






# Anonymizing $k$ -NN Classification on MapReduce

Sibghat Ullah Bazai<sup>(✉)</sup>, Julian Jang-Jaccard, and Ruili Wang

Institute of Natural and Mathematical Sciences, Massey University,  
Auckland, New Zealand  
{s.bazai, j.jang-jaccard, r.wang}@massey.ac.nz

**Abstract.** Data analytics scenario such as a classification algorithm plays an important role in data mining to identify a category of a new observation and is often used to drive new knowledge. However, classification algorithm on a big data analytics platform such as MapReduce and Spark, often runs on plain text without an appropriate privacy protection mechanism. This leaves user's data to be vulnerable from unauthorized access and puts the data at a great privacy risk. To address such concern, we propose a new novel  $k$ -NN classifier which can run on an anonymized dataset on MapReduce platform. We describe new Map and Reduce algorithms to produce different anonymized datasets for  $k$ -NN classifier. We also illustrate the details of experiments we performed on the multiple anonymized data sets to understand the effects between the level of privacy protection (data privacy) and the high-value insights (data utility) trade-off before and after data anonymization.

**Keywords:** MapReduce · Data anonymization ·  $K$ -anonymity  
 $k$ -NN classification

## 1 Introduction

In recent years, we witness big data containing a huge amount of personal data such as seen in the data acquired by government administrations, health insurances, social networking sites and IoT devices, to name a few. This exponential growth of big data has demanded a requirement for a system which can provide powerful computation and other related technologies. A big data processing framework using distributed environment, such as MapReduce and Spark, has been widely used to handle such computation to find insights such as correlation between large datasets using Machine Learning algorithms.

In Machine Learning, classification algorithms play an important role in data mining to identify a category of a new observation of data into a set of predefined classes or groups. The  $k$ -Nearest Neighbour ( $k$ -NN) [1] is one of such classification methods. In recent years, we witness the adoption of  $k$ -NN algorithm in distributed environments to overcome the computational intensity of having to compare distances of every single training data point.

However, processing  $k$ -NN in MapReduce raises a number of security issues. In MapReduce, Mappers transform the original input key/value pairs into intermediate

key/value pairs after some calculation while reducers aggregate the intermediate values, compute and write them to an output file. These operations at different stage of MapReduce operations are done on plain text which is vulnerable from unauthorized access that puts users' data at a privacy risk. The unauthorized privacy attack can either directly leak sensitive information or indirectly leak information via composite attacks where the adversary can link users' data, illegally obtained at various stage of MapReduce, with public information available via different sources such as Facebook or Twitter.

Providing privacy guarantee during computations of sensitive data can be achieved using privacy preserving techniques such as  $K$ -anonymity [2].  $K$ -anonymity uses *generalization* to hide individual features (also called attribute) or records (also called as tuples) within a crowd or *suppression* to remove highly sensitive records. The size of crowd is typically determined by a privacy parameter  $K$  group size. The use of  $K$ -anonymity in MapReduce platform to provide a certain level of privacy are found in [3–5]. However, these existing studies do not illustrate how to process a privacy preserving technique such as  $K$ -anonymity to be applied in different data analytics scenarios such as classification.

Extending from our earlier study where we illustrated how to apply a  $K$ -anonymity in aggregation scenario [4], this time we illustrate how one can apply a  $K$ -anonymity in a classification scenario that utilizes  $k$ -NN algorithm. We propose a  $k$ -NN classifier which can run on an anonymized data in the MapReduce platform. To the best of our knowledge, our proposed algorithm in this paper is the first attempt to address the classification implementation on anonymized data in the MapReduce platform. Our main contributions are;

- We illustrate the details of the  $k$ -NN classifier algorithms that can run on an anonymized dataset.
- We demonstrate that it is possible to generate different sets of anonymized data using varying degree of privacy parameters (i.e.,  $K$  group size) either applied in the different number of features or the different number of records in the  $K$ -anonymity algorithm.
- We illustrate that different classification results can be obtained based on the different sets of anonymized data sets.
- We provide the impact in the trade-off between the level of privacy protection (data privacy) and the high-value insights (data utility) on classification before and after different anonymized data.

The rest of the paper is organized as follows. In Sect. 2, we provide the necessary background knowledge needed for the paper. In Sect. 3, we describe the related work. In Sect. 4, we describe the details of data anonymization strategies we use and explain the algorithms needed for Map and Reducer operations. In Sect. 5, the experiments and results are discussed. Finally, we conclude our work and discuss the future work planned ahead of us in Sect. 6.

## 2 Background

### 2.1 $k$ -Nearest Neighbour

The  $k$ -nearest neighbour method ( $k$ -NN) is one of the most widely used classification algorithm in machine learning. Cover and Hart in 1967 formally introduced the original idea of  $k$ -NN and its properties [1].  $k$ -NN works directly on the actual instances of the training data as it does not require building a model to represent the underlying statistics and distributions of the original training data [1].  $k$ -NN is based on learning by analogy, that is, by comparing a given test record with training record sets.

Euclidean Distance ( $ED$ ) is often used to measure the distance of two records where the distance indicates the degree of difference (i.e., if  $ED$  is small the two records are likely to be similar while two records are different if  $ED$  is big). The distance measure based on  $ED$  is defined as (1):

$$D(X, Y) = \|X - Y\| = \sum_{i=1}^p (X(i) - Y(i))^2 \quad (1)$$

where  $X(i)$ ,  $Y(i)$  are the  $i$ th dimension attribute values of vector  $X$ ,  $Y$  respectively. In  $k$ -NN classification, an output can be seen as a class membership as an object is classified by a majority vote of its neighbours. Thus, a class is typically assigned to the object based on the most common classes observed among its neighbours.

There are many different ways to implement  $k$ -NN algorithms including where the classification should be performed (e.g., [6] proposed the centralized paradigm where the  $k$ -NN join is performed on a single centralized server) and looking into improving performance overheads (e.g. Parallelization of  $k$ -NN algorithm [7]). Especially, many existing approaches have been criticized as the computation costs sharply rise when the number of dimensions and the sizes of training sets become large. The use of MapReduce as a processing platform has been regarded as a practical solution to resolve such criticism. The MapReduce framework takes the input data, depending on the size, it automatically splits the input data into smaller manageable chunks. Each smaller chunk is processed by a map task (also often called a mapper interchangeably). The result of a mapper is summarized as key and value pairs. The output (e.g., values) with the same keys are shuffled and reduced by a reducer function.

### 2.2 $K$ -Anonymity

$K$ -anonymity is one of the first data anonymization techniques with formal mathematical support as a proof. Sweeney [2] introduced  $K$ -anonymity in 2002 by stating that without ensuring  $K$  individuals in aggregation single aggregate statistic should not be published. This definition helps every user being able to hide in  $K-1$  crowd [9]. In his definition, Quasi-Identifiers ( $QID$ ) are attributes in a dataset which may be linked to a publicly available dataset. The main goal to achieve  $K$ -anonymity is to replace  $QID$  values with more general values so that  $QIDs$  cannot be linked to an individual.

$K$ -anonymity is typically achieved by using two techniques called *generalization* and *suppression* with the aim to decrease  $QIDs$  (i.e., there is less obvious identifier to link individual data). Using *generalization*, more granular values are combined

together to create a broader category. This can be achieved both for numerical variables (e.g., *generalization* the monthly salary of \$56,600, \$52,300, and \$73,320 to a single value “above \$50,000”) and for categorical variables (e.g., generalizing the separate degrees of “bachelor”, “masters”, and “PhD” into a single “higher degree”). *Generalization* replaces the original record attributes with less exact but constant values. *QIDs* becomes generalized to a certain point where a few conclusions can be drawn about their relationship with other records. However, the core art of this technique is to understand as what is the optimal level of *generalization* for a given data because repeated *generalization* could decrease the quality of the entire data set. *Suppression* technique involves the removal of records that violates anonymity standards from the data set entirely rather than chaining the value of the records. Also, it is necessary to take a considerable caution because *suppression* can skew the integrity of a dataset when values are eliminated disproportionately to the original distribution of the data. Most often, *suppression* is used in conjunction with the *generalization* to improve the anonymization efficiency. For example, the records that were not within the boundary of  $K$ -anonymity after *generalization* can be automatically suppressed.

### 3 Related Work

Performing  $k$ -NN to provide performance gain has been extensively studied in the literature [7]. Nevertheless, this work only focuses on the centralized and single-thread approach that is not applicable in many modern day applications which requires a large input data for computation. In [8], the authors reported the nearest neighbour classification with *generalization* applied in a large dataset. The main purpose of generalizing exemplars, which merges data into hyper-rectangles, is to improve speed and accuracy but they do not mention how to handle anonymized data. The study in [9] reported a privacy preserving classification techniques. However, the techniques they use focus on neural network as an underlying algorithm then use homomorphic encryption as a data anonymization technique. The direction they took is quite orthogonal to our work. In [10], the authors propose a new nearest neighbour approach using correlation analysis under a MapReduce framework on a Hadoop platform to address the difficult problem of real-time prediction with very large training data sets. However, using their approach, the performance of an algorithm can be seriously affected if the size of the training samples becomes extremely large. For many modern day uses of  $k$ -NN, the computational and the storage issues has become a critical problem [11]. This is because the new applications of  $k$ -NN requires a rather large storage device to contain the whole training set as well as a large computation support in the classification stage.

Airavat [12] proposed a framework for MapReduce by defining mandatory access control (MAC) with differential privacy (DP) on a secure operating system SELinux. Airavat however describes a data anonymization via DP only with a very strict sensitivity pre-defined value which is only applicable to a specific case of applications where the distribution of the input data and types of operations performed on that data is pre-defined.

## 4 Data Anonymization

In this section, we discuss the details of  $K$ -anonymity on  $k$ -NN classifier for MapReduce operations. Our implementation is done based on the privacy-preserving platform we proposed previously [4]. The privacy preserving mechanism receives the user input data and defines a privacy protection mechanism, in our case  $K$ -anonymity. In the algorithm implementation layer, we choose  $k$ -NN as a classifier, transforms the original data into an anonymized equivalent still retaining the content value so that further analytics can be performed on the anonymized data. We measure the classification error on the privacy and utility measurement layer to understand the privacy and utility trade-off between the original data and the anonymized data.

### 4.1 Dataset and Pre-processing

We use the Adult dataset [13] to demonstrate our study. The dataset consists of personal information records extracted from the US census database. We use the dataset for a classification prediction as whether a given person has a potential to earn an annual income over or under \$50,000. The original Adult data set has six continuous and eight categorical features as seen in Table 1.

**Table 1.** Original adult dataset

Age	Workclass*	Fnlwgt	Edu	Edu-num	Marital-status	Occupation	Relationship	Race	Sex	Capital-gain	Capital-loss	Hours-per-week	Native-country	Income
66*	Private	142624	Assoc-acdm	12	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	5556	0	40	Yugoslavia	>50K
55	Private	160631	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	4508	0	8	Yugoslavia	<=50K
53	Private	153064	5th-6 <sup>th</sup>	3	Married-civ-spouse	Exec-managerial	Husband	White	Male	7688	0	10	Yugoslavia	>50K
51	Private	179479	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	3325	0	40	Yugoslavia	<=50K
51	Federal-gov	223206	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	15024	0	40	Vietnam	>50K

\*a light gray represents an example of a tuple

\*\* a dark gray shade represents an example of a feature

The  $k$ -NN algorithm often processes both categorical features and continuous features [1]. To overcome the difficulty of having to process the string data often found in categorical features, many implementations of  $k$ -NN often require the conversion of the categorical features to discrete numerical features. We adopt the conversion from the work of [14], which utilizes unique numerical labels to convert each categorical value into its numerical counterparts. Using this technique, we transform eight categorical features (workclass, edu, marital-status, occupation, relationship, race, sex, native-country) into numerical features. For example, instead of using a country name such as Cambodia, Canada, China a numeric value is used such as 1 to represent the country Cambodia, 2 to as Canada so on. Table 2 represents the Adult dataset after the conversation of the categorical values.

**Table 2.** Adult Dataset after categorical value conversation to numeric

Age	Work-class**	Fnlwgt	Edu	Edu-num	Marital-status	Occupation	Relationship	Race	Sex	Capital-gain	Capital-loss	Hours-per-week	Native-country	Income
66	3	142624	8	12	3	7	1	5	1	5556	0	40	35	>50 K
55	3	160631	12	9	3	7	1	5	1	4508	0	8	35	≤ 50 K
53	3	153064	5	3	3	4	1	5	1	7688	0	10	35	>50 K
51	3	179479	12	9	7	4	2	5	2	3325	0	40	35	≤ 50 K
51	1	223206	11	16	3	9	1	2	1	15024	0	40	34	>50 K

## 4.2 $k$ -NN Implementation on MapReduce

This section defines the  $k$ -NN algorithm we implemented in MapReduce operations. Our implementation strategies were inspired by the work on [15]. We use the following  $k$ -NN algorithm to get classification errors before data was anonymized. The general processing of data for MapReduce operations follows.

- Reading data: Consider a training dataset  $TR_s$  and a test dataset  $TS_s$ , they are formed by a number of records  $m$ -th (in  $TR_s$ ) and  $t$ -th (in  $TS_s$ ) respectively. Each training sample  $ST_r$  (line) is read and split as a tuple  $(Tp_1, Tp_2, \dots, Tp_D, W_c)$ , where,  $Tp_E$  represent  $E$ -th feature in  $p$ -th tuple, and  $ST_r$  belongs to a class  $W_c$ , for given  $T_p^{W_c}$  and  $D$  diminutions.
- $k$ -NN training: In order to train the  $k$ -NN algorithm, the training dataset  $TR_s$  should contain the value of  $W_c$  while it is unknown for the test dataset  $TS_s$ . For each test sample  $ST_s$  contained in the  $TS_s$  test dataset, the  $k$ -NN model looks for records whose distance proximity is smallest (i.e., indicating the records are similar) in the  $TR_s$  set. To do this, it computes the Euclidean Distances ( $ED$ ) between  $TS_s$  and all  $ST_r$  of  $TR_s$  (i.e., for each sample of test data set with all the sample of train data set). Whereas the  $k$ -nearest neighbours samples ( $NB_1, NB_2, \dots, NB_k$ ) are obtained by ranking the training samples according to the computed distance.
- Alignment with Mapper operations: To apply this in the MapReduce model, we first organize a mapper to compute the classes  $W_c$  from the distance to the  $k$  nearest neighbours for each test and training data.
- Alignment with Reducer operations: The reduce function is responsible for processing the  $ED$  of the  $k$  nearest neighbours from each map and creates a list of  $k$  nearest neighbours by taking those with minimum distance. Reducer shuffles the distances and examines for majority voting, then to assign the  $W_c$  class for  $TS_s$ .  $k$ -NN mapper and  $k$ -NN reducer are described in more detail as follows:

**$k$ -NN Mapper:** In our implementation of  $k$ -NN for MapReduce, we represent our training set as  $TR_s$  and test dataset  $TS_s$ , both with a random number of records store in Hadoop Distributed File System (HDFS) as single file. The first step Mapper accesses the input file from the HDFS and disjoint  $TR_s$  into given number subsets. The training set  $TR_s$  is split into tuples containing the attributes (also known as features)  $(test\_tp_1, test\_tp_2, \dots, test\_tp_D, W_c)$ , where, each  $test\_tp$  represent one feature of adult data set and  $W_c$  represent as an income class (the feature to be classified). Suppose, we have mappers from 1 to  $n$ , for each of the mapper task, it will create  $TR_{s_j}$  from  $1 \leq j \leq n$ , which represent the training set sample  $ST_r$ . It should be noted that partitions of given

processes are sequentially executed, for example,  $mapper_j$  corresponds to  $j$  data chunk. In other words, each mapper will have its corresponding  $TR_{sj}$  and a class label  $W_c$  for every  $k$  nearest neighbours. Each training record is divided into a subset of  $TR_s$  in order to compare each subset with its  $TS_s$  to find out a distance  $DC$ . The other small subsets are obtained based on  $k$  (degree of neighbours) and number of records in  $TS_s$ . Distances are stored in the distance matrix  $DC_j$  pairs as “<class, distance>” which can be represent as  $\langle W_c, k\text{-distance} \rangle$  with dimension  $k.m$  (i.e.,  $DC_j$  compute all the distances for each tuple of  $TR_s$  with all element of  $TS_s$ ). Each Row  $i$  will have  $W_c$  (classifier value) and  $k$ -nearest distance of class. The row  $i$  will repeat till  $t$  for each  $ST_s$ . After mapper completes its process, it stores the <key, value> pairs as  $\langle (Mapper_{ID}, W_c), DC_j \rangle$ , where  $Mapper_{ID}$  is used to identify the mapper in single reducer. The complete pseudo-code for the  $k$ -NN mapper is described in the following Algorithm 1.

---

**Algorithm 1.**  $k$ -NN Mapper
 

---

**Input:**  $k$  value,  $TS_s$ .

**Output:** key as mapper identifier  $Mapper_{ID}$  and class  $W_c$  value as  $DC_j$

```

1: Create  $TR_{sj}$  with the instances of split  $j$ .
2: for  $i = 0$  to  $i < \text{size}(TS_s)$  do
3:   Compute  $k$ -NN ( $ST_s, i, TR_{sj}, k$ )
4:   for  $m = 0$  to  $m < k$  do
5:      $DC_j(i, m) = \langle W_c(NB_m), ST_s(NB_m) \rangle$ 
6:   end for
7: end for
8: key =  $Mapper_{ID}, W_c$ 
9: return  $\langle \text{key}, DC_j \rangle$ 

```

---

**$k$ -NN Reducer:** Reducer is responsible for selecting most relevant neighbours, examines their classes and finds optimal classes for tuples in  $TS_s$ . The reducer phase can be divided into following four steps:

1. The setup step reads and allocates the distance matrix  $DC_{\text{reduce}}$  of the fix size of  $(TS_s, k\text{-neighbours})$ .  $DC_{\text{reduce}}$  value is assigned once a mapper completes  $DC_j$  and sends the data to reducer. Keys from the mapper is separated as  $Mapper_{ID}$  and  $W_c$ . The size of the distance matrix is initialized with the total number of  $TS_s$ . Once the setup is done, it moves to the next reduce step.
2. The reduce merge two sorted lists (i.e., one list containing the distances calculated with class and the other list contains the distances calculated with its neighbours). Thus, for every  $TS_s$ , every distance is compared to its neighbours one at a time starting from the nearest one and sorted according to distances.
3. The third step is cleanup. The cleanup process receives the list of neighbours for all  $TS_s$  as (class, distance) in the form of  $DC_{\text{reduce}}$  for majority voting in order to identify the predicted classes  $W_{cp}$  for  $TS_s$ . After the cleanup, the key value pair are redefined as  $TS_s$  classes  $W_{cp}$  and  $TR_s$  classes  $W_c$ .

4. The comparison of the classifiers between these two classes are done in the classification error step to compute the error rate of  $k$ -NN for each  $k$  values. Following is the pseudocode that we use in this study for the  $k$ -NN reducer.

---

**Algorithm 2.**  $k$ -NN Reducer
 

---

**Input:** Size of  $TS_s$ ,  $k$  and  $DC_j$ .

**Output:** actual class  $W_c$  and predicated class  $W_{cp}$  as key and  $Error\_rate$  as value

```

1: for  $i = 0$  to  $i < size(TS_s)$  do           //setup
2:    $cont = 0$ 
3:   for  $m = 0$  to  $k$  do                       //reduce
4:     if  $DC_j(i, cont). ST_s < DC_{reduce}(i, m). ST_s$  then
5:        $DC_{reduce}(i, m) = DC_j(i, cont)$ .
6:        $cont ++$ 
7:     end if
8:   end for
9: end for
10:  $Error\_rate = 0, TP = 0, TN = 0$ 
11: for  $l = 0$  to  $l < size(TS_s)$  do           // CleanUp
12:    $PredClass_l = MajorityVoting(Classes(DC_{reduce}))$ 
13:    $key = W_{cp}, W_c$ 
14:   if  $(W_{cp} == W_c)$  then
15:      $TP ++$ 
16:   else
17:      $TN ++$ 
18:   end if
19:    $Error\_rate = (TP - TN / TP)$            // classification error
20: end for
21: return ( $<Error\_rate >$ )

```

---

### 4.3 K-Anonymity with $k$ -NN in MapReduce

In this section, we illustrate how we anonymize the original input data using  $k$ -anonymity technique. We run  $k$ -NN classifier on the anonymized data set and get classification error. We compare the classifications errors obtained from a non-anonymized dataset as well as an anonymized dataset to understand whether there has been any impact on classification error. For this task, we extend the mapper operation in the previous section and produce multiple sets of anonymized data sets.

1. The first step is to read  $TS_s$  into tuples containing the attributes (also known as features) ( $test\_tp_1, test\_tp_2, \dots, test\_tp_D, W_c$ ), where, each  $test\_tp$  represent one feature of adult data set and  $W_c$  represent as an income class (feature to be classified). The second step is to anonymize the number of features ( $\alpha$ ), while  $test\_tp_\alpha$  denote the particular feature to be anonymized.
2. Then the third step is to assign  $K$  group size ( $KG$ ) where  $KG$  is the degree of anonymity (i.e., the number of records to hide in a crowd).



3. The forth step is to calculate an average value on each attribute of  $\alpha$  *QIDs* which to be anonymized. Now  $\alpha$  values are replaced by the average of each feature.
4. In the last step, we replace the average value against each value of *KG* in continuous features while the average value is used to find more generalized categorical value for the categorical converted numerical features. The input test data now changes from  $TS_s$  to it anonymized counterpart  $TS_{sa}$ .

The pseudo-code for this description is in the following Algorithm 3.

---

**Algorithm 3.** (*K*-anonymity for *k*-NN Mapper)

---

**INPUT:** *K* group size as *KG*, *QIDs* attribute as  $\alpha$ , *k*-NN as *k* value, *TRs* and *TSs*.

**OUTPUT:** *key* as mapper identifier  $Mapper_{ID}$  and class  $W_c$  value as  $DC_j$

Initialize all variables  $Avg=0$ ,  $l=0$

```

1: Read  $test\_tp_\alpha$  as classifier from TSs
2: for each  $i \in K$  do
3:    $KG =$  averages of all  $K[l]$ .
4:   for each average do
5:      $Noise[l] = Avg - KG[l]$ ;
6:      $test\_tp_\alpha[i] = Noise[i]$ 
7:      $test\_tp_\alpha$  store value on  $TS_{s1}$ 
8:   end for each
9:   if ( $\alpha$  is greater than initialized value) then
10:    Goto Step 3 for every  $test\_tp_\alpha$  on TSs
11:   end if
12: end for each
13: Create  $TR_{sj}$  with the instances of split  $j$ .
14: for  $i=0$  to  $i < size(TS_{s1})$  do
15:   Compute k-NN ( $ST_s, i, TR_{sj}, k$ )
16:   for  $m = 0$  to  $m < k$  do
17:      $DC_j(i, m) = < W_c(NB_m), ST_s(NB_m) > i$ 
18:   end for
19: end for
20:  $key = Mapper_{ID}, W_c$ 
21: return ( $<key, DC_j >$ )

```

---

In our study, we observe the effect of an anonymisation in two different aspects: tuple-based vs feature-based *generalization*. We first examine the effect of anonymization by its usual tuple-based (i.e., making the number of records same), secondly, we examine the effect of anonymization by its feature-based (i.e., making the number of features across records same). For the former, we analyze 4 different tuple-based degree  $K = \{5, 10, 100, 1000\}$  for example  $K = \{5\}$  indicates that there will be 5 records made same where  $K = \{10\}$  indicates that there will be 10 records made same and so on. *Simple* represent no anonymisation and transformation is applied on data. For the latter, we analyze 5 different feature-based degree  $Ax = \{2, 4, 8, 12, all\}$  where  $x$  indicates the number of *QIDs*. From the feature-based generalization, A2 represents  $\alpha = 2$ , i.e., age and workclass are used as *QIDs* and generalized whereas A4 represents

$\alpha = 4$ , i.e., the four features age, work-class, fnlwgt, and education are used as  $QIDs$ . A8 and A12 represents in the similar fashion. We use the special notation AA to mean all features are used as  $QIDs$  and generalized accordingly. Table 3 represents the snippet of data anonymization on  $K$ -5 degree on different numbers of  $QIDs$  being generalized.

**Table 3.** The sample of  $K$ -5 tuple with different number of column generalisation

Age	Work-class**	Fnlwgt	Edu	Edu-num	Marital-status	Occupation	Relation-ship	Race	Sex	Capital-gain	Capital-loss	Hours-per-week	Native-country	Income
<b>5 – anonymity with on 2 Features age and workclass</b>														
55.2	2.6	142624	8	12	3	7	1	5	1	5556	0	40	35	>50 K
55.2	2.6	160631	12	9	3	7	1	5	1	4508	0	8	35	≤50 K
55.2	2.6	153064	5	3	3	4	1	5	1	7688	0	10	35	>50 K
55.2	2.6	179479	12	9	7	4	2	5	2	3325	0	40	35	≤50 K
55.2	2.6	223206	11	16	3	9	1	2	1	15024	0	40	34	>50 K
<b>5 – anonymity with on 4 Features age and workclass, fnlwgt and education</b>														
55.2	2.6	171800.8	9.6	12	3	7	1	5	1	5556	0	40	35	>50 K
55.2	2.6	171800.8	9.6	9	3	7	1	5	1	4508	0	8	35	≤50 K
55.2	2.6	171800.8	9.6	3	3	4	1	5	1	7688	0	10	35	>50 K
55.2	2.6	171800.8	9.6	9	7	4	2	5	2	3325	0	40	35	≤50 K
55.2	2.6	171800.8	9.6	16	3	9	1	2	1	15024	0	40	34	>50 K
<b>5 – anonymity with on all Features</b>														
55.2	2.6	171800.8	9.6	9.8	3.8	6.2	1.2	4.4	1.2	0	7220.2	27.6	34.8	>50 K
55.2	2.6	171800.8	9.6	9.8	3.8	6.2	1.2	4.4	1.2	0	7220.2	27.6	34.8	≤50 K
55.2	2.6	171800.8	9.6	9.8	3.8	6.2	1.2	4.4	1.2	0	7220.2	27.6	34.8	>50 K
55.2	2.6	171800.8	9.6	9.8	3.8	6.2	1.2	4.4	1.2	0	7220.2	27.6	34.8	≤50 K
55.2	2.6	171800.8	9.6	9.8	3.8	6.2	1.2	4.4	1.2	0	7220.2	27.6	34.8	>50 K

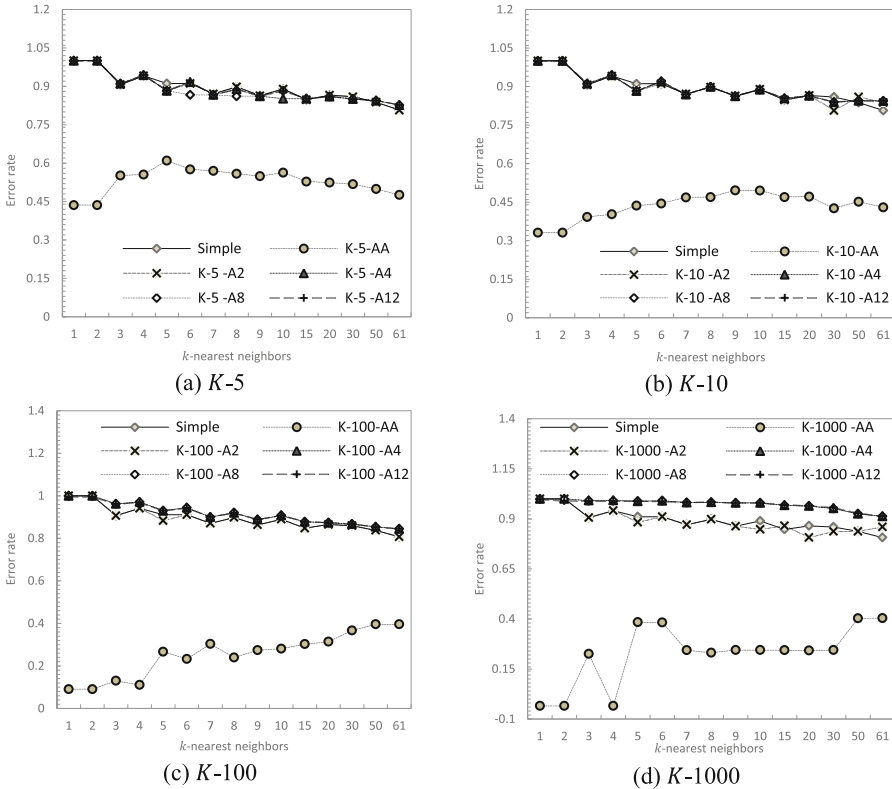
## 5 Experiments and Results

A set of experiments have been conducted on the Adult dataset to observe the  $k$ -NN based classification errors on data anonymized using  $k$ -anonymity. The experiment was performed on the single node cluster with the following specification: (1) the CPU model: Intel(R) Xeon(R) CPU E5-1650 v3, (2) the processing speed: @ 3.50 GHz, (3) the number of core processors: 6, (4) the storage capacity: 4 Tera bytes, and (5) the memory size 32 GB of RAM.

We first run the experiment on both the training and test datasets on  $k$ -NN classifier without any anonymization then check the classification error.

### 5.1 Applying $K$ -Anonymity on $k$ -NN Classifier

Figure 1 illustrates the result on the classification error studying from the feature-based anonymization. For example Fig. 1a shows the results of 5 records anonymized on 2  $QIDs = \{\text{age, workclass}\}$  which is denoted as  $K$ -5-A2 while the results of 5 records anonymized on 4  $QIDs = \{\text{age, workclass, fnlwgt, edu}\}$  is denoted as  $K$ -5-A4. The same notation is used for other number of  $QIDs$ .



**Fig. 1.** K-anonymity on varying degrees of anonymized feature sets

Here is the summary of our observations;

- The number of features being anonymized attributes to the decreasing accuracy (i.e., increasing classification error) as we see this in all graphs.
- As the number of k-nearest neighbours increases, more classification errors are generated. This is due to the increasing size of the sample being the subject of the classification, that is, there is increasing probability of producing an error as there are more data.
- There is a huge amount of classification errors when all features are anonymized in comparison to when there are at least a few features still not anonymized.
- The distribution of the data within a feature affects on the number of classification errors. If the distribution of the data is wide and if they are generalized, they tend to subject to more classification errors.
- With the increasing number of K degrees, the fluctuation of classification errors becomes unpredictable. For example, with K-5 and K-10, we observe a steady increasing or decreasing of classification errors in a smaller range scale which was between 45%–60% with K-5 while 33%–45% with K-10. In the meantime, the

classification errors were sharply increased from 10% to 40% in K-100 while it was between 0%–45% with K-1000.

Figure 2 illustrates the result on the classification error studying from the tuple-based– anonymization. For example, the Fig. 2a shows the results of 2  $QID_s = \{age, work - class\}$  anonymized with different degree of  $K = \{5, 10, 100, 1000\}$ .

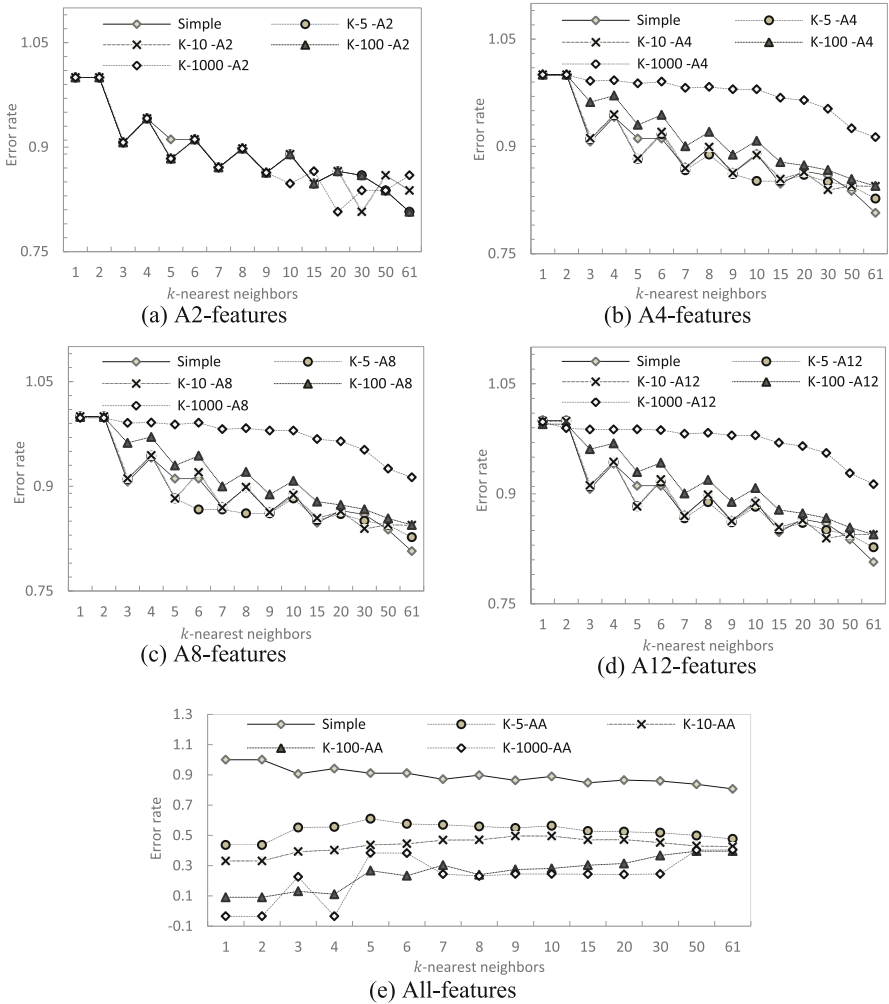


Fig. 2.  $QID_s$  on varying degrees of  $K$ -anonymity

Here is the summary of our observations;

- There is more classification errors produced as the degree of  $K$  increases. It is easy to understand this pattern because simply more data means the increasing possibility

with classification errors. This is observed in all graphs, irrelevant to the number of *QIDs* involved in the anonymization process.

- When only two *QIDs* were anonymized, as shown in Fig. 2a, the effect of increasing *K* degree is negligent. As the number of *QIDs* increased to be anonymized, the scale of classification error range becomes wider. For example, in Fig. 2a where it is only two *QIDs* anonymized, there is almost no difference in classification errors among *K*-degrees. However, in Fig. 2e where all *QIDs* were anonymized, *K*-5 classification errors stay around 50%, *K*-10 classification errors stay around 30% whereas *K*-100 stays around 10%.
- The utility of anonymized data is higher with a fewer *QIDs* regardless *K* degree as we do not see much difference in the classification errors between non-anonymized data and anonymized data.

## 6 Conclusions and Future Work

This research work is an extension from our previous work [4] where we focus running a classification algorithm on the anonymized dataset running a MapReduce platform. In this research we used k-NN as a classifier on anonymised data. We used the measurement of classification errors to observe the effects between privacy verses utility trade-offs when different sets of data were anonymized using multiple privacy parameters of *K*-anonymity. We used two different approaches; anonymizing the data based on (1) tuples and (2) features. As expected, the number of k-nearest neighbour has the close relationship with classification errors introduced. More data in a dataset produced higher probability of classification errors. We also observed that the distribution of the data within a feature for given dataset affects quite significantly on classification error.

In our future work, we plan to run our experiments in multiple node cluster which may need modification in the algorithms we used in this study. We also plan to make more close observations on the classification errors on different parameters on *K*-anonymity and differential privacy such as finding the most optimal point for privacy and utility trade off.

## References

1. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **13**, 21–27 (1967)
2. Sweeney, L.: *K*-anonymity: a model for protecting privacy 1. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* **10**, 557–570 (2002)
3. Zhang, X., Yang, L.T., Liu, C., Chen, J.: A scalable two-phase top-down specialization approach for data anonymization using mapreduce on cloud. *IEEE Trans. Parallel Distrib. Syst.* **25**, 363–373 (2014)
4. Bazai, S.U., Jang-Jaccard, J., Zhang, X.: A privacy preserving platform for MapReduce. In: Batten, L., Kim, D.S., Zhang, X., Li, G. (eds.) *ATIS 2017. CCIS*, vol. 719, pp. 88–99. Springer, Singapore (2017). [https://doi.org/10.1007/978-981-10-5421-1\\_8](https://doi.org/10.1007/978-981-10-5421-1_8)

5. Zhang, X., Dou, W., Pei, J., Nepal, S., Yang, C., Liu, C., Chen, J.: Proximity-aware local-recoding anonymization with MapReduce for scalable big data privacy preservation in cloud. *IEEE Trans. Comput.* **64**, 2293–2307 (2015)
6. Stupar, A., Michel, S., Schenkel, R.: RankReduce - processing K-nearest neighbor queries on top of mapreduce. In: *CEUR Workshop Proceedings*. vol. 630, pp. 13–18 (2010)
7. Zhang, C., Li, F., Jestes, J.: Efficient parallel k NN joins for large data in MapReduce. In: *Proceedings of the 15th International Conference on Extending Database Technology - EDBT 2012*, p. 38 (2012)
8. Inan, A., Kantarcioglu, M., Bertino, E.: Using anonymized data for classification. In: *Proceedings - International Conference on Data Engineering*, pp. 429–440 (2009)
9. Baryalai, M., Jang-Jaccard, J., Liu, D.: Towards privacy-preserving classification in neural networks. In: *2016 14th Annual Conference on Privacy, Security and Trust, PST 2016*, pp. 392–399 (2016)
10. Xia, D., Li, H., Wang, B., Li, Y., Zhang, Z.: A map reduce-based nearest neighbor approach for big-data-driven traffic flow prediction. *IEEE Access.* **4**, 2920–2934 (2016)
11. Zhou, L., Wang, H., Wang, W.: Parallel implementation of classification algorithms based on cloud computing environment. *TELKOMNIKA Indones. J. Elect. Eng.* **10**, 1087–1092 (2012)
12. Roy, I., Setty, S.T.V., Kilzer, A., Shmatikov, V., Witchel, E.: Airavat: security and privacy for MapReduce. In: *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, p. 20 (2010)
13. Frank, A., Asuncion, A.: UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science, Irvine, CA. 2008, (2010)
14. Inan, A., Kantarcioglu, M., Ghinita, G., Bertino, E.: Private record matching using differential privacy. In: *Proceedings of the 13th International Conference on Extending Database Technology - EDBT 2010*, p. 123 (2010)
15. Maillo, J., Triguero, I., Herrera, F.: A MapReduce-based k-Nearest neighbor approach for big data classification. In: *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom*. vol. 2, pp. 167–172 (2015)