $O(md\xi)$ and $O(md\ell^2)$ space, respectively, for ART and SOM signatures. Pythia requires $O(\mathcal{K})$ communication with cohorts from $\mathcal{S}$.

### 4.4 Limitation of the Signatures Algorithms

The main limitations of the ART- and SOM-based signatures algorithms are the nature of their parameters that need to be defined in advance. In the SOM algorithm, a fixed number of representatives in the lattice $\mathcal{L}$ must be determined beforehand. In this context, a good choice of the lattice size (number of representatives $\ell^2$) has a significant impact on the overall quality of the derived self-organized clusters. A small $\ell$ value, i.e., a low number of representatives, might not be sufficient enough to represent the topological structure of the data, thus capturing their statistical characteristics. On the other hand, a huge number of representatives scatter similar data items into more than one clusters. In addition, this comes with a significant number of 'training' samples for the SOM algorithm to converge. Likewise, a good choice of the vigilance parameter $\rho$ plays an important role on incremental data space partitioning in the ART algorithm. A relatively high vigilance value produces few clusters/representatives, in which 'non-similar' data points are grouped together. On the other hand, a small vigilance value yields the formation of a high number of representatives that might contain few data points and increase the signature size, i.e., the representatives set $\mathcal{C}$, per cohort. An appropriate determination of the vigilance $\rho$ and the lattice size $\ell$ highly depends on the statistical properties of the underlying data in each cohort, individually. That is a 'good' vigilance or lattice size for one cohort might not be suitable for another one. Therefore, a fine tuning of these parameters is essential in order to get optimal quality of data space partitioning. The reader could also refer to for a discussion on good values for vigilance [27] and lattice size [28] corresponding to the ART and SOM algorithms, respectively.

## 5. PERFORMANCE EVALUATION

### 5.1 Experimental Setup

We conducted an extensive series of experiments to assess the performance of Godzilla and Pythia's over the best cohort scheme ($\mathcal{K} = 1$) on a real dataset. The dataset is adopted from the UCI Machine Learning Repository [33]. We selected randomly 1.2 million real valued 50-dimensional vectors ($d = 50$) that have no MVs from physical activity monitoring features. We synthetically produce MVs randomly and independently marked as missing with probability $q \in (0, 1)$. Therefore, we expected $|\mathcal{X}| \sum_{k=1}^{d-1} \binom{d}{k} q^k (1-q)^{d-k}$ points with MVs. We set $q = 0.3$, which is a relatively high probability of MVs per dimension, thus, being able to test Pythia's robustness in terms of accuracy.

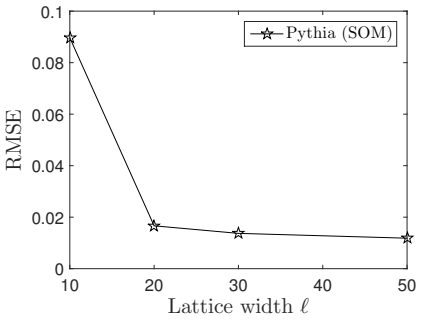


**Figure 1: RMSE vs. lattice width $\ell$ for SOM signature.**

On average, an ART signature $P_i$ contains 0.05% of points of the entire dataset $\mathcal{X}$ (this amount refers to the number of representatives stored in Pythia per cohort); whereas, for SOM we varied the size of the lattice $\ell \in \{10, 20, 30, 50\}$. As shown in Fig. 1, we obtained no significant change in the root mean squared error (defined formally later) when $\ell$ increases from 20 to 50. This implies that there is no notable trade off between accuracy and lattice size, thus, we set $\ell = 20$ to increase the efficiency of Pythia in the

cohorts prediction phase. When $\ell = 20$, each SOM signature (on average) contains 0.01% of points of the entire set $\mathcal{X}$. Furthermore, we set the convergence threshold $\epsilon = 0.01$ and observed that (on average) the SOM on each cohort has converged after 10,000 input vectors. This denotes the avoidance of unnecessary extra iterations that might run while representatives have already converged and also the fact that the size of each cohort's dataset is adequately huge for the on-line learning algorithm of SOM. We also adjusted the initial width of the neighborhood function boundaries $\beta_0 = \ell$ (the width of the lattice) so that these boundaries could contain almost all representatives during the first few iterations (note: it decays exponentially with each iteration). In SOM, we used initial learning rate $\eta(1) = 0.5$, which gradually decreases as discussed in Section 4.1.2. In ART we used learning rate $\eta = 0.1$. Moreover, we set the range percentage $\alpha_k = \alpha = 0.2$, $1 \leq k \leq d$, in ART for all dimensions in order to construct the $\mathcal{P}$ set. We run all experiments 10,000 times and took their average values for all performance metrics. The number of cohorts $m$ ranges in $\{20, \ldots, 100\}$. Pythia's cohorts prediction algorithms and the MVAs reported in Section 3.2 were developed in Java. Table 1 summarizes the parameter values used in our experiments.

| Parameter | Notation | Value/Range |
|---|---|---|
| $d$ | dimension | 50 |
| $m$ | number of cohorts | $\{20, 40, 60, 80, 100\}$ |
| $\epsilon$ | convergence threshold | 0.01 |
| $q$ | MV probability | 0.3 |
| $\lvert\mathcal{X}\rvert$ | dataset size | $1,2 \cdot 10^6$ |
| $T$ | imputation requests | $10^4$ |
| $\alpha$ | vigilance range in ART | 0.2 |
| $\eta$ | learning rate in ART | 0.1 |
| $\ell$ | lattice width in SOM | {10, 20, 30, 50} |
| $\beta_0$ | initial width in SOM | 20 |

**Table 1: Experimental parameters.**

## 5.2 Performance metrics

Our metrics include *efficiency metrics* and *accuracy metrics*. A scale-out system consisting of $m$ cohorts affords two types of parallelism: *intra-imputation* and *inter-imputation* parallelism. The former refers to the capability of processing any single imputation using a number of cohorts in parallel, each accessing a dataset partition. The latter refers to the systems' capability of running in parallel a number of imputations, each of which engages a subset of cohorts. It is crucial to note that Godzilla affords neither of these parallelism. This latter scenario is particularly important as typically a system is presented with a (large) batch of (vector-) inputs, each with missing values and the goal is to impute all input vectors in the batch as quickly/scalably as possible. Given this, our efficiency metrics embody various efficiency aspects impacting scalability.

First, we report on *imputation latency*, defined as the time (in seconds) a system (i.e., Godzilla or Pythia) requires to impute a single input (vector) using a MVA. The rate of latency increase as dataset sizes grow is a strong aspect of scalability. In Pythia, latency refers to the time to predict best cohort $S^*$, plus the latency to run MVA in parallel at the engaged cohort.

*Imputation speedup* is defined as the ratio of Godzilla latency over Pythia latency; it indicates how much a system is faster than Godzilla for a single imputation. The linear imputation speedup ratio is $m$.

We measure *imputation accuracy* using the RMSE metric, i.e., the root-mean squared difference between actual vector $\mathbf{x}_a$ and estimated vector $\hat{\mathbf{x}}$ after $T$ imputation requests:

$$RMSE = \left( \frac{1}{T} \sum_{t=1}^{T} \frac{\sum_{k=1}^{d} w_{tk} (x_{(a)tk} - \hat{x}_{tk})^2}{\sum_{k=1}^{d} w_{tk}} \right)^{1/2}. \quad (6)$$

Finally, we measure the *storage* metric $b$ for Pythia adopting either ART or SOM signatures. Specifically, this metric refers to the total number of representatives in the ART signature and total number of representatives in the SOM signature. For the latter case, this corresponds to $b = \ell^2 m$ given that all SOM signatures adopt the same lattice width $\ell$. In the ART signature case, each signature has constructed different number of representatives, which depends on the underlying data distribution of each cohort's dataset. If $\xi_i$ is the number of representatives of an ART signature $P_i$ then the storage metric corresponds to $b = \sum_{i=1}^{m} \xi_i$.

## 5.3 Imputation efficiency

Figures 2 and 3 show the imputation speedup against number of cohorts $m$ using the EM and KNN imputation algorithms, respectively, utilizing Pythia with ART and SOM signatures. In KNN at Figure 3, a slightly super linear speedup is observed for all number of cohorts, e.g., speedup ratio is a slightly greater than $m$ when $m$ cohorts are engaged in the imputation process. Super linear speedup is also noticed for EM at Figure 2 when the number of cohorts increases. This is due to the fact that the MVAs algorithms (KNN and EM) highly depend on the size of dataset $\mathcal{X}$, i.e., their computational complexity is proportional to $O(|\mathcal{X}|)$. Hence, a portion of dataset $|\mathcal{X}_i| = \frac{1}{m}|\mathcal{X}|$ which is processed by an imputation algorithm over a cohort $S_i$ yields a decrease in the corresponding latency of imputation by at least a factor of $m$. More interestingly, the more demanding (in terms of computational effort) an imputation algorithm is, the more benefit we get if we run it over a portion of the entire dataset; see also Remarks 1 and 2 in Section 3.2 for the computational complexity of the imputation algorithms. Overall, Pythia has achieved substantial speedup using ART and SOM signatures in both MVAs.

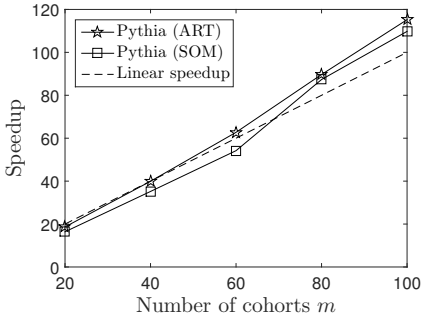


**Figure 2: Speedup vs. number of cohorts $m$ for ART and SOM signatures using EM.**

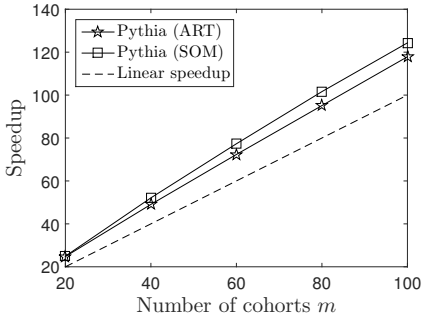


**Figure 3: Speedup vs. number of cohorts $m$ for ART and SOM signatures using KNN, $K = 10$.**

Figures 4 and 5 show the latency in seconds of Godzilla and Pythia for ART and SOM signatures, respectively, in logarithmic scale for different number of cohorts $m$ against different number of dataset size $|\mathcal{X}|$. The dataset size varies from $10^5$ to $7 \cdot 10^5$ of 50-dimensional vectors. Godzilla struggles with increasing dataset size. Specifically, for Godzilla, by an increasing dataset size, a supper linear increase in latency is observed in both Figures 4 and 5. Pythia scales nicely with its latency increasing linearly utilizing both ART and SOM signatures. Indicatively, given a dataset size $|\mathcal{X}| = 5 \cdot 10^5$, Godzilla requires 60 seconds and Pythia (with $m = 100$ and SOM signature) requires 0.2 seconds to impute an input, respectively. Moreover, when the number of cohorts increases, a sub-linear increase in latency is obtained for Pythia. Pythia can easily handle large datasets if more cohorts are available to scale to big data missing values. Our results up to now clearly make a strong case for the scale-out advantages of the Pythia framework.

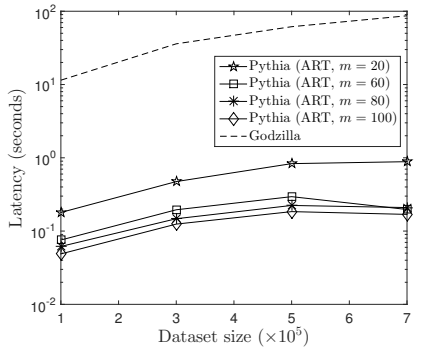


**Figure 4: Latency in seconds vs. dataset size $\times 10^5$ for Pythia ART and Godzilla with different number of cohorts $m = \{20, 60, 80, 100\}$.**

## 5.4 Imputation accuracy

We now experiment with the expected achieved imputation accuracy utilizing the most relevant cohorts in parallel. We focus on the best cohort prediction scheme where Pythia based on ART and SOM signatures engages only the best cohort out of the $m$ cohorts. Figures 6 and 7 show the RMSE against the number of cohorts $m$ using KNN and EM, respectively. Pythia (in both ART and SOM signatures) using KNN, obtains a relatively low RMSE (on average for all $m$)

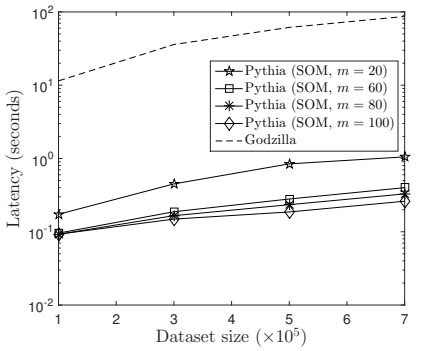


**Figure 5: Latency in seconds vs. dataset size $\times 10^5$ for Pythia SOM and Godzilla with different number of cohorts $m = \{20, 60, 80, 100\}$.**

as observed in Figure 6. The same RMSE was also achieved by Godzilla. Accordingly, there was no significant statistical difference observed between the accuracy of Pythia and Godzilla. Please note that Pythia adopting the best cohort prediction scheme and using KNN yields a higher RMSE compared to Godzilla. This is due to the fact that, using KNN, Godzilla would provide the *global nearest* K points, whereas in Pythia, the best cohort, even when storing irrelevant data, will be contributing its local nearest K points. The latter necessarily implies that a single-cohort imputation might involve points, which adversely affect imputation accuracy. Using EM, however, the lowest and highest RMSE were achieved by Pythia (base on ART signature), 0.11 and 0.16, respectively; see Figure 7. Comparing against the RMSE of Godzilla, 0.12, still there is no significant difference. This indicates that Pythia has comparable RMSE with Godzilla regardless of the imputation algorithms and the signature creation algorithms.

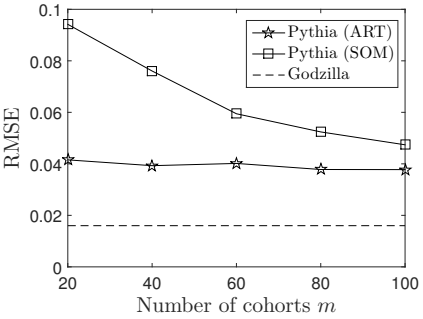


**Figure 6: RMSE vs. number of cohorts $m$ for Pythia ART, Pythia SOM and Godzilla using KNN.**

Finally, Table 2 shows the total number of representatives $b = \sum_{i=1}^{m} \xi_i$ and representatives $b = \ell^2 m$ in ART and SOM signatures that are stored in Pythia for making predictions against the number of cohorts $m$ for $\alpha = 0.2$ and $\ell = 20$, respectively; we also show the percentage storage $\frac{b}{|\mathcal{X}|}$ with respect to the entire dataset size. One can observe that, in the case of the ART signature, the average number of representatives per cohort decreases with the number of cohorts. This indicates the robust behavior of Pythia in terms

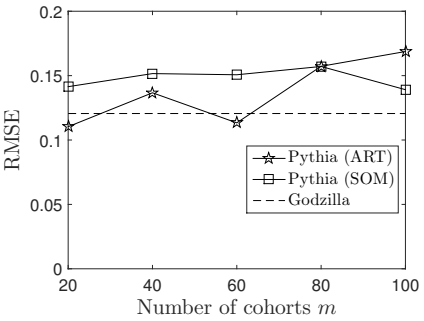


**Figure 7: RMSE vs. number of cohorts $m$ for Pythia ART, Pythia SOM and Godzilla using EM.**

of storage requirements. Specifically, the total number of representatives remains the same for all values of $m$. In the case of the SOM signature, the number of representatives is explicitly controlled by the $\ell$ parameter and is not determined by the underlying data distribution. Given a number of cohorts $m \leq 50$, by comparing the imputation accuracy of both signature methods, ART and SOM (see also Figures 6 and 7), we observe that a Pythia variant with SOM signature requires 50% less storage than a Pythia variant with ART signature for achieving quite similar accuracy levels for both KNN and EM imputation algorithms. In this case, we can conclude that, when deploying a relatively low number of cohorts for missing values imputation, the adoption of SOM signature is preferable than an ART signature scheme. On the other hand, for a relatively high number of deployed cohorts, both variants (ART and SOM) can be applied with ART signature resulting into slightly better accuracy performance and SOM signature requiring 2% less storage.

| | ART | ART pct. | SOM | SOM pct. |
|---|---|---|---|---|
| $m$ | $b = \sum_{i=1}^{m} \xi_i$ | $(\%) \frac{b}{\lvert\mathcal{X}\rvert}$ | $b = \ell^2 m$ | $(\%) \frac{b}{\lvert\mathcal{X}\rvert}$ |
| 20 | 40,047 | 3.09 | 8,000 | 0.61 |
| 40 | 40,630 | 3.13 | 16,000 | 1.23 |
| 60 | 40,813 | 3.15 | 24,000 | 1.85 |
| 80 | 41,153 | 3.17 | 32,000 | 2.47 |
| 100 | 41,190 | 3.18 | 40,000 | 3.08 |

**Table 2: Storage requirement in Pythia.**

# 6. CONCLUSIONS & FUTURE RESEARCH

We have tackled the problem of scaling out MV imputations, a common problem in many big data applications. We studied and developed some of the fundamentals of the problem, based on which we developed Pythia, a framework and algorithms designed for this aim. The Pythia framework is drastically different, as it on the one hand avoids the need to access all cohorts (and all associated costs for communication and for running MVAs at all cohorts), while on the other can achieve better or comparable MV imputation accuracy, compared to centralized solutions. The major purpose of this paper is to examine whether the Pythia framework introduced in [1] is robust and independent of any MVA and signature creation algorithm. Specifically, through our comprehensive experiments in this paper we showed that Pythia can provide drastically better efficiency/scalability and competitive accuracy compared to a centralized approach (Godzilla). This is achieved by introducing the idea of the signature, a statistical learning structure over the distributed datasets. The signatures are exploited by Pythia to decide on the most appropriate subset of data nodes to be access upon a stream of imputation requests. We proposed two methods for constructing a signature structure based on adaptive vector quantization and competitive learning. The central conclusions of our study are as follows. Godzilla suffers from obvious severe scalability and efficiency limitations. Hence, Pythia is deemed as an appropriate solution since it not only significantly outperforms Godzilla in terms of efficiency (storage, latency) but, also, performs as good as Godzilla with respect to imputation accuracy. Moreover, the Pythia is independent of any particular imputation algorithm and signature construction algorithm. This renders the Pythia framework capable of coping with MV imputation requests, which are directed to subsets of cohorts for local MVA invocations.

## 6.1 Discussion on Limitations

A primary functionality of the Pythia is the swiftly determination of the most relevant subset of cohorts to direct the incoming MV imputation request (input vector $\mathbf{i}$) based on the signatures. In this context, the Pythia node decides on the closest representative $\mathbf{c}$ of each cohort's signature by calculating the Euclidean distance over the dimensions of the input that contain non-missing values. In the case of high-dimensional data ($d$ is relatively high) and when the probability of a missing value is low ($p$ is relatively low) then the Pythia node has to calculate the Euclidean distance over $(1-p)d$ dimensions. In that case, the Euclidean distance metric might change in some non-obvious ways [31]. Specifically, as it has been argued in [32], under certain reasonable assumptions on the underlying data distribution, the ratio of the distances of the nearest and farthest neighbors to a given target in a high dimensional space is almost unity for a wide variety of data distributions and distance functions. In such a case, the nearest neighbor identification (which refers to the closest representative in our context) becomes ill defined, since the contrast between the distances to different data points does not exist. In such cases, even the concept of proximity may not be meaningful from a qualitative perspective: a problem which is even more fundamental than the performance degradation of high dimensional algorithms. In our case, the $L_k = (\sum_{j=1}^{d}(i_j - c_j)^k)^{1/k}$ norm with $k = 2$, i.e., $L_2 = (\sum_{j=1}^{d}(i_j - c_j)^2)^{1/2}$ is susceptible to the dimensionality curse for many classes of data distributions [32]. Specifically, based on the analysis in [32], the relative contrast of the distance of an input vector $\mathbf{i}$ with a representative vector $\mathbf{c}$ depends heavily on the adopted $L_k$ distance metric. This provides considerable evidence that the meaningfulness of the $L_k$ norm worsens faster with increasing dimensionality for higher values of $k$. Thus, in our problem with a high value of the dimensionality $d$, it may be preferable to use lower values of $k$. This means that the $L_1$ distance metric (i.e., the Manhattan distance metric) is the most preferable for high dimensional applications, followed by the Euclidean ($L_2$), then the $L_3$ metric, and so on. Encouraged by the analysis in [32], we are planning, as a future work, to examine the behavior of *fractional* distance metrics for the distance between $\mathbf{i}$ and $\mathbf{c}$, in which $k$ is allowed to

be a fraction smaller than unity. Further, the limitations of the adopted algorithms for the signatures construction and the possible directions for dealing with these limitations are discussed in Section 4.4.

## 6.2 Future Research

Apart from the future work discussed in Sections 4.4 and 6.1 triggered by the limiations of the Pythia framework, we further plan to incorporate to our research agenda the following items. This work has shown that the Pythia framework improves not only the imputation efficiency but also achieves at least the same imputation accuracy comparing against the performance of the Godzilla variant. The experimental evaluation focuses on stationary data. In stationary data, the underlying probability distribution function does not change over time. Hence, the signatures of the Pythia do not change frequently so that they remain as a reliable representation (through the derived clusters). In a non-stationary data environment, e.g., an environment dealing with data streams, such distribution function (estimated through clusters) changes over time swiftly [30]. In this context, *new* clusters can be formated and *existing* clusters have to be updated to follow the data streams trend. Therefore, our future research items include the enhancement of the Pythia framework to access only a relevant part of the whole dataset in order to improve scalability, efficiency and prediction accuracy in a dynamic environment with ever-changing data patterns. As the probability distribution function of the data streams changes frequently, we plan to investigate methods for updating the Pythia signatures to efficiently support MV imputation requests.

# 7. REFERENCES


[1] C. Anagnostopoulos, *et al*, Scaling out big data missing value imputations: pythia vs. godzilla. *Proc. 20th ACM SIGKDD (KDD' 14) International Conference on Knowledge discovery and data mining. ACM, New York, NY, USA, pp.651–660.*

[2] X. Su, *et al*, 'Using Classifier-Based Nominal Imputation to Improve Machine Learning', *Proc. 15th PAKDD*, Part I, LNAI 6634, pp. 124–135, 2011.

[3] A. Farhangfar, *et al*, 'Impact of imputation of missing values on classification error for discrete data', *Pattern Recognition*, 41(12): 3692–3705, Dec 2008.

[4] M.T. Asif, *et al*, 'Low–Dimensional Models for Missing Data Imputation in Road Networks', *Proc. 38th IEEE ICASSP*, pp.3527–3531, 2013.

[5] T. Kohonen. 2001. 'Self-Organizing Maps' (3rd ed.). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[6] E.C. Chi, *et al*, 'Genotype imputation via matrix completion', *Genome Research*, 23(3):509–18, Mar 2013.

[7] I.B. Aydilek, *et al*, 'A novel hybrid appoach to estimating missing values in databases using $k$–nearest neighbors and neural networks', *Innovative Computing, Information and Control*, 8(7A): 1349–4198, Jul 2012.

[8] A. Farhangfar, *et al*, 'A Novel Framework for Imputation of Missing Values in Databases', *IEEE Trans. Sys. Man Cyber. (A)*, 37(5): 692–709, Sep 2007.

[9] K. Lakshminarayan, *et al*, 'Imputation of missing data in industrial databases', *Appl. Intell.*, 11(3): 259–275, Nov / Dec 1999.

[10] L. A. Kurgan, *et al*, 'Mining the cystic fibrosis data', J. Zurada & M. Kantardzic (Eds.), *Next Generation of Data–Mining Applications*, IEEE Press, 415–444, 2005.

[11] A.W. Liew, *et al*, 'Missing value imputation for gene expression data: computational techniques to recover missing data from available information', *Brief. Bioinform.*, 12(5): 498–513, Sep 2011.

[12] J. Dean, *et al*, 'MapReduce: Simplified Data Processing on Large Clusters', *Proc. USENIX OSDI*, 2004.

[13] S. Ghemawat, *et al*, 'The Google File System', *Proc. ACM SOSP*, 2003.

[14] C-T. Chu, *et al*, 'Map-Reduce for Machine Learning on Multicore', *NIPS 19*, MIT press, 281–288, 2006.

[15] C. K. Enders, 'Applied Missing Data Analysis', *Guilford Press*, NY, 2010.

[16] D. W. Joenssen, *et al*, 'Hot Deck Methods for Imputing Missing Data', *Proc. 8th MLDM* , LNCS 7376, pp.63–75, 2012.

[17] O. Troyanskaya, *et al*, 'Missing value estimation methods for DNA microarrays', *Bioinformatics*, 17(6):520–525, 2001.

[18] R.J. Little, *et al*, 'Statistical Analysis with Missing Data', *Wiley*, NY, 1987.

[19] T.E. Raghunathan, *et al*, 'A multivariate technique for multiply imputing missing values using a sequence of regression models', *Survey Methodology*, 27(1):85–95, 2001.

[20] D.B. Rubin, 'Multiple Imputation After 18+ Years', *J. of the American Statistical Association*, 91(434):473–489, 1996.

[21] L. Li, *et al*, 'DynaMMo: mining and summarization of coevolving sequences with missing values', *Proc. 15th KDD*, 527–534, 2009.

[22] S. Yang, *et al*, 'Online recovery of missing values in vital signs data streams using low–rank matrix completion', *Proc. 11th IEEE ICMLA*, 281–287, 2012.

[23] M. Ouyang, *et al*, 'Gaussian mixture clustering and imputation of microarray data', *Bioinformatics*, 20(6): 917–923, Apr 2004.

[24] T. Aittokallio, *et al*, 'Dealing with missing values in large-scale studies: microarray data imputation and beyond' *Brief. Bioinform.* 11(2):253–264, 2010.

[25] D-W. Kim, *et al*, 'Iterative Clustering Analysis for Grouping Missing Data in Gene Expression Profiles', *Proc. PAKDD 2006*, LNAI 3918, pp.129–138, 2006.

[26] G. A. Carpenter, *et al*, 'The ART of adaptive pattern recognition by a self–organizing neural network', *IEEE Computer*, 21(3): 77–88, Mar 1988.

[27] L. Meng, *et al*, 'Vigilance adaptation in adaptive resonance theory' *Neural Networks (IJCNN), IEEE International Joint Conference on*, pp.1–7, 2013.

[28] Y. Prudent, *et al* 'An incremental growing neural gas learns topologies' *Neural Networks (IJCNN), IEEE International Joint Conference on* , vol.2, no., pp.1211–1216, 2005.

[29] A. Ahmad, *et al*, 'A $k$–mean clustering algorithm for mixed numeric and categorical data' *Data & Knowledge Engineering*, 63(2):503–527, 2007.

[30] Y. Chen and L. Tu. Density-based clustering for

real-time stream data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 133–142, ACM, 2007.

[31] C. Aggarwal, *et al*, On the surprising behavior of distance metrics in high dimensional space. *Springer. 2001.*

[32] K. Beyer, *et al.* 'When is Nearest Neighbors Meaningful?' ICDT Conference Proceedings, 1999.

[33] K. Bache, *et al*, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml] Irvine, Uni. of California, School of Inform. and Comp. Sci., 2013.