Identifying Classification Algorithms Most Suitable for Imbalanced Data

Ray Marie Tischio, Gary M. Weiss
Dept. of Computer & Info. Science
Fordham University
Bronx, New York, USA
{rtischio, gaweiss}@fordham.edu

***Abstract*—Many real-world data sets have significant levels of class imbalance, which can impact the performance of the induced classifiers. In such cases, classification algorithms may produce models that have high accuracy but fail to perform well on examples belonging to the minority class. Many techniques have been developed to address class imbalance, including sampling and cost-sensitive learning—but each of these has drawbacks. One approach that has not been extensively studied is to utilize classification algorithms that natively perform well in the presence of class imbalance. The research study described in this paper comprehensively evaluates the degree to which different algorithms are impacted by class imbalance, with the goal of identifying the algorithms that perform best and worst on imbalanced data. In particular, this study assesses the relative impact of class imbalance on thirteen different algorithms as they are applied to twenty-nine data sets with varying levels of class imbalance. The results from this study demonstrate that certain classification algorithms, such as decision trees, Adaboost, and gradient boosting, perform very well in the presence of class imbalance, while others algorithms, such as logistic regression, support vector machines, and Bernoulli Naïve Bayes, perform very poorly.**

Keywords—classification, class imbalance, classifier performance, machine learning,

I. INTRODUCTION

Real world applications often are associated with data sets with class imbalance. For example, predicting fraudulent credit-card transactions, identifying telecommunication failures, detecting oil spills from satellite images, and medical diagnosis all involve class imbalance. In applications such as these, the minority class is typically of primary importance, but classification methods will generally optimize performance of the more frequently occurring classes.

The issue of class imbalance has received a great deal of attention, and researchers and practitioners utilize a variety of techniques to improve performance of examples belonging to the less frequently occurring classes. The most common methods involve under-sampling the majority class [10] and/or oversampling the minority class, in order to form a more balanced training set. Cost-sensitive learning is also used sometimes, since one can improve the performance on the minority classes by assigning a higher relative cost to misclassifying examples belonging to these classes.

The goal of the study described in this paper is to determine which classification algorithms naturally perform well—or poorly—on imbalanced data. There are some reasons to believe that not all algorithms will be affected equally. Weiss [16] suggested that boosting methods may perform well at classifying minority examples because boosting places more emphasis on misclassified examples—which are more likely to come from the minority classes. Drummond and Holte [5] determined that decision tree splitting criteria are relatively cost insensitive and hence would make them perform well for varying misclassification costs; this also means they will perform well in the presence of class imbalance because of the relationship between class imbalance and non-uniform misclassification costs. Interestingly, our results show that boosting methods and decision tree methods tend to perform relatively well in the presence of class imbalance.

This article will analyze the performance of thirteen diverse classification algorithms on twenty-nine imbalanced data sets, to determine the relative performance of these algorithms on imbalanced data. Performance will be reported using accuracy, F1-measure, and AUC score, but our analysis will focus on the latter two metrics since they are much more appropriate in the presence of class imbalance.

II. EXPERIMENT METHODOLOGY

The experiments described in this paper will determine the relative effectiveness of different classification algorithms in the presence of class imbalance. This section will describe the twenty-nine data sets, the thirteen classification algorithms, the structure of the experiments, and the metrics used to assess classifier performance.

A. Data Sets

The twenty-nine data sets employed in this study are described in Table 1. Eight data sets (*occ*, *c4*, *seism*, *lett9*, *lett25*, *redw*, *choc*, *whw*) are from the UCI Machine Learning Repository [6], four data sets (*hmeq* [15], *prom* [13], *findis* [7], *train* [17]) are from Kaggle, and the remaining data sets are from the KEEL Dataset Repository [1].

The first two columns in Table 1 specify the abbreviated and full data set names. The next column specifies the total number of attributes, including the class. Our experiments only utilize data sets with two classes, so the data sets with more than two classes were mapped into two classes as speci-

fied by the "Map" field. The first (second) value specifies the number of classes mapped into the majority (minority) class. Then the "Min %" field specifies the percentage of examples that belong to the minority class and the last field specifies the total number of examples in the data set. Table 1 is sorted based on increasing levels of class imbalance.

TABLE 1: DATA SET DESCRIPTION

Data Set	Full Name	Attr.	Map	Min %	Total Size
glass1	glass1	10		35.5	214
hbman	haberman	4		26.5	306
ecoli1	ecoli1	8		22.9	336
occ	occupancy	7		21.2	8,143
hmeq	hmeq	13		8.9	3,364
thy	thyroid1	6		16.3	215
glass6	glass6	10		13.6	214
yeast3	yeast3	9		11.0	1,484
ecoli3	ecoli3	8		10.4	336
pbloc	page-blocks0	11		10.2	5,472
c4	connect-4	43	2/1	9.5	67,556
vow0	vowel0	14		9.1	988
prom	promoted	8		6.9	25,000
seism	seismic-bumps	19		6.6	2,578
aba9	abalone9-18	9		5.7	731
car	car-vgood	7		3.8	1,728
lett9	letter-recognition	17	25/1	3.7	19,999
findis	financial distress	86		3.7	3,672
lett25	letter-recognition	17	25/1	3.7	19,999
yeast4	yeast4	9		3.4	1,484
yeast5	yeast5	9		3.0	1,482
choc	flavors_of_ cacao	9	10/3	2.8	1,795
train	train	18	2/1	2.9	31,378
yeast6	yeast6	9		2.4	1,484
redw	winequality-red	12	4/2	1.8	1,599
poker	poker8_vs_6	11		1.2	1,477
root	kddcup-rootkit-imap_vs_back	42		1.0	2,225
aba1	abalone19	9		0.8	4,174
whw	winequality-white	12	6/1	0.4	4,898

B. Classification Algorithms

The thirteen classification algorithms utilized in this study are listed in Table 2, along with their abbreviations. The implementations of these classifiers are from the *scikit-learn* machine learning library for the Python programming language [11], and the default settings are always used.

TABLE 2: CLASSIFIERS

LR	Logistic Regression
LDA	Linear Discriminant Analysis
QDA	Quadratic Discriminant Analysis
KNN	K-Nearest Neighbors
DT	Decision Tree
RF	Random Forest
SV	Support Vector
MLP	Multi-layer Perceptron
ADB	AdaBoost
GRB	GradientBoosting
GNB	Gaussian Naive Bayes
BNB	Bernoulli Naive Bayes
CNB	Complement Naive Bayes

LR is regularized logistic regression using the 'liblinear' library. LDA and QDA differ in that when using LDA, all classes share a common covariance matrix whereas the QDA implementation has a covariance matrix for each class. KNN uses the default value of 5 nearest neighbors. DT is an optimized version of the CART decision tree algorithm, and RF uses the default of 10 trees in the random forest. SV is a support vector machine implementation based on the *libsvm* library and uses the default radial basis kernel. The MLP neural network trains using backpropagation.

The next two classifiers in Table 2 are ensemble boosting methods: ADB implements the AdaBoost-SAMME method and uses the SAMME.R real boosting algorithm, and boosting is terminated after the default maximum number of estimators set to 50. GRB performs 100 boosting stages and the criterion function to measure quality of each split is set to mean squared error with improvement score by Friedman ('Friedman_mse'). For binary classification, a single regression tree is induced and in each stage fit on the negative gradient of the 'deviance' (default) loss function.

The final three classifiers in the table are variations of the Naive Bayes algorithm. GNB assumes a Gaussian distribution and continuous feature data, BNB assumes a binomial distribution, and CNB is an adaption of multinomial naive Bayes (which computes weights in terms of frequency) using the complement of each class to compute the model's weights. CNB is cited as being a particularly suitable option when dealing with imbalanced data. The CNB algorithm does not accept non-positive input for training. Therefore, for the six data sets with features with non-positive values, the feature values were scaled to a new value between 0 and 1. The number of scaled attributes for CNB for each of the six data set is as follows: *thy*(1), *vow0*(10), *prom*(4), *seism*(2), *finds*(34), and *choc*(2).

C. Evaluation Procedure

The data is partitioned into training and test sets using stratified 10-fold cross-validation, so that both sets maintain the same class distribution [9]. Categorical attributes are mapped to numerical values using label encoding.

Classifier performance is measured using accuracy, F1-measure, and the Area Under the ROC curve (AUC). Accuracy is included because it is a common metric, but it is not very meaningful in the presence of class imbalance [12], and hence our analysis focuses more on the other two metrics. The F1-measure and AUC values are asymmetric with respect to the class, and in this study, as is usual, the positive class refers to the minority class. The F1-measure represents the weighted harmonic mean of precision and recall, where in this context precision is the fraction of minority-class predictions that are correct, and recall is the fraction of minority class examples that are correctly classified. F1-score is defined as:

$$\text{F1-score} = 2 \times precision \times recall \,/\, (precision + recall) \quad (1)$$

The AUC metric is the area under the curve formed by plotting the True Positive Rate against the False Positive Rate. The AUC score is commonly used in studies of class imbalance because it does not depend to the specific class distribution and is insensitive to changes in class distribution.

III. RESULTS

The thirteen classification algorithms from Table 2 were applied to the twenty-nine data sets listed in Table 1, using stratified 10-fold cross validation. The results of these experiments are reported and analyzed in this section. We begin by presenting the summary results, aggregated over all twenty-nine data sets, and then look at some more refined results, aggregated over the least and then the most imbalanced data sets. We end this section by describing the lowest level results at the data set level.

The summary results of our experiments are presented in Table 3. There is a row for each classifier and the three evaluation metrics are presented in the three sets of columns. Each of the three metrics includes a raw score as well as a rank score. The rank is computed by ranking the performance of each of the thirteen classifiers for each data set and then taking the average over all twenty-nine data sets. If there is a tie between two or more algorithms, then each algorithm receives the average rank value within the associated range (e.g., if two algorithms are tied for second and third place then each gets a value of 2.5). Each rank value can vary from 1 (best) to 13 (worst). The rank values are useful because the raw scores could be thrown off by a small number of very good or very bad values. To facilitate the analysis of the results in Table 3, each cell is placed into one of three categories: very good results (better than most) are highlighted in bold, poor results (worse than most) are italicized, and the results between these extremes are left unchanged (i.e., normal font). The best classifier value for the entire column is underlined.

Based on the results in Table 3, and recalling that accuracy is not a very good measure of performance for imbalanced data, we conclude that DT, ADB, and GRB are consistently very good-performing classifiers: they are in the top category for the F1-measure and AUC for both raw score and ranking. DT also has the very best average values for three of the four non-accuracy values. If we focus on AUC we may also consider CNB to be high performing and if we focus on F1-measure then RF can be considered high performing.

TABLE 3: SUMMARY OF MAIN RESULTS

	ACCURACY		F1-MEASURE		AUC	
	RAW	RANK	RAW	RANK	RAW	RANK
LR	**93.2**	5.8	*.280*	*9.8*	*.620*	*9.4*
LDA	**93.2**	6.5	.412	6.5	.681	6.7
QDA	*82.3*	*9.5*	.355	7.2	.661	6.7
KNN	**93.7**	**4.9**	.432	6.2	.688	6.6
DT	90.7	8.1	**.477**	**4.5**	**.720**	**5.0**
RF	**93.5**	**4.6**	**.458**	**5.6**	.698	6.7
SV	92.1	5.5	*.240*	*9.8*	*.603*	*9.5*
MLP	91.3	6.6	.362	6.8	.655	7.3
ADB	**94.0**	5.4	**.471**	**5.2**	**.713**	**5.6**
GRB	**94.1**	**5.1**	**.485**	**4.6**	**.717**	**5.9**
GNB	*78.1*	*9.6*	.349	7.1	.697	**6.0**
BNB	90.0	7.9	*.169*	10.4	*.580*	*9.7*
CNB	*72.7*	*11.6*	.346	7.2	**.713**	**6.0**

Table 3 also shows that several algorithms perform consistently poorly—at least over the averages of the twenty-nine data sets. In particular, LR, SV, and BNB all perform in the worst category for all four F1-measure and AUC values. Overall, we do see very substantial differences in the raw scores for accuracy, F1-measure, and AUC, so it appears that these classifiers perform very differently from DT, ADB, and GRB in the presence of class imbalance. However, since none of the classifiers achieve rankings very near to the best value (1) or worst value (13), we conclude that there is no classifier that consistently performs best or worst for each of the underlying data sets.

We next refine our analysis by noting the number of times a classifier is ranked either top-1 or within the top-3. These values are encoded as #top-1/#top-3 in Table 4 for the six most competitive classification algorithms. The results are further broken down based on whether the data set is within the upper or lower part of Table 1 (the lower part begins with the car data set and contains the fourteen most imbalanced data sets). If class imbalance affects classifiers unequally, we are most apt to see this in the "lower" data sets.

TABLE 4: FREQUENCY OF TOP RANKS

	ACCURACY		F1-MEASURE		AUC	
	UPPER	LOWER	UPPER	LOWER	UPPER	LOWER
KNN	0/3	0/9	0/2	2/6	0/1	2/4
DT	0/2	0/2	0/5	2/5	3/3	0/8
RF	1/5	4/5	1/4	0/1	1/4	0/0
ADB	0/7	0/1	1/6	0/4	1/3	0/2
GRB	4/6	1/4	3/6	2/6	1/5	1/3
CNB	0/0	0/0	3/4	0/4	3/5	7/7

The results in Table 4 generally agree with the results in Table 3, but there are some interesting items to highlight. Based on Table 3 we concluded that DT, ADB, and GRB, performed very well for F1-measure and AUC. If we prioritize the results in Table 4 for the "lower" data sets with more class imbalance, we see that DT has the strongest performance of these three algorithms due to its superiority on AUC, where it is in the top-3 eight times. But perhaps the single most notable result from Table 4 is that CNB has 7 first place rankings for AUC for the lower (most imbalanced) data sets. Since there are only 14 data sets in that part, CNB ranks first out of thirteen algorithms for half of these data sets. RF shows good performance on the accuracy and F1-measure metrics, but does not perform as well with AUC score or with more imbalanced data sets. Although ADB and GRB are among the top performing algorithms, they both have a lesser frequency of top-ranking scores in the lower half of data set results. The KNN classifier also is notable in that it consistently scores better for the more imbalanced data sets and for these data sets is often in the top-3 of results.

Data set level results are provided for the F1-measure raw score and rank, respectively, in Table 5 and Table 6. The imbalance ratio (IR) field is computed as the number of majority class examples divided by the number of minority class examples and measures the level of class imbalance.

TABLE 5: DATA SET LEVEL RESULTS (F1-MEASURE RAW SCORE)

	IR	LR	LDA	QDA	KNN	DT	RF	SV	MLP	ADB	GRB	GNB	BNB	CNB
glass1	1.8	.20	.25	.58	.68	.62	.71	.59	.07	.67	.73	.58	.36	.54
hbmn	2.8	.21	.25	.29	.35	.38	.27	.05	.34	.38	.34	.29	.00	.50
ecoli1	3.4	.63	.72	.46	.70	.68	.66	.65	.67	.71	.74	.53	.00	.71
occ	3.7	.92	.92	.89	.79	.79	.82	.00	.87	.90	.91	.92	.73	.70
hmeq	4.0	.00	.36	.38	.09	.35	.35	.00	.07	.31	.35	.29	.05	.14
thy	5.1	.99	.78	.92	.86	.96	.96	.48	.77	.94	.88	.93	.6	.93
glass6	6.4	.77	.80	.78	.79	.81	.81	.85	.24	.86	.86	.81	.78	.67
yeast3	8.1	.15	.74	.24	.74	.68	.72	.00	.70	.75	.75	.24	.00	.52
ecoli3	8.6	.06	.67	.20	.58	.61	.61	.00	.10	.56	.63	.45	.00	.50
pbloc	8.8	.62	.68	.63	.73	.80	.82	.28	.78	.78	.82	.44	.00	.58
c4	9.5	.00	.01	.15	.05	.07	.06	.00	.06	.00	.01	.12	.03	.17
vow0	10.0	.72	.66	.69	.77	.70	.81	.86	.85	.83	.82	.71	.67	.54
prom	13.6	.00	.00	.02	.01	.09	.01	.00	.07	.00	.00	.00	.00	.12
seism	14.2	.05	.16	.12	.08	.11	.06	.00	.12	.08	.13	.19	.24	.09
aba9	16.4	.05	.59	.53	.17	.33	.24	.00	.09	.44	.37	.26	.00	.14
car	25.6	.07	.00	.07	.24	.83	.73	.76	.57	.77	.86	.24	.00	.28
lett9	25.8	.80	.72	.78	.95	.87	.92	.95	.95	.82	.88	.61	.07	.33
fd	26.0	.06	.37	.14	.15	.27	.25	.00	.19	.27	.33	.08	.42	.07
lett25	26.2	.74	.48	.77	.98	.93	.95	.98	.97	.90	.91	.59	.00	.30
yeast4	28.1	.00	.30	.09	.18	.30	.17	.00	.11	.26	.31	.07	.00	.28
yeast5	32.7	.00	.58	.07	.70	.64	.51	.00	.39	.65	.68	.15	.00	.39
cacao	34.9	.00	.07	.05	.00	.03	.05	.00	.10	.03	.03	.15	.00	.10
train	39.2	.16	.36	.35	.38	.42	.41	.00	.20	.23	.28	.28	.00	.17
yeast6	41.4	.00	.41	.05	.57	.42	.38	.00	.00	.44	.43	.06	.00	.26
redw	56.1	.00	.00	.03	.00	.03	.00	.00	.00	.00	.00	.07	.00	.05
poker	85.9	.00	.00	.00	.00	.07	.00	.00	.21	.00	.00	.00	.00	.01
root	100.1	.95	.90	.87	1.0	.95	.95	.53	.98	1.0	.95	.92	.96	.93
aba1	129.4	.00	.00	.00	.00	.06	.00	.00	.00	.00	.04	.01	.00	.02
whw	243.9	.00	0.18	.13	.00	.04	.08	.00	.07	.08	.04	.14	.00	.01
AVG		.28	.41	.36	.43	.48	.46	.24	.36	.47	.49	.35	.17	.35

TABLE 6: DATA SET LEVEL RESULTS (F1-MEASURE RANK)

	IR	LR	LDA	QDA	KNN	DT	RF	SV	MLP	ADB	GRB	GNB	BNB	CNB
glass1	1.8	12.0	11.0	7.0	3.0	5.0	2.0	6.0	13.0	4.0	1.0	8.0	10.0	9.0
hbmn	2.8	11.0	10.0	7.0	4.0	3.0	9.0	12.0	6.0	2.0	5.0	8.0	13.0	1.0
ecoli1	3.4	10.0	2.0	12.0	5.0	6.0	8.0	9.0	7.0	3.0	1.0	11.0	13.0	4.0
occ	3.7	1.0	2.0	6.0	9.0	10.0	8.0	13.0	7.0	5.0	4.0	3.0	11.0	12.0
hmeq	4.0	12.5	2.0	1.0	9.0	3.0	4.0	12.5	10.0	6.0	5.0	7.0	11.0	8.0
thy	5.1	1.0	10.0	7.0	9.0	2.0	3.0	13.0	11.0	4.0	8.0	5.0	12.0	6.0
glass6	6.4	11.0	7.0	9.5	8.0	5.0	4.0	3.0	13.0	1.0	2.0	6.0	9.5	12.0
yeast3	8.1	11.0	4.0	9.0	3.0	7.0	5.0	12.5	6.0	2.0	1.0	10.0	12.5	8.0
ecoli3	8.6	11.0	1.0	9.0	5.0	4.0	3.0	12.5	10.0	6.0	2.0	8.0	12.5	7.0
pbloc	8.8	9.0	7.0	8.0	6.0	3.0	1.0	12.0	4.0	5.0	2.0	11.0	13.0	10.0
c4	9.5	12.5	10.0	2.0	7.0	4.0	5.0	12.5	6.0	11.0	9.0	3.0	8.0	1.0
vow0	10.0	7.0	12.0	10.0	6.0	9.0	5.0	1.0	2.0	3.0	4.0	8.0	11.0	13.0
prom	13.6	10.5	10.5	4.0	6.0	2.0	5.0	10.5	3.0	7.0	10.5	10.5	10.5	1.0
seism	14.2	12.0	3.0	5.0	10.0	7.0	11.0	13.0	6.0	9.0	4.0	2.0	1.0	8.0
aba9	16.4	11.0	1.0	2.0	8.0	5.0	7.0	12.5	10.0	3.0	4.0	6.0	12.5	9.0
car	25.6	11.0	12.5	10.0	8.0	2.0	5.0	4.0	6.0	3.0	1.0	9.0	12.5	7.0
lett9	25.8	8.0	10.0	9.0	2.0	6.0	4.0	1.0	3.0	7.0	5.0	11.0	13.0	12.0
fd	26.0	12.0	2.0	9.0	8.0	4.0	6.0	13.0	7.0	5.0	3.0	10.0	1.0	11.0
lett25	26.2	9.0	11.0	8.0	2.0	5.0	4.0	1.0	3.0	7.0	6.0	10.0	13.0	12.0
yeast4	28.1	12.0	3.0	9.0	6.0	2.0	7.0	12.0	8.0	5.0	1.0	10.0	12.0	4.0
yeast5	32.7	12.0	5.0	10.0	1.0	4.0	6.0	12.0	7.0	3.0	2.0	9.0	12.0	8.0
cacao	34.9	11.5	4.0	5.0	11.5	7.0	6.0	11.5	2.0	8.0	9.0	1.0	11.5	3.0
train	39.2	11.0	4.0	5.0	3.0	1.0	2.0	12.5	9.0	8.0	7.0	6.0	12.5	10.0
yeast6	41.4	11.5	5.0	9.0	1.0	4.0	6.0	11.5	11.5	2.0	3.0	8.0	11.5	7.0
redw	56.1	9.0	9.0	3.0	9.0	4.0	9.0	9.0	9.0	9.0	9.0	1.0	9.0	2.0
poker	85.9	8.5	8.5	8.5	8.5	2.0	8.5	8.5	1.0	8.5	8.5	8.5	8.5	3.0
root	100.1	6.5	11.0	12.0	1.5	6.5	6.5	13.0	3.0	1.5	6.5	10.0	4.0	9.0
aba1	129.4	9.0	9.0	9.0	9.0	1.0	9.0	9.0	9.0	9.0	2.0	4.0	9.0	3.0
whw	243.9	11.5	1.0	3.0	11.5	8.0	4.5	11.5	6.0	4.5	7.0	2.0	11.5	9.0
AVG		9.8	6.5	7.2	6.2	4.5	5.6	9.8	6.9	5.2	4.6	7.1	10.3	7.2

The results in Table 5 and Table 6, and the corresponding information for accuracy and AUC, were used to generate the summary tables presented earlier in this section. Accuracy results at the data set level are not provided since they are not very meaningful when there is class imbalance. Data set level results for AUC are not included to save space and because they show similar patterns as the F1-measure results.

Table 3, which provides performance results aggregated over all twenty-nine data sets, and Table 4, which provides results aggregated over the most and least imbalanced data sets, provide a good overview of the relative performance of each algorithm. But Table 5 shows one behavior that is not discernable from these other tables; the number of times that an algorithm totally fails to identify any minority class examples, yielding an F1-measure of 0.00 (presumably because the recall is 0). We expect this to happen only for the most imbalanced data sets. Table 7 lists how many times each algorithm generates an F1-measure of 0.00 over the twenty-nine data sets. The table is sorted in order of best performing to worst performing algorithms, with respect to the number of 0.00 entries.

TABLE 7: NUMBER OF F1-MEASURE VALUES OF 0.00

Algorithm	# 0.00
DT, CNB	0
GNB, QDA	2
RF, MLP, GRB	3
LDA	4
ADB, KNN	5
LR	8
BNB, SV	18

Table 7 shows that there is a tremendous difference in the number of F1-measure values of 0.00 generated by the different algorithms. Both DT and CNB, which were previously identified as being within the group of six top performing algorithms, never generate any F1-measure values of 0.00. Of the four others within this group, three of them (GNB, RF, and GRB) have between 2 and 3 of these "null" values, while ADB has 5 null values. The three algorithms previously identified as being the worst performing, LR, BNB, and SV, have the three largest numbers of null values—with BNB and SV having null values for 18 of the 29 datasets. BNB and SV even generate null F1-measure values for data sets that are not even that imbalanced (i.e., with an imbalance ratio of less than 4). Thus we see that the number of null F1-measure values is highly related to the overall performance of the algorithm. It would appear that algorithms that perform extremely poorly do not have the expressive power to separate out very rare groups of examples. This makes some sense in that logistic regression and support vector machines have limited expressive power.

IV. CONCLUSION

This paper provides a comprehensive empirical study that evaluates the performance of thirteen different algorithms when applied to twenty-nine data sets with varying levels of class imbalance. The results clearly demonstrate that different classification algorithms perform very differently. We summarize our main empirical conclusions as follows:

- The DT, ADB, and GRB algorithms perform consistently well on both F1-measure and AUC. The CNB and RF algorithms perform well overall, but not quite as consistently. KNN performs more poorly than these other five algorithms, but still outperforms most others. Thus we view DT, ADB, GRB, CNB, RF, and KNN as the six best-performing algorithms.
- If we focus on the set of data sets with higher class imbalance, DT and CNB perform most impressively. If we focus on the number of times an algorithm fails to produce an F1-measure above 0.00, then DT and CNB perform the best, with RF and GRB a bit further behind. This gives a boost to DT and CNB, especially if there is the potential for extreme levels of class imbalance.
- The LR, SV, and BNB algorithms consistently perform quite poorly. Furthermore, in many cases they cannot identify any minority examples and have an F1-measure of 0.00 (this happens in more than 60% of the cases for SV and BNB and even on data sets that are not extremely imbalanced).
- Other algorithms perform between the best and worst and include: GNB, LDA, QDA and MLP.

Thus we have identified three general performance groups based on the algorithms performance on imbalanced data sets. Our top group has six entries. If we had to pick a single algorithm as the best, it would be DT since it performs consistently well over all twenty-nine data sets, performs among the top two on the fourteen data sets that are most imbalanced, and is one of only two algorithms that never produces an F1-measure of 0. If we had to select our second best algorithm it would be CNB because it performs very well on the fourteen most imbalanced data sets and also never produces an F1-measure of 0. However, since performance varies by data set, we would generally recommend that for imbalanced data sets the practitioner try several of the algorithms listed in our top six, but make sure to always include DT and CNB.

The good performance of certain types of algorithms on imbalanced data was predicted by some prior work. Drummond and Holte [5] suggested that the decision tree splitting criteria would perform well on class imbalance and both tree-based methods, DT and RF, were amongst the best-performing algorithms. Weiss [16] asserted that boosting algorithms would perform well on imbalanced data and the two boosting algorithms, ADB and GRB, were also amongst the best performing algorithms.

This study showed that some algorithms perform much better than others on imbalanced data. We presume that this means that they are specifically better suited to learning from imbalanced data—but it could be that they are just generally superior to the other algorithms in our study (i.e., even on balanced data sets). In order to address this question, in future

work we plan to artificially balance each of the twenty-nine data sets employed in this study and see if the observed superiority of certain algorithms is eliminated or reduced. We will also consider extending this study to include algorithms that are specifically designed to handle class imbalance, including methods that rely on sampling.

REFERENCES

1. J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17(2-3): 255-287, 2011.
2. A. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, 30(7): 1145-1159, 1997.
3. N. V. Chawla. C4.5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In Workshop on Learning from Imbalanced Datasets II, International Conference on Machine Learning, 2003.
4. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16: 321- 357, 2002.
5. C. Drummond, and R.C. Holte. Exploiting the cost(in)sensitivity of decision tree splitting criteria. In Proceedings of the Seventeenth International Conference on Machine Learning, pages 239-246, 2000.
6. D. Dua, and C. Graff. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2019.
7. Ebrahimi. Financial Distress Prediction, Version 1. Retrieved October 1, 2018 from https://www.kaggle.com/shebrahimi/financial-distress, 2017.
8. N. Japkowicz, and S. Stephen. The class imbalance problem: a systematic study. Intelligent Data Analysis, 6(5):429-450, 2002.
9. R. Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. IJCAI'95 Proceedings of the 14th international joint conference on Artificial intelligence, vol. 2: 1137-1143, 1995.
10. X. Liu, J. Wu and Z. Zhou, "Exploratory Undersampling for Class-Imbalance Learning," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2): 539-550, 2009.
11. Pedregosa et al., Scikit-learn: Machine Learning in Python. JMLR 12: 2825-2830, 2011.
12. F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In Proceedings of the Fifteenth International Conference on Machine Learning, 43–48. San Francisco: Morgan Kaufmann, 1998.
13. Regi. Promotion Response and Target Datasets, Version 1. Retrieved October 1, 2018 from https://www.kaggle.com/regivm/promotion-response-and-target-datasets, 2018.
14. K. M. Ting. The problem of small disjuncts: its remedy in decision trees. In Proceeding of the Tenth Canadian Conference on Artificial Intelligence, pages 91-97, 1994.
15. Vallala, A. HMEQ_Data, Version 1. Retrieved June 20, 2018 from https://www.kaggle.com/ajay1735/hmeq-data, 2018.
16. G.M. Weiss. Mining with Rarity: A Unifying Framework. *SIGKDD* Explorations, 6(1): 7-19, 2004.
17. Zyaj. Unbalanced_clf, Version 1. Retrieved December 20, 2018 from https://www.kaggle.com/zyajnokid/train#train.csv, 2017.