Exhaustive Search for Weighted Ensemble Classifiers to Improve Performance on Imbalanced Dataset

Tri Handhika, Ahmad Sabri, and Murni

Abstract We compare performance of six single classifiers trained on German credit dataset, an imbalanced dataset of 1000 instances with binary-valued dependent variable. To improve the performance, we consider resampling the dataset and ensembling the classifiers. The benchmarks are taken from the best performance among six considered classifiers. Resampling the dataset includes oversampling and undersampling. The performance of ensemble classifiers are then analyzed and examined. The experimental results provide three benchmarks, i.e. SVM trained on plain dataset, NB trained on plain dataset, and SVM trained on undersampled dataset. Furthermore, ensemble of kNN, LDA and SVM outperforms the first benchmark for all metrics used in this research, i.e. recall 92.71%, precision 79.14%, F1 84.73%, AUC 79.96%, and accuracy 76.88%. The ensemble of LR, SVM and NB and the ensemble of LDA, SVM, and NB outperforms the second and third benchmark, respectively.

Keywords Ensemble classifiers · Imbalanced dataset · Resampling

This document is the results of the research project funded by Direktorat Riset dan Pengabdian Masyarakat Deputi Bidang Penguatan Riset dan Pengembangan Kementerian Riset dan Teknologi/Badan Riset dan Inovasi Nasional (Grant No. 234/SP2H/LT/DRPM/2021).

T. Handhika (✉) · Murni
Centre for Computational Mathematics Studies, Gunadarma University, Depok 16424, Indonesia
e-mail: trihandika@staff.gunadarma.ac.id

Murni
e-mail: murnipskm@staff.gunadarma.ac.id

A. Sabri
Department of Informatics, Gunadarma University, Depok 16424, Indonesia
e-mail: sabri@staff.gunadarma.ac.id

N. H. T. Dang et al. (eds.), *Artificial Intelligence in Data and Big Data Processing*,
Lecture Notes on Data Engineering and Communications Technologies 124,
https://doi.org/10.1007/978-3-030-97610-1_3

1 Introduction

Classifiers trained on an imbalanced dataset might have low performance. Because of unequal distribution of classes in the dataset, the classifiers might not have enough knowledge to identify the minority class. This unequal distribution of the classes happens in dataset such as the people having a certain sickness compared to those who are healthy, the number of non eligible credit applicants compared to those who are eligible applicants, etc.

By oversampling the minority, undersampling the majority, or combination of both oversampling and undersampling, the balanced dataset can be achieved. Resampling is done by creating synthetic instances for the minority, and/or removing certain instances of the majority with appropriate rules or randomly chosen. According to [1], combining oversampling and undersampling gives better accuracy than plain undersampling.

In this research, we apply weighted ensemble voting classifier on an imbalanced dataset and search exhaustively for weight sets that improves performance compared to those of six considered single classifiers. We test this method on the German credit dataset [2], an imbalanced dataset consisting of 1000 instances of credit applicants with 20 recorded features, where the dependent variables are classification whether an applicant is eligible or not.

The benchmarks are taken from the best performance among six single classifiers, that are LR, kNN, LDA, DT, Gaussian NB, and SVM, on the same dataset. The goal is to obtain ensemble models that perform better than those single models in terms of recall, precision, F1 measurement, Area Under Curve (AUC), and accuracy.

The search for weight set that gives maximum performance adopts the generating algorithm given in [3, 4]. It is done by generating a combination of weights, continued by fitting ensemble classifier with those weights to the dataset. This procedure runs recursively until all possible weight sets are considered. We call this procedure as *exhaustive weight generating algorithm*.

2 Literature Review

In terms of classifiers, research [5] revealed that the Support Vector Machines (SVM) model outperforms the Artificial Neural Network (ANN) model in terms of generalization (i.e., giving better performance on the test set). This result was then confirmed in [6], that SVM outperforms Logistic Regression (LR) and ANN on their dataset. In [7] was shown that SVM performs better than LR, Linear Discriminant Analysis (LDA) and k-nearest neighbors (kNN), while [8] shows that ANN is the best among five other classifiers: LDA, LR, kNN [9], Naïve Bayes (NB) and Decision Tree (DT) [10, 11], to predict the probability of credit default.

To improve classifier performance, there are several approaches such as by ensembling several classifiers [12–14], by feature selection [15, 16], or combination of both

ensembling and feature selection [14]. In ensembled classifiers, the decisions are combined in some way, typically by weighted or unweighted voting, when classifying new examples [17]. In most cases the ensemble classifiers produce more accurate predictions than the base classifiers [18].

3 Methodology

3.1 General Approach

Our general approach is divided into two steps (see Fig. 1). The first step is to set the benchmarks based on the best metric performance of single classifiers. The second step is to conduct weighted ensemble voting classifier that consists of the best

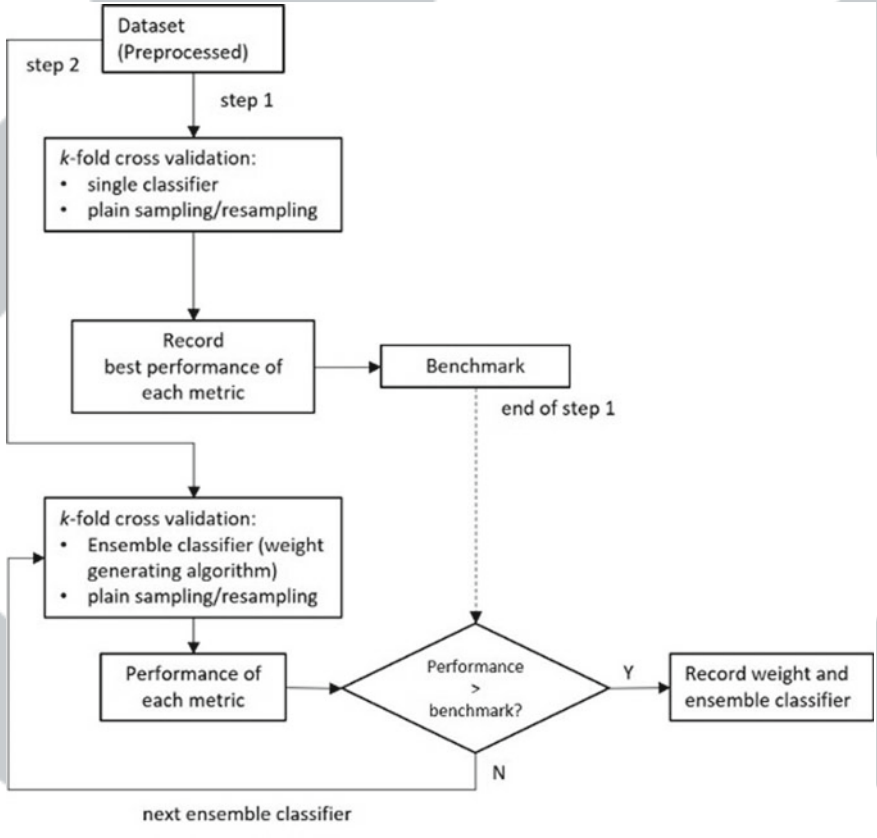

Fig. 1 General scheme

classifiers based on the previous step (in this work, we employ 2 or 3 classifiers). The goal is to obtain ensemble classifiers along with their optimal weight.

3.2 Evaluating the Model

The dataset is preprocessed by applying one-hot encoding to all categorical columns and standardizing all numerical columns. In the dependent column, the labels 1 and 2, each represents "good" and "bad", are transformed to 1 and 0, respectively. In this context, the value 1 is considered as positive case, while 0 as negative case.

Based on such labeling, it follows that the false positives (FP) case, i.e., actual bad applicant predicted as good applicant, is more harmful than the opposite case (false negatives (FN)). This happens more frequent if the classifier is trained on an imbalanced dataset. Thus, beside accuracy, we consider *precision* to measure the proportion of true positives (TP) over all positives predictions. The precision is obtained by:

$$\text{precision} = \frac{TP}{TP + FP}. \tag{1}$$

A classifier with high precision indicates low occurrences of FP cases.

We also consider *recall*, the proportion of TP over actual positives, i.e., the proportion of actual positives predicted as positives. It is obtained by:

$$\text{recall} = \frac{TP}{TP + FN}. \tag{2}$$

Recall is less harmful, but low recall indicates high FN, which means many good applicants classified as bad. This might cause the company loses the opportunity to get higher revenue. To convey the balance between the precision and the recall we can use F1 score which is formulated as follows:

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{3}$$

The accuracy measurement is defined as the percentage of correctly classified instances which is obtained by:

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN}, \tag{4}$$

where TN represents the number of true negatives.

The measurement of AUC indicates how skilled the classifier is. The higher the score, the more ability for the classifier to predict the correct class. The score 0.5 means it has no skill to differentiate between classes. The score < 0.5 means the

classifier predicts against the correct class, i.e., actual 1 predicted as 0 and vice versa.

Due to the imbalancement of the dataset, our investigation involves oversampling the minority and undersampling majority to obtain a more balanced data. The following sampling scheme is applied:

1. Plain sampling, using the original dataset with 1000 instances where the ratio majority: minority = 7:3.
2. Oversampling 100%. This is to achieve 1:1 ratio between majority and minority. By generating synthetically 400 additional minority instances. The resampled dataset contains 1400 instances.
3. Undersampling 100%. This is an alternative way to achieve 1:1 ratio between majority and minority. It is done by reducing the number of majority instances to the same as that of minority. The resampled dataset contains 600 instances.
4. Oversampling 50%. The resampled dataset contains 1050 instances.

We consider six single classifiers as mentioned earlier. Oversampling and undersampling is done by Synthetic Minority Oversampling Technique (SMOTE) [1] and by random technique, respectively.

Evaluation uses repeated stratified 5-fold cross-validation. Performance measured are recall, precision, F1, AUC and accuracy. Oversampling and undersampling are only applied to the folds that contains training data. Figure 2 shows the methods for evaluating classifier.

3.3 The Exhaustive Weight Generating Algorithm

The ensemble voting classifier is expressed by $\sum_{i=1}^{n} w_i C_i$ where: C_i is the binary label in $\{0, 1\}$ predicted by $i-$th classifier, and w_i is an integer belongs to $\{1, 2, \cdots, n\}$ that represents the weight of $i-$th classifier. The threshold is $T = \frac{1}{2}\sum_{i=1}^{n} w_i$, which means the predicted label is 1 if $\sum_{i=1}^{n} w_i C_i \geq T$, and 0 otherwise. We denote by $(C_1, C_2, \ldots, C_n)$ the ensemble of n classifiers $C_1, C_2, \cdots, C_n$. The sequence of weights that correspond to $(C_1, C_2, \cdots, C_n)$ is denoted by $n-$tuple $(w_1, w_2, \ldots, w_n)$ where w_i is the weight of C_i.

The weight sequences are generated exhaustively by recursive generating algorithm called *weight*, as shown in Algorithm 1. The algorithm is adapted from that of [3, 4], and it generates the sequence exactly once for every possible combination. The recursive scheme is preferred rather than using ordinary fixed block loop (e.g. for loop) because of its ability to generate sequences of any length depending to the number of classifiers ensembled, while using fixed block loop needs additional physical loop block for any additional weight.

For convenience, the initial weight is set to $W = (w_1, w_2, \ldots, w_n) = (0, 0, \ldots, 0)$. Every recursive call in level k, weight(k), assigns a weight value for w_k. Recursive calls are initiated by first call weight(1). When $k = n$, a weight sequence of length n is generated. Then the ensemble classifiers with current weight sequence is fitted to

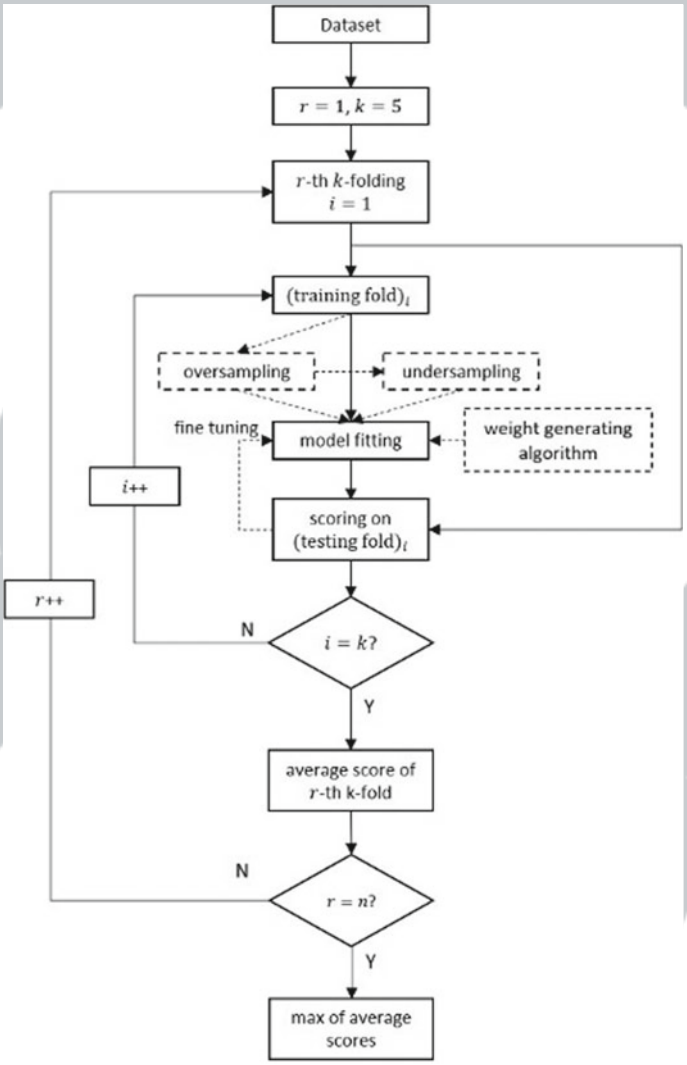

Fig. 2 The flow for evaluating the model

the training data and validated by testing data. This process goes continuously until all n^n possible sets are generated. The weight sets that give best performance are recorded. Figure 3 shows generating tree of the weight sequences for ensemble of 3 classifiers, generated by algorithm weight.

Algorithm 1 Algorithm to construct weighted ensembled classifier.

```
Input: (C_1, C_2, ··· , C_n)
Output: (w_1, w_2, ··· , w_n)
function weight(k):
    if k > length of W then
        ensemble_classifier = w_1 C_1 + w_2 C_2 + ··· + w_n C_n
        fit ensemble_classifier to the training set
        calculate score
    else
    for i = 1 to length of W
        w_k = i
        weight(k+1)
```

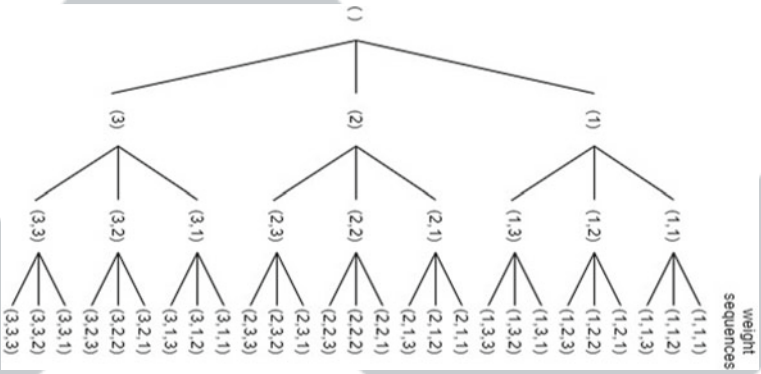

Fig. 3 Generating tree produced by algorithm weight for ensemble of 3 classifiers. The classifiers then is fitted for each generated set of weights

3.4 Selecting Classifiers to Ensemble

We define the following steps to select classifiers to ensemble:

1. Take the best p classifiers on each metric performance.
2. Count the total occurrences of those p classifiers.
3. For a $q \leq p$, select q classifiers with frequent occurences.
4. Construct all possible ensemble of q classifiers using these selected classifiers.

We conjecture that if two classifiers are ensembled, then the performance of the ensembled classifiers are at least the same as the smallest performance among them.

4 Experiment and Results

Table 1 shows the score performance of the six single models in plain sampling, oversampling 100%, undersampling 100%, and oversampling 50%. In plain sampling,

Table 1 Performance of single models with plain and resampling scheme (the results in bold give the best score)

Model	Metric	Scores (%)			
		Plain	Oversampling 100%	Undersampling 100%	Oversampling 50%
LR	Recall	86.91	73.14	69.69	84.61
	Precision	79.68	84.70	85.51	80.98
	F1	83.11	78.45	76.73	82.71
	AUC	78.73	78.36	77.93	78.54
	Accuracy	75.30	71.92	70.48	75.28
kNN	Recall	91.63	65.91	72.00	85.80
	Precision	76.27	81.89	82.91	78.25
	F1	83.23	72.96	77.00	81.81
	AUC	73.39	70.63	74.13	73.42
	Accuracy	74.16	65.87	69.95	73.32
LDA	Recall	86.69	72.07	69.84	84.57
	Precision	79.96	84.95	85.68	81.00
	F1	83.16	77.93	76.89	82.71
	AUC	78.39	78.34	77.80	78.34
	Accuracy	75.45	71.47	70.69	75.28
DT	Recall	77.11	75.19	62.54	76.20
	Precision	77.20	77.52	79.62	77.28
	F1	77.11	76.28	69.98	76.68
	AUC	61.92	62.11	63.41	61.97
	Accuracy	68.00	67.34	62.53	67.66
NB	Recall	61.91	54.86	63.13	60.44
	Precision	**84.25**	85.12	83.53	84.86
	F1	69.54	65.61	69.99	68.91
	AUC	73.69	72.68	72.92	73.62
	Accuracy	64.98	61.65	64.92	64.59
SVM	Recall	**92.34**	79.83	68.17	88.66
	Precision	77.79	82.03	**86.70**	79.53
	F1	**84.42**	80.87	76.26	83.82
	AUC	**79.36**	77.14	78.89	78.02
	Accuracy	**76.16**	73.60	70.36	76.06

the highest score for recall, F1, AUC, and accuracy is achieved by SVM, and the highest precision is given by NB. Resampling scheme is only superior in precision score, which is given by SVM in undersampling 100% scheme.

Table 2 Benchmark models

Metrics	Benchmark (%)		
	I	II	III
Recall	92.34	61.91	68.17
Precision	77.79	84.25	86.70
F1	84.42	69.54	76.26
AUC	79.36	73.69	78.89
Accuracy	76.16	64.98	70.36

In addition, we observed some regularities on this experiment:

- kNN and SVM give recall score greater than 90% in plain dataset.
- In general, recall scores drops significantly (greater than 10%) in oversampling 100% (except for DT), and get worst in undersampling 100%, except for kNN and NB where the score in undersampling 100% is better than in oversampling 100%.
- In general, precision score is better in undersampling 100% than the other resampling schemes.
- Plain sampling gives better accuracy.

Based on the results in Table 1, we define three benchmarks as shown on the Table 2. Benchmark I is taken from performance of SVM trained on plain dataset, since it gives the best scores of four metrics out of five, benchmark II taken from NB, also trained on plain dataset, since it gives the best precision among all single classifiers scores, and benchmark III is taken from SVM trained on undersampled dataset, since it gives the best precision among all considered classifiers score in both plain and resampling scheme.

4.1 Outperforming Benchmark I

Here we construct ensemble of two or three classifiers. First, we select which classifiers to ensemble by using method described in Sect. 3.4. Referring to performance in Table 1, the best three classifiers in each metric are as follows (starting from the best):

–Recall	:	SVM, kNN, LR
–Precision	:	NB, LR, LDA
–F1	:	SVM, KNN, LDA
–AUC	:	SVM, LR, LDA
–Accuracy	:	SVM, LDA, LR

Next, we count the total occurences of each classifier in all metrics above. Each of SVM, LR, and LDA has four occurences, while kNN has two and NB has one. We decide to discard NB and select SVM, LR, LDA, and kNN to form ensemble of three classifiers. Accordingly, to form ensemble of two classifiers, one can consider

the best two classifiers. In this case, SVM has four occurences, kNN and LR have two, LDA and NB have one. We select SVM, kNN, LR, and discard LDA and NB. Ensemble classifiers that go to the experiment are:

–Two classifiers	:	(LR, SVM), (LR, kNN), (kNN, SVM)
–Three classifiers	:	(LR, kNN, SVM), (LR, kNN, LDA) (kNN, LDA, SVM), (SVM, LR, LDA)

The experiment results are shown in Table 3. It follows that ensemble (kNN, LDA, SVM) has all five scores above benchmark I. However, those are achieved in different weight sequences. The model (LR, SVM) and (LR, kNN, SVM) almost have the same performance, but they do not outperform benchmark I in recall scores. The results are shown in Table 4. It follows that our considered weighted ensemble voting models outperform benchmark I in all metrics except recall.

4.2 Outperforming Benchmark II

From Table 3, the performance of (LR, SVM) outperforms benchmark II except for the precision. By ensembling NB with (LR, SVM), we expect (LR, SVM, NB) gives better precision than that of NB. On the contrary, (LR, SVM, NB) probably gives lower performance on four other metrics (recall, F1, AUC, accuracy) compared to that of (LR, SVM). This expectation is confirmed by the experiment results in Table 5. The results show that the performance of (LR, SVM, NB) is in between that of NB and of (LR, SVM). For particular weight set, the ensemble of (LR, SVM, NB) with weights (1, 3, 3) outperforms all metrics in benchmark II as shown on Table 6.

4.3 Outperforming Benchmark III

Ensembling SVM with LDA and NB outperforms all the metrics of benchmark III. See the general results given in Table 7. For particular weight set, the ensemble of (LDA, SVM, NB) with weights (1, 1, 3) outperforms all metrics in benchmark III as shown on Table 8.

5 Conclusions

We give a method to determine classifiers to ensemble. It needs further works to have more general insights on this method. The use of weight generating algorithm gives the weight set that maximizes performance of the weighted ensemble voting classifiers. As our experiment shows, this weight set probably different for each metric. The weighted ensemble voting classifier might increase the performance

Table 3 Maximum of each metric performance of weighted ensemble voting classifier consisting of two and three classifiers, along with their weights (the results in bold outperform the corresponding score of SVM plain)

Ensemble model	Metric	Max mean (%)	Standard deviation	Weights
(LR, SVM)	Recall	91.36	0.0275	(1, 2)
	Precision	**79.50**	0.0164	(2, 1)
	F1	**84.62**	0.0145	(1, 2)
	AUC	**80.06**	0.0218	(2, 2)
	Accuracy	**76.77**	0.0209	(1, 2)
(LR, kNN)	Recall	92.21	0.0241	(1, 1)
	Precision	**78.44**	0.0149	(2, 1)
	F1	84.29	0.0147	(1, 1)
	AUC	79.18	0.0206	(2, 1)
	Accuracy	76.10	0.0212	(1, 1)
(kNN, SVM)	Recall	**93.05**	0.0225	(1, 1)
	Precision	77.44	0.0146	(1, 2)
	F1	84.31	0.0147	(1, 2)
	AUC	78.03	0.0232	(1, 2)
	Accuracy	75.90	0.0218	(1, 2)
(LR, kNN, SVM)	Recall	92.29	0.0234	(1, 3, 1)
	Precision	**79.40**	0.0154	(3, 1, 3)
	F1	**84.54**	0.0153	(1, 2, 2)
	AUC	**79.95**	0.0208	(3, 1, 3)
	Accuracy	**76.78**	0.0212	(3, 1, 3)
(LR, kNN, LDA)	Recall	91.90	0.0246	(1, 3, 1)
	Precision	**79.52**	0.0175	(3, 1, 3)
	F1	84.24	0.0146	(1, 2, 1)
	AUC	79.22	0.0205	(3, 2, 3)
	Accuracy	76.05	0.0208	(1, 2, 1)
(kNN, LDA, SVM)	Recall	**92.71**	0.0219	(3, 1, 1)
	Precision	**79.14**	0.0155	(1, 3, 1)
	F1	**84.73**	0.0157	(1, 2, 3)
	AUC	**79.96**	0.0202	(1, 3, 3)
	Accuracy	**76.88**	0.0229	(1, 2, 3)
(SVM, LR, LDA)	Recall	90.52	0.0296	(1, 1, 3)
	Precision	**79.58**	0.0219	(3, 3, 1)
	F1	84.31	0.0183	(1, 1, 2)
	AUC	**80.19**	0.0237	(2, 1, 3)
	Accuracy	**76.50**	0.0268	(1, 1, 2)

Table 4 Performance of ensembled models trained on plain dataset that beat benchmark I in at least 4 metrics (the results in bold outperform their corresponding benchmark)

Ensemble model	Weights	Metric deviation	Mean (%)	Standard	Difference with benchmark I
(LR, SVM)	(1, 2)	Recall	91.36	0.0275	−0.99
		Precision	**78.85**	0.0158	**1.07**
		F1	**84.62**	0.0145	**0.20**
		AUC	**80.04**	0.0225	**0.68**
		Accuracy	**76.77**	0.0209	**0.61**
(LR, kNN, SVM)	(3, 1, 3)	Recall	90.31	0.0281	−2.03
		Precision	**79.40**	0.0154	**1.61**
		F1	**84.47**	0.0151	**0.06**
		AUC	**79.95**	0.0208	**0.59**
		Accuracy	**76.78**	0.0212	**0.62**
(kNN, LDA, SVM)	(1, 2, 3)	Recall	91.67	0.0285	−0.68
		Precision	**78.82**	0.0175	**1.03**
		F1	**84.73**	0.0157	**0.31**
		AUC	**79.90**	0.0208	**0.54**
		Accuracy	**76.88**	0.0229	**0.72**

Table 5 The best performance of (LR, SVM, NB) (all scores outperform their counterparts in benchmark II)

Ensemble model	Weights	Metric	Mean (%)	Standard deviation
(LR, SVM, NB)	(3, 1, 3)	Recall	86.62	0.0341
	(1, 3, 3)	Precision	84.42	0.0332
	(3, 1, 3)	F1	83.48	0.0184
	(3, 1, 3)	AUC	80.00	0.0239
	(3, 1, 3)	Accuracy	76.03	0.0253

Table 6 Ensemble model (LR, SVM, NB) with weights (1, 3, 3) trained in resampled dataset that beats benchmark II.

Ensemble model	Weights	Metric	Mean (%)	Standard deviation	Difference with benchmark II
(LR, SVM, NB)	(1, 3, 3)	Recall	71.74	0.1135	9.82
		Precision	84.42	0.0332	0.17
		F1	76.82	0.0668	7.29
		AUC	78.98	0.0254	5.29
		Accuracy	70.60	0.0537	5.62

Table 7 The best performance of (LDA, SVM, NB) (all scores outperform their counterparts in benchmark III).

Ensemble model	Weights	Metric	Mean (%)	Standard deviation
(LR, SVM, NB)	(3, 1, 3)	Recall	69.45	0.0448
	(1, 1, 3)	Precision	86.77	0.0246
	(3, 1, 3)	F1	76.97	0.0289
	(1, 1, 3)	AUC	79.05	0.0243
	(3, 1, 3)	Accuracy	71.02	0.0292

Table 8 Ensemble model (LDA, SVM, NB) with weights (1, 1, 3) trained in resampled dataset that beats benchmark III

Ensemble model	Weights	Metric	Mean (%)	Standard deviation	Difference with benchmark III
(LDA, SVM, NB)	(1, 1, 3)	Recall	68.95	0.0580	0.78
		Precision	86.77	0.0246	0.07
		F1	76.67	0.0357	0.41
		AUC	79.05	0.0243	0.16
		Accuracy	70.83	0.0338	0.47

compared to that given by single classifier. The experiments also show that the performance of ensemble classifiers are at least the same as the weakest performance of the involved classifier. On the contrary, its best performance might outperform the best performance of the involved classifiers. In general, resampling the German credit dataset by oversampling 100%, undersampling 100%, or oversampling 50%, decrease the performance except for precision. It needs further investigation to find the ratio that maximizes the performance. For future work, the exhaustive weight generating algorithm can also be implemented in other single classifiers for credit scoring proposed by Lessmann et al. [19].

References

1. Chawla NV, Bowyer KW, Hall LO, Kagelmeyer WP (2002) SMOTE: synthetic minority oversampling technique. J Artif Intell Res 16:321–357
2. Dua D, Graff C (2019) UCI machine learning repository, Irvine, University of California, School of Information and Computer Science. http://archive.ics.uci.edu/ml
3. Sabri A, Vajnovszki V (2015) Two reflected Gray code based orders on some restricted growth sequences. Comput J 58(5):1099–1111
4. Sabri A, Vajnovszki V (2019) On the exhaustive generation for ballot sequences in lexicographic and Gray code order. Pure Math Appl 28(1):109–119
5. Li ST, Shiueb W, Huang MH (2006) The evaluation of consumer loans using support vector machines. Expert Syst Appl 30(4):772–782

6. Khemakhem S, Boujelbene Y (2017) Artificial intelligence for credit risk assessment: artificial neural network and support vector machines. ACRN Oxford J Fin Risk Perspect 6(2):1–17
7. Bellotti T, Crook J (2009) Support vector machines for credit scoring and discovery of significant features. Expert Syst Appl 36(2):3302–3308
8. Yeh IC, Lien CH (2009) The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Exp Syst Appl 36(2):2473–2480
9. Henley WE, Hand DJ (1996) A k-nearest-neighbour classifier for assessing consumer credit risk. J Royal Statist Soc: Ser D (The Statistician) 45(1):77–95
10. Davis RH, Edelman DB, Gammerman AJ (1992) Machine-learning algorithms for credit-card applications. IMA J Manage Math 4(1):43–51
11. Frydman H, Altman EI, Kao DL (1985) Introducing recursive partitioning for financial classification: the case of financial distress. J Fin 40(1):269–291
12. Handhika T, Fahrurozi A, Zen RIM, Lestari DP, Sari I (2019) Murni: modified average of the base-level models in the hill-climbing bagged ensemble selection algorithm for credit scoring. Proc Comput Sci 157:229–237
13. Hung C, Chen JH (2009) A selective ensemble based on expected probabilities for bankruptcy prediction. Expert Syst Appl 36(3):5297–5303
14. Singh NP, Dahiya S, Handa SS (2015) Credit scoring using ensemble of various classifiers on reduced feature set. Industrija 43(4):163–172
15. Waad B, Ghazi BM, Mohamed L (2013) A three-stage feature selection using quadratic programming for credit scoring. Appl Artif Intell 27(8):721–742
16. Sang HV, Nam NH, Nhan ND (2016) A novel credit scoring prediction model based on feature selection approach. Ind J Sci Technol 9(20):1–6
17. Kuncheva LI (2014) Combining pattern classifiers: methods and algorithms, 2nd edn. Wiley, Hoboken
18. Dietterich TG (1997) Machine-learning research: four current directions. AI Magaz 18(4):97–136
19. Lessmann S, Baesens B, Seow H-V, Thomas LC (2015) Benchmarking state-of-the-art classification algorithms for credit scoring: an update of research. Eur J Operat Res 247:124–136