Paper 3604-2018

MITIGATING THE EFFECTS OF CLASS IMBALANCE USING SMOTE AND TOMEK LINK UNDERSAMPLING IN SAS®

Jonathan Boardman, Kennesaw State University; Kyle Biron, Kennesaw State University; Ryan Rimbey, Kennesaw State University

ABSTRACT

When dealing with imbalanced data, many standard learning algorithms have trouble adequately learning the underrepresented class(es). One approach to mitigating the effects of class imbalance is using sampling methods to alter the training data in a way that makes it easier for the classifier to learn the class(es) of interest. Two such techniques are SMOTE, which generates new, synthetic minority class examples, and Tomek link undersampling (Tomek), which helps to clear majority class examples from class boundaries. Both methods were implemented in SAS® along with a combination of SMOTE followed by Tomek link undersampling (SMOTE+Tomek). Using a dataset of credit card fraud transactions where the class of interest was either fraud (minority class) or not fraud (majority class), the efficacy of these techniques was tested by training four classifiers – a random forest, a neural network, a support vector machine, and a rule induction classifier – on training datasets processed using each method and testing them on a validation set. The performance of the classifiers on the validation set was assessed using the area under the ROC curve (ROC index), precision, recall, and the ratio of false negatives (FN) to false positives (FP). SMOTE and SMOTE+Tomek were the most effective preprocessing methods for improving the detection of fraudulent transactions in the credit card dataset. Both methods improved recall and lowered the FN/FP ratio for every classifier, indicating improved sensitivity to fraud. At the same time, SMOTE and SMOTE+Tomek improved the ROC index, indicating an improved ability to distinguish fraud from non-fraud.

INTRODUCTION

A dataset is considered to be imbalanced if the target classes are not present in equal proportions. Such data are commonplace in many settings, including rare disease diagnostics, high energy physics, defective product detection, identification of oil spills from satellite imagery, and credit card fraud detection (Kubat, Holte, & Matwin 1998). Small deviations from equality are usually not an issue, but when the discrepancy becomes large, the ability of a classifier to distinguish minority class examples will generally degrade. In general, this is because classifiers attempt to minimize some cost, and if errors in both classes are treated equally, then a classifier will develop a bias towards the majority class. If the cost of misclassification is indeed equal, or if misclassifying majority class instances is more costly, then the imbalance is a non-issue. However, it is usually the minority class that is of primary interest; in this case, class imbalance is a problem.

Many methods have been developed for dealing with class imbalance. Roughly, they can be divided into two categories: algorithm-level methods and data-level methods. Typically, some combination of both data-level methods and algorithm-level methods is employed. Algorithm-level methods include using appropriate search algorithms guided by appropriate metrics and using algorithms that focus on the rare classes (He & Ma 2013). Data-level methods include strategic information acquisition (e.g. active learning) and, when this is not feasible, sampling methods. Some well-known sampling methods are oversampling the minority class, undersampling the majority class, generating synthetic minority class examples, and — the focus of this research project — cleaning up class boundaries using Tomek links.

PROBLEM

Although the Tomek link algorithm was originally developed for the purpose of reducing training set size, the algorithm has shown promise as a preprocessing technique for imbalanced datasets, particularly with respect to increasing recall on the minority class (Tomek 1976; Kubat & Matwin 1997). Tomek links have been particularly successful when used in concert with other sampling techniques such as SMOTE, an algorithm for generating new synthetic minority class examples (Kubat & Matwin 1997; He & Garcia 2009; Batista, Prati, & Monard 2004; More 2016). Wang et al. have already implemented the SMOTE algorithm in SAS® and demonstrated its success in improving the classification of rare events (Wang, Lee, & Wei 2015). However, SAS® code for Tomek link removal or Tomek link undersampling – if it exists – is not publicly available. One of the major objectives of this project was to rectify this deficiency by creating an effective tool for preprocessing training data using a combination of SMOTE and Tomek link undersampling.

Thus, the goal of this project was twofold: (1) implement Tomek link undersampling in SAS® and (2) compare the efficacy of standalone Tomek link undersampling (Tomek), SMOTE, and SMOTE followed by Tomek Link undersampling (SMOTE+Tomek) as preprocessing methods for a highly imbalanced credit card fraud dataset. The methods were applied to a training dataset (70% of the original data stratified by class), which was used to train four separate classifiers for each method in SAS® Enterprise Miner™ - a random forest, a neural network, a support vector machine, and a rule induction classifier. The resulting classifiers were then evaluated on a validation dataset consisting of the remaining 30% of the original data. Since accuracy is known to be a poor metric when working with imbalanced data, the classifiers were evaluated using the area under their ROC curves (ROC Index), precision, recall, and the ratio of false negatives to false positives.

ALGORITHMS

TOMEK LINK UNDERSAMPLING

For a dataset with some target class, a Tomek link is a pair of examples that are (1) nearest neighbors of one another and (2) have different target class labels (Tomek 1976). Examples that belong to Tomek link pairs are likely to be either noise points or points that lie close to the optimal decision boundary (Kubat & Matwin 1997). Removing these points can result in more well-defined class clusters in the training data, which can lead to better classifiers (He & Garcia 2009).

In Tomek link undersampling (as opposed to Tomek link removal), only the majority class example in each Tomek link pair is removed. There are two reasons for this. First, in an imbalanced dataset, the minority class examples may be too valuable to waste, especially if the minority class is underrepresented. Second, recall on the minority class is frequently of greater importance than precision, so it is worth inadvertently retaining some noisy data in order to avoid losing rare cases (e.g. catching credit card fraud is worth placing a few holds on legitimate accounts).

The Tomek link undersampling algorithm used in this paper works as follows:

1. Start with an input dataset *D*.
2. Generate a dataset *F* with only minority class observations from the input dataset *D*.
3. Generate a dataset *N* with the nearest neighbor for each observation in the input dataset *D*, and create and populate a variable with the nearest neighbor of each nearest neighbor.
4. Merge the two datasets *F* and *N*, keeping just the minority class observations that are the nearest neighbors of their nearest neighbors. Call this dataset *M*.
5. If any nearest neighbor in the merged dataset *M* is not a minority class observation, the nearest neighbor is the majority class member of a Tomek link and is removed from the input dataset *D*.
6. Stop. The remaining observations in *D* comprise the output dataset.

The above steps provide a conceptual overview of the Tomek link undersampling procedure, but more details can be found in the commented code in the appendix (see Tomek link Undersampling section).

SMOTE

Synthetic minority oversampling technique (SMOTE) artificially increases the number of minority class examples by generating synthetic examples (Chawla, Bowyer, Hall, & Kegelmeyer 2002). There are several subtle variations on the SMOTE algorithm, but the implementation used in this paper works as follows:

1. Start with an input dataset *D*.
2. Generate a dataset *F* with only minority class observations from the input dataset *D*.
3. Set the number *k* of nearest minority class neighbors to use for SMOTE-ing.
4. Set the SMOTE multiplier *m*, which is the number of additional minority class instances desired for each of the original minority class examples.
5. Generate a dataset *N* with the *k* nearest minority class neighbors for each minority class example.
6. Generate a dataset *L* of observation-nearest neighbor pairs by randomly sampling from one of the *k* nearest minority class neighbors for each observation in *N* and repeating this procedure *m* times.
7. For each observation-nearest neighbor pair in *L*:
    a. Take the difference between the feature vectors of the neighbor and the observation under consideration.
    b. Multiply this difference by a uniform random number between 0 and 1.
    c. Add the scaled difference vector to the feature vector of the observation under consideration to obtain a new, synthetic minority class example.
    d. Store the new, synthetic observations in a dataset *R*.
8. Add the new, synthetic minority class cases in *R* to the input dataset *D*.
9. Stop. The observations in *D* comprise the output dataset.

The above steps provide a conceptual overview of the SMOTE procedure, but more details can be found in the commented code in the appendix (see SMOTE section). The code in the appendix is a modified version of the code used by Wang et al. (Wang, Lee, & Wei 2015).

For most basic versions of SMOTE, there are two parameters that can be adjusted. The first is the SMOTE multiplier *m*. The second parameter is the number of nearest neighbors to use *k*. In the original SMOTE paper, Chawla et al. used the 5 nearest neighbors and randomly selected between 1 and 5 of those nearest neighbors to use for SMOTE-ing depending upon the amount of oversampling desired (Chawla, Bowyer, Hall, & Kegelmeyer 2002). In the implementation in this paper, the *k*=5 nearest minority class neighbors were identified for each minority class observation, and a bootstrap sample of size *m*=100 neighbors was drawn for each minority class observation.

SMOTE + TOMEK LINK UNDERSAMPLING

SMOTE + Tomek link undersampling (SMOTE+Tomek) is a combination preprocessing method whereby the two aforementioned algorithms (see SMOTE and Tomek link Undersampling sections) are applied back-to-back. SMOTE is applied first in order to generate synthetic minority class examples, and, subsequently, Tomek link undersampling is applied to the dataset composed of both the original and new, synthetic observations. While full Tomek link removal has been very successful when applied as a secondary preprocessing measure after balancing datasets with SMOTE (He & Garcia 2009; Batista, Prati, & Monard 2004; More 2016), Tomek link undersampling following synthetic minority example generation does not appear to have been explored. It is plausible that this combination could yield higher recall than full Tomek link removal, though likely at the expense of precision.

DATA

The dataset for this analysis was obtained from Kaggle. There were a total of 284,807 transactions that were recorded in two days by a confidential credit card company in 2013. Out of the total number of transactions, there were only 492 fraud cases, which illustrates the highly unbalanced nature of credit card fraud data. Due to confidentiality, the dataset provider performed a dimension reduction transformation (principal components analyses) on the true, unknown input variables resulting in the quantitative variables *V1* through *V28*. Variables *Time* (seconds elapsed between row transaction and first row transaction) and *Amount* (transaction amount) remained unchanged. The target binary variable was *Class* (1 = fraud, 0 = non-fraud). For this analysis, the *Class* of interest was fraud.

DATA CLEANING/VALIDATION

The credit card fraud dataset was already exceptionally clean. It had only numerical input variables, no missing values, and the principal components analysis, which had been applied for confidentiality reasons, yielded a largely homogenous variable set. A PROC FREQ procedure identified 1,854 duplicate observations with identical values for *V1* through *V28*, *Amount*, and *Time*. The 1,854 duplicates were removed from the dataset (19 of the removed cases were of the minority class), which resulted in a new total of 282,953 observations. Next, a PROC STANDARD procedure was completed to standardize the variables above.

A PROC SURVEYSELECT procedure generated a 70/30 split (stratified by *Class*) from the cleaned dataset. Moreover, 70% of the observations were used as the baseline training dataset, while the remaining 30% comprised the validation dataset (see Data Preparation section). Three further training datasets were created by applying SMOTE, Tomek, and SMOTE+Tomek to the original training dataset. Table 1 displays the distribution of classes after each pre-processing algorithm was applied.

ANALYSIS

METHODOLOGY

The standalone Tomek link undersampling (Tomek), SMOTE, and SMOTE + Tomek link undersampling (SMOTE+Tomek) algorithms as described in the Tomek link Undersampling Algorithm and SMOTE Algorithm sections were implemented in SAS® and applied to copies of the cleaned training dataset (the commented code can be found in the SAS® Code). Having successfully accomplished the primary goal, the next step was to evaluate the efficacy of the different preprocessing methods on the credit card dataset.

Using SAS® Enterprise Miner™, four separate classifiers – a random forest, a neural network, a support vector machine, and a rule induction classifier – were learned on each of the four training datasets and validated on the validation dataset. A diverse group of classifiers was chosen to encourage learning-algorithm-independent conclusions. Since the pre-processing of the training datasets was completed in SAS®, a single datasource node could not be used in SAS® Enterprise Miner™. Instead, all four training datasets were imported separately along with the validation dataset.

SAS® Enterprise Miner™ Workflow (see Figure 1):

1. A Training dataset was imported and Role was set to Train.
2. The Validation dataset was imported and Role was set to Validate.
3. The Training and Validation datasets pass through an HP Forest, HP Neural, HP SVM, and Rule Induction node (default settings).
4. Steps 1 through 3 were repeated for each of the remaining training datasets.
5. All classifiers pass through an overall Model Comparison node to compare model performance on the Validation dataset.

From the Model Comparison node and Model Results windows, information on the area under their ROC Curves (ROC Index), precision, recall, number of false positives, number of true positives, number of false negatives, and number of true negatives was obtained for each model/dataset combination and summarized in Table 2.

RESULTS

The performance of each preprocessing method on a variety of metrics was compared side-by-side for each classifier (see Table 2). Performance was assessed using the area under the ROC curve (ROC index), precision, recall, and the ratio of false negatives to false positives. ROC curves were also generated for each combination of preprocessing method and classifier and grouped by classifier (see Figure 2-5). From the ROC curves, it is clear the SMOTE and SMOTE+Tomek provided the best performance over a wide range of threshold values. Tomek alone appears to provide little improvement over no preprocessing. The same story is told more succinctly by the ROC index (see Table 2). SMOTE+Tomek and SMOTE were consistently the best, and Tomek alone performed similarly to no preprocessing.

All of the preprocessing techniques generally improved recall at the expense of precision with the exception of Tomek for the rule induction classifier, which greatly increased precision with little loss in recall. The most marked increase in performance was obtained by SMOTE and SMOTE+Tomek for the support vector machine. There were so few fraud examples in the dataset prior to SMOTE-ing that the classifier simply labeled everything as non-fraud. SMOTE and SMOTE+Tomek allowed this classifier to be competitive with the others.

Although the loss in precision was generally numerically greater than the gain in recall, in imbalanced classification problems, recall is usually more important since false negatives are frequently more costly than false positives. The false negative to false positive ratio (FN/FP) was included to illustrate the relative improvement in this domain. In every case except Tomek for rule induction, the preprocessing techniques drastically shifted the bias of the classifier in favor of the minority class (reduced the FN/FP ratio). For the random forest with no preprocessing, there were 5.5 false negatives for every false positive, and SMOTE+Tomek reduced the ratio to 0.74. Even standalone Tomek, which appeared to be ineffective as judged by the other metrics yielded significant improvement, dropping the ratio from 5.5 to 2.91. It is important to note that this metric provides a measure of bias, and lower is not always better. The ideal ratio depends upon the relative misclassification costs, and, even then, a more unbalanced ratio may be preferred if a lower overall cost is achieved.

GENERALIZATION

Overall, SMOTE and SMOTE+Tomek were the most effective preprocessing methods for improving the detection of fraudulent transactions in the credit card dataset. Both methods improved recall and lowered the FN/FP ratio for every classifier, indicating improved sensitivity to fraud. At the same time, SMOTE and SMOTE+Tomek improved the ROC index, indicating an improved ability to distinguish fraud from non-fraud.

Although Tomek on its own was ineffective as judged by ROC index, the increase in recall and the drop in FN/FP ratio for the random forest and neural net suggest that Tomek somewhat reduced the influence of nonfraud cases along the class boundaries on the induced classifiers as intended. Interestingly, Tomek was very effective at boosting precision for the rule induction classifier both on its own and when applied after SMOTE, and recall was only minimally affected. Though the authors are not entirely sure why this occurred, it may have something to do with the fact that this classifier was specifically designed to learn rare classes and clearing out the border regions allowed the classifier to more accurately learn the idiosyncrasies of the fraud cases.

SUGGESTION FOR FUTURE STUDIES

Although reasonable results were obtained with the utilized methodologies, there were some limitations and possible avenues for improvement:

- The Tomek link undersampling methodology used here removes only the majority class neighbor; this tactic is potentially very sensitive to noise. It may be beneficial to investigate two-sided removal (full Tomek link removal), at least in cases with less class imbalance.
- Cross-validation could not only allow for further tuning of model hyperparameters, but could also offer the potential to develop confidence intervals for the performance metric(s) for a given methodology.
- The space of possible hyperparameter values was limited since the default settings were used for each model. Exploring a wider and/or finer hyperparameter space could improve results or potentially lead to different conclusions regarding methodology.

- The conclusions in this analysis are applicable to the dataset used, which features a dimension reduction on many of the predictor variables. Future research could improve heuristic power by using more real-world datasets and a variety of class imbalances.
- The generation of the synthetic minority examples was based on randomly generating numbers between 0-1 from a uniform distribution. Perhaps a Bayesian argument could be made for accounting for prior information and using different distributions for random number generation.
- The area under the Precision/Recall curve is a more appropriate model evaluation metric, since fraud detection data mining tasks are more interested in correctly identifying true positive cases and minimizing false negative cases.
    - A manual method would be required since SAS® Enterprise Miner™ lacks this feature.

CONCLUSIONS

Both SMOTE and SMOTE followed by Tomek link undersampling are effective methods for improving the performance of classifiers on datasets with underrepresented minority classes. An analyst dealing with highly imbalanced data may be able to improve accuracy on the minority class(es) with a judicious application of SMOTE and/or Tomek link undersampling. SMOTE can improve recall by expanding and generalizing the minority class boundaries, while Tomek link undersampling can improve recall by shifting the classifier bias in favor of the minority class. Since only one dataset was used, it is difficult to say to what degree these results will generalize. However, it is reasonable to believe these results may hold for classifiers induced over datasets with similar characteristics (e.g. dimensionality, degree of imbalance, class distributions).

Though a fair amount of research had already been done on the efficacy of SMOTE, Tomek link undersampling, and combinations of SMOTE and full Tomek link removal, SMOTE followed by Tomek link undersampling does not appear to have been explored until now. Additionally, SAS® code for Tomek link removal or Tomek link undersampling – if it existed – was not publicly available. This project has helped to fill this gap and serves as a foundation for future research.

As a final note, the random forest algorithm consistently retained reasonably high precision after the application of SMOTE and SMOTE+Tomek (see Table 2). Even though the random forest with SMOTE+Tomek attained one of the lowest FN counts, the number of FP was not greatly inflated. Thus, this classifier/pre-processing method combination may be a good starting point when developing predictive models learned on imbalanced datasets.

APPENDIX

TABLE OF CONTENTS

Figures

FIGURES

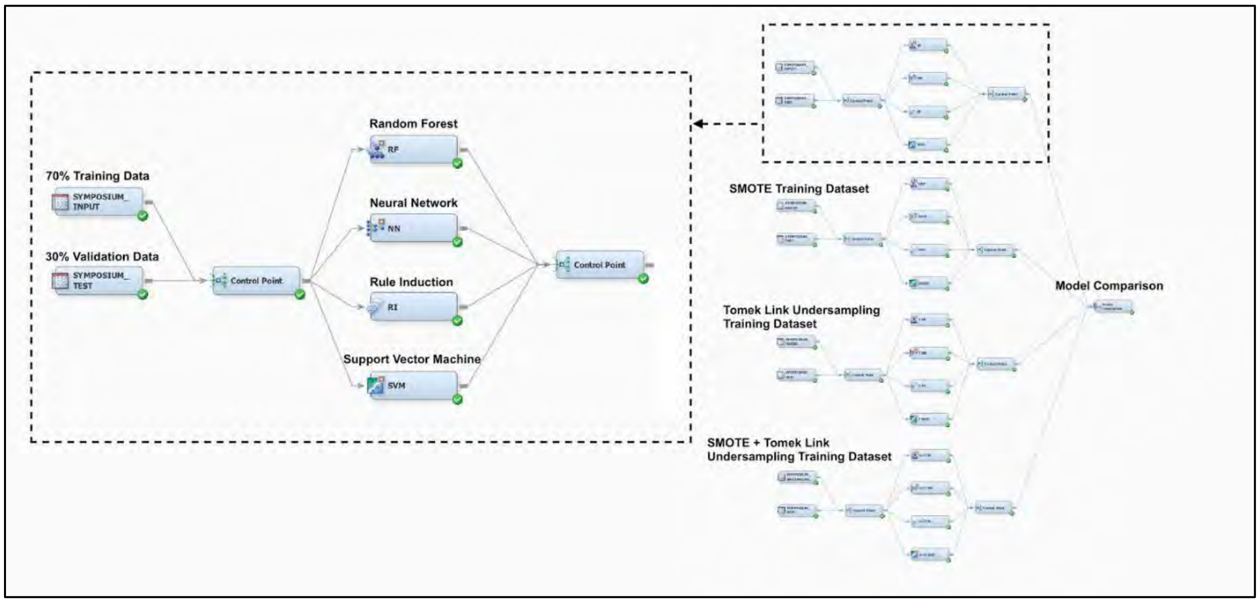


Figure 1. SAS® Enterprise Miner™ Workflow

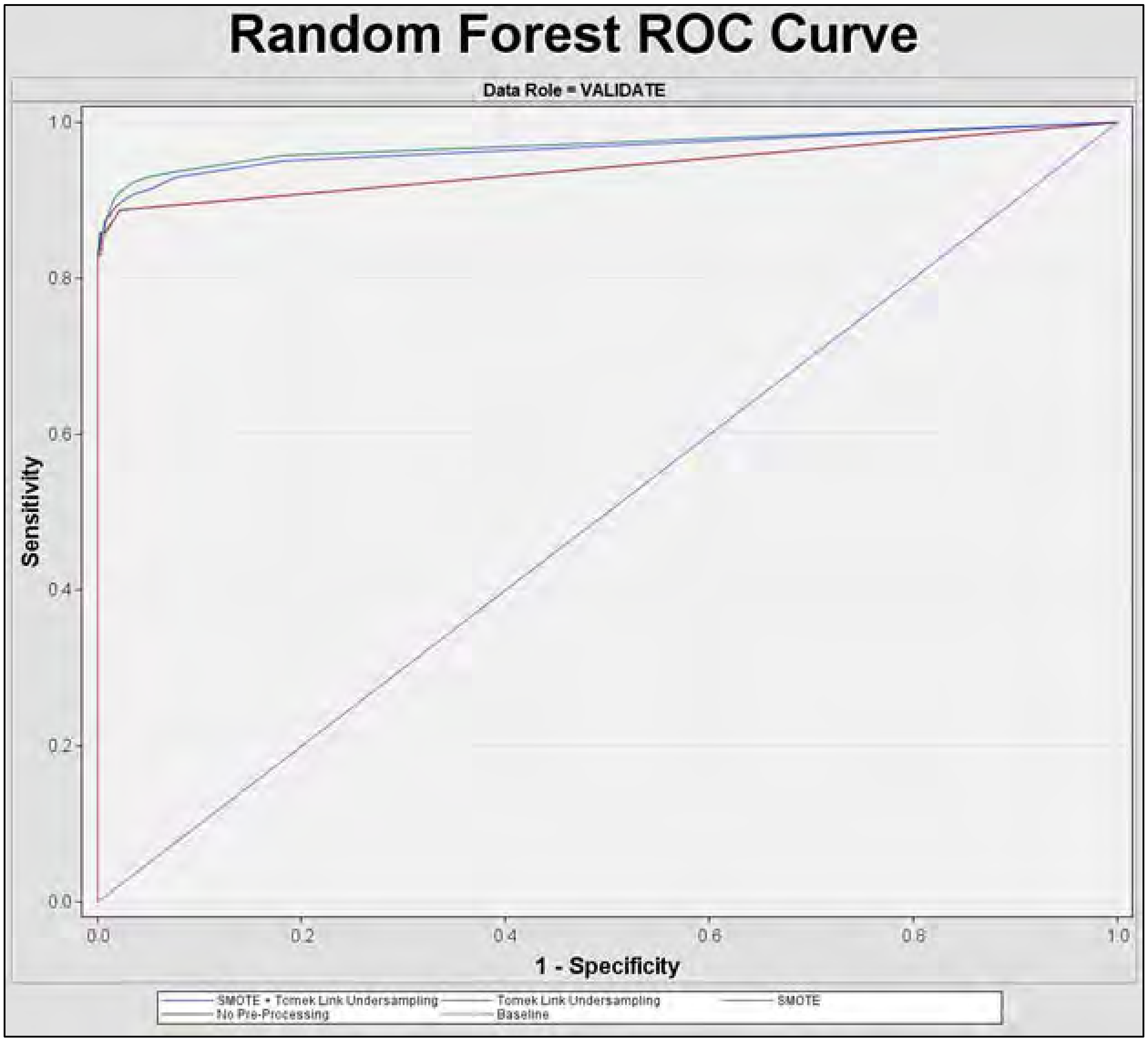


Figure 2. Random Forest ROC Curve

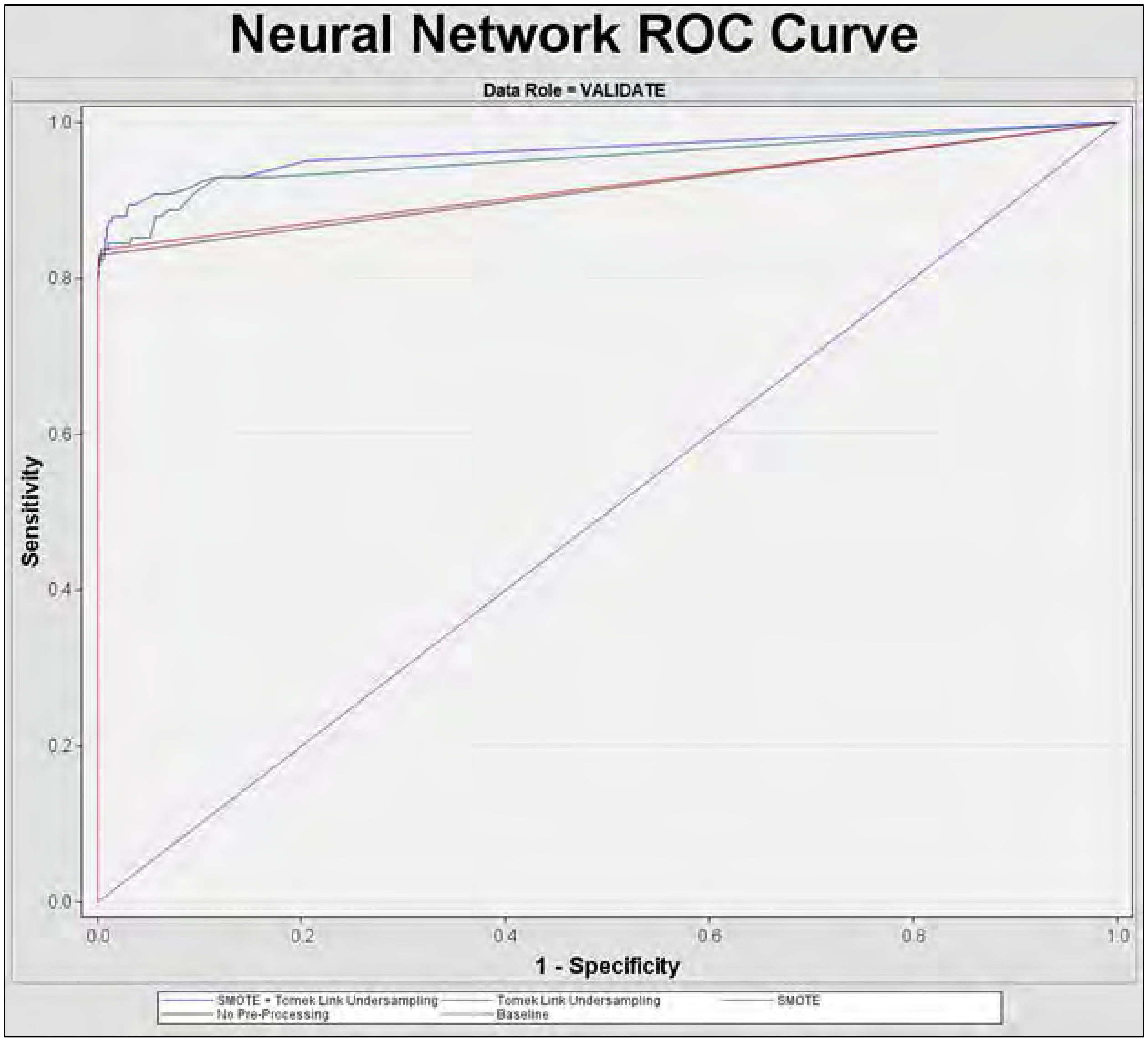


Figure 3. Neural Network ROC Curves

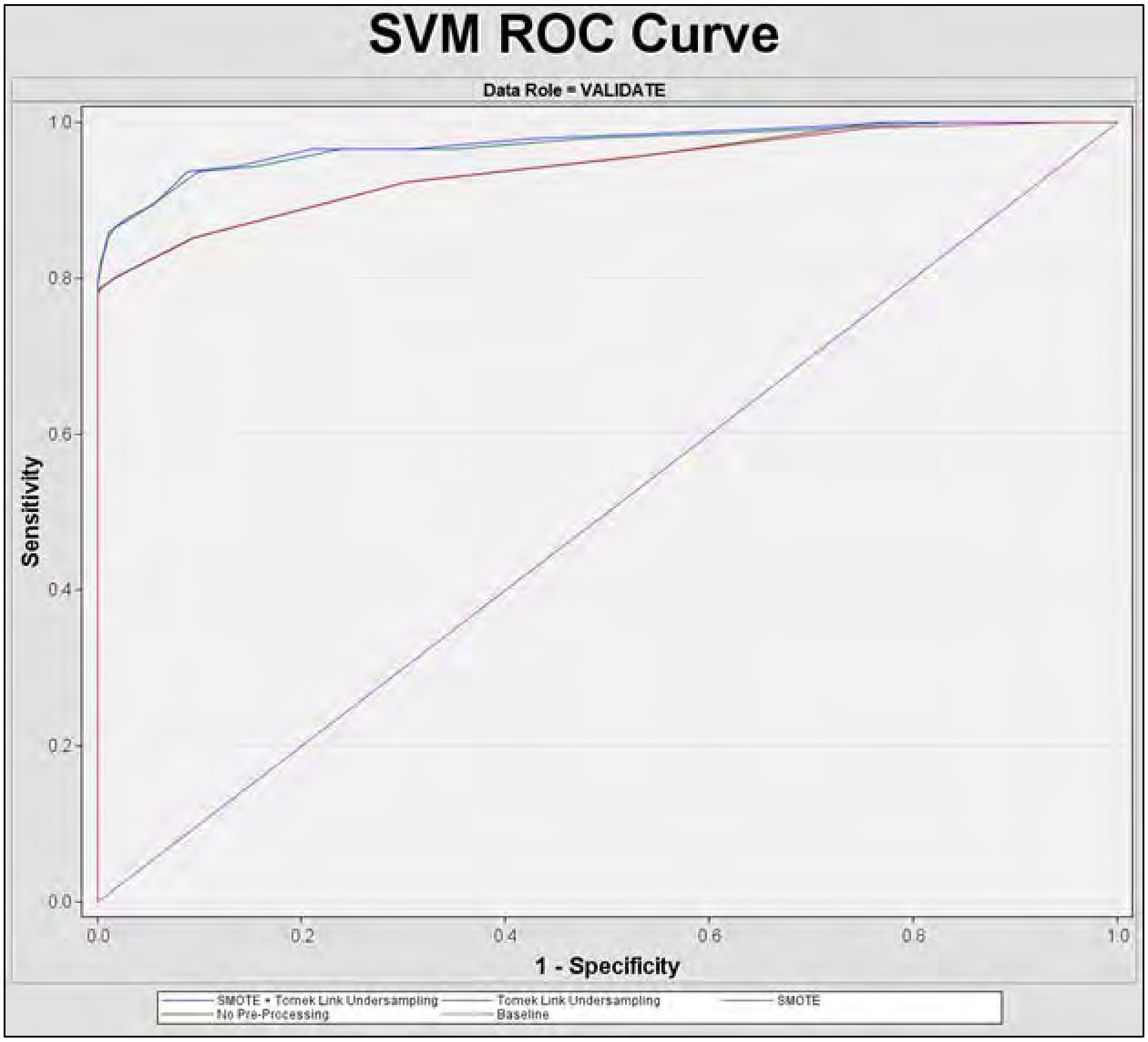


Figure 4. Support Vector Machine ROC Curve

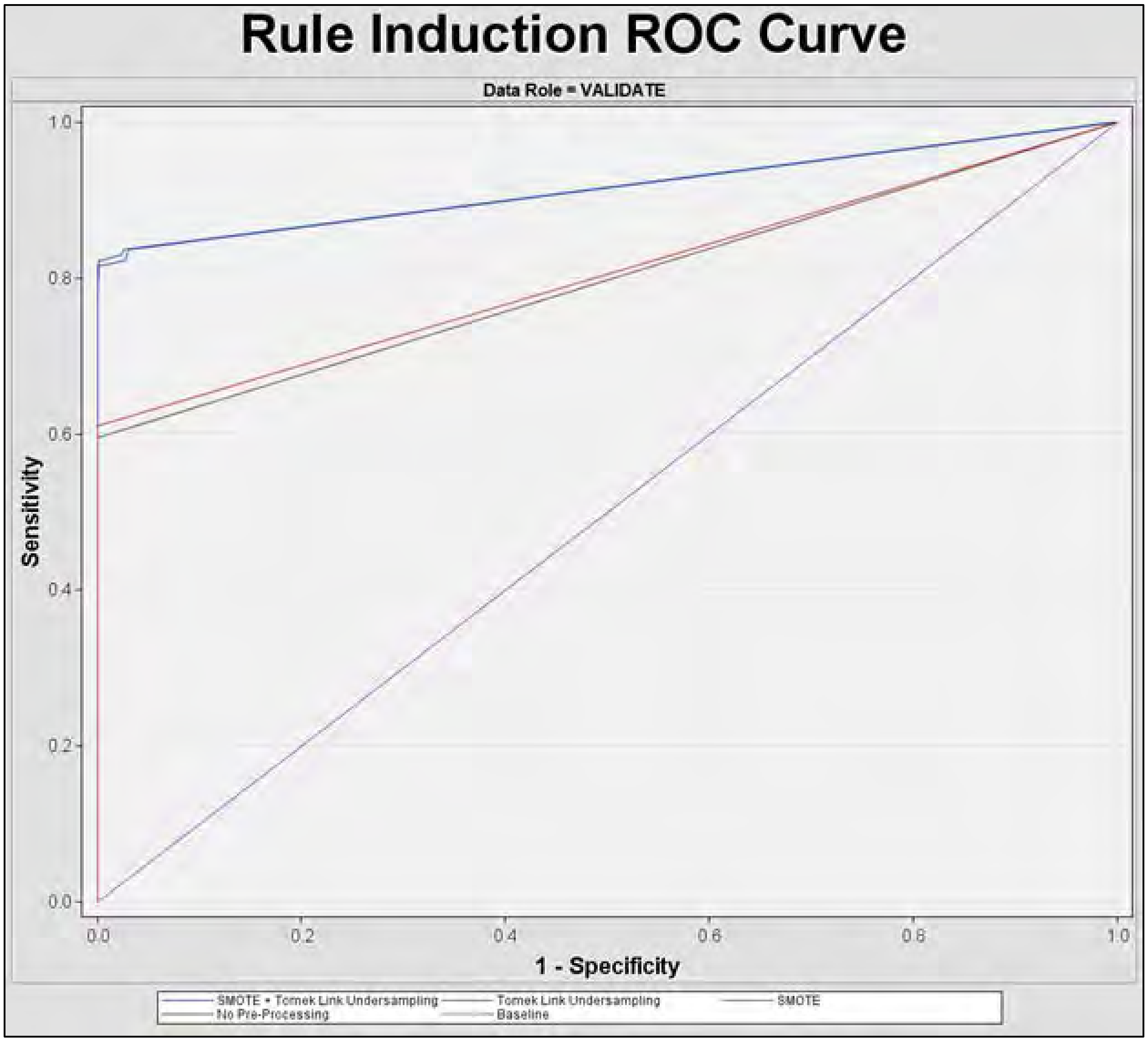
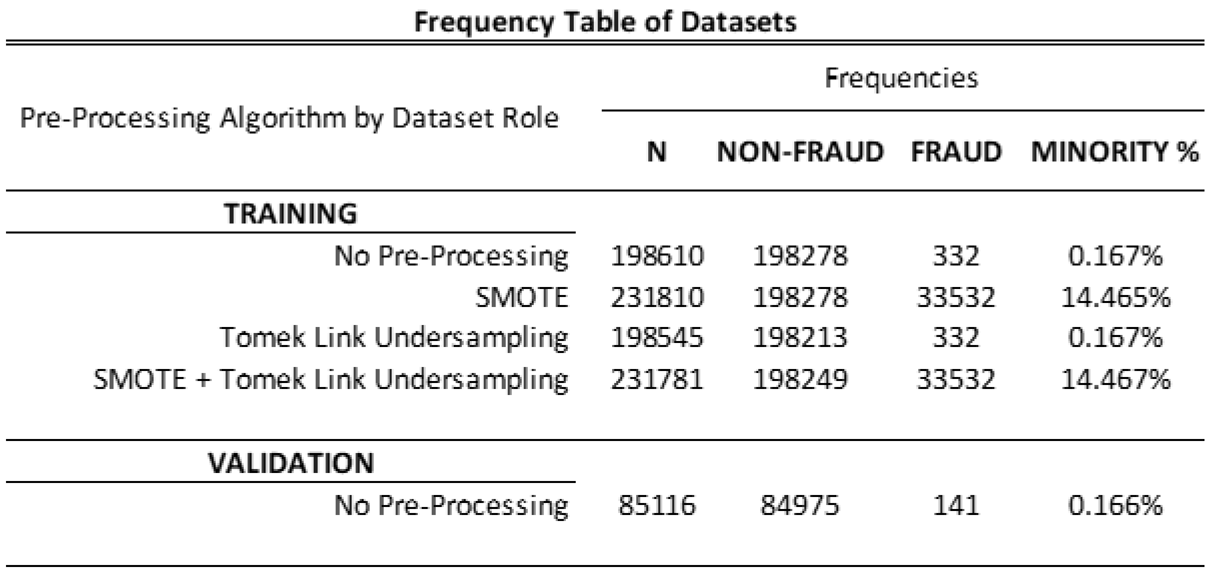


Figure 5. Rule Induction ROC Curve

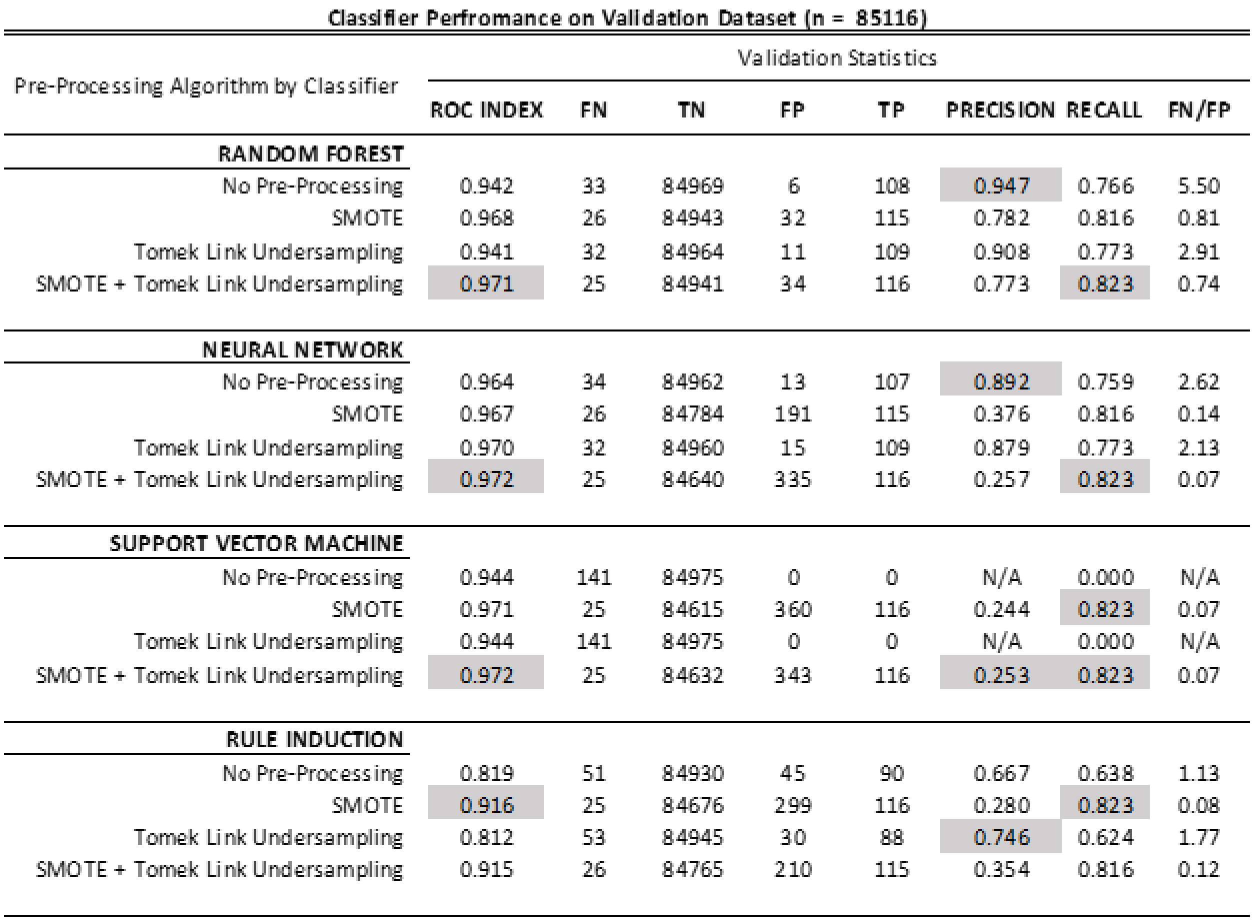
TABLES



Frequency Table of Datasets

Pre-Processing Algorithm by Dataset Role	Frequencies			
	N	**NON-FRAUD**	**FRAUD**	**MINORITY %**
TRAINING				
No Pre-Processing	198610	198278	332	0.167%
SMOTE	231810	198278	33532	14.465%
Tomek Link Undersampling	198545	198213	332	0.167%
SMOTE + Tomek Link Undersampling	231781	198249	33532	14.467%
VALIDATION				
No Pre-Processing	85116	84975	141	0.166%

Table 1. Frequency Table of Datasets Used for Analysis



Classifier Perfromance on Validation Dataset (n = 85116)

Pre-Processing Algorithm by Classifier	Validation Statistics							
	ROC INDEX	FN	TN	FP	TP	PRECISION	RECALL	FN/FP
RANDOM FOREST								
No Pre-Processing	0.942	33	84969	6	108	0.947	0.766	5.50
SMOTE	0.968	26	84943	32	115	0.782	0.816	0.81
Tomek Link Undersampling	0.941	32	84964	11	109	0.908	0.773	2.91
SMOTE + Tomek Link Undersampling	0.971	25	84941	34	116	0.773	0.823	0.74
NEURAL NETWORK								
No Pre-Processing	0.964	34	84962	13	107	0.892	0.759	2.62
SMOTE	0.967	26	84784	191	115	0.376	0.816	0.14
Tomek Link Undersampling	0.970	32	84960	15	109	0.879	0.773	2.13
SMOTE + Tomek Link Undersampling	0.972	25	84640	335	116	0.257	0.823	0.07
SUPPORT VECTOR MACHINE								
No Pre-Processing	0.944	141	84975	0	0	N/A	0.000	N/A
SMOTE	0.971	25	84615	360	116	0.244	0.823	0.07
Tomek Link Undersampling	0.944	141	84975	0	0	N/A	0.000	N/A
SMOTE + Tomek Link Undersampling	0.972	25	84632	343	116	0.253	0.823	0.07
RULE INDUCTION								
No Pre-Processing	0.819	51	84930	45	90	0.667	0.638	1.13
SMOTE	0.916	25	84676	299	116	0.280	0.823	0.08
Tomek Link Undersampling	0.812	53	84945	30	88	0.746	0.624	1.77
SMOTE + Tomek Link Undersampling	0.915	26	84765	210	115	0.354	0.816	0.12

Table 2. Modeling Results

SAS® CODE

Data Preparation

```
/**************************************************************************************************************/
/* PRELIMINARIES                                                                                              */
/**************************************************************************************************************/
/*In this section, the credit card data (soteam2.creditcard_cleaned) is read in from the team folder, and
the variables are standardized. Next the observations in the standardized data are randomly split into two
subsets (stratified by "target") - "input" or "test". "test" is set aside to be used as a validation set later.
"input" will be used for training (either before or after preprocessing).
*/

*Creating a copy of the original credit card data with a two new variables: target (to replace class), and
obs_id to specify a unique id variable for each observation;
data creditcard (drop=class);
         set soteam2.creditcard_cleaned;
         obs_id = _n_; *required to specify the unique id variable of each observation;
         if class = "0" then target = 0; *matching format of the target variable (for some reason class was formatted
as character variable);
         else if class = "1" then target = 1;
run;

*Creating a character string of all of the predictor variables;
%let var_ = v1-v28 amount time;

*Standardizing the predictor variables;
proc standard data=creditcard mean=0 std=1 out=zcreditcard;
         var &var_ ;
run;

*Sorting the standardized credit card data by target before running proc surveyselect;
proc sort data = zcreditcard;
         by target;
run;

*Randomly selecting (samprate*100)% of the observations stratified by "target";
*The output dataset - sample - contains a new variable - selected - that is a 1 if the observation
 was sampled;
proc surveyselect data = zcreditcard out = sample method = srs samprate = 0.7 seed = 12345 outall;
         strata target;
run;

*sample is split into two datasets - "input" and "test" - where "input" contains observations with
"selected"=1 and "test" contains the others;
data input test;
         set sample;
         if selected = 1 then output input;
         else output test;
run;

/*
*Saving the test dataset;
data soteam2.symposium_test (keep=obs_id target &var_);
         set test;
run;

*Saving the input dataset;
data soteam2.symposium_input (keep=obs_id target &var_);
         set input;
run;
*/
```

SMOTE

```
/*****************************************************************************************************************/
/* SMOTE                                                                                                        */
/*****************************************************************************************************************/

* Loads the cleaned input dataset;
data input; set soteam2.symposium_input; run;

*Creates a character string of all of the predictor variables;
%let var_ = v1-v28 amount time;

*Closes all open ODS output destinations;
ods _all_ close;

*Creates a dataset - fraud - that contains only the observations in input where target=1;
data fraud;
    set input;
    where target=1;
run;

/*****************************************************************************************************************/
/* The purpose of this sub-section is to create a dataset - neighbor_id2 - that contains a list of
observation/neighbor pairs to be used for SMOTEing. For each observation in fraud, &Mult pairs of that
observation with one of its &NN-1 nearest neighbors will be generated. The neighbor used for each pair for each
observation is chosen randomly from the &NN-1 nearest neighbors.
*/

*Creates a macro variable - NN - that is the number of k nearest neighbors to SMOTE +1;
%Let NN = 6;

*Creates a macro variable - Mult - that is the value of the SMOTE multiplier - the number of additional
minority class instances desired for each of the original minority class examples;
%Let Mult = 100;

*Creates a dataset - nTest - with the dk-1 nearest neighbors(target=1) of each observation where target=1.
The ID of and distance to the dk-1 nearest neighbors are given by the "Nbor" and "Distance" variables;
proc modeclus data = fraud dk=&NN neighbor out = modeclus_out;
         var &var_: ; /*Specify the input numeric variables here*/
         id obs_id ; /*Specify the unique id variable of each observation*/
         ods output Neighbor = nTest; /*output the nearest neighbors' id along with their distance*/
run;

* Opens the Listing destination;
ods listing;

*Creates a dataset - neighbor_id - of each pair of target=1 observations and dk-1 nearest neighbors (also target=1);
*Essentially modifying nTest so that all blank IDs are filled with the appropriate ID (now called center_id);
*Also, an index variable is added, and every variable except "center_id", "nbor", and "index" is dropped;
data neighbor_id (keep=center_id nbor index);
         set ntest;
         retain center_id '000000000000';
         if not missing(id) then center_ID = put(id,$12.);
         else center_id = center_id;
         index = _N_; /*prepare id for the new cases*/
run;

*Transposes the neighbor_id dataset and stores this as neighbor_trans;
*Essentially creates a list of center_ids with each of the 5 nearest neighbors listed as a variable (in a column);
*The other superfluous columns/variables are dropped in the following data step;
proc transpose data = neighbor_id out= neighbor_trans;
         by center_id;
         var nbor;
run;

*Removes unwanted columns/variables from neighbor_trans;
data neighbor_trans (drop= _name_ _label_ col6);
         set neighbor_trans;
run;

*Creates neighbor_id2 containing a list of observation/neighbor pairs to be used for SMOTEing. For each observation
in fraud, &Mult pairs of that observation with one of its &NN-1 nearest neighbors will be generated. The neighbor
used for each pair for each observation is chosen randomly from the &NN-1 nearest neighbors;
data neighbor_id2 (keep = index center_id nbor);
         set neighbor_trans;
         do i = 1 to &Mult;
         random = rand("uniform");
         if random < .2 then nbor = col1;
         else if random < .4 then nbor = col2;
         else if random < .6 then nbor = col3;
         else if random < .8 then nbor = col4;
         else nbor = col5;
         output;
         end;
run;

*Adds an index variable to neighbor_id2;
```

```
data neighbor_id2;
         set neighbor_id2;
         index = _n_;
run;

/*****************************************************************************************************************/
/* The purpose of this sub-section is to create a dataset - raw2 - that contains the new cases generated by
SMOTEing.
*/

*Transposes the "fraud" dataset and outputs it as "raw1". The observations become the variables, and the variables
become the observations;
proc transpose data = fraud out=work.raw1 prefix=ID_;
         var &var_:;
         id obs_id ;
run;

*Creates new synthetic cases for target=1 and stores the new observations as columns appended onto raw1 (the
transposed version of the fraud dataset). Only the new synthetic fraud observation columns are kept, and the
resulting dataset is stored as raw2. The number of new cases is equal to the number of observations in the
neighbor_id2 dataset or, equivalently, the number of observations in the "fraud" dataset multiplied by &Mult;
data _NULL_;
         set neighbor_id2 end=eof;
         if not eof then do;
         call execute('DATA raw1; ');
         call execute(  'set raw1 ;');
         call execute(  "_R_=rand('uniform');");
         call execute(  'new_'||trim(left((index)))|| ' = (id_'||trim(left((nbor)))||' -
                           id_'||trim(left((center_id)))||') * _R_ ');
         call execute(  ' + ID_'||trim(left((center_id)))||' ;');
         call execute('run;');
         end;

         if eof then do;
         call execute('DATA raw1; ');
         call execute(  'set raw1 ;');
         call execute(  "_R_=rand('uniform');");
         call execute(  'new_'||trim(left((index)))|| ' = (id_'||trim(left((nbor)))||' -
                           id_'||trim(left((center_id)))||') * _R_ ');
         call execute(  ' + ID_'||trim(left((center_id)))||' ;');
         call execute('run;');

         call execute('data work.raw2 (keep=_NAME_ new:); set raw1; run; ');
         end;
run;

*Transposes "raw2" to obtain a dataset of the new cases called "new_cases", which has the same structure as the
original dataset;
proc transpose data = work.raw2 out = new_cases;
         id _NAME_;
         var _NUMERIC_;
run;

/*****************************************************************************************************************/
/* COMBINING NEW CASES WITH ORIGINAL FRAUD AND SAVING
OUTPUT                                                                                                    */
/*****************************************************************************************************************/

data new (keep=&var_);
         set new_cases;
run;

data old (keep=&var_);
         set fraud;
run;

data newfraud (keep=target &var_);
         set old new;
         target = 1;
run;

data nonfraud (keep=target &var_);
         set input (where=(target=0));
run;

data soteam2.Symposium_SMOTE;
         set newfraud nonfraud;
run;

/*
data Symposium_SMOTE;
         set newfraud nonfraud;
run;
*/
```

Tomek Link Undersampling

```
/*****************************************************************************************************************/
/* TOMEK                                                                                                         */
/*****************************************************************************************************************/

* Loads the cleaned input dataset;
data input; set soteam2.symposium_input; run;

*Creates a character string of all of the predictor variables;
%let var_ = v1-v28 amount time;

*Closes all open ODS output destinations;
ods _all_ close;

*Creates a dataset - fraud - that contains only the observations in input where target=1;
data fraud;
         set input;
         where target=1;
run;

/*****************************************************************************************************************/
/* The purpose of this sub-section is to create a dataset - neighbor_id - that contains a list of
observation/nearest-neighbor pairs for each observation in the input dataset. For each observation, the
nearest neighbor is found and the neighbor's ID is given by the "Nbor" variable.
*/

proc modeclus data = input dk=2 neighbor out = modeclus_out;
         var &var_: ; /*Specify the input numeric variables here*/
         id obs_id ; /*Specify the unique id variable of each observation*/
         ods output Neighbor = nTest; /*output the nearest neighbors' id along with their distance*/
run;

* Opens the Listing destination;
ods listing;

*Creates a dataset -neighbor_id - that is essentially a list of observation IDs (given by "obs_id") and the
IDs of their nearest neighbors (given by "neighbor"). The input functions are used to convert id and nbor
into numerics;

data neighbor_id (keep=obs_id neighbor);
         set nTest;
         obs_id = input(id,12.);
         neighbor = input(nbor,12.);
run;

/*****************************************************************************************************************/
/* The purpose of this sub-section is to identify Tomek links (pairs of nearest neighbors with differing class
labels) and, for all Tomek link pairs, to remove the majority class (non-fraud) observation from the input
dataset.
*/

*Sorting fraud and neighbor_id by obs_id to facilitate merging;
proc sort data=fraud;
         by obs_id;
proc sort data=neighbor_id;
         by obs_id;
run;

*Creating fraud_nbor, which is just the fraud dataset with a new column - neighbor - containing the nearest
neighbor for each observation;
data fraud_nbor;
         merge fraud neighbor_id;
         by obs_id;
         if target = 1;
run;

/* Initializing a new column called neighbor_nn */
data neighbor_id_new;
    set neighbor_id;
    neighbor_nn = .;
run;

/* proc sql to populate neighbor_nn */
/* neighbor_nn is the nearest neighbor for the neighbor column */
proc sql;
update neighbor_id_new as a
    set neighbor_nn=(select neighbor from neighbor_id as b
        where a.neighbor=b.obs_id);
quit;
run;

/* additional if statement only outputs if obs_id = neighbor_nn */
/* if obs_id = neighbor_nn, that means they are potential tomek links */

data fraud_nbor;
    merge fraud neighbor_id_new;
    by obs_id;
    if target = 1;
    if obs_id= neighbor_nn;
run;

* Removing majority class members of Tomek link pairs;
proc sql;
    delete from input
    where obs_id in (select neighbor from fraud_nbor) & target = 0;
quit;
run;


/*****************************************************************************************************************/
/* SAVING FINAL DATASET                                                                                         */
/*****************************************************************************************************************/

data soteam2.Symposium_Tomek;
    set input;
run;
```

REFERENCES

Batista, G., Prati, R.C., and Monard, M.C. 2004. “A study of the behavior of several methods for balancing machine learning training data.” *ACM SIGKDD Explorations Newsletter*, 6:20–29.

Chawla, N., Bowyer, K.W., Hall, L.O., and Kegelmeyer, W.P. 2002. “SMOTE: Synthetic Minority Over-sampling Technique." *Journal of Artificial Intelligence and Research*, 16:321–357.

He, H. and Garcia, E.A., 2009. “Learning from Imbalanced Data.” *IEEE Transactions on Knowledge and Data Engineering*, 21:1263–1284.

He, H. and Ma, Y. 2013. *Imbalanced Learning: Foundations, Algorithms, and Applications.* 1st ed. Hoboken, NJ: John Wiley & Sons.

Kubat, M. and Matwin, S. 1997. “Addressing the Curse of Imbalanced Training Sets: One-Sided Selection.” ICML. 179–186.

Kubat M., Holte R.C., and Matwin S. 1998. “Machine Learning for the Detection of Oil Spills in Satellite Radar Images.” *Machine Learning*, 30:195–215.

More, A. 2016. “Survey of resampling techniques for improving classification performance in unbalanced datasets.” *CORR*, abs/1608.06048.

Tomek, I. 1976. “Two modifications of CNN.” *IEEE Transaction on Systems Man and Communications*, 6: 769–772.

Wang, R., Lee, N., and Wei, Y. 2015. “A Case Study: Improve Classification of Rare Events with SAS® Enterprise Miner™ .” *SAS Institute Inc.*, Paper 3282–2015.

ACKNOWLEDGEMENTS

The authors of this paper would like to acknowledge Dr. Xuelei Ni for supervising this project.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Kyle Biron
Kennesaw State University
(404) 403-9103
kylejbiron@gmail.com

Jonathan Boardman
Kennesaw State University
(770) 401-2812
jwb220@gmail.com

Ryan Rimbey
Kennesaw State University
(412) 974-7075
ryrimbey@yahoo.com