Prototype Selection and Generation with Minority Classes Preservation

Konstantinos Xouveroudis
Dept. of Information and Electronic Engineering
International Hellenic University
Sindos, Thessaloniki, Greece
xouveroudiskostas@yahoo.com

Stefanos Ougiaroglou
Dept. of Information and Electronic Engineering
International Hellenic University
Sindos, Thessaloniki, Greece
stoug@ihu.gr

Georgios Evangelidis
Dept. of Applied Informatics, School of Information Sciences
University of Macedonia
Thessaloniki, Greece
gevan@uom.gr

Dimitris A. Dervos
Dept. of Information and Electronic Engineering
International Hellenic University
Sindos, Thessaloniki, Greece
dad@ihu.gr

***Abstract*—Instance-based classifiers become inefficient when the size of their training dataset or model is large. Therefore, they are usually applied in conjunction with a Data Reduction Technique that collects prototypes from the available training data. The set of prototypes is called the condensing set and has the benefit of low computational cost during classification, while, at the same time, accuracy is not negatively affected. In case of imbalanced training data, the number of prototypes collected for the minority (rare) classes may be insufficient. Even worse, the rare classes may be eliminated. This paper presents three methods that preserve the rare classes when data reduction is applied. Two of the methods apply data reduction only on the instances that belong to common classes and avoid costly under-sampling or over-sampling procedures that deal with class imbalances. The third method utilizes SMOTE over-sampling before data reduction. The three methods were tested by conducting experiments on twelve imbalanced datasets. Experimental results reveal high recall and very good reduction rates.**

***Index Terms*—k-NN Classification, Imbalanced data, Prototype Selection, Prototype Generation, SMOTE, Rare classes**

I. Introduction

The k-Nearest Neighbours (k-NN) classifier is a simple and widely-used instance-based classification algorithm. For each unclassified instance x, it searches in the training data and retrieves the k nearest to x instances, which are called nearest neighbours. Then, x is classified to the majority class among the classes in the set neighbourhood. The k-NN classifier is effective and can by used in many application domains. However, it is a CPU intensive classifier and it cannot handle large volumes of training data since all distances between the unclassified instance and the training data must be computed.

Many Data Reduction Techniques for efficient instance-based classification have been proposed in the past. They are divided into two main categories: (i) Prototype Selection (PS) algorithms [1] and (ii) Prototype Generation (PG) algorithms [2]. PS algorithms may be either condensing or editing algorithms and work by selecting instances (prototypes) from the original training data. Condensing algorithms store in the condensing set only the close-border instances, while editing algorithms remove noise and smooth the decision boundaries between different classes. On the other hand, PG algorithms summarize instances that are close to each other and generate artificial prototypes that represent groups of similar instances. They generate more prototypes for the decision boundaries areas and fewer prototypes for the "internal" class areas. PG and PS-condensing algorithms try to reduce the computational cost of the k-NN classifier by keeping the size of the condensing set as small as possible without sacrificing classification accuracy. It is worth mentioning that the reduction rate achieved by PS-condensing and PG algorithms depends on the level of noise and the number of classes in the training data. Thus, editing may be applied beforehand.

Although PS-condensing and PG algorithms can efficiently reduce the training data, they cannot deal with imbalanced data [3], [4]. The number of prototypes selected or generated for the minority (rare) classes may be insufficient, and, sometimes rare classes may even be eliminated. As a result, crucial rare situations (such as extreme weather conditions, earthquakes, tsunamis, diseases, etc.) will be misclassified when the k-NN classifier runs over the condensing set built by a PS-condensing or a PG algorithm. Yet, the correct prediction of a rare situation is usually more significant than the correct prediction of common (non-rare) ones. On the other hand, the misclassification of a rare situation may be pernicious. The present work is motivated by these observations.

The contribution of the present paper is the development of three methods for the preservation of the instances of the rare classes in data reduction tasks. The methods are applied in conjunction with a PS-condensing or PG algorithm. The first two methods preserve the instances that belong to rare classes by applying data reduction only on instances that belong to common classes and avoid costly under-sampling or over-sampling techniques that are usually used to balance

978-1-6654-0032-9/21/$31.00 ©2021 IEEE

such training datasets. The third method utilizes the popular Synthetic Minority Over-Sampling Technique (SMOTE) [5] to balance the training data and then applies data reduction. Furthermore, the paper presents an extended experimental study conducted on twelve binary and multi-class imbalanced training datasets, where various PS-condensing and PG algorithms are tested on the original training datasets with and without the application of the proposed Rare Class Preservation Methods.

The rest of this paper is organized as follows: Section II reviews on classification with imbalanced datasets, Section III discusses the proposed Rare Class Preservation Methods, Section IV presents the experimental outcomes, and, Section V concludes the paper.

II. Classification with imbalanced training datasets

Real-life training datasets are usually imbalanced [3], [4]. When performing classification on such datasets, the metric of Accuracy is insufficient. Suppose that a classification system has been trained to predict earthquakes. The training set used contains the classes: "no-earthquake" and "earthquake". Surely, the class "earthquake" is rare. A wrong prediction for "no-earthquake" may result in loss of human lives. On the other hand, the cost of a wrong prediction for "earthquake" is not so high. In effect, the cost is some unneeded protective actions for an earthquake that will not happen. The ZeroR classifier, that always predicts "no-earthquake", achieves high accuracy. However, it is a totally unacceptable classifier because it does not predict any earthquake (i.e., it predicts False-Negatives). On the other hand, a classification system that very often wrongly predicts "earthquake" (i.e., False-Positives) is unreliable.

In the case of imbalanced training data, Precision, Recall, F-measure and other metrics must be considered [6]. These metrics are computed by considering the confusion matrix that contains the number of True-Positives (TP), True-Negatives (TN), False-Positives (FP) and False-Negatives (FN). Precision is computed by the formula $\frac{|TP|}{|TP|+|FP|}$ and counts how many positive predictions are correct. Recall is computed by the formula $\frac{|TP|}{(|TP|+|FN|)}$ and counts how many positives are correctly predicted.

In the case of the classification system for earthquake predictions, Precision counts how many earthquake predictions are correct while Recall counts how many earthquakes were predicted. Thus, Recall is more significant than Precision because even few wrong predictions for "no-earthquake" (i.e., FN) may be disastrous. However, a classification system that always predicts positive achieves Recall equal to one (there are no FNs - all earthquakes are correctly predicted), but it is unreliable since it predicts many FPs and, as a result, the precision is low. Consequently, in the case of systems that predict extreme situations like earthquakes, Recall is more significant than Precision, but Precision should not be ignored.

On the other hand, suppose that a web user submits a query on a search engine. Suppose that the search engine indexes one thousand web pages. Ten web pages contain the information that the user is looking for. Ideally, the search engine returns the ten relevant web pages and no more. In that case, both Precision and Recall will be equal to one. In contrast, Precision and Recall will be equal to zero if the search engine returns only irrelevant web pages. In the case of search engines, Precision is more significant than Recall, but Recall should not be ignored. The user cannot afford low Precision and high Recall because the search engine returns many relevant web pages and many more irrelevant. Here, there is a risk that the relevant web pages will be lost among the many irrelevant ones and the user may not find what he/she is looking for. On the other hand, the user can afford a relatively low Recall with high Precision. This means that the user has the chance to find what he or she is looking for from the few relevant web pages retrieved but at the same time the retrieved irrelevant web pages that mislead the user are kept at a minimum level.

In effect, the significance of Precision and Recall depends on the application domain. The F-score is the harmonic mean of Precision and Recall. It is calculated as follows:

$$F - score = 2 * \frac{\frac{|TP|}{|TP|+|FP|} * \frac{|TP|}{|TP|+|FN|}}{\frac{|TP|}{|TP|+|FP|} + \frac{|TP|}{|TP|+|FN|}}$$

Although there are some under-sampling techniques [7] that try to deal with class imbalances, the most popular techniques are based on over-sampling [7], [8]. Both over-sampling and under-sampling aim at balancing the class distributions. Oversampling techniques strengthen the rare classes by adding new instances to them. The most common oversampling technique is called Synthetic Minority Over-Sampling Technique (SMOTE) [5]. SMOTE works as follows: In each iteration, a random instance x that belongs to the rare class is selected. Then, the k nearest to x neighbors that also belong to the same rare class are retrieved (typically $k = 5$). SMOTE continues by randomly selecting one of the neighbors (nn) and generating a synthetic instance at a randomly selected point between x and nn in the feature space. In effect, x and nn form a line in the feature space and the synthetic instances are generated on a random point on this line. The iterations stop when the data is balanced or when a predefined number of synthetic instances are generated. Thus, SMOTE can be used to generate as many synthetic rare class instances as required.

III. Rare Class Preservation Methods

This section presents three methods that preserve the instances that belong to rare classes in data reduction tasks. The preservation of those instances in the condensing set is important since they contribute to the correct prediction of rare situations of vital importance. For data reduction, any PS-condensing or PG algorithm can be used. The methods divide the training data into two subsets. One subset contains only the instances of rare classes while the other contains the instances of common classes. The first subset is called subset of rare classes while the second one is called subset of common classes. The training dataset is divided using a prespecified threshold. The classes with fewer instances than the threshold are taken to comprise rare classes.

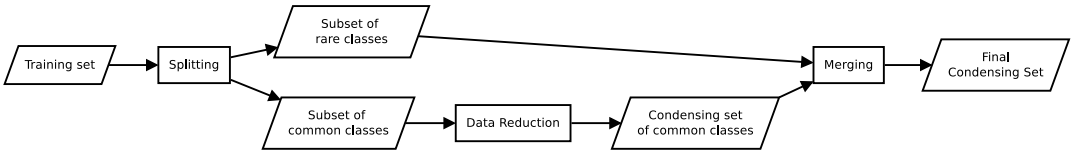

Fig. 1. RCPM1 Flow chart

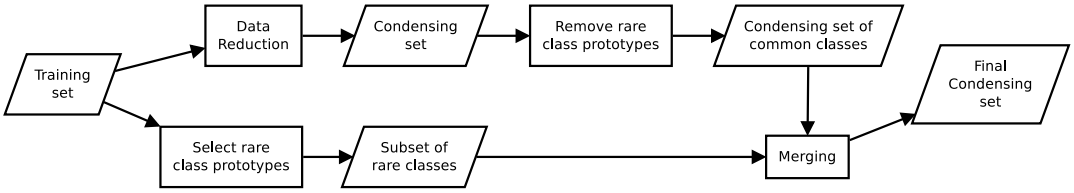

Fig. 2. RCPM2 Flow chart

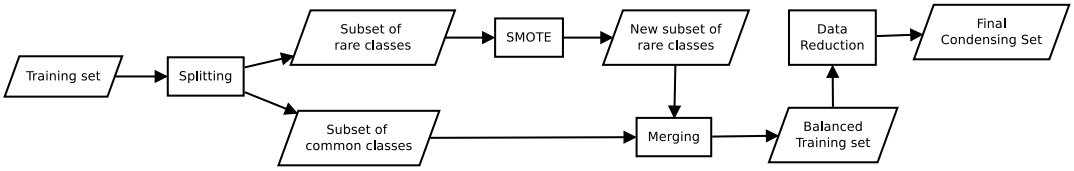

Fig. 3. RCPM-SMOTE Flow chart

A. Rare Class Preservation Method 1

Rare Class Preservation Method 1 (RCPM1) is quite simple. It performs data reduction on the subset of common classes and produces a condensing set for this subset. Then, RCPM1 merges the condensing set with the subset of rare classes. The result is the final condensing set that is used by the k-NN classifier. Figure 1 illustrates the flow chart of RCPM1.

Since data reduction is applied on a set of instances that do not contain the rare classes (fewer decision boundaries exist), RCPM1 collects fewer close-border instances and thus, it achieves higher reduction rates than when data reduction is applied on the complete training dataset. RCPM1 can be applied only on training datasets that contain more than one common classes. This is because PS-condensing and PG algorithms will collect only one prototype when applied on datasets with only one class. Consequently, RCPM1 is inappropriate for binary imbalanced datasets and for imbalanced datasets with only one common class.

B. Rare Class Preservation Method 2

The Rare Class Preservation Method 2 (RCPM2) applies data reduction on the complete training dataset. Next, it replaces the prototypes collected for the rare classes with the instances contained in the original subset of rare classes. Hence, data reduction is applied on the complete datasets where all decision boundaries exist, thus, RCPM2 achieves lower reduction rates than RCPM1. Here, the number of common classes is irrelevant. Therefore, RCPM2 can be applied on either binary training datasets or multi-class training datasets with any number of common classes. Figure 2 presents the flow chart of RCPM2.

C. Rare Class Preservation Method though SMOTE

The Rare Class Preservation Method through SMOTE (RCPM-SMOTE) is based on SMOTE oversampling. Initially, it utilizes SMOTE in order to balance the dataset. Next, it applies data reduction on the resulting dataset. The flow chart is presented in Figure 3. Here, rare classes preservation is achieved through oversampling. Like RCPM2, RCPM-SMOTE is appropriate for both binary and multi-class classification problems and the number of common classes is irrelevant. However, the more the rare classes in the dataset, the more the synthetic instances generated. Therefore, the reduction rates achieved by RCPM-SMOTE depend on the number of rare classes. From this point of view, RCPM-SMOTE is appropriate for binary datasets and for datasets with few rare classes. If the number of rare classes is high, the reduction rates may not be high.

Surely, the preservation level of rare classes depends on the k value used in SMOTE oversampling. If k is large, the synthetic instances will be generated in a larger neighbourhood where instances of other classes probably reside. Thus, a PS-condensing or PG algorithm will collect a sufficient number of prototypes for the rare classes. In contrast, a small k value may result in few prototypes for rare classes. This is because instances of different classes may be not mixed in small neighbourhoods.

IV. Experimental study

A. Experimental setup

To evaluate the performance of the proposed Rare Class Preservation Methods, we applied PS-condensing and PG algorithms on the training datasets of twelve imbalanced

TABLE I
DATASETS DESCRIPTION

Dataset	Size	Attr.	Classes	Rare Classes	A class is rare if it has $\leq$
Avila (AV)	20867	10	12	3	200
Balance (BL)	625	4	3	1	100
CAR	1.728	6	4	2	60
KDD	141481	36	23	12	100
Page-Blocks 1 (PB1)	5473	10	5	3	100
Shuttle (SH1)	57999	9	7	4	150
Yeast (YS)	1484	8	10	6	100
Page-Blocks 2 (PB2)	5472	10	2	1	500
Segment (SG)	2308	18	2	1	300
Shuttle (SH2)	1829	9	2	1	100
Vowel (VW)	988	13	2	1	100
Wine Quality Red (WQR)	1599	11	2	1	100

datasets, distributed by the KEEL repository [9], with and without preprocessing. Table I summarizes on the datasets used. The last column lists the threshold value set for splitting each dataset into the subset of common classes and the subset of rare classes. Seven datasets are multi-class while the rest are binary. RCPM1 cannot be applied on the five binary datasets. Hold out was used to validate the performance. We randomized and normalized all datasets in the $[0, 1]$ range, and then divided them into training and testing sets. We used 67% of data for training purposes and 33% for testing purposes.

For the data reduction stage, two PS-condensing algorithms were used: the well-known CNN-rule [10] and one-pass variation of IB2 [11]. They work as follows: Initially, they move a random instance to the condensing set. Then, for each instance x of the training set, they retrieve the nearest instance y from the condensing set. If x and y belong to different classes, x is moved to the condensing set. CNN-rule makes multiple passes and terminates when there is no movement from the training set to the condensing set. On the other hand, IB2 performs only one pass.

Moreover, we used the PG algorithms RSP3 [12], RHC [13], and AIB2 [14]. RSP3 and RHC try to find homogeneous groups of instances in the training data. Next, we generate a prototype for each group. AIB2 constitutes a PG variation of IB2. It works similar to IB2, but the instances that have the same class with their nearest prototype in the condensing set contribute to the construction of the condensing set by updating their nearest prototype. This way, each prototype lies in the center of the area that it represents.

All algorithms and Rare Class Preservation Methods were implemented in C. We ran each data reduction technique in conjunction with each Rare Class Preservation Method on each dataset, and, we measured Precision, Recall and F-score for each rare class in each dataset as well as Reduction Rate and Accuracy. In the case of RCPM-SMOTE, we used the SMOTE implementation of WEKA [15]. We set $k = 5$ for SMOTE over-sampling, that is a typical setting for SMOTE over-sampling and it is recommended by WEKA. Also, each rare class was over-sampled so that it contained at least as many instances as one of the common classes.

TABLE II
ACCURACY/PRECISION/RECALL/F-SCORE ACHIEVED BY 1-NN CLASSIFIER ON THE ORIGINAL TRAINING DATA

Dataset	Accuracy	Rare Classes	Precision	Recall	F-Score
AV	0.813	Class [1]	1.000	1.000	1.000
		Class [2]	0.828	0.679	0.746
		Class [9]	0.839	0.929	0.882
BL	0.813	Class [1]	0.100	0.050	0.067
CAR	0.859	Class [2]	0.579	0.478	0.524
		Class [3]	0.640	0.696	0.667
KDD	0.997	Class [1]	0.625	0.769	0.690
		Class [2]	1.000	0.500	0.667
		Class [3]	1.000	0.905	0.950
		Class [4]	0.833	0.833	0.833
		Class [6]	1.000	0.714	0.833
		Class [7]	0.000	0.000	0.000
		Class [8]	0.250	0.200	0.222
		Class [12]	0.500	0.500	0.500
		Class [13]	1.000	1.000	1.000
		Class [16]	0.000	0.000	0.000
		Class [22]	1.000	0.833	0.909
PB1	0.954	Class [2]	0.875	0.538	0.666
		Class [3]	0.609	0.737	0.667
		Class [4]	0.524	0.550	0.537
SH1	0.999	Class [1]	1.000	1.000	1.000
		Class [2]	0.964	0.841	0.898
		Class [5]	nan	0.000	nan
		Class [6]	0.500	0.500	0.500
YS	0.475	Class [3]	0.692	0.750	0.720
		Class [4]	0.350	0.389	0.368
		Class [6]	0.455	0.385	0.417
		Class [7]	0.000	0.000	0.000
		Class [8]	0.364	0.800	0.500
		Class [9]	1.000	1.000	1.000
PB2	0.951	positive	0.739	0.764	0.751
SG	0.995	positive	0.988	0.966	0.977
SH2	1.000	positive	1.000	1.000	1.000
VW	1.000	positive	1.000	1.000	1.000
WQR	0.934	positive	0.130	0.167	0.146

B. Experimental results

Table II lists the results obtained by applying the k-NN classifier on the original datasets, i.e., without applying data reduction and Rare Class Preservation Methods. For each data reduction technique, we present a table (Tables III, IV, V and VII) where we evaluate the performance of classification when no rare class preservation is used or when each one of the proposed Rare Class Preservation Methods is applied before the data reduction step.

As expected, the Rare Class Preservation Methods achieved lower reduction rates than those of data reduction without rare classes preservation. This is an absolutely reasonable fact since the instances that belong to rare classes are not reduced. It is worth noting that there are few exceptions where RCPM1 achieved higher reduction rate than that of data reduction without preserving rare classes. For datasets with many rare classes, RCPM-SMOTE achieved the lowest reduction rates. In the case of the YS dataset, RSP3 and CNN with RCPM-SMOTE produced condensing sets that are larger than the original datasets. Thus, the experimental results confirmed our initial thoughts about RCPM-SMOTE. RSP3 achieved the lowest reduction rates. On the other hand, RHC seems to achieve the highest reduction rates. As expected, AIB2 achieved higher reduction rates than IB2.

RSP3 achieved the highest accuracy measurements. It is not

TABLE III
ACCURACY/PRECISION/RECALL/F-SCORE ACHIEVED BY 1-NN THAT USES THE CONDENSING SET BUILT BY CNN-RULE

Datasets	Accuracy				Reduction Rate				Rare Classes	Precision				Recall				F-Score			
	Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE		Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE
AV	0.776	0.772	0.775	0.779	61.671	61.246	60.837	58.723	Class [1]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
									Class [2]	0.764	0.589	0.724	0.727	0.705	0.718	0.705	0.821	0.733	0.647	0.714	0.771
									Class [9]	0.735	0.578	0.643	0.722	0.893	0.929	0.964	0.929	0.806	0.713	0.771	0.813
BL	0.718	0.746	0.751	0.708	67.308	74.038	67.308	43.990	Class [1]	0.200	0.200	0.300	0.148	0.200	0.150	0.150	0.200	0.200	0.171	0.200	0.170
CAR	0.865	0.844	0.861	0.852	74.544	75.847	71.242	69.939	Class [2]	0.571	0.472	0.556	0.552	0.522	0.739	0.652	0.696	0.545	0.576	0.600	0.616
									Class [3]	0.640	0.500	0.594	0.613	0.696	0.957	0.826	0.826	0.667	0.657	0.691	0.704
KDD	0.997	0.985	0.997	0.996	99.060	99.025	98.996	98.457	Class [1]	0.625	0.458	0.588	0.611	0.769	0.846	0.769	0.846	0.690	0.594	0.666	0.710
									Class [2]	1.000	1.000	1.000	1.000	0.500	0.500	0.500	0.500	0.667	0.667	0.667	0.667
									Class [3]	1.000	0.040	1.000	1.000	0.905	0.952	0.952	0.905	0.950	0.077	0.975	0.950
									Class [4]	0.833	0.455	0.714	0.714	0.833	0.833	0.833	0.833	0.833	0.589	0.769	0.769
									Class [6]	1.000	1.000	1.000	1.000	0.714	0.714	0.714	0.857	0.833	0.833	0.833	0.923
									Class [7]	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
									Class [8]	0.250	0.250	0.250	0.250	0.200	0.200	0.200	0.200	0.222	0.222	0.222	0.222
									Class [12]	0.500	0.500	5.000	1.000	0.500	0.500	0.500	0.500	0.500	0.500	0.909	0.667
									Class [13]	1.000	0.333	1.000	1.000	1.000	1.000	0.100	1.000	1.000	0.500	0.182	1.000
									Class [16]	0.000	0.021	0.000	0.083	0.000	0.167	0.000	0.167	0.000	0.037	0.000	0.111
									Class [22]	1.000	1.000	1.000	1.000	0.833	0.833	0.833	0.833	0.909	0.909	0.909	0.909
PB1	0.945	0.839	0.946	0.912	89.200	89.940	87.664	84.501	Class [2]	0.800	0.562	0.800	0.667	0.615	0.692	0.615	0.769	0.695	0.620	0.695	0.714
									Class [3]	0.538	0.389	0.583	0.342	0.737	0.737	0.737	0.684	0.622	0.509	0.651	0.456
									Class [4]	0.438	0.125	0.451	0.262	0.525	0.775	0.575	0.675	0.478	0.215	0.506	0.377
SH1	0.999	0.464	0.990	0.999	99.566	99.472	99.255	99.366	Class [1]	1.000	0.007	0.153	1.000	1.000	1.000	1.000	1.000	1.000	0.014	0.265	1.000
									Class [2]	0.964	0.007	0.368	0.982	0.857	1.000	0.889	0.873	0.907	0.014	0.521	0.924
									Class [5]	nan	0.000	nan	1.000	0.000	0.000	0.000	0.250	nan	0.000	0.000	0.400
									Class [6]	0.500	0.667	0.500	0.067	0.500	1.000	0.500	1.000	0.500	0.800	0.500	0.125
YS	0.448	0.430	0.448	0.426	31.749	31.446	28.514	-13.448	Class [3]	0.800	0.692	0.692	0.889	0.667	0.750	0.750	0.667	0.727	0.720	0.720	0.762
									Class [4]	0.300	0.318	0.350	0.225	0.333	0.389	0.389	0.500	0.316	0.350	0.368	0.310
									Class [6]	0.455	0.455	0.455	0.267	0.385	0.385	0.385	0.308	0.417	0.417	0.417	0.286
									Class [7]	0.000	0.059	0.000	0.021	0.000	0.200	0.000	0.200	0.000	0.091	0.000	0.038
									Class [8]	0.364	0.333	0.364	0.176	0.800	0.800	0.800	0.600	0.500	0.470	0.500	0.272
									Class [9]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PB1	0.947	-	0.936	0.939	90.625	-	83.936	83.032	positive	0.707	-	0.642	0.649	0.787	-	0.775	0.820	0.745	-	0.702	0.725
SG	0.992	-	0.974	0.991	96.684	-	81.925	96.684	positive	0.966	-	0.833	0.955	0.966	-	0.966	0.966	0.966	-	0.895	0.960
SH2	1.000	-	1.000	1.000	99.672	-	94.258	99.590	positive	1.000	-	1.000	1.000	1.000	-	1.000	1.000	1.000	-	1.000	1.000
VW	0.991	-	0.979	0.991	94.529	-	86.778	93.769	positive	0.923	-	0.781	0.923	0.960	-	1.000	0.960	0.941	-	0.877	0.941
WQR	0.906	-	0.906	0.857	85.336	-	85.178	65.291	positive	0.079	-	0.079	0.097	0.167	-	0.167	0.389	0.107	-	0.107	0.155

TABLE IV
ACCURACY/PRECISION/RECALL/F-SCORE ACHIEVED BY 1-NN THAT USES THE CONDENSING SET BUILT BY IB2

Datasets	Accuracy				Reduction Rate				Rare Classes	Precision				Recall				F-Score			
	Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE		Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE
AV	0.756	0.753	0.754	0.756	67.386	66.868	66.537	64.755	Class [1]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
									Class [2]	0.707	0.564	0.629	0.650	0.679	0.731	0.718	0.833	0.693	0.637	0.671	0.730
									Class [9]	0.735	0.565	0.643	0.703	0.893	0.929	0.964	0.929	0.806	0.703	0.771	0.800
BL	0.703	0.737	0.703	0.670	71.875	78.125	71.635	55.769	Class [1]	0.190	0.185	0.176	0.125	0.200	0.250	0.150	0.200	0.195	0.213	0.162	0.154
CAR	0.861	0.812	0.851	0.847	80.626	80.278	76.455	76.021	Class [2]	0.522	0.400	0.455	0.516	0.522	0.783	0.652	0.696	0.522	0.783	0.652	0.696
									Class [3]	0.679	0.468	0.553	0.625	0.826	0.957	0.913	0.870	0.826	0.957	0.913	0.870
KDD	0.995	0.984	0.995	0.987	99.198	99.154	99.133	98.834	Class [1]	0.471	0.458	0.476	0.478	0.615	0.846	0.769	0.846	0.533	0.594	0.588	0.611
									Class [2]	1.000	1.000	1.000	0.500	0.500	0.500	0.500	0.500	0.667	0.667	0.667	0.500
									Class [3]	1.000	0.040	1.000	1.000	0.905	0.952	0.952	0.905	0.950	0.077	0.975	0.950
									Class [4]	0.500	0.455	0.625	0.500	0.667	0.833	0.833	0.667	0.572	0.589	0.714	0.572
									Class [6]	1.000	1.000	1.000	1.000	0.714	0.714	0.714	0.857	0.833	0.833	0.833	0.923
									Class [7]	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
									Class [8]	0.250	0.250	0.250	0.250	0.200	0.200	0.200	0.200	0.222	0.222	0.222	0.222
									Class [12]	0.500	0.500	0.500	1.000	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.667
									Class [13]	1.000	0.333	1.000	1.000	1.000	1.000	0.100	1.000	1.000	0.500	0.182	1.000
									Class [16]	0.000	0.021	0.000	0.010	0.000	0.167	0.000	0.167	0.000	0.037	0.000	0.019
									Class [22]	1.000	1.000	1.000	0.833	0.833	0.833	0.833	0.833	0.909	0.909	0.909	0.833
PB1	0.925	0.820	0.926	0.739	91.530	91.091	89.419	74.589	Class [2]	0.600	0.529	0.615	0.600	0.462	0.692	0.615	0.692	0.522	0.600	0.615	0.643
									Class [3]	0.520	0.341	0.600	0.259	0.684	0.737	0.789	0.789	0.591	0.466	0.682	0.390
									Class [4]	0.344	0.123	0.342	0.078	0.525	0.775	0.625	0.700	0.416	0.212	0.442	0.140
SH1	0.997	0.464	0.988	0.948	99.604	99.478	99.289	99.501	Class [1]	0.241	0.007	0.099	0.027	1.000	1.000	1.000	1.000	0.388	0.014	0.180	0.053
									Class [2]	0.966	0.007	0.377	0.098	0.889	1.000	0.921	0.905	0.926	0.014	0.535	0.177
									Class [5]	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.250	0.000	0.000	0.000	0.400
									Class [6]	0.333	0.667	0.500	0.667	0.250	1.000	0.500	1.000	0.286	0.800	0.500	0.800
YS	0.432	0.412	0.434	0.394	43.883	42.568	39.838	10.920	Class [3]	0.750	0.692	0.692	0.875	0.500	0.750	0.750	0.583	0.600	0.720	0.720	0.700
									Class [4]	0.316	0.280	0.318	0.237	0.333	0.389	0.389	0.500	0.324	0.326	0.350	0.322
									Class [6]	0.417	0.455	0.455	0.250	0.385	0.385	0.385	0.380	0.400	0.417	0.417	0.302
									Class [7]	0.000	0.048	0.000	0.017	0.000	0.200	0.000	0.200	0.000	0.077	0.000	0.031
									Class [8]	0.500	0.333	0.364	0.150	0.600	0.800	0.800	0.600	0.545	0.470	0.500	0.240
									Class [9]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PB2	0.925	-	0.917	0.858	92.873	-	85.307	88.734	positive	0.598	-	0.549	0.396	0.702	-	0.815	0.865	0.646	-	0.656	0.543
SG	0.988	-	0.951	0.986	97.594	-	82.575	97.529	positive	0.934	-	0.708	0.914	0.966	-	0.966	0.966	0.950	-	0.817	0.939
SH2	1.000	-	1.000	1.000	99.672	-	94.258	99.590	positive	1.000	-	1.000	1.000	1.000	-	1.000	1.000	1.000	-	1.000	1.000
VW	0.982	-	0.964	0.982	95.137	-	87.234	94.883	positive	0.828	-	0.676	0.828	0.960	-	1.000	0.960	0.889	-	0.807	0.889
WQR	0.859	-	0.856	0.659	88.743	-	88.462	81.426	positive	0.062	-	0.060	0.044	0.222	-	0.222	0.444	0.097	-	0.094	0.080

TABLE V
ACCURACY/PRECISION/RECALL/F-SCORE ACHIEVED BY 1-NN THAT USES THE CONDENSING SET BUILT BY RSP3

Datasets	Accuracy				Reduction Rate				Rare Classes	Precision				Recall				F-Score			
	Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE		Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE
AV	0.792	0.794	0.792	0.791	47.581	47.150	47.085	38.358	Class [1]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
									Class [2]	0.788	0.743	0.783	0.713	0.667	0.705	0.692	0.795	0.722	0.724	0.735	0.752
									Class [9]	0.818	0.711	0.794	0.844	0.964	0.964	0.964	0.964	0.885	0.818	0.871	0.900
BL	0.737	0.746	0.764	0.703	61.058	67.308	60.817	25.240	Class [1]	0.059	0.077	0.077	0.154	0.050	0.050	0.050	0.200	0.054	0.061	0.061	0.174
CAR	0.856	0.849	0.849	0.870	67.420	69.244	65.595	56.773	Class [2]	0.667	0.515	0.591	0.652	0.522	0.739	0.565	0.652	0.586	0.607	0.578	0.652
									Class [3]	0.609	0.514	0.533	0.680	0.609	0.783	0.696	0.739	0.609	0.621	0.604	0.708
KDD	0.996	0.994	0.996	0.996	99.118	98.432	98.411	97.211	Class [1]	0.524	0.370	0.524	0.647	0.846	0.769	0.846	0.846	0.647	0.500	0.647	0.733
									Class [2]	0.667	1.000	0.667	0.667	0.500	0.500	0.500	0.500	0.572	0.667	0.572	0.572
									Class [3]	1.000	0.370	1.000	0.950	0.905	0.952	0.905	0.905	0.950	0.533	0.950	0.927
									Class [4]	0.714	0.455	0.714	0.714	0.833	0.833	0.833	0.833	0.769	0.589	0.769	0.769
									Class [6]	1.000	1.000	1.000	1.000	0.857	1.000	1.000	0.857	0.923	1.000	1.000	0.923
									Class [7]	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
									Class [8]	0.250	0.250	0.250	0.250	0.200	0.200	0.200	0.200	0.222	0.222	0.222	0.222
									Class [12]	0.500	0.500	0.500	1.000	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.667
									Class [13]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
									Class [16]	0.000	0.031	0.000	0.100	0.000	0.167	0.000	0.167	0.000	0.052	0.000	0.125
									Class [22]	1.000	1.000	1.000	1.000	0.833	0.833	0.833	0.833	0.909	0.909	0.909	0.909
PB1	0.941	0.895	0.942	0.911	86.157	88.158	84.814	49.260	Class [2]	0.800	0.769	0.818	0.733	0.615	0.769	0.692	0.846	0.695	0.769	0.750	0.785
									Class [3]	0.524	0.412	0.636	0.389	0.579	0.737	0.737	0.737	0.550	0.529	0.683	0.509
									Class [4]	0.458	0.200	0.434	0.261	0.550	0.725	0.575	0.725	0.500	0.314	0.495	0.384
SH1	0.996	0.587	0.995	0.995	99.312	99.403	99.059	98.849	Class [1]	0.684	0.007	0.565	0.867	1.000	1.000	1.000	1.000	0.812	0.014	0.722	0.929
									Class [2]	0.859	0.010	0.814	0.465	0.873	1.000	0.905	0.937	0.866	0.020	0.857	0.622
									Class [5]	nan	0.000	nan	nan	0.000	0.000	0.000	0.000	nan	0.000	0.000	0.000
									Class [6]	0.500	0.667	0.500	0.667	0.500	1.000	0.500	1.000	0.500	0.800	0.500	0.800
YS	0.481	0.463	0.477	0.432	26.997	26.694	24.469	-45.602	Class [3]	0.750	0.750	0.692	0.750	0.750	0.750	0.750	0.750	0.750	0.750	0.720	0.750
									Class [4]	0.368	0.318	0.350	0.235	0.389	0.389	0.389	0.444	0.378	0.350	0.368	0.307
									Class [6]	0.500	0.417	0.455	0.235	0.462	0.385	0.385	0.308	0.480	0.400	0.417	0.267
									Class [7]	0.167	0.000	0.167	0.042	0.200	0.000	0.200	0.400	0.182	0.000	0.000	0.076
									Class [8]	0.364	0.308	0.364	0.143	0.800	0.800	0.800	0.600	0.500	0.445	0.500	0.231
									Class [9]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PB2	0.940	-	0.932	0.933	87.747	-	81.880	76.974	positive	0.657	-	0.619	0.618	0.798	-	0.787	0.809	0.721	-	0.693	0.701
SG	0.988	-	0.975	0.988	94.083	-	80.429	93.888	positive	0.934	-	0.842	0.934	0.966	-	0.966	0.966	0.950	-	0.900	0.950
SH2	1.000	-	1.000	0.998	99.590	-	94.258	99.508	positive	1.000	-	1.000	0.982	1.000	-	1.000	1.000	1.000	-	1.000	0.991
VW	0.985	-	0.982	0.988	90.578	-	83.739	90.729	positive	0.833	-	0.806	0.862	1.000	-	1.000	1.000	0.909	-	0.893	0.926
WQR	0.889	-	0.887	0.854	84.897	-	84.803	46.154	positive	0.098	-	0.096	0.083	0.278	-	0.278	0.333	0.145	-	0.143	0.133

TABLE VI
ACCURACY/PRECISION/RECALL/F-SCORE ACHIEVED BY 1-NN THAT USES THE CONDENSING SET BUILT BY AIB2

Datasets	Accuracy				Reduction Rate				Rare Classes	Precision				Recall				F-Score			
	Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE		Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE
AV	0.767	0.764	0.767	0.765	69.104	68.349	68.227	66.846	Class [1]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
									Class [2]	0.754	0.667	0.718	0.645	0.667	0.718	0.718	0.769	0.708	0.692	0.718	0.702
									Class [9]	0.774	0.683	0.692	0.812	0.857	1.000	0.964	0.929	0.813	0.812	0.806	0.867
BL	0.785	0.718	0.766	0.770	73.077	81.971	72.115	63.462	Class [1]	0.000	0.194	0.143	0.185	0.000	0.300	0.150	0.250	0.000	0.236	0.146	0.213
CAR	0.873	0.816	0.851	0.863	83.406	82.103	78.454	81.060	Class [2]	0.667	0.353	0.472	0.533	0.609	0.783	0.739	0.696	0.637	0.487	0.576	0.604
									Class [3]	0.680	0.488	0.556	0.586	0.739	0.913	0.870	0.739	0.708	0.636	0.678	0.654
KDD	0.994	0.992	0.994	0.992	99.104	99.067	99.035	98.686	Class [1]	0.474	0.414	0.407	0.550	0.692	0.923	0.846	0.846	0.563	0.572	0.550	0.667
									Class [2]	0.667	0.667	0.667	0.667	0.500	0.500	0.500	0.500	0.572	0.572	0.572	0.572
									Class [3]	0.950	0.299	0.870	0.952	0.905	0.952	0.952	0.952	0.927	0.455	0.909	0.952
									Class [4]	0.714	0.455	0.714	0.714	0.833	0.833	0.833	0.833	0.769	0.589	0.769	0.769
									Class [6]	1.000	1.000	1.000	1.000	0.857	1.000	1.000	1.000	0.923	1.000	1.000	1.000
									Class [7]	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
									Class [8]	0.250	0.250	0.250	0.250	0.200	0.200	0.200	0.200	0.222	0.222	0.222	0.222
									Class [12]	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
									Class [13]	1.000	0.333	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	1.000	1.000
									Class [16]	0.000	0.017	0.000	0.000	0.000	0.167	0.000	0.000	0.000	0.031	0.000	0.000
									Class [22]	1.000	1.000	1.000	0.714	0.833	0.833	0.833	0.833	0.909	0.909	0.909	0.769
PB1	0.922	0.885	0.921	0.808	91.913	91.283	89.446	49.260	Class [2]	0.875	0.692	0.800	0.769	0.538	0.692	0.615	0.769	0.666	0.692	0.695	0.769
									Class [3]	0.609	0.593	0.615	0.309	0.737	0.842	0.842	0.895	0.667	0.696	0.711	0.459
									Class [4]	0.388	0.370	0.382	0.129	0.650	0.750	0.725	0.800	0.486	0.496	0.500	0.222
SH1	0.990	0.679	0.986	0.982	99.563	99.457	99.250	99.421	Class [1]	0.078	0.011	0.075	0.075	1.000	1.000	1.000	1.000	0.145	0.022	0.140	0.140
									Class [2]	0.812	0.012	0.401	0.261	0.889	1.000	0.937	0.921	0.849	0.024	0.562	0.407
									Class [5]	0.000	0.000	0.000	nan	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
									Class [6]	0.500	0.667	0.500	0.667	0.500	1.000	0.500	1.000	0.500	0.800	0.500	0.800
YS	0.469	0.451	0.463	0.412	45.501	44.388	40.950	13.852	Class [3]	0.700	0.692	0.692	0.875	0.583	0.750	0.750	0.583	0.636	0.720	0.720	0.700
									Class [4]	0.375	0.286	0.308	0.229	0.500	0.444	0.444	0.444	0.429	0.348	0.364	0.302
									Class [6]	0.500	0.417	0.417	0.263	0.462	0.385	0.385	0.385	0.480	0.400	0.400	0.313
									Class [7]	0.000	0.050	0.000	0.019	0.000	0.200	0.000	0.200	0.000	0.080	0.000	0.035
									Class [8]	0.429	0.308	0.286	0.150	0.600	0.800	0.800	0.600	0.500	0.445	0.421	0.240
									Class [9]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PB2	0.921	-	0.922	0.889	92.708	-	85.280	88.816	positive	0.576	-	0.566	0.463	0.702	-	0.843	0.843	0.633	-	0.677	0.598
SG	0.982	-	0.948	0.981	97.854	-	82.705	97.724	positive	0.885	-	0.688	0.876	0.966	-	1.000	0.966	0.924	-	0.815	0.919
SH2	1.000	-	0.997	1.000	99.754	-	94.340	99.754	positive	1.000	-	0.965	1.000	1.000	-	1.000	1.000	1.000	-	0.982	1.000
VW	0.997	-	0.997	0.997	95.745	-	87.234	95.259	positive	0.962		0.962	0.962	1.000		1.000	1.000	0.981	-	0.981	0.981
WQR	0.865	-	0.850	0.777	89.587	-	89.024	84.240	positive	0.078	-	0.081	0.068	0.278	-	0.333	0.444	0.122	-	0.130	0.118

TABLE VII
ACCURACY/PRECISION/RECALL/F-SCORE ACHIEVED BY 1-NN THAT USES THE CONDENSING SET BUILT BY RHC

Datasets	Accuracy				Reduction Rate				Rare Classes	Precision				Recall				F-Score			
	Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE		Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE	Original	RCPM1	RCPM2	RCPM-SMOTE
AV	0.759	0.764	0.760	0.759	70.175	69.758	69.334	66.933	Class [1]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
									Class [2]	0.738	0.585	0.688	0.686	0.615	0.705	0.705	0.756	0.671	0.639	0.696	0.719
									Class [9]	0.800	0.684	0.737	0.833	0.857	0.929	1.000	0.893	0.828	0.788	0.849	0.862
BL	0.732	0.713	0.742	0.722	79.087	81.490	76.202	53.606	Class [1]	0.091	0.156	0.130	0.100	0.100	0.250	0.150	0.100	0.095	0.192	0.139	0.100
CAR	0.833	0.674	0.806	0.818	85.491	83.753	80.799	80.712	Class [2]	0.556	0.260	0.425	0.607	0.652	0.826	0.736	0.739	0.600	0.396	0.539	0.667
									Class [3]	0.654	0.389	0.500	0.667	0.739	0.913	0.913	0.696	0.694	0.546	0.646	0.681
KDD	0.995	0.993	0.994	0.994	99.118	99.063	99.054	98.598	Class [1]	0.500	0.324	0.588	0.611	0.538	0.846	0.769	0.864	0.518	0.469	0.666	0.716
									Class [2]	0.667	0.667	0.667	1.000	0.500	0.500	0.500	0.500	0.572	0.572	0.572	0.667
									Class [3]	0.952	0.357	0.800	0.950	0.952	0.952	0.952	0.905	0.952	0.519	0.869	0.927
									Class [4]	0.714	0.714	0.714	0.625	0.833	0.833	0.833	0.833	0.769	0.769	0.769	0.714
									Class [6]	1.000	1.000	1.000	1.000	0.714	1.000	0.714	0.714	0.833	1.000	0.833	0.833
									Class [7]	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
									Class [8]	0.250	0.250	0.250	0.250	0.200	0.200	0.200	0.200	0.222	0.222	0.222	0.222
									Class [12]	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
									Class [13]	1.000	0.333	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	1.000	1.000
									Class [16]	0.000	0.020	0.000	0.000	0.000	0.167	0.000	0.000	0.000	0.036	0.000	0.000
									Class [22]	1.000	0.714	1.000	0.714	0.833	0.833	0.833	0.833	0.909	0.769	0.909	0.769
PB1	0.931	0.868	0.928	0.913	90.680	90.707	88.651	77.961	Class [2]	0.615	0.321	0.615	0.867	0.615	0.692	0.615	1.000	0.615	0.439	0.615	0.929
									Class [3]	0.577	0.410	0.577	0.457	0.789	0.842	0.789	0.842	0.667	0.551	0.667	0.592
									Class [4]	0.391	0.183	0.357	0.257	0.625	0.775	0.625	0.725	0.481	0.296	0.454	0.379
SH1	0.997	0.460	0.988	0.997	99.573	99.496	99.294	99.426	Class [1]	0.929	0.006	0.542	0.382	1.000	1.000	1.000	1.000	0.963	0.012	0.703	0.553
									Class [2]	0.609	0.008	0.214	0.887	0.889	1.000	0.889	0.837	0.723	0.016	0.345	0.861
									Class [5]	nan	0.000	nan	nan	0.000	0.000	0.000	0.000	nan	0.000	0.000	0.000
									Class [6]	0.500	0.667	0.500	0.667	0.500	1.000	0.500	1.000	0.500	0.800	0.500	0.800
YS	0.420	0.438	0.416	0.416	48.534	46.714	44.186	11.426	Class [3]	0.800	0.750	0.692	0.889	0.667	0.750	0.750	0.667	0.727	0.750	0.720	0.762
									Class [4]	0.348	0.250	0.350	0.229	0.444	0.389	0.389	0.444	0.390	0.304	0.368	0.302
									Class [6]	0.500	0.333	0.417	0.333	0.462	0.385	0.385	0.385	0.480	0.357	0.400	0.357
									Class [7]	0.000	0.056	0.000	0.000	0.000	0.200	0.000	0.000	0.000	0.088	0.000	0.000
									Class [8]	0.364	0.267	0.364	0.176	0.800	0.800	0.800	0.600	0.500	0.400	0.500	0.272
									Class [9]	1.000	1.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000
PB2	0.930	-	0.922	0.920	91.859	-	84.923	85.307	positive	0.615	-	0.571	0.562	0.764	-	0.815	0.809	0.681	-	0.672	0.663
SG	0.987	-	0.962	0.984	98.049	-	83.030	97.984	positive	0.924	-	0.761	0.904	0.966	-	0.977	0.966	0.945	-	0.856	0.934
SH2	1.000	-	0.997	1.000	99.754	-	94.340	99.754	positive	1.000	-	0.965	1.000	1.000	-	1.000	1.000	1.000	-	0.982	1.000
VW	1.000	-	0.985	1.000	97.112	-	88.298	96.960	positive	1.000	-	0.833	1.000	1.000	-	1.000	1.000	1.000	-	0.909	1.000
WQR	0.856	-	0.856	0.824	92.026	-	91.651	76.642	positive	0.117	-	0.117	0.068	0.500	-	0.500	0.333	0.190	-	0.190	0.113

TABLE VIII
COMPARISONS

Methods	RR			Acc.			Precision			Recall			F-Score		
	W	L	T	W	L	T	W	L	T	W	L	T	W	L	T
Original vs RCPM1	5	2	0	7	0	0	6	1	0	0	7	0	6	1	0
Original vs RCPM2	12	0	0	8	1	3	8	3	1	2	9	1	9	3	0
Original vs RCPM-SMOTE	12	0	0	9	1	2	8	4	0	0	9	3	7	5	0
RCPM1 vs RCPM2	7	0	0	0	7	0	0	7	0	5	2	0	1	6	0
RCPM1 vs RCPM-SMOTE	7	0	0	3	4	0	2	5	0	5	2	0	2	5	0
RCPM2 vs RCPM-SMOTE	7	5	0	7	4	1	6	6	0	4	7	1	4	8	0

clear which of the other algorithms is more accurate in terms of accuracy. Rare Class Preservation Methods achieved lower rare classes precision and overall accuracy measurements than the data reduction without preserving rare classes. Rare classes precision achieved by RCPM-SMOTE seems to be similar to that of RCPM2. Both achieved higher precision than RCPM1. In contrast, all Rare Class Preservation Methods seem to achieve higher recall than the data reduction without preserving rare classes. RCPM1 is the method with the greatest improvements in terms of recall. RCPM2 and RCPM-SMOTE also improve recall, but the improvements are not so high as the improvements achieved by RCPM1. It is not clear which data reduction technique is better than the other in terms of recall and precision. It depends on the dataset used. For instance, RSP3 with RCPM2 achieved much higher precision on SH than the other methods. Also, in some cases, AIB2 and RHC seem to achieve higher recall than the other three algorithms. Unfortunately, F-score measurements show that the decreases in precision are higher than the increases in recall. However, there are some cases with satisfactory improvements in terms of F-Score. For example, for the CAR dataset, all three methods produce very good results. Moreover, RCPM1 and RCPM-SMOTE produce good F-score results in some datasets as well.

For each dataset used, a value for each Rare Class Preservation Method is computed by averaging the measurements achieved by each PS-condensing and PG algorithm. Then, we count the wins, the losses and the ties for each measure used. Table VIII compares the methods in pairs. It is seen that, in terms of reduction rate, accuracy and precision, data reduction without preserving rare classes is more effective. In contrast, RCPM1, RCPM2 and RCPM-SMOTE improve recall.

V. CONCLUSION

Although rare classes are often more significant than common classes, data reduction may even eliminate the training instances that belong to rare classes. Therefore, they are inadequate for imbalanced training data. This paper presented methods for rare classes preservation. They are applied in conjunction with either a PS-condensing or a PG algorithm. The first method is called RCPM1 and preserves the rare classes by applying data reduction only on instances that belong to common classes. The second method is called RCPM2 and applies data reduction on the complete training datasets and then all the instances that belong to rare classes are placed back in the condensing set replacing the rare class prototypes. In both RCPM1 and RCPM2, instances that belong to rare classes remain intact in the condensing set. Both RCPM1 and RCPM2 avoid costly under-sampling or over-sampling that deal with class imbalances. Furthermore, an extra rare class preservation method that utilizes SMOTE oversampling, and is called RCPM-SMOTE, was developed. Initially, it applies SMOTE in order to balance the training dataset and then it applies data reduction. RCPM1 can be applied only on

multi-class datasets with more than one common classes while RCPM2 and RCPM-SMOTE may be applied on any classification dataset. The three methods were tested by conducting experiments on twelve imbalanced datasets. The experimental measurements obtained showed that RCPM1, RCPM2 and RCPM-SMOTE improve recall measurements while reduction rates remain high. In many problems with imbalanced data, high recall is what is required (e.g., earthquake prediction, extreme weather conditions, etc.). Therefore, the Rare Class Preservation Methods presented in this paper can be applied in such domains where high recall is required and a slightly lower precision is acceptable.

REFERENCES

[1] Salvador Garcia, Joaquin Derrac, Jose Cano, and Francisco Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(3):417–435, March 2012.

[2] Isaac Triguero, Joaqun Derrac, Salvador Garcia, and Francisco Herrera. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *Trans. Sys. Man Cyber Part C*, 42(1):86–100, January 2012.

[3] Haibo He and E.A. Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, Sept 2009.

[4] Haibo He and Yunqian Ma. *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley-IEEE Press, 1st edition, 2013.

[5] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321357, June 2002.

[6] Amalia Luque, Alejandro Carrasco, Alejandro Martn, and Ana de las Heras. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, 91:216–231, 2019.

[7] V. S. Spelmen and R. Porkodi. A review on handling imbalanced data. In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pages 1–11, 2018.

[8] A. Gosain and S. Sardana. Handling class imbalance problem using oversampling techniques: A review. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 79–85, 2017.

[9] Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, and Salvador García. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2011.

[10] P E Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516, 1968.

[11] David W. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *Int. J. Man-Mach. Stud.*, 36(2):267–287, February 1992.

[12] José Salvador Sánchez. High training set size reduction by space partitioning and prototype abstraction. *Pattern Recognition*, 37(7):1561–1564, 2004.

[13] Stefanos Ougiaroglou and Georgios Evangelidis. Rhc: Non-parametric cluster-based data reduction for efficient k-nn classification. *Pattern Anal. Appl.*, 19(1):93109, February 2016.

[14] Stefanos Ougiaroglou and Georgios Evangelidis. AIB2: An abstraction data reduction technique based on ib2. In *Proceedings of the 6th Balkan Conference in Informatics*, BCI '13, pages 13–16, New York, NY, USA, 2013. ACM.

[15] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.