Surgical Endoscopy (2023) 37:4754–4765
https://doi.org/10.1007/s00464-023-09955-2

ORIGINAL ARTICLE

Skill-level classification and performance evaluation for endoscopic sleeve gastroplasty

James Dials[1] · Doga Demirel[1] · Reinaldo Sanchez-Arias[2] · Tansel Halic[3] · Uwe Kruger[4] · Suvranu De[5] · Mark A. Gromski[6]

Received: 30 October 2022 / Accepted: 12 February 2023 / Published online: 10 March 2023

Abstract

Background We previously developed grading metrics for quantitative performance measurement for simulated endoscopic sleeve gastroplasty (ESG) to create a scalar reference to classify subjects into experts and novices. In this work, we used synthetic data generation and expanded our skill level analysis using machine learning techniques.

Methods We used the synthetic data generation algorithm SMOTE to expand and balance our dataset of seven actual simulated ESG procedures using synthetic data. We performed optimization to seek optimum metrics to classify experts and novices by identifying the most critical and distinctive sub-tasks. We used support vector machine (SVM), AdaBoost, K-nearest neighbors (KNN) Kernel Fisher discriminant analysis (KFDA), random forest, and decision tree classifiers to classify surgeons as experts or novices after grading. Furthermore, we used an optimization model to create weights for each task and separate the clusters by maximizing the distance between the expert and novice scores.

Results We split our dataset into a training set of 15 samples and a testing dataset of five samples. We put this dataset through six classifiers, SVM, KFDA, AdaBoost, KNN, random forest, and decision tree, resulting in 0.94, 0.94, 1.00, 1.00, 1.00, and 1.00 accuracy, respectively, for training and 1.00 accuracy for the testing results for SVM and AdaBoost. Our optimization model maximized the distance between the expert and novice groups from 2 to 53.72.

Conclusion This paper shows that feature reduction, in combination with classification algorithms such as SVM and KNN, can be used in tandem to classify endoscopists as experts or novices based on their results recorded using our grading metrics. Furthermore, this work introduces a non-linear constraint optimization to separate the two clusters and find the most important tasks using weights.

Keywords Endoscopic simulator · Endoscopic sleeve gastroplasty · Non-linear constraint optimization · Synthetic data generation · Machine learning classification

Endoscopic sleeve gastroplasty (ESG) is a relatively new endoscopic procedure for primary weight loss. Therefore, the standardized training for the procedure is still in the early stages of development within the endoscopic and weight loss community. The need for skill classification is pertinent and essential to developing high-impact training modules. One current training method produced by the American Society for Gastrointestinal Endoscopy (ASGE) is called the Skill, Training, Assessment, and Reinforcement (STAR) program, which includes an online curriculum, a live course on Ex-vivo porcine specimen, and a post-course skill assessment [1, 2]. This training module is currently available for endoscopic suturing in general but is not yet developed for ESG. Outside of this program, other sources of training, such as device training from the manufacturer of the endoscopic

✉ Doga Demirel
ddemirel@floridapoly.edu

1 Department of Computer Science, Florida Polytechnic University, Lakeland, FL, USA

2 Department of Data Science and Business Analytics, Florida Polytechnic University, Lakeland, FL, USA

3 Intuitive Surgical, Peachtree Corners, GA, USA

4 Department of Biomedical Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA

5 College of Engineering, Florida A&M University - Florida State University, Tallahassee, FL, USA

6 Division of Gastroenterology and Hepatology, Indiana University School of Medicine, Indianapolis, IN, USA

suturing company or formal/informal preceptorships, are ad hoc and not systematized in any way with any formal assessment after learning sessions. With the help of virtual reality (VR), training for the ESG procedure could be significantly improved, considering the model can be mapped closer to a human procedure and can be repeated without any cadaver/specimen cost and risk to human patients.

Our ultimate project goal (NIH/NIBIB 1R01EB033674-01A1) is to develop a VR training platform where endoscopists can learn and master the ESG procedure. Since VR allows for practice repetition on the ESG procedure, endoscopists could receive feedback and different difficulty scenarios to improve their progress. We need grading metrics to provide descriptive feedback that relies on quantitative measurements to achieve this goal. In addition, we need to classify endoscopists with respect to their skill level.

Our previous work created a Hierarchical Task Analysis (HTA) and a Likert scale grading metric for the ESG procedure [3]. These metrics could convey the performance relation and the expertise level. We successfully demonstrated that an accurate performance evaluation of an endoscopist could be provided with our scoring metrics [3, 4]. One of the limitations of the metrics was that the measurement could not describe deficiencies in the expertise level at the fine-granularity level. For instance, the classification of the subjects can be identified with respect to their performance, but the level of expertise (e.g., how novice they are) among the subjects remains to be answered. Also, the metrics cannot identify the most critical tasks and sub-tasks with respect to the other tasks in the overall procedure. There are no weights in the metric tasks, as all the metrics are uniformly equal. Furthermore, the metrics are not adaptive such that after proficiency is attained, further improvement or refinement cannot be delineated. We, therefore, require a more in-depth analysis where more precise classification can be accomplished.

This study uses machine learning methods to improve classification accuracy and support multi-labeling. Over the years, classification methods have been used for different purposes, touching on many other subjects, such as classifying diabetes and cardiovascular disease [5], classifying plant species [6], and even numerical classification in neuroanatomy [7]. These classification methods require a large dataset making it inefficient to apply in our case. Larger and more balanced datasets are needed to eliminate incorrect training and bias [8]. Our dataset was small, composed of only four experts and three novice scores. Therefore, we first plan to expand and balance our dataset using the synthetic data generation technique, Synthetic Minority Oversampling (SMOTE), which adds to the minority class.

In some cases, after generating data and balancing the dataset, there are too many features compared to the number of samples, which can cause the problem of overfitting. This leaves a classification algorithm that will have a high training accuracy but, when tested on real-world data, performs poorly. To fix this problem, we performed feature extraction to map multiple features into a smaller number of features to have a smaller but more meaningful feature space.

We applied Support Vector Machine (SVM) and Kernel Fisher Discriminant Analysis (KFDA) techniques and other standard techniques on the expanded dataset to examine the separability of experts from novices. In addition, we developed an optimization method that provides further flexibility in the performance differentiation of the subjects at the task level. The model is aimed to seek an optimal solution by either maximizing or minimizing the objective by given parameter constraints for the task performance distribution. In this work, we present and compare the results of machine learning techniques and our non-linear optimization model. The workflow/overview of our study can be seen in Fig. 1.

Literature review

In the literature review, we will define multiple methods of synthetic data generation, classification, dimensionality reduction, and optimization. Synthetic data generation is used in many studies to balance an imbalanced data set. In a study by Mohammad et al. [9], SMOTE was used to improve the classification of an imbalanced diabetes dataset. The diabetes dataset used, ZADA, contains around 909 patient records. This imbalanced two-class dataset consists of seven independent variables with a class label classifying the patient as diabetic or non-diabetic. Mohammed et al. used six classification algorithms: Naïve Bayes, K-nearest neighbor (KNN), decision tree, logistic regression, Support Vector Machines, and artificial neural networks. The SVM scored the highest accuracy with 0.87 out of all these methods. After using SMOTE, the accuracy of the KNN algorithm improved from 0.82 to 0.91, with a 1.00 imbalance correction. Although this paper achieved significantly higher accuracy using SMOTE and some other preprocessing, one possible improvement on this paper would be to use feature reduction using a technique like PCA. In a work by Xu et al. [10], decision trees were used as their classification algorithm. They used a cluster-based oversampling algorithm (KNSMOTE), which combined SMOTE and K-means clustering to maintain a

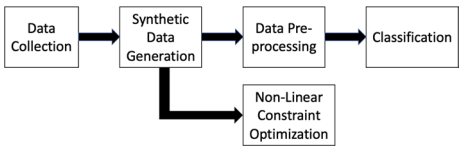

Fig. 1 The workflow of our study

proper sample class. Using eight UCI datasets, this KNSMOTE algorithm performed 0.99 and 0.99 on sensitivity and specificity metrics using the random forest algorithm.

Roopa et al. [11] presented a feature extraction method for extracting valuable features for tuberculosis analysis. Although tuberculosis can be diagnosed through chest x-ray images, many features prohibit the possibility of a computer diagnosing this disease. They used both PCA and KPCA to perform feature extraction on the x-ray images, allowing a computer to classify the images as being affected or normal. Roopa et al., concluded that in a linear regression model for the classification, PCA resulted in 0.96 while KPCA resulted in 0.62 accuracy. The result of KPCA being 0.62 is an unacceptable result. Wu et al. [12] classified gait patterns using KPCA feature extraction and SVM. They used KPCA to extract human movement information from an OPTOTRAK 3020 motion analysis system of 24 young and 24 elderly participants. When combining KPCA with the SVM, their model could classify young or old patients' gait patterns with 0.91 accuracy. Although this study achieved a decent accuracy (0.91), it could be improved by gathering a larger dataset.

In a paper by Neffati et al. [13], brain MR image data classification is performed using SVMs and KPCA. The first step in their process was to use KPCA for feature extraction. The data initially had 1024 dimensions and were reduced by KPCA to 16 principal components. The next step they performed was to use K-Fold cross-validation to avoid overfitting. Once completed, they trained the SVM classifier and then tested it using new brain MRIs, outputting a prediction. Using an SVM-KPCA classifier with an RBF kernel resulted in 1.00 accuracy for three different brain MR image databases. With the lack of dimensionality reduction in other results, such as only reducing to 70 features instead of 16, the accuracy was 0.97, 0.97, and 0.96 for the three databases. In a study on small organic molecules, an SVM correctly classified approximately 0.90 of the data with a Matthews correlation coefficient of 0.78 [14].

In a study by Liu et al. [15], KFDA and KPCA were used for facial recognition. With ten features, KPCA accurately classified 0.76, and with 60 features, the accuracy was 0.83. In a paper by Azar et al. [16], lymph disease was diagnosed using a random forest classifier and PCA for feature reduction. They selected 15 attributes for PCA and used the average of tenfold cross-validation. The accuracy of the random forest, when used in unison with PCA, was 0.83. In a study by Masetic and Subasi [17], heart failure was diagnosed based on automatic electrocardiogram (ECG) results. They used five different classifiers or comparisons, including decision tree, KNN, SVM, and random forest. Their best result was from the random forest classifier, which resulted in 1.00 accuracy in identifying and classifying ECG signals.

A study by Jadhav and Channe [18] compared KNN, Naïve Bayes, and decision tree classifiers to classify three different datasets (weather nominal, Segment challenge, and supermarket). The accuracy for the weather nominal dataset was 1.00, 0.92, and 1.00, respectively. For the second dataset, the results were 1.0, 0.816, and 0.99, respectively. The final dataset's accuracies were 0.89, 0.63, and 0.63. Although KNN had the best accuracies overall, decision tree was the second best.

In a study by Lavanya and Rani [19], three breast cancer datasets were classified using a decision tree classifier, classification, and regression trees (CART) using tenfold cross-validation. Their CART classifier classified the three datasets with 0.69, 0.94, and 0.92 accuracy. In [20], the authors classified X-ray images of good and defective pecans. They used 100 images of good and 100 images of defective pecans and compared AdaBoost and SVM classifiers. The AdaBoost classifier resulted in 0.92, while the SVM classified at 0.90 accuracy. In a paper by Hu et al. [21], they used UAV images to identify diseased Pinus trees. They compared DCGAN, the deep learning method, and the AdaBoost classifier. They combined multiple methods and used precision as the primary indicator. AdaBoost had a precision of 0.48, combined with AlexNet resulted in 0.58, and combined with VGG resulted in 0.69.

In a study by Hossain et al. [22], plant leaf disease detection was performed using a KNN classifier. The classifier was tasked with classifying between six different diseases. The classifier in this study resulted in a 0.96 accuracy when classifying these six diseases. In a paper by Moldagulova [23], KNN was used to classify textual documents. When classifying, they used different numbers of neighbors. 1, 5, 25, 50, 51, 75, and 300 neighbors to compare the results. With one neighbor, the accuracy was 1.00, with five neighbors, the accuracy was 0.97, with 25 neighbors, the accuracy was 0.94, and for 300 neighbors, the accuracy was 0.48. Their results showed that as the neighbors increased, the accuracy became worse.

With constraint optimization, the problem arises where not all the functions or constraints are linear. To solve these problems, we use non-linear constraint optimization (NCO). NCOs can have unbounded variables, upper bounds, and range constraints [24]. In a paper written by Schouwenaars et al. [25], MILP is used for vehicle path planning. This paper uses LP, mixed integer, and linear constraints in tandem to find optimal solutions for a vehicle's path and collision avoidance. This research's two main decision variables were time and fuel, with constraints to avoid collision with other vehicles and obstacles. Since this study is for collision avoidance, the techniques do not apply to our study. Schouwenaars et al. used CPLEX optimization software in unison with AMPL/Matlab interface in this paper. This paper used MILP to create a novel approach to multiple-vehicle path

planning with collision avoidance. A detailed method of constraint distance optimization is outlined in a study on finding the optimum blend of fractal methods for automatic malignancy determination in dermoscopy images [25]. They used four different mixed integer programming classification models: maximizing the distance between groups, minimizing the total distance within groups, maximizing the total distance between groups, and minimizing the maximum within the group.

Methods

The Indiana University Human Research Protection Program has determined the project does not require an IRB review due to the project not involving human subjects.

Data collection

We used seven recorded complete ESG procedural videos performed on porcine models as part of our HTA-based development of the initial metrics for the ESG procedure [3]. These seven video recordings each included three camera angles: the endoscope view, the GoPro view, and an external camera view. These procedures were performed by a total of four experts and three novices. Two experts performed the four expert procedures, and three different novices performed all three novice procedures (one each). Using the recordings, we scored each performance using our metrics [3]. Our metrics consist of 7 tasks and 45 sub-tasks with a Likert scale type of grade, with some consisting of an intermediate score. The metrics are inverted; therefore, when the task is completed properly, the endoscopist will be given a zero, and when the task is performed incorrectly, they will be given a five. Following the performance evaluation, we performed the statistical analysis and determined the task time and performance relation as previously reported [3].

Synthetic data generation

The machine learning algorithm's accuracy is proportional to the size of the data. As the number of procedures recorded for the initial study was limited, we needed to expand the dataset. Intuitively, we could record more surgical performances and evaluate them. Unfortunately, this was not a feasible option due to COVID-19, where access to the lab setting was very limited. We, therefore, used an accurate synthetic data creation method to expand the current data. To avoid creating an imbalanced dataset, we utilized oversampling to create our synthetic data. The purpose of oversampling in data analysis is to compensate for imbalanced datasets and functions by taking the minority class of the dataset and creating synthetic data, enlarging the minority class to balance the dataset [26].

There are a few main types of oversampling techniques, including random oversampling, SMOTE, and data augmentation. Random oversampling takes random samples from the minority class and copies them until the dataset is balanced. This was not a viable option since we desired more complex synthetic data. Other forms of data augmentation are mainly used in image data, where the image is cropped or rotated and added to the data set [27]. Since we are not using image data, we ended up utilizing SMOTE. SMOTE works by taking a sample from the minority class and considering its K-nearest neighbors, taking the vector between the K-neighbor and the selected data point, then parameterizing the value with a random number between 0 and 1 [28]. This allowed us to create a balanced dataset with ten values in each class.

Data preprocessing and feature reduction

The main steps in preprocessing were scaling the data, reducing the feature dimensionality, tuning the parameters, and performing K-Fold classification. The standard scaler standardizes the features by subtracting the mean and scaling to unit variance. The formula for calculating the standardized score is as follows:

$$z = \frac{x - u}{s}$$

where s is the standard deviation of the training samples, u is the mean of the training samples, and x is a value, or element, of our sample.

Principal component analysis (PCA) [29, 30] was used for the next step in preprocessing the data. PCA was used to reduce the features from three to two dimensions. The three features (suturing, total grade, and time completion) are linearly combined and projected onto the PC1-PC2 two-dimensional space. In this lower dimensional representation, the first two principal components already explain 0.98 of the variance in our data. Since every single classifier went through the same process of PCA feature reduction, all of them used the same linear combinations of the old features. Below is the PCA biplot shown in Fig. 2.

The next step in preprocessing the data was to tune the parameters using a K-fold cross-validation framework. For each classifier, the distribution of the parameters was defined. For SVM, the kernel parameter and the penalty parameter were taken in as parameters. The decision tree's max depth, min samples leaf, and criterion were taken in. For AdaBoost, the n estimators, learning rate, and algorithm were taken in as parameters. For the random forest n estimator, max depth and criterion were taken in. For KNN n neighbors, weights, leaf size, and algorithm were taken in as

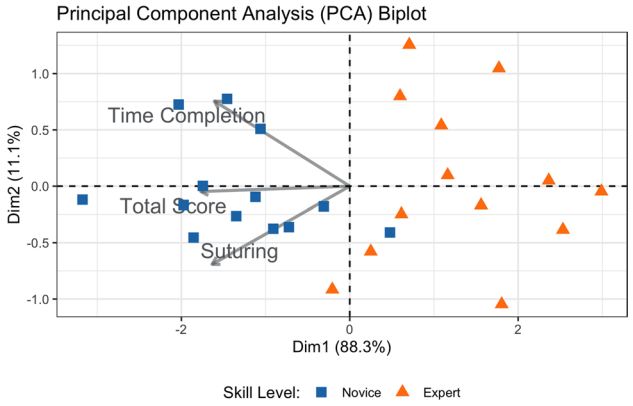

Fig. 2 PCA bivariate plot of first two principal components

parameters. For KFDA, *n* components, the kernel parameter, and regularization parameter ε = 1e-08 were taken in. All the classifiers used a randomized search to search for the best possible combination of parameters. Each classifier ran six cross-fold validation since the number of splits cannot exceed the number of members in the least populated class (six).

Classification methods

Since our goal is to create a proper threshold to classify between expert and novice, we used our new preprocessed data to train six classifiers. The six classifiers were SVM, KNN, AdaBoost, KFDA, random forest, and decision trees. SVMs work by using a soft margin classifier which allows for soft misclassification (outliers), creating a better long-term classification accuracy [31]. KNN works by finding the number k of the nearest "neighboring" points based on a specified number of neighbors to look for. Based on those nearest neighbors, it will calculate its similarity to the already classified points and decide [32]. AdaBoost is an ensemble boosting classifier that combines multiple classifiers to increase accuracy and sets weights of those classifiers that focus on unusual observations [33]. KFDA works by maximizing the between-class scatter and minimizing the within-class scatter. KFDA improves upon Fischer discriminant analysis (FDA) by enabling non-linear subspaces using the kernel trick. Decision trees predict a target value by learning decision rules inferred from the data features. Random forest classifier is an ensemble classifier that builds multiple decision trees on different samples and takes the majority vote. Since we used PCA to reduce our data from three dimensions (total score, suture score, total duration) to two dimensions, we now had linearly separable data to classify. We used PCA for all six classifiers and recorded the results for all of them.

Table 1 Objective function of the optimization model

Optimization model
Objective function = $Max : \lvert optArray(x) \rvert$

Non-linear constraint optimization

Our objective in the optimization model has a several fold outcome; first, we desire to separate the distance between two groups, expert and novice. Second, properly identify/weigh each task in the procedure. As the preliminary step, we calculated the distance between these two groups after clustering them as expert and novice. Once we have the distance between the two groups, we optimize the task scores that add to the total score. This leads the distance between the clusters to "maximize," creating a more significant distinction between an expert and a novice. This objective is achieved through non-linear constraint optimization, which finds the optimal solution to our objective function [24].

Weighting each task

To find the weight of each task, an optimization array was made with all 45 sub-tasks and then ran through the objective function defined in Table 1. The objective function returns the array of sub-tasks based on a guess input. In our case, the guess input(x) is the average of all the scores per task, called the optimization array (*optArray*). We set a constraint with 197 as the maximum score since that was our original worst score and − 197 as the best possible score. The solver then used this maximum score and the guess input to calculate the new scores for each sub-task. Based on this process, the weights of each sub-task were found and recorded in the results section.

Separating distance between groups

We provide custom constraints using a distribution function for a task. We customize the performance score distribution at the fine granularity at the task level. Therefore, we use beta distribution to parametrize our solution. Although beta distribution is a probability distribution, we used it as our optimized weight. Hence, the model we created uses Beta Distribution as a weighting technique to separate the two clusters effectively. With our objective function of maximizing the distance between the expert cluster (*EC*) and novice cluster (*NC*), we created a list of constraints that will guide the optimization model to an acceptable maximization of the scoring metrics. Beta distribution is typically used as

Table 2 Optimization model

Optimization model
Objective function $= Max: \left\lvert \frac{BNC}{NCWeight} - \frac{WEC}{ECWeight} \right\rvert$
Functions/Equations
1. $u_T = \frac{\sum_{i=1}^{I} T_i}{I}$
2. $T_x = \frac{u_T}{\sum_{k=1}^{n} T_k}$
3. $T_w = \frac{T_x^{\alpha-1}(1-T_x)^{\beta-1}}{B(\alpha,\beta)}$
4. $\alpha_T = \left(\frac{1-\mu_T}{\sigma_T^2} - \frac{1}{\mu_T}\right)\mu_T^2$
5. $\beta_T = \alpha_T\left(\frac{1}{\mu_t} - 1\right)$
6. $B_T(\alpha_T, \beta_T) = \frac{\Gamma(\alpha_T)\Gamma(\beta_T)}{\Gamma(\alpha_T+\beta_T)}$
Constraints
1. $\left(\frac{NC}{NCWeight}\right) - \left(\frac{EC}{ECWeight}\right), lb = 0, ub = 197$
2. -StdNC < NCWeight < StdNC
3. -StdEC < ECWeight < StdEC

a probability distribution, but we will use this value as a weight in our model. Our maximization model is shown in Table 2.

The original distance is calculated by taking the best novice score (*BNS*) and subtracting it from the worst expert score (*WES*). Our maximization model uses the same technique but divides a specific weight, created through beta distribution unique to each cluster, into the respective scores. The model then iterates through the possible weights to maximize the distance between the two groups until landing upon the optimal solution.

Table 2 also shows the beta distribution functions. The first equation shows the task average (u_T) where T stands for the task itself and i stands for the index of the task list. The second equation shows the normalized task average (T_x) where k stands for the index of the task list. The third equation is the weight of each task based on the beta distribution (T_w). Equations four and five are used to calculate the alpha (α_T) and beta (β_T) values for the task, where μ is the mean and σ^2 is the variance. Equation six is the beta function itself ($B_T(\alpha_T, \beta_T)$), where $\Gamma(y)$ is the gamma function.

We created constraints for this objective function to limit the maximum distance appropriately, as seen in Table 2. We define the weighted distance in the first constraint as between 0 and 197. We chose the value 197 because our original maximum score was 197. In the second constraint, we define a constraint for the novice weight (*NCWeight*), with a lower bound of the negative standard deviation of the *NC* total grade ($-StdNC$), and an upper bound of the standard deviation of the *NC* total grade (*StdNC*). For the third constraint, we define a similar constraint as the second one but for the expert weight (*ECWeight*) with a lower bound of the negative standard deviation of the *EC* total grade(*-StdEC*) and an upper bound that is the standard deviation of the *EC* total grade (*StdEC*). The model can maximize the distance between *EC* and *NC with these constraints*, effectively separating the two clusters.

Results

SMOTE results

After applying SMOTE as explained in Section "Synthetic data generation," our dataset has been expanded to consist of twenty values with ten in each class, expert and novice. After expansion, we conduct a simple SVM for future use of classification in our ESG evaluation platform. The comparison of the original and expanded dataset can be seen in Fig. 3 for the suturing task score, time completion score, and total score. The original expert class had worse grades (higher grades) than the expert class's grades (lower grades) in the expanded dataset. As for the novice class, the original class was affected by a small number of outliers. In contrast, the expanded novice class was skewed with a more homogenous distribution of the performance scores, reflecting a better representation of novice scores. These data indicate that SMOTE created greater separation between the two classes allowing for better classification accuracy by the SVM.

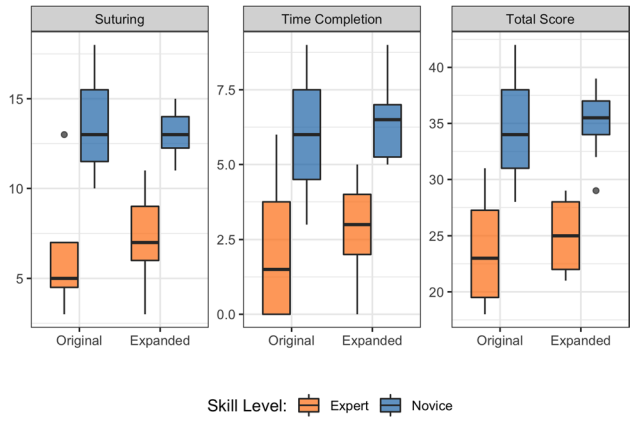

Fig. 3 Comparison of suturing, time completion, and total score for original and expanded datasets

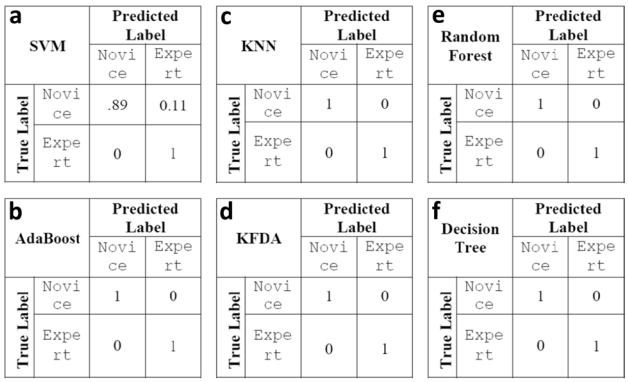

Fig. 4 Confusion matrix comparison of each classifier

Classifiers/PCA results

When running PCA, our feature space went from three to two features creating a linearly separable dataset. Since PCA takes the original variable space (suturing, time completion, and total grade) and maps it into a new feature space, the three original features get mapped into two newly created orthogonal features built as linear combinations of the three original features.

After running our dataset through SMOTE and PCA, we ended up with our dataset of twenty-six samples with two features, *i.e.*, the first two principal components. We then split the data into a training and a testing dataset. Our training data were 19 samples (ten experts, nine novices), while our testing data were seven samples (three experts, four novices). The metrics used for the results of the classifiers implemented in this study were the confusion matrix, accuracy, precision, recall, and Fl score [34]. Accuracy, precision, recall, and F1 score for each classifier can be seen in Table 3. The confusion matrix comparison of each classifier used can be seen in Fig. 4.

When tuning the parameters of the SVM classifier, the best parameters were RBF kernel, gamma of 1e-05, and a C of 1000. Figure 4a shows the confusion matrix for the SVM's training data, where 1.00 of the experts were classified correctly, and 0.89 of the novices were classified correctly, leaving only 0.11 of the novices who were falsely classified as experts. The precision for each training and testing for the SVM was 0.95 and 1.00, respectively, while recall for the training and testing was 0.94 and 1.00, and F1 score for the training and testing was 0.95 and 1.00.

For the AdaBoost classifier, the best parameters in the fivefold validation were 400 estimators, the learning rate set to 0.001, and the algorithm as SAMME.R. For the KNN classifier, the best parameters in the fivefold validation were 11 neighbors, uniform weights, leaf size of thirty, and the ball tree algorithm. Figure 4b and c depicts the training data confusion matrix for the AdaBoost and KNN classifiers. AdaBoost and KNN correctly calculated 1.00 of the true negative and 1.00 of the true positive. Thus, leaving the false negative at zero and the false true at zero. For both the AdaBoost and the KNN classifiers, the precision for each class, expert and novice, was 1.00 and 1.00, respectively. The F1 scores for the experts and novices were 1.00 and 1.00 for the AdaBoost and the KNN.

For the KFDA classifier, the best parameters in the fivefold validation were a linear kernel two component and a robustness offset of 1e-08. KFDA calculated 1.00 of the novices correctly and 1.00 of the experts correctly, while misclassifying none. This can be seen in Fig. 4d. The precision for training and testing for the KFDA classifier was

Table 3 Comparison of classifier methods using accuracy, precision, recall, and F1 score

Method	Accuracy		Precision		Recall		F1 Score	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
SVM	0.94	1.00	0.95	1.00	0.94	1.00	0.95	1.00
AdaBoost	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
KNN	1.00	0.85	1.00	0.83	1.00	0.90	1.00	0.84
KFDA	1.00	0.85	1.00	0.83	1.00	0.90	1.00	0.84
Random forest	1.00	0.85	1.00	0.92	1.00	0.75	1.00	0.79
Decision tree	1.00	0.71	1.00	0.71	1.00	0.71	1.00	0.71

1.00 and 0.83, respectively. The F1 scores for the training and testing were 1.00 and 0.84.

While for the parameters of the random forest classifier, the best parameters in the fivefold validation were 300 estimators, no max depth, and an entropy criterion. As seen in Fig. 4e, random forest correctly calculated 1.00 of the novices and 1.00 of the experts correctly. The precision for training and testing for the random forest classifier were 1.00 and 0.92, respectively. The F1 scores for the training and testing were 1.00 and 0.79.

Finally, for tuning the parameters of the decision tree classifier, the best parameters in the fivefold validation were a max depth of three, a minimum sample leaf of five, and a Gini criterion. Above in Fig. 4f is the training data confusion matrix for the decision tree classifier. Decision tree correctly calculated 1.00 of the novices and 1.00 of the experts. The precision for training and testing for the decision tree classifier were 1.00 and 0.71, respectively. The F1 scores for the training and testing were 1.00 and 0.71.

Figures 5, 6, 7, 8, 9, 10 display training and testing graphs for each classifier. In these figures, experts are represented as blue squares, and novices are represented as orange triangles for the training case. Figure 5a shows the SVM classified the training data with 0.94 accuracy. Figure 5b shows the SVM classified the testing data with 1.00 accuracy with the seven data samples.

AdaBoost's training data classification results in a decision plot where nine expert surgeons were all classified accurately, and five out of six novices were classified accurately. When training the AdaBoost classifier with the tuned parameters, the training accuracy was 1.00, as seen in Fig. 6a. Given seven data samples, AdaBoost classified the testing data at 1.00, as seen in Fig. 6b.

For KNN, training data were classified with 1.00 accuracy (Fig. 7a). In contrast, for the testing data, KNN classified the testing data at 0.85 accuracy, where seven data samples were given (Fig. 7b).

Figures 8a and b show that linear kernel in KFDA classified the training data with 1.00 accuracy and testing data with 0.85 accuracy, respectively. Random forest classified the training data with 1.00 (Fig. 9a) accuracy and testing data at 0.85 (Fig. 9b).

Finally, Fig. 10a shows the decision tree graph displaying the classification separating the two groups, novice, and expert, for the training case. Decision tree classified the training data with 1.00 accuracy. Figure 10b shows the

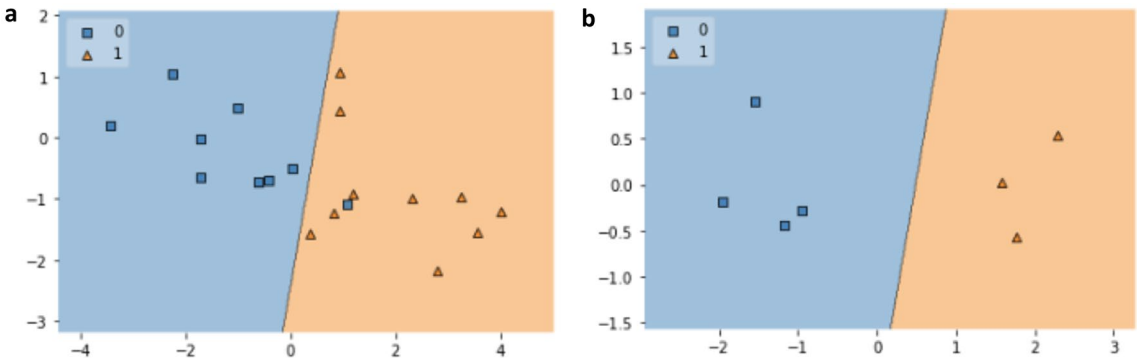

Fig. 5 Support vector classification **a** training and **b** testing graph

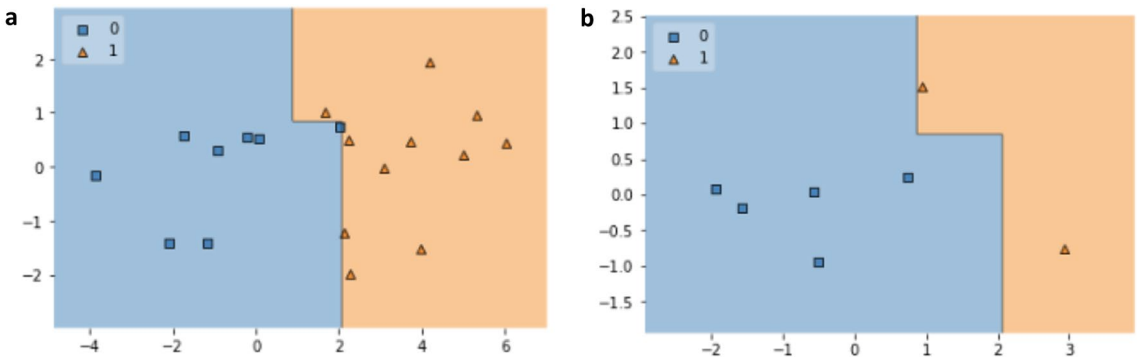

Fig. 6 AdaBoost classification **a** training and b testing graph

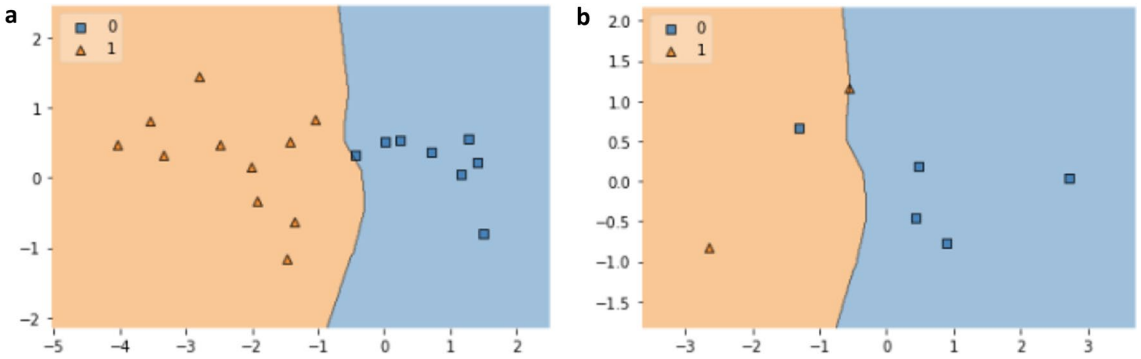

Fig. 7 K-Nearest neighbors classification **a** training and **b** testing graph

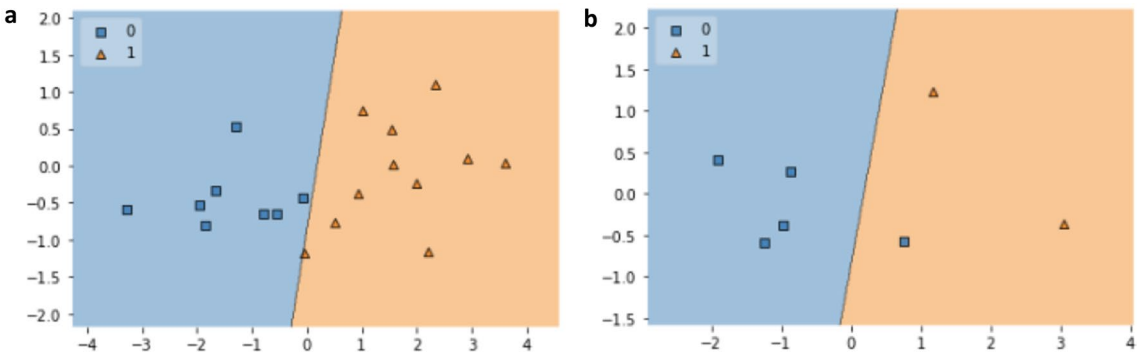

Fig. 8 KFDA (linear Kernel) classification **a** training and **b** testing graph

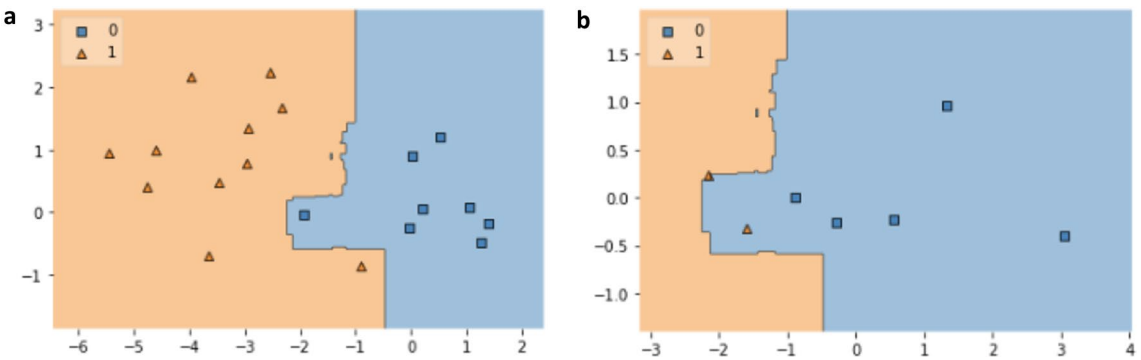

Fig. 9 Random forest classification **a** training and **b** testing graph

decision tree graph displaying the classification of both groups for the testing data. Decision tree classified the testing data at 0.71 accuracy, where seven data samples were given.

Non-linear constraint optimization results

Based on the optimization model, we determined that the suturing bite is the most critical task, given a maximum score of 8.353. The second most important component was time completion, with a maximum score of 8.203. All the other maximum scores are shown in Table 4.

Before performing NCO on our data, our original distance between the BNS and the WES was only 2.0. After completing NCO on our data with the given constraints as explained in Section 'Non–Linear Constraint Optimization," our optimized weights were −0.9661 (Novice Weight) and 1.1353 (Expert Weight), giving us a new optimized distance

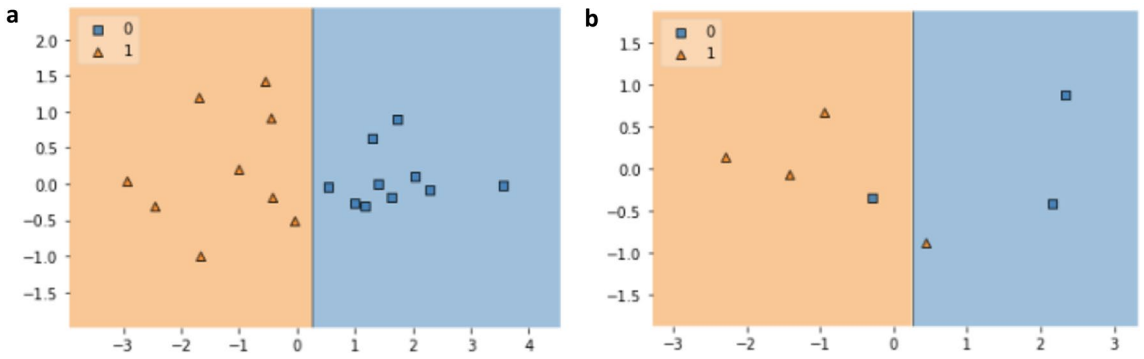

Fig. 10 Decision tree classification **a** training and **b** testing graph

Table 4 Optimized sub-task scores

Sub-task optimized scores	
Insertion of Over tube: 4.05	Suture posterior wall: 4.05
Insert into Posterior Pharynx: 4.05	Suture direction: 4.05
Advance into Esophagus: 4.05	Suture bite: 8.35
Diagnostic evaluation of esophagus: 4.05	Number bites per running suture: 4.05
Advance into stomach: 4.05	U-shaped pattern: 4.05
Advance into duodenum: 4.05	Tighten sutures: 4.05
Diagnostic of duodenum: 4.05	Suture line: 6.85
Diagnostic of stomach: 4.05	End suture: 4.60
Advance argon plasma: 4.05	Evaluate sleeves need for reinforcement sutures: 6.20
Mark anterior wall: 4.05	Deploy reinforcement sutures: 4.05
Mark posterior wall: 4.05	Suture set: 4.70
Mark greater curvature: 4.05	Removal double channel gastroscope: 4.05
Mount overstitch: 4.05	Severe bleeding: 4.05
Insert into pharynx: 4.05	Bent tag: 4.05
Insert into esophagus: 4.05	Load tissue helix command: 4.05
Insert into stomach: 4.05	Extend tissue helix command: 4.05
Start of suture: 4.05	Rotate blue cross counterclockwise command: 4.05
Grasp tissue anterior wall: 4.05	Retract helix command: 4.05
Suture anterior wall: 4.05	Remove helix command: 4.05
Grasp tissue greater curvature: 4.05	Load cinch command: 4.05
Suture greater curvature: 4.05	Deploy cinch Command: 4.05
Grasp tissue posterior wall: 4.05	Time completion: 8.20

of 55.7162 between the two clusters. This is a distance of 53.7162, which is ≈27.8 times larger than our pre-optimization results. With this more significant distance, we now have two clusters further away, leaving less room for error in classification for future procedure grading.

Conclusion

With a long-term goal of creating a well-performing simulation-based trainer for the ESG procedure, we presented a method for the simulator to classify between expert and novice and separate the distance between the two groups. After creating synthetic data using SMOTE and balancing our dataset, we reduced the dimensionality of our features using PCA. We classified experts and novices by comparing six different algorithms SVMs, KNN, AdaBoost, KFDA, decision tree, and random forest. The most accurate training result was 1.00 accuracy from AdaBoost, KNN, random forest, and decision tree, while the most accurate testing result was 1.00 accuracy from SVM and AdaBoost.

Afterward, using a non-linear constraint optimization, we created weights for the tasks in our grading metrics which we will use in our ESG simulator. We found that the most

important component to differentiating expert from novice was the suture bite, and the next most important was time to completion. We were able to create a distinction between the expert and the novice group by 51.72. Although, in future of the ESG simulator, there will be more data available, and the classifiers may not generalize at 1.00 accuracy, this model can still be retrained with new data, and the same techniques can be used to create a reliable classification tool for the trainer. In future work, we plan to record more ESG procedures to get a better dataset and rerun this process with the newly created dataset for validation and comparison purposes. This will improve our model with a more accurate synthetic dataset and reflect the actual procedural scores. In addition, we will employ other data generation techniques, such as GANs, to create synthetic data, then classify them with the same classifiers used in this study to compare with optimization to validate the authenticity of our workflow.

Acknowledgements This project was supported by grants from the National Institutes of Health (NIH)/ NIBIB 1R01EB033674-01, 5R01EB025241-04 and 5R01EB005807-11.

Funding This project was supported by grants from the National Institutes of Health (NIH)/NIBIB 1R01EB033674-01, 5R01EB025241-04 and 5R01EB005807-11.

Declarations

Disclosures James Dials, Drs. Doga Demirel, Reinaldo Sanchez-Arias, Tansel Halic, Uwe Kruger, Suvranu De, and Mark A. Gromski have no conflict of interest or financial ties to disclose.

References

1. STAR Certificate Programs (2021) Default. https://www.asge.org/home/education/advanced-education-training/star-certificate-programs. Accessed June 21, 2021.
2. Bazarbashi AN (2020) Training in bariatric endoscopy. ACG Case Rep J 7(3):e00358. https://doi.org/10.14309/crj.0000000000000358
3. Dials J et al (2021) Hierarchical task analysis of endoscopic sleeve gastroplasty. Surg Endosc. https://doi.org/10.1007/s00464-021-08893-1
4. Halic T et al (2020) S1191 Task Analysis and Performance Metrics of Endoscopic Sleeve Gastroplasty: Preparation for Virtual Simulation Development. Off J Am Coll Gastroenterol ACG 115:S595. https://doi.org/10.1309/01.ajg.0000706812.30100.05
5. Alić B, Gurbeta L, Badnjević A (2017) Machine learning techniques for classification of diabetes and cardiovascular diseases. 2017 6th Mediterranean Conference on Embedded Computing (MECO), pp. 1–4. https://doi.org/10.1109/MECO.2017.7977152.
6. Austin MP, Belbin L (1982) A new approach to the species classification problem in floristic analysis. Aust J Ecol 7(1):75–89. https://doi.org/10.1111/j.1442-9993.1982.tb01302.x
7. Agnati LF, Zoli M, Benfenati F, Pich EM, Grimaldi R, Fuxe K (1990) Aspects of neural plasticity in the central nervous system—II. Numerical classification in neuroanatomy. Neurochem Int 16(4):419–425. https://doi.org/10.1016/0197-0186(90)90003-C
8. Akbani R, Kwek S, Japkowicz N (2004) Applying support vector machines to imbalanced datasets. In: Boulicaut J-F, Esposito F, Giannotti F, Pedreschi D (eds) Machine learning: ECML 2004, vol 3201. Springer, Berlin, Heidelberg, pp 39–50. https://doi.org/10.1007/978-3-540-30115-8_7
9. Mohammed AJ (2020) Improving classification performance for a novel imbalanced medical dataset using SMOTE method. Int J Adv Trends Comput Sci Eng 9(3):3161–3172. https://doi.org/10.30534/ijatcse/2020/104932020
10. Xu Z, Shen D, Nie T, Kou Y, Yin N, Han X (2021) A cluster-based oversampling algorithm combining SMOTE and k-means for imbalanced medical data. Inf Sci 572:574–589. https://doi.org/10.1016/j.ins.2021.02.056
11. Roopa H, Asha T (2018) Feature extraction of chest X-ray images and analysis using PCA and kPCA. Int J Electr Comput Eng IJECE 8(5):3392. https://doi.org/10.1591/ijece.v8i5.pp3392-3398
12. Wu J, Wang J, Liu L (2007) Feature extraction via KPCA for classification of gait patterns. Hum Mov Sci 26(3):393–411. https://doi.org/10.1016/j.humov.2007.01.015
13. Neffati S, Ben Abdellafou K, Taouali O, Bouzrara K (2020) Enhanced SVM–KPCA method for brain MR image classification. Comput J 63(3):383–394. https://doi.org/10.1093/comjnl/bxz035
14. Byvatov E, Schneider G (2003) Support vector machine applications in bioinformatics. Appl Bioinform 2(2):67–77
15. Liu Q, Lu H, Ma S (2004) Improving Kernel Fisher discriminant analysis for face recognition. IEEE Trans Circuits Syst Video Technol 14(1):42–49
16. Azar AT, Elshazly HI, Hassanien AE, Elkorany AM (2014) A random forest classifier for lymph diseases. Comput Methods Programs Biomed 113(2):465–473
17. Masetic Z, Subasi A (2016) Congestive heart failure detection using random forest classifier. Comput Methods Programs Biomed 130:54–64
18. Jadhav SD, Channe HP (2016) Comparative study of K-NN, naive Bayes and decision tree classification techniques. Int J Sci Res IJSR 5(1):1842–1845
19. Lavanya D, Rani KU (2012) Ensemble decision tree classifier for breast cancer data. Int J Inf Technol Converg Serv 2(1):17–24
20. Mathanker SK, Weckler PR, Bowser TJ, Wang N, Maness NO (2011) AdaBoost classifiers for pecan defect classification. Comput Electron Agric 77(1):60–68
21. Hu G, Yin C, Wan M, Zhang Y, Fang Y (2020) Recognition of diseased pinus trees in UAV images using deep learning and AdaBoost classifier. Biosyst Eng 194:138–151
22. Hossain E, Hossain MF, Rahaman MA (2019) A color and texture based approach for the detection and classification of plant leaf disease using KNN classifier. 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), pp. 1–6.
23. Moldagulova A, Sulaiman RB (2017) Using KNN algorithm for classification of textual documents. 2017 8th International Conference on Information Technology (ICIT), pp. 665–671.
24. Leyffer S, Mahajan A (2010) Nonlinear constrained optimization: methods and software. Argonne National Laboratory, Argonne, Illinois.
25. Schouwenaars T, De Moor B, Feron E, How J (2001) Mixed integer programming for multi-vehicle path planning. 2001 European Control Conference (ECC), pp. 2603–2608. https://doi.org/10.23919/ECC.2001.7076321.
26. Kovács G (2019) An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. Appl Soft Comput 83:105662. https://doi.org/10.1016/j.asoc.2019.105662

27. Shorten C, Khoshgoftaar TM (2019) A survey on image data augmentation for deep learning. J Big Data 6(1):60. https://doi.org/10.1186/s40537-019-0197-0
28. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357. https://doi.org/10.1613/jair.953
29. Abdi H, Williams LJ (2010) Principal component analysis. Wiley Interdiscip Rev Comput Stat 2(4):433–459
30. Jolliffe IT, Cadima J (2016) Principal component analysis: a review and recent developments. Philos Trans R Soc Math Phys Eng Sci 374(2065):20150202
31. Noble WS (2006) What is a support vector machine? Nat Biotechnol 24(12):1565–1567. https://doi.org/10.1038/nbt1206-1565
32. Xing W, Bei Y (2020) Medical health big data classification based on KNN classification algorithm. IEEE Access 8:28808–28819. https://doi.org/10.1109/ACCESS.2019.2955754
33. Schapire RE (2013) Explaining AdaBoost. In: Schölkopf B, Luo Z, Vovk V (eds.) Empirical Inference. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 37–52. https://doi.org/10.1007/978-3-642-41136-6_5.
34. Kuhn M, Johnson K (2013) Applied predictive modeling, vol 26. Springer, New York