Knowledge-Based Systems 194 (2020) 105534

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

# GEV-NN: A deep neural network architecture for class imbalance problem in binary classification

Lkhagvadorj Munkhdalai[a], Tsendsuren Munkhdalai[b], Keun Ho Ryu[c,*]

[a] Database/Bioinformatics Laboratory, School of Electrical and Computer Engineering, Chungbuk National University, Cheongju, 28644, Republic of Korea
[b] Microsoft Research, Montreal, QC H3A 3H3, Canada
[c] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, 700000, Viet Nam

ARTICLE INFO

*Article history:*
Received 20 September 2019
Received in revised form 13 January 2020
Accepted 16 January 2020
Available online 18 January 2020

*Keywords:*
Neural networks
Auto-encoder
Gumbel distribution
Imbalanced classification

ABSTRACT

Class imbalance is a common issue in many applications such as medical diagnosis, fraud detection, web advertising, etc. Although standard deep learning method has achieved remarkably high-performance on datasets with balanced classes, its ability to classify imbalanced dataset is still limited. This paper proposes a novel end-to-end deep neural network architecture and adopts Gumbel distribution as an activation function in neural networks for class imbalance problem in the application of binary classification. Our proposed architecture, named GEV-NN, consists of three components: the first component serves to score input variables to determine a set of suitable input, the second component is an auto-encoder that learns efficient explanatory features for the minority class, and in the last component, the combination of the scored input and extracted features are then used to make the final prediction. We jointly optimize these components in an end-to-end training. Extensive experiments using real-world imbalanced datasets showed that GEV-NN significantly outperforms the state-of-the-art baselines by around 2% at most. In addition, the GEV-NN gives a beneficial advantage to interpret variable importance. We find key risk factors for hypertension, which are consistent with other scientific researches, using the first component of GEV-NN.

## 1. Introduction

Deep neural network has been well developed and successfully applied to many applications on classification tasks. However, imbalanced data with a skewed class distribution forces standard deep learning architecture to achieve poor classification performance [1,2]. The imbalanced data is characterized as some of its classes (minority) in the dataset are heavily rare compared to other classes (majority). As minority samples occur infrequently, standard deep learning architectures tend to misclassify the minority classes compared to the majority classes [3,4]. Currently, several approaches have been proposed on this issue in the field of deep neural networks. Most researchers proposed the specific implementation of either over/under-sampling [5–9] or cost-sensitive learning [10–14] on neural networks. Another line of researchers has been focused on specific loss function for imbalanced classification problem [3,13,15]. Recently, Wang et al. [3] and Raj et al. [15] introduced a loss function, which captures classification errors from both the majority and minority classes. In addition, some researchers have applied asymmetric link functions in the generalized linear model (GLM) to estimate the class probability for binary classification problem [16,17]. For instance, to deal with imbalanced datasets, the generalized extreme value distribution (GEV) function has been used to form the asymmetric link function in GLM. GEV function is a family of continuous probability distributions, which have been used for modeling rare events in statistics [18,19].

However, a limited number of studies have been done on imbalanced classification problem in the field of deep learning and they have not considered deep neural network architecture and its activation function [3,4].

In this paper, we propose a novel end-to-end deep neural network architecture and adopt Gumbel distribution as an activation function to improve the predictive accuracy for the class imbalance problem in binary classification. In our proposed framework named GEV-NN, there are three components: the first component helps to adaptively score inputs for each instance, the second component is an auto-encoder that extracts efficient features for explaining the minority class as well as encoded representation of input are created. Subsequently, scored inputs, encoded representation and extracted features are concatenated to be further

[*] Correspondence to: 19 Nguyen Huu Tho, District 7, Ho Chi Minh City, Vietnam.

*E-mail addresses:* lhagii@dblab.chungbuk.ac.kr (L. Munkhdalai), tsendsuren.munkhdalai@microsoft.com (T. Munkhdalai), khryu@tdtu.edu.vn (K.H. Ryu).

*URL:* http://dblab.chungbuk.ac.kr (K.H. Ryu).

https://doi.org/10.1016/j.knosys.2020.105534
0950-7051/© 2020 Elsevier B.V. All rights reserved.

fed into the third network for the final prediction. Fig. 1 shows GEV-NN architecture, where the first network is a feedforward neural network (FNN) with softmax output. This FNN generates the scores that can adaptively select the important variables because we use a skip connection and perform an element-wise multiplication of input variables and those scores that represent the variable importance. In addition, auto-encoder is parallelly trained to learn important features for the minority class such as Euclidean, Cosine distance, Mahalanobis and Chebyshev distance between reconstructed and input features, and to generate encoded representation of input variables. Finally, we concatenate scored input, encoded representation and extracted features to train the prediction network for the final prediction. Previously, Zong et al. [20] and Laptev et al. [21] studies proposed unlike neural network architectures for unsupervised anomaly detection and time series extreme event forecasting, respectively.

Furthermore, the sigmoid activation function shows some important drawbacks in class imbalance issue: the probability of the minority class is underestimated and this function is a symmetric function, thus the response curve reaches zero as the same rate it reaches one [22]. To overcome these drawbacks of sigmoid function, we use the Gumbel distribution as an activation function in GEV-NN. This function is continuously differentiable, thus it can be easily used as an activation function in neural networks with stochastic gradient descent (SGD) optimization.

In the experimental part, GEV-NN is evaluated on real-life benchmark datasets and compared to the baseline state-of-the-art methods. The results showed that GEV-NN demonstrates the promising results and this network can define key risk factors for hypertension, which are consistent with other research work.

The remainder of this paper is organized as follows: Section 2 describes GEV-NN architecture. Then Section 3 presents datasets and experimental results. Finally, Section 4 summarizes the general findings from this study.

## 2. Proposed GEV-NN

### 2.1. GEV-NN architecture

In this section, we introduce our proposed GEV-NN architecture in detail. Fig. 2 shows the overview of GEV-NN architecture. This model consists of three major components: a weighting network, an auto-encoder and a prediction network. As shown in Fig. 2, it works as follows: (1) the weighting network helps to generate variable scores that can adaptively control input variables; (2) the auto-encoder prepares encoded representation and efficient explanatory features for the minority class; (3) the prediction network takes the combination of scored input variables, the encoded representation and the generated features for the final prediction.

#### 2.1.1. The weighting network

The goal of weighting network is to find scores that govern the influence of input variable for each instance. The output layer of weighting network is a *softmax* function that provides the weights or probabilities to adaptively control input variables [23].

Assuming we use $k$ different variables as an input, the $i$th input of $N$ instances can be represented by tuples $(x_i^{(j)}) \in R^k$, $i = 1, \ldots, N$ and $j = 1, \ldots, k$. We use FNN with *softmax*$(k)$ output layer to find weights $(\omega)$ in weighting network as follows:

$$\omega_i^{(j)} = \frac{e^{f^j(x_i;\theta_w)}}{\sum_{j=1}^{k} e^{f^j(x_i;\theta_w)}} \tag{1}$$

where $\omega_i^{(j)}$ is $j$th variable importance for $i$th instance, $f^j(x_i; \theta_w)$ denotes $j$th node's input of *softmax* function for $i$th instance, $\theta_w$ denotes the weight parameters of the weighting network. According to Eq. (1), the weighting network can adaptively score input variables for each instance. We then take an element-wise multiplication of input variables and importance weights come from the weighting network using a skip connection [24,25].

$$z_w = x \odot \omega \tag{2}$$

where $z_w$ denotes the weighted input.

Note that the variable importance scores can be seen as fast-weights, which has successfully been utilized in the meta-learning context for rapid adaptation [26–30]. Our importance weights have a direct probabilistic interpretation while the other fast-weights approach inherits the black box property of the neural net.

#### 2.1.2. The auto-encoder

The aim of auto-encoder is to derive efficient features, which can explain the minority class, from reconstructed input. Recently, Zong et al. [20] and Laptev et al. [21] studies used auto-encoder to generate efficient features for unsupervised anomaly detection and time series extreme event forecasting, respectively.

Given a sample $x$, the auto-encoder computes reconstructed input $x'$ as follows:

$$\begin{aligned} z_e &= f_{encoding}\,(x; \theta_e) \\ x' &= f_{decoding}\,(z_e; \theta_d) \end{aligned} \tag{3}$$

where $z_e$ is the reduced low-dimensional encoded representation learned by the encoder, $x'$ is the reconstructed input, $f_{encoding}$ denotes the encoding network, $f_{decoding}$ denotes the decoding network and $\theta_e, \theta_d$ are the weight parameters of auto-encoder, respectively.

Subsequently, we generate some efficient features for the minority class using original and reconstructed input.

$$z_{dist} = dist(x, x') \tag{4}$$

where $z_{dist}$ denotes the generated features from original and reconstructed input, and $dist(*)$ denotes the distance function of calculating efficient features such as Euclidean distance, cosine distance, etc. [20]. In this study, we used Euclidean and cosine distances.

#### 2.1.3. The prediction network

As previously explained, the concatenated outputs of weighting network and auto-encoder are used to train the prediction network. In this component, we use FNN for building model to predict the final class label.

$$\begin{aligned} z &= [z_\omega, z_e, z_{dist}] \\ \hat{y} &= FNN(z; \theta_m) \end{aligned} \tag{5}$$

where $z$ denotes the concatenated input for the prediction network, $\theta_m$ is the weight parameters of FNN and $\hat{y}$ denotes the predicted class label.

### 2.2. Gumbel distribution as an activation function

We also adopt the Gumbel distribution as an activation function because sigmoid does not perform well in imbalanced data as it tends to underestimate the probability of the minority class [16,17,22]. The Gumbel distribution, also known as Generalized Extreme Value (GEV) distribution Type-I, is widely used to design the distribution of extreme samples of various distributions [31]. This function has been extensively applied either as the parent distribution or as an asymptotic approximation, to characterize

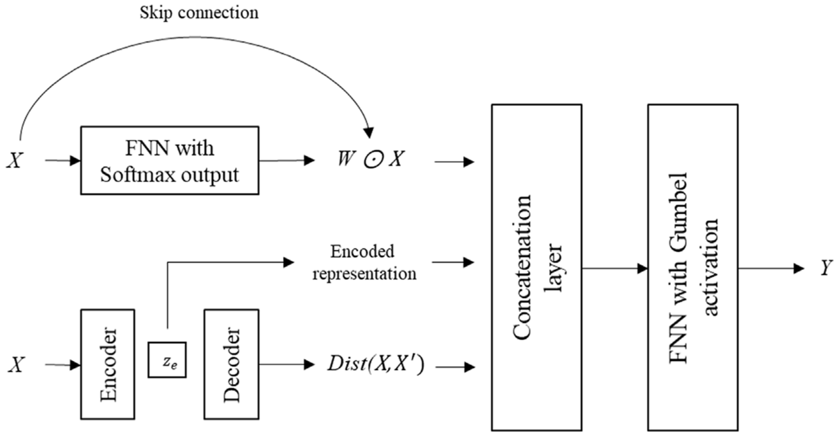

**Fig. 1.** GEV-NN architecture. $\boldsymbol{X}$ denotes input, $\boldsymbol{W}$ denotes the importance weight for input, which is the output of first network, $\boldsymbol{X}'$ is the reconstructed input by decoder, $\boldsymbol{Dist}(*)$ is the distance functions, $\boldsymbol{z_e}$ is the encoded representation of input and $\boldsymbol{Y}$ denotes the predicted class label. Feedforward neural network (FNN) is used as a model.

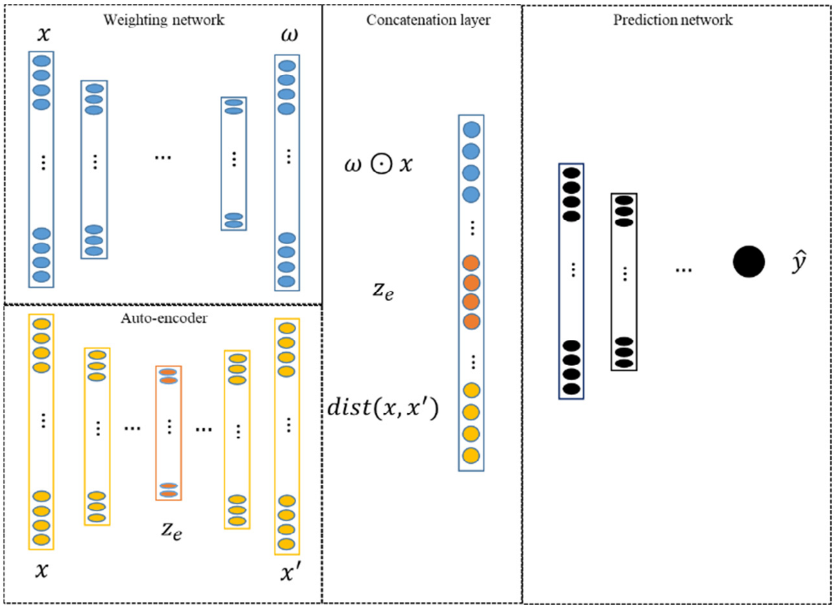

**Fig. 2.** An overview of GEV-NN architecture. $\boldsymbol{x}$ denotes input, $\boldsymbol{\omega}$ denotes the importance weights, $\boldsymbol{x}'$ denotes the reconstructed input by auto-encoder, $\boldsymbol{dist}(*)$ is the distance functions, $\boldsymbol{z_e}$ denotes the encoded representation and $\hat{\boldsymbol{y}}$ denotes the predicted class label.

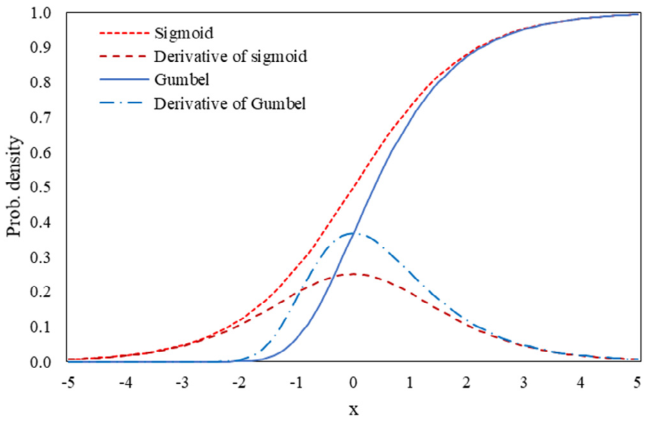

**Fig. 3.** Comparison between sigmoid and Gumbel activation functions.

extreme wind speeds, floods, age at death, minimum or maximum temperature, electrical strength of materials, risk assessment in financial, geological problems, etc. [32]. The cumulative distribution function (CDF) is as follows:

$$F(x) = e^{-e^{(-x)}} \tag{6}$$

The Gumbel distribution function is continuously differentiable, thus it can be easily used as an activation function in neural networks with SGD optimization. In addition, the asymmetry implies the Gumbel function will penalize misclassification differently on both classes [33]. Fig. 3 shows the curves of CDF and derivatives of both sigmoid and Gumbel functions.

### 2.3. Training GEV-NN

Our goal is to jointly optimize GEV-NN. The objective function of GEV-NN consists of two losses: the first loss is a loss for auto-encoder, which can be the mean squared error (MSE) that measures the average of the squares of the reconstruction error, and the second loss is a loss for the prediction network, which can

**Table 1**
Description of the benchmark datasets.

| Datasets | #Instances | #Variables | IR | Datasets | #Instances | #Variables | IR |
|---|---|---|---|---|---|---|---|
| abalone17vs78910 | 2338 | 8 | 39.31 | krvskzeroonevsdraw | 2901 | 6 | 26.63 |
| abalone19 | 4174 | 8 | 129.44 | led7digit02456789vs1 | 443 | 7 | 10.97 |
| abalone19vs10111213 | 1622 | 8 | 49.69 | lymphography | 148 | 18 | 23.67 |
| abalone20vs8910 | 1916 | 8 | 72.69 | newthyroid1 | 215 | 5 | 5.14 |
| abalone21vs8 | 581 | 8 | 40.5 | newthyroid2 | 215 | 5 | 5.14 |
| abalone3vs11 | 502 | 8 | 32.47 | pageblocks0 | 5472 | 10 | 8.79 |
| abalone9vs18 | 731 | 8 | 16.4 | pageblocks13vs4 | 472 | 10 | 15.86 |
| cargood | 1728 | 6 | 24.04 | pima | 768 | 8 | 1.87 |
| carvgood | 1728 | 6 | 25.58 | poker8vs6 | 1477 | 10 | 85.88 |
| cleveland0vs4 | 177 | 13 | 12.62 | poker89vs5 | 2075 | 10 | 82 |
| dermatology6 | 358 | 34 | 16.9 | poker89vs6 | 1485 | 10 | 58.4 |
| ecoli0vs1 | 220 | 7 | 1.86 | poker9vs7 | 244 | 10 | 29.5 |
| ecoli01vs235 | 244 | 7 | 9.17 | segment0 | 2308 | 19 | 6.02 |
| ecoli01vs5 | 240 | 6 | 11 | shuttle2vs5 | 3316 | 9 | 66.67 |
| ecoli0137vs26 | 281 | 7 | 39.14 | shuttle6vs23 | 230 | 9 | 22 |
| ecoli0146vs5 | 280 | 6 | 13 | shuttlec0vsc4 | 1829 | 9 | 13.87 |
| ecoli0147vs2356 | 336 | 7 | 10.59 | shuttlec2vsc4 | 129 | 9 | 20.5 |
| ecoli0147vs56 | 332 | 6 | 12.28 | vehicle0 | 846 | 18 | 3.25 |
| ecoli0234vs5 | 202 | 7 | 9.1 | vehicle1 | 846 | 18 | 2.9 |
| ecoli0267vs35 | 224 | 7 | 9.18 | vehicle2 | 846 | 18 | 2.88 |
| ecoli034vs5 | 200 | 7 | 9 | vehicle3 | 846 | 18 | 2.99 |
| ecoli0346vs5 | 205 | 7 | 9.25 | vowel0 | 988 | 13 | 9.98 |
| ecoli0347vs56 | 257 | 7 | 9.28 | winequalityred3vs5 | 691 | 11 | 68.1 |
| ecoli046vs5 | 203 | 6 | 9.15 | winequalityred4 | 1599 | 11 | 29.17 |
| ecoli067vs35 | 222 | 7 | 9.09 | winequalityred8vs6 | 656 | 11 | 35.44 |
| ecoli067vs5 | 220 | 6 | 10 | winequalityred8vs67 | 855 | 11 | 46.5 |
| ecoli1 | 336 | 7 | 3.36 | winequalitywhite3vs7 | 900 | 11 | 44 |
| ecoli2 | 336 | 7 | 5.46 | winequalitywhite39vs5 | 1482 | 11 | 58.28 |
| ecoli3 | 336 | 7 | 8.6 | winequalitywhite9vs4 | 168 | 11 | 32.6 |
| ecoli4 | 336 | 7 | 15.8 | wisconsin | 683 | 9 | 1.86 |
| flareF | 1066 | 11 | 23.79 | yeast0256vs3789 | 1004 | 8 | 9.14 |
| glass0 | 214 | 9 | 2.06 | yeast02579vs368 | 1004 | 8 | 9.14 |
| glass0123vs456 | 214 | 9 | 3.2 | yeast0359vs78 | 506 | 8 | 9.12 |
| glass0146vs2 | 205 | 9 | 11.06 | yeast05679vs4 | 528 | 8 | 9.35 |
| glass015vs2 | 172 | 9 | 9.12 | yeast1 | 1484 | 8 | 2.46 |
| glass016vs2 | 192 | 9 | 10.29 | yeast1vs7 | 459 | 7 | 14.3 |
| glass016vs5 | 184 | 9 | 19.44 | yeast1289vs7 | 947 | 8 | 30.57 |
| glass04vs5 | 92 | 9 | 9.22 | yeast1458vs7 | 693 | 8 | 22.1 |
| glass06vs5 | 108 | 9 | 11 | yeast2vs4 | 514 | 8 | 9.08 |
| glass1 | 214 | 9 | 1.82 | yeast2vs8 | 482 | 8 | 23.1 |
| glass2 | 214 | 9 | 11.59 | yeast3 | 1484 | 8 | 8.1 |
| glass4 | 214 | 9 | 15.46 | yeast4 | 1484 | 8 | 28.1 |
| glass5 | 214 | 9 | 22.78 | yeast5 | 1484 | 8 | 32.73 |
| glass6 | 214 | 9 | 6.38 | yeast6 | 1484 | 8 | 41.4 |
| haberman | 306 | 3 | 2.78 | zoo3 | 101 | 16 | 19.2 |
| iris0 | 150 | 4 | 2 | **Sonar** | 208 | 5 | 1.05 |
| krvskonevsfteen | 2244 | 6 | 27.77 | **Bupa** | 345 | 6 | 1.38 |
| krvskthreevseleven | 2935 | 6 | 35.23 | **Iono** | 351 | 32 | 1.78 |
| krvskzerovseight | 1460 | 6 | 53.07 | **Vert2** | 310 | 6 | 2.1 |
| krvskzerovsfteen | 2193 | 6 | 80.22 | **Park** | 195 | 22 | 3.06 |

be the binary cross-entropy that measures the performance of a classification model whose output is a probability value between 0 and 1.

Given a dataset of $N$ instances, the objective function of GEV-NN is constructed as follows:

$$J(\theta_w, \theta_s, \theta_e, \theta_d, \theta_m) = \frac{\lambda}{N}\sum_{i=1}^{N} L(x_i, x_i'; \theta_e, \theta_d) + \frac{(1-\lambda)}{N}\sum_{i=1}^{N} E(y_i, y_i'; \theta_w, \theta_s, \theta_e, \theta_d, \theta_m) \tag{7}$$

where $L(*)$ denotes MSE loss, $E(*)$ denotes binary cross-entropy loss and $\lambda$ is the loss weight parameter. This parameter can be selected by cross-validation or by using a validation set.

To produce meaningful features for the minority class and the well-learned low-dimensional representations from the auto-encoder, our auto-encoder of GEV-NN have to be well-trained. Therefore, we first perform pre-training for the auto-encoder to find the well-trained model. Second, it is also found that the auto-encoder and prediction network could mutually improve each other's performance [20].

## 3. Experimental result

The experimental result is reported in this section. Dataset is presented in Section 3.1 and experimental setup is briefly introduced in Section 3.2 as well as this section provides the evaluation setup. Statistical tests for performance comparison is briefly introduce in Section 3.3. The prediction performance and comparison are described in Section 3.4. In the end, we apply GEV-NN to real-world hypertension dataset to compare the state-of-the-art machine learning baselines and to demonstrate its interpretability by using the weighting network.

### 3.1. Dataset

We use 100 real-world benchmark imbalanced datasets from Keel dataset repository and UCI repository (Sonar, Bupa, Iono, Vert2 and Park) for evaluating and comparing our GEV-NN and

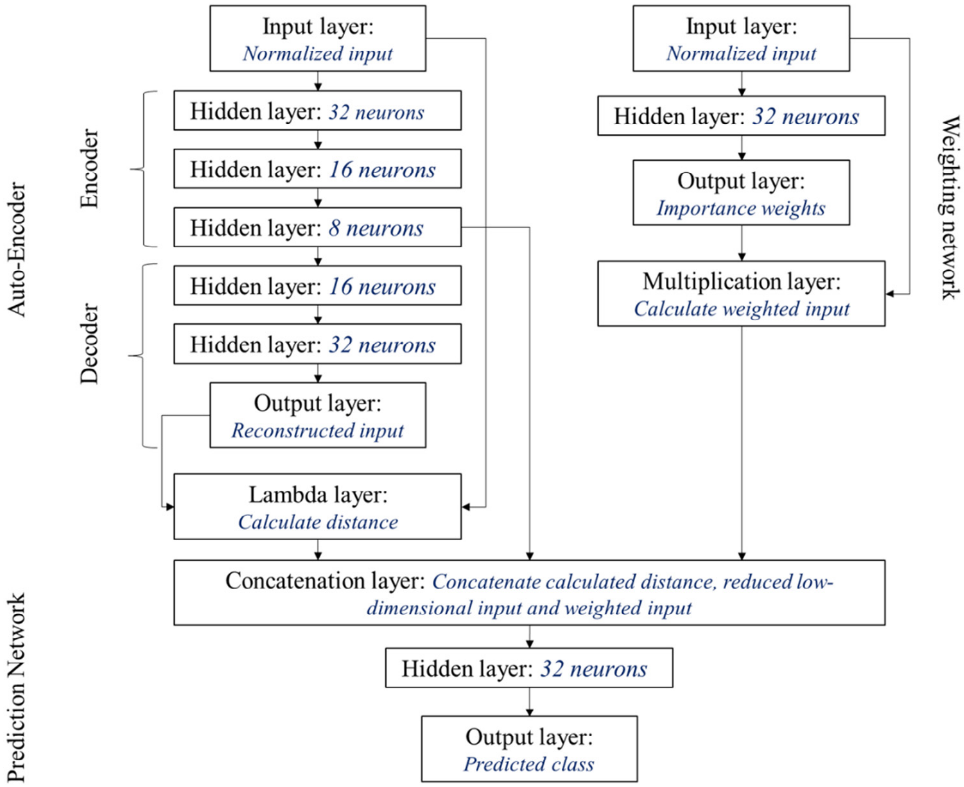
Input layer: Normalized input
Hidden layer: 32 neurons
Hidden layer: 16 neurons
Hidden layer: 8 neurons
Hidden layer: 16 neurons
Hidden layer: 32 neurons
Output layer: Reconstructed input
Lambda layer: Calculate distance
Encoder
Decoder
Auto-Encoder
Input layer: Normalized input
Hidden layer: 32 neurons
Output layer: Importance weights
Multiplication layer: Calculate weighted input
Weighting network
Concatenation layer: Concatenate calculated distance, reduced low-dimensional input and weighted input
Hidden layer: 32 neurons
Output layer: Predicted class
Prediction Network

**Fig. 4.** Visualization of GEV-NN model.

baselines [34,35]. These datasets cover a wide range of different degrees of imbalance and consist of different number of instances and variables. In Table 1, we show the summarized datasets. In order to evaluate and compare the predictive accuracy of GEV-NN to the result of baselines, 5-fold cross validation is utilized. We also split training set into two parts; i.e., training (80%) and validation (20%) sets to train GEV-NN.

### 3.2. Experimental setup

For GEV-NN, once a large number of datasets are utilized, we used the same neural network architecture for all datasets. The weighting network contains only one hidden layer with 32 neurons and auto-encoder uses five hidden layers with {32, 16, 8, 16, and 32} neurons, respectively. The prediction network consists of only one hidden layer with 32 neurons as shown in Fig. 4.

For hyper-parameters: learning rate, batch size, and epoch number must be pre-defined to train a model. We set the learning rate to 0.001, epoch number for training to 500 and use a mini-batch with 8 instances at each iteration. Furthermore, the loss weight parameter $\lambda$ is chosen to be equal to 0.05 and an Early Stopping algorithm is used for finding the optimal epoch number based on given other hyper-parameters. We also compared Gumbel activation function to Sigmoid (SIG-NN) function with the same neural network architecture in our experiments.

We are able to optimize the hyper-parameters of GEV-NN using additional experiments to improve the results, but the chosen parameters render desirable results in this study. Therefore, to save the computation time, we did not make any additional experiments for hyper-parameter optimization.

Regarding the performance measures, we use the most used measures, which are the AUC [35] and the geometric mean (G-mean) of the true rates [36], to compare our performance to other experimental studies and results with Keel dataset repository [37–40]. G-mean can be defined as follows:

$$G-mean=\sqrt{\frac{TP}{TP+FN}\bullet\frac{TN}{TN+FP}} \tag{8}$$

where TP/TN indicates for the correct classifications of the majority/minority class, and FN/FP for the misclassifications of majority/minority class, respectively.

### 3.3. Statistical tests for performance comparison

However, we can show that one model is better than another by comparing their achieved performance, but it is inadequate evidence [41]. In this study, we will perform statistical tests for performance comparison to find difference in classifiers across multiple experiments.

Friedman's test, which is a non-parametric test that aims to determine significant differences between the results of two or more classifiers over multiple datasets, will be performed. If the null hypothesis of the Friedman test is rejected, we can then perform a post-hoc test to find the pair-wise comparisons that produce significant differences. We will perform the Bergmann–Hommel's dynamic post-hoc procedure, as suggested in [38]. With this procedure, we can know which pairs of classifiers demonstrate significant differences in their results as well as we can see a critical difference (CD) plot, which is the order of the classifiers, in post-hoc results.

### 3.4. Result and comparison

To show the superiority of our model, we will directly compare our performances to the results of several studies, which are used mutual datasets with our work [37–40].

**Table 2**
Bergmann–Hommel's test results for the comparison of SIG-NN and GEV-NN (the control algorithm) versus Keel repository models on the 33 datasets considering G-mean metrics.

| № | Models | SIG-NN | | GEV-NN | | Average rank |
|---|---|---|---|---|---|---|
| | | Adjusted *p*-value | Hypothesis | Adjusted *p*-value | Hypothesis | |
| 1 | SIG-NN | NA | | 0.9595 | Not rejected | 2.21 |
| 2 | GEV-NN | 0.9595 | Not rejected | NA | | 2.27 |
| 3 | XCS_SM | 0.3963 | Not rejected | 0.3686 | Not rejected | 7.97 |
| 4 | Gassist_SM | 0.3192 | Not rejected | 0.2952 | Not rejected | 9.24 |
| 5 | C45_SM | 0.1696 | Not rejected | 0.1543 | Not rejected | 9.77 |
| 6 | C45.Rules_SM | 0.1130 | Not rejected | 0.1019 | Not rejected | 10.95 |
| 7 | SIA_SM | 0.0820 | Not rejected | 0.0734 | Not rejected | 11.97 |
| 8 | DT.GA_SM | 0.0515 | Not rejected | 0.0457 | Rejected | 12.52 |
| 9 | Oblique.DT_SM | 0.0568 | Not rejected | 0.0505 | Not rejected | 14.08 |
| 10 | Ripper | 0.0258 | Rejected | 0.0226 | Rejected | 14.82 |
| 11 | Ripper_SM | 0.0242 | Rejected | 0.0212 | Rejected | 16.08 |
| 12 | Target_SM | 0.0109 | Rejected | 0.0094 | Rejected | 17.30 |
| 13 | OIGA_SM | 0.0059 | Rejected | 0.0050 | Rejected | 17.74 |
| 14 | HIDER_SM | 0.0035 | Rejected | 0.0030 | Rejected | 17.89 |
| 15 | CORE_SM | 0.0010 | Rejected | 0.0009 | Rejected | 18.83 |
| 16 | C45.Rules | 0.0008 | Rejected | 0.0007 | Rejected | 19.32 |
| 17 | ILGA_SM | 0.0013 | Rejected | 0.0011 | Rejected | 19.42 |
| 18 | Oblique.DT | 0.0007 | Rejected | 0.0006 | Rejected | 20.17 |
| 19 | C45 | 0.0001 | Rejected | 0.0001 | Rejected | 20.36 |
| 20 | Gassist | 0.0000 | Rejected | 0.0000 | Rejected | 22.36 |
| 21 | SIA | 0.0000 | Rejected | 0.0000 | Rejected | 22.56 |
| 22 | UCS_SM | 0.0000 | Rejected | 0.0000 | Rejected | 22.64 |
| 23 | XCS | 0.0000 | Rejected | 0.0000 | Rejected | 22.95 |
| 24 | CART_SM | 0.0000 | Rejected | 0.0000 | Rejected | 23.23 |
| 25 | DT.GA | 0.0000 | Rejected | 0.0000 | Rejected | 23.42 |
| 26 | OCEC_SM | 0.0000 | Rejected | 0.0000 | Rejected | 23.80 |
| 27 | CART | 0.0000 | Rejected | 0.0000 | Rejected | 24.08 |
| 28 | UCS | 0.0000 | Rejected | 0.0000 | Rejected | 24.82 |
| 29 | OCEC | 0.0000 | Rejected | 0.0000 | Rejected | 25.02 |
| 30 | GIL | 0.0000 | Rejected | 0.0000 | Rejected | 25.14 |
| 31 | GIL_SM | 0.0000 | Rejected | 0.0000 | Rejected | 28.42 |
| 32 | CN2_SM | 0.0000 | Rejected | 0.0000 | Rejected | 30.68 |
| 33 | HIDER | 0.0000 | Rejected | 0.0000 | Rejected | 31.53 |
| 34 | CORE | 0.0000 | Rejected | 0.0000 | Rejected | 31.94 |
| 35 | AQ | 0.0000 | Rejected | 0.0000 | Rejected | 32.42 |
| 36 | OIGA | 0.0000 | Rejected | 0.0000 | Rejected | 32.74 |
| 37 | Target | 0.0000 | Rejected | 0.0000 | Rejected | 32.88 |
| 38 | COGIN_SM | 0.0000 | Rejected | 0.0000 | Rejected | 33.05 |
| 39 | ILGA | 0.0000 | Rejected | 0.0000 | Rejected | 34.89 |
| 40 | Pitts.GIRLA | 0.0000 | Rejected | 0.0000 | Rejected | 35.41 |
| 41 | AQ_SM | 0.0000 | Rejected | 0.0000 | Rejected | 35.83 |
| 42 | CN2 | 0.0000 | Rejected | 0.0000 | Rejected | 37.17 |
| 43 | COGIN | 0.0000 | Rejected | 0.0000 | Rejected | 37.21 |
| 44 | Pitts.GIRLA_SM | 0.0000 | Rejected | 0.0000 | Rejected | 39.56 |
| 45 | DMEL_SM | 0.0000 | Rejected | 0.0000 | Rejected | 41.92 |
| 46 | DMEL | 0.0000 | Rejected | 0.0000 | Rejected | 42.39 |

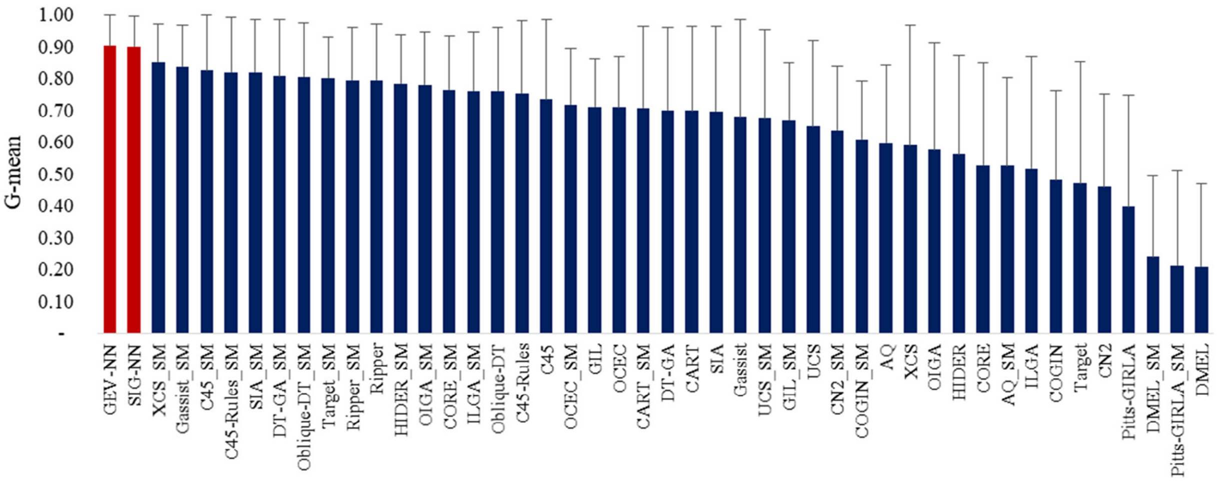

**Fig. 5.** CD plot with statistical comparison of the G-mean for comparison between Keel repository models and GEV-NN.

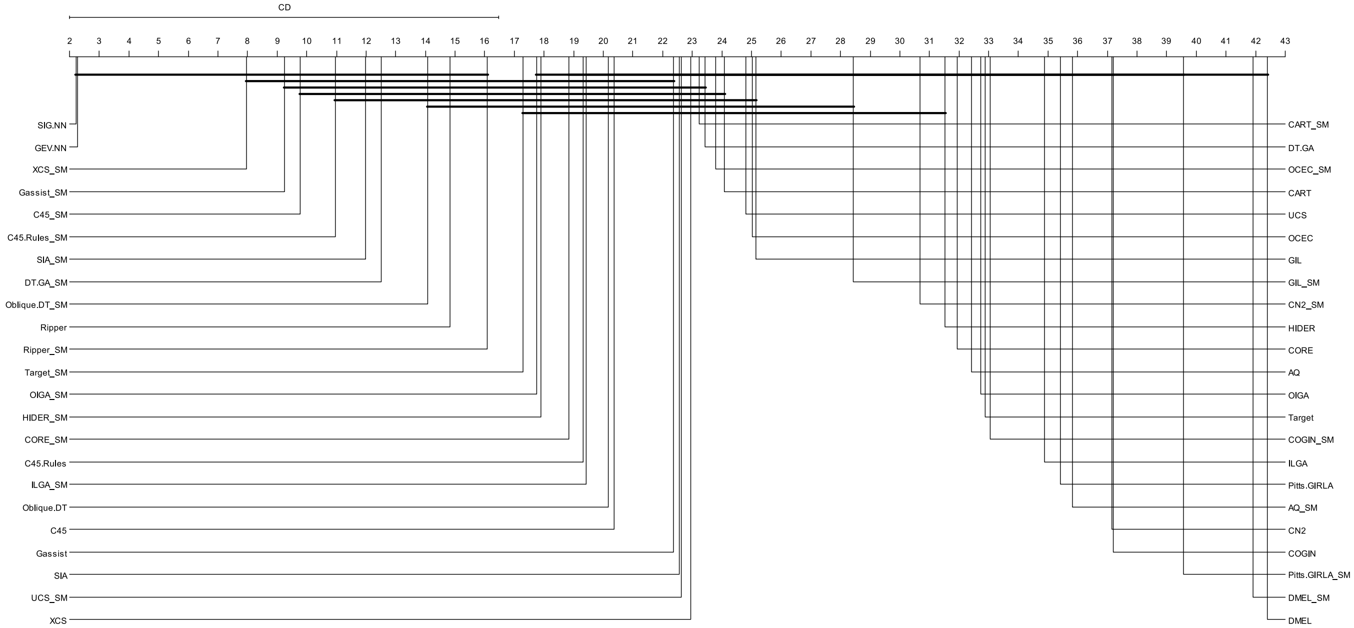

**Fig. 6.** The performance comparison between contrast pattern-based classifier (PBC4cip) and other state-of-the-art models and GEV-NN.

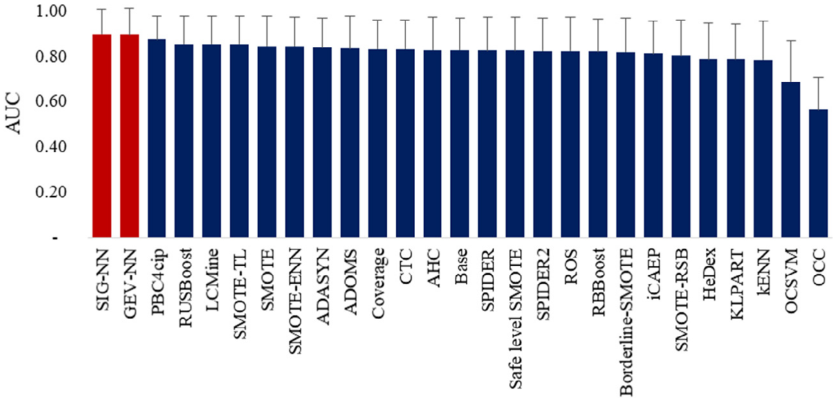

**Fig. 7.** The performance comparison between Keel dataset repository models and GEV-NN.

First, we directly compared our performances to Fernández et al. [37]. They performed an exhaustive study of the performance of a large set of genetics-based machine learning algorithms and non-evolutionary learners, which are 22 baselines as well as SMOTE over-sampling method with 22 baseline models. We compared the average performance of classifiers on 33 benchmark imbalanced datasets in terms of G-mean. It is observed that 22 baseline models in Keel dataset repository underperform our GEV-NN model as well as the combinations of those models of SMOTE over-sampling the minority class could not show better performances. Our GEV-NN framework outperformed baselines by around 4% at most (see Fig. 5).

Table 2 reports the p-values achieved on Bergmann–Hommel's post-hoc test and the average rankings. Our SIG-NN and GEV-NN models are significantly different from most of the models and ranked the first and second by average rank, respectively. In addition, we show CD diagram with a statistical comparison of the G-mean for all models in Fig. 6. It can be proven that our proposed deep learning architecture is statistically significantly different from most of the models in Keel dataset repository. Only XCS-SM, Gassist-SM, C45-SM, C45.Rules-SM, SIA-SM, DT.GA-SM and Oblique.DT-SM models have statistically similar behavior with SIG-NN and GEV-NN.

We also compared our GEV-NN to state-of-the-art models, which are the recently proposed contrast pattern based classifiers [38,39] and fuzzy rule-based oversampling technique [40] for class imbalance problems. Loyola-González et al. [38] studied the impact of using resampling methods for improving the performance of contrast pattern-based classifiers. They compared the most used resampling techniques (nine oversampling, three hybrid, and eight undersampling techniques) such as SMOTE, SMOTE-ENN (SMOTE + Edited Nearest Neighbor), SMOTE-TL (SMOTE + Tomek's modification of Condensed Nearest Neighbor), ADASYN (Adaptive Synthetic Sampling), Borderline-SMOTE (Borderline + SMOTE), Safe LevelSMOTE (Safe Level + SMOTE), ROS (Random oversampling), ADOMS (Adjusting the Direction of the synthetic Minority class) etc.

In addition, Loyola-González et al. [39] showed that their proposed classifier significantly better than other state-of-the-art classifiers, which are not only directly based on contrast patterns but also designed to deal with class imbalance problems, on 95 imbalanced datasets in Keel repository. We directly compared

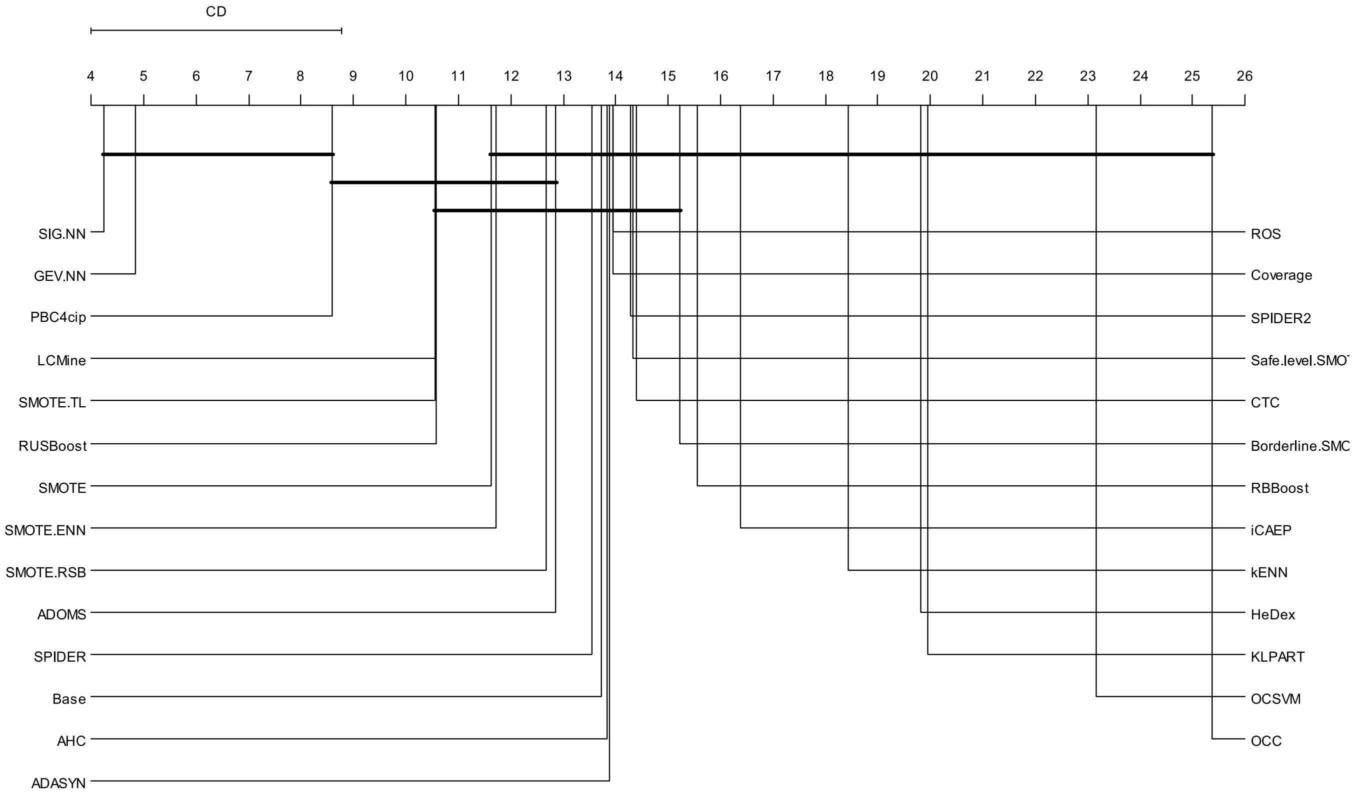

**Fig. 8.** CD plot with statistical comparison of the AUC for comparison between contrast pattern-based classifier (PBC4cip), other state-of-the-art models and GEV-NN.

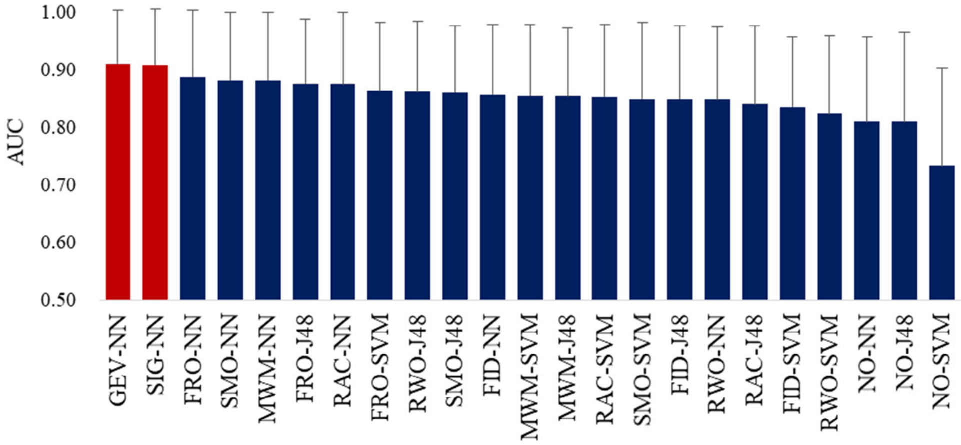

**Fig. 9.** The performance comparison between fuzzy rule-based oversampling technique and GEV-NN.

the results of those models to our proposed model as shown in Fig. 7. Our models showed better performances than the contrast pattern-based classifiers, combinations of resampling methods and the contrast pattern-based classifier, and other state-of-the-art models. GEV-NN outperformed a new contrast pattern-based classifier (PBC4cip) around by 2% in terms of AUC.

We also provided statistical tests for comparison between GEV-NN and other state-of-the-art models including contrast pattern-based classifier (PBC4cip) in Table 3 and Fig. 8. From the statistical tests, it can be observed that our proposed models statistically significantly outperformed PBC4cip and other state-of-the-art models on 95 imbalanced datasets in Keel repository. Only the PBC4cip model has statistically similar behavior with the GEV-NN model.

In the end, we also compare our performances to the fuzzy rule-based oversampling (FRO) technique that creates fuzzy rules from the imbalanced data and assigns each of them a rule weight [40]. FRO synthesizes new minority samples under its guidance of fuzzy rules. They applied this technique to neural network (NN), support vector machine (SVM) and J48 decision tree (J48) models and compared to six resampling techniques such as NO (no sampling), SMO (SMOTE), RAC (RACOG), MWM (Majority Weighted Minority Oversampling Technique), RWO (A random walk oversampling), and FID (fuzzy-based information decomposition). Extensive experiments using 52 real-world imbalanced datasets showed that the FRO technique is better than or comparable with a set of alternative state-of-the-art imbalanced classification algorithms. We also directly compared the performance of the GEV-NN to the results of those models in Fig. 9. Note that they

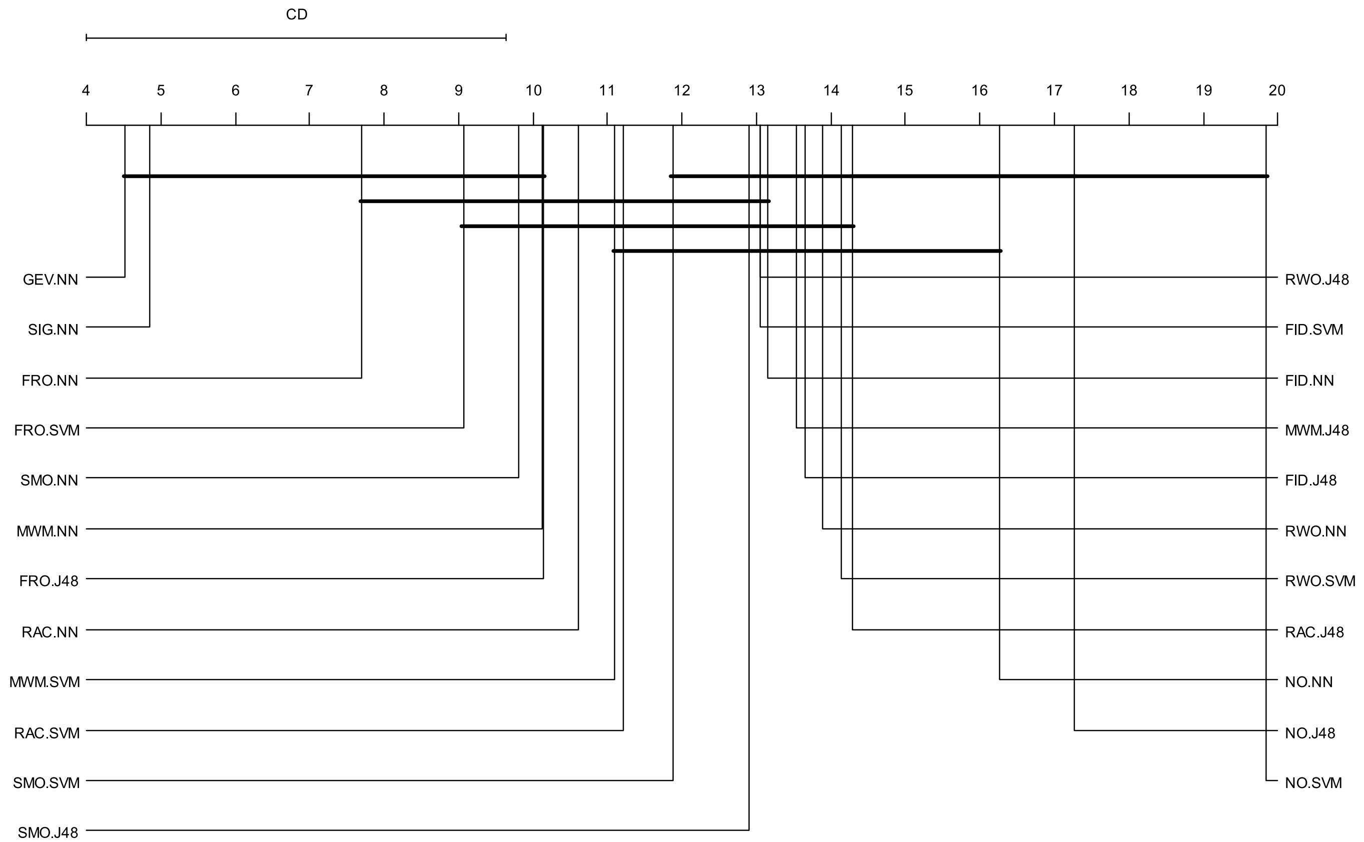

**Fig. 10.** CD plot with statistical comparison of the AUC for comparison between fuzzy rule-based oversampling technique (FRO), other state-of-the-art models and GEV-NN.

**Table 3**
Bergmann–Hommel's test results for the comparison of SIG-NN and GEV-NN (the control algorithm) versus contrast pattern-based classifier (PBC4cip) and other state-of-the-art models on the 95 datasets considering AUC metrics.

| № | Models | SIG-NN | | GEV-NN | | Average rank |
|---|---|---|---|---|---|---|
| | | Adjusted *p*-value | Hypothesis | Adjusted *p*-value | Hypothesis | |
| 1 | SIG-NN | NA | | 0.7254 | Not rejected | 4.25 |
| 2 | GEV-NN | 0.7254 | Not rejected | NA | | 4.84 |
| 3 | PBC4cip | 0.0378 | Rejected | 0.0844 | Not rejected | 8.60 |
| 4 | LCMine | 0.0000 | Rejected | 0.0000 | Rejected | 10.55 |
| 5 | SMOTE.TL | 0.0000 | Rejected | 0.0000 | Rejected | 10.55 |
| 6 | RUSBoost | 0.0000 | Rejected | 0.0002 | Rejected | 10.58 |
| 7 | SMOTE | 0.0000 | Rejected | 0.0000 | Rejected | 11.62 |
| 8 | SMOTE.ENN | 0.0000 | Rejected | 0.0000 | Rejected | 11.72 |
| 9 | SMOTE.RSB | 0.0000 | Rejected | 0.0000 | Rejected | 12.68 |
| 10 | ADOMS | 0.0000 | Rejected | 0.0000 | Rejected | 12.86 |
| 11 | SPIDER | 0.0000 | Rejected | 0.0000 | Rejected | 13.55 |
| 12 | Base | 0.0000 | Rejected | 0.0000 | Rejected | 13.73 |
| 13 | AHC | 0.0000 | Rejected | 0.0000 | Rejected | 13.84 |
| 14 | ADASYN | 0.0000 | Rejected | 0.0000 | Rejected | 13.87 |
| 15 | ROS | 0.0000 | Rejected | 0.0000 | Rejected | 13.94 |
| 16 | Coverage | 0.0000 | Rejected | 0.0000 | Rejected | 13.95 |
| 17 | SPIDER2 | 0.0000 | Rejected | 0.0000 | Rejected | 14.28 |
| 18 | Safe.level.SMOTE | 0.0000 | Rejected | 0.0000 | Rejected | 14.33 |
| 19 | CTC | 0.0000 | Rejected | 0.0000 | Rejected | 14.39 |
| 20 | Borderline.SMOTE | 0.0000 | Rejected | 0.0000 | Rejected | 15.21 |
| 21 | RBBoost | 0.0000 | Rejected | 0.0000 | Rejected | 15.55 |
| 22 | iCAEP | 0.0000 | Rejected | 0.0000 | Rejected | 16.38 |
| 23 | kENN | 0.0000 | Rejected | 0.0000 | Rejected | 18.44 |
| 24 | HeDex | 0.0000 | Rejected | 0.0000 | Rejected | 19.81 |
| 25 | KLPART | 0.0000 | Rejected | 0.0000 | Rejected | 19.96 |
| 26 | OCSVM | 0.0000 | Rejected | 0.0000 | Rejected | 23.16 |
| 27 | OCC | 0.0000 | Rejected | 0.0000 | Rejected | 25.36 |

used a total of 52 imbalanced datasets; 6 of them retrieved from UCI machine learning repository, and the rest of datasets come from the Keel dataset repository. We trained our GEV-NN on 48 of them because other 4 datasets are multi-class classification

**Table 4**
Bergmann–Hommel's test results for the comparison of SIG-NN and GEV-NN (the control algorithm) versus fuzzy rule-based oversampling technique and other state-of-the-art models on the 48 datasets considering AUC metrics.

| № | Models | SIG-NN | | GEV-NN | | Average rank |
|---|---|---|---|---|---|---|
| | | Adjusted *p*-value | Hypothesis | Adjusted *p*-value | Hypothesis | |
| 1 | GEV-NN | 0.9253 | Not rejected | NA | | 4.52 |
| 2 | SIG-NN | NA | | 0.9253 | Not rejected | 4.85 |
| 3 | FRO.NN | 0.0255 | Rejected | 0.0199 | Rejected | 7.70 |
| 4 | FRO.SVM | 0.0002 | Rejected | 0.0001 | Rejected | 9.06 |
| 5 | SMO.NN | 0.0027 | Rejected | 0.0020 | Rejected | 9.80 |
| 6 | MWM.NN | 0.0030 | Rejected | 0.0022 | Rejected | 10.13 |
| 7 | FRO.J48 | 0.0003 | Rejected | 0.0002 | Rejected | 10.14 |
| 8 | RAC.NN | 0.0006 | Rejected | 0.0005 | Rejected | 10.60 |
| 9 | MWM.SVM | 0.0000 | Rejected | 0.0000 | Rejected | 11.09 |
| 10 | RAC.SVM | 0.0000 | Rejected | 0.0000 | Rejected | 11.21 |
| 11 | SMO.SVM | 0.0000 | Rejected | 0.0000 | Rejected | 11.88 |
| 12 | SMO.J48 | 0.0000 | Rejected | 0.0000 | Rejected | 12.91 |
| 13 | RWO.J48 | 0.0000 | Rejected | 0.0000 | Rejected | 13.05 |
| 14 | FID.SVM | 0.0000 | Rejected | 0.0000 | Rejected | 13.05 |
| 15 | FID.NN | 0.0000 | Rejected | 0.0000 | Rejected | 13.15 |
| 16 | MWM.J48 | 0.0000 | Rejected | 0.0000 | Rejected | 13.54 |
| 17 | FID.J48 | 0.0000 | Rejected | 0.0000 | Rejected | 13.65 |
| 18 | RWO.NN | 0.0000 | Rejected | 0.0000 | Rejected | 13.89 |
| 19 | RWO.SVM | 0.0000 | Rejected | 0.0000 | Rejected | 14.14 |
| 20 | RAC.J48 | 0.0000 | Rejected | 0.0000 | Rejected | 14.28 |
| 21 | NO.NN | 0.0000 | Rejected | 0.0000 | Rejected | 16.26 |
| 22 | NO.J48 | 0.0000 | Rejected | 0.0000 | Rejected | 17.27 |
| 23 | NO.SVM | 0.0000 | Rejected | 0.0000 | Rejected | 19.84 |

**Table 5**
The performance comparison of GEV-NN and SIG-NN by AUC.

| № | IR | GEV-NN | SIG-NN |
|---|---|---|---|
| 1 | >40 | 0.82 | **0.83** |
| 2 | 40>, >10 | **0.91** | 0.90 |
| 3 | <10 | **0.91** | 0.91 |

**Table 6**
Hypertension dataset.

| | Weighted N (%) | Women (%) | Men (%) |
|---|---|---|---|
| Age | | | |
| 20–44 | 5313 (39.2) | 2220 (38.9) | 3093 (39.4) |
| 45–64 | 4960 (36.6) | 2040 (35.8) | 2920 (37.2) |
| 65+ | 3272 (24.2) | 1444 (25.3) | 1828 (23.3) |
| Education | | | |
| University | 4369 (32.3) | 2033 (35.6) | 2336 (29.8) |
| High school | 4611 (34) | 2102 (36.9) | 2509 (32) |
| Middle school | 1418 (10.5) | 624 (10.9) | 794 (10.1) |
| Elementary school | 3147 (23.2) | 945 (16.6) | 2202 (28.1) |
| Occupation | | | |
| Office worker | 3056 (22.6) | 1559 (27.3) | 1497 (19.1) |
| Manual worker | 4866 (35.9) | 2536 (44.5) | 2330 (29.7) |
| Unemployed | 5623 (41.5) | 1609 (28.2) | 4014 (51.2) |
| Income level | | | |
| 1st quartile | 3221 (23.8) | 1359 (23.8) | 1862 (23.7) |
| 2nd quartile | 3421 (25.3) | 1448 (25.4) | 1973 (25.2) |
| 3rd quartile | 3454 (25.5) | 1434 (25.1) | 2020 (25.8) |
| 4th quartile | 3449 (25.5) | 1463 (25.6) | 1986 (25.3) |
| Hypertension | | | |
| No | 9463 (69.9) | 3782 (66.3) | 5681 (72.5) |
| Yes | 4082 (30.1) | 1922 (33.7) | 2160 (27.5) |

**Table 7**
Searching space of hyper-parameters.

| Method | Parameters | Search space |
|---|---|---|
| Random forest | n_estimators | [500, 3000] |
| | max_depth | [2, 8] |
| | min_samples_split | [1, 8] |
| | min_samples_leaf | [1, 8] |
| | criterion | {'gini', 'entropy'} |
| | bootstrap | {True, False} |
| Ada Boost | n_estimators | [500, 3000] |
| | learning_rate | [0.1, 1] |
| | algorithm | {'SAMME.R', 'SAMME'} |
| Support vector machine | Penalty parameter C of the error term | {0.001, 0.01, 1, 10, 100, 1000} |
| | Degree of the polynomial kernel function | [1, 5] |
| | Tolerance of stopping criteria | {0.0005, 0.001, 0.01, 0.1} |
| | Kernel | {'rbf', 'poly'} |
| XGBoost | n_estimators | [10, 100] |
| | min_child_weight | [1, 10] |
| | gamma | [0, 1] |
| | subsample | [0.5, 1] |
| | colsample_bytree | [0.5, 1] |
| | max_depth | [2, 8] |
| | learning_rate | [0.01, 0.5] |

datasets. The experimental results show that our model outperformed the FRO with NN model around by 2% in terms of AUC as well.

In addition, the statistical tests confirmed that our proposed GEV-NN is significantly better than the FRO technique with NN, SVM and J48 and others as shown in Table 4 and Fig. 10.

To summarize, GEV-NN indicates significantly better performances than other state-of-the-art models. Although SIG-NN shows the comparable result with GEV-NN, its performances on highly imbalanced datasets are worse than GEV-NN due to the symmetric nature of sigmoid function as shown in Table 5. Then it is proven that our proposed a novel deep neural network architecture with Gumbel activation function is the applicable approach for the class imbalance problem in binary classification. Average AUC and G-mean results obtained by GEV-NN and SIG-NN for each imbalance dataset are reported in Table A.1.

## 3.5. Evaluation on a real-world dataset

### 3.5.1. Hypertension dataset

In this section, we compare GEV-NN to baseline machine learning models on a real-world imbalanced dataset. We collected data from the Korea National Health and Nutrient Examination Survey (KNHANES) to predict hypertension [42]. Hypertension

**Table 8**
The prediction performance for GEV-NN and baseline models.

| Sampling | Models | G-mean | AUC | Accuracy | Brier score | F score |
|---|---|---|---|---|---|---|
| No sampling | LR | 0.761 ± 0.02 | 0.830 ± 0.02 | 0.754 ± 0.02 | 0.150 ± 0.00 | 0.763 ± 0.02 |
| | RF | 0.761 ± 0.02 | 0.833 ± 0.02 | 0.756 ± 0.02 | 0.151 ± 0.00 | 0.764 ± 0.02 |
| | SVM | 0.657 ± 0.07 | 0.666 ± 0.06 | 0.761 ± 0.02 | 0.238 ± 0.02 | 0.737 ± 0.04 |
| | AdaB | 0.753 ± 0.02 | 0.825 ± 0.02 | 0.746 ± 0.02 | 0.154 ± 0.00 | 0.755 ± 0.02 |
| | XGB | 0.759 ± 0.01 | 0.835 ± 0.02 | 0.753 ± 0.02 | 0.148 ± 0.00 | 0.762 ± 0.02 |
| | SIG | 0.763 ± 0.02 | 0.834 ± 0.02 | 0.755 ± 0.02 | 0.150 ± 0.01 | 0.764 ± 0.01 |
| | GEV | 0.761 ± 0.01 | 0.832 ± 0.02 | 0.754 ± 0.01 | 0.154 ± 0.00 | 0.763 ± 0.01 |
| SMOTE | LR | 0.755 ± 0.02 | 0.825 ± 0.02 | 0.746 ± 0.02 | 0.171 ± 0.01 | 0.755 ± 0.02 |
| | RF | 0.749 ± 0.02 | 0.820 ± 0.02 | 0.737 ± 0.03 | 0.168 ± 0.01 | 0.747 ± 0.02 |
| | SVM | 0.738 ± 0.02 | 0.738 ± 0.02 | 0.734 ± 0.03 | 0.265 ± 0.03 | 0.742 ± 0.03 |
| | AdaB | 0.746 ± 0.02 | 0.816 ± 0.02 | 0.739 ± 0.03 | 0.208 ± 0.03 | 0.748 ± 0.02 |
| | XGB | 0.741 ± 0.03 | 0.811 ± 0.02 | 0.733 ± 0.02 | 0.168 ± 0.01 | 0.742 ± 0.02 |
| | SIG | 0.734 ± 0.02 | 0.804 ± 0.02 | 0.722 ± 0.02 | 0.178 ± 0.01 | 0.733 ± 0.02 |
| | GEV | 0.712 ± 0.04 | 0.775 ± 0.05 | 0.704 ± 0.04 | 0.205 ± 0.05 | 0.715 ± 0.04 |
| Borderline-SMOTE | LR | 0.753 ± 0.02 | 0.824 ± 0.02 | 0.744 ± 0.02 | 0.171 ± 0.01 | 0.753 ± 0.02 |
| | RF | 0.751 ± 0.02 | 0.820 ± 0.02 | 0.735 ± 0.03 | 0.168 ± 0.01 | 0.745 ± 0.02 |
| | SVM | 0.741 ± 0.02 | 0.741 ± 0.02 | 0.729 ± 0.03 | 0.270 ± 0.03 | 0.739 ± 0.03 |
| | AdaB | 0.744 ± 0.02 | 0.815 ± 0.02 | 0.740 ± 0.02 | 0.163 ± 0.01 | 0.749 ± 0.02 |
| | XGB | 0.744 ± 0.02 | 0.813 ± 0.02 | 0.733 ± 0.03 | 0.166 ± 0.01 | 0.743 ± 0.02 |
| | SIG | 0.695 ± 0.07 | 0.757 ± 0.09 | 0.687 ± 0.07 | 0.214 ± 0.07 | 0.699 ± 0.06 |
| | GEV | 0.734 ± 0.02 | 0.803 ± 0.03 | 0.723 ± 0.02 | 0.179 ± 0.02 | 0.734 ± 0.02 |
| SVM-SMOTE | LR | 0.756 ± 0.02 | 0.827 ± 0.02 | 0.751 ± 0.02 | 0.169 ± 0.01 | 0.759 ± 0.02 |
| | RF | 0.753 ± 0.02 | 0.826 ± 0.02 | 0.743 ± 0.02 | 0.165 ± 0.01 | 0.752 ± 0.02 |
| | SVM | 0.750 ± 0.02 | 0.750 ± 0.02 | 0.732 ± 0.03 | 0.267 ± 0.03 | 0.742 ± 0.03 |
| | AdaB | 0.753 ± 0.02 | 0.826 ± 0.02 | 0.746 ± 0.02 | 0.158 ± 0.01 | 0.755 ± 0.02 |
| | XGB | 0.755 ± 0.02 | 0.827 ± 0.02 | 0.744 ± 0.02 | 0.154 ± 0.01 | 0.753 ± 0.02 |
| | SIG | 0.744 ± 0.02 | 0.817 ± 0.02 | 0.733 ± 0.02 | 0.176 ± 0.02 | 0.743 ± 0.02 |
| | GEV | 0.751 ± 0.02 | 0.819 ± 0.02 | 0.738 ± 0.02 | 0.174 ± 0.02 | 0.748 ± 0.02 |
| SMOTENC | LR | 0.755 ± 0.02 | 0.826 ± 0.02 | 0.748 ± 0.02 | 0.168 ± 0.01 | 0.757 ± 0.01 |
| | RF | 0.754 ± 0.02 | 0.824 ± 0.02 | 0.744 ± 0.02 | 0.165 ± 0.01 | 0.753 ± 0.02 |
| | SVM | 0.746 ± 0.01 | 0.746 ± 0.01 | 0.735 ± 0.03 | 0.264 ± 0.03 | 0.744 ± 0.02 |
| | AdaB | 0.753 ± 0.02 | 0.822 ± 0.02 | 0.741 ± 0.02 | 0.160 ± 0.01 | 0.750 ± 0.02 |
| | XGB | 0.751 ± 0.02 | 0.821 ± 0.02 | 0.745 ± 0.02 | 0.162 ± 0.01 | 0.754 ± 0.02 |
| | SIG | 0.719 ± 0.04 | 0.781 ± 0.05 | 0.718 ± 0.03 | 0.242 ± 0.11 | 0.728 ± 0.03 |
| | GEV | 0.740 ± 0.03 | 0.808 ± 0.03 | 0.729 ± 0.03 | 0.196 ± 0.06 | 0.739 ± 0.03 |
| ADASYN | LR | 0.754 ± 0.02 | 0.825 ± 0.02 | 0.743 ± 0.02 | 0.173 ± 0.01 | 0.752 ± 0.02 |
| | RF | 0.751 ± 0.02 | 0.818 ± 0.02 | 0.732 ± 0.02 | 0.170 ± 0.01 | 0.743 ± 0.02 |
| | SVM | 0.736 ± 0.02 | 0.736 ± 0.02 | 0.728 ± 0.03 | 0.271 ± 0.03 | 0.737 ± 0.03 |
| | AdaB | 0.744 ± 0.02 | 0.815 ± 0.02 | 0.736 ± 0.03 | 0.224 ± 0.02 | 0.745 ± 0.02 |
| | XGB | 0.739 ± 0.02 | 0.810 ± 0.03 | 0.728 ± 0.03 | 0.168 ± 0.01 | 0.738 ± 0.02 |
| | SIG | 0.700 ± 0.09 | 0.756 ± 0.12 | 0.684 ± 0.10 | 0.223 ± 0.11 | 0.695 ± 0.10 |
| | GEV | 0.736 ± 0.03 | 0.803 ± 0.03 | 0.722 ± 0.02 | 0.179 ± 0.02 | 0.733 ± 0.02 |
| ROS | LR | 0.759 ± 0.02 | 0.830 ± 0.02 | 0.750 ± 0.02 | 0.170 ± 0.01 | 0.759 ± 0.02 |
| | RF | 0.759 ± 0.01 | 0.832 ± 0.02 | 0.753 ± 0.02 | 0.169 ± 0.01 | 0.762 ± 0.02 |
| | SVM | 0.756 ± 0.02 | 0.756 ± 0.02 | 0.737 ± 0.03 | 0.262 ± 0.03 | 0.747 ± 0.03 |
| | AdaB | 0.755 ± 0.02 | 0.827 ± 0.02 | 0.744 ± 0.01 | 0.153 ± 0.00 | 0.753 ± 0.01 |
| | XGB | 0.729 ± 0.02 | 0.798 ± 0.02 | 0.714 ± 0.02 | 0.181 ± 0.01 | 0.725 ± 0.02 |
| | SIG | 0.736 ± 0.01 | 0.806 ± 0.02 | 0.728 ± 0.01 | 0.183 ± 0.02 | 0.738 ± 0.01 |
| | GEV | 0.758 ± 0.02 | 0.824 ± 0.02 | 0.748 ± 0.02 | 0.180 ± 0.02 | 0.757 ± 0.02 |
| Ours | SIG-NN | **0.795 ± 0.01** | 0.863 ± 0.02 | **0.776 ± 0.02** | 0.135 ± 0.00 | 0.784 ± 0.02 |
| | GEV-NN | 0.795 ± 0.02 | **0.865 ± 0.02** | 0.785 ± 0.02 | **0.135 ± 0.00** | **0.792 ± 0.02** |

is one of the most common health problems that can lead to severe and life-threatening diseases such as stroke, heart failure, coronary artery disease, etc. [43]. Accordingly, it is a priority for healthcare applications to provide preventative care to users and to reduce users' overall health risks. We then consider accurate prediction of hypertension as well as key risk factors for hypertension using GEV-NN.

We conducted KHANES from 2013 to 2015. Our hypertension dataset consists of 13,545 (30.1% minority classes) individuals and 22 variables including personal records (sex, age, income level, etc.) lifestyle (smoking, exercise, etc.) and nutrition (intake, energy, etc.) variables as shown in Table 6.

### *3.5.2. Baseline models*

The baseline machine learning classifiers include:

LR refers to logistic regression, which have been the most widely used method for binary classification task [44].

RF refers to random forest classification [45], which is ensemble learning method defined as an aggregation of a multiple decision tree classifiers.

AdaB refers to AdaBoost classification [46], which is boosting algorithm that focuses on classification problems and aims to combine a set of weak classifiers into a strong one. We use base estimator is a Decision tree classification.

XGB refers to XGBoost classification [47], which is a boosting ensemble algorithm; it optimizes the objective of function, size of the tree and the magnitude of the weights are controlled by standard regularization parameters. This method uses Classification and Regression Trees (CART).

SVM refers to Support Vector Machine classification that finds a function, where it has at most $\varepsilon -$ insensitive loss deviation from the actually obtained class label for all the training data [48].

SIG and GEV refer to standard architecture of feed-forward neural network with Sigmoid and Gumbel activation functions,

**Table A.1**
Average AUC and G-mean results obtained by SIG-NN and GEV-NN. The best result for each imbalance dataset appears bold faced.

| Datasets | AUC | | G-mean | | Datasets | AUC | | G-mean | |
|---|---|---|---|---|---|---|---|---|---|
| | SIG-NN | GEV-NN | SIG-NN | GEV-NN | | SIG-NN | GEV-NN | SIG-NN | GEV-NN |
| abalone17vs78910 | **0.9408** | 0.9361 | **0.8892** | 0.8873 | krvskzerovsfteen | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| abalone19 | **0.7850** | 0.7419 | **0.7805** | 0.7247 | led7digit02456789vs1 | **0.9549** | 0.9505 | **0.9298** | 0.9218 |
| abalone19vs10111213 | 0.6291 | **0.7675** | 0.6828 | **0.7647** | lymphography | 0.9612 | **0.9717** | 0.9666 | **0.9741** |
| abalone20vs8910 | **0.9581** | 0.9009 | **0.9225** | 0.8840 | newthyroid1 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| abalone21vs8 | **0.9080** | 0.9057 | 0.8892 | **0.8898** | newthyroid2 | 0.9992 | 0.9992 | 0.9972 | 0.9972 |
| abalone3vs11 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | pageblocks0 | 0.9862 | **0.9890** | 0.9531 | **0.9574** |
| abalone9vs18 | **0.9388** | 0.9246 | **0.9036** | 0.8862 | pageblocks13vs4 | **0.9996** | 0.9940 | **0.9989** | 0.9921 |
| cargood | **0.9685** | 0.9667 | **0.9595** | 0.9545 | pima | 0.8285 | **0.8311** | **0.7748** | 0.7712 |
| carvgood | **0.9957** | 0.9950 | **0.9925** | 0.9924 | poker89vs5 | **0.5812** | 0.4080 | **0.6318** | 0.5165 |
| cleveland0vs4 | **0.9521** | 0.9344 | **0.9429** | 0.9429 | poker89vs6 | 0.8775 | **0.8932** | 0.9022 | **0.9152** |
| dermatology6 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | poker8vs6 | 0.8189 | **0.9660** | 0.8447 | **0.9714** |
| ecoli0137vs26 | **0.8890** | 0.8260 | **0.9326** | 0.8759 | poker9vs7 | **0.8278** | 0.7553 | **0.8585** | 0.8405 |
| ecoli0146vs5 | **0.9452** | 0.9245 | **0.9249** | 0.9026 | segment0 | 0.9999 | 0.9999 | **0.9990** | 0.9982 |
| ecoli0147vs2356 | **0.9382** | 0.9377 | 0.9020 | **0.9135** | shuttle2vs5 | 0.9999 | **1.0000** | 0.9998 | **1.0000** |
| ecoli0147vs56 | **0.9457** | 0.9397 | **0.9395** | 0.9275 | shuttle6vs23 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| ecoli01vs235 | 0.9152 | **0.9309** | 0.8994 | **0.9204** | shuttlec0vsc4 | 0.9953 | **0.9957** | 0.9960 | 0.9960 |
| ecoli01vs5 | **0.9392** | 0.9278 | **0.9429** | 0.9383 | shuttlec2vsc4 | **1.0000** | 0.9920 | **1.0000** | 0.9960 |
| ecoli0234vs5 | **0.9191** | 0.8934 | **0.9135** | 0.9086 | vehicle0 | 0.9940 | **0.9951** | **0.9851** | 0.9836 |
| ecoli0267vs35 | **0.9219** | 0.9215 | **0.9308** | 0.9285 | vehicle1 | **0.8853** | 0.8773 | **0.8225** | 0.8197 |
| ecoli0346vs5 | **0.8959** | 0.8635 | 0.9037 | **0.9215** | vehicle2 | **0.9957** | 0.9924 | **0.9798** | 0.9782 |
| ecoli0347vs56 | 0.9205 | **0.9395** | 0.9316 | **0.9457** | vehicle3 | 0.8490 | **0.8548** | **0.8040** | 0.8013 |
| ecoli034vs5 | 0.9146 | **0.9444** | 0.9424 | **0.9424** | vowel0 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| ecoli046vs5 | **0.9131** | 0.9022 | **0.9054** | 0.8850 | winequalityred3vs5 | **0.6683** | 0.6588 | 0.7564 | **0.7582** |
| ecoli067vs35 | **0.9185** | 0.9038 | **0.9400** | 0.9190 | winequalityred4 | **0.7721** | 0.7609 | **0.7542** | 0.7325 |
| ecoli067vs5 | 0.8900 | **0.9038** | 0.9222 | **0.9324** | winequalityred8vs6 | 0.8195 | **0.9040** | 0.8151 | **0.8970** |
| ecoli0vs1 | **0.9954** | 0.9905 | **0.9866** | 0.9831 | winequalityred8vs67 | **0.7412** | 0.6755 | **0.7725** | 0.7383 |
| ecoli1 | 0.9436 | **0.9539** | **0.9014** | 0.8965 | winequalitywhite39vs5 | 0.6959 | **0.7334** | 0.7403 | **0.7679** |
| ecoli2 | 0.9485 | **0.9503** | 0.9295 | **0.9348** | winequalitywhite3vs7 | 0.7520 | **0.7898** | 0.7223 | **0.7890** |
| ecoli3 | 0.9239 | **0.9243** | **0.9299** | 0.9160 | winequalitywhite9vs4 | **0.7184** | 0.5797 | **0.8220** | 0.7350 |
| ecoli4 | 0.9337 | **0.9520** | 0.9393 | **0.9441** | wisconsin | 0.9947 | **0.9929** | 0.9791 | **0.9834** |
| flareF | **0.9106** | 0.9038 | **0.8771** | 0.8768 | yeast0256vs3789 | **0.8464** | 0.8155 | **0.8001** | 0.7823 |
| glass0 | **0.8564** | 0.8381 | **0.8246** | 0.8175 | yeast02579vs368 | **0.9338** | 0.9313 | 0.9184 | **0.9282** |
| glass0123vs456 | 0.9692 | **0.9747** | 0.9653 | **0.9682** | yeast0359vs78 | 0.7805 | **0.8065** | 0.7548 | **0.7730** |
| glass0146vs2 | 0.7308 | **0.7876** | 0.7628 | **0.8313** | yeast05679vs4 | **0.8650** | 0.8483 | **0.8466** | 0.8288 |
| glass015vs2 | **0.6801** | 0.5366 | **0.7413** | 0.6179 | yeast1 | **0.8040** | 0.8010 | **0.7419** | 0.7402 |
| glass016vs2 | 0.6395 | **0.7164** | 0.6748 | **0.7652** | yeast1289vs7 | 0.8125 | **0.8200** | 0.7695 | **0.7967** |
| glass016vs5 | 0.9914 | 0.9914 | 0.9943 | 0.9943 | yeast1458vs7 | **0.7166** | 0.6818 | **0.7183** | 0.7065 |
| glass04vs5 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | yeast1vs7 | 0.8105 | **0.8285** | **0.7920** | 0.7848 |
| glass06vs5 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | yeast2vs4 | 0.9408 | **0.9483** | 0.9184 | **0.9223** |
| glass1 | 0.8153 | **0.8295** | **0.7978** | 0.7915 | yeast2vs8 | 0.7345 | **0.7922** | 0.7584 | **0.7894** |
| glass2 | 0.6231 | **0.6976** | 0.7066 | **0.7636** | yeast3 | 0.9699 | **0.9703** | 0.9306 | **0.9318** |
| glass4 | 0.9373 | **0.9623** | 0.9408 | **0.9773** | yeast4 | 0.8897 | **0.8930** | **0.8746** | 0.8591 |
| glass5 | **0.9976** | 0.9756 | **0.9975** | 0.9850 | yeast5 | 0.9901 | 0.9901 | 0.9828 | **0.9850** |
| glass6 | 0.9036 | **0.9553** | 0.8981 | **0.9331** | yeast6 | 0.9304 | **0.9441** | 0.9146 | 0.9146 |
| haberman | **0.6629** | 0.6530 | **0.6945** | 0.6762 | zoo3 | **0.9053** | 0.7579 | **0.9480** | 0.8567 |
| iris0 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | **Sonar** | 0.6958 | **0.7329** | 0.6825 | **0.7200** |
| krvskonevsfteen | 1.0000 | 1.0000 | 1.0000 | 1.0000 | **Bupa** | **0.7676** | 0.7655 | 0.7273 | **0.7492** |
| krvskthreevseleven | 1.0000 | 1.0000 | **1.0000** | 0.9998 | **Iono** | 0.9337 | **0.9464** | 0.9048 | **0.9079** |
| krvskzeroonevsdraw | 0.9987 | 0.9987 | **0.9970** | 0.9966 | **Vert2** | 0.9012 | **0.9048** | **0.8873** | 0.8703 |
| krvskzerovseight | 1.0000 | 1.0000 | 1.0000 | 1.0000 | **Park** | 0.7566 | **0.8040** | 0.7359 | **0.7993** |

respectively. We use exactly same architecture to the prediction network of GEV-NN. The hyper-parameters of these baseline classifiers are optimized by random search with 5 cross-validation methods over parameter settings as shown in Table 7.

In addition, we apply the most widely used re-sampling techniques with machine learning baselines on the hypertension dataset. The resampling techniques include:

SMOTE: Synthetic Minority Oversampling Technique, which is the most popular method in this area, generates synthetic samples for the minority class by using k-nearest neighbor (KNN) algorithm [5].

Borderline-SMOTE: Borderline Synthetic Minority Oversampling Technique [49] only oversamples the instances that near the borderline of minority class by using SMOTE technique.

SVM-SMOTE: SVM-SMOTE [50] is the variant of SMOTE algorithm that use the SVM approach to determine sample to use for generating synthetic instances.

SMOTENC: Synthetic Minority Over-sampling Technique for Nominal and Continuous is similar with SMOTE, but it generates synthetic samples of the minority class for dataset containing continuous and categorical variables [5].

ADASYN: Adaptive Synthetic Sampling [51] uses a weighted distribution for different minority class instances according to their level of difficulty in learning, where more synthetic data is generated for minority class instances that are harder to learn compared to those minority examples that are easier to learn.

ROS: Random Over Sampling [52] picks an instance from the minority class instances by using random sampling with replacement until dataset is balanced.

### 3.5.3. Prediction results on hypertension dataset

Table 8 compares the performance of classifiers on the hypertension dataset. It is observed that baseline machine learning models underperform GEV-NN and SIG-NN models. Among the machine learning baselines, ensemble algorithms such as Random forest and XGBoost showed better performances on the hypertension dataset. In addition, standard FNN architecture with sigmoid and Gumbel activation functions also achieved higher performances than machine learning models.

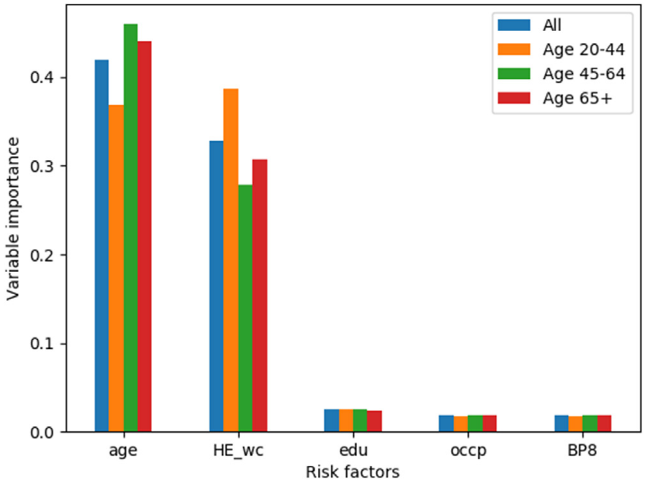

**Fig. 11.** Average importance of top-5 variables is shown by age group on hypertension data using GEV-NN.

For sampling techniques, they have not improved the classification performances compared to no sampling result on the hypertension dataset. ROS and SMOTENC sampling techniques showed better performance than other sampling techniques.

Although our proposed deep neural network architecture (GEV-NN) has achieved the best performance in terms of most evaluation metrics, the sigmoid activation function (SIG-NN) demonstrates comparable performance.

### 3.5.4. *Interpretability of GEV-NN*

The weighting network of GEV-NN gives a beneficial advantage to interpret variable importance. This section describes important variables or risk factors for hypertension using the weighting network. As we explained in Section 2.1.1, the adaptive weights come from the weighting network have a direct

In addition, we can find key risk factors for hypertension, which are consistent with other scientific researches. For example, age (age), education (edu), and occupation (occp) variables are key risk factors for hypertension according to the previous study [43,53]. Andeansah, and Sukandar [54] observed that waist circumference (HE_wc) is positively and significantly correlated with hypertension ($p$-value $< 0.001$) as well as Wang et al. [55] found that excessively longer and shorter periods of sleep (BP8-average sleep duration per day) may both be risk factors for high blood pressure; these associations are stronger in women than men (see Fig. 11).

## 4. Conclusion

In this paper, we proposed a novel end-to-end deep neural network architecture, named GEV-NN, for class imbalance issue in the context of binary classification. We compared our proposed GEV-NN to state-of-the-art models on 100 benchmark imbalanced datasets. The results showed that GEV-NN improves the baselines by up to 2% at most. We also observed that Gumbel activation function works better on extremely imbalanced dataset compared with the standard sigmoid function. In addition, we applied GEV-NN to the real-world hypertension dataset. Our proposed deep learning architecture outperformed baseline state-of-the-art machine learning algorithms as well as sampling methods. We also defined the key risk factors for hypertension consistent with other scientific studies.

## CRediT authorship contribution statement

**Lkhagvadorj Munkhdalai:** Supervision, Writing - original draft, Writing - review & editing. **Tsendsuren Munkhdalai:** Supervision, Writing - original draft, Writing - review & editing. **Keun Ho Ryu:** Supervision, Writing - original draft, Writing - review & editing.

## Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning, Republic of Korea (No. 2017R1A2B4010826 and 2019K2A9A2A06 020672).

## Appendix

See Table A.1.

## References

[1] L. Munkhdalai, T. Munkhdalai, O.E. Namsrai, J.Y. Lee, K.H. Ryu, An empirical comparison of machine-learning methods on bank client credit assessments, Sustainability 11 (3) (2019) 699, http://dx.doi.org/10.3390/su11030699.

[2] L. Munkhdalai, L. Wang, H.W. Park, K.H. Ryu, Advanced neural network approach, its explanation with LIME for credit scoring application, in: Asian Conference on Intelligent Information and Database Systems, ACIIDS, Springer, Cham, 2019, pp. 407–419, http://dx.doi.org/10.1007/978-3-030-14802-7_35.

[3] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, P.J. Kennedy, Training deep neural networks on imbalanced data sets, in: 2016 International Joint Conference on Neural Networks, IJCNN, IEEE, 2016, pp. 4368–4374, http://dx.doi.org/10.1109/IJCNN.2016.7727770.

[4] F. Zhou, S. Yang, H. Fujita, D. Chen, C. Wen, Deep learning fault diagnosis method based on global optimization GAN for unbalanced data, Knowl. Based Syst. 187 (2020) 104837, http://dx.doi.org/10.1016/j.knosys.2019.07.008.

[5] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artificial Intelligence Res. 16 (2002) 321–357, http://dx.doi.org/10.1613/jair.953.

[6] J. Sun, H. Li, H. Fujita, B. Fu, W. Ai, Class-imbalanced dynamic financial distress prediction based on adaboost-SVM ensemble combined with SMOTE and time weighting, Inf. Fusion 54 (2020) 128–144, http://dx.doi.org/10.1016/j.inffus.2019.07.006.

[7] J. Sun, J. Lang, H. Fujita, H. Li, Imbalanced enterprise credit evaluation with DTE-SBD: decision tree ensemble based on SMOTE and bagging with differentiated sampling rates, Inform. Sci. (NY) 425 (2018) 76–91, http://dx.doi.org/10.1016/j.ins.2017.10.017.

[8] X.Y. Liu, J. Wu, Z.H. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Trans. Syst. Man Cybern. B 39 (2) (2008) 539–550, http://dx.doi.org/10.1109/TSMCB.2008.2007853.

[9] M. Lin, K. Tang, X. Yao, Dynamic sampling approach to training neural networks for multiclass imbalance classification, IEEE Trans. Neural Netw. Learn. Syst. 24 (4) (2013) 647–660, http://dx.doi.org/10.1109/TNNLS.2012.2228231.

[10] N. Thai-Nghe, Z. Gantner, L. Schmidt-Thieme, Cost-sensitive learning methods for imbalanced data, in: The 2010 International Joint Conference on Neural Network, IJCNN, IEEE, 2010, pp. 1–8, http://dx.doi.org/10.1109/IJCNN.2010.5596486.

[11] P. Domingos, Metacost: A general method for making classifiers cost-sensitive, in: KDD, Vol. 99, 1999, pp. 155–164, http://dx.doi.org/10.1145/312129.312220.

[12] M. Kukar, I. Kononenko, Cost-sensitive learning with neural networks, in: ECAI, 1998, pp. 445–449.

[13] Z.H. Zhou, X.Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Trans. Knowl. Data Eng. 18 (1) (2005) 63–77, http://dx.doi.org/10.1109/TKDE.2006.17.

[14] C. Zhang, K.C. Tan, H. Li, G.S. Hong, A cost-sensitive deep belief network for imbalanced classification, IEEE Trans. Neural Netw. Learn. Syst. 30 (1) (2018) 109–122, http://dx.doi.org/10.1109/TNNLS.2018.2832648.

[15] V. Raj, S. Magg, S. Wermter, Towards effective classification of imbalanced data with convolutional neural networks, in: IAPR Workshop on Artificial Neural Networks in Pattern Recognition, Springer, Cham, 2016, pp. 150–162, http://dx.doi.org/10.1007/978-3-319-46182-3_13.

[16] X. Wang, D.K. Dey, Generalized extreme value regression for binary response data: An application to B2B electronic payments system adoption, Ann. Appl. Stat. 4 (4) (2010) 2000–2023, http://dx.doi.org/10.1214/10-AOAS354.

[17] A. Agarwal, H. Narasimhan, S. Kalyanakrishnan, S. Agarwal, Gev-canonical regression for accurate binary class probability estimation when one class is rare, in: International Conference on Machine Learning, ICML, 2014, pp. 1989–1997.

[18] P. Embrechts, C. Klüppelberg, T. Mikosch, Modelling Extremal Events: For Insurance and Finance, ninth ed., Springer Science & Business Media, Verlag Berlin Heidelberg, 2013, http://dx.doi.org/10.1007/978-3-642-33483-2.

[19] S. Kotz, S. Nadarajah, Extreme Value Distributions: Theory and Applications, World Scientific, Singapore, 2000, http://dx.doi.org/10.1142/p191.

[20] B. Zong, Q. Song, M.R. Min, W. Cheng, C. Lumezanu, D. Cho, H. Chen, Deep autoencoding gaussian mixture model for unsupervised anomaly detection, in: 2018 the International Conference on Learning Representations, ICLR, 2018, pp. 1–19.

[21] N. Laptev, J. Yosinski, L.E. Li, S. Smyl, Time-series extreme event forecasting with neural networks at uber, in: International Conference on Machine Learning, ICML, 2017, pp. 1–5.

[22] R. Calabrese, S. Osmetti, Generalized extreme value regression for binary rare events data: an application to credit defaults, in: Bulletin of the International Statistical Institute LXII, 58th Session of the International Statistical Institute, 2011, pp. 5631–5634.

[23] L. Munkhdalai, T. Munkhdalai, K.H. Park, T. Amarbayasgalan, E. Erdenebaatar, H.W. Park, K.H. Ryu, An end-to-end adaptive input selection with dynamic weights for forecasting multivariate time series, IEEE Access 7 (2019) 99099–99114, http://dx.doi.org/10.1109/ACCESS.2019.2930069.

[24] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 770–778. http://dx.doi.org/10.1109/CVPR.2016.90.

[25] L. Munkhdalai, T. Munkhdalai, K.H. Park, H.G. Lee, M. Li, K.H. Ryu, Mixture of activation functions with extended min-max normalization for forex market prediction, IEEE Access 7 (2019) 183680–183691, http://dx.doi.org/10.1109/ACCESS.2019.2959789.

[26] J. Schmidhuber, Learning to control fast-weight memories: An alternative to dynamic recurrent networks, Neural Comput. 4 (1) (1992) 131–139, http://dx.doi.org/10.1162/neco.1992.4.1.131.

[27] T. Munkhdalai, H. Yu, Meta networks, in: Proceedings of the 34th International Conference on Machine Learning, ICML, Vol. 70, 2017, pp. 2554–2563.

[28] T. Munkhdalai, A. Sordoni, T. Wang, A. Trischler, Metalearned neural memory, in: Advances in Neural Information Processing Systems, NIPS, Vol. 32, 2019, pp. 13310–13321.

[29] T. Munkhdalai, X. Yuan, S. Mehri, A. Trischler, Rapid adaptation with conditionally shifted neurons, in: Proceedings of the 35th International Conference on Machine Learning, ICML, 2017, pp. 1–10.

[30] G.E. Hinton, D.C. Plaut, Using fast weights to deblur old memories, in: Proceedings of the Ninth Annual Conference of the Cognitive Science Society, Cogsci, 1987, pp. 177–186.

[31] E.J. Gumbel, The return period of flood flows, Ann. Math. Stat. 12 (2) (1941) 163–190, http://dx.doi.org/10.1214/aoms/1177731747.

[32] K. Cooray, Generalized gumbel distribution, J. Appl. Stat. 37 (1) (2010) 171–179, http://dx.doi.org/10.1080/02664760802698995.

[33] H. Zhang, G. Liu, L. Pan, K. Meng, J. Li, GEV regression with convex loss applied to imbalanced binary classification, in: 2016 IEEE First International Conference on Data Science in Cyberspace, DSC, IEEE, 2016, pp. 532–537, http://dx.doi.org/10.1109/DSC.2016.88.

[34] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. 13 (3) (2009) 307–318, http://dx.doi.org/10.1007/s00500-008-0323-y.

[35] J. Huang, C.X. Ling, Using AUC and accuracy in evaluating learning algorithms, IEEE Trans. Knowl. Data Eng. 17 (3) (2005) 299–310, http://dx.doi.org/10.1109/TKDE.2005.50.

[36] M. Kubat, R. Holte, S. Matwin, Learning when negative examples abound, in: European Conference on Machine Learning, ECML, Springer, Berlin, Heidelberg, 1997, pp. 146–153, http://dx.doi.org/10.1007/3-540-62858-4_79.

[37] A. Fernández, S. García, J. Luengo, E. Bernadó-Mansilla, F. Herrera, Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study, IEEE Trans. Evol. Comput. 14 (6) (2010) 913–941, http://dx.doi.org/10.1109/TEVC.2009.2039140.

[38] O. Loyola-González, J.F. Martínez-Trinidad, J.A. Carrasco-Ochoa, M. García-Borroto, Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases, Neurocomputing 175 (2016) 935–947, http://dx.doi.org/10.1016/j.neucom.2015.04.120.

[39] O. Loyola-González, M.A. Medina-Pérez, J.F. Martínez-Trinidad, J.A. Carrasco-Ochoa, R. Monroy, M. García-Borroto, PBC4cip: A new contrast pattern-based classifier for class imbalance problems, Knowl. Based Syst. 115 (2017) 100–109, http://dx.doi.org/10.1016/j.knosys.2016.10.018.

[40] G. Liu, Y. Yang, B. Li, Fuzzy rule-based oversampling technique for imbalanced and incomplete data learning, Knowl. Based Syst. 158 (2018) 154–174, http://dx.doi.org/10.1016/j.knosys.2018.05.044.

[41] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Inform. Sci. 180 (10) (2010) 2044–2064, http://dx.doi.org/10.1016/j.ins.2009.12.010.

[42] S. Kweon, Y. Kim, M.J. Jang, Y. Kim, K. Kim, S. Choi, C. Chun, Y.H. Khang, K. Oh, Data resource profile: the Korea national health and nutrition examination survey (KNHANES), Int. J. Epidemiol. 43 (1) (2014) 69–77, http://dx.doi.org/10.1093/ije/dyt228.

[43] H.W. Park, E. Batbaatar, D. Li, K.H. Ryu, Risk factors rule mining in hypertension: Korean national health and nutrient examinations survey 2007–2014, in: 2016 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB, IEEE, 2016, pp. 1–4, http://dx.doi.org/10.1109/CIBCB.2016.7758128.

[44] D.R. Cox, The regression analysis of binary sequences, J. R. Stat. Soc. Ser. B Stat. Methodol. 20 (2) (1958) 215–232, http://dx.doi.org/10.1111/j.2517-6161.1958.tb00292.x.

[45] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32, http://dx.doi.org/10.1023/A:101093340.

[46] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Proceedings of International Conference on Machine Learning, ICML, Vol. 96, 1996, pp. 148–156.

[47] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd Acm Sigkdd International Conference on KDD, ACM, 2016, pp. 785–794, http://dx.doi.org/10.1145/2939672.2939785.

[48] C. Cortes, V. Vladimir, Support-vector networks., Mach. Learn. 20 (3) (1995) 273–297, http://dx.doi.org/10.1007/BF00994018.

[49] H. Han, W.Y. Wang, B.H. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: Proceedings of International Conference on Intelligent Computing, ICIC, 2005, pp. 878–887. http://dx.doi.org/10.1007/11538059_9.

[50] H.M. Nguyen, E.W. Cooper, K. Kamei, Borderline over-sampling for imbalanced data classification, in: Proceedings of the Fifth International Workshop on Computational Intelligence & Applications, IWCIM, Vol. 1, 2009, pp. 24–29. http://dx.doi.org/10.1504/IJKESDP.2011.039875.

[51] H. He, Y. Bai, E.A. Garcia, S. Li, ADASYN: Adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 1322–1328, http://dx.doi.org/10.1109/IJCNN.2008.4633969.

[52] G.E. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data., ACM SIGKDD Explor. Newsl. 6 (1) (2004) 20–29, http://dx.doi.org/10.1145/1007730.1007735.

[53] H.W. Park, D. Li, Y. Piao, K.H. Ryu, A hybrid feature selection method to classification and its application in hypertension diagnosis, in: International Conference on Information Technology in Bio-and Medical Informatics, Springer, Cham, 2017, pp. 11–19, http://dx.doi.org/10.1007/978-3-319-64265-9_2.

[54] M. Andeansah, H. Sukandar, Correlation between waist circumference and hypertension in jatinangor, J. Hypertens. 33 (2015) e14, http://dx.doi.org/10.1097/01.hjh.0000469769.81770.95.

[55] Y. Wang, H. Mei, Y.R. Jiang, W.Q. Sun, Y.J. Song, S.J. Liu, F. Jiang, Relationship between duration of sleep and hypertension in adults: a meta-analysis, J. Clin. Sleep Med. 11 (09) (2015) 1047–1056, http://dx.doi.org/10.5664/jcsm.5024.