Journal of Computer Science

Original Research Paper

A Reformed K-Nearest Neighbors Algorithm for Big Data Sets

[1]Vo Ngoc Phu and [2]Vo Thi Ngoc Tran

[1]Nguyen Tat Thanh University, 300A Nguyen Tat Thanh Street, Ward 13, District 4, Ho Chi Minh City, 702000, Vietnam
[2]School of Industrial Management (SIM), Ho Chi Minh City University of Technology - HCMUT, Vietnam National University, Ho Chi Minh City, Vietnam

Article history
Received: 01-02-2018
Revised: 25-02-2018
Accepted: 8-03-2018

Corresponding Author:
Vo Ngoc Phu
Nguyen Tat Thanh University, 300A Nguyen Tat Thanh Street, Ward 13, District 4, Ho Chi Minh City, 702000, Vietnam
Email:
vongocphu03hca@gmail.com
vongocphu@ntt.edu.vn

Abstract: A Data Mining Has Already Had Many Algorithms Which A K-Nearest Neighbors Algorithm, K-NN, Is A Famous Algorithm For Researchers. K-NN Is Very Effective On Small Data Sets, However It Takes A Lot Of Time To Run On Big Datasets. Today, Data Sets Often Have Millions Of Data Records, Hence, It Is Difficult To Implement K-NN On Big Data. In This Research, We Propose An Improvement To K-NN To Process Big Datasets In A Shortened Execution Time. The Reformed K-Nearest Neighbors Algorithm (R-K-NN) Can Be Implemented On Large Datasets With Millions Or Even Billions Of Data Records. R-K-NN Is Tested On A Data Set With 500,000 Records. The Execution Time Of R-K-NN Is Much Shorter Than That Of K-NN. In Addition, R-K-NN Is Implemented In A Parallel Network System With Hadoop Map (M) And Hadoop Reduce (R).

Keywords: K-Nearest Neighbors Algorithm, K-NN, Parallel Network Environment, Distributed System, Data Mining, Association Rules, Cloudera, Hadoop Map, Hadoop Reduce

Introduction

The field of data mining has been studied for many years. The K-Nearest Neighbors algorithm (K-NN) is a popular algorithm in the data mining field. We use K-NN to stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions) and it is a simple algorithm. Besides, K-NN has been used in statistical estimation and pattern recognition and it is a non-parametric technique.

It is clear that social networks, information technology and computer science are being developed at a dramatic rate, generating vast amounts of data, information and knowledge. The information in these big data sets belonging to large corporations is very valuable, particularly if this information can be exploited in ways which are of benefit.

Hence, many algorithms have been proposed to run on big datasets. K-NN is a well-known algorithm in data mining and other fields, but K-NN takes a lot of time to run on big data sets, yet it is efficient on small datasets. Thus, we tested K-NN and other algorithms on big data sets. There are many algorithms which can perform efficiently on small data sets but they are not effective on big datasets with millions of data records. To test this, we studied K-NN on big data sets and examined the existing research.

We want this survey to design an improved K-NN algorthim (R-K-NN), with Hadoop Map (M) and Hadoop Reduce (R) to implement on data sets containing millions of records in parallel systems, the results showing that R-K-NN is able to process big datasets in a shortened time. R-K-NN is also tested in the Cloudera distributed system.

According to the existing research related to the K-Nearest Neighbors algorithm (Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al*., 2012; Kuncheva, 1995; Pan *et al*., 2015; Franco-Lopez *et al*., 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al*., 2005; Song and Roussopoulos, 2001), there is no work related to K-NN which is similar to this work, nor is there any study related to the K-Nearest Neighbors algorithm in the parallel network environment (the distributed system).

In the research related to the distributed environment reported in (Favuzza *et al*., 2006; Satyanarayanan *et al*., 2002; Babaoğlu *et al*., 1992; Fujimoto, 2001; Phu *et al*., 2016; Shirazi *et al*., 1995; Chen *et al*., 2011; Sulistio *et al*., 2004; Borges *et al*., 2001), there are no studies related to K-NN which is similar to this work.

The surveys in (Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al*., 2012; Kuncheva, 1995; Pan *et al*., 2015; Franco-Lopez *et al*., 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al*., 2005; Song and Roussopoulos, 2001; Favuzza *et al*., 2006; Satyanarayanan *et al*., 2002; Babaoğlu *et al*., 1992; Fujimoto, 2001; Phu *et al*., 2016; Shirazi *et al*., 1995; Chen *et al*., 2011; Sulistio *et al*., 2004; Borges *et al*., 2001; Fujita *et al*., 1998; Dat *et al*., 2016) are very different from our study in this paper.

As reported in the work on K-NN found in (Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al*., 2012; Kuncheva, 1995; Pan *et al*., 2015; Franco-Lopez *et al*., 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al*., 2005; Song and Roussopoulos, 2001), we have already known many benefits and drawbacks of K-NN. We present the negatives of K-NN as follows: We knew many very simple classifiers such as K-NN which works well on basic recognition problems; and it can work with noisy training data (if inverse square of weighted distance is used as the 'distance'). The drawbacks of K-NN are displayed as follows: The KNN algorithm will find the K closest neighbors to the new instance from the training data and the predicted class label will then be set as the most common label among the K closest neighboring points; the distance must be computed and all the training data at each prediction are shorted, if there is a large number of training examples, it can be slow.

From an examination of the methods in the existing studies reported in (Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al*., 2012; Kuncheva, 1995; Pan *et al*., 2015; Franco-Lopez *et al*., 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al*., 2005; Song and Roussopoulos, 2001; Favuzza *et al*., 2006; Satyanarayanan *et al*., 2002; Babaoğlu *et al*., 1992; Fujimoto, 2001; Phu *et al*., 2016; Shirazi *et al*., 1995; Chen *et al*., 2011; Sulistio *et al*., 2004; Borges *et al*., 2001; Fujita *et al*., 1998; Dat *et al*., 2016), it is clear that there is no work which is similar to the work in this paper, hence the originality and novelty of this work.

We show our work's motivation as follows: K-NN of the data mining field is a popular algorithm which is being increasingly used and applied to many different fields, hence it is very useful for researchers and commercial applications. In today's information age, massive amounts of big datasets are being generated which current algorithms and methods are not able to successfully process. Many methods, applications and systems in the current time are implemented correctly with the current data sets, whereas, these cannot be performed correctly. Therefore, we propose a new model to address these limitations.

The novelty of the proposed approach is as follows: KNN can be run in the parallel network environment and can also handle big data sets. Many KNN used models/methods, many KNN related applications and many KNN used systems can be upgraded on the big data sets and on the distributed systems. Thus, this model is proposed.

We also display the crucial contributions of our survey as follows:

a) This work implements K-NN in the parallel environment
b) This work implements K-NN to process big data with millions of records
c) This research implements K-NN in both sequential systems and distributed environments
d) We propose the K-NN related algorithms which are implemented in distributed systems
e) The results of this novel model can be used in many other studies and commercial applications
f) Using the results of this work, other studies and systems related to KNN can be successfully enhanced

In light of these contributions, the superiority of R-K-NN over K-NN is demonstrated.

We present five sections of this work as follows: This section is the Introduction section. We show Section 2 to detail the related studies on the K-Nearest Neighbors algorithm. The methodology for the implementation of the K-Nearest Neighbors algorithm in the parallel network environment is displayed in Section 4. We show Section 4 which presents details of the experiments. We present the conclusion of this work in Section 5.

Related Work

In this section, we detail the existing research related to the K-Nearest Neighbors algorithm and the parallel network system.

We show algorithms, applications and studies in the distributed system in (Hadoop, 2016; Apache, 2016; Cloudera, 2016). We use Hadoop, an Apache-based framework, in (Hadoop, 2016; Apache, 2016) to handle large data sets on clusters consisting of multiple computers, using the Map and Reduce programming model. There are two components of Hadoop: The Hadoop Distributed File System (HDFS) and Hadoop M/R (Hadoop Map/Reduce). Engineers use Hadoop Map and Hadoop Reduce to program to write applications for the parallel processing of large data sets on clusters consisting of multiple computers. An M/R task has two main components: (1) Map and (2) Reduce. The global provider of the fastest, easiest and most secure data management and analytics platform built on Apache™ Hadoop® and the latest open source technologies, called Cloudera (2016); and it will submit proposals for Impala and Kudu to join the Apache Software Foundation (ASF). Cloudera delivers a modern data management and analytics platform built on Apache Hadoop and the latest open source technologies.

The surveys related to the K-Nearest Neighbors algorithm are presented in (Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al*., 2012; Kuncheva, 1995; Pan *et al*., 2015; Franco-Lopez *et al*., 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al*., 2005; Song and Roussopoulos, 2001). A discussion of the differences between supervised and unsupervised methods is shown in the work (Larose, 2005) and the k-nearest neighbor algorithm is introduced, in the context of a patient-drug classification problem. The authors (Fukunaga and Narendra, 2006) detailed K-nearest neighbor methods for estimation and prediction which are examined, along with methods for choosing the best value for k. Computation of the k-nearest neighbors generally requires a large number of expensive distance computations.

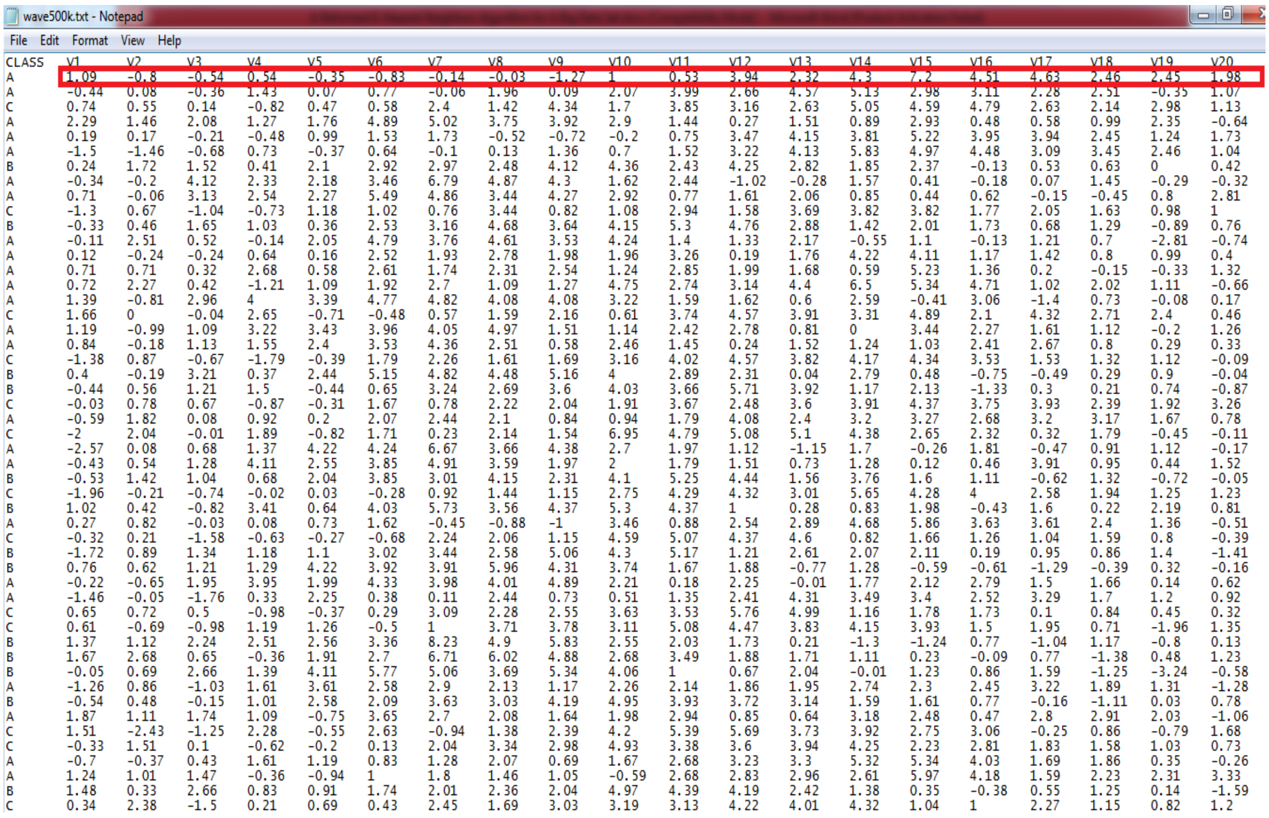
Based on the best of our knowledge, we do not see any researches related to the K-Nearest Neighbors algorithm in the parallel system.

Comparisons of our novel model's results with the studies in (Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al*., 2012; Kuncheva, 1995; Pan *et al*., 2015; Franco-Lopez *et al*., 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al*., 2005; Song and Roussopoulos, 2001; Favuzza *et al*., 2006; Satyanarayanan *et al*., 2002; Babaoğlu *et al*., 1992; Fujimoto, 2001; Phu *et al*., 2016; Shirazi *et al*., 1995; Chen *et al*., 2011; Sulistio *et al*., 2004; Borges *et al*., 2001) and the comparative results are presented in Table 2.

Research related to the distributed environment is reported in (Favuzza *et al*., 2006; Satyanarayanan *et al*., 2002; Babaoğlu *et al*., 1992; Fujimoto, 2001; Phu *et al*., 2016; Shirazi *et al*., 1995; Chen *et al*., 2011; Sulistio *et al*., 2004; Borges *et al*., 2001). The authors (Favuzza *et al*., 2006) displayed the metaheuristic technique of Ant Colony Search which was revised to deal with dynamic search optimization problems having a large search space and mixed integer variables. The authors presented Coda in (Satyanarayanan *et al*., 2002) which is a file system for a large-scale distributed computing environment composed of Unix workstations, etc.

Data Set

Our data set has 500,000 data records (The Data Set, 2016) as shown in the Fig. 1 below.



CLASS	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20
A	1.09	-0.8	-0.54	0.54	-0.35	-0.83	-0.14	-0.03	-1.27	1	0.53	3.94	2.32	4.3	7.2	4.51	4.63	2.46	2.45	1.98
A	-0.44	0.08	-0.36	1.43	0.07	0.77	-0.06	1.96	0.09	2.07	3.99	2.66	4.57	5.13	2.98	3.11	2.28	2.51	-0.35	1.07
C	0.74	0.55	0.14	-0.82	0.47	0.58	2.4	1.42	4.34	1.7	3.85	3.16	2.63	5.05	4.59	4.79	2.63	2.14	2.98	1.13
A	2.29	1.46	2.08	1.27	1.76	4.89	5.02	3.75	3.92	2.9	1.44	0.27	1.51	0.89	2.93	0.48	0.58	0.99	2.35	-0.64
A	0.19	0.17	-0.21	-0.48	0.99	1.53	1.73	-0.52	-0.72	-0.2	0.75	3.47	4.15	3.81	5.22	3.95	3.94	2.45	1.24	1.73
A	-1.5	-1.46	-0.68	0.73	-0.37	0.64	-0.1	0.13	1.36	0.7	1.52	3.22	4.13	5.83	4.97	4.48	3.09	3.45	2.46	1.04
B	0.24	1.72	1.52	0.41	2.1	2.92	2.97	2.48	4.12	4.36	2.43	4.25	2.82	1.85	2.37	-0.13	0.53	0.63	0	0.42
A	-0.34	-0.2	4.12	2.33	2.18	3.46	6.79	4.87	4.3	1.62	2.44	-1.02	-0.28	1.57	0.41	-0.18	0.07	1.45	-0.29	-0.32
A	0.71	-0.06	3.13	2.54	2.27	5.49	4.86	3.44	4.27	2.92	0.77	1.61	2.06	0.85	0.44	0.62	-0.15	-0.45	0.8	2.81
C	-1.3	0.67	-1.04	-0.73	1.18	1.02	0.76	3.44	0.82	1.08	2.94	1.58	3.69	3.82	3.82	1.77	2.05	1.63	0.98	1
B	-0.33	0.46	1.65	1.03	0.36	2.53	3.16	4.68	3.64	4.15	5.3	4.76	2.88	1.42	2.01	1.73	0.68	1.29	-0.89	0.76
A	-0.11	2.51	0.52	-0.14	2.05	4.79	3.76	4.61	3.53	4.24	1.4	1.33	2.17	-0.55	1.1	-0.13	1.21	0.7	-2.81	-0.74
A	0.12	-0.24	-0.24	0.64	0.16	2.52	1.93	2.78	1.98	1.96	3.26	0.19	1.76	4.22	4.11	1.17	1.42	0.8	0.99	0.4
A	0.71	0.71	0.32	2.68	0.58	2.61	1.74	2.31	2.54	1.24	2.85	1.99	1.68	0.59	5.23	1.36	0.2	-0.15	-0.33	1.32
A	0.72	2.27	0.42	-1.21	1.09	1.92	2.7	1.09	1.27	4.75	2.74	3.14	4.4	6.5	5.34	4.71	1.02	2.02	1.11	-0.66
A	1.39	-0.81	2.96	4	3.39	4.77	4.82	4.08	4.08	3.22	1.59	1.62	0.6	2.59	-0.41	3.06	-1.4	0.73	-0.08	0.17
C	1.66	0	-0.04	2.65	-0.71	-0.48	0.57	1.59	2.16	0.61	3.74	4.57	3.91	3.31	4.89	2.1	4.32	2.71	2.4	0.46
A	1.19	-0.99	1.09	3.22	3.43	3.96	4.05	4.97	1.51	1.14	2.42	2.78	0.81	0	3.44	2.27	1.61	1.12	-0.2	1.26
A	0.84	-0.18	1.13	1.55	2.4	3.53	4.36	2.51	0.58	2.46	1.45	0.24	1.52	1.24	1.03	2.41	2.67	0.8	0.29	0.33
C	-1.38	0.87	-0.67	-1.79	-0.39	1.79	2.26	1.61	1.69	3.16	4.02	4.57	3.82	4.17	4.34	3.53	1.53	1.32	1.12	-0.09
B	0.4	-0.19	3.21	0.37	2.44	5.15	4.82	4.48	5.16	4	2.89	2.31	0.04	2.79	0.48	-0.75	-0.49	0.29	0.9	-0.04
B	-0.44	0.56	1.21	1.5	-0.44	0.65	3.24	2.69	3.6	4.03	3.66	5.71	3.92	1.17	2.13	-1.33	0.3	0.21	0.74	-0.87
C	-0.03	0.78	0.67	-0.87	-0.31	1.67	0.78	2.22	2.04	1.91	3.67	2.48	3.6	3.91	4.37	3.75	3.93	2.39	1.92	3.26
A	-0.59	1.82	0.08	0.92	0.2	2.07	2.44	2.1	0.84	0.94	1.79	4.08	2.4	3.2	3.27	2.68	3.2	3.17	1.67	0.78
C	-2	2.04	-0.01	1.89	-0.82	1.71	0.23	2.14	1.54	6.95	4.79	5.08	5.1	4.38	2.65	2.32	0.32	1.79	-0.45	-0.11
A	-2.57	0.08	0.68	1.37	4.22	4.24	6.67	3.66	4.38	2.7	1.97	1.12	-1.15	1.7	-0.26	1.81	-0.47	0.91	1.12	-0.17
A	-0.43	0.54	1.28	4.11	2.55	3.85	4.91	3.59	1.97	2	1.79	1.51	0.73	1.28	0.12	0.46	3.91	0.95	0.44	1.52
B	-0.53	1.42	1.04	0.68	2.04	3.85	3.01	4.15	2.31	4.1	5.25	4.44	1.56	3.76	1.6	1.11	-0.62	1.32	-0.72	-0.05
C	-1.96	-0.21	-0.74	-0.02	0.03	-0.28	0.92	1.44	1.15	2.75	4.29	4.32	3.01	5.65	4.28	4	2.58	1.94	1.25	1.23
B	1.02	0.42	-0.82	3.41	0.64	4.03	5.73	3.56	4.37	5.3	4.37	1	0.28	0.83	1.98	-0.43	1.6	0.22	2.19	0.81
A	0.27	0.82	-0.03	0.08	0.73	1.62	-0.45	-0.88	-1	3.46	0.88	2.54	2.89	4.68	5.86	3.63	3.61	2.4	1.36	-0.51
C	-0.32	0.21	-1.58	-0.63	-0.27	-0.68	2.24	2.06	1.15	4.59	5.07	4.37	4.6	0.82	1.66	1.26	1.04	1.59	0.8	-0.39
B	-1.72	0.89	1.34	1.18	1.1	3.02	3.44	2.58	5.06	4.3	5.17	1.21	2.61	2.07	2.11	0.19	0.95	0.86	1.4	-1.41
B	0.76	0.62	1.21	1.29	4.22	3.92	3.91	5.96	4.31	3.74	1.67	1.88	-0.77	1.28	-0.59	-0.61	-1.29	-0.39	0.32	-0.16
A	-0.22	-0.65	1.95	3.95	1.99	4.33	3.98	4.01	4.89	2.21	0.18	2.25	-0.01	1.77	2.12	2.79	1.5	1.66	0.14	0.62
A	-1.46	-0.05	-1.76	0.33	2.25	0.38	0.11	2.44	0.73	0.51	1.35	2.41	4.31	3.49	3.4	2.52	3.29	1.7	1.2	0.92
C	0.65	0.72	0.5	-0.98	-0.37	0.29	3.09	2.28	2.55	3.63	3.53	5.76	4.99	1.16	1.78	1.73	0.1	0.84	0.45	0.32
C	0.61	-0.69	-0.98	1.19	1.26	-0.5	1	3.71	3.78	3.11	5.08	4.47	3.83	4.15	3.93	1.5	1.95	0.71	-1.96	1.35
B	1.37	1.12	2.24	2.51	2.56	3.36	8.23	4.9	5.83	2.55	2.03	1.73	0.21	-1.3	-1.24	0.77	-1.04	1.17	-0.8	0.13
B	1.67	2.68	0.65	-0.36	1.91	2.7	6.71	6.02	4.88	2.68	3.49	1.88	1.71	1.11	0.23	-0.09	0.77	-1.38	0.48	1.23
B	-0.05	0.69	2.66	1.39	4.11	5.77	5.06	3.69	5.34	4.06	1	0.67	2.04	-0.01	1.23	0.86	1.59	-1.25	-3.24	-0.58
A	-1.26	0.86	-1.03	1.61	3.61	2.58	2.9	2.13	1.17	2.26	2.14	1.86	1.95	2.74	2.3	2.45	3.22	1.89	1.31	-1.28
B	-0.54	0.48	-0.15	1.01	2.58	2.09	3.63	3.03	4.19	4.95	3.93	3.72	3.14	1.59	1.61	0.77	-0.16	-1.11	0.03	0.78
A	1.87	1.11	1.74	1.09	-0.75	3.65	2.7	2.08	1.64	1.98	2.94	0.85	0.64	3.18	2.48	0.47	2.8	2.91	2.03	-1.06
C	1.51	-2.43	-1.25	2.28	-0.55	2.63	-0.94	1.38	2.39	4.2	5.39	5.69	3.73	3.92	2.75	3.06	-0.25	0.86	-0.79	1.68
C	-0.33	1.51	0.1	-0.62	-0.2	0.13	2.04	3.34	2.98	4.93	3.38	3.6	3.94	4.25	2.23	2.81	1.83	1.58	1.03	0.73
A	-0.7	-0.37	0.43	1.61	1.19	0.83	1.28	2.07	0.69	1.67	2.68	3.23	3.3	5.32	5.34	4.03	1.69	1.86	0.35	-0.26
A	1.24	1.01	1.47	-0.36	-0.94	1	1.8	1.46	1.05	-0.59	2.68	2.83	2.96	2.61	5.97	4.18	1.59	2.23	2.31	3.33
B	1.48	0.33	2.66	0.83	0.91	1.74	2.01	2.36	2.04	4.97	4.39	4.19	2.42	1.38	0.35	-0.38	0.55	1.25	0.14	-1.59
C	0.34	2.38	-1.5	0.21	0.69	0.43	2.45	1.69	3.03	3.19	3.13	4.22	4.01	4.32	1.04	1	2.27	1.15	0.82	1.2

Fig. 1: The data in this research

As shown in Fig. 1, there are 21 columns, namely Column CLASS, Column V1, Column V2, Column V3, Column V4, Column V5, Column V6, Column V7, Column V8, Column V9, Column V10, Column V11, Column V12, Column V13, Column V14, Column V15, Column V16, Column V17, Column V18, Column V19, Column V20, Column V21, however we remove the column CLASS and keep the remaining columns. For each data record, we have one vector and therefore the length of each vector is 20.

Methodology

The implementation of the K-Nearest Neighbors algorithm in the sequential environment is firstly detailed and we also present the implementation of the reformed K-Nearest Neighbors algorithm in the Cloudera environment secondly in this section.

K-Nearest Neighbors Algorithm in the Sequential Environment

An overview of K-NN in the sequential system is shown in Fig. 2

We use Algorithm 1 to transfer the 500,000 data records of our data set (The Data Set, 2016) into the 500,000 vectors.

ALGORITHM 1: Transferring the 500,000 data records of our data set [1] into the 500,000 vectors

```
Input:
        The 500,000 data records of the data set (The
        Data Set, 2016): D.
Output: a vector group, including the 500,000 vectors.
Begin
1. Set VectorGroup := {};
2. For i := 0; i < 500,000; i++, do:
3.      OneVector := new Vector();
4.      For j := 1; j <= 21; j++; do:
5.              OneVector[j-1] := D[i]["V" + j];
6.      End For;
7.      VectorGroup.AddVector (OneVector);
8. End For;
9. Return VectorGroup;
End;
```

According to the K-NN algorithm in (Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al.*, 2012; Kuncheva, 1995; Pan *et al.*, 2015; Franco-Lopez *et al.*, 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al.*, 2005; Song and Roussopoulos, 2001), we perform K-NN in the sequential environment as follows:

ALGORITHM 2: K-Nearest Neighbor algorithm (K-NN) in the sequential environment.

```
Input:
        A vector group, including the 500,000 vectors.
Output: the results of clustering of K-NN.
Begin
1. Identify the K parameter (K - Nearest Neighbors): in
        this survey, we choose K = 5;
2. Calculate the distance between the vectors (which
        need to be clustered) with the vectors in the
        training data by Euclidean distance;
3. Arrange the distances in ascending order; and identify
        the K - nearest neighbors with the vectors
        which need to be clustered;
4. Get all clusters of K-nearest neighbors which are
        identified;
5. Identify the cluster of the vector according to all the
        clusters of K-NN; 6. Return the results of
        clustering;
End;
```

Euclidean distance is used by K-NN to identify the distance between two vectors as follows: Vector V1(column x1, column x2, column x3, column x4, column x5, column x6, column x7, column x8, column x9, column x10, column x11, column x12, column x13, column x14, column x15, column x16, column x17, column x18, column x19, column x20) and vector V2(column y1, column y2, column y3, column y4, column y5, column y6, column y7, column y8, column y9, column y10, column y11, column y12, column y13, column y14, column y15, column y16, column y17, column y18, column y19, column y20):

$$d = \sqrt{\begin{array}{l}(x1-y1)^2+(x2-y2)^2+\ldots+(x18-y18)^2\\ +(x19-y19)^2+(x20-y20)^2\end{array}}$$

K-Nearest Neighbors algorithm in the Parallel Network Environment (R-K-NN)

We improve K-NN, called R-K-NN, to utilize in the Cloudera based on the basis of the K-NN algorithm in (Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al.*, 2012; Kuncheva, 1995; Pan *et al.*, 2015; Franco-Lopez *et al.*, 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al.*, 2005; Song and Roussopoulos, 2001).

We also present R-K-NN by using Hadoop Map (M) and Hadoop Reduce (R) in the Cloudera in Fig. 3.

Transferring of the data records comprises two phases as follows: Hadoop Map of the Cloudera in the first phase and Hadoop Reduce of the Cloudera in the second phase.

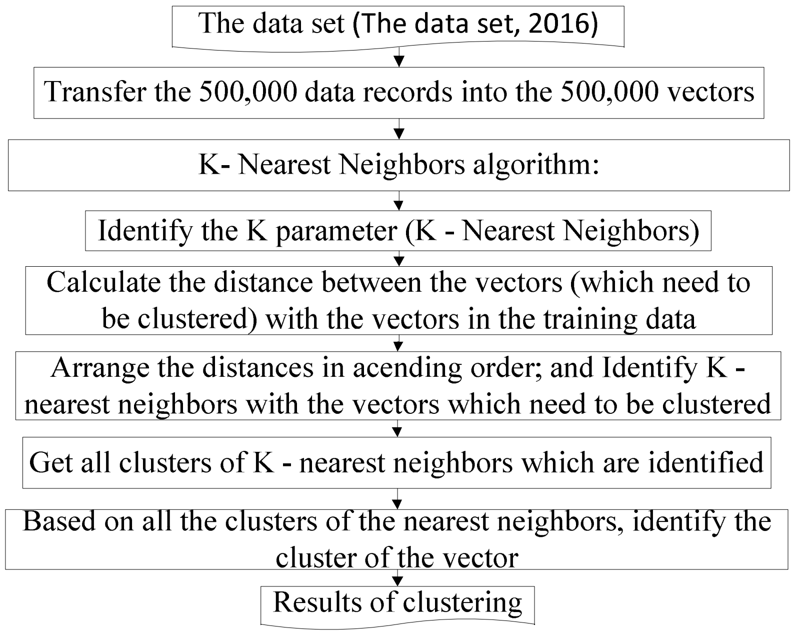

The data set (The data set, 2016)
Transfer the 500,000 data records into the 500,000 vectors
K- Nearest Neighbors algorithm:
Identify the K parameter (K - Nearest Neighbors)
Calculate the distance between the vectors (which need to be clustered) with the vectors in the training data
Arrange the distances in acending order; and Identify K - nearest neighbors with the vectors which need to be clustered
Get all clusters of K - nearest neighbors which are identified
Based on all the clusters of the nearest neighbors, identify the cluster of the vector
Results of clustering

Fig. 2: Overview of K-NN in the sequential system

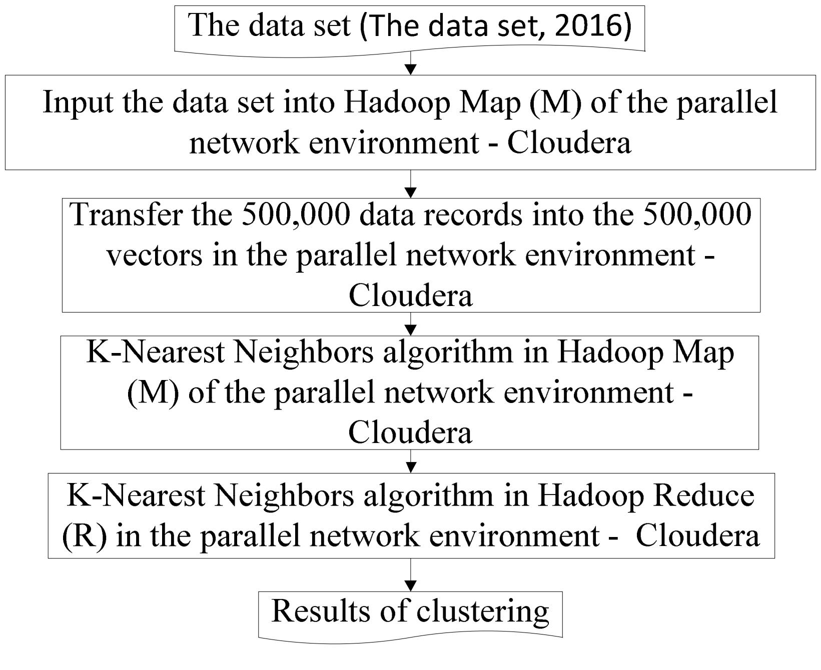

The data set (The data set, 2016)
Input the data set into Hadoop Map (M) of the parallel network environment - Cloudera
Transfer the 500,000 data records into the 500,000 vectors in the parallel network environment - Cloudera
K-Nearest Neighbors algorithm in Hadoop Map (M) of the parallel network environment - Cloudera
K-Nearest Neighbors algorithm in Hadoop Reduce (R) in the parallel network environment - Cloudera
Results of clustering

Fig. 3: Overview of the K-Nearest Neighbors algorithm in the Cloudera, called R-K-NN

The Fig. 4 shows the first phase of transferring the 500,000 data records into the 500,000 vectors in Cloudera.

The Fig. 5 illustrates the second phase.

After transferring the 500,000 data records into the 500,000 vectors in Cloudera, we reform K-NN in the distributed system which involves two phases as shown in the Fig. 6.

The Fig. 7 illustrates the first phase of K-NN in Hadoop Map in the parallel system.

The Fig. 8 illustrates the second phase of K-NN in Hadoop Reduce of the Cloudera distributed system.

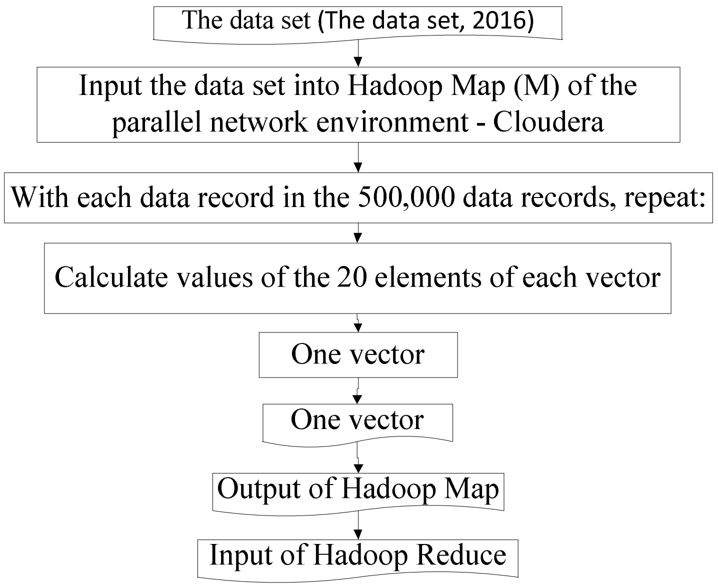

The data set (The data set, 2016)
Input the data set into Hadoop Map (M) of the parallel network environment - Cloudera
With each data record in the 500,000 data records, repeat:
Calculate values of the 20 elements of each vector
One vector
One vector
Output of Hadoop Map
Input of Hadoop Reduce

Fig. 4: The first phase of transferring the 500,000 data records into the 500,000 vectors in Cloudera

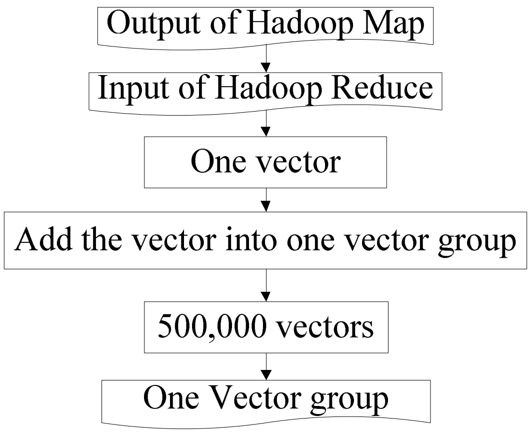


Fig. 5: The second phase of transferring the 500,000 data records into the 500,000 vectors in Cloudera

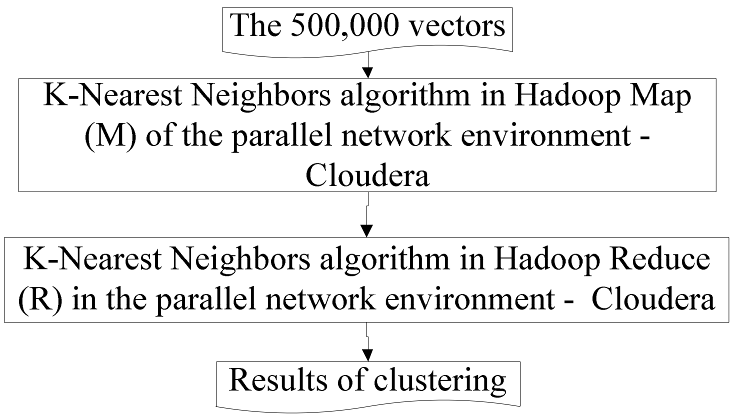

The 500,000 vectors
K-Nearest Neighbors algorithm in Hadoop Map (M) of the parallel network environment - Cloudera
K-Nearest Neighbors algorithm in Hadoop Reduce (R) in the parallel network environment - Cloudera
Results of clustering

Fig. 6: Overview of the K-Nearest Neighbors algorithm in the parallel network environment – Cloudera (R-K-NN)

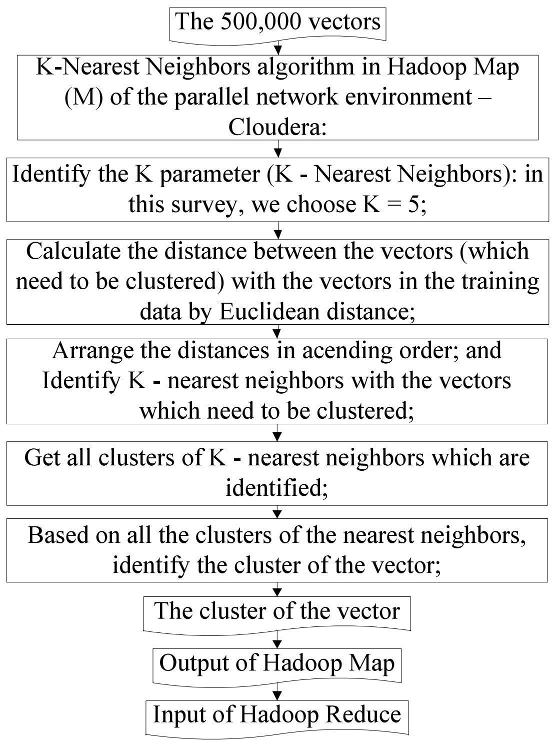

The 500,000 vectors
K-Nearest Neighbors algorithm in Hadoop Map (M) of the parallel network environment – Cloudera:
Identify the K parameter (K - Nearest Neighbors): in this survey, we choose K = 5;
Calculate the distance between the vectors (which need to be clustered) with the vectors in the training data by Euclidean distance;
Arrange the distances in acending order; and Identify K - nearest neighbors with the vectors which need to be clustered;
Get all clusters of K - nearest neighbors which are identified;
Based on all the clusters of the nearest neighbors, identify the cluster of the vector;
The cluster of the vector
Output of Hadoop Map
Input of Hadoop Reduce

Fig. 7: Overview of the K-Nearest Neighbors algorithm in Hadoop Map (M) in Cloudera

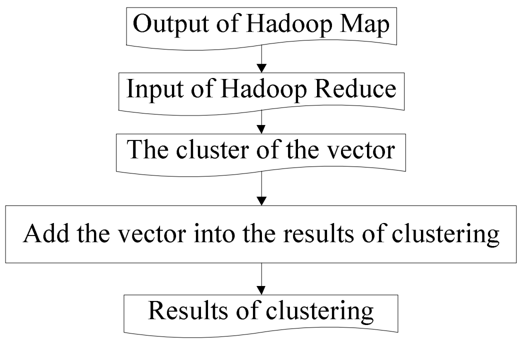

Output of Hadoop Map
Input of Hadoop Reduce
The cluster of the vector
Add the vector into the results of clustering
Results of clustering

Fig. 8: Overview of the K-Nearest Neighbors algorithm in Hadoop Reduce (R) in Cloudera

Experiment

We used a Java programming language to implement the novel model on a dataset with 500,000 observations (The Data Set, 2016).

We used one node (one server) to perform our survey in the sequential environment. We also used the Java programming language to program R-K-NN.

The server in the sequential system had the configuration which was Intel® Server Board S1200V3RPS, Intel® Pentium® Processor G3220 (3M Cache, 3.00 GHz), 2GB PC3-10600 ECC 1333 MHz LP Unbuffered DIMMs. Cloudera system was used as the operating system of the server.

We implemented R-K-NN in the Cloudera parallel network environment. The Java language was used in programming the application of R-K-NN in Cloudera. The Cloudera system comprised five nodes (five servers). The operating system of each server in the five nodes was Cloudera. The configuration of each server in the Cloudera system was Intel® Server Board S1200V3RPS, Intel® Pentium® Processor G3220 (3M Cache, 3.00 GHz), 2GB PC3-10600 ECC 1333 MHz LP Unbuffered DIMMs. All five nodes had the same configuration information.

The results of K-NN and R-K-NN are similar and due to space limitations, we do not present the detailed results in this study. The results of K-NN in the sequential environment are similar to that of R-K-NN in the Cloudera parallel environment. In addition, the execution time of K-NN in the sequential system is much shorter than that of R-K-NN in the Cloudera distributed system.

The execution times of K-NN in the sequential system and R-K-NN in the distributed system are shown in the Table 1.

To better understand the advantages of this study, we compare this work with other work related to the K-Nearest Neighbors algorithm in (Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al*., 2012; Kuncheva, 1995; Pan *et al*., 2015; Franco-Lopez *et al*., 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al*., 2005; Song and Roussopoulos, 2001), as shown in Table 2. As indicated in the Tables 2 and 3, no other studies used K-NN in the parallel network environment nor was there any research which was similar to the model developed in this work.

Next, we compare our study with those related to the distributed system reported in (Favuzza *et al*., 2006; Satyanarayanan *et al*., 2002; Babaoğlu *et al*., 1992; Fujimoto, 2001; Phu *et al*., 2016; Shirazi *et al*., 1995; Chen *et al*., 2011; Sulistio *et al*., 2004; Borges *et al*., 2001; Fujita *et al*., 1998) in the Table 4 and 5.

Table 1: The execution times of K-NN in the sequential environment and R-K-NN in the Cloudera parallel network

Our proposed model	Time
K-NN in the sequential environment	4,603 sec
R-K-NN in the Cloudera parallel system – 4 nodes	1,233 sec
R-K-NN in the Cloudera distributed system – 5 nodes	890 sec

Table 2: Comparison of our work with the studies related to K-NN in [(Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al*., 2012; Kuncheva, 1995; Pan *et al*., 2015; Franco-Lopez *et al*., 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al*., 2005; Song and Roussopoulos, 2001)]. Parallel network environment: PNE (or distributed system, parallel system)

Studies	K-NN	PNE	Model/method
Larose (2005)	Yes	No	k-Nearest Neighbor Algorithm
Fukunaga and Narendra (2006)	Yes	No	A branch and bound algorithm for computing k-Nearest Neighbors
Keller *et al*. (2012)	Yes	No	A fuzzy k-Nearest Neighbor algorithm
Kuncheva (1995)	Yes	No	Editing for the k-Nearest Neighbors rule by a genetic algorithm
Ruilin Pan *et al*. (2015)	Yes	No	Missing data imputation by k-Nearest Neighbours based on grey relational structure and mutual information
Franco-Lopez *et al*. (2001)	Yes	No	Estimation and mapping of forest stand density, volume and cover type using the k-Nearest Neighbors method
Callahan and Kosaraju (1995)	Yes	No	A decomposition of multi-dimensional point sets with applications to k-nearest neighbors and n-body potential fields
Horton and Nakai (1997)	Yes	No	Better prediction of protein cellular localization sites with the k-Nearest Neighbors classifier
Denoeux (2002)	Yes	No	A k-nearest neighbor classification rule based on Dempster-Shafer theory
Zhang and Zhou (2005)	Yes	No	A k-nearest neighbor based algorithm for multi-label classification
Seidl and Kriegel (1998)	Yes	No	Optimal multi-step k-nearest neighbor search
Mouratidis *et al*. (2005)	Yes	No	A threshold-based algorithm for the continuous monitoring of k-nearest neighbors
Song and Roussopoulos (2001)	Yes	No	K-Nearest Neighbor search for moving query point
Our work	Yes	Yes	An improved K-Nearest Neighbors algorithm in the Cloudera parallel network system

Table 3: Comparison of the advantages and disadvantages of our work with the studies related to K-NN in [(Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al*., 2012; Kuncheva, 1995; Pan *et al*., 2015; Franco-Lopez *et al*., 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al*., 2005; Song and Roussopoulos, 2001)]

Studies	Advantages	Disadvantages
Larose (2005)	The k-nearest neighbor algorithm is introduced in the context of a patient-drug classification problem. Voting for different values of k are shown to sometimes lead to differentresults. The distance function, or distance metric, is defined, with Euclidean distance being typically chosen for thisalgorithm. The combination Function is defined, for both simple unweighted voting and weighted voting. Stretching the axesis shown as a method for quantifying the relevance of various attributes. Database considerations, such as balancing,are discussed. Finally, k-nearest neighbor methods for estimationand prediction are examined, along with methods for choosing the best value for k	No mention
Fukunaga and Narendra (2006)	Experimental results demonstrate the efficiency of the algorithm. Typically, an average of only 61 distance computations were made to find the nearest neighbor of a test sample among 1000 design samples	No mention

Table 3: Continue

Keller *et al*. (2012)	The theory of fuzzy sets is introduced into the K-nearest neighbor technique to develop a fuzzy version of the algorithm. Three methods of assigning fuzzy memberships to the labeled samples are proposed and experimental results and comparisons to the crisp version are presented	No mention
Kuncheva (1995)	The performance has been evaluated on a medical data set by the rotation method. The results are reported together with those obtained with the standard k-NN, random selection, Wilson's technique and the MULTIEDIT algorithm	No mention
Pan *et al*. (2015)	The authors propose a novel method to impute missing data, named the feature weighted grey KNN (FWGKNN) imputation algorithm. This approach employs mutual information (MI) to measure feature relevance. The authors present an experimental evaluation for five UCI datasets in three missingness mechanisms with various missing rates. Experimental results show that feature relevance has a non-ignorable influence on missing data estimation based on grey theory and this method is considered superior to the other four estimation strategies. Moreover, the classification bias can be significantly reduced by using this approach in classification tasks	No mention
Franco-Lopez *et al*. (2001)	The authors describe the k-nearest neighbors (kNN) method for improving estimation and to produce wall-to-wall basal area, volume and cover type maps, in the context of the USDA Forest Service's Forest Inventory and Analysis (FIA) monitoring system. Several variations within the kNN were tested, including: Distance metric, weighting function, feature weighting parameters and number Of neighbors. Specific procedures to incorporate ancillary information and image enhancement techniques were also tested. Using the nearest neighbor (k=1), Euclidean distance, a three date 18-band composite image and feature weighting parameters, maps were constructed for basal area, volume and cover type	No mention
Callahan and Kosaraju (1995)	The authors define the notion of a well-separated pair decomposition of points in d-dimensional space. They then develop efficient sequential and parallel algorithms for computing such a decomposition. They apply the resulting decomposition to the efficient computation of k-nearest neighbors and n-body potential fields	No mention
Horton and Nakai (1997)	The result of tests using stratified cross validation shows the k-nearest neighbors classifier performs better than the other methods. In the case of yeast, this difference was statistically significant using a cross-validated paired t test. The result is an accuracy of approximately 60°/o for 10 yeast classes and 86% for 8 E.coli classes. The best previously reported accuracies for these data sets were 55% and 81% respectively	The authors consider a relatively minor error for two reasons. First, for some uses, the distinction between different types of inner membrane proteins may be immaterial. Second, the definition of the presence or absence of an uncleavable signal sequence is somewhat arbitrary and thus the labels for some training examples include some uncertainty. If the authors collapse the two classes to form a class "inner membrane protein without a cleavable signal sequence" a surprisingly high accuracy of 94% is attained
Denoeux (2002)	This approach provides a global treatment of such issues as ambiguity and distance rejection and imperfect knowledge regarding the class membership of training patterns. The effectiveness of this classification scheme as compared to the voting and distance-weighted k-NN procedures is demonstrated using several sets of simulated and real-world data	No mention
Zhang and Zhou (2005)	A multi-label lazy learning approach named ML-kNN is presented, which is derived from the traditional k-nearest neighbor (kNN) algorithm. In detail, for each new instance, its k-nearest neighbors are firstly identified. After that, according to the label sets of these neighboring instances, the maximum a posteriori (MAP) principle is utilized to determine the label set for the new instance. Experiments on a real-world multi-label bioinformatic data show that ML-kNN is highly comparable to existing multi-label learning algorithms.	No mention
Seidl and Kriegel (1998)	The authors present a novel multi-step algorithm which is guaranteed to produce the minimum number of candidates. Experimental evaluations demonstrate the significant performance gain over the previous solution and the authors observed average improvement factors of up to 120 for the number of candidates and up to 48 for the total runtime	No mention
Mouratidis *et al*. (2005)	The authors present a threshold-based algorithm for the continuous monitoring of nearest neighbors that minimizes the communication overhead between the server and the data objects. The proposed method can be used with multiple, static, or moving queries, for any distance definition and does not require additional knowledge (e.g., velocity vectors) besides object locations	No mention

Table 3: Continue

Song and Roussopoulos (2001)	Four different methods are proposed for solving the problem. Discussion about the parameters affecting the performance of the algorithms is also presented. A sequence of experiments with both synthetic and real point data sets are studied. In the experiments, the authors' algorithms always outperform the existing ones by fetching 70% less disk pages. In some settings, the saving can be as much as one order of magnitude	No mention
Our work	The execution time is shorter; the improved algorithm can process big datasets with millions of records in the shortest time.	It takes a lot of time and cost to implement this model; sometimes it causes confusion to implement true; in the process of making this model we must meet many errors

Table 4: Comparison of our work with the studies related to the parallel environment in (Favuzza *et al*., 2006; Satyanarayanan *et al*., 2002; Babaoğlu *et al*., 1992; Fujimoto, 2001; Phu *et al*., 2016; Shirazi *et al*., 1995; Chen *et al*., 2011; Sulistio *et al*., 2004; Borges *et al*., 2001; Fujita *et al*., 1998)

Studies	K-NN	PNE	Model/Method
Favuzza *et al*. (2006)	No	Yes	Adaptive and dynamic ant colony search algorithm for optimal distribution systems reinforcement strategy
Satyanarayanan *et al*. (2002)	No	Yes	A highly available file system for a distributed workstation environment
Babaoğlu *et al*. (1992)	No	Yes	An environment for parallel programming in distributed systems
Fujimoto (2001)	No	Yes	Parallel simulation: Parallel and distributed simulation systems
Phu *et al*. (2016)	No	Yes	Fuzzy C-means for english sentiment classification in a distributed system
Shirazi *et al*. (1995)	No	Yes	Scheduling and Load Balancing in Parallel and Distributed Systems
Chen *et al*. (2011)	No	Yes	Parallel Spectral Clustering in Distributed Systems
Sulistio *et al*. (2004)	No	Yes	A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools
Borges *et al*. (2001)	No	Yes	Composite reliability evaluation of sequential Monte Carlo simulation on parallel and distributed processing environments
Fujita *et al*. (1998)	No	Yes	Agent-based design model of adaptive distributed systems
Our research	Yes	Yes	An improved K-Nearest Neighbors algorithm in the Cloudera parallel network system

Table 5: Comparison of the advantages and disadvantages of our work with the studies related to the parallel environment in (Favuzza *et al*., 2006; Satyanarayanan *et al*., 2002; Babaoğlu *et al*., 1992; Fujimoto, 2001; Phu *et al*., 2016; Shirazi *et al*., 1995; Chen *et al*., 2011; Sulistio *et al*., 2004; Borges *et al*., 2001; Fujita *et al*., 1998)

Studies	Advantages	Disadvantages
Favuzza *et al*. (2006)	Distributed Generation (DG) technology considered in compound solutions with the installation of feeder and substations is viewed as a new option for solving distribution systems capacity problems, along several years. The objective to be minimized is therefore the overall cost of distribution systemsreinforcement strategy in a given time-frame. An application on a medium size network is carried out using the proposed technique that allows the identification of optimal paths in extremely large or non-finite spaces. The proposed algorithm uses an adaptive parameter in order to push exploration or exploitation as the search procedure stops in a local minimum. The algorithm allows the easy investigation of these kinds of complex problems and allows useful comparisons to be made as the intervention strategy and type of DG sources vary	No mention
Satyanarayanan *et al*. (2002)	The authors' goal in building Coda is to develop a distributed file system that retains the positive characteristics of AFS while providing substantially better availability. In this survey, the authors have shown how these goals have been achieved through the use of two complementary mechanisms, server replication and disconnected operation. The authors also show how disconnected operation can be used to support portable workstations.	Although Coda is far from maturity, the initial experience with it reflects favorably on its design. Performance measurements from the Coda prototype are promising, although they also reveal areas where further improvement is possible. The authors believe that a well-tuned version of Coda will indeed meet its goal of providing high availability without serious loss of performance, scalability, or security. A general question about optimistic replication schemes that remains open is whether users will indeed be willing to tolerate occasional conflicts in return for higher availability

Table 5: Continue

Babaoğlu *et al.* (1992)	The Paralex system aims to explore the extent to which the parallel application programmer can be liberated from the complexities of distributed systems. Paralex is a complete programming environment and makes extensive use of graphics to define, edit, execute and debug parallel scientific applications. All of the necessary code for distributing the computation across a network and replicating it to achieve fault tolerance and dynamic load balancing is automatically generated by the system. In this work, the authors give an overview of Paralex and present their experiences with a prototype implementation	No mention
Fujimoto (2001)	PADS expert Richard M. Fujimoto provides software developers with cutting-edge techniques for speeding up the execution of simulations across multiple processors and dealing with data distribution over wide area networks, including the Internet. With an emphasis on parallel and distributed discrete event simulation technologies, Dr. Fujimoto compiles and consolidates research results in the field spanning the last twenty years, discussing the use of parallel and distributed computers in both the modeling and analysis of system behavior and the creation of distributed virtual environments. While other works on PADS concentrate on applications, Parallel and Distributed Simulation Systems clearly show how to implement the technology. This work explains in detail the synchronization algorithms needed to properly realize the simulations, including an in-depth discussion of time warp and advanced optimistic techniques. Finally, the study is richly supplemented with references, tables and illustrations and examples of contemporary systems such as the Department of Defense's High Level Architecture (HLA), which has become the standard architecture for defense programs in the United States	No mention
Phu *et al.* (2016)	It processes big data involving millions of English documents. The execution time of this model to conduct sentiment analysis on big data is short	It takes a long time to implement and it is costly to build the algorithms of the model in the distributed system
Chen *et al.* (2011)	To perform clustering on large data sets, the authors investigate two representative ways of approximating the dense similarity matrix. The authors compare one approach by sparsifying the matrix with another by the Nyström method. The authors then pick the strategy of sparsifying the matrix via retaining nearest neighbors and investigate its parallelization. The authors parallelize both memory use and computation on distributed computers. Through an empirical study on a document data set of 193; 844instances and a photo data set of 2; 121; 863, the authors show that the authors' parallel algorithm can effectively handle large problems.	No mention
Sulistio *et al.* (2004)	This work presents a taxonomy for computer simulations and applies the taxonomy on simulation tools for PDSs. The taxonomy comprises PDS, usage, simulation and design taxonomies. The design taxonomy emphasizes respective components and features of a simulation tool, including the simulation engine, modeling framework, programming framework, design environment, user interface and system support. A number of selected simulation tools have been surveyed using the taxonomy. This study thus helps to identify some approaches for designing simulation tools for PDSs and possible future research directions	No mention
Borges *et al.* (2001)	This work describes two parallel methodologies for composite reliability evaluation using sequential Monte Carlo simulation. The methodologies are based on coarse grain asynchronous implementations. In the first methodology, a complete simulation year is analyzed on a single processor and the many simulated years necessary for convergence are analyzed in parallel. In the second methodology, the adequacy analysis of the system operating states within the simulated years is performed in parallel and the convergence is checked on one processor at the end of each simulated year. The methodologies are implemented on a 10-node IBM RS/6000 SP scalable distributed memory parallel computer and on a network of 8 IBM RS/6000 43P workstations.The results obtained in tests with actual power system models showed high speed-up and efficiency on both parallel platforms.	No mention
Fujita *et al.* (1998)	The aim of the authors' research is to establish a new design model of an adaptive distributed system (ADS) to deal with various changes occurred in the system environment. In this research, the authors propose an agent-based architecture of ADS, based on the agent-based computing paradigm. Then, the authors implement a prototype of the ADS with respect to video conferencing applications and also evaluate the adaptive functions of the ADS realized on the basis of the proposed architecture	No mention
Our research	The execution time is shortened; it can process big datasets with millions of records in a shortened time	It takes a lot of time and cost to implement this model; sometimes it causes confusion to implement true; in the process of making this model we must meet many errors

Results and Discussion

As shown in Table 1, the execution time of K-NN in the sequential system is 14,603 sec whereas that of R-K-NN in the Cloudera distributed environment (four nodes) is 1233 sec; and that of R-K-NN in the Cloudera parallel system (five nodes) is 890 sec.

The results of the sequential environment are similar to those in the Cloudera distributed system. The execution time in the sequential system is longer than that in the Cloudera parallel system.

The execution time of R-K-NN in the distributed environment is up to the performance similar to the performance of the parallel system. If the performance of the distributed system is higher, R-K-NN is faster.

Conclusion

In this research, we developed an improved K-Nearest Neighbors algorithm to implement in the Cloudera parallel network environment for processing big datasets containing millions of records. The data set used in our work comprises 500,000 data records (The Data Set, 2016).

Based on K-NN in (Larose, 2005; Fukunaga and Narendra, 2006; Keller *et al*., 2012; Kuncheva, 1995; Pan *et al*., 2015; Franco-Lopez *et al*., 2001; Callahan and Kosaraju, 1995; Horton and Nakai, 1997; Denoeux, 2002; Zhang and Zhou, 2005; Seidl and Kriegel, 1998; Mouratidis *et al*., 2005; Song and Roussopoulos, 2001), we build algorithms related to K-NN to implement this model in the Cloudera environment.

Our model has many advantages and disadvantages. The advantages are: The execution time is shorter and it can process big datasets with millions of records in a shortened time. However, its disadvantages are: It takes a lot of time and cost to perform construct and implement this model; sometimes it causes confusion to implement true; in the process of making this model we must meet many errors.

As shown in the Table 3, the studies related to the distributed network environment did not use K-NN hence, they are not similar to this work. It is well known that K-NN is not efficient for big data sets. Our proposed model solved the problems very effectively and R-K-NN was able to be implemented on very large data sets. As shown in Table 1, the experiment results proved that the execution time of R-K-NN is much shorter than that of K-NN for the same data set.

In the near future, we will use the results of the proposed model to classify the semantics (positive, negative, neutral) of millions of English documents.

Author's Contributions and Ethics

"Dr.Vo Ngoc Phu"is a main author and "Dr.VoThi Ngoc Tran" is the second co-author of our manuscript.

"Dr.VoThi Ngoc Tran" built our data sets and "Vo Ngoc Phu" checked them finally.

"Dr.Vo Ngoc Phu" implemented this survey and "Dr.VoThi Ngoc Tran" helps "Dr.Vo Ngoc Phu" a lot to perform this study.

"Dr.VoThi Ngoc Tran" wrote the draft document of our manuscript; and "Dr.Vo Ngoc Phu" checked, fixed, and wrote it finally.

Conflict of Interest

The authors declare that they have no conflict of interest.

Future Work

Based on the results of this proposed model, many future projects can be proposed, such as creating full emotional lexicons in a parallel network environment to shorten execution times, creating many search engines, creating many translation engines, creating many applications that can check grammar correctly. This model can be applied to many different languages, creating applications that can analyze the emotions of texts and speeches and machines that can analyze sentiments.

References

Apache, 2016. http://apache.org

Babaoğlu, O., L. Alvisi, A. Amoroso, R. Davoli and L.A. Giachini, 1992. Paralex: An environment for parallel programming in distributed systems. Proceedings of the 6th International Conference on Supercomputing, Jul. 19-24, ACM, Washington, D. C., USA, pp: 178-187. DOI: 10.1145/143369.143406

Borges, C.L.T., D.M. Falcao, J.C.O. Mello and A.C.G. Melo, 2001. Composite reliability evaluation by sequential Monte Carlo simulation on parallel and distributed processing environments. IEEE Trans. Power Syst., 16: 2003-209. DOI: 10.1109/59.918287

Callahan, P.B. and S.R. Kosaraju, 1995. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. J. ACM, 42: 67-90. DOI: 10.1145/200836.200853

Chen, W.Y., Y. Song, H. Bai, C.J. Lin and E.Y. Chang, 2011. Parallel spectral clustering in distributed systems. IEEE Trans. Pattern Analysis Machine Intelligence, 33: 568-586. DOI: 10.1109/TPAMI.2010.88

Cloudera, 2016. http://www.cloudera.com

Dat, N.D., V.N. Phu, V.T.N. Tran, V.T.N. Chau and T.A. Nguyen, 2016. STING algorithm used English semantic classification in parallel environment. Int. J. Pattern Recognition Artificial Intelligence.

Denoeux, T., 2002. A k-nearest neighbor classification rule based on Dempster-Shafer theory. IEEE Trans. Syst. Man Cybernet.

Favuzza, S., G. Graditi and E.R. Sanseverino, 2006. Adaptive and dynamic ant colony search algorithm for optimal distribution systems reinforcement strategy. Applied Intelligence, 24: 31-42. DOI: 10.1007/s10489-006-6927-y

Franco-Lopez, H., A.R. Ek and M.E. Bauer, 2001. Estimation and mapping of forest stand density, volume and cover type using the k-nearest neighbors method. Remote Sens. Environ., 77: 251-274. DOI: 10.1016/S0034-4257(01)00209-7

Fujimoto, R.M., 2001. Parallel simulation: parallel and distributed simulation systems. Proceedings of the 33nd Conference on Winter Simulation, Dec. 09-12, IEEE Computer Society Washington, DC, USA, Arlington, Virginia, pp: 147-157.

Fujita, S., H. Hara, K. Sugawara, T. Kinoshita and N. Shiratori, 1998. Agent-based design model of adaptive distributed systems. Applied Intelligence, 9: 57-70. DOI: 10.1023/A:1008299131268

Fukunaga, K. and P.M. Narendra, 2006. A branch and bound algorithm for computing k-nearest neighbors. IEEE Trans. Comput., 24: 750-753. DOI: 10.1109/T-C.1975.224297

Hadoop, 2016. http://hadoop.apache.org

Horton, P. and K. Nakai, 1997. Better prediction of protein cellular localization sites with the it k nearest neighbors classifier. Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology, Jun. 21-26, pp: 147-152.

Keller, J.M., M.R. Gray and J.A. Givens, 2012. A fuzzy K-nearest neighbor algorithm. IEEE Trans. Syst. Man Cybernet.

Kuncheva, L.I., 1995. Editing for the k-nearest neighbors rule by a genetic algorithm. Pattern Recognition Lett., 16: 809-814. DOI: 10.1016/0167-8655(95)00047-K

Larose, D.T., 2005. k-Nearest Neighbor Algorithm. In: Discovering Knowledge in Data: An Introduction to Data Mining, Larose, D.T. (Ed.), John Wiley & Sons, ISBN-10: 0471687537.

Mouratidis, K., D. Papadias, S. Bakiras and Y. Tao, 2005. A threshold-based algorithm for continuous monitoring of k nearest neighbors. IEEE Trans. Knowledge Data Eng., 17: 1451-1464. DOI: 10.1109/TKDE.2005.172

Pan, R., T. Yang, J. Cao, K. Lu and Z. Zhang, 2015. Missing data imputation by K nearest neighbours based on grey relational structure and mutual information. Applied Intelligence, 43: 614-632. DOI: 10.1007/s10489-015-0666-x

Phu, V.N., N.D. Dat, V.T.N. Tran, V.T.N. Chau and T.A. Nguyen, 2016. Fuzzy C-means for english sentiment classification in a distributed system. Applied Intelligence, 46: 717-738. DOI: 10.1007/s10489-016-0858-z

Satyanarayanan, M., J.J. Kistler, P. Kumar, M.E. Okasaki and E.H. Siegel *et al*., 2002. Coda: A highly available file system for a distributed workstation environment. IEEE Trans. Comput.

Seidl, T. and H.P. Kriegel, 1998. Optimal multi-step k-nearest neighbor search. Proceedings of the International Conference on Management of Data, Jun. 01-04, ACM, Seattle, Washington, pp: 154-165. DOI: 10.1145/276304.276319

Shirazi, B.A. K.M. Kavi and A.R. Hurson, 1995. Scheduling and Load Balancing in Parallel and Distributed Systems. 1st Edn., Wiley-IEEE Computer Society Press, Los Alamitos, CA, USA, ISBN-10: 0818665874, pp: 448.

Song, Z. and N. Roussopoulos, 2001. K-nearest neighbor search for moving query point. Proceedings of the 7th International Symposium on Spatial and Temporal Databases, Springer, Berlin, Heidelberg, Redondo Beach, CA., pp: 79-96. DOI: 10.1007/3-540-47724-1_5

Sulistio, A., C.S. Yeo and R. Buyya, 2004. A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. Softw. Pract. Exper., 34: 653-673. DOI: 10.1002/spe.585

The Data Set, 2016. http://data-mining-tutorials.blogspot.com/2008/11/decision-tree-and-large-dataset.html

Zhang, M.L. and Z.H. Zhou, 2005. A k-nearest neighbor based algorithm for multi-label classification. Proceedings of the International Conference on Granular Computing, Jul. 25-27, IEEE Xplore Press, Beijing, China. DOI: 10.1109/GRC.2005.1547385