2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)

From Big data to Smart Data with the K-Nearest Neighbours algorithm

Isaac Triguero, Jesús Maillo, Julián Luengo, Salvador García, and Francisco Herrera

***Abstract*—The k-nearest neighbours algorithm is one of the most widely used data mining models because of its simplicity and accurate results. However, when it comes to deal with big datasets, with potentially noisy and missing information, this technique becomes ineffective and inefficient. Due to its drawbacks to tackle large amounts of imperfect data, plenty of research has aimed at improving this algorithm by means of data preprocessing techniques. These weaknesses have turned out as strengths and the k-nearest neighbours rule has become a core model to actually detect and correct imperfect data, eliminating noisy and redundant data, as well as correcting missing values.**

In this work, we delve and review the role of the k nearest neighbour algorithm can play to come up with smart data from big datasets. We analyse how this model is affected by the big data problem, but at the same time, how it can be used to transform raw data into useful data. Concretely, we discuss the benefits of recent big data technologies (Hadoop and Spark) to enable this model to address large amounts of data, as well as the usefulness of prototype reduction and missing values imputation techniques based on it. As a result, guidelines on the use of the k-nearest neighbour to obtain Smart data are provided and new potential research trends are discussed.

***Index Terms*—k-Nearest Neighbours, Prototype Reduction, Data Preprocessing, Smart Data, Big Data.**

I. Introduction

THE great advances on technology allow us to automatically gather information in a relatively inexpensive manner. This has resulted in a severe increment of the amount of available data. The impact of handling this data may be reflected in competitive benefits for companies and firms or important discoveries in multiple science fields [1]. Nevertheless, both companies and researchers are facing major difficulties to store and analyse vast amounts of data. These issues are being referred as the big data problem.

Extracting valuable knowledge from big data datasets with machine learning techniques [2] may provide more accurate models than ever before. The leverage of recent advances achieved in distributed technologies enables us to discover unknown patterns or hidden relations in data-intensive applications [3] more rapidly. However, most of the existing methods fail to directly tackle the new data space, as the issues posed by (real-world) complex data go beyond computational complexity, and big data mining techniques are confronted with multiple challenges w.r.t. scalability, dimensionality, inaccurate data (noisy, or incomplete), etc.

I. Triguero is with the School of Computer Science, University of Nottingham, United Kingdom

J. Maillo, J. Luengo, S. García, and F. Herrera are with Department of Computer Science and Artificial Intelligence, University of Granada, Spain

The term of smart data [4] is increasingly being used to refer to the challenge of transforming raw data into data that can be later processed to obtain valuable insights [5]. According to the report presented by Gartner, Inc in 2015[1], smart data discovery is "a next-generation data discovery capability that provides business users or citizen data scientists with insights from advanced analytics". Therefore, smart data discovery consists of filtering big data holding useful information, becoming a subset of data (big or not) that is important for companies and researchers. Obtaining a reduced/filtered amount of data may imply a great reduction in terms of data storage costs (i.e. less disk space requirements), as well as a great impact in the successful application of data mining techniques.

Data preprocessing [6] is clearly linked to (and resembles) the smart data concept. This is one of the most relevant stages in data mining that aims to clean and correct original data in order to apply machine learning algorithms faster and more accurately. As such, these techniques should enable data mining algorithms to address big data problems with greater ease, but these methods are also affected by the increase in size and complexity of datasets and they may be unable to provide a preprocessed dataset in a bounded time.

A well-known data mining technique is the k-Nearest Neighbour algorithm (k-NN) [7]. This is based on the concept of similarity, and in classification problems, for example, it means that patterns that are similar have to be assigned to the same class. As a lazy learning algorithm, it does not explicitly perform a training phase, and unseen cases are classified by finding the class labels of the closest instances to them. Due to this way of working, the k-NN algorithm may suffer from several disadvantages to tackle big datasets, such as high computational cost, high storage requirements, sensitivity to noise and inability to deal with incomplete information.

Several distributed alternatives have been proposed to enable k-NN to handle big data [8], [9]. Most of them are based on the MapReduce [10] programming paradigm, and its open-source implementation in Hadoop, to transparently parallelise the k-NN processing, alleviating memory and computational cost limitations. Very recently, in [11], a new design that goes beyond the standard Hadoop MapReduce approach provides a flexible scheme to classify big amounts of unseen cases against a big training dataset, based on Apache Spark [12].

Another way to tackle some weaknesses of the k-NN algorithm is by means of data reduction models. These techniques reduce the original training data at the level of instances

[1]Smart Data Discovery Will Enable a New Class of Citizen Data Scientist. https://www.gartner.com/doc/3084217/smart-data-discovery-enable-new

978-1-5090-5880-8/16 $31.00 © 2016 IEEE
DOI 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.177

(prototype reduction (PR) [13], [14]), or at the attributes level (feature selection [15]), eliminating redundant and noisy information. In the literature, we find many works in which the k-NN algorithm takes an important part of the data preprocessing process, especially those using evolutionary algorithms [16].

Dealing with incomplete information is a big challenge for most data mining techniques [17]. The k-NN model is not an exception and it may not be able to compute distances between examples containing missing values. However, the underlying idea of the k-NN has been used to impute missing values (KNNI, [18]) based on the k nearest neighbours.

Interestingly, the resulting preprocessed dataset provided by the above approaches can be used not only by the k-NN algorithm but also by other kind of algorithms. This work discusses the applications of the k-NN algorithm to come up with smart data. First, we will detail the main challenges to deal with big data and existing solutions so far (Section II). Next, we will dig into how the k-NN algorithm has been used as a core model for data preprocessing (Section III). Later, we will show some interesting experiments, discussing how transform big data into smart data (Section IV). Finally, we will point out open research lines (Section V).

II. k-NN algorithm and Big Data

This section introduces the k-NN algorithm and its main drawbacks to work with big data (Section II-A), as well as new designs based on big data technologies to speed up its processing (Section II-A).

A. k-NN background

The k-NN algorithm is a non-parametric method that can manage classification and regression problems. This section defines the k-NN rule and its drawbacks to deal with big datasets. A formal notation for k-NN is as follows:

Let TR be a training dataset and TS a test set, they are formed by a determined number **n** and **t** of samples, respectively. Each sample $\mathbf{x}_p$ is a vector $(\mathbf{x}_{p1}, \mathbf{x}_{p2}, ..., \mathbf{x}_{pD}, \omega)$, where, $\mathbf{x}_{pf}$ is the value of the f-th feature of the p-th sample. Every sample of TR belongs to a known class ω, while it is unknown for TS. For every sample included in the TS, the k-NN algorithm calculates the distance between this and all the samples of TR. Euclidean distance is the most used distance function. Thus, k-NN takes the k closest samples in TR by ranking in ascending order according to the distance. Then, it computes a majority voting with the class label of the k nearest neighbours. The chosen value of k may influence the performance and the noise tolerance of this technique.

Although the k-NN is known because its good performance in a wide variety of problems, it presents issues to handle large-scale datasets. The two main problems found are:

- Memory consumption: If it has training and test data stored in memory, it will get a rapid computation of the distances. However, when TR and TS sets are really big, they easily exceed the available main memory.
- Computation complexity: The complexity to obtain the nearest neighbour samples of a single test instance in the training set is $O((n \cdot D))$, where n is the number of training examples and D the number of features. In order to find the k closest neighbours, the computational complexity is increased on $O(n \cdot log(n))$ because of the necessary sorting. Note that this computational effort must be repeated for each test sample.

B. k-NN design for Hadoop and Spark

The MapReduce programming paradigm is a scale-out data processing tool. It defines three stages to handle distributed data: Map, Shuffle and Reduce. Map stage reads the raw data in form of key-value pairs and it is distributed among the different computing nodes that are available. Then, it computes the same function for each data split. Shuffle phase is responsible for merging all the values associated with the same intermediate key. Finally, reduce stage aggregates the pairs with the same key into smaller key-value pairs. MapReduce automatically processes data in a cluster, distributing the data, releasing the developer from technical issues such as data partitioning, fault-tolerance or job communication.

Apache Hadoop is the most popular open-source implementation of MapReduce paradigm. In [19] authors proposed an approach of k-NN using Hadoop. First, it splits the TR set and it is distributed over the computing nodes. The map phase computes the k nearest neighbours between TS for each TR part. Then, it sends a vector of pairs (Pair<distance, class>) to reduce stage. After shuffle phase, the reducer aggregates all the candidate of each test sample and calculates a majority voting to obtain the predicted label.

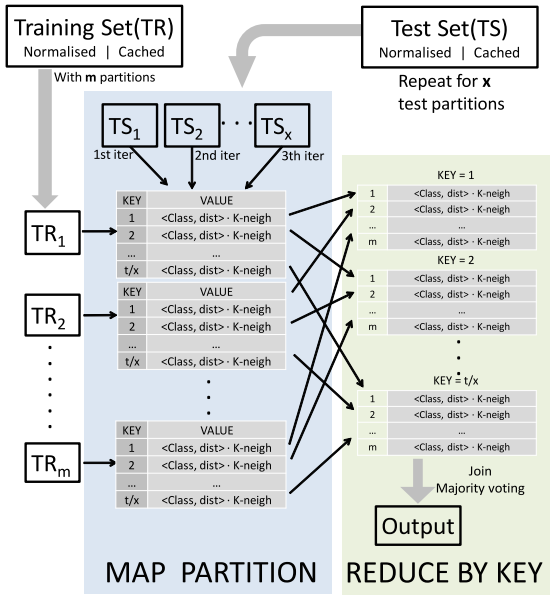

Fig. 1. MapReduce workflow of the kNN-IS algorithm in Apache Spark.

However, Hadoop cannot reuse data through successive iteration, becoming an issue to handle very big (test) datasets. Apache Spark may solve some of the Hadoop drawbacks. The most important feature is the type of data structure that parallelise the computations in a transparent way. For

this reason, a k-NN developed over Spark is faster than the Hadoop-based implementation.

Figure 1 shows an improved version of Hadoop-based k-NN approach, which was designed for Spark [11]. Map and reduce phases do not change their duties. However, in-memory primitives boost the runtime. In addition, Spark provides the necessary mechanisms to efficiently split the TS set when it does not fit in main memory.

These two approaches obtain exactly the same results as the original k-NN algorithm, but they soften the runtime and memory consumption issues. The Spark-based version becomes faster than Hadoop's version mainly because of the in-memory primitives of Spark, which allow us to process different chunks of the TS set iteratively.

III. Data preprocessing with the k-NN algorihtm

As stated before, the k-NN algorithm has been widely used to perform data preprocessing. In many cases motivated on addressing the own k-NN drawbacks, researchers have made many data preprocessing contributions towards the alleviation of such problems. Nevertheless, these techniques are in essence general data preprocessing techniques that allow us to refine imperfect raw data, obtaining useful (smart) data (free of noise, missing values and/or redundant information). In what follows, we briefly discuss two different scenarios in which the k-NN has been used for correcting data imperfection and data reduction (Section III-B).

A. Handling Imperfect data with k-NN based algorithms

The underlying idea of the k-NN algorithm have served of inspiration to tackle data imperfection. Here, we distinguish between two main kinds of data imperfection that need to be addressed: noisy data and incomplete data.

1) Noise filtering and correction: The presence of noise in real world data is an unavoidable problem, which heavily affects the data collection preparation processes in data mining. Noise information may form small clusters of examples of a particular class in areas of the instance space that originally belong to another class. It may also remove instances located in a key region within a specific class or disrupt the boundaries of the classes and increase the overlapping among them. Under these circumstances, data mining models may not be sufficiently robust to extract valuable knowledge. Therefore, identifying noisy instances which can be eliminated from the training data is an important step.

In the specialised literature, two different kinds of noise are defined: class and attribute noise. The former occurs when a sample is incorrectly labelled. The latter refers to corruptions in the values of one or more attributes. While the class noise problem has been tackled in many different way, attribute noise remains to be an under-explored field.

As we mentioned before, the original k-NN classifier is also affected by noisy data, however, the distance-based similarity idea of the k-NN has been widely applied to detect and remove class noise. For instance, the well-known Edited Nearest Neighbour (ENN, [20]) consists of removing all those training examples whose class label does not agree with the majority of their k nearest neighbours. In [21], the authors proposed a variant of the ENN that rather than eliminating all potential noisy examples, it may change the class label of clearly erroneous examples. More examples of k-NN based noise filters can be found in [14] and [13] under the name of edition-based models.

2) Missing values imputation: Rather than erroneous data, many datasets also contain missing values (MVs) in their attribute values. Intuitively, a MV is simply a value for an attribute that was not annotated or was missing. Human or equipment errors are some of the reasons of their existence. Once again, this imperfection on the data influence the mining process and its outcome.

The simplest way of dealing with MVs is to discard the examples that contain them. However, this method is impractical when the number of affected examples is too big. The imputation of MVs is a procedure that aims to fill in the MVs by estimating them. In most cases, attributes are not independent from each other and therefore, identifying that relationships among attributes, MVs can be approximated.

One of the most used imputation methods is based on the k-NN algorithm (named as KNNI) [18]. For each instance containing one or more MVs, KNNI calculates the k nearest neighbours, and the MVs are imputed based on the existing values of these neighbours. For nominal values, the most common value among all neighbours is selected, and for numerical values the average value is used. Note that to compute the distances of the attributes with MVs are ignored.

B. Data reduction with the k-NN algorithm

Data reduction approaches aim to obtain a smaller representative set of examples from raw data without losing important information. This process allows us to alleviate data storage necessities as well as improving the later data mining process. This process may result on the elimination of noisy information, but also redundant or irrelevant data.

From the perspective of attributes, the most popular data reduction processes are feature selection and feature extraction [15]. At the instance level, we find instance reduction methods [14], [13]. This latter is usually divided into instance selection [13], which are limited to take a subset from the original training data, and instance generation [14] that may generate artificial data if needed to better represent the training set.

Most of the proposed instance reduction techniques were originally designed to improve the k-NN algorithm. Existing models are usually based on the k-NN algorithm and its way of computing similarities between examples. Among the most relevant proposals, evolutionary algorithms highlight as good performing approaches in which the fitness function consists of classifying the entire training set using the k-NN algorithm.

For feature selection, we can also find that the k-NN algorithm has been the core idea of many proposals [22]. As in the case of instance reduction, many evolutionary-based feature selection models [23] have been also focused on k-NN.

As we mentioned previously, these techniques are supposed to ease data mining algorithms to address with big data problems, however, these methods are also affected by the

increase of the size and complexity of datasets and they are unable to provide a preprocessed dataset in a reasonable time. For this reason, several approaches have been proposed to enable data reduction techniques to tackle big datasets based on Hadoop MapReduce. Concretely, based on k-NN, we can find an approach in [24] to perform feature selection on big datasets using the k-NN rule within a evolutionary mode. And in [25], a framework named MRPR was designed to enable instance reduction techniques to be applied on big datasets.

IV. From Big Data to Smart Data: k-NN as a keystone

This section presents different cases of study that show the potential of the k-NN algorithm as a unique model to obtain smart data from large amounts of potentially imperfect data. First, Section IV-A shows the weaknesses of the k-NN algorithm in the big data context, and how parallel strategies may help us to mitigate them. Second, we analyse the problem of MVs and the role of k-NN to tackle it. Finally, Section IV-C presents a case of study on instance reduction for big datasets based on k-NN.

A. k-NN in the big data context

This section illustrates the necessity of a scalable design of the k-NN algorithm as well as the issue of k-NN with noisy data. Part of this experimental study is focused on our previous contribution in [11].

To do this, we make use of Susy dataset which is available on the UCI machine learning repository[2]. In order to analyse the speed up between sequential and distributed approaches, we could not go further than Susy dataset to obtain the results of the sequential k-NN. It has 5 million of samples with 18 features that may belong to 2 different classes.

Table I shows the runtime of original k-NN version (Sequential), the runtime of distributed k-NN version (kNN-IS) and the speed up achieved with the Spark-based implementation. Both runtimes are displayed in seconds. From this table, it can be concluded that kNN-IS provides the necessary scalability to handle big data problems without losing the exact results reported by the original k-NN algorithm.

Figure 2 presents the accuracy for k parameter equal to 1, 3, 5 and 7 versus the percentage of noise data.

TABLE I
Speed up between sequential k-NN and kNN-IS approaches. Number of maps set to 256

k	Sequential	kNN-IS	Speed up
1	3,258,848.81	1,900.03	1,715.14
3	3,259,619.49	2,615.01	1,246.20
5	3,265,185.90	2,273.63	1,436.10
7	3,325,338.14	2,372.41	1,401.67

Therefore, it is clear that noisy data must be treated in order to improve the accuracy. Nevertheless, there is a lack of methods to preprocess noisy data in big data problems, where the runtime matters.

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

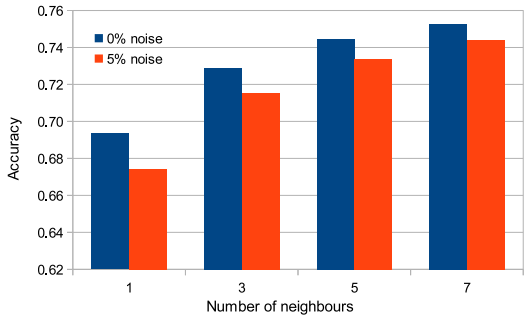

Fig. 2. Influence of noisy data on the k-NN method for big datasets

TABLE II
Data with MVs properties

Dataset	#Inst.	#Atts.	#Cl.	%MV	% Inst. with MV
Wisconsin	699	10	2	0.23	2.29
Credit	689	16	2	0.61	5.37
Autos	205	26	6	1.11	22.44
Dermatology	365	35	6	0.06	2.19
Sponge	76	46	12	0.63	28.95
Bands	540	40	2	4.63	48.7
Horse-colic	368	24	2	21.82	98.1
Audiology	226	71	24	1.98	98.23
Hepatitis	155	20	2	5.39	48.39
Ozone	2534	73	2	8.07	27.11

B. A case of study: Missing values imputation

In this section, we study the influence of MVs to perform a proper classification, and how the KNNI algorithm may be used to successfully impute MVs. Part of this study comes from our previous paper in [17].

For this experiment, we take 10 small datasets from UCI Machine learning repository. In all the experiments, we have adopted a 10-fold cross-validation model. Table II summarises the properties of the selected datasets. It presents for each dataset the number of Instances (#Inst.), the number of attributes (#Atts.), the number of classes (#Cl.), the total percentage of MVs (% MV) and the percentage of instances with at least one MV (% Inst. with MV).

To analyse the validity of the imputation produced by the KNNI algorithm, we compare it with:

1) Example deletion or Ignore Missing (IM). All instances with at least one MV are removed from the dataset.
2) Most Common Attribute Value for Symbolic Attributes, and Global Average Value for Numerical Attributes (MC). For nominal attributes, the MV is replaced with the most common attribute value, and numerical values are replaced with the average of all values of the corresponding attribute.

To show that the preprocessing performed by KNNI goes beyond the improvement of the k-NN rule, we test the different imputation mechanisms on three different classifiers: one nearest neighbour (1-NN) , decision trees (C4.5) and Support

TABLE III
MISSING VALUES IMPUTATION RESULTS

	1-NN			C4.5			SVM		
Datasets	IM	KNNI	MC	IM	KNNI	MC	IM	KNNI	MC
Wisconsin	95.90	95.42	96.13	94.73	93.71	94.14	96.77	96.57	96.71
Credit	81.02	80.13	80.27	84.70	84.91	84.62	86.02	84.77	84.77
Autos	76.44	77.02	76.48	80.93	82.19	80.74	72.81	71.33	69.53
Dermatology	92.71	92.86	92.86	93.53	93.42	93.42	96.94	97.00	97.00
Sponge	82.86	80.54	81.96	63.14	65.89	65.89	86.29	85.36	85.36
Bands	72.47	73.33	72.78	64.62	69.44	69.34	75.36	80.37	80.74
Horse-colic	40.00	81.53	80.68	40.00	83.41	82.60	40.00	82.58	82.33
Audiology	20.00	78.79	74.80	0.00	78.42	78.20	20.00	79.29	78.44
Hepatitis	81.65	82.00	80.71	82.98	76.79	80.04	82.84	80.67	81.38
Ozone	91.50	92.62	92.19	91.12	91.71	90.81	93.05	93.69	93.69
Average	73.45	83.42	82.89	69.58	81.99	81.98	75.01	85.16	84.99

Vector Machines (SVM). Parameter configuration for these classifier is as in [17].

Table III collects the obtained results classifying the test set with the different classifier and imputation techniques. The average result for each pair classifier, imputation can be found in the last row in order to analyse the global behaviour.

According to this table, we can see that IM reports the lowest accuracy for the three classifiers, while both MVs imputation mechanisms seem to be clearly better than ignore them. Moreover, the imputation performed by the KNNI algorithm systematically provides the best results for every single classifier. This shows how the preprocessing capabilities of this model go beyond improving the k-NN algorithm.

However, the use of KNNI in the big data context is still an under-explored area. It has to handle the same problems that we defined in Section II-A for the standard k-NN algorithm: memory consumption and runtime. In addition, the distance function has restrictions because of the missing values, complicating the parallelisation process.

C. A case of study: Instance Reduction based on k-NN

Rather than tackling directly the original training data with a classifier, instance reduction techniques will reduce the size of the input data to later classify more efficiently and even more effectively. In [25], the scheme MRPR allows any instance reduction model to be applied in big datasets. Here, we show some results that illustrate the benefits of applying such scheme in terms of runtime and storage requirements.

For this study, we take the Poker-Hand dataset from UCI repository, which consists of 1 million instances and 10 attributes. Again, we apply a 10 fold cross validation. We selected a number of representative variety of instance reduction methods: LVQ3, FCNN, DROP3, RSP3 and SSMASFLSDE, which possess different characteristics (See [25] for more details). These methods are used within the big data framework MRPR to quickly handle this dataset. After the preprocessing stage, the resulting reduced set is used by a k-NN algorithm ($k = 1$) as training set in order to classify the test set.

Table IV summarises the results obtained with all the considered instance reduction techniques. It shows test accuracy, reduction rate obtained (comparing the resulting preprocessed set and the original training set), runtime in seconds. For each one of these measures, average (Avg.) and standard deviation (Std.) results are collected. In addition, the required time to classify the entire test set using the resulting preprocessed dataset. As a baseline, we include the classification done with the k-NN algorithm ($k = 1$), using the whole TR set to classify all the instances of TS.

Figure 3 depicts the data storage reduction (in Megabytes) on this dataset for each of the instance reduction techniques used.

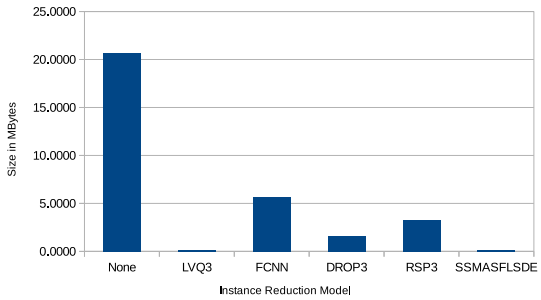

Fig. 3. Storage requirements reduction

We can observe how all the analysed instance reduction techniques provide a great reduction rate which results in high reduction of storage requirements and classification time of test instances. In Figure 3, we can check the significant reduction in Megabytes required to store the training. Nevertheless, the main goal of instance reduction techniques is to reduce the space requirements without losing accuracy. In Table IV, we can see that none of the preprocessing methods are losing that much accuracy w.r.t to the baseline algorithm (1-NN). Actually, some of them are able to obtain better performance as they have possibly removed noisy and redundant examples.

We can also notice that the runtime required to do the preprocessing varies from one model to another. Specially, best performing approaches may take a long time to produce an ap-

TABLE IV
INSTANCE REDUCTION RESULTS WITH DIFFERENT METHODS INCORPORATED IN MRPR (64 MAPPERS - POKER-HAND DATASET)

	Test		Reduction		Time		Clasif.
	AccTst	StdTst	AvgRed	StdRed	AvgTime	StdTime	Time.
1-NN	0.5001	0.0011	0.0000	0.0000	0.0000	0.0000	48760.8242
LVQ3	0.4918	0.0012	99.3811	0.0067	83.7830	4.8944	273.4192
FCNN	0.4862	0.0006	72.5604	0.0080	3207.8540	37.2208	9854.8956
DROP3	0.5011	0.0005	92.3467	0.0043	198.1450	5.2750	1811.0866
RSP3	0.5107	0.0010	84.3655	0.0189	1448.4272	60.5462	5741.6588
SSMASFLSDE	0.5181	0.0015	99.1413	0.0217	14419.3926	209.9481	374.8814

propriate reduced set. In practice, however, the preprocessing process would only need to be carried out once.

V. CONCLUSIONS

In this work, we have discussed the influence of the well-known k-nearest neighbour algorithm to transform big datasets into smart datasets.

After stating the main drawbacks of the standard k-NN algorithm, we have analysed how to enable this technique to manage large amounts of data by means of big data technologies. Then, we have covered how the main weaknesses of this technique have been used in the specialised literature to obtain cleaner (smart) data. Concretely, we have shown k-NN based strategies to correct data imperfection (noisy and/or missing values), and its role to perform data reduction.

From the studies presented in this work, we can conclude that the k-NN algorithm is a very promising technique to obtain smart data from big data. As future work, we consider the use of k-NN to deal with data imperfection in big datasets.

ACKNOWLEDGEMENT

This work has been partially supported by the Spanish National Research Project TIN2014-57251-P. J. Maillo holds FPU scholarships from the Spanish Ministry of Education.

REFERENCES

[1] V. Marx, "Biology: The big challenges of big data," *Nature*, vol. 498, no. 7453, pp. 255–260, Jun. 2013.

[2] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, no. 1, pp. 37–66, 1991.

[3] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information Sciences*, vol. 275, pp. 314 – 347, 2014.

[4] F. Iafrate, "A journey from big data to smart data," in *Digital Enterprise Design & Management: Proceedings of the Second International Conference on Digital Enterprise Design and Management DED&M 2014*, Cham, 2014, pp. 25–33.

[5] A. Lenk, L. Bonorden, A. Hellmanns, N. Roedder, and S. Jaehnichen, "Towards a taxonomy of standards in smart data," in *Proceedings of the 2015 IEEE International Conference on Big Data*. Washington, DC, USA: IEEE Computer Society, 2015, pp. 1749–1754.

[6] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*. Springer Publishing Company, Incorporated, 2014.

[7] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[8] C. Zhang, F. Li, and J. Jestes, "Efficient parallel knn joins for large data in mapreduce," in *Proceedings of the 15th International Conference on Extending Database Technology*. New York, NY, USA: ACM, 2012, pp. 38–49.

[9] K. Sun, H. Kang, and H.-H. Park, "Tagging and classifying facial images in cloud environments based on kNN using mapreduce," *Optik - International Journal for Light and Electron Optics*, vol. 126, no. 21, pp. 3227 – 3233, 2015.

[10] J. Dean and S. Ghemawat, "Map reduce: A flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

[11] J. Maillo, S. Ramirez, I. Triguero, and F. Herrera, "kNN-IS: An iterative spark-based design of the k-nearest neighbors classifier for big data," *Knowledge-Based Systems.*, vol. in press, 2016.

[12] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012, pp. 1–14.

[13] S. García, J. Derrac, J. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 417–435, 2012.

[14] I. Triguero, J. Derrac, S. García, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification," *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, vol. 42, no. 1, pp. 86–100, 2012.

[15] H. Liu and H. Motoda, Eds., *Computational Methods of Feature Selection*, ser. Chapman & Hall/Crc Data Mining and Knowledge Discovery Series, 2007.

[16] S. Garcia, J. R. Cano, and F. Herrera, "A memetic algorithm for evolutionary prototype selection: A scaling up approach," *Pattern Recognition*, vol. 41, no. 8, pp. 2693 – 2709, 2008.

[17] J. Luengo, S. García, and F. Herrera, "On the choice of the best imputation methods for missing values considering three groups of classification methods," *Knowledge and Information Systems*, vol. 32, no. 1, pp. 77–108, 2012.

[18] G. Batista and M. Monard, "An analysis of four missing data treatment methods for supervised learning," *Applied Artificial Intelligence*, vol. 17, no. 5, pp. 519–533, 2003.

[19] J. Maillo, I. Triguero, and F. Herrera, "A mapreduce-based k-nearest neighbor approach for big data classification," in *9th International Conference on Big Data Science and Engineering*, 2015, pp. 167–172.

[20] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on System, Man and Cybernetics*, vol. 2, no. 3, pp. 408–421, 1972.

[21] J. S. Sánchez, R. Barandela, A. I. Marqués, R. Alejo, and J. Badenas, "Analysis of new techniques to obtain quality training sets," *Pattern Recognition Letters*, vol. 24, no. 7, pp. 1015–1022, 2003.

[22] A. Navot, L. Shpigelman, N. Tishby, and E. Vaadia, "Nearest neighbor based feature selection for regression and its application to neural activity," in *Advances in Neural Information Processing Systems 18*. MIT Press, 2006, pp. 995–1002.

[23] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, Aug 2016.

[24] D. Peralta, S. del Rio, S. Ramirez-Gallego, I. Triguero, J. M. Benitez, and F. Herrera, "Evolutionary feature selection for big data classification: A mapreduce approach," *Mathematical Problems in Engineering*, 2016, article ID 246139.

[25] I. Triguero, D. Peralta, J. Bacardit, S. García, and F. Herrera, "MRPR: A mapreduce solution for prototype reduction in big data classification," *Neurocomputing*, vol. 150, pp. 331–345, 2015.