









A Novel Pre-processing Method for Enhancing Classification Over Sensor Data Streams Using Subspace Probability Detection

Yan Zhong¹ , Tengyue Li¹ , Simon Fong¹ , Xuqi Li²,
Antonio J. Tallón-Ballesteros³  , and Sabah Mohammed⁴ 

¹ University of Macau, Taipa, Macau SAR, China

² University of Edinburgh, Edinburgh, Scotland

³ University of Huelva, Huelva, Spain

antonio.tallon@diesia.uhu.es

⁴ Lakehead University, Thunder Bay, Canada

Abstract. The rapid development of the Internet of Things has led to the widespread use of sensors in everyday life. Large amounts of data through sensing devices are collected. The data quantity is massive, but most of the data are repetitive and noisy. When traditional classification algorithms are used for classifying sensor data, the performance of the model is often poor because the classification granularity is too small. In order to better data mine the knowledge from the Internet of Things data which is a kind of big data, a new classification model based on subspace probability detection is proposed. This model can be well integrated with traditional data mining algorithms, and the performance on sensor data mining is greatly improved.

Keywords: Internet-of-things · Data pre-processing · Sensor data streams · Big data analytics

1 Introduction

The rapid development of the Internet of Things (IoT) at present, the affordability of sensor equipment and the maturing connectivity technologies, allow us to collect a lot of useful data. The sensing data hence become ingredients to applications that are designed to provide better quality for everyday life. However, this kind of data has unique characteristics. First, the data collected by the sensor is usually numerical data. Second, when the sensor collects the data, the collection frequency is relatively fast, and data can be collected sporadically, within a few seconds at each time depending on the sampling rate. During each period of time, a large amount of data is gathered. But they may be about the same across successive periods because the changes in the environment or the outdoor activities are slow (compared to body activity recognition). Furthermore, the data collected by the sensors over a certain period of time is repetitive but uninteresting. For example, crowd-sensing and security-oriented sensor applications collect a huge

amount of normal data, in the hope of detecting something that deviates from normal. Therefore, the data collected in a certain period of time may often contain irrelevant data that come from uninteresting and repetitive activities. In short, the data collected by these sensors are characterized by time series, large quantity over a period of time, easy data repetition, and certain noise data. This leads to the consequence that the classifiers that are trained by such data cannot effectively classify tasks. When we conducted data analysis, we found that the reason for the deterioration of the classification effect is often due to that the data was divided too much at high resolution. In daily life, we should pay more attention to the results and phenomena of some abstracted time periods. One extreme example of abstract time period is morning, afternoon and evening. In our new model, a coarse-grained level is adopted for data partitioning to reveal prominent data features while maintaining the data in effective structures. For solving the problem of “diluted data” due to IoT operational nature, a new model is proposed. In this paper, the proposed classification model is empowered by a new probability evaluation classification method treating the input data as data sequence. The advantages of proposed mechanisms can improve the robustness of the model, reduce the sensitivity to noisy training data in the data stream that come from the sensors of the Internet of Things. Therefore, the machine learning model will become better in alignment with real-life prediction objectives. The classification model that is induced using the proposed learning method will be more useful than the direct use of the classification algorithms alone.

2 Related Work

With the advancement of hardware technology, sensor devices are increasing. The sensor-centric Internet of Things has also experienced rapid development. According to statistics [1] until 2017, the total value of the IoT reached 29 billion U.S. dollars. Such a huge market has attracted the attention of industry personnel and academic staff. Over the years, people have been investigating and building smart systems such as smart homes, smart transportation, and smart security [2–4]. The massive increase in IoT devices helps people obtain large amounts of sensory data. How to tap valuable information from this vast amount of data and form knowledge to serve life more effectively is an important issue. Some researchers have tried to use the data mining technology in the development of the IoT to make it more intelligent [5–8]. Clustering is commonly used in data mining of the IoT and the most common clustering method is K-means [9] which is very mature in traditional data mining. The distribution of Internet of Things data in some cases is a clustering problem [10, 11], but the classification results presented by clustering are only similar data and cannot be judged. If people are unfamiliar or unclear with the collected data, they cannot rely on the clustering results to dig out effective knowledge. In supervised learning, people often use decision tree algorithms for data mining in the IoT [12, 13]. In addition, probabilistic models are also widely used, such as the Naïve Bayesian model [14, 15]. In machine learning, there is also a simple and efficient classification method that is SVM [16], combining with the kernel function can linearly separate data in high-dimensional space. However, the traditional classification method has unstable performance on the actual sensor data because of the special nature of IoT data. Through analysis and observation, all collected sensor data

have highly repetitive characteristics and often contain noise data whose source may be due to sensor detection errors. This leads to too many samples of negative instances in the classification process, and the accuracy of the trained model decreases. In real life, people pay more attention to the results of a period of time, which is inconsistent with the frequency with which sensors collect data. Therefore, in order to solve this problem, this paper proposes a new type of data mining model, with a pre-processing which consists of constructing the subspace from the initial data set, and finally using the traditional classification method for classification. This model can greatly improve the accuracy of classification. Moreover, this model is robust, and it can be combined with various classification methods.

3 Our Proposed Model

In this section, the paper is going to introduce the model overview and example, formulating the high-level mathematics model for this new pre-processing process (PP). Compared with other classical classifier algorithms and pre-processing process, this pre-processing method combining with other classical classifier algorithms are applied for predicting the major label among a small group of continuous sequential data which are ordered by time. This new method will change the unit of information from a singular time point to a period of time. Each dataset tested in this paper is the data that come from a typical wearable sensor. The data have several labels such as "walking", "running" and so on in the prediction class. In our daily life, be it walking or running, there is a well-known observation that the sensed data would have similar adjacent instances along the data stream. The data instances carry the same label except when the instances are generated located at the boundary of two different actions and the noisy data. When people expect to classify which actions the subject under monitoring is doing through the data, it would be more effective for the machine learning model to be trained with a group of continuous data instances that are grouped with a common target label, than singular data instances with precise but similar data values individually. Our proposed pre-processing method is designed to generate new datasets for training and testing from the original train dataset and test dataset. The meaning of the instance in the new dataset is not merely the information converted from a singular instance which is in the original dataset, but the information from a continuous period of instances. How to convert the relevant information effectively is the most important part of the design, which is reported in this paper. To begin with, the feasibility of this algorithm in overall is defined, and this paper is going to illustrate how this algorithm works in detail in the next section, followed by the experiment result. First of all, when the train dataset is coming, we would like to get some information from them especially from the data which has the same label. Then we collect some sample data instances from each group of instances which have the same label (such as we collect sample instances from a set that each instance inside are labelled by "walking" and then do the same process in the "running" set). And we assume that these sample instances could represent and basically conclude most of traits from class labels. Then these sample subsets are called "standard sets" for each class label which means every class label has one sample set. If there are n labels in the dataset, then there will be n sample dataset. This is just the beginning of the transformation. In the next step, in order to stimulate a normal period of instances in the dataset,

we use the sampling method to collect some data from each class to form n label dataset. Theoretically, the size of these label sets is smaller than standard sets because we want to find some similarity index between standard sets and label sets using isolation forest (iForest) detection algorithm [17] which is an algorithm to detect the non-isolation rate between two datasets.

The workflow is divided into three main steps: a) the original sequence training dataset (T0) will be transformed into a new training dataset (T1) while the attributes and the length of T1 are changed and optimized by PP, b) the original testing dataset (D0) will be transformed into a new test dataset (D1) by PP. It needs to be noticed that the core of PP method is based on Isolation Forest algorithm and c) the user could apply these new training dataset and testing dataset for prediction coupled with some known classification algorithms. Abstract flow charts are shown in Figs. 1 and 2, indicating how overall this concept works for reconstructing the training and testing datasets respectively using subspace division.

A set of formulae are developed which are used to explain each step in aforementioned figures by explaining the operation pertaining to how the data is processed and converted between successive steps. Suppose $S = \{x_1, \dots, x_t, x_{t+1}, \dots\}$ to be the original dataset, $x_t \in S$ where $t = 1, 2, \dots$, and the length of dataset S is fixed. Here SP is denoted as a collection of all the subsets of S . For every data $x_t \in S$, there are m attributes characterizing them. And then the attributes space A would be defined as $A = \{(a_1, a_2, \dots, a_i, \dots, a_m) | a_i \text{ is the value of the } i^{\text{th}} \text{ attribute, for } i = 1, \dots, m\}$ and the attributes' types are mixed by numeric and nominal data. All the data were labeled, here it is called class and the collection of classes is $C = \{C_1, \dots, C_n\}$. Then the data $x_t = (x^*t, c)$ where $x^*t = (x_1, t', \dots, x_i, t', \dots, x_m, t')$, $x^*t \in A$ and $c \in C$. In this experiment, cross-validation was applied for splitting the original dataset S into training dataset and testing dataset. We denote one of the training datasets as T_0 and testing dataset as D_0 whose instances have no labels. Before pre-processing the training data, we need to define some functions and notations to make it more accessible.

Formulae #1

Define a function $Class$ to print out the class c_h of instance h where $h \in S$ and $c_h \in C$.

$$Class : S \rightarrow C, Class(h) = c_h \tag{1}$$

Formulae #2

The function Maj is defined on SP which means to print out the major class c_{maj} of dataset w_x and the operation $| \cdot |$ means to calculate the length of the set.

$$Maj : SP \rightarrow C, Maj(w_x) = \arg_{c \in C} \frac{|\{h \in w_x | Class(h) = c\}|}{|w_x|} = c_{maj} \tag{2}$$

Formulae #3

Define a function $Div(\cdot)$ to collect the instances whose class is c from T_0 then create a subset T_0^c of T_0 .

$$Div : SP \times C \rightarrow SP, Div(T_0, c) = \{x \in T_0 | Class(x) = c\} = T_0^c \tag{3}$$

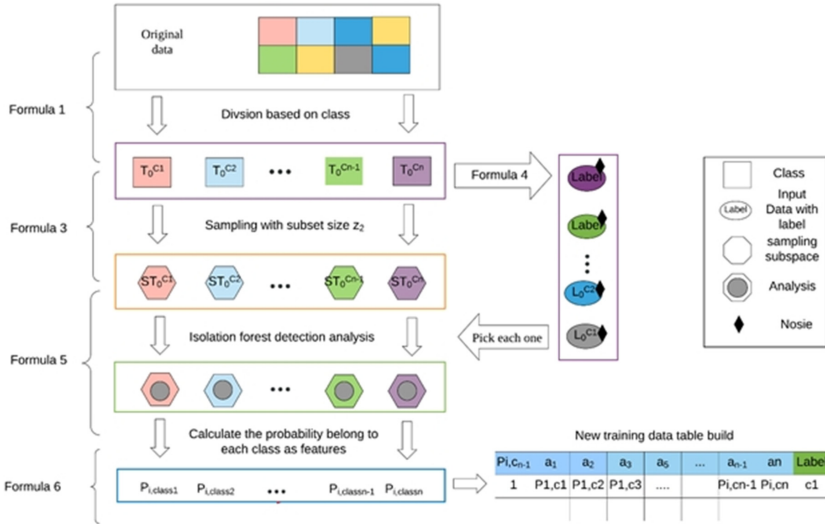


Fig. 1. Block diagram that shows how a new training dataset is reconstructed by our proposed preprocessing method.

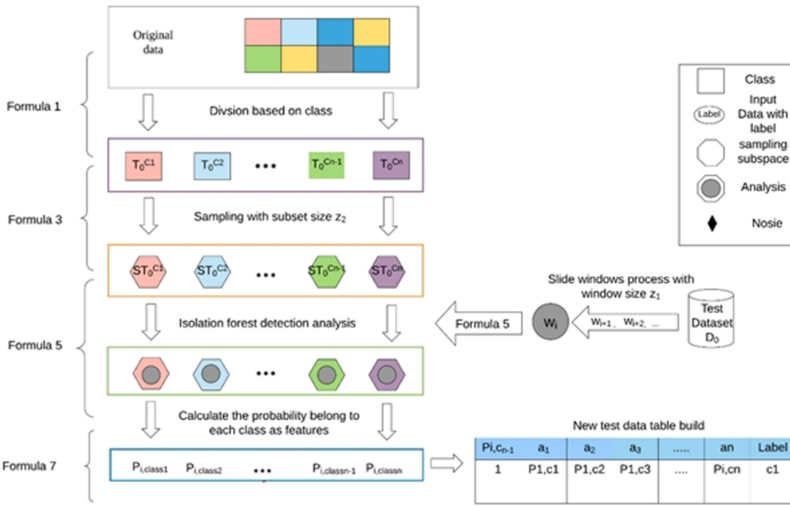


Fig. 2. Block diagram that shows how a new testing dataset is reconstructed by our proposed preprocessing method.

Formulae #4

$Sam_{md}(\cdot)$ is a function that take z samples from T_0^c based on curtain sampling method md , here md could be one of the sample random sampling methods and stratified random sampling. Besides the sample set is named as ST_0^c .

$$Sam_{md} : SP \times IR \rightarrow SP, Sam_{md}(T_0, z) = ST_0^c \quad (4)$$

Formulae #5

Function $ITR(\cdot)$ used $ST_0^{C_j}$ as a standard case to train an Isolation Forest which is an algorithm created by Prof. Zhihua Zhou [17] for detecting the isolation point and then putting the L_{C_i} sample set into the Isolation Forest model to classify whether there are isolation points or not in the L_{C_i} . Finally computing the rate of data in L_{C_i} that normally obeys the distribution in $T_0^{SC_j}$. In other words, this function is to compute the non-isolation rate $P_{i,j}$.

$$ITR : SP \times SP \rightarrow [0, 1], ITR(L^{C_i}, ST_0^{C_j}) = P_{i,j} \quad (5)$$

Step 1: Reconstruct Training Data Table

With the definitions and notations above, the process of this algorithm will be presented below and also be described in the Figs. 1 and 2. When an original dataset S comes, through cross-validation, it could get one of training datasets T_0 , then divide the T_0 into a collection of sub-dataset $\{T_0^{C_j}\}_{j=1}^n$ where

$$T_0^{C_j} = Div(T_0, c_j) \quad (6)$$

Then the algorithm will do the first time sampling (the reason for why it is first time will be explained at the end of this step) with specific sampling method md to gain the trait from $\{T_0^{C_j}\}_{j=1}^n$ then it will get a series of sampling dataset $\{ST_0^{C_j}\}_{j=1}^n$ where

$$ST_0^{C_j} = Sam_{md}(T_0^{C_j}, z_2) \quad (7)$$

In order to simulate the arbitrary test sliding window w (specific description of w is in the step 2) where $Maj(w) = C_i$, this pre-processing method will do the first time sampling with specific sampling method md to gain the trait from $\{T_0^{C_j}\}_{j=1}^n$ then we get a series of Label dataset $\{L_0^{C_i}\}_{i=1}^n$ where

$$L_0^{C_i} = \left\{ Sam_{md}(T_0^{C_i}, z_1) \right\} \cup N_i \quad (8)$$

and N_i is a special noise set to simulate the noise in the arbitrary test sliding windows w where $Maj(w) = C_i$. It is clear to find that the length of $\{ST_0^{C_j}\}_{j=1}^n$ and $\{L_0^{C_i}\}_{i=1}^n$ are same such as n . Therefore, that is easy to get $n \times n$ combinations such as $\left\{ \left(L_0^{C_i}, ST_0^{C_j} \right) \right\}_{i=1}^n$ $n_j = 1$. With these $n \times n$ combinations, the preprocess sets $ST_0^{C_j}$ as the second input element (standardcase) of ITR and $L_0^{C_i}$ as the first input element (detection case) of ITR . At the end of the whole process, a training table TDT could be generated.

Formulae #6

Construct a Matrix $TDT \in [[0, 1]^{n \times n} | C^{n \times 1}]$ the element in the i th row and j th column of TDT is computed by function $ITR(L_0^{C_i}, ST_0^{C_j})$ and the element in the k th row and n

+ 1 column is C_i :

$$[\text{TDT}]_{i,j} = \text{ITR}\left(L_0^{C_i}, ST_0^{C_j}\right) \text{ for } 1 \leq i \leq n \text{ and } 1 \leq j \leq n$$

$$[\text{TDT}]_{k,n+1} = C_i \text{ for } 1 \leq k \leq n \quad (9)$$

Finally, the definition of function PP would be constructed from TDT.

Definition: Function PP is defined to generate a new training dataset T1 from T0 with the setting of z_1 (the length of $L_0^{C_i}$) and z_2 (the length of $ST_0^{C_j}$).

$$\text{PP1: } SP \times IR \rightarrow SP', \text{ PP1}(T_0, z_1, z_2) = T_1$$

where $T_1 = \{xx_1, \dots, xx_n\}$ and

$$xx_i = \{[\text{TDT}]_{i,j}\}_{j=1}^{n+1} = \left(\{[\text{ITR}(\{\text{Sam}_{\text{md}}(\text{Div}(T_0, C_i), z_1) \cup N_i, \text{Sam}_{\text{md}}(\text{Div}(T_0, C_j), z_2)\}_{i,j})\}_{j=1}^n, C_i) \right) \quad (10)$$

Now, we are going to explain why it is written as “the first time”. If this algorithm just does sampling for the one time, then the TDT just have n instances which are too few. So, in order to increase the size of the dataset, this algorithm will do the step 1 for many times and the repeating frequency depends on user’s choice. Finally, we combine all the training Tables into a new training data set. So, does it work for the test data table. In the following section, the first-time loop is described because the principle is the same.

Step 2: Reconstruct Testing Data Table

This step is to transform the testing dataset D0 to new testing dataset D1 where $D_0 = \{y_1, \dots, y_r\}$ and $D_1 = \{yy_1, \dots, yy_t, \dots, yy_r\}$. Sliding window w will be applied as a instrument to do so and the length of sliding window p can be set by user, e.g. $P = z_2$. Let $w_1 = \{x_1, \dots, x_p\}$, $w_z = \{x_{1+z}, \dots, x_{p+z}\}$, $W = \{w_1, \dots, w_z, \dots, w_r\}$ where r is determined by the length of slide window and the length of test dataset. Because of the similar technique, the testing dataset is also calculated by ITR with the input. However, the input is no longer the $n \times n$ combinations as TDT but a $r \times n$ combinations such as $\{[\text{ITR}(w_t, ST_0^{C_j})]\}_{j=1}^n$ $rt = 1$.

Formulae #7

Construct a Matrix $\text{TDT}_1 \in [0, 1]^{r \times n}$:

$$[\text{TDT}_1]_{t,j} = \text{ITR}(w_t, ST_0^{C_j}) \quad (11)$$

where $ST_0^{C_j} = \text{Sam}_{\text{md}}(T_0^{C_j}, z_2)$ for $j = 1 \dots n$ and $t = 1 \dots r$. w_t is the t th sliding window.

While the element in the t th row and j th column of TDT_1 is computed by function $\text{ITR}(w_t, ST_0^{C_j})$.

With the help of TDT_1 the final high-level function of step two can be defined.

Formulae #8

Function PP2 is defined for transforming testing dataset D0 into new testing dataset D1 which has same categories of attributes as T1. But part of computing input changes because we no longer utilize T0 to gain sample set $L_0^{C_i}$ whose size is z1 but using sliding windows of D0 while the class of new testing data is replaced by major class of a certain sliding window.

$$PP2: SP \times IR \rightarrow SP'', PP(T_0, D_0, z_2) = D_1$$

where $D_1 = \{yy_1, \dots, yy_t, \dots yy_r\}$, and the

$$yy_t = (\{[TDT_1]_{(i,j)}\}_{j=1}^n, Maj(w_t)) = (\{ITR(w_t, Sam_{md}(Div(T_0, c_i), z_2))\}_{j=1}^n, Maj(w_t)) \quad (12)$$

Step 3: Model Learning

After using the pre-processing method to generate the new training dataset T1 and new testing dataset D1, user could apply different algorithms to make the prediction with the help of T1 and D1. Then a group of high-level equations that represent these processes $\{algo_i\}_{i=1}^5$ is defined as follow:

Formulae #9

$\{algo_i\}_{i=1}^5$ is defined for a group of high-level equations that use training data set (such as T1), testing dataset (such as D1) as well as a group of parameters for training the model and testing the model, then finally it gets the performance evaluation index pf.

$$algo_{i,1} : SP' \times IR \rightarrow SP'', Pre \rightarrow IR, algo(T_1, D_1, P) = pf \quad (13)$$

where Pre is a collection of all possible specific parameters with respect to the demand of user and $P \in Pre$, pf is a performance evaluation index which is a combination of several statistical parameters of model such as accuracy, recall and F1-score. In this paper, there are five Algorithms involved which are SVM, logistic regression, C4.5, Bayes classifier and KNN. After comparing the performance between before and after per-processing, we found that this new method could improve the accuracy of the prediction. The following section will show the experiment results in detail.

4 Experiment

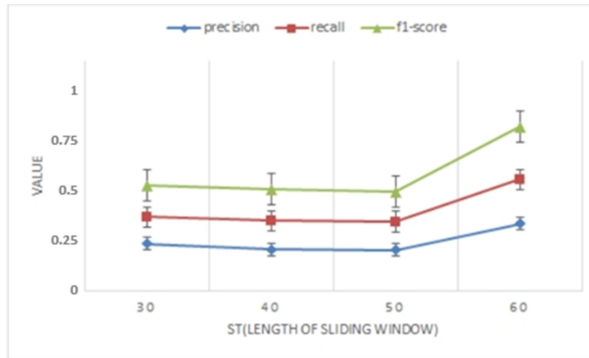
In this section we evaluate the performance of the proposed model through extensive experiments. We choose a sensor dataset which is a typical data stream in mobile IoT applications and comparing the performance of the algorithms after model optimization to the original algorithms. The Heterogeneity Human Activity Recognition (HHAR) data set, from Smartphones and Smart watches, devised to benchmark human activity recognition algorithms (classification, automatic data segmentation, sensor fusion, feature extraction, etc.) in real-world contexts; specifically, the dataset is gathered with a variety of different device models and use-scenarios, in order to reflect sensing heterogeneities to be expected in real deployments [18, 19]. The data can be obtained free from the public archive at UCI repository.

We choose 5 traditional methods and use three evaluation indicators (Recall, Precision, F1-score). They are: K-Neighbors Classifier, Logistic Regression, Gaussian Naïve Bayes, Decision Tree, and Support Vector Machines. The parameters of the machine learning models are set by their default values. When constructing a new training set, the length of the selected window is 100, and 20 serialized instances are constructed from serialized samples in each category. Then Formula #4 is applied to select the number of $T0c$. This number should be greater than the number of spaces selected by the window. Therefore, 30 instances are selected as $T0c$ in different categories, and then the iForest algorithm is used to calculate the category probability. The category probability calculations here are using a similar number of percentages. Building a new test set also has the same steps, the length of the selected window is 20, and 20 serialization instances are constructed from random serialized samples in each category of the test set. In the different categories, 30 instances are still selected as $T0c$, and similar probability is calculated using Eq. 5, so that a new test set is formed. Using a preliminary experiment using the sensor dataset two, it can be seen that the model performed well with large improvement over the classification model that is built without the proposed pre-processing. The comparison results are tabulated in Table 1. The last two columns namely ‘Score’ and ‘Our Model’ are the performance indicator values from the classification model which has not been pre-processed and pre-processed, respectively. The running time is only a few minutes for every classifier. In fact, we are more concerned about what has happened over time from the perspective of doing machine learning from time-series. Therefore, the partition of granularity and the length of the window are particularly important. Furthermore, granularity experiments and iterative experiments are designed. If the selected window length is appropriate, the more iterations, the more training instances will be formed. The same parameters in building the training dataset and the testing set are maintained. In the additional experiment W is selected as 20, 30, 40, 50 and ST as 30, 40, 50, 60. The number of training set iterations is set at 100. As it can be seen in Fig. 3, the size of the sliding window which decides how much per pass the instances would enter into the pre-processing and training the classifier, matters. When the window size reaches over 60 approaching 100, almost full score can be obtained in the cross-validation mode of testing of the classifier. This implies sufficient amount of data per pass would help in framing up the subspaces. However, too large the sliding window may lead to a problem of incurring high latency. Having large sliding window is like reverting the incremental learning which is fast as it learns online, to traditional batch learning where the full set of data is used for model induction. The appropriate size of window for balancing between latency and the highest possible accuracy worth in-depth investigation in the future work. Although we can tune the sliding window size to be moderately suitable for accuracy and latency, what if only a limited (small) amount of training samples are available? To test out such extreme situation, another experiment is simulated where only relatively little training and testing data are assumed available and used in the pre-processing. The objective is to test the correlation between accuracy performance and the volume of the training dataset.

From Table 2, when the number of epochs is equal, the more data, the better the effect it shows from our pre-processing approach. Please note that the performance indicator values are averaged over the five classifiers used in our experiments. When the training

Table 1. Comparison results between classifiers built without and with pre-processing

Original algorithm	Performance	Score	Our model
KNeighbours classifiers	Precision	0.23	1.00
	Recall	0.25	0.91
	F1-score	0.35	0.96
Logistic registration	Precision	0.02	1.00
	Recall	0.19	0.92
	F1-score	0.15	0.96
GaussianNB	Precision	0.10	1.00
	Recall	0.25	0.91
	F1-score	0.18	0.95
SVM	Precision	0.10	1.00
	Recall	0.09	0.91
	F1-score	0.14	0.96
Decision tree	Precision	0.10	1.00
	Recall	0.19	1.00
	F1-score	0.12	1.00

**Fig. 3.** Averaged performance of classifiers with various W sizes.

size and the testing size gradually increase, although precision, recall, and f1-score may fluctuate, the overall trend is rising. This proves that, to a certain extent, the greater the detection from the probabilistic sample size given sufficient amount of data, the higher the accuracy of detection. From the results, it is found that Pearson coefficients are, 0.620456148, 0.828351883 and 0.728648368 respectively which indicates quite high the correlations between the amount of training data size and the precision, the recall and the balanced F-score.

Table 2. Performance results from various dataset sizes.

Train size	Test size	Train epoch	Precision	Recall	F1 score
30	20	100	0.51	0.29	0.34
40	30	100	0.45	0.31	0.34
50	40	100	0.44	0.31	0.32
60	50	100	0.73	0.48	0.37

5 Conclusions

This paper describes a subspace probabilistic detection pre-processing model based on the subspace-attribute probability calculation. The proposed model is to be used as a pre-processing method that transforms the time diluted dataset to one that can be better characterized by the temporal information from the data, hence better classification model training and prediction results. Five popular classification algorithms are used to test with the pre-processing method by performing classification over sensor data that characterize certain human activities. Such sensor data represent a kind of big data streams that possesses new data mining challenges due to their sheer volumes and sequential nature. This model can effectively solve the problem of repeatability and noise that exist in the sensor data. Through experiments, we can see that this model can effectively improve the performance of traditional machine learning classification algorithms in data mining in the sensor data by large magnitude.

Acknowledgment. The authors are thankful to the financial support from the following research grants: 1) MYRG2016–00069, titled “Nature-Inspired Computing and Metaheuristics Algorithms for Optimizing Data Mining Performance”, offered by RDAO/FST, University of Macau and Macau SAR government; 2) FDCT/126/2014/A3, titled “A Scalable Data Stream Mining Methodology: Stream-based Holistic Analytics and Reasoning in Parallel” offered by FDCT of Macau SAR government and 3) TIN2017–88209-C2-R project of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT) and FEDER funds.

References

1. M&M Research Group: Internet of Things (IoT) & M2M communication market - advanced technologies, future cities & adoption trends, roadmaps & worldwide forecasts 2012–2017. Technical report. Electronics.ca Publications (2012)
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. *Comput. Netw.* **54**(15), 2787–2805 (2010)
3. Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I.: Internet of things: Vision, applications and research challenges. *Ad Hoc Netw.* **10**(7), 1497–1516 (2012)
4. Bandyopadhyay, D., Sen, J.: Internet of things: applications and challenges in technology and standardization. *Wirel. Pers. Commun.* **58**(1), 49–69 (2011)

5. Cantoni, V., Lombardi, L., Lombardi, P.: Challenges for data mining in distributed sensor networks. In: Proceedings of International Conference on Pattern Recognition, vol. 1, pp. 1000–1007 (2006)
6. Keller, T.: Mining the internet of things: Detection of false-positive RFID tag reads using low-level reader data. Ph.D. Dissertation. The University of St. Gallen, Germany (2011)
7. Masciari, E.: A framework for outlier mining in RFID data. In: Proceedings of International Database Engineering and Applications Symposium, pp. 263–267 (2007)
8. Bin, S., Yuan, L., Xiaoyi, W.: Research on data mining models for the internet of things. In: Proceedings of International Conference on Image Analysis and Signal Processing, pp. 127–132 (2010)
9. McQueen, J.B.: Some methods of classification and analysis of multivariate observations. In: Proceedings of Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
10. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. *ACM Comput. Surv.* **31**(3) 264–323 (1999). ([43] Xu, R., Wunsch-II, D.C.: Survey of clustering algorithms. *IEEE Trans. Neural Netw.* **16**(3), 645–678 (2005)
11. Xu, R., Wunsch-II, D.C.: Survey of clustering algorithms. *IEEE Trans. Neural Netw.* **16**(3), 645–678 (2005)
12. Safavian, S., Landgrebe, D.: A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man Cybern.* **21**(3), 660–674 (1991)
13. Friedl, M., Brodley, C.: Decision tree classification of land cover from remotely sensed data. *Remote Sens. Environ.* **61**(3), 399–409 (1997)
14. McCallum, A., Nigam, K.: A comparison of event models for Naivebayes text classification. In: Proceedings of National Conference on Artificial Intelligence, pp. 41–48 (1998)
15. Langley, P., Iba, W., Thompson, K.: An analysis of Bayesian classifiers. In: Proceedings of National Conference on Artificial Intelligence, pp. 223–228 (1992)
16. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, Cambridge (2000)
17. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: Eighth IEEE International Conference on Data Mining. ICDM 2008, pp. 413–422. IEEE (2008)
18. Fong, S., Song, W., Cho, K., Wong, R., Wong, K.K.L.: Training classifiers with shadow features for sensor-based human activity recognition. *Sensors* **17**(3), 476, 27 (2017)
19. Fong, S., Liu, K., Cho, K., Wong, R., Mohammed, S., Fiaidhi, J.: Improvised methods for tackling big data stream mining challenges: case study of human activity recognition. *J. Supercomput.* Springer **16**, 1–33 (2016)