INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Genetic Algorithm Based Over-Sampling with DNN in Classifying the Imbalanced Data Distribution Problem

S Karthikeyan[1]*, **T Kathirvalavakumar**[2]

1 Research Scholar, Research centre in Computer Science, V.H.N.Senthikumara Nadar College, Virudhunagar, Tamil Nadu, India
2 Associate Professor, Research centre in Computer Science, V.H.N.Senthikumara Nadar College, Virudhunagar, Tamil Nadu, India

OPEN ACCESS

Received: 26-04-2022
Accepted: 13-01-2023
Published: 27-02-2023

Citation: Karthikeyan S, Kathirvalavakumar T (2023) Genetic Algorithm Based Over-Sampling with DNN in Classifying the Imbalanced Data Distribution Problem. Indian Journal of Science and Technology 16(8): 547-556. https://doi.org/10.17485/IJST/v16i8.863

* **Corresponding author**.

rgskarthi@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objective: Data imbalance exists in many real-life applications. In the imbalanced datasets, the minority class data creates a wrong inference during the classification that leads to more misclassification. More research has been done in the past to solve this issue, but as of now there is no global working solution found to do efficient classification. After analyzing various existing literatures, it is proposed to minimize the misclassification through genetic based oversampling and deep neural network (DNN) classifier. **Method:** In the proposed oversampling method synthetic samples are generated based on genetic algorithm. Initial populations for the genetic algorithm are generated using Gaussian weight initialization technique and the fittest individual from the population are selected by Euclidean distance for further processing to generate synthetic data in double the minority class size and the dataset is classified with the DNN. **Findings:** The performance of the oversampled training data with DNN Classifier is compared with C4.5 and Support Vector Machine (SVM) classifiers and found that the DNN classifier outperforms the other two classifiers. The data generated using SMOTE and ADASYN are considered for comparison. It is found that the proposed approach outperforms the other approaches. It is also proved from the experiment that misclassification is reduced and the proposed method is statistically significant and is comparatively better. **Novelty:** Initial population generation by Gaussian weight initialization, the fittest sample selection by Euclidean distance measure, synthetic samples with double the minority class size and DNN for classification to reduce the misclassification is novelty in this work.

Keywords: Genetic algorithm; Gauss weight initialization; SMOTE; ADASYN; Imbalanced data; Classification

1 Introduction

Many data resampling methods and algorithm level methods proposed in the past are able to handle the imbalanced data in a better way but still there is a lack in handling the imbalanced data problems for classification. Wang et al.[1] have discussed different processing methods for the imbalanced dataset problems and its evaluation criteria. The processing methods include data level, algorithm level and, feature level methods, sensitive cost function and deep learning classification. The perception on solving the imbalanced data varied from authors to authors. Li et al.[2] have fine-tuned the parameterized model by iteratively adjusting the class data and found that less number of samples in the training does not results good; but more number of samples in the training produces better results. Douzas et al.[3] have generated synthetic samples with the basic idea of Geometric SMOTE algorithm; the artificial samples are generated around the geometric region of a particular sample and have concluded that resampled data works better only if the artificial sample is similar to the original data. Gnip et al.[4] have demonstrated the selective oversampling method by applying the method namely one sided support vector machine to remove the outlier patterns and used only the representative patterns for oversampling. This helps in controlling the over-fitting and improves the performance of the classifiers but balancing of class samples takes more time in oversampling.

If the initial population of GA is better, then the probability of obtaining good solutions is high[5]. Randomly selected initial population takes more time to converge but heuristics based initial population converges rapidly[6]. Zhou et al.[7] have experimented an automatic clustering of low-dimensional data using k-means algorithm where GA based initial seed pattern is used for the clustering. It is observed from the results that the initial population for GA plays a vital role in faster convergence. The GA based approach helps in maintaining diversity in the population and avoids getting stuck at the local optimal solution. Karia et al.[8] have demonstrated the oversampling approach using GA and focused on increasing the count of minority class samples to a considerable proportion to attain better results. The synthetic sample generated through this approach is fully balanced and not able to achieve better results for some datasets. Ghazikhani et al.[9] have developed a wrapper strategy along with the GA to oversample the minority class. The minority class samples are randomly oversampled and the process continues based on the classifier performance. But this is a time consuming process as the performance of the classifier is evaluated at each instance. Arun and Lakshmi[10] have done experiments on oversampling approach using the genetic algorithm with Mahalanobis distance measure, which helps in attaining better results but the samples in the minority class are oversampled to the size of majority class samples, but balancing the samples consumes more time and space when the imbalance ratio is high.

Angelos and Francois[11] have concluded that DNN classifiers spend lot of time for performing computations on samples that are not significant and that problem can be solved by picking only the representative samples for the training. The Informative samples help in speeding up the training process and achieve better performance. Sungho et al.[12] have incorporated transfer learning and generative adversarial feature augmentation on the imbalanced data problems. But authors have believed that the classes in the balanced form are still better than their proposed method.

Synthetic Minority Over-sampling Technique (SMOTE)[13] and Adaptive Synthetic sampling (ADASYN)[14] algorithms are widely used in over-sampling. Both the algorithms have the ability to mimic the properties of the minority class data[15]. Even though many advancements are made in these existing methods they are able to produce only minor significance than the original version of SMOTE and ADASYN.

On analyzing the existing literature it is clear that the classification of the imbalanced dataset problem still needs significant improvement. To solve this problem it has been planned to use an evolutionary approach. The importance of generating an appropriate weight initialization strategy for the evolutionary algorithm is not addressed in the prior studies. The space complexity for balancing an extreme imbalanced data is too high and is not properly handled in the previous works. Also, prior experiments using GA for oversampling have not generated a lesser number of synthetic samples with significant features. GA is evolutionary in nature, it generates synthetic samples with good fitness; also selecting an appropriate weight initialization strategy for the GA leads to convergence in a lesser number of cycles. In the proposed work, it has been proposed to oversample by generating minimum number of synthetic samples based on GA with a better initial population for the moderate to extreme imbalanced datasets. Euclidean distance measure is used to select good fitness samples. DNN classifier is used for classification to get higher AUC value. Rest of the article is organized as follows: section 2 describes the methodology, section 3 demonstrates the results and discussion and section 4 draws the conclusion.

2 Methodology

It is hard to achieve better classification on an imbalanced datasets. In this regard, it has been planned to solve the imbalanced classification problem using the genetic algorithm. As per the literature, genetic algorithm based resampling provides better solutions than other resampling methods. Initially Gaussian weight initialization method has been adapted to generate initial population for the genetic algorithm. Successively each phase of the genetic algorithm is properly processed to achieve better

results. DNN classifier has the ability to classify the data with better accuracy; it is considered for training the imbalanced dataset with lesser number of synthetic samples of the minority class generated from oversampling.

2.1 Proposed Method

Genetic algorithm is used to generate synthetic samples. Synthetic sample generation process is carried out for every minority sample. This process oversamples the minority class samples to double its size. This process contains the phases namely population, selection, crossover and mutation. Corresponding to each sample in the minority class, synthetic samples are to be generated through an iterative procedure. Gaussian distribution generates samples for the initial population of the synthetic sample generation process. Optimal numbers of features to be mutated is decided by trial and error. Fittest sample of the mating pool is selected as a new synthetic sample when the generation cycle terminates. Euclidean distance measure is used to find the fitness score of the sample. Initial population generated through the Gaussian distribution has the ability to create samples similar to the original samples in the minority class which substantially reduces the number of cycles. Selection of the fittest individual through Euclidean distance helps in choosing the better samples for the crossover and mutation phases.

2.1.1 Initial population

Synthetic sample generation process begins with an initial population generation process. From[5] it is observed that a population with proper initial samples helps in obtaining better results. In the proposed method, Gaussian distribution generates new samples in the synthetic sample generation process and is repeated n times to generate the initial population of size n. For every new sample the mean and standard deviation needed for Gaussian distribution are calculated from the minority class samples.

2.1.2 Fitness function

The fitness function determines the similarity of the synthetic sample with the selected minority sample. Euclidean distance measure is used to find the fitness score. Fitness score obtained from the fitness function decides the samples to be selected for the matting pool. Also this score is used to decide the new synthetic sample for the minority sample selected from the population pool.

2.1.3 Selection

Samples of size n in the population are ranked based on its fitness score and are arranged in ascending order. The top n/2 samples are selected for the matting pool to do crossover. These selected samples are called parents.

2.1.4 Crossover

New samples are formed by exchanging the feature information of the samples in the matting pool. Assume that the samples in the matting pool are in the same order as it is already in the population. Also assume that all samples are in the circular order. Every adjacent two parent samples are doing cross-over and generate a new sample. From every pair, the first half number of features of the first sample and second half number of features of the second sample are brought to form a new sample without changing position of the features.

2.1.5 Mutation

Mutation alters the feature values of the new samples in the matting pool. In every sample 'n' feature values are altered. For each k adjacent features of the resultant sample, one randomly selected feature is mutated by adding a random number generated from [0, 1]. The mutated samples are children. Now the matting pool contains parent samples and its children. It becomes the new population for the next iteration.

2.1.6 Termination

The selection, crossover and mutation phases are repeated for the selected number of times called cycles. The cycle terminates after the specified number of times or there is no change in the samples of the mating pool of the consecutive two cycles.

2.1.7 Synthetic sample

After the termination, the sample of the mating pool with the lowest fitness score is a new synthetic sample for the selected minority class sample. This procedure is repeated for each minority training sample.

2.1.8 Deep neural network

The samples in the problem and generated synthetic samples are used in the deep neural network for classification.

Algorithm

1. Calculate mean and standard deviation for each feature of the minority class samples
2. Specify size (n) of the initial population
3. Specify number of cycles (g) to generate a synthetic sample
4. Every feature of the sample of the initial population is generated by Gaussian distribution using mean and standard deviation of its corresponding feature obtained in step1.
5. Select a sample from the minority class of the training dataset
6. Generate the initial population for the selected sample using step 4
7. Fitness score for each sample of population is calculated by finding Euclidean distance with a selected minority class sample.
8. Arrange the samples of the population in ascending order based on its fitness score
9. Select top n/2 samples for the mating pool (parent samples)
10. Assume samples of the matting pool are in circular order. Select each successive pair for crossover.
11. Do crossover on each pair
12. Do mutation on the samples obtained in step 11 to generate new samples (children
13. The parents and children in the matting pool become a new population for the next cycle.
14. Repeat the steps 7-13specified number (g) of times or until samples of matting pool in the consecutive cycle are unchanged.
15. The sample in the matting pool with the lowest fitness score is a newly generated synthetic sample for the selected minority class sample in step 5.
16. Repeat steps 6-15 for all the training samples of the minority class
17. Apply the samples in the deep neural network for classification

3 Results and Discussion

3.1 Experiment

The experiment considers 33 imbalanced datasets collected from the Keel repository[16], UCI repository[17], and PROMISE repository[18]. Imbalance ratio(IR) of the datasets ranges from 1.82 to 30.57. In this experiment, 60 percent of the majority class data and 60 percent of the minority class data forms the training dataset and the remaining data forms the testing dataset. Previous experiments on oversampling do not resample the data to double the size of the minority class data, so it has been decided to create synthetic samples using the oversampling methods SMOTE and ADASYN. The experimental setup is shown in Figure 1.

For comparative analysis, two state-of-the-art oversampling methods SMOTE and ADASYN are used to generate synthetic samples. Three well known classifiers namely C4.5, Support Vector Machine (SVM) and Deep Neural Network (DNN) are considered for comparison. Area under the Receiver Operating Characteristics Curve (AUC)[19] is used in this experiment as a statistical evaluation metric. The classification accuracy is a best measure when the data are in balanced form but when it is imbalanced, the misclassification of minority class data is not reflected much on the classification accuracy[20]. So it is better to consider statistical measure AUC to know the exact nature of the classification as it gives equal importance to all the classes irrespective of the sample counts of classes.

The experiment is carried out at different times with different initial populations and its size, matting pool size and mutation parameter. The obtained experimental results are shown in the Table 1. Here P4, P8, P12, and P16 represent the initial population size as 4, 8, 12, and 16 respectively. M1, M3 and M5 represent the number of mutations per sample as 1, 3, and 5 respectively. The values bolded in each table represents the best value in each row. Here, the number of cycles per synthetic sample generation for a minority class sample is fixed. From the experiment trials, it is observed that when the initial population size is eight, the mating pool size is 4, and 5 feature values altered per sample during mutation are producing better results in most of the datasets. Further it is noted in the experiment that when the initial number of population is beyond this size and the number of bits considered for mutation is in even number then it does not give a better solution. The cycle is terminated when the samples in the mating pool are the same in the consecutive two cycles. When comparing the proposed method in, it is observed that the performance of DNN classifier is better when the number of cycles per synthetic data generation is not fixed. It is clear from the tables that classifying the imbalanced data using the proposed oversampling method along with the DNN classifier is best suited. It is also observed that the initial population generated through the Gaussian distribution converges faster in terms of

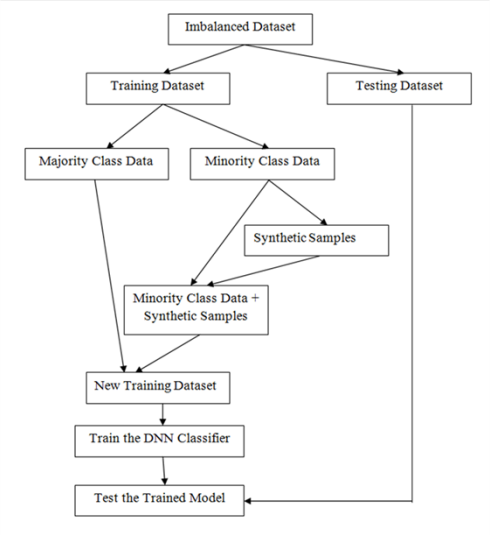

Fig 1. Experimental Setup

Table 1. AUC under different Initial Population and Mutation under DNN Classifier

Dataset	P4-M1	P4-M3	P4-M5	P8-M1	P8-M3	P8-M5	P12-M1	P12-M3	P12-M5	P16-M1	P16-M3	P16-M5
Abalone9-18	0.937	0.956	0.925	0.958	0.92	**0.991**	0.942	0.951	0.937	0.939	0.938	0.95
Ecoli0vs1	0.983	0.982	**0.996**	0.982	0.983	0.987	0.973	0.982	0.986	0.982	0.98	0.982
Ecoli1	0.95	0.944	0.944	0.884	0.949	**0.957**	0.948	0.933	0.931	0.952	0.941	0.945
Ecoli2	0.929	0.932	0.917	0.947	0.903	**0.965**	0.927	0.927	0.929	0.94	0.932	0.941
Glass016vs2	0.728	0.697	0.706	0.716	0.695	**0.789**	0.701	0.716	0.713	0.695	0.681	0.702
Glass1	**0.736**	0.718	0.5	0.516	0.626	0.728	0.682	0.733	0.637	0.678	0.724	0.655
Glass2	0.688	0.639	0.739	0.63	0.631	0.751	0.614	0.709	0.655	0.63	**0.783**	0.752
Haberman	0.716	0.657	0.711	0.742	0.716	0.679	0.74	0.702	0.732	0.726	0.489	**0.77**
New-Thyroid1	0.992	0.998	**1**	**1**	0.998	0.999	0.5	0.997	0.998	0.998	0.991	**1**
New-Thyroid2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Page-blocks0	0.897	0.897	0.972	0.968	0.96	**1**	0.969	0.926	0.971	0.976	0.566	0.919
Pima	0.671	0.702	0.696	0.7	0.707	**0.769**	0.683	0.704	0.713	0.718	0.708	0.72
Shuttlec0vsc4	**1**	0.999	**1**	**1**	0.989	**1**	0.989	**1**	0.989	**1**	0.989	0.999
Vehicle0	0.857	0.5	**0.988**	0.942	0.928	0.982	0.98	0.982	0.954	0.981	0.967	0.964
Vehicle1	0.76	0.746	**0.783**	0.757	0.706	0.781	0.489	0.712	0.69	0.711	0.752	0.661
Vehicle3	0.728	0.745	0.706	0.747	0.523	0.705	0.734	0.699	0.743	0.74	0.745	**0.782**
Vowel0	0.999	**1**	**1**	0.999	0.999	**1**	0.999	**1**	**1**	0.985	**1**	0.999
Wisconsinlmb	0.983	**0.985**	0.965	0.974	0.981	**0.985**	0.968	0.967	0.976	0.98	0.972	0.975
Yeast05679vs4	0.847	0.842	0.839	0.859	0.862	0.855	0.851	0.846	0.851	**0.875**	0.857	0.845
Yeast1	0.81	0.709	0.805	0.796	0.807	**0.847**	0.798	0.788	0.804	0.802	0.813	0.808
Yeast1vs7	0.88	**0.908**	0.865	0.879	0.845	0.851	0.787	0.836	0.84	0.861	0.88	0.806
Yeast1289vs7	0.764	0.767	0.712	0.789	0.792	0.796	0.725	**0.806**	0.802	0.774	0.788	0.805
Yeast2vs4	0.974	0.973	0.949	0.978	0.95	0.972	0.953	0.974	0.978	**0.986**	0.983	0.973
Yeast4	0.836	**0.845**	0.828	0.82	0.828	0.828	0.828	0.82	0.823	0.821	0.835	0.836

cycles to generate a synthetic sample. Instead of Gaussian distribution, other distributions, namely uniform distribution and binomial distribution are also used in generating the initial population, but it has been identified in the trials that Gaussian distribution is best suited for generating initial population than others.

Table 2. AUC under different Initial Population and Mutation under C4.5 Classifier

Dataset	P4-M1	P4-M3	P4-M5	P8-M1	P8-M3	P8-M5	P12-M1	P12-M3	P12-M5	P16-M1	P16-M3	P16-M5
Abalone9-18	0.69	0.75	**0.752**	0.573	0.655	0.592	0.715	0.739	0.729	0.657	0.686	0.684
Ecoli0vs1	0.833	0.86	0.833	0.636	0.893	**0.958**	**0.958**	0.958	0.86	0.803	0.936	0.803
Ecoli1	0.836	0.84	0.867	0.696	0.727	0.858	0.84	0.874	0.863	**0.891**	0.847	0.867
Ecoli2	0.868	0.868	0.85	0.859	0.85	**0.87**	0.868	0.846	0.859	0.846	0.86	0.854
Glass016vs2	0.539	0.514	0.45	0.528	0.528	**0.607**	0.55	0.471	0.464	0.457	0.593	0.45
Glass1	0.664	0.761	0.633	0.719	0.662	**0.767**	0.679	0.723	0.716	0.679	0.717	0.681
Glass2	0.618	0.586	**0.676**	0.579	0.56	0.592	0.502	0.586	0.617	0.573	0.58	0.669
Haberman	0.63	**0.644**	0.595	0.58	0.607	0.597	0.561	0.571	0.59	0.553	0.608	0.566
New-Thyroid1	0.915	0.886	0.866	0.879	0.866	**0.979**	0.886	0.915	0.852	0.943	0.915	0.773
New-Thyroid2	0.886	0.934	0.937	0.794	0.787	**0.956**	0.951	0.947	0.894	0.934	0.934	0.839
Page-blocks0	0.909	0.908	0.924	0.917	0.904	**0.93**	0.921	0.899	0.897	0.907	0.924	0.888
Pima	0.667	0.677	0.685	0.665	0.614	**0.711**	0.655	0.656	0.604	0.656	0.68	0.633
Shuttlec0vsc4	**1**	**1**	0.991	0.986	0.991	**1**	**1**	**1**	0.991	**1**	**1**	0.989
Vehicle0	0.901	0.88	0.868	0.819	0.885	**0.909**	0.904	0.908	0.897	0.903	0.904	0.863
Vehicle1	0.69	0.674	0.706	0.663	0.645	0.687	0.703	0.685	0.678	0.714	0.707	0.681
Vehicle3	0.68	0.662	0.679	0.691	0.687	**0.746**	0.705	0.713	0.681	0.689	0.744	0.728
Vowel0	0.958	0.879	0.895	0.926	0.912	0.929	0.958	0.924	0.902	0.936	**0.96**	0.923
Wisconsinlmb	0.933	0.936	0.922	0.917	0.951	**0.952**	0.933	0.923	0.922	0.92	0.944	0.941
Yeast05679vs4	0.717	0.73	0.718	0.685	0.71	**0.782**	0.678	0.703	0.738	0.71	0.761	0.74
Yeast1	0.651	0.647	0.649	0.646	0.641	0.64	0.641	**0.667**	0.645	0.643	0.636	0.65
Yeast1vs7	0.763	0.719	0.693	0.798	0.718	**0.83**	0.698	0.754	0.715	0.739	0.742	0.75
Yeast1289vs7	0.512	0.713	0.562	0.604	0.562	**0.716**	0.59	0.553	0.515	0.548	0.563	0.56
Yeast2vs4	0.834	0.889	0.858	0.883	0.861	0.884	0.77	0.884	0.861	0.834	**0.926**	0.858
Yeast4	0.671	0.678	**0.683**	0.662	0.662	0.676	0.643	0.631	0.66	0.66	0.641	0.654

Table 3. AUC under different Initial Population andMutation under SVM Classifier

Dataset	P4-M1	P4-M3	P4-M5	P8-M1	P8-M3	P8-M5	P12-M1	P12-M3	P12-M5	P16-M1	P16-M3	P16-M5
Abalone9-18	0.558	0.588	**0.717**	0.5	0.607	0.588	0.588	0.588	0.57	0.588	0.588	0.593
Ecoli0vs1	**0.984**	0.969	0.924	0.893	0.9	**0.984**	**0.984**	**0.984**	0.878	**0.984**	0.969	0.848
Ecoli1	0.89	0.901	0.839	0.893	0.818	**0.906**	0.895	0.89	0.828	0.895	0.895	0.828
Ecoli2	0.934	**0.939**	0.88	0.851	0.864	**0.939**	**0.939**	**0.939**	0.829	**0.939**	**0.939**	0.842
Glass016vs2	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**
Glass1	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**
Glass2	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**
Haberman	0.618	0.612	0.5	0.5	**0.65**	0.612	0.612	0.618	0.5	0.612	0.618	0.5
New-Thyroid1	0.678	0.678	**0.866**	0.513	0.52	0.678	0.678	0.678	0.506	0.678	0.678	0.506
New-Thyroid2	**0.678**	**0.678**	0.506	0.513	0.541	**0.678**	**0.678**	**0.678**	0.506	**0.678**	**0.678**	0.5
Page-blocks0	0.523	0.523	0.547	0.593	0.596	0.523	0.525	0.525	0.61	0.526	0.521	0.609
Pima	**0.699**	0.695	0.552	0.502	0.51	**0.699**	0.694	0.699	0.515	**0.699**	0.696	0.505
Shuttlec0vsc4	**0.999**	**0.999**	0.997	0.995	0.994	**0.999**	**0.999**	**0.999**	0.996	**0.999**	**0.999**	0.994
Vehicle0	0.822	0.813	0.679	0.681	0.673	0.815	0.812	**0.826**	0.667	0.82	0.774	0.669
Vehicle1	0.651	0.647	0.567	0.527	0.53	**0.654**	0.652	0.652	0.516	0.642	0.646	0.526
Vehicle3	0.695	0.695	0.532	0.537	0.537	0.685	**0.701**	0.693	0.525	0.685	0.693	0.533

Continued on next page

Table 3 continued

Vowel0	0.902	0.901	0.78	0.792	0.846	**0.908**	0.906	0.905	0.821	0.902	0.901	0.852
Wisconsinlmb	0.97	0.97	0.963	0.957	0.96	**0.973**	0.97	0.97	0.946	0.97	0.97	0.949
Yeast05679vs4	0.773	**0.779**	0.628	0.613	0.564	0.776	**0.779**	**0.779**	0.611	0.776	**0.779**	0.607
Yeast1	0.691	0.691	0.594	0.594	0.594	**0.703**	0.692	0.688	0.58	0.689	0.691	0.597
Yeast1vs7	0.541	0.541	0.484	0.479	0.624	0.583	**0.625**	0.583	0.573	0.541	**0.625**	0.583
Yeast1289vs7	**0.5**	**0.5**	0.395	0.382	0.412	**0.5**	**0.5**	**0.5**	0.453	**0.5**	**0.5**	0.477
Yeast2vs4	0.716	**0.722**	0.553	0.542	0.593	**0.722**	**0.722**	0.719	0.564	0.719	0.719	0.547
Yeast4	0.722	0.723	0.638	0.627	0.602	**0.727**	0.715	0.72	0.611	0.723	0.723	0.617

Table 4. AUC score without added Noise

Dataset	C4.5			SVM			DNN		
	Proposed	**SMOTE**	**ADASYN**	**Proposed**	**SMOTE**	**ADASYN**	**Proposed**	**SMOTE**	**ADASYN**
Abalone9-18	0.592	**0.686**	0.658	0.588	**0.6158**	0.558	**0.991**	0.939	0.95
Ecoli0vs1	**0.958**	**0.958**	**0.958**	**0.984**	**0.984**	0.957	**0.987**	0.983	0.983
Ecoli1	**0.858**	0.847	0.826	**0.906**	0.879	0.887	**0.957**	0.941	0.928
Ecoli2	**0.87**	0.87	0.861	0.939	**0.943**	0.93	**0.965**	0.921	0.916
Glass016vs2	**0.607**	0.457	0.5	**0.5**	**0.5**	**0.5**	**0.789**	0.721	0.706
Glass1	**0.767**	0.692	0.747	**0.5**	**0.5**	**0.5**	**0.728**	0.5	0.653
Glass2	0.592	0.618	**0.729**	**0.5**	**0.5**	**0.5**	**0.751**	0.715	0.652
Haberman	0.597	0.609	**0.62**	0.612	**0.618**	0.612	0.679	0.5	**0.721**
New-Thyroid1	**0.979**	0.915	0.908	**0.678**	**0.678**	**0.678**	**0.999**	0.5	0.998
New-Thyroid2	0.956	0.886	0.964	**0.678**	**0.678**	**0.678**	**1**	**1**	**1**
Page-blocks0	**0.93**	0.899	0.91	**0.523**	0.519	0.532	**1**	0.923	0.892
Pima	**0.711**	0.684	0.665	0.699	**0.701**	0.69	**0.769**	0.726	0.698
Shuttlec0vsc4	**1**	1	**1**	**0.999**	0.979	0.867	**1**	0.989	0.989
Vehicle0	0.909	0.897	**0.929**	0.815	0.822	**0.823**	**0.982**	0.977	0.504
Vehicle1	**0.687**	0.657	0.667	**0.654**	0.642	0.646	**0.781**	0.731	0.732
Vehicle3	**0.746**	0.687	0.723	0.685	0.689	**0.695**	0.705	0.744	**0.77**
Vowel0	0.929	0.967	0.919	0.908	**0.93**	0.872	**1**	**1**	**1**
Wisconsinlmb	0.952	0.957	0.954	0.973	0.972	**0.977**	**0.985**	0.968	0.957
Yeast05679vs4	**0.782**	0.755	0.74	0.776	**0.779**	0.759	0.855	**0.873**	0.851
Yeast1	0.64	0.665	**0.672**	0.703	**0.715**	0.704	**0.847**	0.804	0.796
Yeast1vs7	**0.83**	0.742	0.703	**0.583**	0.541	**0.583**	**0.851**	0.8	0.784
Yeast1289vs7	**0.716**	0.65	0.59	**0.5**	**0.5**	**0.5**	0.796	**0.812**	0.784
Yeast2vs4	**0.884**	0.861	0.861	0.722	**0.919**	0.916	0.972	0.96	**0.988**
Yeast4	0.676	0.673	**0.715**	0.727	0.72	**0.732**	0.828	0.815	**0.833**

Table 5. AUC score with added Noise

Dataset	C45			SVM			DNN		
	Proposed	**SMOTE**	**ADASYN**	**Proposed**	**SMOTE**	**ADASYN**	**Proposed**	**SMOTE**	**ADASYN**
Abalone9-18	0.657	**0.673**	0.629	**0.948**	**0.948**	0.558	**0.999**	0.955	0.956
Ecoli0vs1	**0.958**	0.864	0.939	**0.984**	0.977	0.969	**0.999**	0.983	0.982
Ecoli1	0.863	0.858	**0.889**	**0.895**	0.842	0.882	0.93	**0.943**	0.925
Ecoli2	0.892	0.85	**0.896**	**0.962**	0.939	0.934	0.921	0.901	**0.924**
Glass016vs2	**0.679**	0.564	0.586	**0.5**	**0.5**	**0.5**	**0.704**	0.667	0.693
Glass1	**0.702**	0.658	0.658	0.352	**0.5**	**0.5**	0.631	0.53	**0.677**
Glass2	0.638	0.672	**0.741**	0.5	**0.5**	**0.5**	**0.752**	0.744	0.658
Haberman	0.586	0.574	**0.635**	**0.77**	0.72	0.618	**0.728**	0.5	0.661
New-Thyroid1	0.857	**0.886**	**0.886**	**0.678**	**0.678**	0.642	**1**	0.5	**1**
New-Thyroid2	**0.879**	0.85	0.85	**0.678**	**0.678**	**0.678**	**1**	**1**	**1**
Page-blocks0	**0.901**	0.851	0.774	0.525	**0.55**	0.532	**0.966**	0.915	0.925
Pima	**0.682**	0.648	0.66	**0.73**	0.699	0.693	**0.716**	0.7	0.712
Shuttlec0vsc4	**1**	**1**	**1**	**0.999**	0.979	0.982	**0.999**	0.989	0.989

Continued on next page

Table 5 continued

Vehicle0	**0.898**	0.692	0.75	0.825	**0.837**	0.83	0.483	0.922	**0.971**
Vehicle1	0.689	0.704	**0.734**	**0.663**	0.64	0.648	0.729	0.764	**0.804**
Vehicle3	**0.743**	0.715	0.713	**0.704**	0.689	0.693	0.738	**0.77**	0.693
Vowel0	**0.974**	0.947	0.951	0.908	0.93	**0.952**	**1**	**1**	**1**
Wisconsinlmb	**0.947**	0.931	0.692	0.973	0.972	**0.977**	**0.972**	0.969	0.955
Yeast05679vs4	**0.753**	0.511	0.514	**0.776**	0.773	0.748	**0.982**	0.852	0.846
Yeast1	0.623	0.624	**0.634**	0.69	**0.716**	0.706	**0.949**	0.808	0.789
Yeast1vs7	**0.721**	0.638	0.605	**0.72**	0.622	0.583	**0.968**	0.857	0.753
Yeast1289vs7	0.59	0.642	**0.661**	**0.5**	**0.5**	0.5	**0.805**	0.764	0.748
Yeast2vs4	0.906	**0.981**	0.795	0.919	0.975	0.919	0.978	0.988	**0.99**
Yeast4	0.658	**0.68**	0.651	0.72	0.718	0.71	0.831	**0.84**	0.815

Table 6. Comparison of AUC Scores

Dataset	**IR**	**C4.5**		**DNN**		
		GenSample	**Proposed (50% Train + 50% Test)**	**Proposed (50% Train + 50% Test)**	**Proposed (70% Train + 30% Test)**	**Proposed (80% Train + 20% Test)**
Ionosphere	1.8	0.86	0.88	**0.915**	0.904	0.889
Heart	1.8	0.73	0.76	0.775	0.857	**0.867**
Iris	2	0.94	0.95	**1**	**1**	**1**
Parkinson	3.1	0.78	0.758	0.676	0.681	**0.903**
Blood Transfusion	3.2	0.58	0.69	0.746	0.757	**0.764**
Vehicle	3.3	0.9	0.909	0.832	0.914	**0.932**
CMC	3.4	0.59	0.606	0.645	0.658	**0.677**
Yeast	5.1	0.69	0.716	0.586	0.654	**0.743**
PC1	13.4	0.65	0.669	0.568	0.578	**0.696**

Table 7. Overall Performance Measure of Classifiers

Method/ Classifier	**C4.5**	**SVM**	**DNN**	**F Ratio**	**P value**	**Statistically Significant?**
Proposed	**0.798667**	0.714667	**0.884042**	**22.860936**	**1.3E-07**	Yes
SMOTE	0.776333	**0.721825**	0.82675	7.347252	0.001702	Yes
ADASYN	0.784125	0.71233333	0.836875	10.16406	0.000221	Yes

To demonstrate the efficiency of the proposed work, the existing oversampling approaches SMOTE and ADASYN are compared with the results of the proposed work. The number of synthetic samples generated using the proposed method equals the number of minority samples in the training dataset. The same numbers of synthetic samples are generated using the SMOTE and ADASYN and the obtained AUC are shown in the Table 4 . From the results of Table 4 , it is noted that the proposed oversampling method with C4.5 gives better results for 12 datasets, SVM gives better results for 4 datasets but DNN gives better results for 16 datasets than the SMOTE and ADASYN. It is observed that the SVM classifier is not giving better classification accuracy when the feature values of majority and minority class samples are more or less similar.

To experiment the results with noise, Gaussian noise is added to the training dataset before the oversampling procedure. From Table 5, it is clear that the performance of the proposed method is better during classification in most of the chosen datasets under all the three classifiers even after adding noise to the data; it is also observed from Tables 4 and 5 that for few datasets there is a downfall in the AUC values. From the experiment trials it is observed that the performance of the classifiers degrades when the amount of noise added is more than 5 percentages.

The classification performance of the proposed method on the dataset without noise (Table 4) and with the noise (Table 5) tells that the addition of noise to the training data produces notable results in only three datasets Abalone9-18, Yeast1vs7 and Yeast2vs4 but on the remaining datasets the AUC score is equal or near-equal. Adding noise to the data does not make any significant changes.

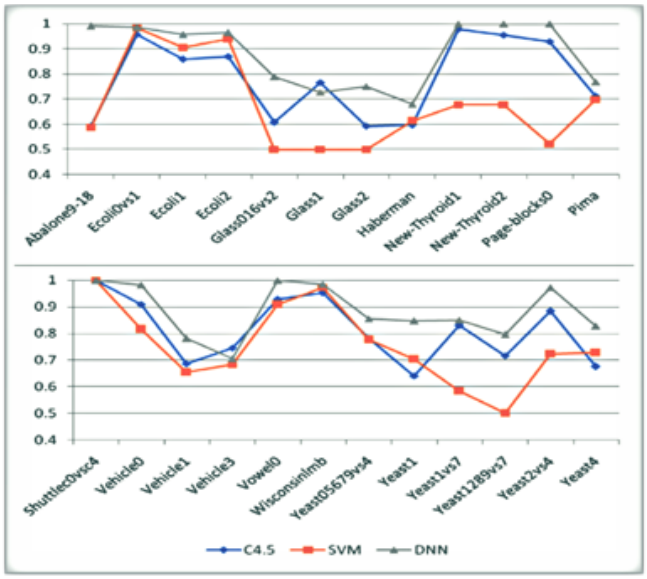

Fig 2. Comparison of AUC under C4.5, SVM and DNN

3.2 Comparison with GenSample

The result of an existing oversampling method with GA namely GenSample is considered for comparison. The chosen method considers 50 percent of data for training and 50 percent data for testing and uses C4.5 classifier for classification. The proposed oversampling method with C4.5 and DNN classifiers are compared with GenSample [8] to show the efficiency. From Table 6, it is clear that the proposed method with C4.5 performs better in 8 datasets out of 9 than the GenSample. Also, the proposed method with DNN classifier is processed with different percentages of training and testing dataset and it is observed that 80 percent data for training and 20 percent data for testing works better in all the datasets except the Ionosphere dataset. Also it is observed that DNN performs better for the proposed oversampling method than the C4.5 classifier and also than the GenSample method. This result is achieved by changing the number of neurons in hidden layers of DNN for each dataset. From the experiments, it is observed that DNN classifier performs better when less number of neurons in each hidden layer for the datasets with more number of features(>20) and more number of neurons in each hidden layer for the dataset with lesser number of features.

3.3 Statistical Significance Test

To assess the performance of the proposed method, SMOTE and ADASYN under three classifiers on a specific problem domain it is necessary to perform statistical significance tests. Here the performance measures of the classifiers are evaluated with One-Way Repeated-Measure ANOVA along with F test with the degree of freedom be 2 and 46. The critical values obtained for the degree of freedom 2 and 46 are 3.232 for one-sided test and 4.051 for the two-sided test respectively. The calculated F and P values for the proposed oversampling method and the existing methods SMOTE and ADASYN are shown in Table 7. It is clear that the proposed method produces better P value than SMOTE and ADASYN with the significance level 0.05. Null hypothesis is rejected for the proposed method and for SMOTE and ADASYN at the significance level of 0.05 using both one-tailed and two-tailed tests. The hypothesis testing concludes that the performance of the DNN classifier is statistically significant and when comparing the P values, the proposed method outperforms SMOTE and ADASYN.

4 Conclusion

The proposed oversampling approach is able to solve different kinds of application problems with the imbalanced dataset. When the oversampling methods balance the count of minority and majority class samples it occupies more memory space when the imbalance ratio increases. But the proposed approach only doubles the minority class samples instead of balancing the count of majority and minority class samples, so it won't occupy more memory space even though the imbalance ratio is high. It is observed that Gaussian distribution for initial population generates synthetic samples with lesser cycles (maximum of 7 and minimum of 1) for generating each synthetic sample. The proposed oversampling and DNN combination gives better results in

all the selected 9 datasets than the combination GenSample and C4.5. Also it has been observed from the values of AUC that the DNN classifier with the proposed method reduces the misclassification in minority and majority class samples. It is also proved from the P value (0.00000013) that the proposed method is statistically significant and is better than ADASYN and SMOTE. This work provides a good starting point for discussion and future research. Here we use only a minimum number of samples for the minority class and believe that this sample count can be further decreased to achieve better results by generating more resemble data for the training.

Acknowledgement

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions that are helpful to improve the quality of this paper.

References

1) Wang L, Han M, Li X, Zhang N, Cheng H. Review of Classification Methods on Unbalanced Data Sets. *IEEE Access*. 2021;9:64606–64628. Available from: https://doi.org/10.1109/ACCESS.2021.3074243.
2) Li Q, Zhao C, He X, Chen K, Wang R. The Impact of Partial Balance of Imbalanced Dataset on Classification Performance. *Electronics*. 2022;11(9):1322. Available from: https://doi.org/10.3390/electronics11091322.
3) Douzas G, Lechleitner M, Bacao F. Improving the quality of predictive models in small data GSDOT: A new algorithm for generating synthetic data. *PLOS ONE*. 2022;17(4):e0265626. Available from: https://doi.org/10.1371/journal.pone.0265626.
4) Gnip P, Vokorokos L, Drotár P. Selective oversampling approach for strongly imbalanced data. *PeerJ Computer Science*. 2021;7:e604. Available from: https://doi.org/10.7717/peerj-cs.604.
5) Rahnamayan S, Tizhoosh HR, Salama MMA. A novel population initialization method for accelerating evolutionary algorithms. *Computers & Mathematics with Applications*. 2007;53(10):1605–1614. Available from: https://doi.org/10.1016/j.camwa.2006.07.013.
6) Hasanzadeh MR, Keynia F. A new population initialisation method based on the Pareto 80/20 rule for meta-heuristic optimisation algorithms. *IET Software*. 2021;15(5):323–347. Available from: https://doi.org/10.1049/sfw2.12025.
7) Zhou X, Miao F, Ma H. Genetic Algorithm with an Improved Initial Population Technique for Automatic Clustering of Low-Dimensional Data. *Information*. 2018;9(4):101. Available from: https://doi.org/10.3390/info9040101.
8) Karia V, Zhang W, Naeim A, Ramezani R. GenSample: A genetic algorithm for oversampling in imbalanced datasets. 2019. Available from: https://doi.org/10.48550/arXiv.1910.10806.
9) Ghazikhani A, Yazdi HS, Monsefi R. Class imbalance handling using wrapper-based random oversampling. *20th Iranian Conference on Electrical Engineering (ICEE2012)*. 2012;2012:1–6. Available from: https://doi.org/10.1109/IranianCEE.2012.6292428.
10) Arun C, Lakshmi C. Genetic algorithm-based oversampling approach to prune the class imbalance issue in software defect prediction. *Soft Computing*. 2022;26(23):12915–12931. Available from: https://doi.org/10.1007/s00500-021-06112-6.
11) Katharopoulos A, Fleuret F. Not All Samples Are Created Equal: Deep Learning with Importance Sampling. 2018. Available from: https://arxiv.org/abs/1803.00942v3.
12) Suh S, Lukowicz P, Lee YO. Discriminative feature generation for classification of imbalanced data. *Pattern Recognition*. 2022;122:108302. Available from: https://doi.org/10.1016/j.patcog.2021.108302.
13) Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*. 2002;16:321–357. Available from: https://doi.org/10.1613/jair.953.
14) He H, Bai Y, Garcia EA, Li S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *IEEE International Joint Conference on Neural Networks*. 2008;p. 1322–1328. Available from: https://doi.org/10.1109/IJCNN.2008.4633969.
15) Xu J, Chen Z, Lu Y, Yang X, Pumir A. Improved Preterm Prediction Based on Optimized Synthetic Sampling of EHG Signal. . Available from: https://doi.org/10.48550/arXiv.2007.01447.
16) Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing. . Available from: https://sci2s.ugr.es/keel/datasets.php.
17) Imbalanced Datasets from UCI repository . . Available from: https://archive.ics.uci.edu/ml/datasets.php.
18) Imbalanced Datasets from PROMISE repository. . Available from: http://promise.site.uottawa.ca/SERepository/.
19) Chawla NV, C4. Investigating the Effect of Sampling Method, Probabilistic Estimate, and Decision Tree Structure. *Proc Intl Conf Mach Learn Work Learn from Imbalanced Data Sets II*. 2003;8. Available from: https://www3.nd.edu/~dial/publications/chawla2003c45.pdf.
20) Kulkarni A, Chong D, Batarseh FA. Foundations of data imbalance and solutions for a data democracy. 2021. Available from: https://doi.org/10.48550/arXiv.2108.00071.