CS699 Project Assignment

Classification of Class-imbalanced Dataset

using Oversampling and Undersampling techniques

Archana Balachandran

Submitted on: April 16, 2016

Index

Title	Page
Chapter 1 Introduction	3
Chapter 2 Process and Performance Measures - The Class Imbalanced Dataset - Building classification models using initial dataset - Oversampling and Classification of the Initial Dataset - Undersampling and Classification of the Initial Dataset	 4 5 6 10
Chapter 3 Analysis of Best Classification Model and Training Dataset	13
Chapter 4 Conclusion	14
References	15

.

1. Introduction

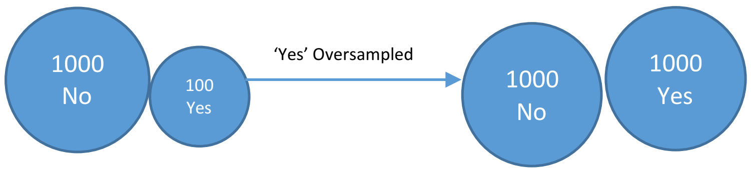
The project aims to build an efficient classification model with high accuracy from a class imbalanced dataset. The initial dataset has 16 attributes and 3165 tuples, of which 2802 are 'no' tuples and 363 are 'yes' tuples. This represents a **class imbalance problem**, where the total number of a class of is far less than the total number of another class of data. Real life scenarios of class imbalance problem include fraud detection, anomaly detection, medical diagnosis, oil spillage detection, facial recognition, etc.

The task is to build a model that predicts 'yes' tuples with high accuracy, which is to ensure the highest TP rate, while being cautious of increasing FP rates.
Most machine learning algorithms and works best when the number of instances of each classes are roughly equal.

Two approaches identified to deal with class imbalance dataset by adjusting the class distribution of a dataset are cost function based approach and sampling approach. This project explores two of the popular sampling approaches –Oversampling and undersampling.

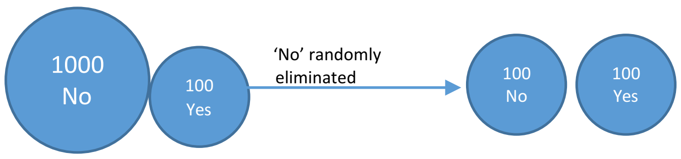
1. Oversampling:

This project explores the classification of the dataset using oversampling techniques with the help of SMOTE algorithm. Oversampling generates samples of the minority class until its comparable to the size of the majority class.



2. Undersampling

This is another technique demonstrated in this project, where samples in the majority class is eliminated to equate the size of the minority class. Classification algorithms are then applied to predict new samples into the minority class with better accuracy.

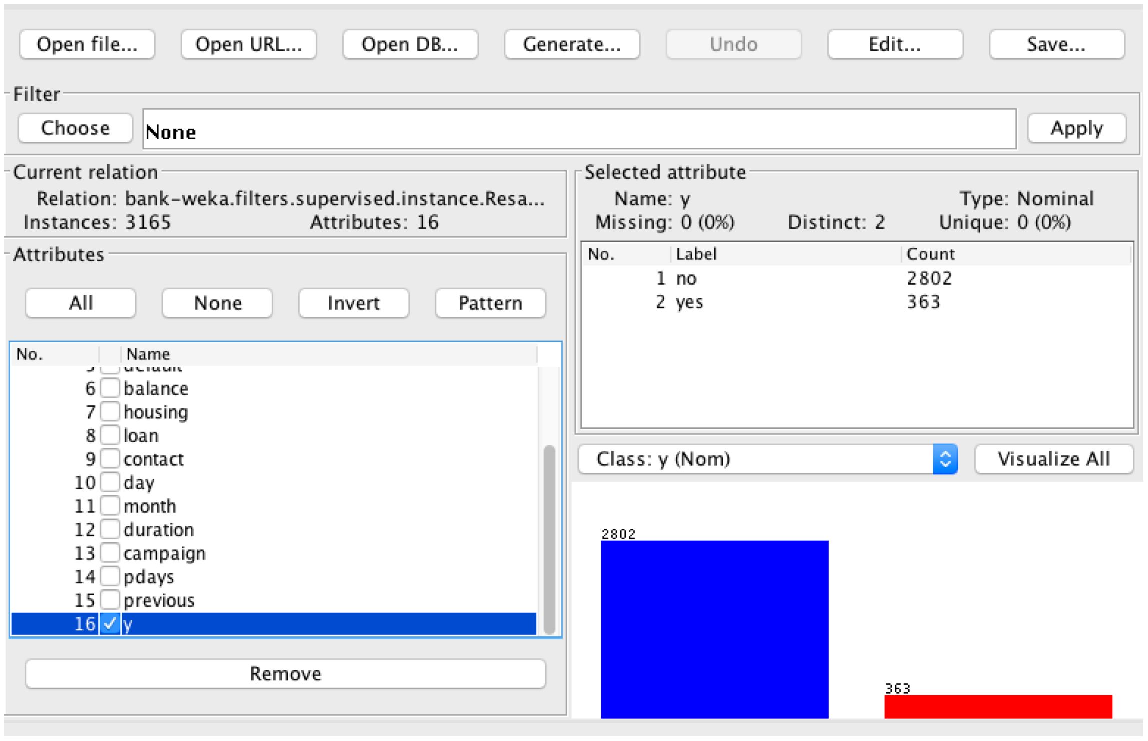


Chapter 2
The Process and Performance Measures

The Class Imbalanced Dataset:

The initial dataset in this project has 16 attributes, and classification if made for the class label ‘y’. 2802 instances are marked as ‘No’ and 363 instances are marked ‘yes’ making it a minority class, thus creating a class imbalanced dataset where the ‘yes’ is the minority class, and ‘no’ is the majority class.

Analyzing class distribution of the dataset in Weka explorer:



Filter: None

Current relation
Relation: bank-weka.filters.supervised.instance.Resa...
Instances: 3165 Attributes: 16

Attributes

No.	Name
6	balance
7	housing
8	loan
9	contact
10	day
11	month
12	duration
13	campaign
14	pdays
15	previous
16	✓ y

Selected attribute
Name: y Type: Nominal
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count
1	no	2802
2	yes	363

Class: y (Nom)

Building Classification Models using the initial Dataset:

Classification Models for the initial dataset are created on Weka using classifier algorithms such as Naïve Bayes, Multilayer Perceptron, Logistic Regression and decision trees. For testing, cross validation method was used, and the performance measure table thus obtained has been recorded. It was observed that the TP rate for 'yes' is very low.

1. Naïve Bayes: (87% correctly classified instances)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.925	0.548	0.929	0.925	0.927	0.846	no
	0.452	0.075	0.437	0.452	0.444	0.846	yes
Weighted Avg	0.87	0.494	0.872	0.87	0.871	0.846	

2. Decision Trees-J48 algorithm: (89.51% correctly classified)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.959	0.598	0.925	0.959	0.942	0.733	no
	0.402	0.041	0.559	0.402	0.468	0.733	yes
Weighted Avg	0.895	0.534	0.883	0.895	0.887	0.733	

3. Multilayer Perceptron: (87.93% correctly classified)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.946	0.636	0.92	0.946	0.933	0.811	no
	0.364	0.054	0.466	0.364	0.409	0.811	yes
Weighted Avg	0.879	0.57	0.868	0.879	0.873	0.811	

4. Logistic Regression: (89.38% correctly classified)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.975	0.733	0.911	0.975	0.942	0.881	no
	0.267	0.025	0.581	0.267	0.366	0.881	yes
Weighted Avg	0.894	0.652	0.873	0.894	0.876	0.881	

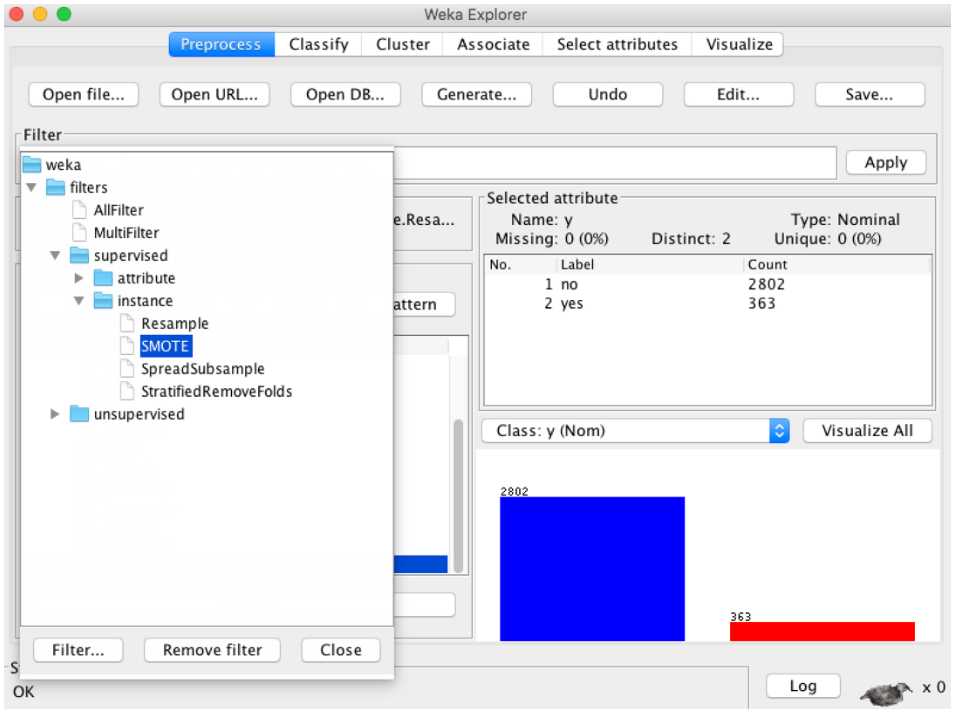
Oversampling and Classification of the Initial Dataset:

Oversampling replicates the minority class instances to counter the effects of imbalanced datasets by using various techniques such as Random Oversampling with replacement, Synthetic Minority Oversampling Technique(SMOTE), Adaptive Synthetic sampling approach. For this project, we explore the SMOTE algorithm, due to its high performance efficiency, which can also be seamlessly implemented using Weka. In the concluding chapter of this project, we will also discuss how SMOTE can be combined with techniques like TOMEK, ENN and Boosting which further increases the accuracy of classification.

Another advantage of SMOTE is that it provides new related information on the minority class to the learning algorithms, along with undersampling the majority class.
Other efficient techniques

We will now discuss how oversampling was performed in Weka to further classify the imbalanced dataset to observe the results of classification.

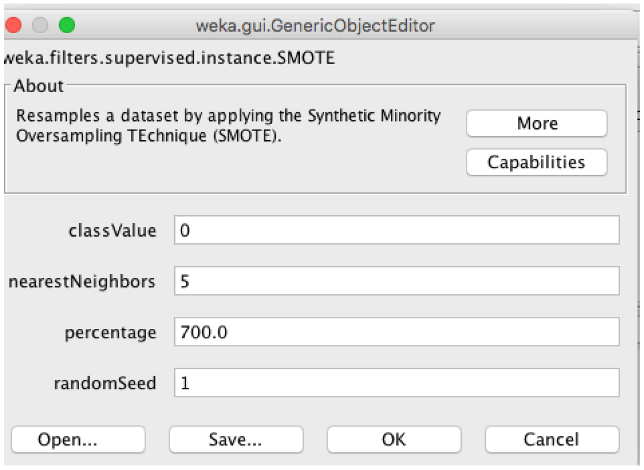
In the preprocess tab, once the initial dataset was loaded, SMOTE algorithm can be selected from the supervised filter instances.



Name: y
Type: Nominal
Missing: 0 (0%)
Distinct: 2
Unique: 0 (0%)

No.	Label	Count
1	no	2802
2	yes	363

Edit the algorithm setting to set the percentage of SMOTE instances to generate to 700. This is to resample the minority instances to level up to that of the majority instances. Click Apply.



weka.gui.GenericObjectEditor

weka.filters.supervised.instance.SMOTE

About

Resamples a dataset by applying the Synthetic Minority Oversampling TEchnique (SMOTE).

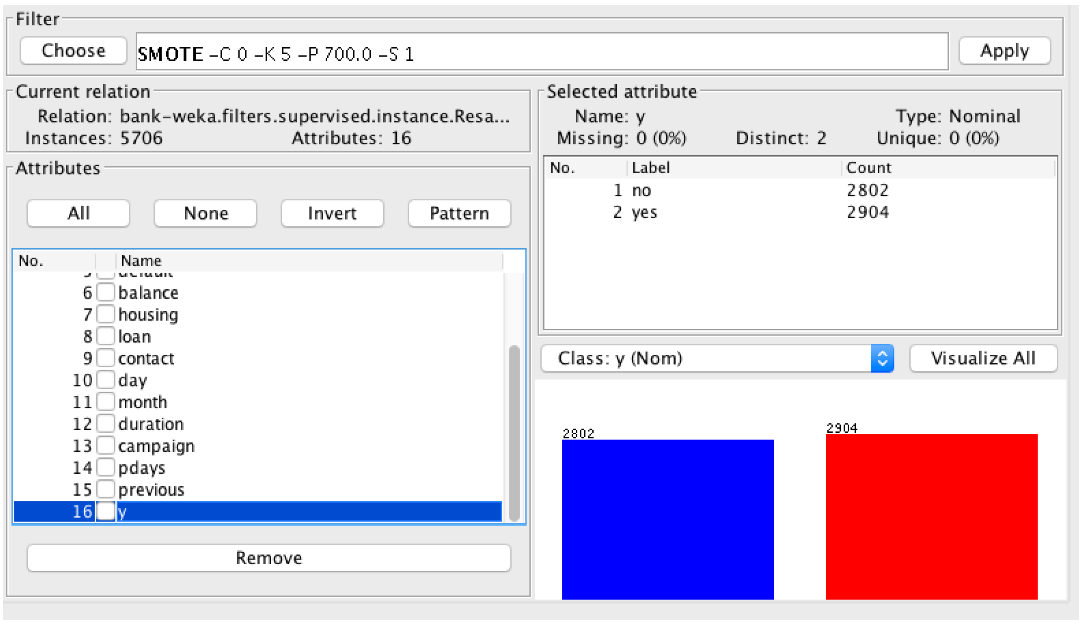
classValue 0

nearestNeighbors 5

percentage 700.0

randomSeed 1

We then obtain two roughly equal classes.



Filter

SMOTE -C 0 -K 5 -P 700.0 -S 1

Current relation

Relation: bank-weka.filters.supervised.instance.Resa...
Instances: 5706 Attributes: 16

Selected attribute

Name: y Type: Nominal
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count
1	no	2802
2	yes	2904

Attributes

No.	Name
6	balance
7	housing
8	loan
9	contact
10	day
11	month
12	duration
13	campaign
14	pdays
15	previous
16	y

This dataset is saved as bank-SMOTE.arff.

Classification models are built using the four classifiers specified above, and tested to note the performance measures of each model. Classification is performed on this new dataset, using the supplied test set under test options. For each classifier algorithm, the new dataset is selected, and the the test is run and the performance measures are noted. For the purpose of extracting the optimal classifier for classification, we will use cross validation and supplied test set as test options to perform a comparative study.

Performance Measure Tables:

1. Naïve Bayes with Supplied test set: (86.8% correctly classified)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.795	0.06	0.927	0.795	0.856	0.932	no
	0.94	0.205	0.826	0.94	0.879	0.932	yes
Weighted Avg	0.869	0.134	0.876	0.869	0.868	0.932	

Naïve Bayes with Cross Validation (86.6% correctly classified instances):

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.792	0.061	0.926	0.792	0.854	0.93	no
	0.939	0.208	0.824	0.939	0.878	0.93	yes
Weighted Avg	0.867	0.136	0.874	0.867	0.866	0.93	

2. J48 with Supplied test set (95.9% correctly classified)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.959	0.039	0.959	0.959	0.959	0.982	no
	0.961	0.041	0.96	0.961	0.96	0.982	yes
Weighted Avg	0.96	0.04	0.96	0.96	0.96	0.982	

J48 with Cross Validation (92.6% correctly classified):

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.922	0.069	0.928	0.922	0.925	0.943	no
	0.931	0.078	0.925	0.931	0.928	0.943	yes
Weighted Avg	0.927	0.074	0.927	0.927	0.927	0.943	

3. Multilayer Perceptron with Supplied test set (96.5% correctly classified):

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.94	0.01	0.989	0.94	0.964	0.983	no
	0.99	0.06	0.944	0.99	0.967	0.983	yes
Weighted Avg	0.965	0.036	0.966	0.965	0.965	0.983	

Multilayer Perceptron with 2-fold cross validation (91.1% correctly classified):

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.887	0.065	0.929	0.887	0.907	0.952	no
	0.935	0.113	0.895	0.935	0.915	0.952	yes
Weighted Avg	0.911	0.09	0.912	0.911	0.911	0.952	

4. Logistic Regression with supplied test set (91.2% correctly classified):

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.898	0.073	0.922	0.898	0.91	0.957	no
	0.927	0.102	0.904	0.927	0.915	0.957	yes
Weighted Avg	0.913	0.088	0.913	0.913	0.913	0.957	

Logistic Regression with Cross Validation (90.8% correctly classified):

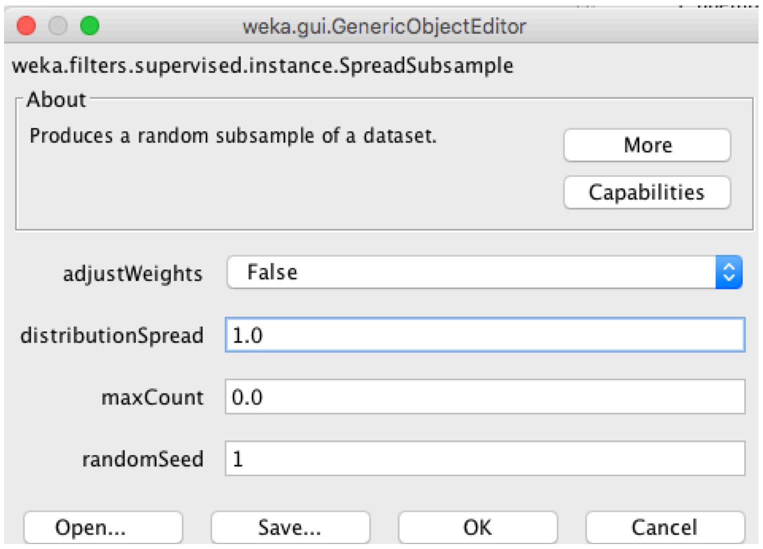
	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.894	0.078	0.917	0.894	0.906	0.954	no
	0.922	0.106	0.9	0.922	0.911	0.954	yes
Weighted Avg	0.909	0.092	0.909	0.909	0.908	0.954	

Undersampling and Classification of the Initial Dataset:

As discussed in the previous section, undersampling is a sampling approach to minimize the effect of an imbalanced class by eliminating instances of the majority class to make its size comparable to the minority class for classification.

Undersampling can be performed in Weka using many sampling algorithms that creates random sub samples, such as Resample and SpreadSubsample. For the purpose of this project, SpreadSubsample algorithm is used to implement undersampling because f its increased efficiency and also because it has additional features for specifying the maximum "spread" between the rarest and most common class.

After loading the initial dataset, Spreadsample sampling technique is selected from the supervised filters using weka.filters.supervised.instance.SpreadSubsample. Class value is set to 0 to automatically identify and replicate minority class, and the distance spread is set to 1 to represent uniform distribution.



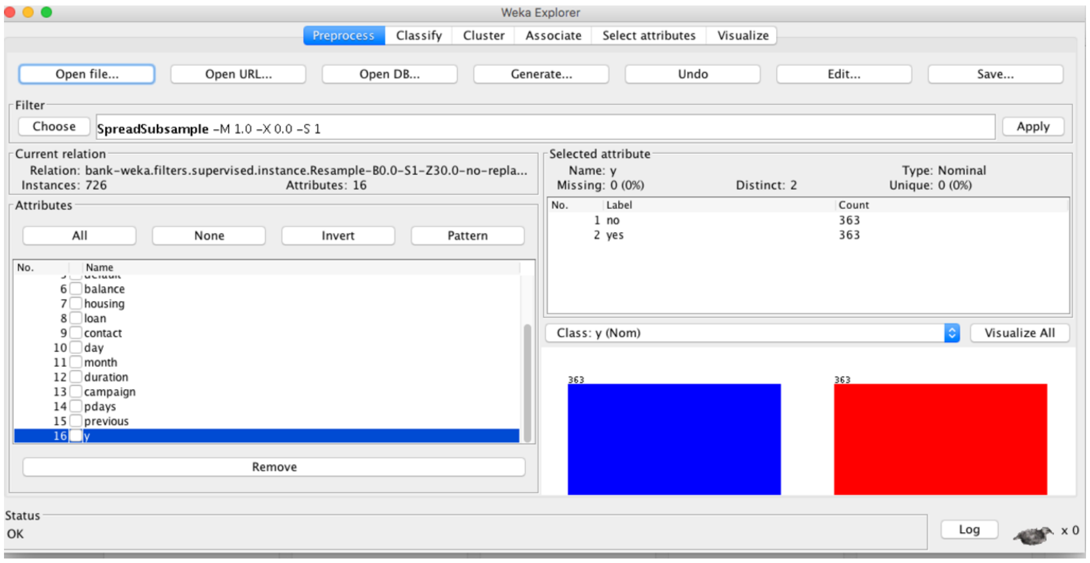
weka.filters.supervised.instance.SpreadSubsample

About

Produces a random subsample of a dataset.

adjustWeights	False
distributionSpread	1.0
maxCount	0.0
randomSeed	1

After applying the sampling technique, we get a dataset of size 726, which has 363 instances of each class.



Filter: SpreadSubsample -M 1.0 -X 0.0 -S 1

Current relation
Relation: bank-weka.filters.supervised.instance.Resample-B0.0-S1-Z30.0-no-repla...
Instances: 726 Attributes: 16

Selected attribute
Name: y Type: Nominal
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count
1	no	363
2	yes	363

Attributes

No.	Name
6	balance
7	housing
8	loan
9	contact
10	day
11	month
12	duration
13	campaign
14	pdays
15	previous
16	y

Class: y (Nom)

Status
OK

This test set is saved as bankspreadsample.arff (saved as Balachandran_Archana_best_dataset.arff for submission purposes).

Classification using the four specified classifier algorithms are performed on the newly generated test set, and their performance measures were tabulated.

Performance Measure Tables:

1. Naïve Bayes with Supplied test set: (79.4% correctly classified)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.788	0.198	0.799	0.788	0.793	0.864	no
	0.802	0.212	0.791	0.802	0.796	0.864	yes
Weighted Avg	0.795	0.205	0.795	0.795	0.795	0.864	

Naïve Bayes with Cross validation: (76% correctly classified)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.769	0.248	0.756	0.769	0.762	0.839	no
	0.752	0.231	0.765	0.752	0.758	0.839	yes
Weighted Avg	0.76	0.24	0.76	0.76	0.76	0.839	

2. J48 with Supplied test set (92.9% correctly classified)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.934	0.074	0.926	0.934	0.93	0.971	no
	0.926	0.066	0.933	0.926	0.929	0.971	yes
Weighted Avg	0.93	0.07	0.93	0.93	0.93	0.971	

J48 with Cross Validation (79.4% correctly classified)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.782	0.193	0.802	0.782	0.792	0.831	no
	0.807	0.218	0.788	0.807	0.797	0.831	yes
Weighted Avg	0.795	0.205	0.795	0.795	0.795	0.831	

3. Multilayer Perceptron with Supplied test set (99.03% correctly classified)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.983	0.003	0.997	0.983	0.99	0.986	no
	0.997	0.017	0.984	0.997	0.99	0.986	yes
Weighted Avg	0.99	0.01	0.99	0.99	0.99	0.986	

Multilayer Perceptron with 10-fold Cross Validation (77% correctly classified):

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.774	0.229	0.772	0.774	0.773	0.86	no
	0.771	0.226	0.773	0.771	0.772	0.86	yes
Weighted Avg	0.773	0.227	0.773	0.773	0.773	0.86	

4. Logistic Regression with Supplied test set (84.2% correctly classified)

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.857	0.171	0.834	0.857	0.845	0.913	no
	0.829	0.143	0.853	0.829	0.841	0.913	yes
Weighted Avg	0.843	0.157	0.843	0.843	0.843	0.913	

Logistic Regression with Cross Validation (80.3% correctly classified)

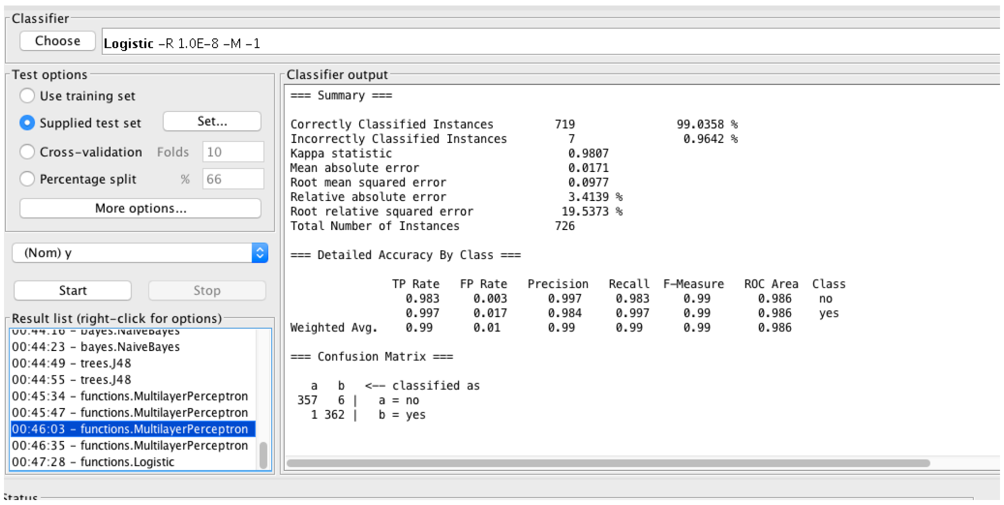
	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.815	0.209	0.796	0.815	0.805	0.878	no
	0.791	0.185	0.811	0.791	0.801	0.878	yes
Weighted Avg	0.803	0.197	0.803	0.803	0.803	0.878	

Chapter 3
Analysis of Best Classification Model and Training Dataset

Undersampling the initial dataset using Spreadsubsample sampling technique, and classifying the new dataset(bankspreadsample.arff) using Multilayer Perceptron classifier using the undersampled dataset as the supplied test set yielded the best performing classifying model, which gave 99.03% correctly classified instances.

The performance measure table of this classification model:

	TP rate	FP rate	Precision	Recall	F-measure	ROC Area	Class
	0.983	0.003	0.997	0.983	0.99	0.986	no
	0.997	0.017	0.984	0.997	0.99	0.986	yes
Weighted Avg	0.99	0.01	0.99	0.99	0.99	0.986	


Classifier
Choose
Test options
Use training set
Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) y
Start
Stop
Result list (right-click for options)
00:44:23 - bayes.NaiveBayes
00:44:49 - trees.J48
00:44:55 - trees.J48
00:45:34 - functions.MultilayerPerceptron
00:45:47 - functions.MultilayerPerceptron
00:46:03 - functions.MultilayerPerceptron
00:46:35 - functions.MultilayerPerceptron
00:47:28 - functions.Logistic
Classifier output
Status

Logistic -R 1.0E-8 -M -1

Test options: Supplied test set

```
=== Summary ===

Correctly Classified Instances         719               99.0358 %
Incorrectly Classified Instances         7                0.9642 %
Kappa statistic                          0.9807
Mean absolute error                      0.0171
Root mean squared error                  0.0977
Relative absolute error                  3.4139 %
Root relative squared error             19.5373 %
Total Number of Instances              726

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                  0.983    0.003    0.997     0.983    0.99      0.986     no
                  0.997    0.017    0.984     0.997    0.99      0.986     yes
Weighted Avg.     0.99     0.01     0.99      0.99     0.99      0.986

=== Confusion Matrix ===

   a   b   <-- classified as
 357   6 |   a = no
   1 362 |   b = yes
```

Fig: Output screen of Multilayer Perceptron classifier test

Taking into consideration the following reasons, this model was selected as the best classification model:

1. Highest sample accuracy or percentage of correctly classified instances – 99.03%
2. Lowest Mean absolute error of 0.01
3. Highest ROC Area (closest to 1) which denotes very high accuracy- 0.986
4. Has the highest TP rate (0.997) while maintaining a very low FP rate (0.017)

Chapter 4
Conclusion

The purpose of this project was to carry out an experimental study to assess the impact of class imbalance distribution over the performance of four popular classifiers and to identify the best classifier to handle the imbalance problem.

In this study we found that the classifier which best responds to unbalanced data set is a Neural Network based algorithm. Neural networks are well known for handling large data sets with the ability of implicitly detect complex nonlinear relationships between dependent and independent variables. They are also self-adaptive flexible with a high degree of accuracy.

The experiment has provided information to declare that undersampling with SpreadSubSample technique, along with Multilayer perception classifier generates the best classification model. It is also interesting to note that oversampling with SMOTE technique, and then using a Multilayer perceptron classifier yields very similar classification results. We chose the undersampling approach because it yielded a model with a considerably lower FP rate than that of the oversampling technique.

We can observe that Multilayer Perceptron+ SpreadSubsample has the highest number of yes instances correctly classified (362 out of 363), but it has the highest number of no instances correctly classified(357 out of 363) too. For this reason, Multilayer Perceptron +SpreadSubsample has the highest F-measure than all the other models.

Other promising observations include the use of advanced SMOTE algorithms such as SMOTE+TOMEK, SMOTE+ENN and SMOTEBoost, which is beyond the scope of Weka's current implementation. SMOTEBoost[1] is suggested to be an excellent oversampling technique for improving the accuracy of the minority class, while s=using boosting to maintain the accuracy of the entire dataset. However, it is found to have a disadvantage of over generalization of the minority class space which affects the accuracy. For undersampling, One-side selection (OSS) method is used by combining Tomek Links and Condensed Nearest Neighbor Rule. This is an efficient method as it reduces the possibilities of noise. However, due to the usage of Tome links, this has reduced speed, which is an important factor while handling large datasets.

The experiments also allowed us to note that, even though under-sampling with Multilayer perceptron was the best model, over-sampling techniques using SMOTE are also effective to deal with imbalanced class distributions.

---

[1] Source available on https://github.com/danrodgar/WekaClassifiers/blob/master/WekaClassifiers/src/main/java/weka/classifiers/meta/SMOTEBoost.java

References:

1. http://www.ijser.org/researchpaper/A_New_approach_for_Classification_of_Highly_Imbalanced_Datasets_using_Evolutionary_Algorithms.pdf
2. http://www.esei.uvigo.es/fileadmin/docs/estudiantes/Traballos_fin_de_carreira/TFM_SSIA_2010-11/The_class_imbalance_problem_in_Medline_documents_classification.pdf
3. http://www.solver.com/xlminer/help/neural-networks-classification-intro
4. http://worldcomp-proceedings.com/proc/p2013/DMI8016.pdf
5. http://www.chioka.in/class-imbalance-problem/
6. http://gim.unmc.edu/dxtests/roc3.htm