Original Research Article

Exploration of the Stacking Ensemble Machine Learning Algorithm for Cheating Detection in Large-Scale Assessment

Educational and Psychological Measurement
2023, Vol. 83(4) 831–854

DOI: 10.1177/00131644221117193
journals.sagepub.com/home/epm

Todd Zhou[1] and Hong Jiao[2]

Abstract

Cheating detection in large-scale assessment received considerable attention in the extant literature. However, none of the previous studies in this line of research investigated the stacking ensemble machine learning algorithm for cheating detection. Furthermore, no study addressed the issue of class imbalance using resampling. This study explored the application of the stacking ensemble machine learning algorithm to analyze the item response, response time, and augmented data of test-takers to detect cheating behaviors. The performance of the stacking method was compared with that of two other ensemble methods (bagging and boosting) as well as six base non-ensemble machine learning algorithms. Issues related to class imbalance and input features were addressed. The study results indicated that stacking, resampling, and feature sets including augmented summary data generally performed better than its counterparts in cheating detection. Compared with other competing machine learning algorithms investigated in this study, the meta-model from stacking using discriminant analysis based on the top two base models—Gradient Boosting and Random Forest—generally performed the best when item responses and the augmented summary statistics were used as the input features with an under-sampling ratio of 10:1 among all the study conditions.

[1]Winston Churchill High School, Potomac, MD, USA
[2]University of Maryland, College Park, USA

Corresponding Author:
Hong Jiao, Measurement, Statistics and Evaluation, Department of Human Development and Quantitative Methodology, University of Maryland, 1230C Benjamin Building, College Park, MD 20742, USA.
Email: hjiao@umd.edu

Keywords

cheating detection, stacking, machine learning, ensemble learning algorithms, response time, resampling, oversampling, SMOTE, under-sampling, dual resampling, data augmentation

Test fairness is one important validity issue in test score use and interpretation in large-scale assessment. When some test-takers take advantage of external assistance to spuriously gain scores in a test, it is unfair to the majority of test-takers who have no access to any external assistance. In general, such cheating behaviors lead to test security concerns and jeopardize the validity of test score interpretation.

Cheating could take on different forms. External assistance could occur before, during, or after test administration. Before a test, a test-taker could have pre-knowledge of the items due to illegal access to test items. During test administration, cheating may occur as copying from others, using prohibited notes, asking for help from other people, searching from the internet, or using a proxy test-taker. After test administration, answer changes may occur. Some proctors such as classroom teachers may change students' answers to help them to gain scores. Different methods and technology have been explored to analyze the examinees' behaviors for cheating detection.

In psychometric research, item-level data collected during test administration has often been used for cheating detection. The methods used for cheating detection in large-scale assessment fall into different categories such as person fit index-based, latent variable model-based, and machine learning methods. Both item product and process data could be used for cheating detection. Person fit statistics based on item responses (e.g., Drasgow et al., 1985; Meijer & Sijtsma, 2001; Sijtsma & Meijer, 1992; van der Flier, 1982) and response time (Man et al., 2019) have been developed to detect cheating in large-scale assessments. Furthermore, indices based on answer changes (Bishop & Egan, 2017) and the relation between the responses and the response times (Toton & Maynes, 2019) were proposed for cheating detection as well. In addition, different latent variable models based on item responses and response time have been explored to detect cheating (e.g., Chen et al., 2020; Shu et al., 2013; Skorupski et al., 2016; Toton & Maynes, 2019; Wang et al., 2018; Wollack, 1997). Recently, researchers (Man & Harring, 2020) assessed item pre-knowledge via multiple-group joint modeling of item responses, response time, and visual fixation counts. It is expected that joint modeling of multiple related data types leads to higher accuracy in cheating detection.

In recent years, machine learning algorithms have been explored by different researchers for cheating detection. Zopluoglu (2019) studied Extreme Gradient Boosting to detect item pre-knowledge. Man et al. (2019) explored both supervised (K-nearest mean, random forest, and support vector machine [SVM]) and unsupervised (K-means and self-organizing mapping) machine learning algorithms for fraud detection. However, none of these studies investigated the impact of class imbalance

in the cheater and non-cheater groups on cheating detection. No previous studies in cheating detection investigated the stacking ensemble learning method that combines several base machine learning algorithms into one ensemble learning model to improve model performance for classification. Thus, this current study investigates stacking, one of the ensemble machine learning methods, and compares its performance with other ensemble methods and base models in cheating detection when the classes are rebalanced via resampling.

Machine Learning Methods for Cheating Detection

Machine learning methods have been used widely and successfully in a variety of fields. For cheating detection, machine learning turns out to be a powerful tool to recognize patterns and find the similarities of answers due to plagiarism. This section introduces the machine learning algorithms used in this study.

Common Machine Learning Methods

Decision Tree algorithm is the footstone of many other high-level machine learning algorithms. It reflects how different attributes affect the results and process different types of data at the same time. However, the decisions are made at each node locally and it lacks global optimality. SVM computes multidimensional data points to find a hyperplane, which acts as the decision boundary to classify the samples. Logistic regression is a supervised learning algorithm used for classification based on its predictive probability. It uses a cost function, such as the Sigmoid function, to predict the probability of a target variable and makes classification based on a threshold value. Naive Bayes is a generative model, which constructs a probabilistic union model on the problem utilizing the multiplication rules of probability. Linear discriminant analysis classifies multidimensional data using probabilities to estimate whether a new set of inputs belongs to a separate class and make prediction based on the highest attained probability value. Neural network adopts a hierarchical structure with layers connecting the adjacent-layer nodes but not the same-layer and cross-layer nodes. It uses the back propagation method to adjust parameters and applies the iterative algorithm to train the entire network.

Ensemble Learning Methods

Ensemble learning is engineered to combine multiple model results to develop a meta-model to achieve higher prediction accuracy. There are three main ensemble learning methods: bagging, boosting, and stacking. Standing for Bootstrap Aggregating, Bagging is a parallel ensemble method that is designed for reducing the variance of the prediction model of decision trees. The data used for bagging is a bootstrapped sample from the whole dataset. A decision tree is developed based on each sub-sample. The results from the multiple decision trees are aggregated to find

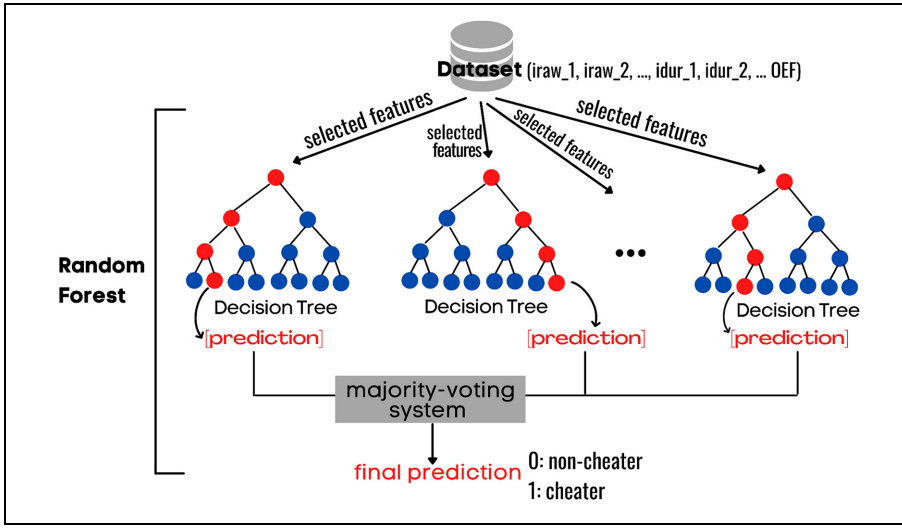

Figure 1. Bagging Structure.

the best predicted results. The final output is selected by the majority-voting or averaging system, as shown in Figure 1. Random Forest is an improvement over the bagging algorithm. Random Forest aggregates the results of multiple decision trees from subsets of features while searching for the best feature among its randomly selected subset of features. Random Forest, therefore, yields better results than its sub-models: Decision Trees.

Boosting is a technique to promote the weak learning algorithm to the strong learning algorithm sequentially following a deterministic strategy. Gradient Boosting is a boosting ensemble learning method. As illustrated in Figure 2, during the training process, each sub-model depends on the previous one in an adaptive way. Based on the gradient descent optimization process, Gradient Boosting attempts to identify the shortcomings of weak learners and minimize the loss function of the local sub-model gradually during continuous model iteration to enhance the overall effectiveness. Gradient Boosting assembles decision tree sub-models by adding them sequentially and correcting the prior sub-model's performance. In a gradient descent approach, each of its sub-models attempts to enhance the overall effectiveness through the continuous model iterations.

Stacking is a different ensemble technique of different base machine learning models. Bagging and boosting combine the sub-models of the same type (homogeneous weak learners), while stacking uses different base learning algorithms (heterogeneous weak learners). After fitting different decision trees to the bootstrapped subsamples of the same dataset, bagging averages or majority votes the prediction results from each sub-sample. Boosting develops an ensemble learning algorithm by training the model on the dataset with wrong predictions by correcting the prediction errors

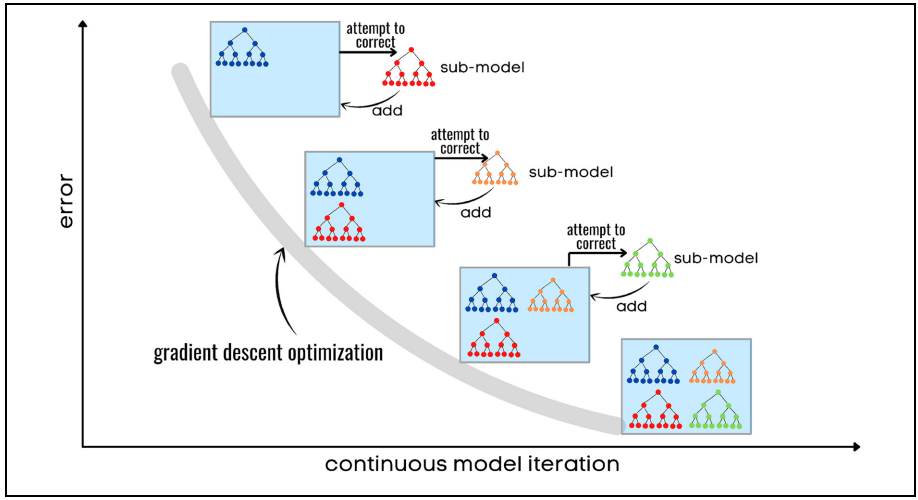

Figure 2. Boosting Structure.

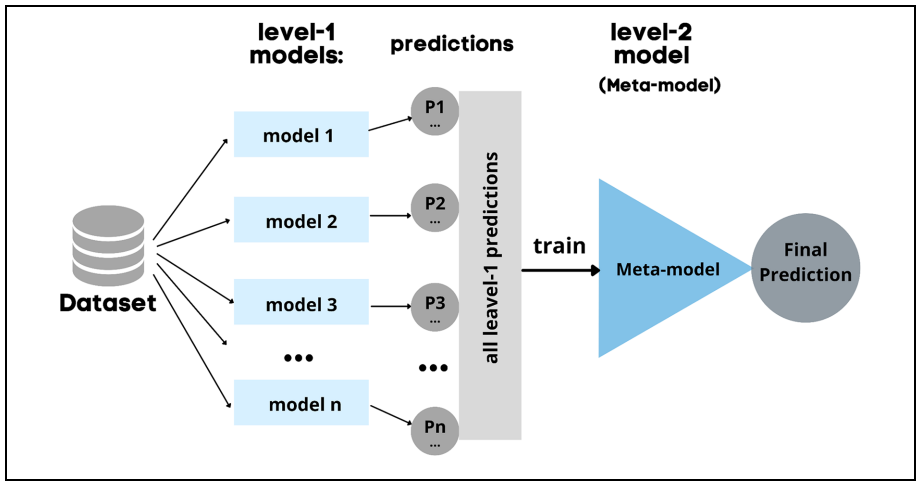

Figure 3. Stacking Structure.

and outputs a weighted average of all the models. Stacking, also called Stacked Generalization, has a 2-level structure as shown in Figure 3, where the level-2 model learns from the outputs of different level-1 models. The first level, containing multiple models called base models, trains each base model separately using the same dataset and makes the predictions. The second level has one model, called the meta-

model, which is a learning model to combine all the prediction outputs from each of the level-1 base models. Based on those predictions, a machine learning algorithm such as SVM or logistic regression ideal for ensemble (Pavlyshenko, 2018) is used to develop a meta-model to make final predictions. Chan and Stolfo (1997) demonstrated that the meta-model improved the prediction accuracy of a single classifier. The benefit of stacking is that it combines the outputs of several good-performing models to make predictions that are expected to outperform a single model in the ensemble. In general, stacking is a competitive ensemble strategy by integrating model predictions from different base models in a more optimal way to produce a new set of predictions. Ultimately, the final model is stacked on the top of the contributing base models to improve the overall prediction.

Resampling for Class Imbalance in Cheating Detection

In cheating detection, only a very small portion of test-takers may engage in cheating. This leads to imbalanced classes in machine learning with cheaters in the minority group while the non-cheaters in the majority group. Most machine learning algorithms are developed for equally balanced classes (He & Garcia, 2009). When the classes are imbalanced, machine learning algorithms tend to produce biased results favoring the majority class, leading to low prediction accuracy in the minority class. Guo et al. (2008) noted that the lack of representation and information of the key characteristic of the minority class makes machine learning difficult to predict the probability of the minority class. The skewed class distribution of the cheater and the non-cheater classes in cheating detection makes the prediction models biased toward the majority class, yielding high accuracy for predicting the non-cheaters but low accuracy in predicting the cheaters.

Resampling is a method to tackle the class imbalance issue in machine learning. It balances the minority and the majority classes by adding more cases in the minority class or reducing the cases in the majority class. The resampling methods include oversampling of the minority class, under-sampling of the majority class, and the concurrent use of both. Japkowicz (2000) compared different methods in solving the imbalanced class problem and found that oversampling the minority class and under-sampling the majority class are both very successful approaches to balance the imbalanced classes. Among the oversampling methods, the Synthetic Minority Oversampling Technique (SMOTE) proposed by Chawla et al. (2002) is an effective oversampling method by generating synthetic cases in the minority class using the K-nearest neighbor algorithm. First, a case from the minority class is randomly selected, its nearest neighbor is found, the difference between the data point and its nearest neighbor is obtained, finally the difference is multiplied by a random number R, where $0 < R < 1$, then the synthetic data is added to the feature vector. These steps are repeated until the targeted balanced level is achieved. This strategy effectively broadens the minority class's decision-making region. Therefore, by generating synthetic data points at the feature level, SMOTE is more advanced than the

random over-sampling. The SMOTE function allows the selection of different ratios of the minority to the majority classes, indicating the sample size of the minority class after oversampling. For example, a value of 0.4 set for SMOTE means the ratio of the sample size of the oversampled minority class to that of the un-resampled majority class is 0.4 to 1 using the majority class as the reference.

Under-sampling the majority class is an opposite strategy for balancing classes by randomly selecting cases in the majority class to remove. Using the *RandomUnderSampler* function in Python, the number of cases in the majority class will be reduced to attain more balanced class sizes. The *RandomUnderSampler* function allows the selection of a targeted class ratio. For example, a value of 0.5 set for the *RandomUnderSampler* function means the sample size of the un-resampled minority class to that of the under-sampled majority class is 1 to 2 using the minority class as the reference.

Chawla et al. (2002) explored combining oversampling of the minority class via SMOTE with random under-sampling of the majority class. It turned out that the concurrent application of SMOTE and random under-sampling outperformed either under-sampling or oversampling in all study conditions. Essentially, the imbalanced data issue was resolved by simultaneously oversampling the minority class and under-sampling the majority class to provide balanced data for the machine learning models. By adjusting the ratios in the SMOTE and *RandomUnderSampler* functions simultaneously, the ratio of the majority to the minority classes was balanced to the midway. A note worthy of attention is that the resampling procedure should be applied to the training set for the machine learning model development and the evaluation of the model performance should be conducted on the testing dataset without resampling.

Method

The purpose of the study was to investigate the stacking algorithm for cheating detection in large-scale assessments with consideration of class imbalance. The performance of the proposed stacking models was compared with those of the competing machine learning models for cheating detection in study conditions with different resampling methods and different input features.

Data

The test data used in this study is from a large-scale licensure test with cheating cases flagged. This dataset consists of two test forms, made publicly available by Cizek and Wollack (2017). Each test form consists of 170 dichotomously scored operational items. In addition, each test-taker responded to 10 field-test items which were excluded in the analyses. The dataset consists of item responses, response time, and the number of attempts of the test for 1,636 test-takers with 46 flagged as likely cheaters. No information was given about the types of cheating behaviors each flagged

test-taker committed. As only 46 out of 1,636 examinees were flagged, the cheater class has a very low proportion, 2.81% of the total sample. This dataset is very imbalanced. In the raw item responses showing the option chosen by each test-taker on each item, there are 141 ''NA'' values indicating missing data. However, the scored item responses contain no missing, implying missing values have been coded as 0s.

Zopluoglu (2019) and Man et al. (2019) explored cheating detection using different machine learning methods, both using the same data from Cizek and Wollack (2017). However, none of them considered the issue of class imbalance using resampling. This study explored different resampling approaches to balancing the classes: oversampling, under-sampling, and both labeled as dual resampling. More specifically, the SMOTE and *RandomUnderSampler* functions in Python were utilized for oversampling and under-sampling, respectively. For the base model comparison, different ratios of the non-cheater versus the cheater classes were explored including 1:1, 2:1, 5:1, and 10:1.

Feature Selection

The input features include item response scores, item response time, and augmented summary data which were used to train the machine learning models. The augmented summary data labeled as Other Effective Features (OEF) for every test-taker include the number of attempts of taking the test and the summary statistics such as the total test scores and the total response time spent on all 170 items, as well as the mean, the median, the maximum, and the minimum item response time across 170 items for every test-taker. The inclusion of the OEF is a type of data augmentation which is expected to provide more information at a higher level in addition to the information from each specific item. In total, there are 347 variables.

In this study, different input data sets were explored. The six sets of input features include Item Response Score (iraw) only, Response Time (idur) only, Item Response Score plus Response Time (iraw_idur), Item Response Score plus Other Effective Features (iraw_OEF), Response Time plus Other Effective Features (idur_OEF), and Item Response Score plus Response Time plus Other Effective Features (iraw_idur_OEF).

Base Models

To develop a meta-model using stacking ensemble learning, this study investigated six basic machine learning models: Decision Tree, SVM, Logistic Regression, Naïve Bayes, Discriminant Analysis, and Neural Network as well as two ensemble learning models: Random Forest and Gradient Boosting. The outputs from these eight base models were used for developing a meta-model.

Given different resampling methods and ratios, different feature sets, and base models, this study explored the optimal designs among 624 study conditions at the level 1 for stacking for cheating detection. The non-resampling condition is not fully

crossed with the ratios of resampling. Therefore, the 624 study conditions consist of 576 conditions (8 base models × 6 feature sets × 3 resampling methods × 4 resampling ratios) with fully crossed designs for resampling and 48 conditions (8 base models × 6 feature sets) of non-resampling.

Stacking

This study developed a stacking ensemble learning model for cheating detection. The stacking learning model was built based on six basic machine learning models, one bagging model, and one boosting model. All of these models including the meta-model were developed using the Scikit-learn (sklearn) Machine Learning library in Python (Pedregosa et al., 2011), and the structure of stacking learning was implemented using MLxtend, a machine learning extension Python library (Raschka, 2018). Interested readers can refer to the online supplementary document for the code snippets.

Stacking model development contains two phases, base model development and stacking construction. The flowchart for stacking is graphically presented in Figure 4. Phase 1 is for examining the performance of the base models. Each of the six basic machine learning models and the bagging (Random Forest) and boosting (Gradient Boosting) ensemble models was used to analyze the data, respectively. This stage focuses on feature selection, Hyperparameter Tuning, and model performance evaluation of the eight base models. Each of the eight machine learning methods was fitted to each of the six sets of input features separately using *k*-fold cross-validation with different resampling methods. The hyper-parameter tuning was performed to optimize each model. The feature selection was based on the evaluation criteria. A list of the top classifiers was identified. The base classifiers with the best data features and the optimal set of hyper-parameters of each classifier were rank-ordered.

Phase 2 is for constructing the stacking model. Stacking ensemble learning is engineered in a way to develop a meta-model to make final predictions based on the base classifiers' predictions to improve overall performance. With the best data features and the optimal hyper-parameters of each model obtained from Phase 1, each set of the top classifiers was trained. To select a set of base models, different numbers of top rank-ordered models from Phase 1 were combined. Thus, seven sets of base models were obtained as set 1 consisting of top two base classifiers in Phase 1, set 2 consisting of top three base classifiers, and so on. A meta-model was developed stacking the detection results from each base set under *k*-fold cross-validation, and their final prediction results were evaluated. Specifically, the green text under Phase 2 in Figure 4 indicates the whole prediction results of each classifier, p_n, compiled within each of the four random folds. Then, all the prediction results from each base classifier were put together, shown as the light gray upright rectangle in the flowchart. In the end, the full list of the prediction results with the labels was the input data to train the meta-classifier for the final predictions.

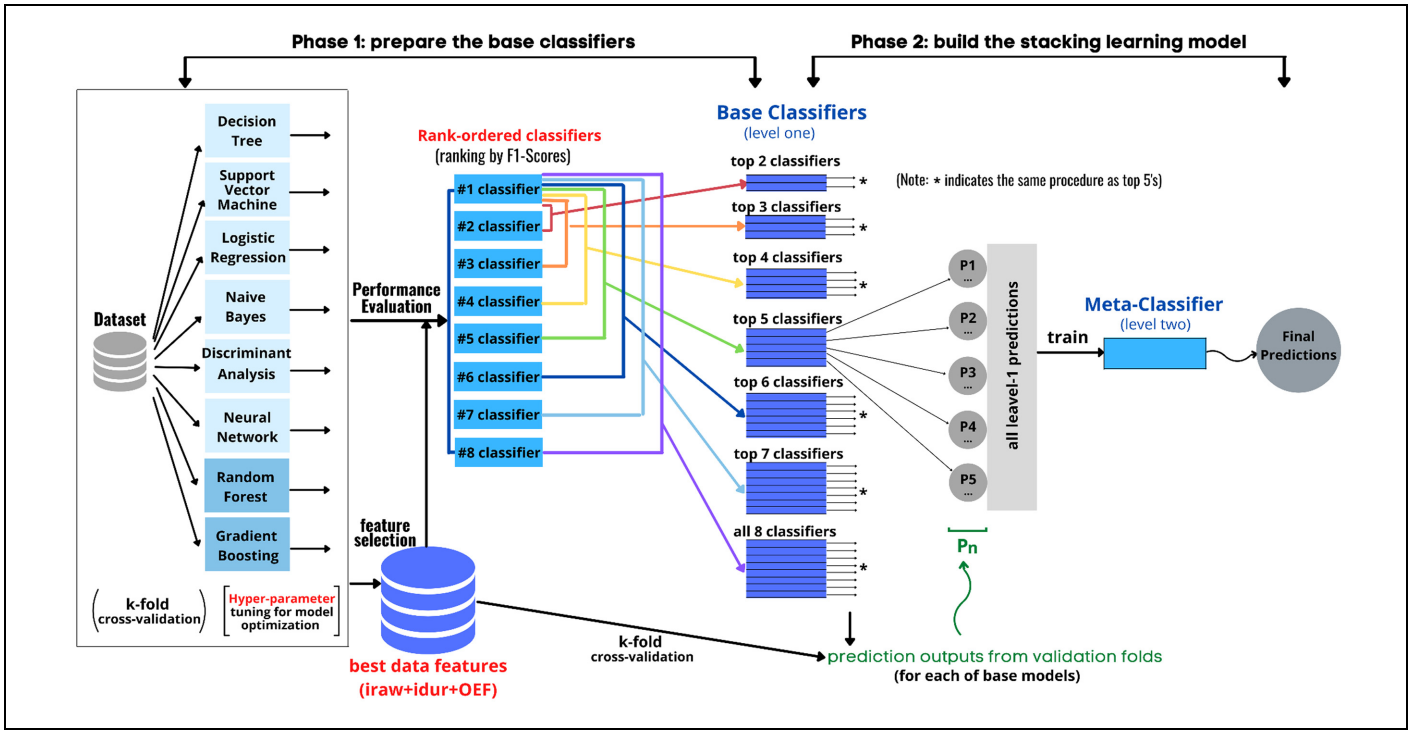

Figure 4. The Constructed Stacking Ensemble Learning Model.

K-Fold Cross-Validation

For supervised machine learning, we need to split the data into a training set (for model training) and a test set (for model evaluation). Zopluoglu (2019) split the dataset to 80% for training and 20% for test. This study applied the split of 75% training versus 25% test on *K*-Fold Cross-Validation. *K*-Fold Cross-Validation provides train/test indices to randomly split the dataset into k exclusive folds and each fold is then used as a test set once while the $k-1$ remaining folds are used as the training set. The split is done iteratively for k times (Anguita et al., 2012). This technique uses multiple exclusive data splits for training and test to reduce bias. Four-fold Cross-Validation ($K = 4$) was used in all analyses. The resampling procedure was applied on the training set in each fold of cross-validation, while the evaluation results were generated based on the test set in each fold which was part of the original data without being resampled.

Hyper-Parameter Tuning

Hyper-parameters control the model's behaviors and allow the model to find the parameters that would yield the best performance. Conducting hyper-parameter tuning is an important step to improve the model performance as it finds the best combination of hyper-parameters which minimizes the loss function and produces the best results. The TPOTClassifier in TPOT API (Le et al., 2020) performs an intelligent search over the hyper-parameters in machine learning pipelines with the customized parameters. For model optimization, TPOTClassifier with various parameters were applied to find the optimal hyper-parameters on each of the models in this study.

Evaluation Criteria

In cheating detection with extremely imbalanced classes, overall accuracy defined in terms of the percentage of correct classification is not a proper index to quantify the classification accuracy due to the much larger class size in the majority non-cheater group. Instead, Precision, Recall, F1 Score, and the false-positive rate (FPR) were used as the evaluation criteria for model performances. All these indices were computed based on the confusion matrix (refer to https://en.wikipedia.org/wiki/Confusion_matrix for details), showing correct classifications of true positive and true negative and misclassifications of false positive and false negative. Recall (sensitivity, power, or true positive rate) is the metric that evaluates a model's ability to predict true positives (cheaters) out of the total actual positives, indicating the proportion of *actual* cheaters which are predicted as cheaters, computed as in Equation 1.

$$Recall = \frac{True\ Positive}{TruePositive + FalseNegative}. \quad (1)$$

Precision computes the proportion of actual cheaters out of the *predicted* cheaters. It measures how a model performs at detecting true cheaters out of those predicted as cheaters. It is computed as shown in Equation 2.

$$Precision = \frac{True\ Positive}{TruePositive + FalsePositive}. \quad (2)$$

To seek the balance of both Recall and Precision, F1-scores, the harmonic mean of Recall and Precision, is suggested as a measure for imbalanced class sizes by Forman and Scholz (2010). According to Sasaki (2007), the F1 score is computed as follows in Equation 3.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TruePositive}{2 * TruePositive + FalsePositive + FalseNegative}. \quad (3)$$

The FPR indicates the percentage of the non-cheaters who are incorrectly classified as cheaters. Cheating has severe validity, moral, ethical, and legal implications in any high-stakes testing program. To minimize such concerns, it requires extreme caution in flagging a test-taker as a cheater. The FPR measures the degree to which the non-cheaters are misclassified as cheaters which helps to address such concerns. It is computed as in Equation 4.

$$FPR = \frac{False\ Positive}{False\ Positive + True\ Negative}. \quad (4)$$

Results

This section summarizes the major results of the study due to space limit. It includes the results for resampling, input feature selection, base model performance, and the meta-model performance in terms of the evaluation criteria. It is noted that model performance was evaluated based on the non-resampled test dataset.

Resampling for Class Imbalance

As noted, resampling was only applied to the training dataset in each fold. The performances of oversampling, under-sampling, and dual resampling are compared with no resampling in terms of four evaluation criteria. Six sets of input features were investigated in this study. For resampling, different ratios were explored including 1:1, 2:1, 5:1, and 10:1 for the non-cheater versus cheater classes. To achieve the 1:1 ratio, the SMOTE oversampling ratio was set at 0.4 while the under-sampling ratio was 1. Table 1 presents some examples of the sample sizes for the non-cheater and cheater classes for one fold for the training and test datasets, respectively. As expected, under-sampling led to the smallest sample sizes while oversampling led to the largest sample sizes. In general, the ratios of 2:1 between the non-cheater and the cheater

Table 1. Sample Sizes for the Non-Cheater and Cheater Classes for Different Resampling Methods Based on One-Fold Cross-Validation Sample.

Ratios		No resampling	Under-sampling	Oversampling	Dual resampling
1:1	Training	1,189:38	38:38	1,189:1,189	475:475
	Test	401:8	401:8	401:8	401:8
2:1	Training	1,189:38	76:38	1,189:594	950:475
	Test	401:8	401:8	401:8	401:8
5:1	Training	1,189:38	190:38	1,189:237	590:118
	Test	401:8	401:8	401:8	401:8
10:1	Training	1,189:38	380:38	1,189:118	590:59
	Test	401:8	401:8	401:8	401:8

classes led to the largest total sample sizes for both oversampling and dual resampling, even larger than the original dataset without resampling. In general, sample sizes in different classes differed under different resampling schemes.

To demonstrate the impact of resampling on the evaluation criteria, the model performances with different resampling methods based on all input features (i.e., item responses, response time, and summary statistics) with a ratio of 5:1 are presented in Figure 5 for illustration. Interested readers can find summary figures for resampling ratios of 1:1, 2:1, and 10:1 in the online supplementary document in Figures S1 to S3. Each figure contains information about Recall, Precision, F1 scores, and the FP rate for each model under each resampling method compared with that based on non-resampling. For resampling with a ratio of 1:1, almost all resampling regardless of models led to higher Recall than no resampling with under-sampling and oversampling performing better than dual resampling. Discriminant Analysis led to the highest Recall. However, in terms of Precision, only dual resampling performed about the same as the original data or slightly better than no resampling except Decision Tree. Random Forest produced the highest Precision, Gradient Boosting the second best while other models fell far below. Similar patterns were observed for F1 scores with Gradient Boosting performed the best. Gradient Boosting and Random Forest with dual resampling produced higher F1 scores than no resampling and other models. On the contrary, under-sampling and oversampling led to higher FP rates than no resampling while the lowest values were found in Random Forest.

For the study condition with a resampling ratio of 2:1, under-sampling led to higher Recall than other resampling or non-resampling for all models. Discriminant Analysis led to the highest Recall. For Precision, oversampling and dual resampling performed better than or about the same as non-resampling for all models except Decision Tree with non-resampling produced the highest Precision. Among all models, Random Forest with over-sampling produced the highest Precision, followed by Gradient Boosting, but all other models fell far below. The differences in F1 scores were not large for majority of the models except Decision Tree and Gradient Boosting which had lower values with under-sampling. On the contrary, under-

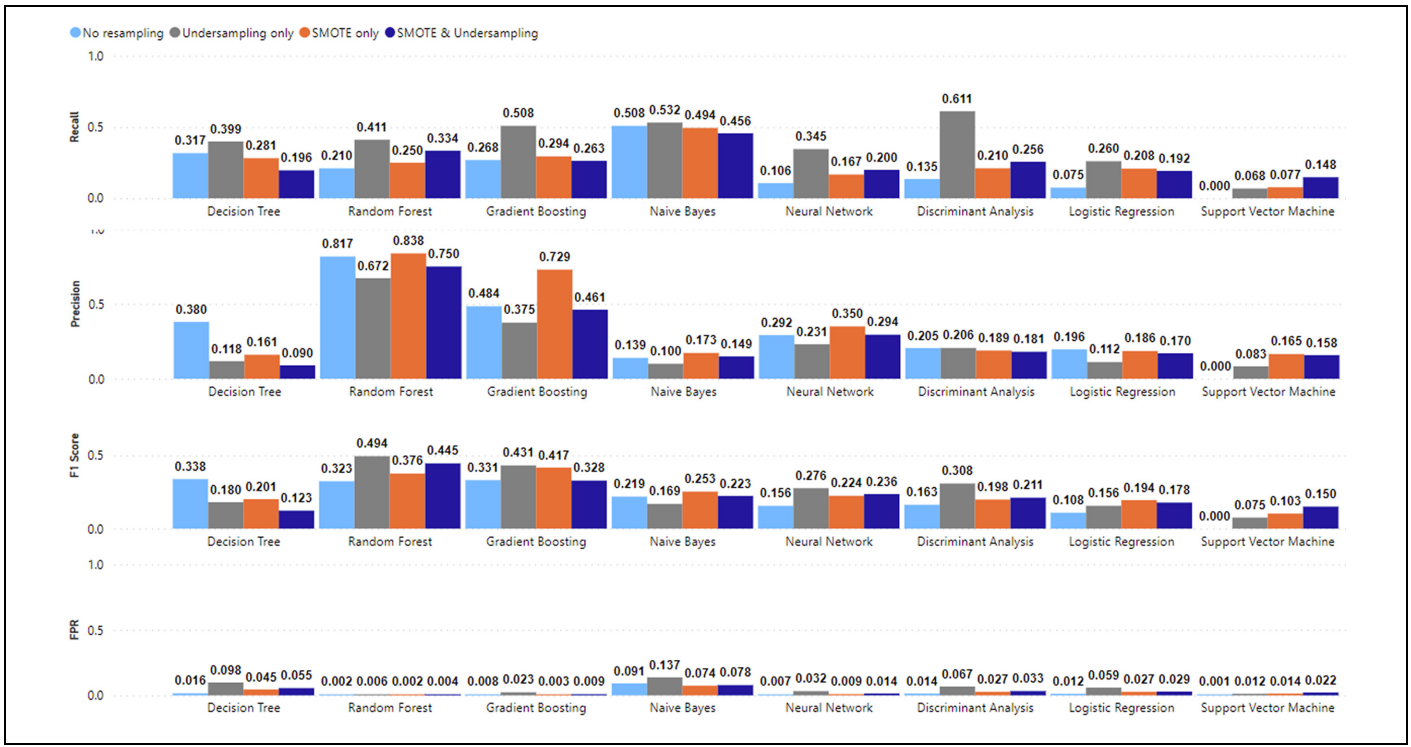

Figure 5. Model Performance Comparison With and Without Resampling Based on Item Responses, Response Time, and Summative Statistics With a Ratio of 5:1 Between the Non-Cheater and Cheater Classes.

sampling led to the highest FP rates with the lowest two values found in Random Forest and Gradient Boosting.

As seen in Figure 5 for a resampling ratio of 5:1 for the non-cheater and cheater class rebalancing, under-sampling led to the highest Recall than any other resampling or non-resampling for all models except SVM. Among all the models, Discriminant Analysis with under-sampling led to the highest Recall. There was not much difference in Recall among the non-resampling and the resampling methods for Naïve Bayes, which was the second best performer. For Precision, oversampling performed better than dual resampling which performed about the same as non-resampling for all other models except Decision Tree. Random Forest produced the highest Precision with over-sampling, followed by Gradient Boosting, but all other models fell far below. For F1 scores, Random Forest with under-sampling and dual resampling produced higher scores than over-sampling or non-resampling and other models. On the contrary, under-sampling led to the highest FP rates for all models while the lowest FP values were found for Random Forest followed by Gradient Boosting. Similarly, for the study conditions with a resampling ratio of 10:1, under-sampling generally produced the highest Recall for all models with Naïve Bayes having higher values. Oversampling produced the lowest FP rates and the highest Precision except for Decision Tree and Discriminant Analysis. Resampling in general yielded higher F1 scores and FP rates with the highest F1 score yielded from Random Forest with dual resampling.

In general, under-sampling led to higher Recall but lower Precision. Although its F1 scores could be as high as other resampling methods when the two class size ratios were larger, its FP rates were often the highest. Oversampling in general led to higher Precision and lower FP rates except for the resampling ratio of 1:1. Precision produced by dual resampling was generally higher than Recall. Its F1 scores and FP rates fell in between all resampling and non-resampling methods. In summary, compared with non-resampling, resampling may increase Recall or Precision depending on the resampling method and the ratio. As noted, there is a trade-off among Recall, Precision, and the FP rate. An increase in Recall is often associated with a decrease in Precision and an increase in the FP rate.

Input Feature Sets

The model performance with different input features were compared in terms of the four evaluation criteria. The results for dual resampling with a dual resampling ratio of 1:1 are presented in Figure 6. The summary for other ratios can be found in the online supplementary document in Figures S4 to S6. In general, the feature sets with augmented summary statistics performed much better than its counterpart without data augmentation. The feature set consisting of item responses and summary statistics was generally identified as the best performing one across all models and all resampling ratios. For Recall, two feature sets—response time only and response time plus the summary statistics—often led to the highest values compared with other features for Discriminant Analysis, Logistic Regression, and SVM. Three feature sets with augmented summary data often led to higher Recall for Gradient

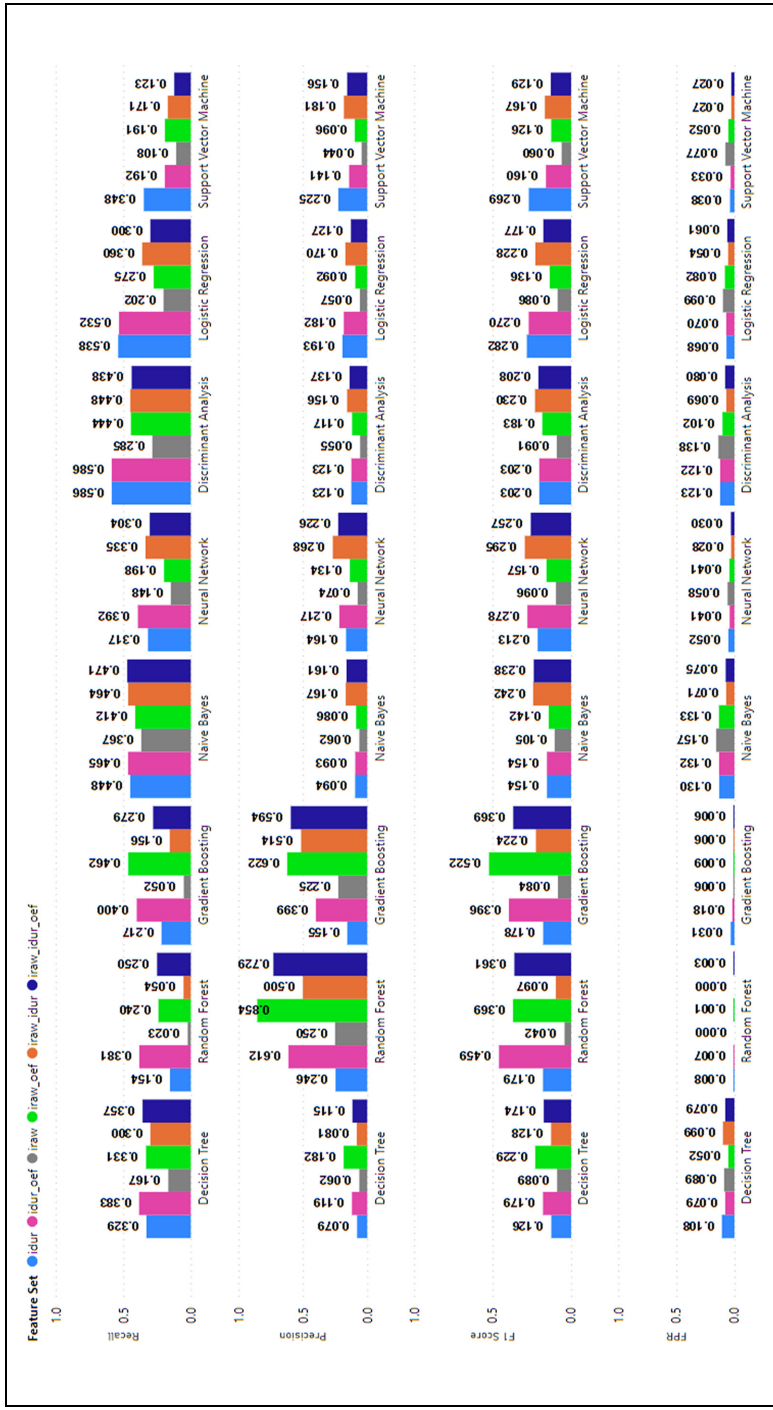

Figure 6. Model Performance Comparison With Six Different Input Features for Dual Resampling With a Ratio of 1:1 Between the Non-Cheater and Cheater Classes.

Boosting compared with its counterpart without augmented data. Input features affected Random Forest much with response time plus the summary features leading to the highest Recall. The impact of the input features on Precision was salient for Random Forest with response plus the summary statistics leading to the highest Precision, followed by the feature set containing all features. With the balance of Recall and Precision, responses plus the summary statistics led to the highest F1 score for Gradient Boosting. The input features did not affect the FP rates much, but Random Forest and Gradient Boosting had much smaller FP rates no matter the input features. Similar patterns were observed for the conditions with other resampling ratios.

In general, the impact of the input features was not consistent across models. However, using item response only always led to the lowest Recall, Precision, and F1 scores though the differences in the FP rates were not salient. Using response time only or response time plus the summary statistics performed better for each of the three models: Discriminant Analysis, Logistic Regression, and SVM. For Naïve Bayes and Neural Network, two feature sets—responses only and responses plus the summary statistics—performed worse than other feature sets. For Random Forest and Gradient Boosting, the top three performers were response time plus the summary statistics, item responses plus the summary statistics, and all features. In general, data augmentation by including the summary statistics improved the model performance for the two ensemble models: Random Forest and Gradient Boosting saliently, but not always the case for other base models.

Level 1 Base Model Performance Comparison

Regardless of the resampling methods and the input features, Gradient Boosting and Random Forest, the two ensemble machine learning models were in general the top performers in terms of the F1 score and the FP rates. Among all 624 study conditions, three top optimal designs for cheating detection include item responses plus the summary statistics with a dual resampling ratio of 10:1, item response plus the summary statistics with an oversampling ratio of 5:1, and item responses plus the summary statistics with an under-sampling ratio of 10:1.

Figure 7 presents Recall, Precision, F1 scores, and the FP rates for the six basic machine learning models and two ensemble learning models for dual resampling with a ratio of 10:1 with input features of item responses and the summary statistics. Naive Bayes produced the highest Recall of 0.525, followed by Gradient Boosting. Random Forest produced the highest Precision of 0.838, followed by Gradient Boosting of 0.771. There were not many differences in Precision among all other models. Due to a larger difference between Recall (0.309) and Precision (0.838) for Random Forest, its F1 score (0.447) was lower than that for Gradient Boosting which had the highest value (0.573) among all level 1 base models across all conditions. Random Forest and Gradient Boosting had about the same FP rates, with a slightly lower value for the former, but both were way lower than those from other base models.

Although Recall and Precision alone are not recommended evaluation criteria for classification with imbalanced classes, they are presented to show different

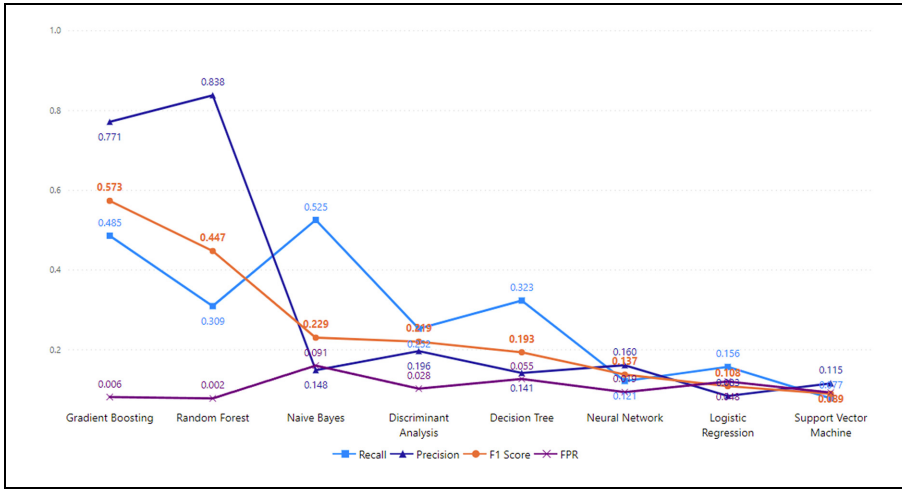

Figure 7. Base Model Comparison Based on Item Response and the Summative Statistics With a Dual Resampling Ratio of 10:1 Between the Non-Cheater and Cheater Classes.

perspectives of classification accuracy. In general, the two ensemble learning methods, Gradient Boosting and Random Forest, had the highest Precision values, indicating these two models performed the best in detecting cheaters who were predicted as cheaters. The Recall scores for these two models generally fell in the middle between the highest from Naïve Bayes and the lowest most often from SVM, indicating these two models performed neither the best nor the worst in detecting cheaters who were actual cheaters. In terms of the F1 score which is a balance between Recall and Precision, models were rank ordered from the best to the worst as Gradient Boosting, Random Forest, Naïve Bayes, Discriminant Analysis, Decision Tree, Neural Network, Logistic Regression, and SVM. In general, the two ensemble models had the lowest FP rates which were about 10 times smaller than those for other models. Similar patterns were found for the other two top performers as shown in Figures S7 and S8 in the online supplementary document with the study condition for Gradient Boosting with an oversampling ratio of 5:1 (F1 score = 0.546 and FP rate = 0.004) performed better than that with an under-sampling ratio of 10:1 (F1 score = 0.529 and FP rate = 0.013).

Development of the Meta-Models

To develop a meta-model in stacking for each of the top three optimal designs as identified in the above section, a set of rank-ordered base classifiers needs to be identified first. Then, a meta-model is developed by taking the outputs of the base models as its input features for training and prediction. Given the results from each level 1 base model, different combinations of the base models were sequentially explored

based on the model rankings. In total, there were seven sets of base models fed to develop a meta-model respectively.

For the study condition with a dual resampling ratio of 10:1 using the feature set consisting of item responses and the summary statistics, when four models Random Forest, Naïve Bayes, Discriminant Analysis, and SVM were used for stacking, the meta-model with the top two base models produced the highest Recall and F1 scores with comparable FP rates but not the highest Precision. For other stacking models, different combination of base models led to different best performers for this study condition. When Random Forest was used as the stacking model based on the outputs from the top two base models, it produced the highest F1 score (0.573) and Recall (0.594). For the same feature set with an oversampling ratio of 5:1 using the top two base models for stacking, Naïve Bayes as the stacking model performed the best in terms of the F1 score (0.569) and Recall (0.492). All other meta-models did not perform well and did not produce a F1 score higher than 0.5. The best performing meta-model was Discriminant Analysis used for stacking based on item responses and the summary statistics with an under-sampling ratio of 10:1. Its Recall (0.638), Precision (0.551), and F1 score (0.586) were all above 0.5 which is considered good model performance and the FP rate (0.014) was relatively low. The detail of the stacking model performance can be found in Figures S9, S10, and S11 in the online supplementary document.

Furthermore, the performance of the base models and the meta-model built upon stacking was compared. For the feature set consisting of item responses and the summary statistics with a dual resampling ratio of 10:1 with Random Forest used for stacking to produce the meta-model, the meta-model stacked on the top two base models performed about the same as Gradient Boosting and better than any other individual base classifier. They both had the same F1 score of 0.573. However, the meta-model yielded more balanced Recall (0.594) and Precision (0.568) with a FP rate of 0.013 while Gradient Boosting yielded lower Recall (0.485) but higher Precision (0.771) with a lower FP rate of 0.006 (see online supplementary Figure S12). For the feature set consisting of item responses and the summary statistics with an oversampling ratio of 5:1 using Naïve Bayes for stacking to develop the meta-model, the meta-model stacked on the top two base models performed the best compared with any other individual base classifier in terms of Recall (0.492) and F1 score (0.569) with more balance between Recall and Precision (0.739) and a FP rate of 0.006 (see online supplementary Figure S13).

Figure 8 summarizes the comparison for the feature set consisting of item responses and the summary statistics with an under-sampling ratio of 10:1 using Discriminant Analysis for stacking to develop the meta-model. The meta-model stacked on the top two base models performed the best compared with any other individual base classifier in terms of Recall (0.638) and F1 score (0.586) with more balance between Recall and Precision (0.551) and a FP rate of 0.014. To sum up, among all the compared models, the meta-model stacked on Discriminant Analysis using item responses plus the augmented summary statistics with an under-sampling

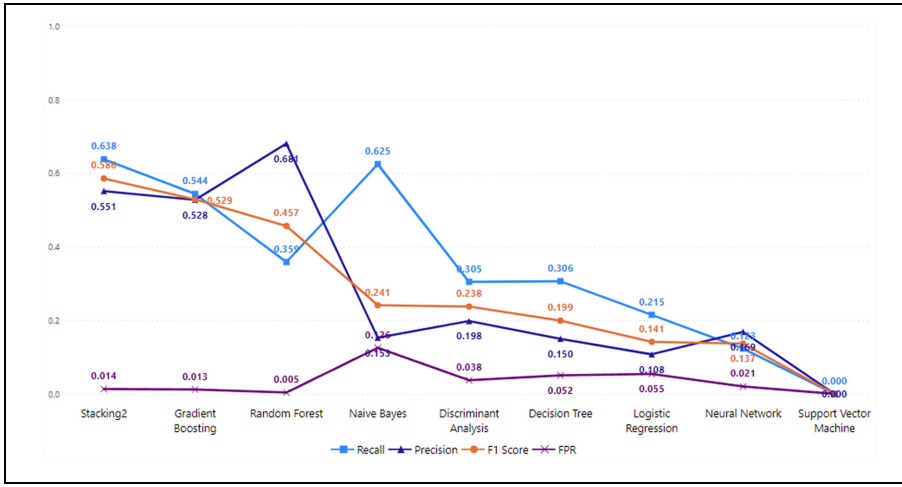

Figure 8. Comparison Between the Base Models and the Meta-Model Built Upon Stacking Using Discriminant Analysis Based on Item Response and Summative Statistics for an Under-Sampling Ratio of 10:1 Between the Non-Cheater and Cheater Classes.

ratio of 10:1 produced the highest F1 score (0.586) and balanced F1 score, Recall (0.638), and Precision (0.551). These three values were all higher than 0.5, which is considered as the best performing model in detecting the cheaters.

Summary and Discussions

Selecting a specific machine learning algorithm for out-of-sample data, the performance of competing machine learning algorithms needs to be evaluated to develop a classifier with the optimal design that leads to the highest detection accuracy of cheaters. Researchers have explored basic machine learning algorithms and some ensemble methods such as Random Forest (bagging) and Gradient Boosting (boosting) in cheating detection. However, no studies have explored stacking to build a meta-model for cheating detection, which is expected to perform better. Furthermore, class imbalance in cheating detection was not investigated either using resampling. Thus, this study explored the development of a meta-model based on the stacking ensemble method with the input from different base models using different stacking models for the meta-model development with resampling to tackle the issue of class imbalance. The stacking method was demonstrated empirically with a large-scale test dataset with potential cheaters flagged. Model performance was compared among the meta-model and base models in terms of Recall, Precision, F1 scores, and FP rates. The meta-model with stacking using Discriminant Analysis based on the top two base models generally performed the best when item responses and the summary statistics were used as the input features with an under-sampling ratio of 10:1. In general,

stacking, resampling, and feature sets including augmented summary data worked better than its counterparts.

Related to the performance of the three feature sets—responses only, response time only, and responses plus response time only—there were no consistent patterns. Sometimes including both item responses and response time performed worse than using either item responses only or response time only. This seems counterintuitive and inconsistent with other studies that used the same dataset. However, it is noted that other studies using the same dataset did not apply resampling to deal with class imbalance issue. Thus, the comparison between their findings and the findings from this study might not be meaningful and valid. No previous studies used the same dataset with data augmentation. This study included the augmented summary data at the test level and the aggregated level of items. The augmented summary data could be an additional source of information for cheating detection, which is expected to improve cheating detection accuracy. A plausible explanation for the worse performance of the feature set combining item response and response time data than either the feature set consisting of item responses only or response time only is that these two types of data may tackle different facets of cheating behaviors and do not converge to the same point in terms of providing useful information to facilitate cheating detection.

The threshold value used in this study for classifying a test-taker as a cheater or a non-cheater in all analyses was 0.5 for evaluating every model throughout the study. When resampling is applied, it re-balances the classes. It is expected that the use of a cut-off value of 0.5 is valid and no further adjustment of the threshold value needs to be made. However, the impact of different threshold values could be investigated more extensively in a future study given this is not the focus of this current study.

Machine learning algorithms have drawn increasing attention in psychometric analysis for test development in large-scale assessment where latent variable modeling of item responses prevailed for a long time. In recent years, classification and prediction based on machine learning algorithms for different psychometric purposes have marked their debut. These pioneering explorations include automated scoring, cheating detection, classification decisions based on multiple data sources, automated item generation, and cognitive diagnosis. When online testing becomes the mainstay of testing, the integration of multiple product and process data may impose more methodological challenges on the traditional psychometric analysis methods and models. Although joint modeling of product data and process data in psychometric models of responses and response time has been explored extensively, it is expected that when more data types are included in such joint models, model parameter estimation could impose challenges and issues. Thus, it is worthwhile to explore machine learning algorithms for different psychometric purposes. This article extended the use of machine learning methods for cheating detection by applying the stacking ensemble method, which involves using multiple base classifiers to develop a meta-classifier for classification and prediction with resampling for balancing the extreme imbalanced classes.

Future explorations may expand the current study by including more data features for cheating detection in large-scale testing such as answer changes, mouse clicks, and other biometric data such as eye-tracking data though the current study is limited by the availability of such data. Furthermore, the explored stacking machine learning can be extended to innovative assessments such as game-based or simulation-based assessment where the assessment data do not follow the standard, structured product and process data format. When test efficiency is really called for by test stakeholders, it is worthwhile to look into more test delivery algorithms such as computerized adaptive tests (Cui et al., 2018) with item review when cheating could be intentionally committed in a larger scale by answering each item incorrectly to get easier items, then going back to change their answers to early items.

In summary, this study explored the stacking ensemble learning method with resampling for cheating detection. Its performance was compared with different basic and other ensemble machine learning methods on detecting cheaters in large-scale assessment. It is generally believed that more data is better for decisions. However, how to decode and synthesize different data types to come up with informed decisions is still a challenge when more assessment data become available. The empirical results from this study demonstrated how to integrate and take advantage of more data types to facilitate cheating detection using stacking to develop a meta-model. Machine learning of the outputs of base machine learning algorithms is expected to produce more accurate prediction results by synthesizing information from different base models.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Hong Jiao https://orcid.org/0000-0001-5014-6698

Supplemental Material

Supplemental material for this article is available online.

References

Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., & Ridella, S. (2012). The ''K'' in K-fold cross validation. In *ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (pp. 25–27). https://www.esann.org/sites/default/files/proceedings/legacy/es2012-62.pdf

Bishop, S., & Egan, K. (2017). Detecting erasures and unusual gain scores. In G. J. Cizek & J. A. Wollack (Eds.), *Handbook of quantitative methods for detecting cheating on tests* (pp. 193–213). Routledge. https://doi.org/10.4324/9781315743097-10

Chan, K., & Stolfo, J. (1997). On the accuracy of meta-learning for scalable data mining. *Journal of Intelligent Information Systems*, *8*(1), 5–28.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *JAIR Journal of Artificial Intelligence Research*, *16*, 321–357.

Chen, Y., Lu, Y., & Moustaki, I. (2020). *Detection of two-way outliers in multivariate data and application to cheating detection in educational tests*. arXiv:1911.09408. https://arxiv.org/abs/1911.09408v2

Cizek, G. J., & Wollack, J. A. (2017). *Handbook of quantitative methods for detecting cheating on tests*. Routledge.

Cui, Z., Liu, C., He, Y., & Chen, H. (2018). *Comparison of algorithms that allow item review in computerized adaptive test* [ACT Research Report]. https://www.act.org/content/dam/act/unsecured/documents/pdfs/R1709-cat-comparison-2018-10.pdf

Drasgow, F., Levine, M. V., & Williams, E. (1985). Appropriateness measurement with polychotomous item response models and standardized indices. *British Journal of Mathematical and Statistical Psychology*, *38*, 67–86.

Forman, G., & Scholz, M. (2010). Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *SIGKDD Explorations*, *12*, 49–57.

Guo, X., Yin, Y., Dong, C., Yang, G., & Zhou, G. (2008, October 18–20). *On the class imbalance problem* [Conference session]. Fourth International Conference on Natural Computation, ICNC '08 (Vol. *4*), Jinan, China. https://doi.org/10.1109/ICNC.2008.871

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284.

Japkowicz, N. (2000). *The class imbalance problem: Significance and strategies*. In Proceedings of the *2000 International Conference on Artificial Intelligence (IC-AI'2000): Special Track on Inductive Learning*, Las Vegas, NV, United States.

Le, T., Fu, W., & Moore, J. (2020). Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics*, *36*(1), 250–256.

Man, K., & Harring, J. R. (2020). Assessing preknowledge cheating via innovative measures: A multiple-group analysis of jointly modeling item responses, response times, and visual fixation counts. *Educational and Psychological Measurement*, *81*(3), 441–465.

Man, K., Harring, J. R., & Sinharay, S. (2019). Use of data mining methods to detect test fraud. *Journal of Educational Measurement*, *56*(2), 251–279.

Meijer, R. R., & Sijtsma, K. (2001). Methodology review: Evaluating person fit. *Applied Psychological Measurement*, *25*(2), 107–135.

Pavlyshenko, B. (2018). Using stacking approaches for machine learning models. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)* (pp. 255–258). https://doi.org/10.1109/DSMP.2018.8478522

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Raschka, S. (2018). MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *The Journal of Open Source Software*, *3*(24), Article 638. https://doi.org/10.21105/joss.00638

Sasaki, Y. (2007). The truth of the F-measure. *Teach Tutor Mater*. https://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf

Shu, Z., Henson, R., & Luecht, R. (2013). Using deterministic, gated item response theory model to detect test cheating due to item compromise. *Psychometrika*, *78*, 1–17. https://doi.org/10.1007/s11336-012

Sijtsma, K., & Meijer, R. R. (1992). A method for investigating the intersection of item response functions in Mokken's nonparametric IRT model. *Applied Psychological Measurement*, *16*(2), 149–157.

Skorupski, W., Fitzpatrick, J., & Egan, K. (2016). A Bayesian hierarchical model for detecting aberrant growth at the group level. In G. J. Cizek & J. A. Wollack *(Eds.), Handbook of quantitative methods for detecting cheating on tests* (pp. 232–244). Routledge.

Toton, S. L., & Maynes, D. D. (2019). Detecting examinees with pre-knowledge in experimental data using conditional scaling of response times. *Frontiers in Education, 4*, Article 49.

van der Flier, H. (1982). Deviant response patterns and comparability of test scores. *Journal of Cross-Cultural Psychology*, *13*, 267–298.

Wang, C., Xu, G., Shang, Z., & Kuncel, N. (2018). Detecting aberrant behavior and item preknowledge: A comparison of mixture modeling method and residual method. *Journal of Educational and Behavioral Statistics*, *43*, 469–501.

Wollack, J. A. (1997). A nominal response model approach for detecting answer copying. *Applied Psychological Measurement*, *21*, 307–320.

Zopluoglu, C. (2019). Detecting examinees with item preknowledge in large-scale testing using extreme gradient boosting (XGBoost). *Educational and Psychological Measurement*, *79*(5), 931–961. https://doi.org/10.1177/0013164419839439