



An investigation of feature selection methods for soil liquefaction prediction based on tree-based ensemble algorithms using AdaBoost, gradient boosting, and XGBoost

Selçuk Demir¹ · Emrehan Kutlug Sahin¹

Received: 11 November 2021 / Accepted: 21 September 2022 / Published online: 8 October 2022
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract

Previous major earthquake events have revealed that soils susceptible to liquefaction are one of the factors causing significant damages to the structures. Therefore, accurate prediction of the liquefaction phenomenon is an important task in earthquake engineering. Over the past decade, several researchers have been extensively applied machine learning (ML) methods to predict soil liquefaction. This paper presents the prediction of soil liquefaction from the SPT dataset by using relatively new and robust tree-based ensemble algorithms, namely Adaptive Boosting, Gradient Boosting Machine, and eXtreme Gradient Boosting (XGBoost). The innovation points introduced in this paper are presented briefly as follows. Firstly, Stratified Random Sampling was utilized to ensure equalized sampling between each class selection. Secondly, feature selection methods such as Recursive Feature Elimination, Boruta, and Stepwise Regression were applied to develop models with a high degree of accuracy and minimal complexity by selecting the variables with significant predictive features. Thirdly, the performance of ML algorithms with feature selection methods was compared in terms of four performance metrics, Overall Accuracy, Precision, Recall, and F-measure to select the best model. Lastly, the best predictive model was determined using a statistical significance test called Wilcoxon's sign rank test. Furthermore, computational cost analyses of the tree-based ensemble algorithms were performed based on parallel and non-parallel processing. The results of the study suggest that all developed tree-based ensemble models could reliably estimate soil liquefaction. In conclusion, according to both validation and statistical results, the XGBoost with the Boruta model achieved the most stable and better prediction performance than the other models in all considered cases.

Keywords AdaBoost · Boruta · Liquefaction · Recursive feature elimination · Stepwise feature selection · XGBoost

1 Introduction

The term “liquefaction” is frequently known as the transformation of saturated loose sandy soil from a solid to a liquid state as a result of an increase in pore water pressures and consequent complete loss of effective stresses when it is subjected to strong and rapid seismic loading conditions.

This phenomenon was not seriously considered by engineers until 1964 [1]. However, the Alaska and the Niigata earthquakes in 1964 led to significant damages to the environment, structures, and underground facilities due to soil liquefaction [2–4]. The damaging effects of liquefaction were also observed during recent earthquakes [5–8]. Therefore, it is crucial to appropriately evaluate the soil liquefaction potential when designing any soil-structure system. In this regard, many researchers commenced extensive investigations on earthquake-induced soil liquefaction and its prediction.

Various empirical or semi-empirical methods are proposed to evaluate liquefaction potential. The most common way for estimation of soil liquefaction is the stress-based approach called simplified procedure [9]. This approach

✉ Selçuk Demir
selcukdemir@ibu.edu.tr

Emrehan Kutlug Sahin
emrehansahin@ibu.edu.tr

¹ Department of Civil Engineering, Bolu Abant Izzet Baysal University, 14030 Bolu, Turkey

needs the capacity of soil to resist liquefaction. The liquefaction resistance of soil can be determined from laboratory tests or in-situ geotechnical investigations. In-situ tests are much preferred for soil liquefaction evaluation due to the defects of laboratory tests, such as representing actual field conditions and obtaining high-quality soil samples. Some of the in-situ tests preferred for liquefaction triggering analysis are shear wave velocity test, cone penetration test (CPT), and standard penetration test (SPT) [10–17]. Moreover, numerous nonlinear numerical analyses have been performed for the calculation of soil liquefaction. However, numerical approach may require a sophisticated constitutive soil model and solid knowledge during dynamic analyses. Therefore, the complexity of the model may render numerical approach ineffective and time-consuming. Considering these facts, nonconventional approaches (e.g., machine learning) become an alternative tool for providing solutions to various problems in this topic. Machine learning (ML) methods have an adequate capacity to capture the potential correlations among information without any prior assumptions [18].

Currently, ML methods, such as support vector machine (SVM), logistic regression (LR), artificial neural network (ANN), random forest (RF), naive bayes, eXtreme gradient boosting (XGBoost), decision tree, and etc., are being increasingly used in different engineering applications. With regard to geotechnical engineering, employing ML methods can supply our understanding of the complex behavior of liquefiable soils. It can be also a contributory tool for improving liquefaction hazard analysis [19]. According to Xie, et al. [20], various ML techniques have been adopted many times for liquefaction assessment [21–33].

In recent years, ensemble algorithms have gained great attention in various fields due to the predictive capabilities of these methods [31, 34–37]. Ensemble method is an algorithm that combines multiple classifiers to solve a complex problem and improve the model's performance. The idea of ensemble learning is to reduce the chance of error while expanding the general dependability and certainty of the model [38]. Over the past decades, widely used algorithms such as SVM, ANN, and LR have showed strong performance for both regression and classification tasks. However, tree-based ensemble (TBE) approaches have significantly improved performance and are becoming more widely accepted methods [39]. Bagging [40] (i.e., RF) and boosting (i.e., XGBoost) are the classical well-known TBE approaches. Bagging and boosting algorithms mainly differ from each other in two aspects. First, while the boosting algorithm mainly utilizes weighted averages to make multiple weak learners into stronger learners, bagging (which stands for Bootstrapping) is a combination of multiple independent learners [41]. Second, the boosting

algorithm aims to produce an ensemble model less biased, whereas the bagging algorithm mainly aims to get strong models with less variance than its components [42]. Perhaps not surprisingly, TBE methods have become a particularly popular approach since it combines properties from both statistical and ML methods [43] and they provide a high accuracy prediction model with ease of interpretation and feature importance analysis for large datasets. Also, ensemble methods based on decision trees (e.g., RF and XGBoost) have been widely used to deal with nonlinear problems. Beside, the primary advantages of TBE methods are that they work on fewer assumptions compared to other existing alternatives for classification (such as SVM, LR, and discriminant analysis) and they require minimum data pre-processing [43, 44]. With the growing interest in ML, many studies have been conducted in various domains by using TBE methods, such as information science [45], biological [46], energy [47], and healthcare [48]. In these studies, prediction results are compared with other classification methods. They are concluded that the TBE methods provide better prediction accuracy than the other classification models.

TBE methods have also been successfully introduced in geotechnical engineering. For example, Wang, et al. [35] and Bharti, et al. [36] applied the XGBoost approach for the slope stability problems. Zhang, et al. [37] adopted XGBoost and RF algorithms to predict the relationship between the undrained shear strength and several basic soil parameters. Pham, et al. [49] utilized the Adaptive Boosting (AdaBoost) algorithm for the classification of soils, and the results indicated that AdaBoost offers important results for soil samples by the automatic classification. Wang, et al. [50] used five different algorithms, including gradient boosting machine (GBM) and RF to estimate bearing deformation and column drift ratio responses of extended pile-shaft-supported bridges. They concluded that the GBM algorithm well predicted the seismic responses of the soil-bridge systems as compared to other studied methods. However, the literature review shows that advanced boosting algorithms such as GBM and XGBoost have been rarely employed for liquefaction prediction [31, 51]. Moreover, no previous study investigated the AdaBoost algorithm for the seismic soil liquefaction prediction. Indeed, there is not enough example of a quantitative-systematic comparison of boosting algorithms in liquefaction prediction. As previously stated, the goal is to develop the process of building ML models not only using robust algorithms, such as the cases of ensemble learners but also simpler and faster learning algorithms seeking to assess which of the algorithms can better predict. Considering many advantages of TBE methods (e.g., reliability, robustness, and high accuracy) and the lack of the liquefaction prediction studies based on TBE methods in the

literature, AdaBoost, GBM, and XGBoost algorithms are applied in the present study.

It is well known that when building an ML-based model for making a prediction, lots of data and features are required. Not all features in the dataset may be necessary during the modeling phase. The principal goal for engineers or researchers is to reach the best predictive ability of the created model. Hence, removing the irrelevant data may be contributed minimizing the errors, enhancing learning accuracy, and reducing the computation time [52–54]. Furthermore, using the dataset without pre-processing would increase the overall complexity of the model. This limitation can be remedied by using feature selection (FS) methods as it reduces the size of the training dataset and removes the superfluous features. FS is a procedure of specifying an optimal subset of features through all possible combinations of feature subsets from the original dataset, which reduces the number of predictors as far as possible without compromising predictive performance [55]. Das, et al. [54] determined important input features of SPT, CPT, and Vs datasets using FS methods in a multi-objective optimization framework. Hu [56] used the filter method, which is one of the classes of FS methods, to move all irrelevant variables for gravelly soil liquefaction. Demir and Sahin [57] applied the RFE method as an efficient FS technique for liquefaction prediction. All studies concluded that FS methods are able to improve the predictive capability of models. Therefore, identifying relevant features of a dataset is a noteworthy process in the preparation of the prediction model.

The objective of the proposed study is to predict the soil liquefaction through AdaBoost, GBM, and XGBoost algorithms considering three FS methods, namely Recursive Feature Elimination (RFE), Boruta, and Stepwise Regression (SR), for enhancing the performance of TBE algorithms. For this, 620 SPT case studies collected from Kocaeli and Chi-Chi earthquakes are used in the experiments. Four performance metrics such as Overall Accuracy, Precision, Recall, and F-measure were used to measure the performance of the models. The entire analyses were performed with the *R* package software [58]. The novelty of this paper can be summarized with the following headings: (1) The application of GBM and XGBoost is still rare in the prediction of soil liquefaction. In addition, the AdaBoost algorithm was first time applied in this study and compared the other studied boosting algorithms for proper liquefaction prediction assessment. The investigation of these algorithms and their comparison with each other is highly necessary to reach sufficient background and obtain some proper findings. (2) The growing popularity of FS methods and their frequent application raise new questions about their influence on the prediction performance of the

models. Hence, the results of RFE, Boruta, and SR methods were compared to the original dataset including all features to enhance our understanding in terms of the skills of these algorithms in providing the optimal features. (3) One of the important things when building a prediction model is sampling. In current ML-based liquefaction prediction practices, data is randomly subdivided into training and testing samples by generally using the simple random sampling (SRS) technique, which may be problematic when the distribution of the liquefaction events in the dataset is imbalanced [55]. However, in this study, the training and test samples were produced through the Stratified Random Sampling (StrRS) technique to ensure the selection of balanced samples. This approach generates random sampling points and distributes them equally between each class (liquefied and non-liquefied). (4) A nonparametric statistical test called the Wilcoxon sign rank test [59] was applied to find out whether there is a statistical significance between the prediction results of the algorithms. (5) Lastly, computation costs of the TBE algorithms were evaluated in the cases of parallel and non-parallel processing.

2 Methodology

This section presents the theoretical details of TBE algorithms and FS methods applied in this study. Moreover, an introduction to the liquefaction dataset and the performance measurement and accuracy assessment steps are briefly mentioned. A flowchart of the methodology is presented in Fig. 1.

2.1 Description of the dataset

A total of 620 SPT records with 12 parameters collected from the two major earthquakes in 1999 are considered for the purpose of the study [24]. The dataset consists of binary classification, including 256 liquefied (Yes) and 364 non-liquefied (No) cases. The further details about the dataset are summarized in Table 1.

2.2 Overview of tree-based ensemble (TBE) algorithms

Ensemble algorithms utilize several weak learners and aggregate their outcomes to improve the performance of a model. There are several types of TBE algorithms. Among them, AdaBoost, GBM, and XGBoost were employed to predict soil liquefaction. A brief discussion of the three algorithms is provided here.

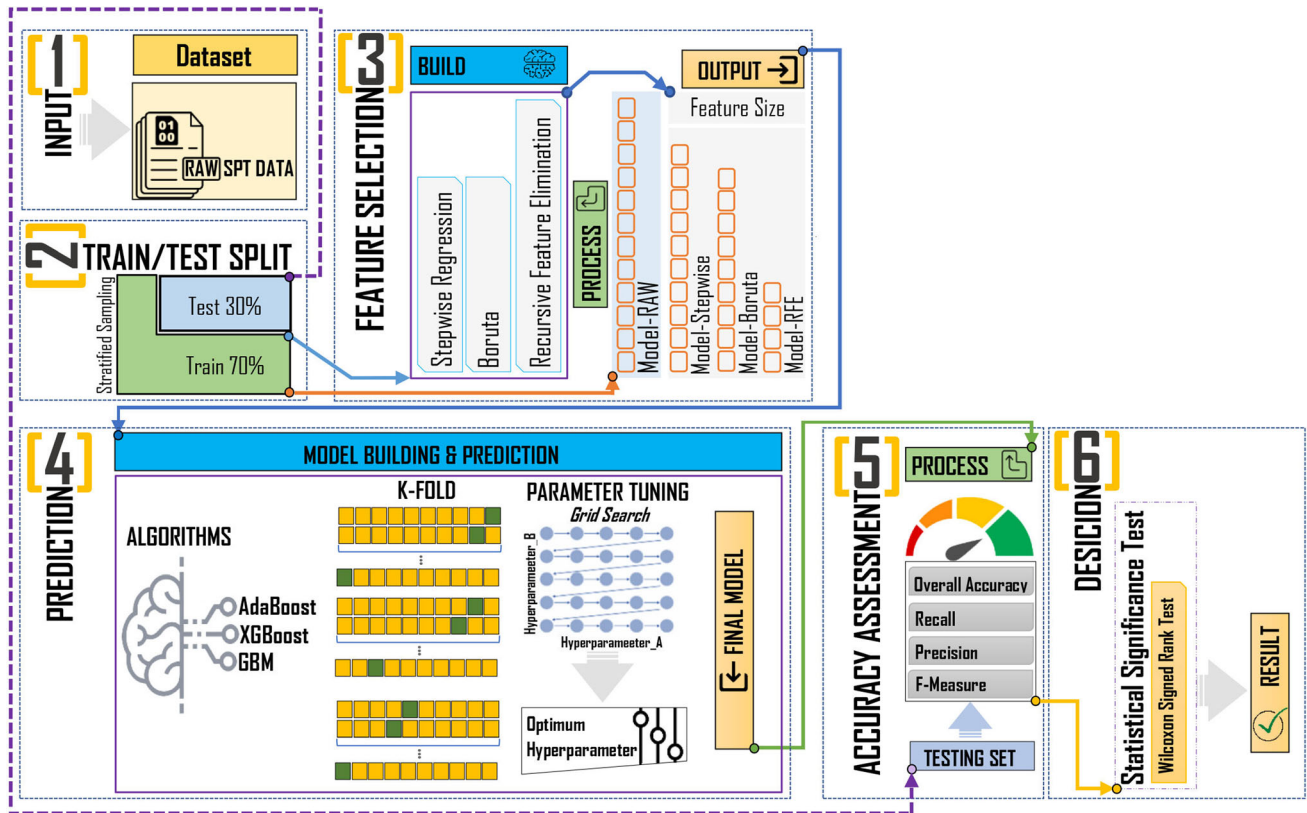


Fig. 1 The route of the methodology followed in this study

Table 1 Parameters and some statistical measures of the dataset [24]

No	Parameter	Notation	Unit	Mean	Median	Min	Max
1	The depth of the soil layer	z	m	7.655	6.7	0.8	19.8
2	SPT blow number	$(N_1)_{60}$	–	14.480	11	1	75
3	Groundwater table depth	d_w	m	1.454	1.1	0.35	10
4	Total vertical stress	σ_v	kPa	144.60	121.6	12.1	408.9
5	Effective vertical stress	σ'_v	kPa	82.475	68.15	7.5	233.7
6	Peak horizontal acceleration	a_{max}	g	0.384	0.4	0.18	0.67
7	Threshold acceleration	a_t	g	0.074	0.06	0	0.85
8	Fine content less than 75 μm	FC	%	62.990	74.5	1	100
9	Internal soil friction angle	ϕ'	($^\circ$)	31.962	31.41	23.46	52.08
10	Cyclic stress ratio	CSR	–	0.3737	0.39	0.12	0.77
11	Earthquake magnitude	M_w	–	7.494	7.4	7.4	7.6
12	Shear wave velocity	V_s	m/s	166.98	155	37	500

2.2.1 Adaptive boosting (AdaBoost)

AdaBoost is one of the most commonly applied boosting algorithms introduced by Freund and Schapire [60]. The AdaBoost algorithm exhibits an efficient performance since it is capable of generating expanding diversity. It has been successfully applied for solving two-class, multi-class single-label, multi-class or multi-label, and categories of single-label problems. This algorithm is an iterative

process that tries to generate a strong classifier with weak classifiers [61]. The weak classifiers are chosen to minimize the errors in each iteration step during the training process and used to build a much better classifier so that boosts the performance of the weak classification algorithm [62]. This boosting is accomplished by averaging the output of the set of weak classifiers. Pseudocode for the AdaBoost algorithm is presented in Algorithm 1 [53].

Algorithm 1. AdaBoost

```

0: START
1: Input:
   Dataset:  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ 
   Parameters: Learning Algorithm
                  The number of iterations
                  Maximum depth of tree
2: Process   for  $t = 1 \dots T$ :
3:             Initialize the weight using distribution  $D_t$ :
                  $D_1(i) = 1/m, i = 1 \dots m$ 
4:             Fit a classifier  $G_m(x)$  to the training data using
                 weights  $w_i$ 
5:             Compute weighted error:
                  $err_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}$ ,  $I(\bullet)$  represents indicator function
6:             Compute weighting factor:
                  $\alpha_m = \log((1 - err_m) / (err_m))$ 
7:             Update the weight distribution:
                  $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))], i = 1, 2, \dots, N$ 
8: Output   Compute class outputs:
                  $G(x) = \text{sign} \left[ \sum_{m=1}^M \alpha_m G_m(x) \right]$ 
9: END

```

2.2.2 Gradient boosting machine (GBM)

GBM can be utilized for both classification and regression problems in terms of ML applications. GBM is used to find an additive model that will minimize the loss function, which is to construct the new base-learners to be maximally correlated, associated with the whole ensemble [63]. It is possible to assign the loss function arbitrarily. If the

error function is a classic squared-error loss, the process of learning may result in consecutive error-fitting to achieve a better intuition. A GBM model mainly contains a few hyperparameters such as *max_depth*, *min_rows*, *ntrees*, *col_sample_rate* and *learn_rate*. The GBM algorithm may be summarized as the following pseudocode given in Algorithm 2 [63].

Algorithm 2. Gradient Boosting Machine

```

0: START
1: Input:
   Dataset:  $(x, y)_{i=1}^N$ 
   Parameters:
     n.trees= total number of trees ( $M$ )
     max_depth= max. depth of each tree
     min_rows = min. number of observations for a leaf
     learn_rate = the rate at which algorithm learns when
     building a model
     col_sample_rate = column sampling rate
2: Process
   for  $t=1$  to  $M$  do
3:   Compute the negative gradient:
      $g_t(x)$ 
4:   Fit a new base-learner function:
      $h(x, \theta_t)$ 
5:   Find the best gradient descent step-size  $\rho_t$ :
     
$$\rho_t = \arg \min_{\rho} \sum_{i=1}^N \psi \left[ y_i, \hat{f}_{t-1}(x_i) + \rho h(x_i, \theta_t) \right]$$

6:   Output
     Update the function estimate:
     
$$\hat{f}_t \leftarrow \hat{f}_{t-1} + \rho_t h(x, \theta_t)$$

7: END

```

2.2.3 eXtreme gradient boosting (XGBoost)

The open-source XGBoost is a free-to-use library of the gradient boosted tree algorithm that has recently dominated science competitions for structured or tabular data. The XGBoost combines all the predictions of a set of weak learners by combining several of them to create a strong learner that obtains better prediction performances [64]. This method needs to decide the primary hyperparameters for the prediction of the model. Every ML algorithm achieves the best performance of the model with the best hyperparameters, so appropriate tuning is particularly

important, including XGBoost [65]. Therefore, the grid search (GS) method is utilized to reach the appropriate model hyperparameters of XGBoost, namely *eta* (the learning rate), *subsample* (subsample ratio of the training instance), *max_depth* (maximum depth of a tree), *gamma* (minimum loss reduction), *colsample_bytree* (subsample ratio of columns when constructing each tree), *min_child_weight* (minimum sum of instance weight) and *nrounds* (number of boosting iterations). The package named as “*xgboost*” from the “*caret*” library in *R* was used to perform XGBoost operations [66]. The pseudocode of the XGBoost was given in Algorithm 3 [67].

Algorithm 3. eXtreme Gradient Boosting

```

0: START
1: Input:
   Dataset: Dataset or Features
   Parameters: rounds = max. number of boosting iterations ( $M$ )
                  max_depth= max. depth of a tree
                  eta = control the learning rate
                  gamma = min. loss reduction
                  colsample_bytree = subsample ratio of columns
                  min_child_weight = min. sum of instance weight
                  subsample = subsample ratio of the training instance
2: Process
   for  $t=1$  to  $M$  do
3:   Compute  $g_i = \partial_{\hat{y}_i^{(m-1)}} LOSS_{XGBoost}(y_i, \hat{y}_i^{(m-1)})$ 
4:   Compute  $h_i = \partial_{\hat{y}_i^{(m-1)}}^2 LOSS_{XGBoost}(y_i, \hat{y}_i^{(m-1)})$ 
5:   Use the statistics to greedily grow a new tree  $f_m(x_i)$ :
      
$$obj^{(m)} = -\frac{1}{2} \sum_{j=1}^M \frac{G_j^2}{H_j + \lambda} + \gamma M$$

6:   Add the best tree  $f_m(x_i)$  into current model:
      
$$\hat{y}_i^{(m)} = \hat{y}_i^{(m-1)} + \epsilon f_m(x_i)$$

7:   Compute all sub-trees along the  $M$ 
8:   Obtain a strong regression tree based on all weak regression sub-trees
   Output
   The estimated probability based on the strong regression tree
9: END

```

2.3.1 Recursive feature elimination (RFE)**2.3 Feature selection (FS) methods**

FS is a part of developing predictive models in ML for reducing the number of input variables. The potential benefits of FS consist of facilitating data understanding, shortening computational cost, and getting rid of the problem of dimensionality to improve the performance of the prediction model [68]. Several FS methods have been developed to obtain which features are most relevant and should be used in prediction models. In this study, three FS methods RFE, Boruta, and SR have been utilized to select only important and relevant features. The detail of the three FS methods is described in the following sections.

RFE is commonly used for the FS method proposed by Guyon, et al. [69]. RFE is used to rank the features in a dataset according to the importance provided by the RF algorithm. The RFE method contains several main steps [70, 71]. Firstly, the importance of each feature is calculated for each iteration in the process of the feature elimination step. Secondly, the features are sorted from high to low according to their importance value. Finally, the least important feature(s) is removed from the model. After this step, the model is built again, and feature importance scores are recalculated. This process has recurred until a feature is found not to be redundant or irrelevant. Detailed information on the RFE method is given in Algorithm 4 as pseudocode [72].

Algorithm 4. Recursive Feature Elimination

```

0: START
1: Input:
   Dataset: Dataset or Feature  $D = \{x_i, y_i\}_{i=1}^n$ 
2: Process Compute Random Forest (RF)
3: Rank the importance of features
4: for each subset size  $S_i, i = 1 \dots n$  do
5:     eliminate the least important feature
6:     train RF with  $S_i - 1$  predictors
7:     estimate RF performance
8: end
9: Calculate the performance profile over the  $S_i$ 
10: Output Best Feature Subset
11: END

```

2.3.2 Boruta

Boruta (Algorithm 5 [73]) is a wrapper-built feature ranking and selection method based on RF for feature relevance estimation in the *R* statistical package [74]. The importance of features is established in the RF algorithm. Calculating variable importance with RF is possible by the measure of accuracy decreasing when information about variables in a node is removed from the model. Similar to the RF algorithm, the Boruta method is based on adding randomness to the model and collecting results from the ensemble of randomized samples [75]. The Boruta method involves the following steps [66]; (1) Duplicating all features to extend the information system, (2)

shuffling the added attributes to eliminate their correlations with the response, (3) running the RF model on the extended system and gathering importance (*Z*) scores, (4) gaining the MZSA (the maximum *Z* score among the duplicated (i.e. shadow) features) value and assigning a hit to every feature that scored better than MZSA, (5) applying a two-sided equality test with the MZSA for each feature with undetermined importance, (6) assuming the features which have less importance than MZSA as unimportant and removing them from the information system permanently, (7) removing duplicated variables, and (8) repeating the procedures from step (1) to (7) until the importance is assigned for all attributes. The detail of the method is clearly described in Kursa and Rudnicki [74].

Algorithm 5. Boruta

```

0: START
1: Input:
   Dataset: Dataset or Features
   Parameters: RFruns = the number of RF runs
2: Process for each RFruns do
3:     originalPredictors (OP)  $\leftarrow$  originalData (OD) (predictors)
4:     shadowAttr  $\leftarrow$  permute (OP)
5:     extendedPredictors (EP)  $\leftarrow$  cbind (o, shadowAttr)
6:     extendedData (ED)  $\leftarrow$  cbind (EP, OD (decisions))
7:     zScoreSet  $\leftarrow$  RF (ED)
8:     MZSA  $\leftarrow$  max (zScoreSet(shadowAttr))
9:     for each  $a \in$  OP do
10:        if zScoreSet(a) > MZSA then
11:            hit(a)++
12:    for each  $a \in$  OP do
13:        S(a)  $\leftarrow$  two-sided equality test (a), S: Significance
14:        if S(a) >> MZSA then
15:            confirmedSet  $\leftarrow$  finalSet  $\cup$  a
16:        else if S(a) << MZSA then
17:            rejectedSet  $\leftarrow$  rejectedSet  $\cup$  a
18:    return finalSet  $\leftarrow$  rejectedSet  $\cup$  confirmedSet
19: Output Best Feature Subset
20: END

```

2.3.3 Stepwise regression (SR)

SR [76] is the most well-known method for choosing features in a model that keeps relevant features and removes irrelevant or redundant ones. Indeed, SR was developed as an FS procedure for linear regression models that is a combination of the forward and backward selections. The focus of the method is to modify the forward selection so that after each step of the algorithm, all candidate variables in the model are checked to see if their

significance has been reduced below the specified tolerance level. At the end of the process, if a non-significant variable is observed, it is excluded from the base model [77]. SR requires two independent statistical significance cut-off values for adding and deleting variables from the model [78]. More detail about the mathematic background of SR can be found in the literature [76, 78]. The pseudocode of the SR method is shown in Algorithm 6 [79].

Algorithm 6. Stepwise Regression

```

0: START
1: Input:
   Dataset: Features
2: Process
3:   Initially all observations are in a set
4:   Start from the root node, partition the samples using
   the following recursive procedure
5:   Giving a tree node
6:   Perform stepwise linear regression on samples in
   current node
7:   Calculate the residual sum square ( $RSS_{node}$ )
8:   if
9:     Number of samples > predefined minimum node size
10:    for
11:      Each predictor variable  $x_i$ 
12:      sort
13:         $x_i$  in ascending order
14:      for
15:        Each value  $d$  in the above sorted list
16:        Using  $d$  as the threshold value
17:        Compute stepwise regression on each
        subset and perform the corresponding  $RSS$ 
18:        Calculate the subtotal  $RSS$  as the sum of
        those of the two subsets
19:      End of for loop on threshold value  $d$ 
20:      Find the minimum subtotal  $RSS$  achievable from
        splitting current node on  $x_i$ 
21:    End of for loop on predictor variable  $x_i$ 
22:    Find the minimum  $RSS$  of all possible splits of
        current node ( $MinRSS_{splitting}$ )
23:    Calculate the improvement from splitting the
        current node
24:    if
25:      Improvement > predefined minimum improvement
26:      Split current node into two new nodes using the
        variable and threshold value that give the
         $MinRSS_{splitting}$ 
27:      for
28:        Each of the two new nodes → go to step 5
29:      End of if in step 26
30:    End of if in step 24
31:  End of recursion
32: Output
33:   Best Feature Subset
34: END

```

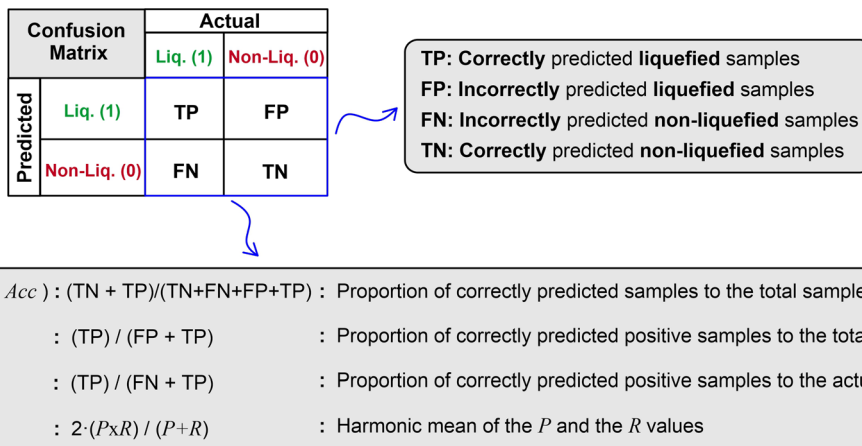


Fig. 2 Description of CM and performance matrices

2.4 Performance evaluation methodology

There are many kinds of performance evaluation metrics to evaluate classification performance of ML models. In this study, the metrics of Overall Accuracy (*Acc*), Precision (*P*), Recall (*R*), and F-measure (*F*) were used. The performance of two-class classification models is described based on the Confusion Matrix (CM). Accordingly, CM parameters namely TP, FN, FP, and TN were utilized to compute performance evaluation metrics as shown in Fig. 2

The accuracy of the liquefaction prediction produced by the ML models is estimated from the CM for the validation data. The produced models can show good performance results considering the measurements, but it would be appropriate to use a statistical significance test to determine the best single model among the other produced models. Therefore, Wilcoxon’s sign rank test [59], which is one of the most important nonparametric tests for multiple comparisons, was used to identify significant differences between the models.

3 Results and discussion

Several TBE algorithms and FS methods were applied in the present work. First, three FS methods were used to detect the most relevant features according to their importance. The objective of FS is to remove irrelevant and redundant features by keeping the ones that can predict the optimum feature. The FS process might help decrease the computational time, improve the performance of the algorithm, and prevent overfitting. After the FS process, for the purpose of analyzing the best feature subset TBE algorithms were employed. Besides, all feature combinations (i.e., RAW SPT data including 12 features) was also

considered as a comparison model. The most important fact in optimum model preparation is that model accuracies mainly depend on the selected hyperparameters. Therefore, the tenfold cross-validation, which is randomly partitioned into *k* equal sized subsamples, was applied to the proposed models for hyperparameter tuning. The algorithms namely AdaBoost, GBM, and XGBoost were utilized to decide the overall best performing one among models. The quality of the resulting models was evaluated using *Acc*, *P*, *R*, and *F* metrics, respectively. Wilcoxon’s sign test was also utilized to acquire the statistical differences of the accuracies of models. For interest, applications in the proposed methodology were performed using the *R* programming language (version 3.6.3) [58] with the following main *R* packages: *caret*, *sp*, *randomForest*, *Boruta*, *adabag*, *gbm*, *e1071*, *h2o*, and *xgboost*, respectively. All applications for liquefaction assessment were performed on a PC with 4.0 GHz AMD Ryzen 9 3950X CPU, 64 GB RAM, and Windows 10 operating system.

3.1 Determination of training and test sample size

ML algorithms build a model that relies on training samples in order to make predictions or decisions. Therefore, training sample size had a larger impact on model accuracy than the algorithm used [80]. However, there is no advice or exact ratio for a minimum number of samples required for ML prediction. It could be said that the determination of the best sample size for the prediction of the model may depend on the ML algorithm, the number of input variables, and the size of the original database [81]. Additionally, another important aspect of determining the training and test sample size is the training data selection method. Sampling strategies can be divided into two types namely, probability or random sampling and non-

probability sampling. In a random sampling technique, each member of the sampling unit has an equal chance of being selected in the sample. There are several random sampling techniques available for managing sampling sizes [82, 83]. Non-probability sampling is a population using a subjective (i.e., non-random) which the user selects samples based on subjective judgment rather than random selection [84]. The following examples of non-probability sampling methods can be found in the literature including, quota, snowball, judgment, and convenience sampling.

Sampling is the technique of selecting a subset of a population from the entire population for the purpose of determining the characteristics of the whole population to make statistical inferences. There are several types of sampling techniques but two sampling techniques namely Simple Random Sampling (SRS) and Stratified Random Sampling (StrRS) are the most preferred methods in the ML area. SRS is the basic sampling technique where a group of samples was selected from a population. This sampling method is the most appropriate option when the entire population from which the sample is taken is homogeneous. Otherwise, StrRS would be the ideal approach in circumstances where the population is heterogeneous or dissimilar. In this study, StrRS was used as a sampling size strategy. The population is directly divided into subgroups in this method and a random sample is taken from each subgroup, meaning each subgroups sample has the same sampling fraction. These mentioned subgroups are called strata. The main advantage of StrRS is that it captures key population characteristics in the sample and the process of stratifying reduces sampling error with ensuring a greater level of representation. Usually, training data size is set to split in the ratio of 60:40, 70:30, or 80:20 (training/testing set). The training dataset is used for model building and test dataset is utilized for model evaluation. On the other hand, many researchers proposed a ratio of 70:30 or 80:20 for producing datasets [26, 29, 31–33, 54, 57, 85]. In this study, the ratio of 70:30 (training/test set) was chosen like other literature research

[54, 57, 85, 86] and SPT data (a total of 256 “Yes” and 364 “No”) was used for the analysis. When the distributions of the two classes (i.e., Yes and No) in the dataset are compared, the No class is proportionally more than the Yes class. It has been revealed that the classes are not homogeneously distributed. In other words, the distribution of the liquefaction events in the dataset is imbalanced. Therefore, the SPT data is divided into training/test set using the StrRS technique. After the sampling process, training data is contained 179 events of “Yes” and “No”, and test data is contained 77 events of “Yes” and “No” liquefaction events. As a result, characteristics of both training and test sample sizes became proportional to the entire population into homogeneous units. Also, a comparative example of between the two sampling strategies are given in Table 2. The mean values of liquefied events for the dataset using StrRS and SRS techniques are 0.500 and 0.4124, respectively. This finding shows that the distribution of each sample (i.e., Yes and No) is disproportionate for SRS. Thus, using the StrRS technique will guarantee that each class has sufficient samples.

3.2 Feature selection for dimensionality reduction

FS methods select variables in the original dataset which are more important and relevant for the prediction process and remove unrelated ones. FS methods provide several benefits to circumvent the curse of dimensionality, increasing learning process speed, simplifying models, and improving the quality of ML methods as well as training efficiency [87, 88]. In this study, three different FS methods RFE, Boruta, and SR were compared. Table 3 shows the ranks of all selected features and their feature importance (*FI*) scores.

The model was initially applied to the training data by utilizing RF-*FI* analysis to identify which features are the most effective in liquefaction prediction. The most important features obtained from RF-*FI* scores are given in

Table 2 A comparative example of the difference between StrRS and SRS techniques according to SPT dataset

SPT Dataset							
Yes (256)	No (364)						
The approximate selected training ratio	Train/Test ratio		Class distribution in Train Data	Class distribution in Test Data	Statistic of Train Set		
	Train	Test			Y/N	Y/N	Mean
<i>Stratified Random Sampling Technique</i>							
58%	70%	30%	179/179	77/77	0.5000	0.2507	
<i>Simple Random Sampling Technique</i>							
70%	70%	30%	179/255	77/109	0.4124	0.2429	

Table 3 Selected factors and their importance scores estimated by FS methods

Model Method	Model_RFE		Model_Boruta		Model_SR		Model_RAW	
	RFE	FI	Boruta	FI	SR	FI	RAW_Data	RF-FI
1	<i>FC</i>	27.22	ϕ'	13.69	ϕ'	4.84	<i>FC</i>	33.11
2	ϕ'	23.35	$(N_1)_{60}$	13.24	<i>CSR</i>	3.14	ϕ'	30.29
3	$(N_1)_{60}$	19.88	<i>CSR</i>	11.11	<i>FC</i>	6.97	$(N_1)_{60}$	21.62
4	<i>CSR</i>	17.07	<i>z</i>	10.71	a_t	2.59	<i>CSR</i>	17.48
5			σ_v	8.90	V_s	2.04	<i>z</i>	14.10
6			a_{max}	7.87	N	1.62	σ_v	13.76
7			σ'_v	7.30	d_w	2.96	σ'_v	13.40
8			V_s	4.96	σ'_v	2.75	d_w	11.11
9			<i>FC</i>	4.55	σ_v	3.22	a_t	9.47
10					a_{max}	1.43	V_s	8.93
11							a_{max}	4.02
12							M_w	1.03

Table 3. The features namely *FC*, ϕ' , and $(N_1)_{60}$ were found the most important parameters based on the SPT dataset. On the other hand, a_t , V_s , a_{max} , and M_w was the less effective features based on the RF-FI score analysis. It is important to note that these results only show which of the features are more or less important. After the model evaluation using the entire set of features (i.e., RAW data) as input features, the FS methods RFE, Boruta, and SR were performed for identifying the least important feature/s and removing them from the dataset. When the FS results were analyzed, it was seen that the RFE, Boruta, and SR methods determined 4 (*FC*, ϕ' , $(N_1)_{60}$, and *CSR*), 9 (ϕ' , $(N_1)_{60}$, *CSR*, *z*, σ'_v , σ_v , a_{max} , V_s , and *FC*), and 10 (ϕ' , *CSR*, *FC*, a_t , V_s , $(N_1)_{60}$, d_w , σ'_v , σ_v , and a_{max}) parameters as the most effective features, respectively. Evaluating the based-on FS methods overlap taken over all each selected feature revealed that ϕ' and *CSR* were found to be common features. The results of Boruta and SR was found to be very similar in that most have a very similar ranking. The biggest difference in performed FS analyzes was seen between the RFE and the other two methods because RFE is a greedy optimization algorithm that eliminates most features. As a result of the FS processes, new data models were created, and each feature model was given a new name such as *Model_RFE*, *Model_Boruta*, and *Model_SR*. In addition, RAW_Data (i.e., original of SPT data) was used as a benchmark model for an objective comparison with the other models (i.e., *Model_RFE*, *Model_Boruta*, and *Model_SR*).

3.3 Optimization of hyperparameter with grid-search

Hyperparameter optimization in ML aims to detect the optimum hyperparameters that deliver the best

performance as measured on a validation set. One of the methods to tune ML problems is the *k*-fold cross-validation (CV). CV is also a very useful approach in cases where needed to mitigate overfitting and provide a less biased estimation of a tuned model’s performance on the dataset. In this approach, the training data are randomly split into *k* subgroups (e.g., *k* = 10 and becoming tenfold CV), and the model is then run *k* times with one of the subsets held back for validation each time. Importantly, each subset in the data sample is designated to an individual group and stays in that group for the duration of the procedure. The results of each run are evaluated using the pending data, and the results are averaged across all *k* scores [81]. At the end of the process, the group that gives the best performance, commonly defined based on the mean of the model skill scores, is chosen.

In this study, to make sure that each fold is a good representative of the whole data, StrRS technique was used as a training sample size strategy. The SPT dataset was split into training/test set in the ratio of 70:30 using StrRS for hyperparameter estimation and performance analysis. The training data, which is offered as the best overall performance score by *k*-fold CV, was used for training the model, and the test data (or validation data) was used for setting the evaluating performance of models. The hyperparameter tuning process was performed using Grid Search based on CV with tenfold (Table 4). This was carried on using only the training data to targeting at a better comparison between models. It should be noted that “train” function on the “caret” package was used to find tuning parameters automatically for these models [89].

Table 4 Details of the hyperparameter tuning of methods according to the FS based models

Method	Model	Best parameters found
AdaBoost	<i>RFE</i>	mfinal = 100, maxdepth = 3 and coeflearn = Freund
	<i>Boruta</i>	mfinal = 50, maxdepth = 3 and coeflearn = Freund
	<i>Stepwise</i>	mfinal = 150, maxdepth = 2 and coeflearn = Zhu
	<i>RAW</i>	mfinal = 100, maxdepth = 3 and coeflearn = Breiman
XGBoost	<i>RFE</i>	nrounds = 150, max_depth = 3, min_child_weight = 1, subsample = 0.75, eta = 0.3, colsample_bytree = 0.8, and gamma = 0
	<i>Boruta</i>	ntrees = 150, min_rows = 10, max_depth = 2, col_sample_rate = 1, and learn_rate = 0.1
	<i>Stepwise</i>	nrounds = 100, max_depth = 3, min_child_weight = 1, colsample_bytree = 0.8, subsample = 0.5, eta = 0.3, and gamma = 0,
	<i>RAW</i>	nrounds = 100, max_depth = 3, min_child_weight = 1, subsample = 0.75, colsample_bytree = 0.6, eta = 0.3, and gamma = 0
GBM	<i>RFE</i>	ntrees = 150, min_rows = 10, max_depth = 2, col_sample_rate = 1, and learn_rate = 0.1
	<i>Boruta</i>	ntrees = 100, min_rows = 10, max_depth = 3, col_sample_rate = 1, and learn_rate = 0.1
	<i>Stepwise</i>	ntrees = 150, min_rows = 10, max_depth = 2, col_sample_rate = 1, and learn_rate = 0.1
	<i>RAW</i>	ntrees = 50, min_rows = 10, max_depth = 3, col_sample_rate = 1, and learn_rate = 0.1

3.4 Evaluating performance of models

Using the same dataset with the different types of ML algorithms or using the different datasets with the same type of algorithms produces very different performances. Therefore, this study systematically compares how performance varies as different combinations of FS methods and ML algorithm types are combined. To this end, FS methods namely RFE, Boruta, and SR were used during the selection of the best feature combination. Three types of TBE methods namely AdaBoost, GBM, and XGBoost were utilized as liquefaction prediction algorithms. Performance evaluation of 12 (4×3) models with the different types of prediction algorithms and FS methods combination were analyzed in two perspectives in this study. The first perspective is to calculate the performance of models with *Acc*, *P*, *R*, and *F* metrics based on CM. The second perspective is to identify statistical significance between the models using the Wilcoxon's test. The results of performance metrics of the models in terms of CM are given in Table 5. When the *Acc* results of the models reviewed, the XGBoost classifier performed the highest accuracies among the other FS-based models. In should also be noted that from Table 5, *Model_RFE* and *Model_Boruta* were found as the best feature subset as compared to *Model_SR* and *Model_RAW* for various performance metrics.

It is essential to assess the *P*, *R*, and *F* values for predicting “Yes” and “No” classes to specify the strengths and weaknesses of each method and fully understand the quality of the result [90]. When the CM result in Table 5 was analyzed, almost all three ML methods predicted the “Yes” class better than the “No” class. The *P* value refers to the number of actual “Yes” classes relative to all

samples identified as “Yes”. Higher *P* means that “Yes” classes are correctly mapped than “No” classes. *R* is the percentage of all liquefaction events (i.e., “Yes” and “No”) that are properly identified. Having low *R* values means that the model predicted more False Negatives (should be positive but labelled negative). Higher *R* values indicate that most of the liquefaction events are labelled as “Yes”. *F*-measure is the harmonic mean of the “Yes” and “No” label on validation datasets and the higher *F* value shows that the final model is in making predictions more accurately. Another important observation is that the best performance accuracy for *Model_RAW* was obtained by using the AdaBoost algorithm.

When the overall performance of all pairs of models was analyzed, the XGBoost algorithm based on *Model_Boruta* feature dataset showed the best *Acc* result (*Acc* = 0.9675) when compared to the results of Adaboost with *Model_RFE* (*Acc* = 0.9545) and GBM with *Model_Boruta* (*Acc* = 0.8961) algorithms. It is clearly seen that there are minimal differences between the *Acc* values of the models. However, identifying the significance of the differences between the models should be conducted by statistical analysis. Therefore, Wilcoxon's sign rank test was utilized for the evaluation of the statistical significance of the difference in the performance of models. Wilcoxon's sign rank test with *p-value* was applied for pairwise comparisons of the models. If the calculated *p-value* is lower than or equal to 0.05, it means that the performance of models is different, otherwise, the *p-value* is higher than 0.05, the results are non-significant, and the performance of the model result tends to be the same. During the comparison of the *p-values* of the models, only *Model_Boruta* with XGBoost, *Model_RFE* with AdaBoost, and *Model_Boruta*

Table 5 Comparison of performance results of the models based on CM

Model_Name	Methods									
	AdaBoost			XGBoost			GBM			
	No (%)	Yes (%)		No (%)	Yes (%)		No (%)	Yes (%)		
Model_RFE	No	73 (95%)	3 (5%)	Acc:0.9545 Pre:0.9487	73 (95%)	2 (3%)	Acc:0.9610 Pre:0.9494	61 (79%)	1 (1%)	Acc:0.8896 Pre:0.8261
	Yes	4 (4%)	74 (96%)	Rec:0.9610 F1:0.9548	4 (5%)	75 (97%)	Rec:0.9740 F1:0.9615	16 (21%)	76 (99%)	Rec:0.9871 F1:0.8994
Model_Boruta	No	70 (91%)	1 (1%)	Acc:0.9481 Pre:0.9157	74 (96%)	2 (3%)	Acc:0.9675 Pre:0.9615	66 (95%)	5 (4%)	Acc:0.8961 Pre:0.8675
	Yes	7 (9%)	76 (99%)	Rec:0.9870 F1:0.9500	3 (4%)	75 (97%)	Rec:0.9740 F1:0.9677	11 (5%)	72 (96%)	Rec:0.9351 F1:0.9000
Model_SR	No	67 (87%)	3 (4%)	Acc:0.9156 Pre:0.8810	68 (88%)	3 (4%)	Acc:0.9221 Pre:0.8916	68 (88%)	7 (9%)	Acc:0.8961 Pre:0.8861
	Yes	10 (13%)	74 (96%)	Rec:0.9610 F1:0.9193	9 (12%)	74 (96%)	Rec:0.9610 F1:0.9250	9 (12%)	70 (91%)	Rec:0.9091 F1:0.8947
Model_RAW	No	71 (92%)	5 (6%)	Acc:0.9286 Pre:0.9231	71 (92%)	6 (8%)	Acc:0.9221 Pre:0.9221	64 (83%)	5 (6%)	Acc:0.8831 Pre:0.8471
	Yes	6 (8%)	72 (94%)	Rec:0.9351 F1:0.9290	6 (8%)	71 (92%)	Rec:0.9221 F1:0.9221	13 (12%)	72 (64%)	Rec:0.9351 F1:0.8889

The best performance of a model in each category are shown in bold

with GBM were considered due to their achievement in *Acc* results. The statistical results of the Wilcoxon's sign rank test given in Table 6 show that the performance of the *Model_Boruta* with the XGBoost method was statistically insignificant than the *Model_RFE* with Adaboost method. Furthermore, when the statistical test results of the XGBoost and AdaBoost methods were compared to the GBM results, the *p-value* was found lower than the significance level of 0.05, which means that both methods indicated different performances. Overall, all models performed acceptable results for liquefaction prediction, but XGBoost with *Model_Boruta* exhibited the most stable and best performance according to validation and statistical results.

Table 6 Results of the Wilcoxon's sign rank test

Algorithm	XGBoost Model_Boruta	AdaBoost Model_RFE	GBM Model_Boruta
XGBoost Model_Boruta	1	0.8121	0.04108
AdaBoost Model_RFE		1	0.03109
GBM Model_Boruta			1

Bold values indicates statistical significance *p-value* lower than 0.05

3.5 Computational costs

Training time of ML algorithms according to two types of feature subsets such as *Model_RAW* and *Model_RFE* were calculated considering two different options, parallel and non-parallel processing. The parallel processing is a technique in running program tasks with two or more computer processors (CPUs) to handle separate parts of an overall task. On the contrary, only running one processor to handle all parts of the task is called non-parallel processing. Training and tuning times of two models with three different kinds of ML algorithms are shown in Table 7. It should be noted that the computational costs stated here is only the training time of average tuning (i.e., not including prediction process), which does count the time with each hyperparameter tuning operation. In order to clearly observe computation costs of the ML algorithms, only *Model_RFE*, which contains the least data (four), and *Model_RAW*, which includes all features (twelve), was selected. From Table 7, it is seen that the XGBoost algorithm performed the training/tuning process quickly than the other algorithms for both datasets using parallel and non-parallel processing. On the other hand, the AdaBoost algorithm was computed the training/tuning process with the longest computation time for both parallel and non-parallel options. In addition, the GBM algorithm has been processed at acceptable times, but the used library called

Table 7 Training/Tuning computation times with parallel and without a parallel process

	<i>Model_RAW</i>			
	Training and Tuning time with <i>k</i> -fold		Training and Tuning time without <i>k</i> -fold	
	Parallel	Non-parallel	Parallel	Non-parallel
AdaBoost	2.37 min	39.50 min	56.57 secs	9.02 min
XGBoost	14.05 secs	2.32 min	5.23 secs	9.87 secs
GBM	Not-Supporting	3.59 min	Not-Supporting	1.05 min
	<i>Model_RFE</i>			
	Parallel	Non-parallel	Parallel	Non-parallel
	AdaBoost	2.32 min	31.17 min	59.40 secs
XGBoost	12.65 secs	2.34 min	5.15 secs	10.43 secs
GBM	Not-Supporting	3.78 min	Not-Supporting	1.08 min

“*Package h2o*” does not currently allow parallelization. Hyperparameter tuning, especially with *k*-fold CV, is an expensive process that can benefit from parallelization. The results showed that parallelization processing was beneficial for reducing computation costs for this study. However, *k*-fold CV obviously required a large computation time, in which case other approaches may be more appropriate for users who do not have a powerful computer and time.

4 Conclusions

Soil liquefaction has been accepted as one of the essential risk factors to the seismic performance of structures in liquefaction-prone areas due to its behavioral complexity. Nowadays, ML algorithms have been considered a useful tool for the prediction of soil liquefaction with impressive predicting accuracy. Therefore, this research investigates and compares the prediction performance of the TBE algorithms AdaBoost, GBM, and XGBoost for predicting soil liquefaction. These algorithms are relatively new in geotechnical applications that have rarely been employed to predict soil liquefaction. Also, performances of three different FS methods (RFE, Boruta, and SR) were compared by combining with the TBE algorithms. The results indicated that although all models performed acceptably good performance, the XGBoost algorithm based on the Boruta method (i.e., *Model_Boruta*) achieved the highest overall accuracy ($Acc = 96.75\%$). Besides, XGBoost with *Model_RFE* successfully predicted liquefaction events even though four out of twelve parameters (FC , ϕ' , $(N_1)_{60}$, and CSR) were selected from the original SPT dataset. The Acc value of the XGBoost model were found to be $Acc = 96.10\%$ and $Acc = 92.21\%$ for the four featured model (*Model_RFE*) and the original dataset (*Model_RAW*),

respectively. On the other hand, the XGBoost algorithm required a significantly shorter training time than the other algorithms. At the same time, while *k*-fold CV obviously required a large computation time, the parallel processing utilized by the TBE algorithms except GBM led to reducing computational costs. This study can provide insights into the parameter settings, feature selection, and algorithm selection for liquefaction prediction analysis. Moreover, the results of this study may be helpful for researchers who build models to make a prediction and evaluate the performance of different problems using TBE algorithms and FS methods. In the future, detailed feature engineering strategies may be addressed to improve the performance of these ensemble methods. Also, implementation of relatively new and sophisticated boosting algorithms such as LightGBM, CatBoost etc. may be considered for the further studies to predict soil liquefaction.

Author contributions SD: Conceptualization, Investigation, Writing-review and editing, Writing-original draft, Visualization. EKS: Conceptualization, Methodology, Software, Writing-review and editing, Writing-original draft, Visualization.

Funding No funding was received for conducting this study.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Data availability The dataset analyzed during the current study are publicly available at location cited in the reference section.

References

- Towhata I (2008) Geotechnical earthquake engineering. Springer-Verlag, Berlin
- Ishihara K, Koga Y (1981) Case studies of liquefaction in the 1964 Niigata earthquake. *Soils Found* 21(3):35–52
- Seed HB, Idriss IM (1967) Analysis of soil liquefaction: Niigata earthquake. *J Soil Mech Found Div* 93(3):83–108
- Youd T (2014) Ground failure investigations following the 1964 Alaska Earthquake. In: Proceedings of the 10th National Conference in Earthquake Engineering, Earthquake Engineering Research Institute, Anchorage, AK
- Chen L, Yuan X, Cao Z, Hou L, Sun R, Dong L, Wang W, Meng F, Chen H (2009) Liquefaction macrophenomena in the great Wenchuan earthquake. *Earthq Eng Vib* 8(2):219–229
- Orense RP, Kiyota T, Yamada S, Cubrinovski M, Hosono Y, Okamura M, Yasuda S (2011) Comparison of liquefaction features observed during the 2010 and 2011 Canterbury earthquakes. *Seis Res Lett* 82(6):905–918
- Yasuda S, Harada K, Ishikawa K, Kanemaru Y (2012) Characteristics of liquefaction in Tokyo Bay area by the 2011 Great East Japan earthquake. *Soils Found* 52(5):793–810
- Papathanassiou G, Mantovani A, Tarabusi G, Rapti D, Caputo R (2015) Assessment of liquefaction potential for two liquefaction prone areas considering the May 20, 2012 Emilia (Italy) earthquake. *Eng Geol* 189:1–16
- Seed HB, Idriss IM (1971) Simplified procedure for evaluating soil liquefaction potential. *J Soil Mech and Found Div* 97(9):1249–1273
- Robertson PK, Wride C (1998) Evaluating cyclic liquefaction potential using the cone penetration test. *Can Geotech J* 35(3):442–459
- Andrus RD, Stokoe KH II (2000) Liquefaction resistance of soils from shear-wave velocity. *J Geotech Geoenviron Eng* 126(11):1015–1025
- Cetin KO, Seed RB, Der Kiureghian A, Tokimatsu K, Harder LF Jr, Kayen RE, Moss RE (2004) Standard penetration test-based probabilistic and deterministic assessment of seismic soil liquefaction potential. *J Geotech Geoenviron Eng* 130(12):1314–1340
- Moss R, Seed RB, Kayen RE, Stewart JP, Der Kiureghian A, Cetin KO (2006) CPT-based probabilistic and deterministic assessment of in situ seismic soil liquefaction potential. *J Geotech Geoenviron Eng* 132(8):1032–1051
- Kayen R, Moss R, Thompson E, Seed R, Cetin K, Kiureghian AD, Tanaka Y, Tokimatsu K (2013) Shear-wave velocity-based probabilistic and deterministic assessment of seismic soil liquefaction potential. *J Geotech Geoenviron Eng* 139(3):407–419
- Boulanger R, Idriss I (2014) CPT and SPT based liquefaction triggering procedures. Report No UCD/CGM-14 1
- Boulanger RW, Idriss I (2016) CPT-based liquefaction triggering procedure. *J Geotech Geoenviron Eng* 142(2):04015065
- Cetin KO, Seed RB, Kayen RE, Moss RE, Bilge HT, Ilgac M, Chowdhury K (2018) SPT-based probabilistic and deterministic assessment of seismic soil liquefaction triggering hazard. *Soil Dynam Earthq Eng* 115:698–709
- Zhang W, Li H, Li Y, Liu H, Chen Y, Ding X (2021) Application of deep learning algorithms in geotechnical engineering: a short critical review. *Artif Intell Rev* 54:5633–5673. <https://doi.org/10.1007/s10462-021-09967-1>
- Durante MG, Rathje EM (2021) An exploration of the use of machine learning to predict lateral spreading. *Earthq Spect.* <https://doi.org/10.1177/87552930211004613>
- Xie Y, Ebad Sichani M, Padgett JE, DesRoches R (2020) The promise of implementing machine learning in earthquake engineering: a state-of-the-art review. *Earthq Spect* 36(4):1769–1801
- Goh AT (1996) Neural-network modeling of CPT seismic liquefaction data. *J Geotech Geoenviron Eng* 122(1):70–73
- Pal M (2006) Support vector machines-based modelling of seismic liquefaction potential. *Int J Num Anal Meth Geomech* 30(10):983–996
- Goh AT, Goh S (2007) Support vector machines: their use in geotechnical engineering as illustrated using seismic liquefaction data. *Comput Geotech* 34(5):410–421
- Hanna AM, Ural D, Saygili G (2007) Neural network model for liquefaction potential in soil deposits using Turkey and Taiwan earthquake data. *Soil Dynam Earthq Eng* 27(6):521–540
- Ülgen D, Engin HK (2007) A study of CPT based liquefaction assessment using artificial neural networks. In: 4th international conference on earthquake geotechnical engineering, pp. 1–12
- Rezania M, Faramarzi A, Javadi AA (2011) An evolutionary based approach for assessment of earthquake-induced soil liquefaction and lateral displacement. *Eng Appl Artif Intell* 24(1):142–153
- Zhang J, Zhang LM, Huang HW (2013) Evaluation of generalized linear models for soil liquefaction probability prediction. *Environ Earth Sci* 68(7):1925–1933
- Kohestani V, Hassanlourad M, Ardakani A (2015) Evaluation of liquefaction potential based on CPT data using random forest. *Nat Hazards* 79(2):1079–1089
- Hoang N-D, Bui DT (2018) Predicting earthquake-induced soil liquefaction based on a hybridization of kernel Fisher discriminant analysis and a least squares support vector machine: a multi-dataset study. *Bull Eng Geol Env* 77(1):191–204
- Pirhadi N, Tang X, Yang Q, Kang F (2019) A new equation to evaluate liquefaction triggering using the response surface method and parametric sensitivity analysis. *Sustainability* 11(1):112
- Zhou J, Li E, Wang M, Chen X, Shi X, Jiang L (2019) Feasibility of stochastic gradient boosting approach for evaluating seismic liquefaction potential based on SPT and CPT case histories. *J Perform Constr Facil* 33(3):04019024
- Cai M, Hocine O, Mohammed AS, Chen X, Amar MN, Hasanihan M (2021) Integrating the LSSVM and RBFNN models with three optimization algorithms to predict the soil liquefaction potential. *Eng Comput*, 1–13
- Zhao Z, Duan W, Cai G (2021) A novel PSO-KELM based soil liquefaction potential evaluation system using CPT and Vs measurements. *Soil Dynam Earthq Eng* 150:106930
- Wang L, Wu C, Tang L, Zhang W, Lacasse S, Liu H, Gao L (2020) Efficient reliability analysis of earth dam slope stability using extreme gradient boosting method. *Acta Geotech* 15(11):3135–3150
- Wang M-X, Huang D, Wang G, Li D-Q (2020) SS-XGBoost: a machine learning framework for predicting newmark sliding displacements of slopes. *J Geotech Geoenviron Eng* 146(9):04020074
- Bharti JP, Mishra P, Sathishkumar V, Cho Y, Samui P (2021) Slope stability analysis using Rf, Gbm, Cart, Bt and Xgboost. *Geotech Geol Eng* 39(5):3741–3752
- Zhang W, Wu C, Zhong H, Li Y, Wang L (2021) Prediction of undrained shear strength using extreme gradient boosting and random forest based on Bayesian optimization. *Geosci Front* 12(1):469–477
- Polikar R (2012) Ensemble learning. *Ensemble machine learning*. Springer, pp. 1–34
- Worasuchep C (2021) Ensemble classifier for stock trading recommendation. *Appl Artif Intell*, 1–32
- Breiman L (1996) Bagging predictors. *Mach Learn* 24(2):123–140
- Quinlan JR (1996) Bagging, boosting, and C4. 5. *Aaai/iaai* 1:725–730

42. Rocca J (2019) Ensemble methods: bagging, boosting and stacking. medium-towards data science. <https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205>
43. Papadopoulos S, Azar E, Woon W-L, Kontokosta CE (2018) Evaluation of tree-based ensemble learning algorithms for building energy performance estimation. *J Build Perform Simul* 11(3):322–332
44. Bou-hamad I, Larocque D, Ben-Ameur H, Mâsse LC, Vitaro F, Tremblay RE (2009) Discrete-time survival trees. *Can J Stat* 37(1):17–32
45. Sabbeh SF (2018) Machine-learning techniques for customer retention: a comparative study. *Int J Adv Comput Sci Appl*, 9(2). <https://doi.org/10.14569/IJACSA.2018.090238>
46. Qi Y, Bar-Joseph Z, Klein-Seetharaman J (2006) Evaluation of different biological data and computational classification methods for use in protein interaction prediction. *Proteins* 63(3):490–500
47. Musbah H, Aly HH, Little TA (2021) Energy management of hybrid energy system sources based on machine learning classification algorithms. *Electric Power Syst Res* 199:107436
48. Muhammad L, Islam MM, Usman SS, Ayon SI (2020) Predictive data mining models for novel coronavirus (COVID-19) infected patients' recovery. *SN Comp Sci* 1(4):1–7
49. Pham BT, Nguyen MD, Nguyen-Thoi T, Ho LS, Koopialipour M, Quoc NK, Armaghani DJ, Van Le H (2021) A novel approach for classification of soils based on laboratory tests using Adaboost. *Tree ANN Model Transp Geotech* 27:100508
50. Wang X, Li Z, Shafieezadeh A (2021) Seismic response prediction and variable importance analysis of extended pile-shaft-supported bridges against lateral spreading: exploring optimized machine learning models. *Eng Struct* 236:112142
51. Chen Z, Li H, Goh ATC, Wu C, Zhang W (2020) Soil liquefaction assessment using soft computing approaches based on capacity energy concept. *Geosciences* 10(9):330
52. Guyon I, Gunn S, Nikravesh M, Zadeh LA (2008) Feature extraction: foundations and applications. Springer, Berlin
53. Zheng A, Casari A (2018) Feature engineering for machine learning: principles and techniques for data scientists. O'Reilly Media Inc, Sebastopol
54. Das SK, Mohanty R, Mohanty M, Mahamaya M (2020) Multi-objective feature selection (MOFS) algorithms for prediction of liquefaction susceptibility of soil based on in situ test methods. *Nat Hazards* 103:2371–2393
55. Kuhn M, Johnson K (2019) Feature engineering and selection: A practical approach for predictive models. CRC Press, Boca Raton
56. Hu J (2021) Data cleaning and feature selection for gravelly soil liquefaction. *Soil Dynam Earthq Eng* 145:106711
57. Demir S, Sahin EK (2021) Assessment of feature selection for liquefaction prediction based on recursive feature elimination. *Eur J Sci Tech* 28:290–294
58. Team RDC (2020) R: A language and environment for statistical computing. R Foundation for Statistical Computing. <https://www.r-project.org>.
59. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evolut Comput* 1(1):3–18
60. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *J Comput Syst Sci* 55(1):119–139
61. Hastie T, Rosset S, Zhu J, Zou H (2009) Multi-class adaboost. *Stat Interface* 2(3):349–360
62. An T-K, Kim M-H (2010) A new diverse AdaBoost classifier. In: 2010 International conference on artificial intelligence and computational intelligence. IEEE, pp 359–363
63. Natekin A, Knoll A (2013) Gradient boosting machines, a tutorial. *Front Neurorobot* 7:21
64. Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785–794
65. Qin C, Zhang Y, Bao F, Zhang C, Liu P, Liu P (2021) XGBoost optimized by adaptive particle swarm optimization for credit scoring. *Math Probl Eng*. <https://doi.org/10.1155/2021/6655510>
66. XGBoost-Documentation (2021). <https://xgboost.readthedocs.io/en/stable/>. Accessed 16 Sept 2021
67. Zhang H, Qiu D, Wu R, Deng Y, Ji D, Li T (2019) Novel framework for image attribute annotation with gene selection XGBoost algorithm and relative attribute model. *Appl Soft Comput* 80:57–79
68. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. *J Mach Learn Res* 3:1157–1182
69. Guyon I, Weston J, Barnhill S, Vapnik V (2002) Gene selection for cancer classification using support vector machines. *Mach Learn* 46(1):389–422
70. Shi F, Peng X, Liu Z, Li E, Hu Y (2020) A data-driven approach for pipe deformation prediction based on soil properties and weather conditions. *Sustain Cities Soc* 55:102012
71. Sun D, Shi S, Wen H, Xu J, Zhou X, Wu J (2021) A hybrid optimization method of factor screening predicated on GeoDetector and random forest for landslide susceptibility mapping. *Geomorphology* 379:107623
72. Svetnik V, Liaw A, Tong C, Wang T (2004) Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. In: International workshop on multiple classifier systems. Springer, pp 334–343
73. Paja W, Pancercz K, Grochowalski P (2018) Generational feature elimination and some other ranking feature selection methods. *Advances in feature selection for data and pattern recognition*. Springer, pp. 97–112
74. Kursa MB, Rudnicki WR (2010) Feature selection with the Boruta package. *J Stat Softw* 36(11):1–13
75. Stańczyk U, Zielosko B, Jain LC (2018) Advances in feature selection for data and pattern recognition: an introduction. *Advances in feature selection for data and pattern recognition*. Springer, pp 1–9
76. Breaux HJ (1967) On stepwise multiple linear regression. Report no. 1369. Ballistic research laboratories aberdeen proving ground, Maryland
77. Kumar S, Attri S, Singh K (2019) Comparison of Lasso and stepwise regression technique for wheat yield prediction. *J Agrometeorol* 21(2):188–192
78. Chowdhury MZI, Turin TC (2020) Variable selection strategies and its importance in clinical prediction modelling. *Fam Med Commun Health* 8(1):e000262. <https://doi.org/10.1136/fmch-2019-000262>
79. Huang C, Townshend J (2003) A stepwise regression tree for nonlinear approximation: applications to estimating subpixel land cover. *Int J Remote Sens* 24(1):75–90
80. Huang C, Davis L, Townshend J (2002) An assessment of support vector machines for land cover classification. *Int J Remote Sens* 23(4):725–749
81. Maxwell AE, Warner TA, Fang F (2018) Implementation of machine-learning classification in remote sensing: an applied review. *Int J Remote Sens* 39(9):2784–2817
82. Etikan I, Bala K (2017) Sampling and sampling methods. *Biom Biostat Int J* 5(6):00149
83. Berndt AE (2020) Sampling methods. *J Hum Lact* 36(2):224–226
84. Fink A (2003) How to sample in surveys. Sage, Thousand Oaks

85. Samui P, Sitharam T (2011) Machine learning modelling for predicting soil liquefaction susceptibility. *Nat Hazards Earth Syst Sci* 11(1):1–9
86. Demir S, Sahin EK (2022) Comparison of tree-based machine learning algorithms for predicting liquefaction potential using canonical correlation forest, rotation forest, and random forest based on CPT data. *Soil Dynam Earth Eng* 154:107130
87. Ao S-I (2008) *Data mining and applications in genomics*. Springer Science & Business Media, Berlin
88. Sahin EK (2022) Comparative analysis of gradient boosting algorithms for landslide susceptibility mapping. *Geocarto Int* 37(9):2441–2465. <https://doi.org/10.1080/10106049.2020.1831623>
89. Kuhn M (2008) Building predictive models in R using the caret package. *J Stat Softw* 28(1):1–26
90. Keyport RN, Oommen T, Martha TR, Sajinkumar K, Gierke JS (2018) A comparative analysis of pixel-and object-based detection of landslides from very high-resolution images. *Int J App Earth Obs Geoinf* 64:1–11

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.