Predictive Modeling for Risk Analysis of Wood Packaging Material

Yanran Guan

July 10, 2018

1 Background

Under various acts and regulations, the Canada Border Services Agency (CBSA) conducts inspections of the Canadian Food Inspection Agency (CFIA) regulated imported commodities to verify compliance. The CFIA and the CBSA have partnered in a joint initiative to improve the efficiency and effectiveness of those oversight activities. This initiative focuses on the wood packaging material (WPM). WPM poses a significant risk because it can act as a pathway for foreign plant pests, diseases, or invasive species. The objective of this work will be to enhance the effectiveness of CBSA oversight resources by developing predictive analytics and decision support tools based on statistically valid and scientific principles.

In order to provide objective and defensive risk analysis regarding imported wood packaging materials, evidence-based research has been proposed. This experiment is based on the data of good packages that have been shipped to Canada over the past 4 years (2014–2017) and aims at building a classification and regression model on the historical data for risk assessment and management.

2 Data Profiling and Preprocessing

The original dataset has 20 variables and 85488 observations in total, with `Compliance` being the target variable. The two values in the target variable were relabeled respectively as `Compliant` and `Non.Compliant`, with `Compliant` being the positive class, `Non.Compliant` being the negative class.

Among the remaining variables, `LiveInsect`, `Frass.Saudust`, `Phyto`, `Bark`, `Tunnels`, and `Fungi` are examining reasons for non-compliance, which have a high correlation with the prediction target. Besides, other nuisance variables, e.g., `ï..Exam.Extract.No.` (with 55729 distinct values), `Exam.Date` (with 1478 distinct values), `Shipper.City` (with 14414 distinct values), `Shipper.Name` (with 37160 distinct values), `Consignee.Name` (with 31581 distinct values), etc., should also be removed from the predictor set.

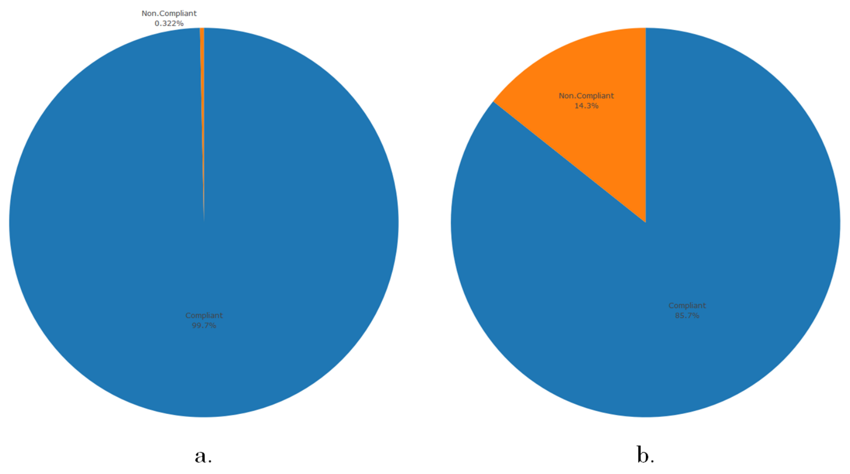

Figure 1: Distribution of target classes.

Looking at the distribution of target classes on manufactured packaging (MP) packages (Figure 1a) and non-MP packages (Figure 1b), we could find a very remarkable pattern: the non-compliance has a very few occurrences among the MP packages. The non-compliant observations only take up 0.322% of the subset of MP packages, which could be ignored. Thus, this experiment focuses on analyzing the WPM packages and double material (WPM & MP) packages, which take up 28398 observations in this dataset.

However, the distribution of target classes is still imbalanced among the WPM packages and WPM & MP packages, as shown in Figure 1b. with the majority class `Compliant` taking up 85.7% of all observations and the minority class `Non.Compliant` only taking up 14.3%. Standard classifier algorithms, having a bias towards classes that have a large number of instances, could make the classification result overfit to the majority class, hence more effort should be made to balance the data in the modeling phase.

There could be redundant features in the dataset as well. It is necessary to have an analysis of the correlation matrix of the data's attributes, as shown in Figure 2.

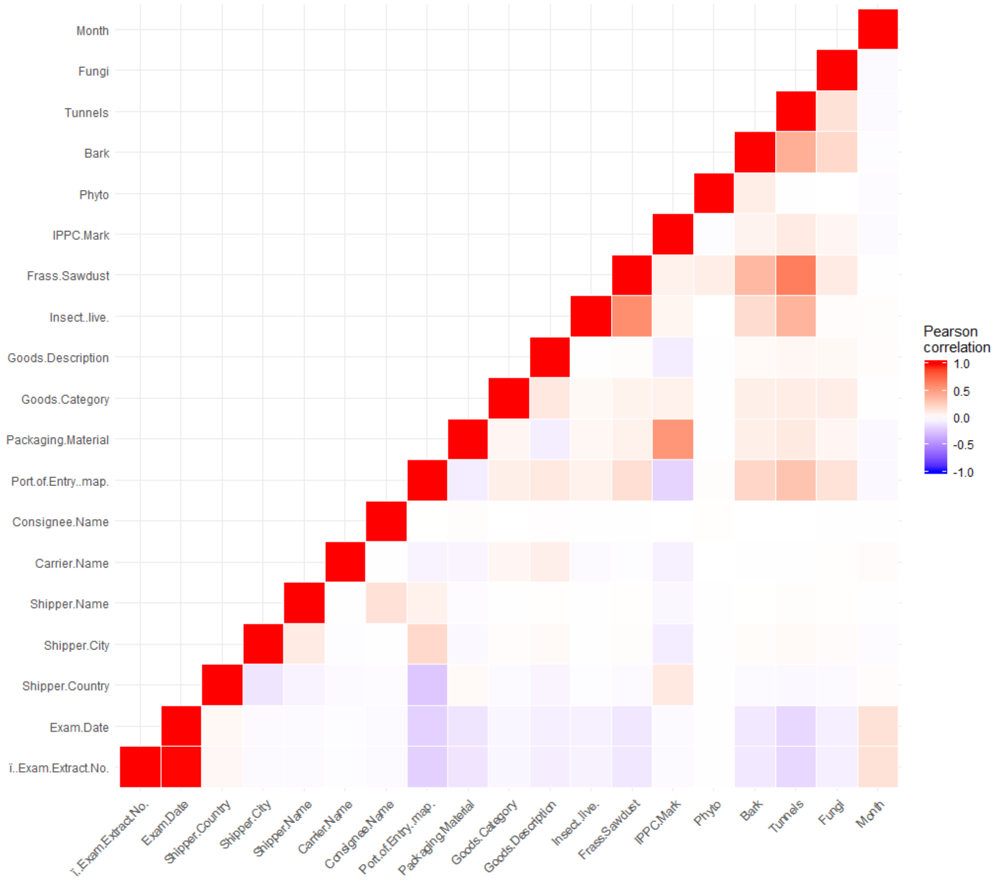

Figure 2: Correlation matrix.

The Pearson correlation coefficient measures both strength and direction of a linear relationship between two variables. The positive or negative value indicates respectively an uphill or downhill linear relationship. To avoid redundancy, highly correlated variable pairs should not be preserved in the predictor set. Thus, we select the variables `Shipper.Country`, `Port.of.Entry`, `Packaging.Material`, `Goods.Category` and `Month` as predictors.

We could take a look at the breakdown of this dataset by the selected features (Figure 3).

As is shown in Figure 3b and Figure 3d, there are empty values in variable `Goods.Category`. Assuming that the type of goods could be a more significant cause of the non-compliance of a package than the port of entry, we first built up a hash table that maps the `Goods.Description` and `Goods.Category` to fill up as many missing `Goods.Category` as possible, then dropped the observations that still remain empty in `Goods.Category`, 16518 observations were kept.

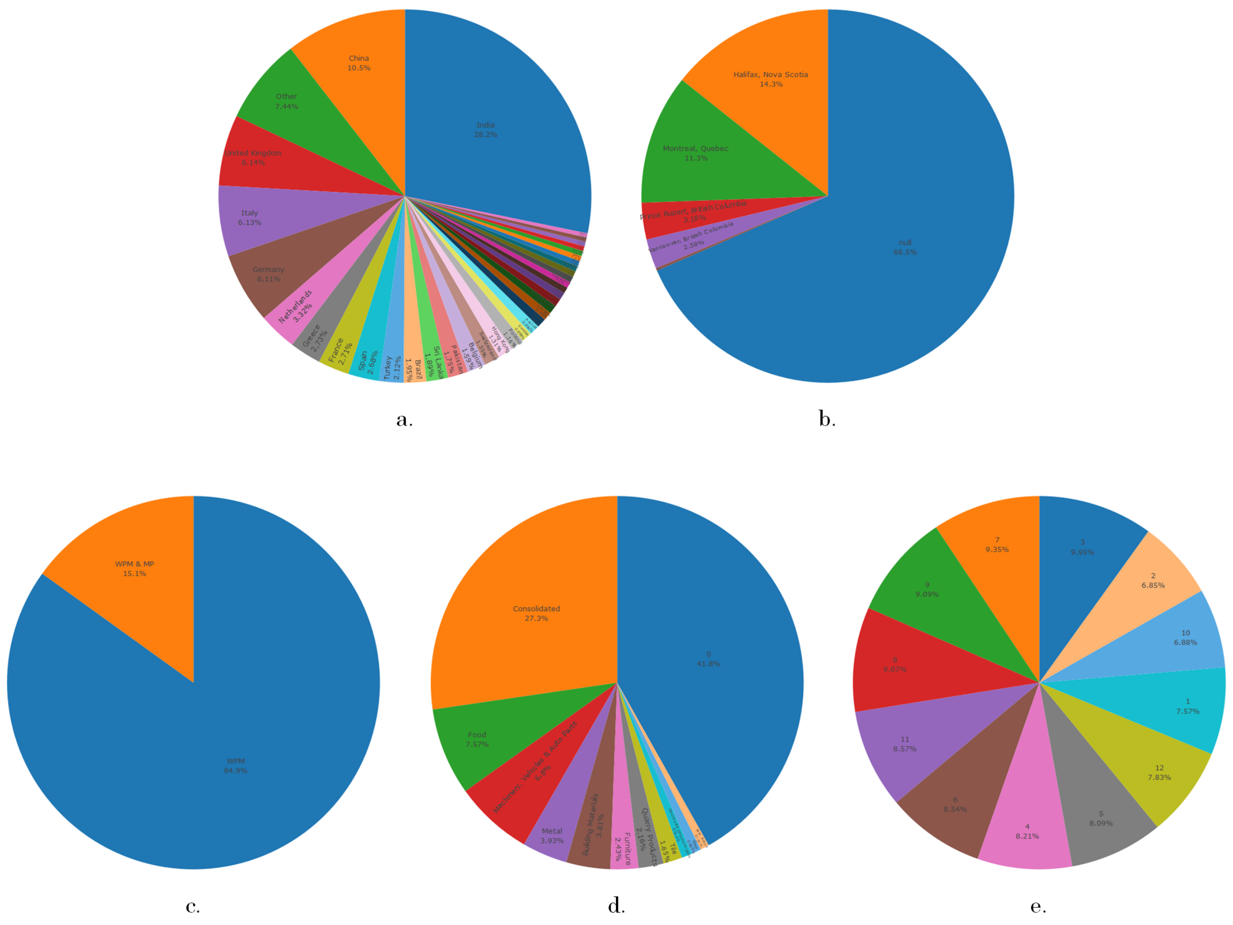

Figure 3: Data breakdown.

In addition, the variable `Shipper.Country` has 174 distinct countries, while the countries having a subset of fewer than 200 observations only make up 7.44% of the whole dataset, which is shown in Figure 3a. In consideration of data fuzzification, the countries in minority (with less than 200 observations) were relabeled as "Other" and the number of distinct countries was reduced to 37.

Lastly, for evaluating model quality, stratified sampling was applied to the dataset, creating a training set with 75% of all the observations and a test set with 25% of the observations.

3 Data Modeling

The classification and regression tree (CART) (Breiman et al. 1984) was chosen as the predictive model in this experiment. CART not only provides a discrete class label as the predicted outcome for each input instance but also makes a distributional prediction, i.e., calculates an approximate probability, for each predicted class. Figure 4 shows an example of a CART node, where `Compliant` is the predicted class (`Compliant` or `Non.Compliant`), 0.05 indicates the predicted probability of non-compliance, 26% refers to the percentage of observations in the node.

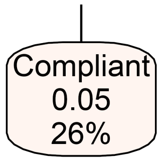

Figure 4: CART node example.

3.1 Gini Index

CART uses the Gini index as the splitting rule by default (Han, Kamber, and Pei 2011). Gini index, also known as Gini impurity, is a measurement of the likelihood of incorrect classification of a chosen tuple from a data partition or training set, given that the tuple was randomly classified according to the distribution of class labels.

Suppose that D is a data partition with n distinct class labels ($C_i, i \in \{1, 2, 3, \ldots, n\}$) in target variable. Let $C_{i,D}$ be the set of tuples of class C_i in D. Let $|D|$ and $|C_{i,D}|$ denote the number of tuples in D $C_{i,D}$, respectively. The impurity of D can be computed by summing the probability $p_i = \frac{|C_{i,D}|}{|D|}$ of a tuple in D belonging to class C_i times the probability $1 - p_i$ of a mistake in categorizing that tuple, written as:

$$\mathrm{Gini}(D) = \sum_{i=1}^{n} p_i(1 - p_i) = 1 - \sum_{i=1}^{n} {p_i}^2. \tag{1}$$

It reaches its minimum (zero) when all tuples fall into a single target category.

The Gini index considers a binary split for each variable, thus a weighted sum of the impurity on each split is computed based on each resulting partition. For example, if a binary split on variable A partitions D into D_1 and D_2, the Gini index of D given that partitioning is defined as:

$$\mathrm{Gini}_A(D) = \frac{|D_1|}{|D|}\mathrm{Gini}(D_1) + \frac{|D_2|}{|D|}\mathrm{Gini}(D_2). \tag{2}$$

For each variable, each of the possible binary splits is considered. The subset that gives the minimum Gini index for a variable is selected as its splitting subset.

Based on the Gini index of D before and after the data partitioning, the reduction in impurity that would be incurred by a binary split on variable A could be calculated as below:

$$\Delta\mathrm{Gini}(A) = \mathrm{Gini}(D) - \mathrm{Gini}_A(D). \tag{3}$$

Through the tree structure of CART, the Gini indices are computed recursively on each node. The variable that maximizes the reduction in impurity, or, equivalently, has the minimum Gini index, is selected as the splitting variable.

3.2 Complexity Parameter

The splitting of a CART stops when its complexity parameter (cp) reduces to a certain value. The cp in CART measures the minimum improvement needed at each node and is calculated based on the cost complexity (cc) of the model, which is defined as:

$$cc = \sum_{i=1}^{n_{\mathrm{tm}}} \mu_i + \lambda n_{\mathrm{split}}, \tag{4}$$

where n_{tm} is the number of terminal nodes of a given tree, n_{split} is the total split number, μ_i, where $i \in \{1, 2, 3, \ldots, n_{\text{tm}}\}$, is the number of misclassifications on each terminal node, and λ is a penalty term derived from cross-validation.

In this experiment, the optimized *cp* value was determined through the tuning length. The tuning length was set to 10, i.e., 10 different *cp* values were being used to build the models and the best one was selected from them.

3.3 Cross-validation

Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent dataset. One round of k-fold cross-validation partitions a sample of data into k complementary subsets, using one of the subsets as the training set to perform analysis, and validating the analysis on the other subsets. In this experiment, a repeated 10-fold cross-validation was applied. The repeats number was set to 5.

3.4 Receiver Operating Characteristic

The performance of machine learning algorithms is typically evaluated by a confusion matrix as illustrated in Table 1 (for a binary classification problem). The columns are the Predicted class and the rows are the Actual class. In the confusion matrix, *TP* is the number of positive examples correctly classified (true positives), *FP* is the number of negative examples incorrectly classified as positive (false positives), *FN* is the number of positive examples incorrectly classified as negative (false negatives) and *TN* is the number of negative examples correctly classified (true negatives).

Table 1: Confusion matrix.

	Actual positive	Actual negative
Predicted positive	TP	FP
Predicted negative	FN	TN

As discussed before, an imbalanced dataset could produce a model overfit to the majority class, and make either the true positive rate ($TPR = \frac{TP}{TP+FN}$) or the true negative rate ($TNR = \frac{TN}{TN+FP}$) unacceptably low. In this case, the receiver operating characteristic (ROC) could be a proper evaluation metric that takes both *TPR* and *TNR* into consideration. The ROC is usually represented as a curve, i.e., the ROC curve. The ROC curve is created by plotting the *TPR* against the false positive rate ($FPR = 1 - TNR$). To measure the ROC of a model for model comparison, we simply compute the area under the curve (AUC) of the ROC curve, a model with higher AUC implies a better ROC and better model quality.

3.5 Data Balancing Techniques

Due to the inherent complex characteristics of the imbalanced dataset, multiple approaches were taken to handle this issue, e.g., class weighting, data resampling, and synthetic sampling. In this experiment, an original decision tree without any data balancing technique is also included for comparison.

3.5.1 Class Weighting

Class weights could be incorporated into the model by assigning a weight to each category of the target variable. To make the model invariant towards sample bias, the minority class was assigned with a lower weight while the majority class was with a higher weight. The class weights were calculated as follows:

$$\begin{cases} w_{\text{major}} = \frac{1}{2n_{\text{major}}}, \\ w_{\text{minor}} = \frac{1}{2n_{\text{minor}}}, \end{cases} \tag{5}$$

where w_{major} and w_{minor} represent respectively the weights for majority class and minority class. The n_{major} and n_{minor} refer to the number of observations in the majority and minority classes. This formula ensures that the weights of all the observations sum up to 1.

3.5.2 Data Resampling

Data resampling refers to the process of data down-sampling or data up-sampling. In a binary classification problem, either down-sampling or up-sampling could be applied to deal with imbalanced data. Down-sampling, also known as under-sampling, creates a balanced dataset by matching the number of samples in the minority class with a random sample from the majority class. Up-sampling, also known as over-sampling, matches the number of samples in the majority class with duplicating data from the minority class.

3.5.3 Data Synthesizing

Instead of duplicating observations, data synthesizing methods create new instances in the neighborhoods of observations of the minority class, by operating in feature space rather than data space. Two data synthesizing algorithms were applied in this experiment, respectively random over-sampling examples (ROSE) (Menardi and Torelli 2014) and synthetic minority over-sampling technique (SMOTE) (Chawla et al. 2002).

ROSE is a data balancing algorithm that deals especially with binary classification problems. The algorithm can be described as below.

Consider a training set T of size n. Its generic instance can be described as $(\mathbf{x}_i, y_i)$, where the class label y_i belongs to the set of target classes $\{C_1, C_2\}$ and $\mathbf{x}_i$ is a vector consists of predictor variables with a probability density function $f(\mathbf{x})$. Let $n_j < n$ be the size of C_j, where $j = 0, 1$. The ROSE procedure generates one new artificial instance in the following steps:

1. Select $y^* = C_j$ with probability $\frac{1}{2}$.
2. Select $(\mathbf{x}_i, y_i)$ in T, such that $y_i = y^*$ with probability $p_i = \frac{1}{n_j}$.
3. Sample $\mathbf{x}^*$ from $K_{H_j}(\cdot, \mathbf{x}_i)$, with K_{H_j} a probability distribution centered at $\mathbf{x}_i$ and depending on the covariance matrix H_j.

Essentially, we draw from the training set an observation belonging to one of the two classes $\{C_1, C_2\}$, and generate a new instance $(\mathbf{x}^*, y^*)$ in its neighborhood, where the neighborhood width is governed by H_j.

SMOTE generates synthetic instances for the minority classes by interpolation. For example, let a minority class in the training set contain T_{minor} instances, which is to be over-sampled to T_{minor}^N, where $N > 1$. Considering an instance from the minority class with a feature vector $\mathbf{x}_i$, where $i \in \{1, 2, 3, \ldots, T_{\text{minor}}\}$, for each $\mathbf{x}_i$, SMOTE generates new instances around it in the following steps:

1. Select k nearest neighbors of $\mathbf{x}_i$ from the T_{minor} minority instances.
2. Select one instance $\mathbf{x}_{i(nn)}$ randomly from the k nearest neighbors, where $nn \in \{1, 2, 3, \ldots, k\}$, and generate a new instance $\mathbf{x}_{i,j}$ based on the feature difference of $\mathbf{x}_{i(nn)} - \mathbf{x}_i$ and a random number ζ between 0 and 1:
$$\mathbf{x}_{i,j} = \mathbf{x}_i + \zeta(\mathbf{x}_{i(nn)} - \mathbf{x}_i). \tag{6}$$
3. Repeat step 2 for N times, N new instances $\mathbf{x}_{i,j}$, where $j \in \{1, 2, 3, \ldots, N\}$ will be generated.

We choose 5 nearest neighbors for sampling and the size N of over-sampling depends upon the proportion of the minority class in the dataset.

4 Results and Conclusions

To compare the performance between different models and find the best one from the models with different data balancing approaches, 6 CART models were being trained and tested in this experiment, they are respectively the original model (without using data balancing approach), the weighted model, the down-sampled model, the up-sampled model, the ROSE model, and the SMOTE model. the ROC was used as the model evaluation metric. The ROC curves for different models are illustrated in Figure 5.

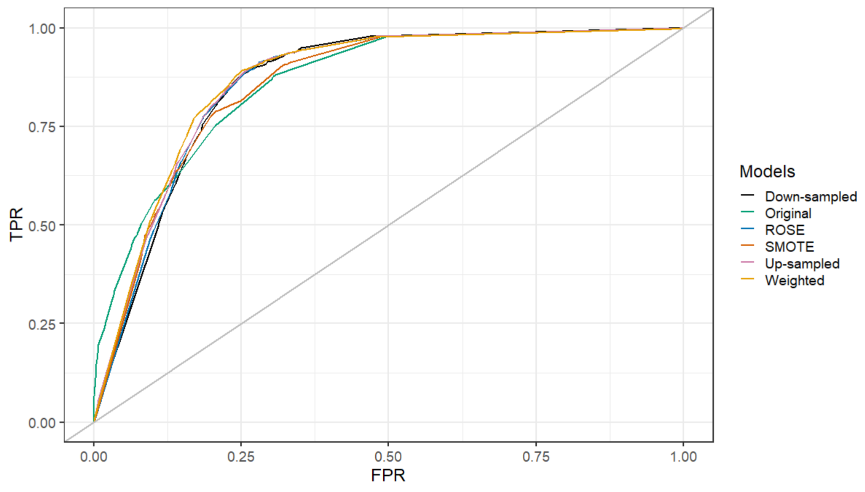

Figure 5: ROC curves of 6 models.

Generally speaking, a better model has a higher AUC value. According to Figure 5, the weighted model, down-sampled model, and up-sampled model are relatively better than the original model, ROSE model, and SMOTE model. The low performance of the ROSE model and SMOTE model could be caused by the unreasonable artificial instances being synthesized by the algorithms. This implies that such data synthesizing algorithms may not fit well with this specific dataset.

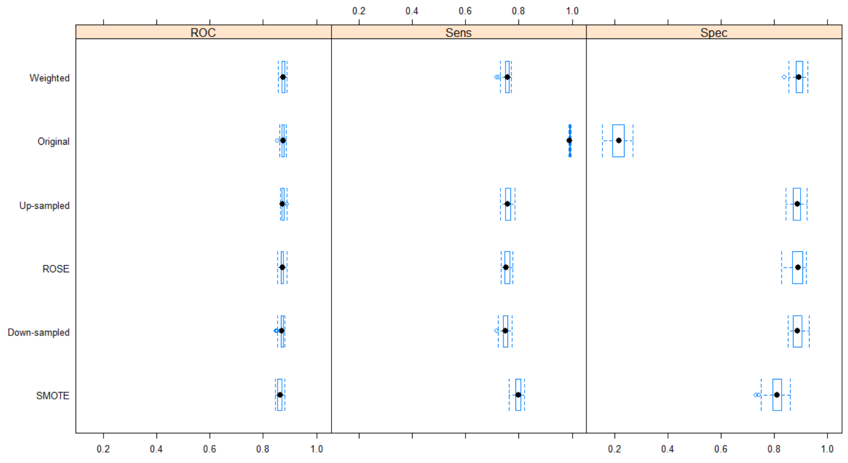

Figure 6: ROC of 6 models.

In this risk analysis study of the wood packaging material, the main objective is to find out as many non-compliant packages as possible and avoid an actually non-compliant package being mistakenly classified, i.e., the type I error (the rejection of a true null hypothesis, equivalent with *FP*) should be minimized as much as possible. Figure 6 shows explicitly the *TPR* and *TNR* on the training set of the 6 models in the shape of box plots, where "Sens" refers to sensitivity, which is synonymous for *TPR*, and "Spec" refers to the specificity, which is synonymous for *TNR*.

As far as the median value of *TNR* is concerned, the down-sampled model, weighted model, and up-sampled model show a distinct better performance in lowering the type I error. The down-sampled model, although having the highest *TNR* median among the 3 models, could cause some distortion to the dataset, as the nature of down-sampling is to shrink the data and thus some patterns could be lost during this process. The up-sampled model, on the other hand, may amplify some existing patterns even noises in the dataset and make the model overfit to the training set. Thus, the weighted model could be a good and neutral choice, as no modification on the original data is involved in this model.

Table 2: Confusion matrices for different models.

	Actual `Compliant`	Actual `Non.Compliant`
Original model		
Predicted `Compliant`	16486	1142
Predicted `Non.Compliant`	136	282
Weighted model		
Predicted `Compliant`	12622	175
Predicted `Non.Compliant`	4000	1249
Down-sampled model		
Predicted `Compliant`	12459	169
Predicted `Non.Compliant`	4163	1255
Up-sampled model		
Predicted `Compliant`	12401	163
Predicted `Non.Compliant`	4221	1261
ROSE model		
Predicted `Compliant`	12352	158
Predicted `Non.Compliant`	4270	1266
SMOTE model		
Predicted `Compliant`	13325	319
Predicted `Non.Compliant`	3297	1105

To have a more unbiased review of the 6 models, we prepared a test set in advance, which is derived from the stratified sampling of the original dataset. The predicting outcomes on the test set are shown in the form of confusion matrices in Table 2. Still, the down-sampled model, weighted model, and up-sampled model have better outcomes with low type I errors.

References

Breiman, Leo, Jerome Friedman, Charles J. Stone, and R.A. Olshen. 1984. *Classification and Regression Trees*. Taylor & Francis.

Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. "SMOTE: Synthetic Minority over-Sampling Technique." *Journal of Artificial Intelligence Research* 16: 321–57.

Han, Jiawei, Micheline Kamber, and Jian Pei. 2011. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science.

Menardi, Giovanna, and Nicola Torelli. 2014. "Training and Assessing Classification Rules with Imbalanced Data." *Data Mining and Knowledge Discovery* 28 (1): 92–122.