K-means Bayes algorithm for imbalanced fault classification and big data application

Gecheng Chen, Yue Liu, Zhiqiang Ge*

State Key Laboratory of Industrial Control Technology, Institute of Industrial Process Control, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, Zhejiang, P. R. China

Abstract

Fault classification is an important part for process monitoring and control in industrial processes. Most traditional fault classification methods are under the assumption that the amount of data in different classes are similar. In practice, however, large amounts of normal data samples (majority) and only a few fault data samples (minority) are the common cases in most industrial processes. In other words, fault classification can be seen as an imbalanced data classification problem, although it has not been considered in this area to date. In this paper, a K-means Bayes algorithm is proposed to deal with the imbalanced fault classification problem, upon which a MapReduce approach is further introduced to implement the proposed method for fault classification in the big data case. Effectiveness of the proposed method is verified through the Tennessee Eastman (TE) benchmark process.

Keywords: Imbalanced data; K-means; Naive Bayes; MapReduce; Fault Classification.

* Corresponding author: +86-571-87951442 , E-mail address: gezhiqiang@zju.edu.cn (Ge Z.)

1. Introduction

As industrial systems become increasingly complex and integrated, process monitoring becomes more and more important to keep process safety and improve product quality. With the development of modern information technologies, an exceptionally large amount of process data has been produced and stored for both offline analyses and online utilizations. As a result, data-driven approaches are developing rapidly and become more and more popular in industrial applications [1-5]. Compared with the traditional first-principle based methods, data-based methods are more flexible and easy to implement and thus can be used in more complicated industrial processes. By mining and analyzing the data of industrial processes, a plenty of missions could be handled, such as process monitoring, soft sensing, modes clustering and fault classification.

Fault classification is a well-studied technique in data mining and machine learning domains. An effective classifier can make class forecasts on new samples and give them reliable labels for prediction or decision. In the past years, a number of data-driven methods have been developed and applied for fault classification in industrial processes [6-13]: Leo applied Fisher discriminant analysis for fault diagnosis in 2004 which seeks a transformation matrix to maximize the between-class scatter and minimize the within-class scatter; Jing studied SVM and PCA for fault classification in 2015. He found that SVM is a powerful tool for multi-classification problems and the PCA also shows good results with less computational efforts; Liu proposed a hierarchical clustering based weighted random forests for fault classification to improve the diversity between classification trees to get a better classifier; Zhang evaluated the key performance indicator of different methods for multivariate statistical process monitoring.

However, most existing fault classification methods work well under the assumption that the amount of samples in different classes are not too much different, which might not perform very well on imbalanced fault data. Imbalanced data, which means the size of one or more classes are much larger than the other

classes. Actually, this is the exact case that exists in modern industrial processes. Typically, there are many normal data in the process which can be regarded as the majority class and the number of fault data that can be considered as the minority class is always limited. Meanwhile, the amount of various fault data may be quite different from each other, making the fault classification problem even more complicated. If the imbalanced feature of process data is not considered for fault classification, the performance of classifier may be seriously influenced. To the best of our knowledge, however, this problem has not been considered in the process monitoring or fault diagnosis area to date.

In recent years, the imbalanced classification problem has drawn much attention. Most research work are focused on two main problems: absolute scarcity of data and relative scarcity of data. For the absolute scarcity of data, which means the excessive lack of minority data, it is unable to learn a clear classification boundary. As a result, the minority classification accuracy will be quite poor. Weiss [14-15] shows that when absolute scarcity of data happens, the classification error was much higher than the general situation, and the samples of the minority can be easily removed as noise during the training process. To this end, a SMOTE (Synthetic Minority Oversampling Technique) based over-sampling method was proposed by Wallace [16], which increases the number of minority samples by SMOTE to eliminate the degree of imbalance and the absolute scarcity of data. For the relative scarcity of data, which means the number of the minority samples is relatively less than the majority, the information of the minority will be overwhelmed by the majority during the training process. Most classifiers work under the assumption of equal classes, which may cause a bad performance on imbalanced data. As Lemnaru pointed out in [17] that one of the premise of Naive Bayes classifier is "equalization of various types of sample data", which will be broken when dealing with imbalanced data problems. Japkowicz indicated in [18] that when using the Decision Tree to deal with the imbalanced data, the majority will inevitably dominate the training process, resulting in a poor recognition

rate of the minority classes. To deal with this problem, Liu proposed an under-sampling-bagging algorithm in [19], which randomly selected a subset from the majority with the same size of the minority, and then trained a sub-classifier by using this subset and the minority. After that, these sub-classifiers were combined to generate the final classification result.

Although some effective methods have been proposed for handling the imbalanced data problem, there are also many disadvantages that need further investigations. For example, the SMOTE method can only add samples within the boundary described by the given minority class, which cannot jump out of the original range of the minority. The under-sampling method is obvious to lose part information of the majority and thus cannot effectively figure out the absolute scarcity problem. To address these problems simultaneously, a K-means Bayes algorithm is proposed in this paper, particularly for the purpose of imbalanced fault classification in industrial processes. The majority dataset is divided into N sub-classes according to the degree of the imbalance by using the K-means algorithm. After that, these sub-classes are combined with M minority classes to form a (M+N) classes training dataset. Then, a Naive Bayes classifier can be trained by using the new training dataset. Finally, through mapping the fault classification result achieved by the Naive Bayes classifier back to the original majority and minority classes, the actual results can be obtained. Compared to the existing under-sampling and oversampling methods, the proposed method neither adds samples nor reduces samples, which ensures the quality of the training data for the model development, and can help the minority classes to describe the boundary more accurately. Moreover, in order to handle the imbalance fault classification problem in the era of big data, a MapReduce form of the proposed K-means Bayes algorithm is further developed.

The main contribution of this article are as follow: First, a “clustering based classification framework” is proposed and an implementation of the framework, the K-means Bayes, is put forward and tested based on

the TE benchmark process; second, the effectiveness of the framework for imbalanced classification problems was evaluated; Besides, the K-means Bayes method was extended to the MapReduce framework to deal with imbalanced problems for the case of big data.

The rest of this paper is organized as follows. Section 2 introduces the Naive Bayes method for fault classification and proposed a K-means Bayes algorithm for imbalanced fault classification. Section 3 introduces the MapReduce implementation of the K-means Bayes method for imbalanced fault classification in the case of big data. The performances of the proposed methods are evaluated through the Tennessee Eastman (TE) benchmark process in Section 4. Finally, conclusions are made.

2. K-means Bayes algorithm for imbalanced fault classification

2.1 Naive Bayes algorithm for fault classification

Given a training set $\mathbf{W}_l = [\mathbf{X}_1; \mathbf{X}_2; ...; \mathbf{X}_{C+1}]$, which contains C fault modes and one normal mode. $\mathbf{X}_i = [\mathbf{x}_1; \mathbf{x}_2; ...; \mathbf{x}_{n_i}], i = 1, 2, ..., C+1$ in the training set represents the data matrix of the i th class, where n_i is the number of samples in the i th class. The labels of the normal mode and C fault modes are from 1 to $C+1$ in sequence. For a test sample $\mathbf{x}$, the posterior probability that it belongs to class c can be calculated as follows:

$$P(c \mid \mathbf{x}) = \frac{P(c)P(\mathbf{x} \mid c)}{P(\mathbf{x})} \tag{1}$$

where $P(c)$ is the priori probability of class, $P(\mathbf{x} \mid c)$ is the conditional probability of the test sample $\mathbf{x}$ belonging to class c, $P(\mathbf{x})$ is the normalization factor. Under the assumption that the variables of an industrial data sample are independent, Eq. (1) can be computed as:

$$P(c \mid \mathbf{x}) = \frac{P(c)\, P(\mathbf{x} \mid c)}{P(\mathbf{x})} = \frac{P(c)}{P(\mathbf{x})} \prod_{i=1}^{m} P(a_i \mid c) \tag{2}$$

where a_i refers to the value of the i th variable of test sample $\mathbf{x}$, $P(a_i \mid c)$ is the conditional probability

of the i th variable.

By assuming each continuous variable a_i obeys a Gaussian distribution $P(a_i \mid c) \sim N(\mu_{c,i}, \sigma_{c,i}^2)$, where $\mu_{c,i}$ and $\sigma_{c,i}^2$ refer to the mean and variance of the i th variable, $P(a_i \mid c)$ can be calculated as:

$$P(a_i \mid c) = \frac{1}{\sqrt{2\pi}\sigma_{c,i}} \exp\left(-\frac{(a_i - \mu_{c,i})^2}{2\sigma_{c,i}^2}\right) \tag{3}$$

Finally, the Naive Bayes fault classification result can be achieved as follows:

$$h(x) = \arg\max_{c \in Y} P(c) \prod_{i=1}^{m} P(a_i \mid c) \tag{4}$$

2.2 K-means Bayes algorithm for imbalanced fault classification

In order to eliminate the impact of imbalance data on fault classification, without changing the information of original dataset, a K-means Bayes algorithm is proposed for the purpose of imbalanced fault classification. For the training set $\mathbf{W}_l$, the normal class $\mathbf{X}_1$ is assumed to be the majority, and the other fault classes are regarded as the minority. It is assumed that the sizes of the minority classes are not too much different. Then the degree of imbalance n is defined as the ratio of majority to minority, denoted as follows:

$$n = \frac{n_1}{n_2} \approx \frac{n_1}{n_3} \approx \ldots \approx \frac{n_1}{n_{C+1}} \tag{5}$$

To illustrate the K-means Bayes method, a binary classification problem is presented as an example, which is demonstrated in Figure 1. As shown in Figure 1(a), the circles represent the majority and the triangles represent the minority. Firstly, cluster the majority $\mathbf{X}_1$ into k subclasses by K-means as $\mathbf{X}_1 = [\mathbf{U}_1; \mathbf{U}_2; \ldots; \mathbf{U}_k]$. However, the clustering result obtained by the original K-means algorithm may introduce new imbalance, in other words, the size of one sub-class may be much larger than the other sub-classes. To deal with this problem, a threshold T is introduced to ensure the numbers of data in subclasses are not much different. The result of clustering is shown in Figure 1(b), in which the dotted lines are the boundaries of subclasses of the majority. The detailed procedures of the T-threshold K-means algorithm are listed in Table 1.

Table1: T-threshold K-means algorithm

Algorithm: T-threshold K-means algorithm

Input: majority training set $\mathbf{X}_1 = [\mathbf{x}_1; \mathbf{x}_2; ...; \mathbf{x}_{n_1}]$, where $\mathbf{x}_j \in R^m, j = 1, 2, \cdots, n_1$, k, T

Process:

Select k samples randomly from $\mathbf{X}_1$ as initial mean vector $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k\}$;

Repeat

Make $\mathbf{U}_i = \varnothing (i = 1, 2, ..., k)$

For $j = 1, 2, ..., n_1$ **do**

Calculate the distance between each sample $\mathbf{x}_j$ in $\mathbf{X}_1$ and each mean vector $\boldsymbol{\mu}_i, i = 1, 2, ..., k$: $d_{ji} = \| \mathbf{x}_j - \boldsymbol{\mu}_i \|$;

Sort the distance set in ascending order $D_j = [d_{ji}], i = 1, 2, ..., k$;

For $g = 1, 2, ..., k$ **do**

Assume $D_j[g] = d_{jt}$, where t belongs to $[1, 2, ..., k]$;

If $|\mathbf{U}_t| == T$: continue;

If $|\mathbf{U}_t| < T$: $\mathbf{U}_t = \mathbf{U}_t \cup \{\mathbf{x}_j\}$, **break**;

End for

For $i = 1, 2, ..., k$ **do**

Calculate new mean vector: $\boldsymbol{\mu}_i^{'} = \frac{1}{|\mathbf{U}_i|} \sum_{\mathbf{x} \in \mathbf{U}_i} \mathbf{x}$;

If $\boldsymbol{\mu}_i^{'} \neq \boldsymbol{\mu}_i$

Let $\boldsymbol{\mu}_i = \boldsymbol{\mu}_i^{'}$

End for

End for

until the current mean vectors do not change

Update the label of the samples, **do** $y_j = i$ **if** $\mathbf{x}_j \in \mathbf{U}_i$, where y_j is the label of $\mathbf{x}_j$

Output: $\mathbf{U}_{major} = \{\mathbf{U}_1, \mathbf{U}_2, ..., \mathbf{U}_k\}$, $\mathbf{Y}_{major} = \{\mathbf{Y}_1; \mathbf{Y}_2; ...; \mathbf{Y}_k\}$.

After that, a Naive Bayes classifier is constructed. The training set is updated to $\mathbf{W}_l^{'} = [\mathbf{U}_1; \mathbf{U}_2; ...; \mathbf{U}_k; \mathbf{X}_2; ...; \mathbf{X}_{C+1}]$, in which the original majority class is replaced with its sub-classes and the labels of these subclasses are set from 1 to k. As a result, the labels of the minority classes are updated to $[k+1, k+2, \cdots, C+k]$. Calculate the mean, variance, and prior probability of each class, according to the following equation.

$$Mean_{pq} = \frac{1}{n_p} \sum_{t_p=1}^{n_p} a_{tq} \tag{6}$$

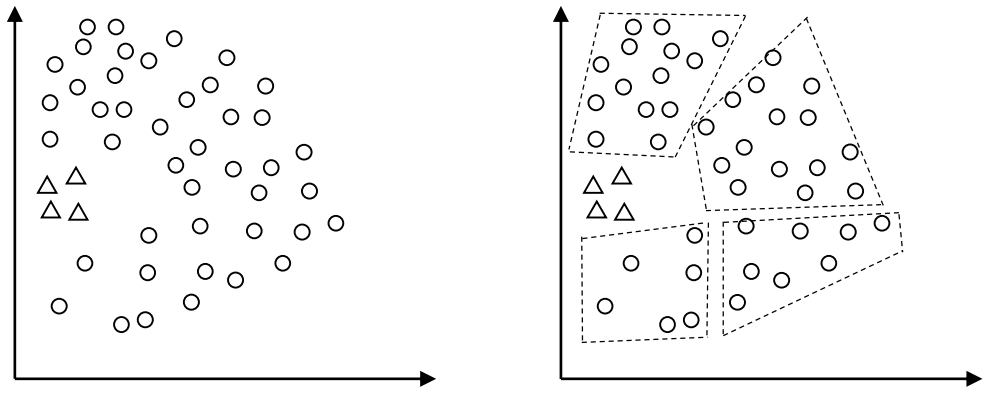
$$Var_{pq} = \frac{1}{n_p} \sqrt{\sum_{t=1}^{n_p} (a_{t_p q} - Mean_{pq})^2} \tag{7}$$

$$P_p = \frac{n_p}{\sum_{t=1}^{C+N} n_{t_p}} \tag{8}$$

where $p = 1, 2, ..., C+k$ refers to the number of class, $q = 1, 2, ..., \mathrm{m}$ refers to the number of dimension of the sample and $t_p = 1, 2, ..., n_p$ refers to the number of sample of the p th class.

For a test sample $\mathbf{x}_o$, the classification result y_o can be calculated by using Eq. (4), and the real label y_r of the test sample can be achieved as:

$$y_r = \begin{cases} 1, & y_o \in [1, k] \\ y_o - k + 1, & y_o \in [k+1, C+k] \end{cases} \tag{9}$$



(a) (b)

Figure1: K-means Bayes, (a) a binary imbalanced classification problem; (b) a binary classification problem is transferred to a multi-classification problem after clustering.

2.3 Performance analyses and discussions

For the example given above, some more assumptions are made: stars represent the center points of classes (or subclasses), diamonds are test samples of the minority, the red thick dotted line is the boundary that can be described by the minority and the black thick dotted line is the real boundary. Obviously, the problems of relative scarcity and absolute scarcity both exist in this example. When testing those 6 samples, three samples closer to the center of the minority are more likely to be accurately classified and the other three samples near the center of the majority are more likely to be wrongly classified.

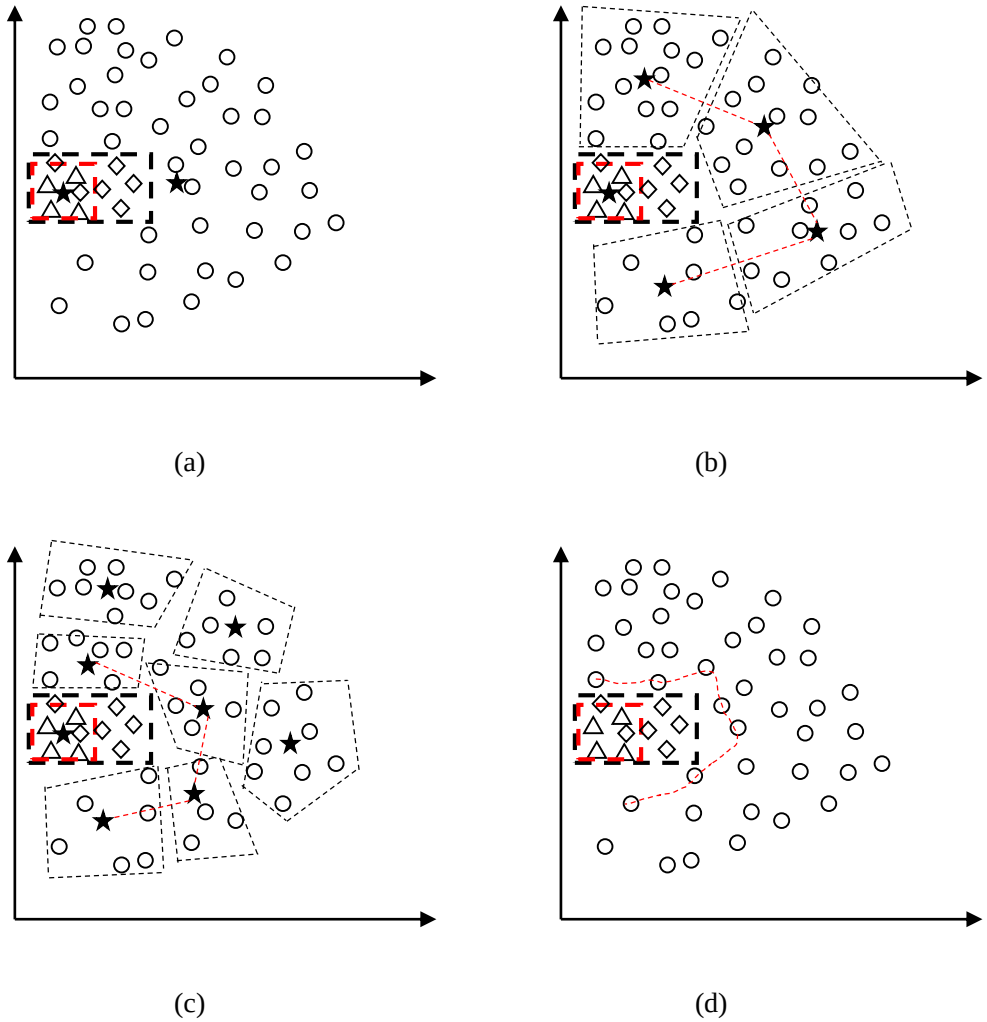


Figure 2: (a) a binary imbalanced classification problem with relative and absolute scarcity of data; (b) a binary classification problem is transferred to a multi-classification problem; (c) the line between the centers of the subclasses of majority get closer to the real boundary after clustering the majority into more subclasses; (d) the line between the centers of the subclasses of majority will overlap the real boundary approximately if the majority is clustered into infinite subclasses.

For the relative scarcity, the K-means Bayes classification method divides the majority into several sub-classes to eliminate the impact of data amount. Compared to the existing under-sampling and oversampling methods, this method neither adds samples nor reduces samples, which can effectively prevent the negative effects such as overfitting and information lose. For the absolute scarcity, it is a greater challenge for the current algorithms. For example, the SMOTE method can only add samples within the boundary described by a given minority class. That is, samples can only be added in the red thick dotted line in Figure 2(a), which is actually not helpful for learning the real boundary. However, the boundary of the minority class can be

described with the help of the majority class by using the proposed K-means Bayes method. As the number of sub-classes increases, different results are shown in Figure 2(b)-(c), in which the red thin dotted lines are the centerlines of the subclasses closest to the minority. The center of each class can represent the information of the whole class to a certain extent. It can be seen that the red thin dotted lines are getting closer to the actual boundary. Ideally, when the majority class is clustered into infinite sub-classes like Figure 2(d) shows, the red dotted line connects the samples in the majority closest to the boundary, which can be considered as the actual boundary.

3. MapReduce implementation of the K-means Bayes algorithm

With the rapid increase of industrial process data, how to deal with industrial big data has attracted a lot of attention in recent years [20-22]. Due to the large scale and high dimension of the data, the industrial big data has put forward many new requirements for the traditional data-driven industrial process monitoring algorithms, such as appropriate time and space complexity. To realize a parallel computing based classification algorithm, the existing classification algorithm can be combined with the MapReduce framework to deal with the classification problems in the case of big data [23]. In this section, the MapReduce framework is introduced into the K-means Bayes algorithm for the purpose of imbalanced fault classification.

3.1 Overview of MapReduce framework

The MapReduce parallel computing model was firstly proposed by Google to deal with large-scale data processing problems, which has caught much attention in recent years [24-26]. The working nodes in MapReduce framework contain one masternode (called master) and some slavernodes (called slaver). When doing a parallel computing work, the user uses the masternode to provide management and control services,

and the slavernodes provide block storage and computing services. Meanwhile the slavernodes provide two classes of working condition including mapping and reducing. The mapping slavernodes are designed to take simple processing for raw data and the reducing are designed to take further calculation for the data output from the mapping slavernodes. Besides, the Google File System (GFS) is designed to provide efficient and reliable distributed data storage service for the computing process.

The Apache Hadoop is the most widely used platform for the realization of the MapReduce framework. Figure 3 shows the general process of implementing MapReduce by using Hadoop. The master is used to manage the whole process and assign Mapping and Reducing functions for the slavers, and the slavers work under the command of the master. Meanwhile, Hadoop Distributed File System (HDFS) is designed to provide efficient and reliable distribute storage services working on the commodity hardware. The whole calculation process is as follows. Firstly, the master assigns the mapping function and reducing function for the slavers. Then, the mappers, which mean the slavers assigned by mapping function, read data from the HDFS and take the mapping function on the data to output some (key, value) pairs. Next, all (key, value) pairs are combined by their keys under the shuffling function. Then, the reducers, which mean the slavers assigned by reducing function, read the combined (key, list(value)) pairs and perform the reducing function to get some results. In the last step, these results are written to the HDFS. As we can see, designing the mapping function and the reducing function is the key to the whole process.

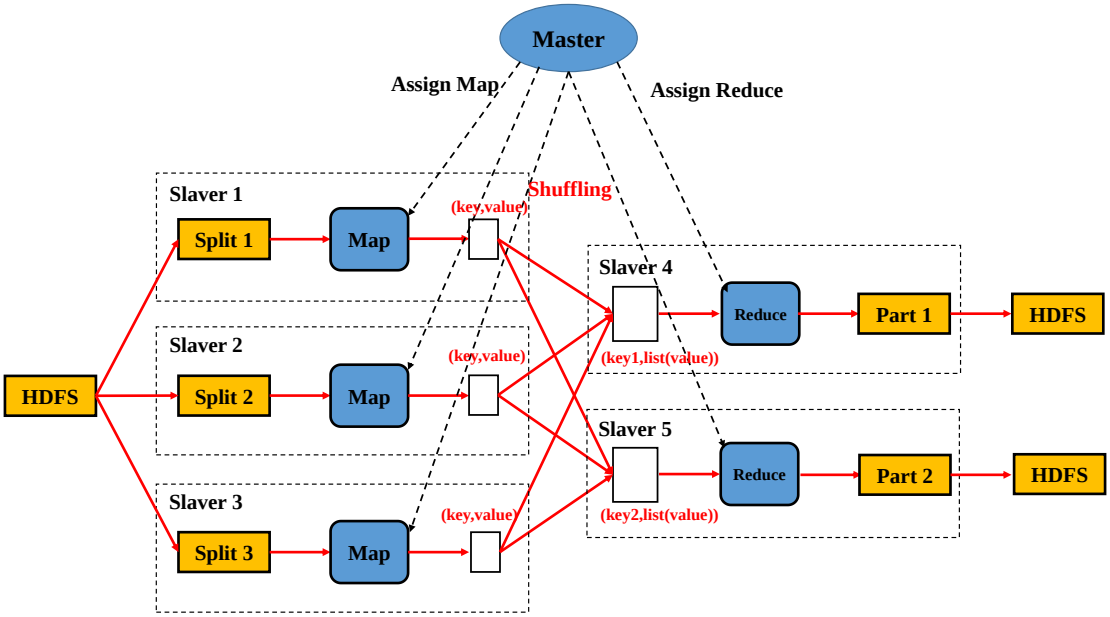


Figure 3: the general process of implementing MapReduce using Hadoop

3.2 MapReduce implementation of the K-means Bayes algorithm

In order to solve the imbalanced fault classification problem in the case of big data, the K-means Bayes algorithm is extended to the MapReduce form in this section.

Firstly, the majority is clustered into some classes. K samples are randomly selected to compose the initial center point set $\mathbf{Q}$, which will be stored in the HDFS file system. The Mapper function is used to determine the cluster that the samples belong to by calculating the distance between each sample and each center point. The detailed steps are listed in Table 2.

Table 2: T-threshold K-means Mapper function

Algorithm: T-threshold K-means Mapper function
Input: majority training set $\mathbf{X}_1=[\mathbf{x}_1;\mathbf{x}_2;...;\mathbf{x}_{n_1}]$, center point set $\mathbf{Q}=\{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,...,\boldsymbol{\mu}_k\}$
Process:
Read the samples line by line from HDFS;
Calculate the distance between each sample $\mathbf{x}_j, j=1,2,\cdots,n_1$ in $\mathbf{X}_1$ and each mean vector $\boldsymbol{\mu}_i, i=1,2,\cdots,k$: $d_{ji}=\|\mathbf{x}_j-\boldsymbol{\mu}_i\|$;
Determine the cluster label of $\mathbf{x}_j$ by choosing the cluster with the nearest distance: $\lambda_j=\arg\min_{i\in\{1,2,...,k\}} d_{ji}$;
Output: $<\lambda_j,\mathbf{x}_j>$, where $\lambda_i=1,2,\cdots,k$.

For all key-value pairs $<\lambda_j,\mathbf{x}_j>$ transmitted from the slave, the Reducer function is used to accumulate

the samples in λ_j cluster as $\mathbf{R}_{\lambda_j}$ and give the number of samples in λ_j cluster $count_{\lambda_j}$, and then use them to update the center point set. During the accumulation process, if $count_{\lambda_j}$ exceeds the cluster number threshold T, the accumulation will be stopped and the sample that has participated in this round of accumulation will be marked as η so that it will no longer participate the next iteration and the cluster label λ_j will not appear in the next iteration as well. The detailed steps are listed in Table 3. The main function needs to repeat the Mapper and Reducer until the center point set does not change, as shown in Table 4.

Table 3: K-means Reducer function

Algorithm: T-threshold K-means Reducer function

Input: $<\lambda_j, \mathbf{x}_j>$, where $\lambda_i = 1, 2, \cdots, k$, center point set $\mathbf{Q} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k\}$

Process:

For λ_j **in** $1, 2, ..., k$:

$\mathbf{R}_{\lambda_j} = [0, 0, ..., 0]_{1\times m}$; $count_{\lambda_j} = 0$; $\mathbf{U}_{\lambda_j} = \varnothing$;

For $<\lambda, x>$ where $\lambda == \lambda_j$:

$\mathbf{R}_{\lambda_j} += \mathbf{x}$, $count_{\lambda_j} += 1$, $\mathbf{U}_{\lambda_j} = \mathbf{U}_{\lambda_j} \cup \{\mathbf{x}\}$;

If $count_{\lambda_j} == T$

Mark the $\mathbf{x}$ in $\mathbf{U}_{\lambda_j}$ as η

Output: $<\boldsymbol{\mu}_{\lambda_j}, \mathbf{U}_{\lambda_j}>$

break;

Calculate the new mean vector for cluster λ_j : $\boldsymbol{\mu}'_{\lambda_j} = \dfrac{\mathbf{R}_{\lambda_j}}{count_{\lambda_j}}$;

End for

End for

Output: $<\boldsymbol{\mu}'_{\lambda_j}, \mathbf{U}_{\lambda_j}>$, new center set $\mathbf{Q}' = \{\boldsymbol{\mu}'_1, \boldsymbol{\mu}'_2, ..., \boldsymbol{\mu}'_k\}$ $<\boldsymbol{\mu}'_{\lambda_j}, \mathbf{U}_{\lambda_j}>$

Table 4: K-means main function

Algorithm: T-threshold K-means main function

Input: majority training set $\mathbf{X}_1 = [\mathbf{x}_1; \mathbf{x}_2; ...; \mathbf{x}_{n_1}]$, initial center point set $\mathbf{Q} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k\}$

Process:

Repeat

K-means Mapper function

K-means Reducer function

Until the current mean vectors do not change

Output: $\mathbf{U}_{major} = \{\mathbf{U}_1, \mathbf{U}_2, ..., \mathbf{U}_k\}$, $\mathbf{Y}_{major} = \{\mathbf{Y}_1; \mathbf{Y}_2; ...; \mathbf{Y}_k\}$

The whole process for T-threshold K-means based on the MapReduce framework is shown in Figure 4.

Firstly, the Master assigns the mapping function and reducing function for the nodes, and gives the initial center set ~~in~~ at random. Then, the mappers read samples and centers in HDFS and determine the cluster label λ_j of $\mathbf{x}_j$ by distance to output $<\lambda_j, \mathbf{x}_j>$. Next, the reducers update the centers based on $count_{\lambda_j}$ to output $<\boldsymbol{\mu}'_{\lambda_j}, \mathbf{U}_{\lambda_j}>$. The steps above are repeated until the centers of all classes do not change. Finally, a Naive Bayes classifier can be trained under the MapReduce framework for fault classification.

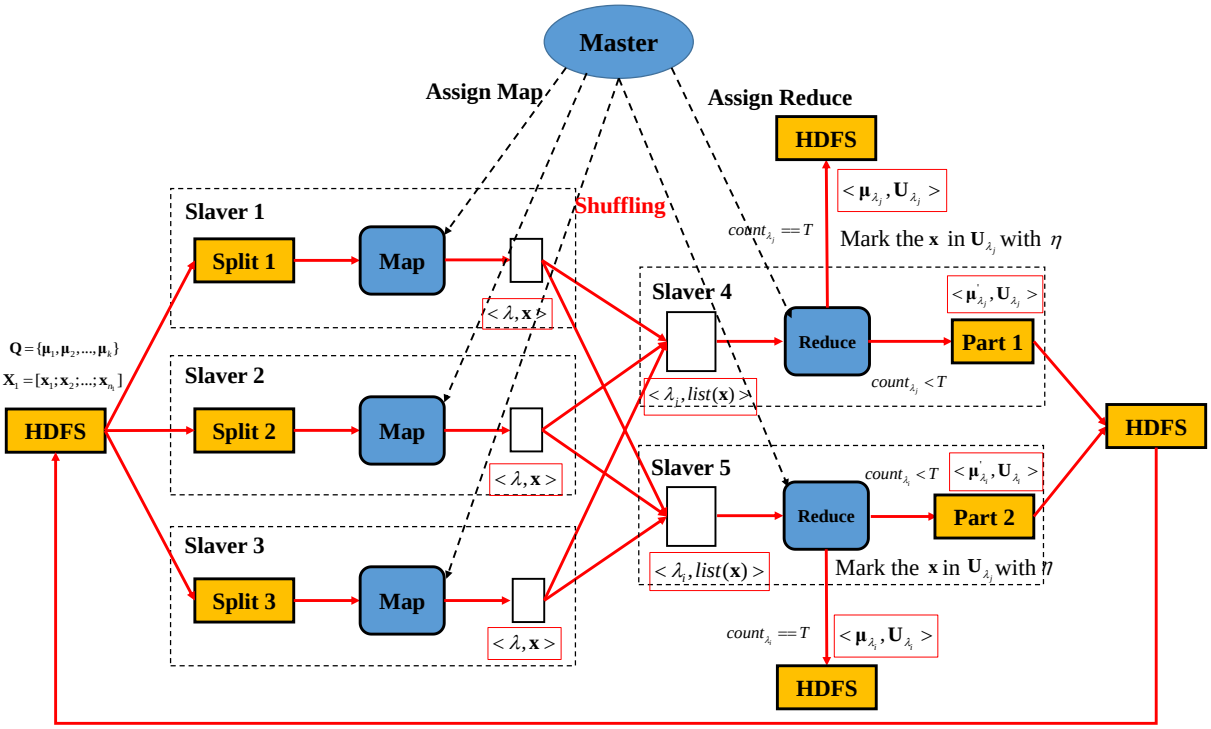


Figure 4: The process for T-threshold based on MapReduce framework

4. Case study

In this section, the Tennessee Eastman (TE) benchmark process, which was first introduced by Downs and Vogel [27], is applied for performance evaluation of the proposed methods proposed as shown in Figure 5, TE process consists of five operation units: a two-phase reactor, a product condenser, a recycle compressor, a product stripper and a separator. There are four gaseous reactants, A, C, D, E, and two liquid products, G and H, and one by-product, F. 21 faults are available for simulation in this process, detailed descriptions of

which are listed in Table 5. 41 measured variables and 12 control variables are involved in this process. All of the 41 measurement variables are selected for fault classification in this paper, which are listed in Table 6.

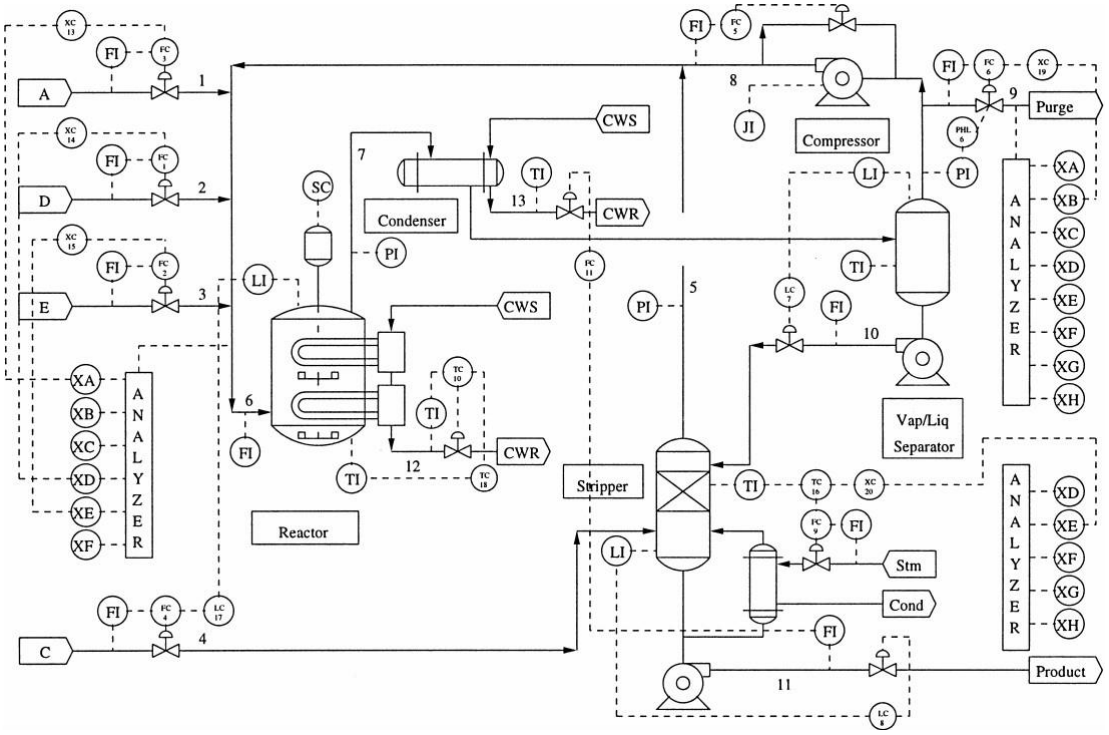


Figure 5: Flow chart of Tennessee Eastman Process

Table 5: 21 Faults in TE process

Fault	Descriptions	type	Fault	Descriptions	type
1	A/C feed ratio, B composition constant (stream 4)	Step	12	Condenser cooling water inlet temperature	Random
2	B composition, A/C ratio constant (stream 4)	Step	13	Reaction kinetics	Slow drift
3	D feed temperature (stream 2)	Step	14	Reactor cooling water valve	Sticking
4	Reactor cooling water inlet temperature	step	15	Condenser cooling water valve	Sticking
5	Condenser cooling water inlet temperature	Step	16	Unknown	Unknown
6	A feed loss (stream 1)	Step	17	Unknown	Unknown
7	C header pressure loss-reduced availability (stream 4)	Step	18	Unknown	Unknown
8	A, B, C feed composition (stream 4)	Random	19	Unknown	Unknown
9	D feed temperature (stream 2)	Random	20	Unknown	Unknown
10	C feed temperature (stream 4)	Random	21	Valve position constant (stream 4)	Constant
11	Reactor cooling water inlet temperature	Random			

Table 6: Measurement Variables

Variable No.	Measurement Variables	Variable No.	Measurement Variables
1	A feed rate	22	Condenser coolant temperature
2	D feed rate	23	Feed % A
3	E feed rate	24	Feed % B
4	A+C feed rate	25	Feed % C
5	Recycle flow rate	26	Feed % D
6	Reactor feed rate	27	Feed % E
7	Reactor pressure	28	Feed % F
8	Reactor level	29	Purge % A
9	Reactor temperature	30	Purge % B
10	Purge rate	31	Purge % C
11	Separator temperature	32	Purge % D
12	Separator level	33	Purge % E
13	Separator pressure	34	Purge % F
14	Separator underflow	35	Purge % G
15	Stripper level	36	Purge % H
16	Stripper pressure	37	Product % D
17	Stripper underflow	38	Product % E
18	Stripper temperature	39	Product % F
19	Stem flow rate	40	Product % G
20	Compressor work	41	Product % H
21	Reactor coolant temperature		

To verify the effectiveness of the K-means Bayes method for fault classification, the normal operating mode and faults 1, 2, 6 and 14 are selected in this case study. It can be seen that fault 1 and fault 2 are component changes in stream 4, fault 6 is caused by A feed loss in stream 1, and fault 14 is abnormal flow at the bottom of the product separator. The sample time is 3min, it is assumed that the numbers of majority and minority samples are 1000 and 10, respectively. The test dataset is composed of 1440 samples, in which 0~640 are normal samples, 641~840 belong to fault 1, 841~1040 belong to fault 2, 1041~1240 belong to fault 6 and 1241~1440 belong to fault 14. The fault classification results of four different methods are illustrated in Figure 6 and the confusion matrix is listed in Table 7.

Figure 6(a) shows the result of the Naive Bayes method, in which one can see that the accuracy of all faults is quite low under this degree of imbalance. Figure 6(b) shows the result of the SMOTE-Bayes method,

in which 20 samples are added into every minority class through the SMOTE method, and the Naive Bayes method is used to train the classifier. This over-sampling method eliminates the relative scarcity of data to some extent. However, because the SMOTE method cannot deal with the possible absolute scarcity of data, the recall of both fault 1 and fault 6 are still very low. What is more, the precision of the normal data of the SMOTE method in Table 7 is very low which indicates that the classifier still tries to classify the testing samples to the normal class as much as possible. In other words, this method is not a good solution to the imbalanced data classification problem.

Figure 6(c) shows the result of the under-sampling-bagging method, 50 samples are randomly selected from the majority and then combined with the minority to train a Naive Bayes classifier each time, through which a total of 10 sub-classifiers have been achieved. This method can improve the accuracy of the minority through eliminating the relative scarcity of data but at the same time it may sacrifice the accuracy of the majority which is also important in practice. Meanwhile, the recall of fault 1 is still very low which indicates that this method cannot deal with the possible absolute scarcity of data very well.

Figure 6(d) shows the result of the proposed K-means Bayes method, in which the majority is clustered into 50 subclasses according to the T-threshold K-means method. Then, the Naive Bayes method is used to train a classifier based on the training set containing 54 classes. The recall of all classes has been greatly improved which indicates that clustering before training the classifier can effectively eliminate the imbalanced problem. Particularly, this method possesses the greatest recall for the fault 1 which is 33% higher than the second one. The reason of low recall of the other three methods for the fault 1 is likely to be the problem of the absolute scarcity of data. Therefore, this method has a good performance on eliminating the absolute scarcity of data. Furthermore, this method has the highest precision for the normal class, indicating that this method is more unbiased than the other three methods toward the minority.

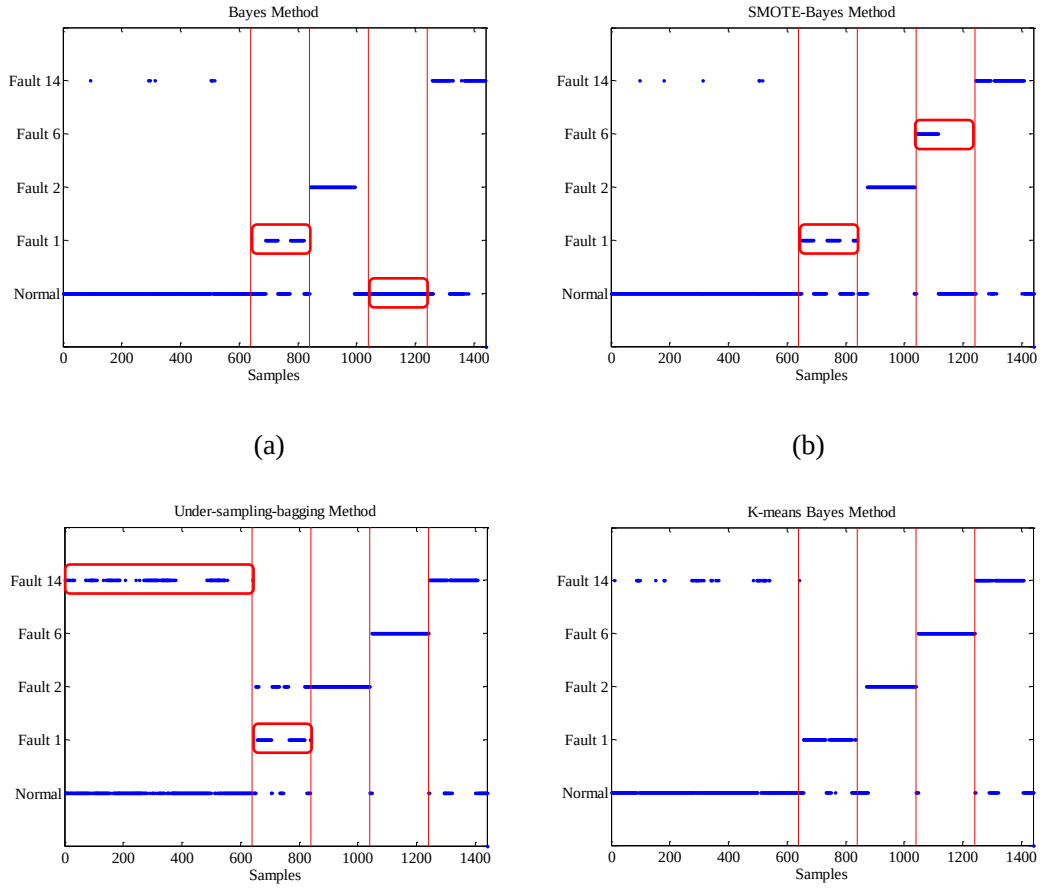


Figure 6: Simulation results of four methods, (a) the accuracy of all faults is quite low under this degree of imbalance using Na ïve Bayes; (b) SMOTE-Bayes performs not well on classifying samples belonging to fault 2 and fault 6; (c) under-sampling-bagging causes many false alarms; (d) K-means Bayes perform quite well for all classes.

Table 7: The confused matrices of four methods

Method	Reality	Prediction						
		Normal	Fault 1	Fault 2	Fault 6	Fault 14	Recall	Precision
Bayes	Normal	629	0	0	0	11	98.2%	58.5%
	Fault 1	131	69	0	0	0	34.5%	100%
	Fault 2	47	0	153	0	0	76.5%	100%
	Fault 6	200	0	0	0	0	0%	0%
	Fault 14	68	0	0	0	132	66.0%	92.3%
SMOTE-Bayes	Normal	619	0	0	0	21	96.7%	66.3%
	Fault 1	94	106	0	0	0	53.0%	100%
	Fault 2	39	0	161	0	0	80.5%	100%
	Fault 6	145	0	0	55	0	27.5%	100%
	Fault 14	36	0	0	0	164	82.0%	88.6%
Under-sampling-bagging	Normal	430	0	0	0	210	67.2%	81.1%
	Fault 1	49	84	67	0	0	42.0%	100%
	Fault 2	4	0	196	0	0	98.0%	74.5%

	Fault 6	2	0	0	198	0	99.0%	100%
	Fault 14	45	0	0	0	155	77.5%	42.5%
K-means Bayes	Normal	601	0	0	0	39	93.9%	87.9%
	Fault 1	28	172	0	0	0	86.0%	100%
	Fault 2	25	0	175	0	0	87.5%	100%
	Fault 6	4	0	0	196	0	98.0%	100%
	Fault 14	36	0	0	0	164	82.0%	80.8%

Joshi systematically studied several criteria including F1 and G-means in [28], which can be used as good evaluation criteria in this study. Figure 7 shows G-means for the four methods. Based on this figure, it can be seen that the blue line representing K-means Bayes is above the green line and black line which represent under-sampling-bagging method and Naïve Bayes method respectively. Though the SMOTE-Bayes has higher score for fault 14 than K-means Bayes, the scores of other four classes are much lower than the K-means Bayes method. On the whole, K-means Bayes gets the greatest performance according to the G-means criterion.

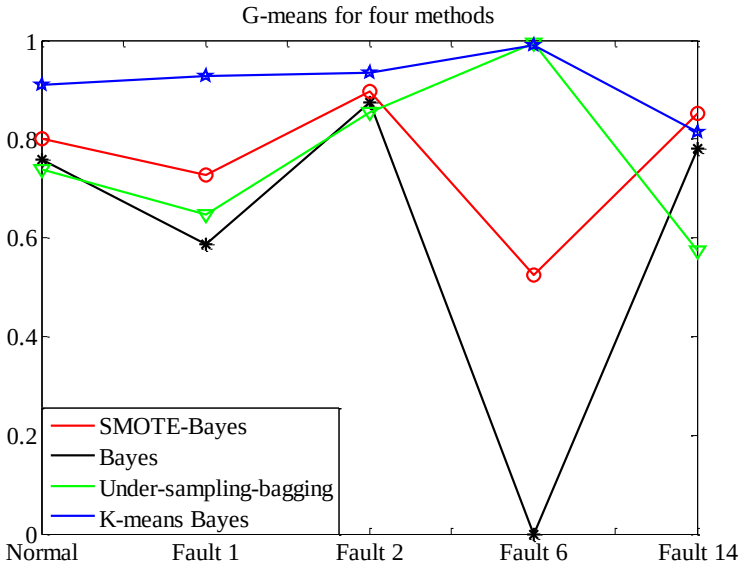


Figure 7: G-means for four methods

In order to evaluate the performance of the proposed method under different numbers of subclasses in the majority, the same test set and training set are selected, except that the training samples of fault 1 are chosen even less to create an absolute scarcity of data artificially. Figure 8 and Table 8 show the results of the K-means Bayes method with different numbers of sub-classes of the majority. Figure 8(a) shows the fault

classification result of the K-means Bayes algorithm with 20 subclasses in majority in which the recall for the fault 1 is very low because of the absolute scarcity of data. Figure 8(b) and Figure 8(c) show the results with 50 subclasses and 200 subclasses respectively, in which the recall for fault 1 gets higher and higher as the number of sub-classes of majority increases. According to the simulation results, it can be concluded that that as the number of subclasses of the majority increases, the fault classification performance of the minority will be improved by the K-means Bayes method, especially for the absolutely scarcity.

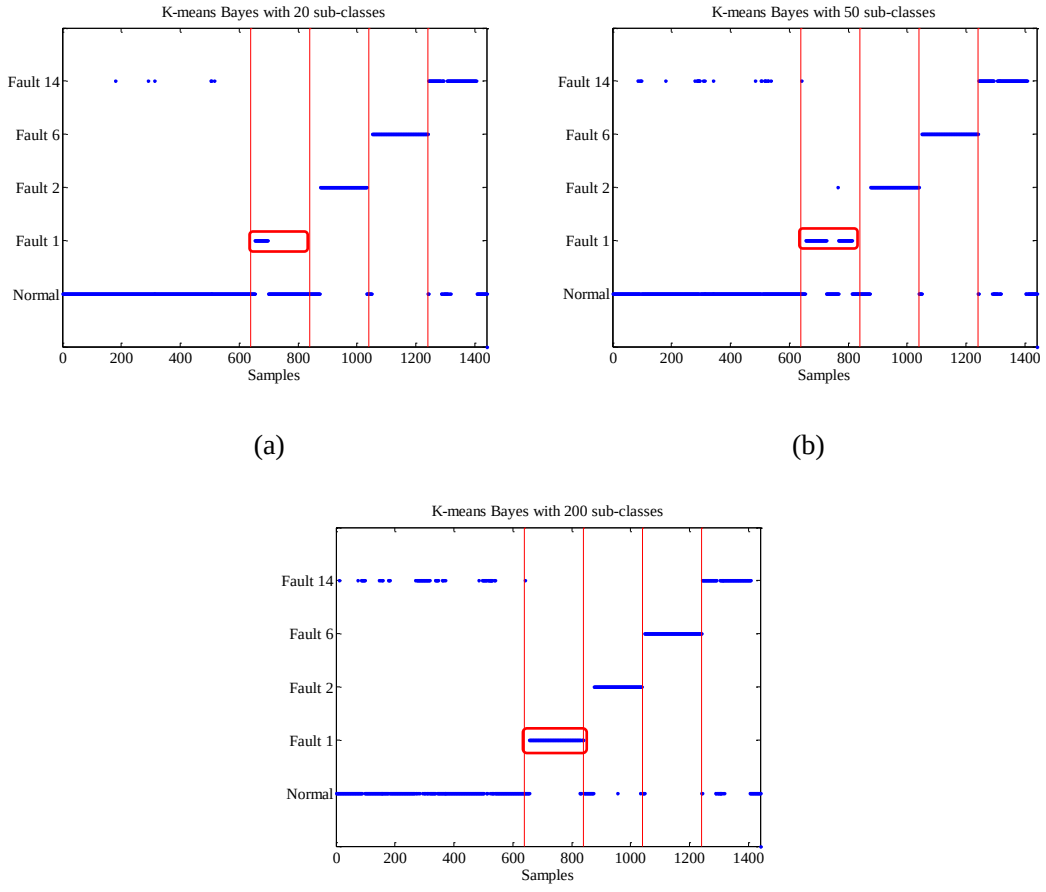


Figure 8: The result of K-means Bayes with different subclasses of majority, (a) classifier does not work well for fault 1 when clustering the majority into 20 subclasses; (b) the accuracy for fault 1 has been improved when clustering the majority into 50 subclasses; (c) when clustering the majority into 200 subclasses, the accuracy for fault 1 is up to 86%.

Table 8: The recall of K-means Bayes with different sub-classes of majority

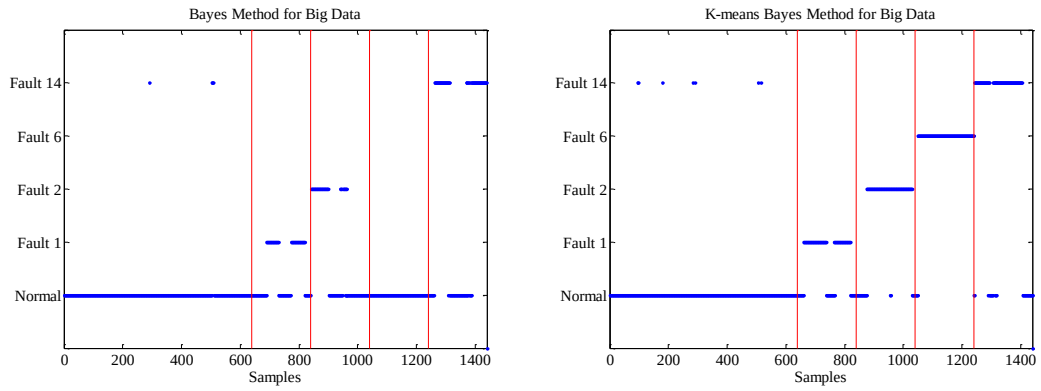
Class	20 sub-classes	50 sub-classes	200 sub-classes
Normal	99.0%	92.0%	88.0%
Fault 1	22.0%	42.5%	86.0%
Fault 2	87.5%	87.5%	87.5%
Fault 6	98.0%	98.0%	98.0%
Fault 14	82.0%	82.0%	82.0%

To carry out simulation of the proposed method under the large scale data or big data case, the number of data under the normal operating mode is determined as 1000000, while the number of each fault class data samples has been collected as 100 for the training dataset. Because it cost much more time to apply the Bayes method without the MapReduce framework, both Na ïve Bayes and K-means Bayes algorithms will be tested on the MapReduce framework to compare their performance. Before conducting the experiment, a computing cluster with the MapReduce framework is established. The cluster consists of 5 computers and the configuration of the four nodes is given in Table 9, and Hadoop 1.6 is used in this experiment.

Table 9: The configuration of the cluster for MapReduce framework

IP of node	Role	CPU	Memory
192.168.1.1	Master	Intel Core i5-4590, 3.3GHz	4 GB
192.168.1.3	Slave	Intel Core i5-4590, 3.3GHz	4 GB
192.168.1.5	Slave	Intel Core i5-4590, 3.3GHz	4 GB
192.168.1.7	Slave	Intel Core i5-4590, 3.3GHz	4 GB
192.168.1.9	Slave	Intel Core i5-4590, 3.3GHz	4 GB

The number of the sub-classes of the majority is set as 1000. The fault classification results of the proposed method and the Naive Bayes method are shown together in Figure 9 and Table 10, respectively. It can be seen that the accuracy of the classifier will be very low if the Na ïve Bayes is used for the big data case, especially for fault 1 and fault 6. When using K-means Bayes, the performance has been greatly improved. Though K-means Bayes costs more time than Na ïve Bayes because of the clustering process before training, the computing time of the K-means Bayes is in an acceptable range. In a whole, we can say the K-means Bayes method works much better than the Naive Bayes method in the case of big data.



(a) (b)

Figure 9: Results of Na ïve Bayes and K-means Bayes for big data, (a) Na ïve Bayes works worse for all classes in big data; (b) K-means Bayes improves the performance of Na ïve Bayes for every class.

Table 10: The precision and computing time of K-means Bayes and Na ïve Bayes for big data

Class	Na ïve Bayes	K-means Bayes
Normal	99.0%	99.0%
Fault 1	44.0%	68.5%
Fault 2	37.5%	72.0%
Fault 6	0%	94.5%
Fault 14	51.5%	76.0%
Computing time	56.3s	223.4s

5. Conclusions

In this paper, a K-means Bayes algorithm is proposed to deal with the imbalanced fault classification problem. The K-means method is used to reduce the degree of the imbalance without losing any information by clustering the majority into subclasses, which can help the minority to describe the boundaries. A MapReduce approach is introduced to implement the proposed method for fault classification in the big data case. Detailed comparative studies between the proposed method and several conventional methods have been carried out through the Tennessee Eastman (TE) benchmark process, results of which demonstrated that the K-means Bayes method achieved the best performance. Besides, the performance of the method has also been tested for different numbers of subclasses and under the big data situation.

In fact, the K-means Bayes method is a preliminary implementation of the “clustering based

classification framework" and was proposed for data with normal distribution. This method is easy to implement and understand for the engineers and is especially suitable to be extended for big data application due to its low computation burden. Further works on this research topic could be carried out by focusing on several promising issues, such as handling non-Gaussian data distributions, nonlinearities among different variables, dynamical nature of process data, deep learning of specific information from imbalanced data and so on [29].

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (61833014, 61722310, 61673337), the Natural Science Foundation of Zhejiang Province (LR18F030001), and the Fundamental Research Funds for the Central Universities 2018XZZX002-09.

References

[1] Z. Ge. Process data analytics via probabilistic latent variable models: A tutorial review. Industrial & Engineering Chemistry Research, 2018, 57, 12646-12661.

[2] Z. Ge. Review on data-driven modeling and monitoring for plant-wide industrial processes. Chemometrics & Intelligent Laboratory Systems, 2017, 171, 16-25.

[3] S.J. Qin, Survey on data-driven industrial process monitoring and diagnosis, Annual Reviews in Control, 36 (2012) 220-234.

[4] Z. Q. Ge, Z. H. Song, F. R. Gao, Review of Recent Research on Data-Based Process Monitoring. Industrial & Engineering Chemistry research 2013, 52, 3543-3562.

[5] S. Yin, S. X. Ding, X. Xie, and H. Luo, "A review on basic data-driven approaches for industrial process monitoring," IEEE Trans. Ind. Electron., vol. 61, no.11, pp. 6418-6428, 2014.

[6] Leo H. Chiang, Mark E. Kotanchek, Arthur K. Kordon. 2004. Fault diagnosis based on Fisher discriminant analysis and support vector machines, Computers & Chemical Engineering, 28, 1389-1401.

[7] Ge Z, Zhong S, Zhang Y. 2016. Semisupervised kernel learning for FDA model and its application for fault classification in industrial processes. IEEE Transactions on Industrial Informatics, 12, 1403-1411.

[8] Jing C, Hou J. 2015. SVM and PCA based fault classification approaches for complicated industrial process, Neurocomputing, 167, 636-642.

[9] Liu Y, Ge Z. 2018. Weighted random forests for fault classification in industrial processes with hierarchical clustering model selection. Journal of Process Control, 64, 62-70.

[10] Tong C, Yan X. A Novel Decentralized Process Monitoring Scheme Using a Modified Multiblock PCA Algorithm. IEEE Trans. Automation Science and Engineering, 2017, 14, 1129-1138.

[11] Z. Chen, S. X. Ding, T. Peng, C. Yang, and W. Gui. Fault Detection for Non-Gaussian Processes Using Generalized Canonical Correlation Analysis and Randomized Algorithms. IEEE Trans. Ind. Electron., vol. PP, pp. 1-1, 2017.

[12] Jiang Q, Yan X. 2014. Monitoring multi-mode plant-wide processes by using mutual information-based multi-block PCA, joint probability, and Bayesian inference, Chemometrics and Intelligent Laboratory Systems, 136, 121-137.

[13] Zhang K, Hao H, Chen Z, Ding X, Peng K. A comparison and evaluation of key performance indicator-based multivariate statistics process monitoring approaches. Journal of Process Control, 2015; 33: 112-126.

[14] Weiss G M. Learning with Rare Cases and Small Disjuncts. Machine Learning Proceedings, 1995:558-565.

[15] Weiss G M, Hirsh H. A Quantitative Study of Small Disjuncts. Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence. AAAI Press, 2000:665-670.

[16] Wallace B C, Small K, Brodley C E, et al. Class Imbalance. International Conference on Data Mining. IEEE, 2011:754-763.

[17] Lemnaru C, Potolea R. Imbalanced Classification Problems: Systematic Study, Issues and Best Practices. International Conference on Enterprise Information Systems. Springer, Berlin, Heidelberg, 2011:35-50.

[18] Japkowicz N, Stephen S. The class imbalance problem: A systematic study. IOS Press, 2002.

[19] Liu X Y, Wu J, Zhou Z H. Exploratory Under-Sampling for Class-Imbalance Learning. International Conference on Data Mining. IEEE Computer Society, 2006:965-969.

[20] Yao L, Ge Z. Scalable learning and probabilistic analytics of industrial big data based on parameter server: framework, methods and applications. Journal of Process Control, 2019, 78, 13-33.

[21] Zhu, J., Ge, Z., & Song, Z. (2017). Distributed parallel pca for modeling and monitoring of large-scale plant-wide processes with big data. IEEE Transactions on Industrial Informatics, 13(4), 1877-1885.

[22] Yao L, Ge Z. Big data quality prediction in the process industry: a distributed parallel modeling framework. Journal of Process Control, 2018, DOI: 10.1016/j.jprocont.2018.04.004

[23] J. Dean, S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107-113, 2008.

[24] N. K. Alham, M. Li, Y. Liu, et al., "A MapReduce-based distributed SVM algorithm for automatic image annotation," *Computers & Mathematics with Applications*, vol. 62, no. 7, pp. 2801-2811, 2011.

[25] C. Chen, Z. Liu, W. H. Lin, et al., "Distributed modeling in a MapReduce framework for data-driven traffic flow forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 22-33, 2013.

[26] V. J. Hodge, S. O'Keefe, J. Austin, "Hadoop neural network for parallel and distributed feature selection," *Neural Networks*, vol. 78, pp. 24-35, 2016.

[27] J. J. Downs, E. F. Vogel, 1993. A plant-wide industrial process control problem. Computers & Chemical Engineering, 17, 245-255.

[28] Joshi M V. On Evaluating Performance of Classifiers for Rare Classes[C]// IEEE International Conference on Data Mining. IEEE Computer Society, 2002:641.

[29] Z. Ge, Z. Song, S. Ding, B. Huang. Data mining and analytics in the process industry: the role of machine learning. IEEE Access, 2017, 5, 20590-20616.