Louvain School of Management

and

Norwegian School of Economics

A comparison of classification models for imbalanced datasets

Research Master's Thesis submitted by
Sergei Kurin

MSc in Economics and Business Administration

with a view of getting the degrees
Master in Business Engineering, professional focus

Supervisor at NHH
Stein Ivar Steinshamn

Supervisor at LSM
Marco Saerens

Academic Year 2016-2017

Abstract

Many practical classiflcation problems are imbalanced; i.e., at least one of the classes constitutes only a very small minority of the data. For such problems, the interest usually leans towards correct classiflcation of the minor class. Examples of such problems include fraud detection, rare disease diagnosing, etc. However, the most commonly used classiflcation algorithms do not work well for such problems because they aim to minimize the overall error rate, rather than paying special attention to the minor class. In the master's thesis, a number of models are evaluated with the objective to find those that better address the classification problem of imbalanced datasets. A special focus is given to the investigation of some ways of dealing with imbalanced datasets based on a Logistic Regression.

Contents

1 Introduction

Currently, in the Big Data (BD) era that we are living in, data science algorithms are of great importance to improve the performance of different applications (especially for those areas where data are collected daily and where meaningful knowledge and valuable assistance can be extracted to improve current systems) (Pérez-Ortiz and Jiménez-Fernández, 2016).

Generally, the term data science refers to the extraction of knowledge from data. This involves a wide range of techniques and theories drawn from many research flelds within mathematics, statistics and information technology, including statistical learning, data engineering, pattern recognition, uncertainty modeling, probability models, high performance computing, signal processing and machine learning (ML), among others. Precisely, the growth and development of this last research area has made data science more relevant, increasing the necessity of data scientists and the development of novel methods in the scientiflc community, given the great breadth and diversity of knowledge and applications of this area.

Classiflcation problems and methods have been considered a key part of ML, with a huge amount of applications published in the last few years. The concept of classiflcation in ML has been traditionally treated in a broad sense, very often including supervised, unsupervised and semi-supervised learning problems. Unsupervised learning is focused on the discovery and analysis of the structure of unlabeled data. This paradigm is especially useful to analyze whether there are differentiable groups or clusters present in the data (e.g., for segmentation). In the case of supervised learning, however, each data input object is preassigned a class label. The main task of supervised algorithms is to learn a model that ideally produces the same labeling for the provided data and generalizes well on unseen data (i.e., prediction). In this regard, classification learning applications are widely used to cope with difficult problems arising, for instance, from new renewable energy (RE) sources. A huge amount of RE applications can be found in the literature, such as prediction problems (e.g., solar radiation (Wang and Zhen, 2015) or signiflcant wave height estimation (Fernandez and Salcedo-Sanz, 2015)), optimization algorithms (wind farm or RE devices'

design), all of them with the common objective of improving RE systems signiflcantly.

1.1 Motivation and Problem Statement

In many supervised classification learning applications, there is a significant difference between the prior probabilities of different classes, i.e., between the probabilities with which an example belongs to the different classes of the classification problem. This situation is known as the class imbalance problem and it is common in many real problems from telecommunications, web, finance-world, ecology, biology, medicine, and which can be considered one of the top problems in data mining today. Examples of such problems include fraud detection, rare disease diagnosing (Chen and Liaw, 2004), new control techniques or fault diagnosis in RE systems (Panigrahi and Dash, 2009).

The hitch with imbalanced datasets is that standard classification learning algorithms are often biased towards the majority class (known as the "negative" class) and therefore there is a higher misclassification rate for the minority class instances (called the "positive" instances) (Lopéz and Fernández, 2013). Most of the studies on the behavior of several standard classifiers in imbalance domains have shown that significant loss of performance is mainly due to the skewed class distribution, given by imbalance ratio (IR), defined as ratio of the number of instances in the majorty class to the number of examples in the minority class.

The research problem in the master's thesis: classification models reveal low accuracy when dealing with imbalanced datasets. Therefore, a number of models will be evaluated with the objective to find those that better address the classification problem of imbalanced datasets. A special focus will be given to the investigation of some ways of dealing with imbalanced datasets based on a Logistic Regression.

The main objectives of the master's thesis are as follows:
1. The analysis of current problems of imbalanced learning;
2. The investigation of different classification models and evaluation metrics for imbalanced datasets;
3. The investigation of classification models based on Logistic Regression;
4. The comparison of different classification models for imbalanced datasets.

The theoretical results will include a literature review of the problem of imbalanced learning and the analysis of different classification models and their performance measures for imbalanced datasets. The empirical results will include the implementation of classification models using statistical software with data mining algorithms (e.g. R).

1.2 Thesis Structure

The thesis is organized as follows. First, some background knowledge on imbalanced problems and the results of related works will be presented (Chapter 2, Chapter 3, Chapter 4). Then, main results of this study will be discussed (Chapter 5) followed by the conclusion (Chapter 6).

Chapter 2

In this Chapter, different methods to deal with imbalanced datasets will be discussed. A special attention will be paid to some sampling techniques including undersampling, oversampling, and synthetic data generation. An important question for imbalanced problems is how to assess the performance of different classifiers. Simple predictive accuracy based on confusion matrix is not appropriate in such situations. It will be shown that the Area Under the ROC Curve (AUC) provides a single measure of a classifier's performance for evaluating which model is better.

Chapter 3

This Chapter is dedicated to different classification models that will be later evaluated with the objective to find those that better address the classification problem of imbalanced datasets. Namely, k-Nearest Neighbors (k-NN), Support Vector Machines (SVMs), and Random Forest are discussed. It will be shown that some of these classifiers can be oriented towards the class imbalance issues. For exampple, focused oversampling method is an efficient tool in combination with SVMs or stratifled bootstrap can be used for Random Forest model.

Chapter 4

In this Chapter, a special focus will be given to the investigation of some ways of dealing with imbalanced datasets based on a Logistic Regression. An iterative procedure that estimates the new a priori probabilities of a new dataset and adjusts the outputs of a classifier will be proposed for evaluation on imbalanced datasets.

Chapter 5

In this Chapter, the main results of performed study are presented and discussed. First, a comparison of different preprocessing techniques and classification mod-

els is made on the basis of several sample imbalanced datasets. Then, Logistic Regression with its adjustment procedure is analyzed. Finally, different classifiers and preprocessing techniques are applied to a real-word dataset 'Credit Card Fraud Detection'.

Chapter 6

This Chapter presents main conclusions and gives proposals for future work related to imbalanced problems.

2 Related Work

Throughout the last years, many solutions have been proposed to deal with the problem of poor performance of imbalance datasets, both for standard learning algorithms and for ensemble techniques (Lopéz and Fernández, 2013). The can be categorized into three major groups:

1. Data sampling in which the training instances are modified in such a way to produce a more or less balanced class distribution that allow classifiers to perform in a similar manner to standard classification.
2. Cost-sensitive learning. This type of solutions incorporate approaches at the data level, at the algorithmic level, or at both levels combines, considering higher costs for the misclassification of examples of the positive class with respect to the negative class, and therefore, trying to minimize higher cost errors.
3. Algorithmic modification. This procedure is oriented towards the adaptation of base learning methods to be more attuned to class imbalance issues.

2.1 Sampling Methods

Sampling is a popular methodology to counter the problem of class imbalance. In the specialized literature, one can find some papers about resampling techniques studying the effect of changing class distribution in order to deal with imbalanced datasets (Ha and Bunke, 1997). The goal of sampling methods is to create a dataset that has a relatively balanced class distribution, so that traditional classiflers are better able to capture the decision boundary between the majority and the minority classes. Those works have proved empirically that applying a preprocessing step in order to balance the class distribution is usually an useful solution. Furthermore, the main advantage of these techniques is that they are independent of the underlying classifier. Since the sampling methods are used to make the classiflcation of the minority class instances easier, the resulting (sampled) dataset should represent a "reasonable" approximation of the original dataset. Speciflcally, the resulting dataset should contain only instances that are, in some sense, similar to those in the original dataset, that is, all instances in the modifled dataset should be drawn from the

same (or similar) distribution to those originally in the dataset. Note that sampling methods need not create an exactly balanced distribution, merely a distribution that the traditional classiflers are better able to handle.

These sampling methods have acquired higher importance after many researches have proved that balanced data results in improved overall classification performance compared to an imbalanced data set (Analytics Vidhya Content Team, 2016).

The main methods used to treat imbalanced datasets are presented below:

1. Undersampling;
2. Oversampling;
3. Synthetic data generation (e.g., synthetic minority oversampling technique (SMOTE));
4. Cost-sensitive learning.

Two of the first sampling methods developed are random undersampling and random oversampling. In the random undersampling, the majority class instances are discarded at random until a more balanced distribution is reached. Consider, for example, a dataset consisting of 10 minority class instances and 100 majority class instances. In random undersampling, one might attempt to create a balanced class distribution by selecting 90 majority class instances at random to be removed. The resulting dataset will then consist of 20 instances: 10 (randomly remaining) majority class instances and (the original) 10 minority class instances.
Alternatively, in random oversampling, minority class instances are copied and repeated in the dataset until a more balanced distribution is reached. Thus, if there are two minority class instances and 100 majority class instances, traditional oversampling would copy the two minority class instances 49 times each. The resulting dataset would then consists of 200 instances: the 100 majority class instances and 100 minority class instances (i.e., 50 copies each of the two minority class instances).
While random undersampling and random oversampling create more balanced distributions, they both suffer from serious drawbacks. For example, in random undersampling (potentially), vast quantities of data are discarded. In the random undersampling example mentioned above, for instance, roughly 82% of the data (the 90 majority class instances) was discarded. This can be highly problematic, as the loss of such data can make the decision boundary between minority and majority instances harder to learn, resulting in a loss in classiflcation performance.
Alternatively, in random oversampling, instances are repeated (sometimes to very

high degrees). Consider the random oversampling example mentioned above, where each instance had to be replicated 49 times in order to balance out the class distribution. By copying instances in this way, one can cause drastic ovefitting to occur in the classifler, making the generalization performance of the classifler exceptionally poor. The potential for ovefitting is especially true as the class imbalance ratio becomes worse, and each instance must be replicated more and more often.
Thus, one major drawback of sampling techniques is that one needs to determine how much sampling to apply. An oversampling level must be chosen so as to promote the minority class, while avoiding ovefitting to the given data. Similarly, an undersampling level must be chosen so as to retain as much information about the majority class as possible, while promoting a balanced class distribution.
In order to overcome these limitations, more sophisticated techniques have been developed, e.g., SMOTE.

2.2 Synthetic Minority Oversampling Technique (SMOTE)

In order to overcome the issues mentioned above, (Chawla and Bowyer, 2002) developed a method of creating synthetic instances instead of merely copying existing instances in the dataset. This techniques is known as the synthetic minority oversampling technique (SMOTE). This approach is inspired by a technique that proved successful in handwritten character recognition (Ha and Bunke, 1997). They created extra training data by performing certain operations on real data. In their case, operations like rotation and skew were natural ways to perturb the training data. In SMOTE, the minority class is oversampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of oversampling required, neighbors from the k nearest neighbors are randomly chosen. There is an implementation that currently uses flve nearest neighbors. For instance, if the amount of oversampling needed is 200%, only two neighbors from the flve nearest neighbors are chosen and one sample is generated in the direction of each. Synthetic samples are generated in the following way:

1. Take the difference between the feature vector (sample) under consideration and its nearest neighbor;
2. Multiply this difference by a random number between 0 and 1;
3. Add it to the feature vector under consideration.

Thus, in SMOTE, the training set is altered by adding synthetically generated minority class instances, causing the class distribution to become more balanced. We say that the instances created are synthetic, as they are, in general, new minority instances that have been extrapolated and created out of existing minority class instances.

2.3 Cost-sensitive Learning

Cost sensitive learning is another commonly used method to handle classification problems with imbalanced data. In simple words, this method evaluates the cost associated with misclassifying observations.

It does not create balanced data distribution. Instead, it highlights the imbalanced learning problem by using cost matrices which describes the cost for misclassification in a particular scenario. These misclassification cost values can be given by domain experts, or can be learned via other approaches. Specifically, when dealing with imbalanced problems, it is usually more interesting to recognize the positive instances rather than the negative ones. Given the cost matrix, an example should be classified into the class that has the lowest expected cost, which is known as the minimum expected cost principle (Lopéz and Fernández, 2013). In the master's thesis, this method is not analyzed.

2.4 Accuracy Measure

A classifler is, typically, evaluated by a confusion matrix as illustrated in Table 2.1 (Lopéz and Fernández, 2013). In the confusion matrix, TN is the number of negative examples correctly classifled (True Negatives), FP is the number of negative examples incorrectly classifled as positive (False Positives), FN is the number of positive examples in correctly classifled as negative(False Negatives) and TP is the number of positive examples correctly classifled (True Positives). Predictive accuracy is deflned as $Accuracy = (TP+TN)/(TP+FP+TN+FN)$. However, predictive accuracy might not be appropriate when the data is imbalanced and/or the costs of different errors vary markedly. As an example, consider the classiflcation of pixels in mammogram images as possibly cancerous (Woods and Doss, 1993). A typical mammography dataset might contain 98% normal pixels and 2% abnormal pixels. A simple default strategy of guessing the majority class would give a predictive accuracy of 98%. The nature of the application requires a fairly high rate of correct

detection in the minority class and allows for a small error rate in the majority class in order to achieve this (Chawla and Bowyer, 2002). Simple predictive accuracy is clearly not appropriate in such situations.

Tab. 2.1: Confusion matrix for a two-class problem

		Predicted classes	
		Positive prediction	Negative prediction
True classes	Positive class	True Positive (TP)	False Negative (FN)
	Negative class	False Positive (FP)	True Negative (TN)

In imbalanced domains, the evaluation of the classifier's performance must be carried out using specific metrics in order to take into account the class distribution. Concretely, there are four metrics from Table 2.1 to measure the classification performance of both, positive and negative, classes independently:

1. True positive rate: $\%TP = TP/(TP + FN)$;
2. True negative rate: $\%TN = TN/(FP + TN)$;
3. False positive rate: $\%FP = FP/(TN + FP)$;
4. False negative rate: $\%FN = FN/(TP + FN)$.

There is a necessity of combining the individual measures of both the positive and negative classes, as none of these measures alone is adequate by itself.

A well-known approach to unify these measures and to produce an evaluation criteria is to use the Receiver Operating Characteristics (ROC) graphic. This graphic allows the visualization of the trade-off between the benefits and costs, as it evidences that any classifier cannot increase the number of true positives without also increasing the false positives. The Area Under the ROC Curve (AUC) provides a single measure of a classifier's performance for evaluating which model is better on average.

ROC curves can be thought of as representing the family of best decision boundaries for relative costs of TP and FP (Egan, 1975). On an ROC curve the X-axis represents $\%FP = FP/(TN + FP)$ and the Y-axis represents $\%TP = TP/(TP + FN)$ (see Fig. 2.1).

Points in (0, 0) and (100, 100) are trivial classifiers where the predicted class is always negative and positive one, respectively. The ideal point on the ROC curve

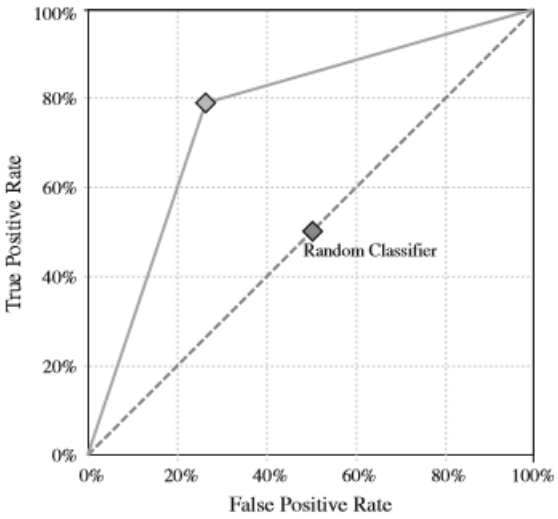

Fig. 2.1: Example of a ROC plot

would be (0,100), that is all positive examples are classifled correctly and no negative examples are misclassifled as positive.

The AUC measure is computed just by obtaining the area of the graphic:

$$AUC = (1 + TP - FP)/2$$

2.5 Conclusion

There are three main ways to deal with imbalanced datasets: data sampling, cost-sensitive learning, algorithmic modification. In this work, only data sampling will be studied thoroughly. It includes undersampling, oversampling, SMOTE.
Many studies have documented the weakness of the most notable threshold metric, accuracy, in comparison to ranking methods and metrics (most notably ROC analysis/AUC) in the case of class imbalances.

3

Classification Models

The general aim of supervised classiflcation algorithms is to separate the classes of the problem (with a margin as wide as possible) using only training data (Pérez-Ortiz and Jiménez-Fernández, 2016). If the output variable has two possible values, the problem is referred to as binary classiflcation. On the other hand, if there are more than two classes, the problem is named multiclass or multinomial classiflcation.

A classiflcation problem can be formally defined as the task of estimating the label y of a K-dimensional input vector $\mathbf{x}$, where $\mathbf{x} \in X \subseteq \mathbb{R}^K$ and $y \in \Omega = (\omega_1, \omega_2, ..., \omega_n)$. ω_1, ω_2,..., ω_n denote n classes. This task is accomplished by using a classiflcation rule or function $g : X \to \Omega$ able to predict the label of new patterns. In the supervised setting, we are given a training set of N_t points, represented by D, from which g will be adjusted, $D = (x_i, y_i), i = 1, ..., N_t$.

In order to compare different classification models for imbalanced datasets, cross-validation (CV) is performed. This approach involves randomly m-fold CV dividing the set of observations into m groups, or folds, of approximately equal size (James and Hastie, 2015).
The first fold is treated as a validation set, and the method is fit on the remaining $m - 1$ folds. The AUC is then computed on the observations in the held-out fold. This procedure is repeated m times; each time, a different group of observations is treated as a validation set. This process results in m estimates of AUC: AUC_1, AUC_2,..., AUC_m. The m-fold CV estimate is computed by averaging these values:

$$CV_{(m)} = \frac{1}{m} \sum_{i=1}^{m} AUC_i$$

In the work, 5-fold CV is performed in most of cases.

3.1 k-Nearest Neighbors

In k-NN classification, a point is classified by a majority vote of its neighbors, which means the point will be assigned to the class most common among its k nearest neighbors (Barber, 2012).
In other words, given a query point x_0, we find the k training points x_r, $r = 1,...,k$ closest in distance to x_0, and then classify using majority vote among the k neighbors.
In order to compute the distance between two points a and b, Euclidean distance can be used: $d(a,b) = \sqrt{\sum_i (a_i - b_i)^2}$.
For k-NN model we define k by CV.

3.2 Support Vector Machines

Support vector machine (SVM) is a statistical learning algorithm that aims to identify the optimal decision boundary between classes to minimize misclassification. Owing to its theoretical and practical advantages, such as solid mathematical background, high generalization capability, and ability to find global and nonlinear classiflcation solutions, SVM have been very popular among the machine learning and data-mining researchers (He and Ma, 2013).
The support vector machine (SVM) is an extension of the support vector classifler that results from enlarging the feature space in a speciflc way, using kernels, in order to accommodate a non-linear boundary between the classes. Thus, prior to classification, a kernel is typically applied to the input feature space to increase separability between classes (Hastie and Tibshirani, 2009).
In this study, SVMs with a radial basis function (RBF) kernel (Johnson and Tateishi, 2012) are used for classification. With the SVMs-RBF classifier, one can adjust the cost parameter (c) and the kernel spread function (γ) prior to classification. A wide range of paired combinations of c values and γ values ahould be tested. The optimal parameters for each of the SVMs classifications are selected by CV.
Although SVMs often work effectively with balanced datasets, they could produce suboptimal results with imbalanced datasets. More specfically, an SVM classifler trained on an imbalanced dataset often produces models that are biased toward the majority class and have low performance on the minority class. All the data preprocessing methods discussed in Chapter 2 can be used to balance the datasets before training the SVMs models. These methods include random and focused under/oversampling methods and synthetic data generation methods such as SMOTE. However, more sophisticated preprocessing techniques for SVMs exist. For example, the paper (Batuwita and Palade, 2010) presents an efflcient focused oversampling method for SVMs. In this method,first the separating hyperplane found by train-

ing an SVM model on the original imbalanced dataset is used to select the most informative examples for a given classification problem, which are the data points lying around the class boundary region. Then, only these selected examples are balanced by oversampling as opposed to blindly oversampling the complete dataset. This method reduces the SVM training time signficantly while obtaining comparable classification results to the original oversampling method.

3.3 Random Forest

The Random Forest approach starts with a standard decision tree technique but it takes this notion to the next level by creating a large number of decision trees (Benyamin, 2012). In this approach illustrated in Fig. 3.1, each tree is induced from a bootstrap sample of the training data, i.e., every observation is fed into every decision tree. Then, the most common outcome for each observation is calculated and used as the final output. Thus, prediction is made by aggregating (majority vote) the predictions of the ensemble.

In this model, there are some sources of randomness in order to make these trees different from one another. Therefore, this approach provides a group of unique trees which all make their classifications differently.

Random Forest generally exhibits a substantial performance improvement over the single tree classifier such as CART and C4.5 (Chen and Liaw, 2004). However, in learning extremely imbalanced data, there is a signiflcant probability that a bootstrap sample contains few or even none of the minority class, resulting in a tree with poor performance for predicting the minority class. One of the way of fixing this problem is to use a stratified bootstrap; i.e., sample with replacement from within each class.

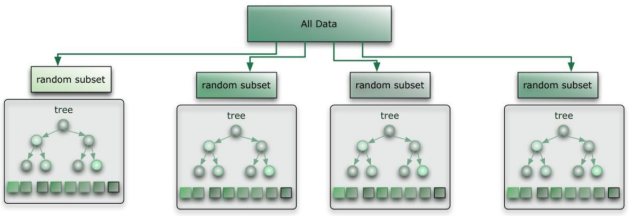

Fig. 3.1: Random Forest model

3.4 Conclusion

Three classification models (k-NN, SVMs-RBF, Random Forest) used for imbalanced datasets in this work are presented. It is shown that some of these classifiers can be oriented towards the class imbalance issues.

4 Logistic Regression

Many classification methods, including Neural Network classifiers or Logistic Regression models, provide estimates of the a posteriori probabilities of the classes. Following Bayes' rule (Hastie and Tibshirani, 2009), the a posteriori probabilities depend in a nonlinear way on the a priori probabilities. Therefore, the knowledge of the 'true' a priori probabilities in the real-world data is often highly desirable since optimal Bayesian decision making is based on the a posteriori probabilities of the classes given the observation (we have to select the class label that has the maximum estimated a posteriori probability) (Saerens, 2001). Therefore, a change of the a priori probabilities may have an important impact on the a posteriori probabilities of membership, which themselves affect the classification rate. From the preceding point, this means that applying such a classifier as-is on new data having different a priori probabilities from the training set can result in a loss of classification accuracy, in comparison with an equivalent classifier that relies on the 'true' a priori probabilities of the new dataset.

4.1 Estimation of a Posteriori Probabilities

In the master's thesis, in order to directly estimate the a posteriori probabilities of n classes, a Logistic Regression model is used. This model indeed ensures the probabilities sum to one and remain in $[0, 1]$ range (Hastie and Tibshirani, 2009). It separates the data by hyperplanes: if the classes are linearly separable, it will separate them and if the classes are not linearly separable, it will find the best linear separation according to the maximum likelihood criterion.

Mathematically it gives:

$$\mathbf{P}(w_n|\mathbf{x}) \approx \hat{y}_n(\mathbf{x}) = \frac{exp(\mathbf{w}_n^T \mathbf{x}')}{\sum_{j=1}^{q} exp(\mathbf{w}_n^T \mathbf{x}')}$$

where $\mathbf{x}' = [1, x_1, x_2, ..., x_p]^T$ is the vector x of p features belonging to one of the n possible classes $w_1, w_2, ..., w_n$. Noted also that the membership values $(\hat{y}_n)$ are

distributed according to a multinomial logistic distribution and, therefore,

$$0 \leq \hat{y}_n(\mathbf{x}) \leq 1 \text{ and } \sum_{n=1}^{q} \hat{y}_n(\mathbf{x}) = 1$$

In other words, the model takes the form

$$\log\left(\frac{\mathbf{P}(w_n|\mathbf{x})}{\mathbf{P}(w_{n'}|\mathbf{x})}\right) \approx (\mathbf{w}_n^T - \mathbf{w}_{n'}^T)\mathbf{x}'$$

That is, the log-odds of the posterior probabilities is linear in $\mathbf{x}$.

4.2 Adjustment for the New a Priori Probabilities

In the master's thesis, an iterative procedure that estimates the new a priori probabilities of a new data set and adjusts the outputs of a classifier (see (Saerens, 2001)) is evaluated on imbalanced datasets. The main goal is to show that this readjustment procedure can be beneficiall (i.e. it increases classification accuracy) when a classifier has been trained on a training set that does not reflect the true a priori probabilities of the classes in real-world conditions. The output readjustment procedure can only be applied if the classifier supplies estimates of the a posteriori probabilities (e.g., Logistic Regression).

Let us suppose a classiflcation problem in 2 classes with the class labels taking their value in $\Omega = (\omega_1, \omega_2)$. In order to train a classiflcation model, we rely on a training set, i.e. a collection of observation vectors, $\mathbf{x}_k$, measured on individuals and allocated to one of the 2 classes $\in \Omega$.
For building this training set, we suppose that, for each class ω_i, observations on N_t^i individuals belonging to the class (with $\sum_{i=1}^{n} N_t^i = N_t$, the total number of training examples) have been independently recorded according to the within-class probability density $p(\mathbf{x}|\omega_i)$. The a priori probability of belonging to class ω_i in the training set will be denoted as $p_t(\omega_i)$ (subscript *t* will be used for estimates carried out on the basis of the training set) and is therefore estimated by the class frequency $\hat{p}_t(\omega_i) = \frac{N_t^i}{N_t}$. Let us now assume that the classiflcation model has been trained, i.e. its parameters have been estimated on the basis of the training set (as indicated by subscript *t*). The classiflcation model could be an artiflcial neural network, a Logistic

Regression, or any other model that provides as outputs estimates of the a posteriori probabilities of the classes given the observations. We therefore assume that the model has 2 outputs, $g_i(x)$ ($i = 1, 2$), providing estimated a posteriori probabilities of membership $\hat{p}_t(\omega_i|\mathbf{x}) = g_i(\mathbf{x})$ (i.e., probability of belonging to class ω_i, given that observation vector $\mathbf{x}$ has been observed) in the conditions of the training set.

There will be considered two cases:

1. An estimate of the new a priori probabilities is known. In other words, the classifier will be applied on a large number of datasets for which we know the probability distribution of the priors.

2. No estimate of the new a priori probabilities is known. In this case, no class can be favored over any other class.

4.3 An Estimate of the New a Priori Probabilities Is Known

Let us now suppose that the trained classification model has to be applied to another dataset from which the class frequencies, estimating the a priori probabilities $p(\omega_i)$ are known. One should make the natural assumption that the generation of the observations within the classes, and thus each within-class density, does not change from the training set to the new dataset ($\hat{p}_t(\mathbf{x}|\omega_i) = \hat{p}(\mathbf{x}|\omega_i)$), only the number of measurements observed from each class has changed. The corrected a posteriori probabilities $\hat{p}(\omega_i|\mathbf{x})$ can be calculated according to the formula (Saerens, 2001):

$$\hat{p}(\omega_i \mid \mathbf{x}) = \frac{\frac{\hat{p}(\omega_i)}{\hat{p}_t(\omega_i)}\hat{p}_t(\omega_i|\mathbf{x})}{\sum_{j=1}^{n}\frac{\hat{p}(\omega_j)}{\hat{p}_t(\omega_j)}\hat{p}_t(\omega_j|\mathbf{x})}$$

4.4 No Estimate of the New a Priori Probabilities Is Known

In many real-world situations, we ignore what the real-world a priori probabilities $p(\omega_i)$ are since we do not know the class labels for these new data (Saerens, 2001).

However, there is an iterative procedure for a priori and a posteriori probabilities adjustment which is based on the EM algorithm (Dempster and Laird, 1977). Let us deflne $g_i(\mathbf{x}_k)$ as the model's output value corresponding to class ω_i for observation $\mathbf{x}_k$ from the new data set to be scored (Saerens, 2001). The model's outputs provide an approximation of the a posteriori probabilities of the classes given the observation in the conditions of the training set (subscript *t*), while the a priori probabilities are estimated by the class frequencies:

$$\hat{p}_t(\omega_i \mid \mathbf{x}_k) = g_i(\mathbf{x}_k)$$
$$\hat{p}_t(\omega_i) = \frac{N_t^i}{N_t}$$

Let us deflne as $p^{(s)}(\omega_i)$ and $p^{(s)}(\omega_i|\mathbf{x}_k)$ the estimates of the new a priori and a posteriori probabilities at step *s* of the iterative procedure. If the $p^{(s)}(\omega_i)$ probabilities are initialized by the frequencies of the classes in the training set, the EM algorithm provides the following iterative steps (Saerens, 2001), for each new observation $\mathbf{x}_k$ and each class ω_i:

$$\hat{p}^{(0)}(\omega_i) = \hat{p}_t(\omega_i)$$
$$\hat{p}^{(s)}(\omega_i \mid \mathbf{x}_k) = \frac{\frac{\hat{p}^{(s)}(\omega_i)}{\hat{p}_t(\omega_i)}\hat{p}_t(\omega_i|\mathbf{x}_k)}{\sum_{j=1}^{n}\frac{\hat{p}^{(s)}(\omega_j)}{\hat{p}_t(\omega_j)}\hat{p}_t(\omega_j|\mathbf{x}_k)}$$
$$\hat{p}^{(s+1)}(\omega_i) = \frac{1}{N}\sum_{k=1}^{N}\hat{p}^{(s)}(\omega_i \mid \mathbf{x}_k)$$

At each iteration step *s*, both the a posteriori ($p^{(s)}(\omega_i|\mathbf{x}_k)$) and the a priori probabilities ($p^{(s)}(\omega_i)$) are re-estimated sequentially for each observation $\mathbf{x}_k$ and each class ω_i. The iterative procedure proceeds until the convergence of the estimated probabilities, ω_i.

4.5 Conclusion

A special focus is given to the investigation of some ways of dealing with imbalanced datasets based on a Logistic Regression. An iterative procedure that estimates the new a priori probabilities of a new dataset and adjusts the outputs of a classifier is proposed for evaluation on imbalanced datasets. The procedure is an instance of the EM algorithm and is applied here to the outputs of a Logistic Regression to improve classification accuracy.

5 Results

In order to achieve well founded conclusions, let us make use of four classifiers based on diffferent paradigms: k-NN as an instance-based learning approach, SVMs, Random Forest that is an extension of a standard decision tree technique, Logistic Regression that gives as an output the estimates of a posteriori probabilities. The analysis is structured in the following way: first, the experimental framework is introduced, that is, the classification algorithms, their parameters and the selected datasets for the study; next, a separate study for preprocessing techniques used is developed.

First, one need to define a set of baseline classifiers to be used in all the experiments. Next, the parameter values used are given. Baseline classifiers are presented below:

1. k-NN: in this case, the number of neighbors for determining the output class was selected by CV (the euclidean distance metric was applied).

2. SVMs: RBFs with the cost parameter (c) and the kernel spread function (γ) were chosen by CV.

3. Random Forest: the parameter $ntree$ (number of decision trees) was set equal to 100.

4. Logistic Regression.

Four imbalanced datasets have been gathered, whose features are summarized in Table 5.1, namely the number of observations ($\#Obs.$), number of variables ($\#Var.$) and IR. Estimates of the AUC metric were obtained by means of a 5-fold CV. That is, the dataset was split into 5 folds, each one containing 20% of the patterns of the dartaset. For each fold, the alghoritm was trained with the examples contained in the remaining folds and then tested with the current fold. This value is set up with the aim of having enough positive class instances in the different folds, hence avoiding additional problems in the data distribution, especially for highly imbalanced datasets. The datasets are available for download at the KEEL (Knowledge

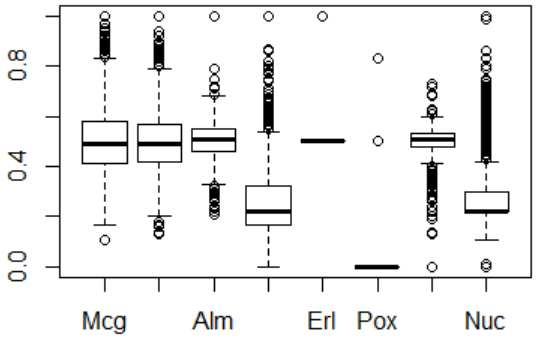

Fig. 5.1: Boxplot of the features of $yeast$ datasets

Extraction Evolutionary Learning) dataset repository (KEEL dataset repository, 2017) that include different binary class datasets with imbalance ratio 1.5 – 9, higher than 9, and multiple class imbalanced datasets.

Table 5.1 presents *yeast* datasets with the same number of observations and variables but different IR.

Tab. 5.1: Summary of imbalanced datasets used (KEEL dataset repository)

Name	$\#Obs.$	$\#Var.$	IR
$yeast3$	1484	8	8.1
$yeast4$	1484	8	28.41
$yeast5$	1484	8	32.78
$yeast6$	1484	8	39.15

In total, eight features Mcg, Gvh, Alm, Mit, Erl, Pox, Vac, Nuc were used for classification. They are displayed as a boxplot in Fig. 5.1.

Before applying any of resampling techniques, the standardization of the data was performed. It means the data was scaled and centered to have mean zero. The result of this operation is shown in Fig. 5.2.

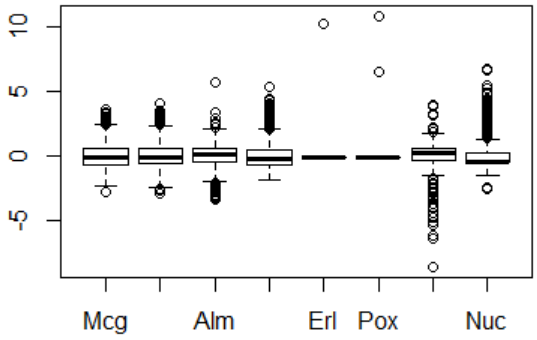

Fig. 5.2: Boxplot of the features of $yeast$ datasets after scaling and centering

5.1 Study on the Preprocessing Methods

In this section, the behavior of the preprocessing methods on imbalanced datasets is analyzed. The representative set of methods is composed by the following techniques:

1. Oversampling;
2. Undersampling;
3. Both oversampling & undersampling;
4. SMOTE.

All those methods can be implemented in R using the parameters presented in Table 5.2). The data generated from oversampling have expected amount of repeated observations. Data generated from undersampling is deprived of important information from the original data. To encounter these issues, SMOTE helps us to generate extra positive observations synthetically as well.

Tab. 5.2: Parameters of the preprocessing methods

Name	$Undersampling$	$Oversampling$	$Both$	$SMOTE$
$yeast3$	N=326	N=2642	N=1000, p=0.5	$perc.over$=100, $perc.under$=200
$yeast4$	N=102	N=2866	N=1000, p=0.5	$perc.over$=100, $perc.under$=200
$yeast5$	N=88	N=2880	N=1000, p=0.5	$perc.over$=100, $perc.under$=200
$yeast6$	N=70	N=2898	N=1000, p=0.5	$perc.over$=100, $perc.under$=200

N refers to the number of observations in the resulting balanced set. For instance, for $yeast3$ dataset, originally there were 163 negative observations. Therefore, with total number N equal to 326, in the resulting balanced dataset (after undersampling) there will be 163 negative and 163 positive instances. The parameter p refers to the probability of positive class in newly generated sample. We set $perc.over = 100$ to double the quantity of positive cases, and set $perc.under = 200$ to keep half of what was created as negative cases.

Imbalanced Classification in R

In R, packages such as $DMwR$ or $ROSE$ perform sampling strategies quickly. $ROSE$ (Random Over Sampling Examples) package provides a function named $ovun.sample$ which enables oversampling, undersampling in one go. This package has well defined accuracy functions to do the tasks quickly. One have to set a desirable resampling method in the function $ovun.sample$ (e.g., $"under"$ for undersampling, $"over"$ for oversampling, $"both"$ if both undersampling and oversampling are used). If we do both undersampling and oversampling on the imbalanced data, the minority class is oversampled with replacement and majority class is undersampled without replacement.

Package $DMwR$ allows implementation of SMOTE algorithm in R that generate data synthetically.

First, let us check the accuracy of the prediction using ROC curve without applying any sampling techniques. This will give us a clear picture, if the model is worth. Using the function $roc.curve$ available in this package, for $yeast4$ dataset after running Random Forest" model we get $AUC = 0.5677$ which is a terribly low score. In this case, the algorithm gets biased toward the majority class and fails to map minority class. Therefore, it is necessary to balance data before applying a machine learning algorithm. We willl use oversampling and undesampling techniques as well as SMOTE algorithm and balance datasets trying to improve this prediction accuracy. The results of using 4 techniques to balance $yeast4$ dataset are presented in Fig. 5.3 for Random Forest" model, in Fig. 5.4 for k-NN, in Fig. 5.5 for SVMs-RBF.

From these figures, it is clear that sampling techniques has improved dramatically the performance of all the analyzed classifiers.

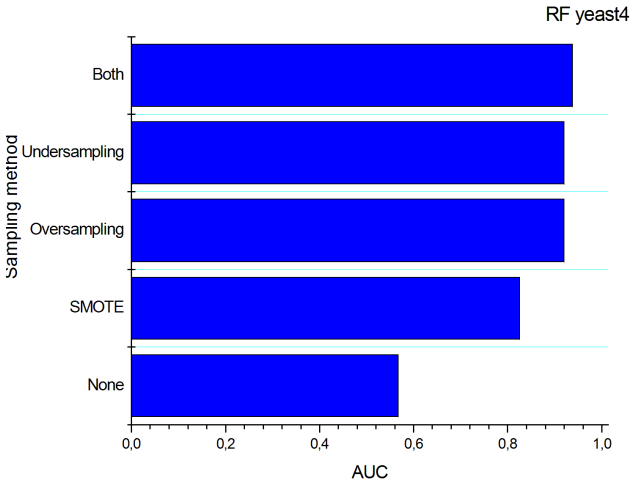

Fig. 5.3: Performance of Random Forest classifer after applying sampling techniques for $yeast4$ dataset

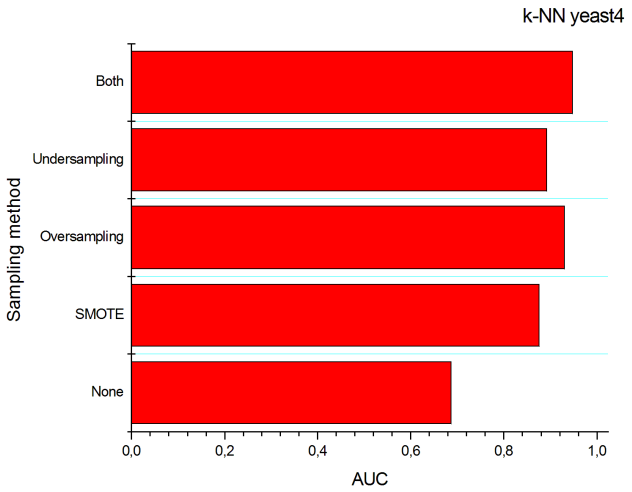

Fig. 5.4: Performance of k-NN classifer after applying sampling techniques for $yeast4$ dataset

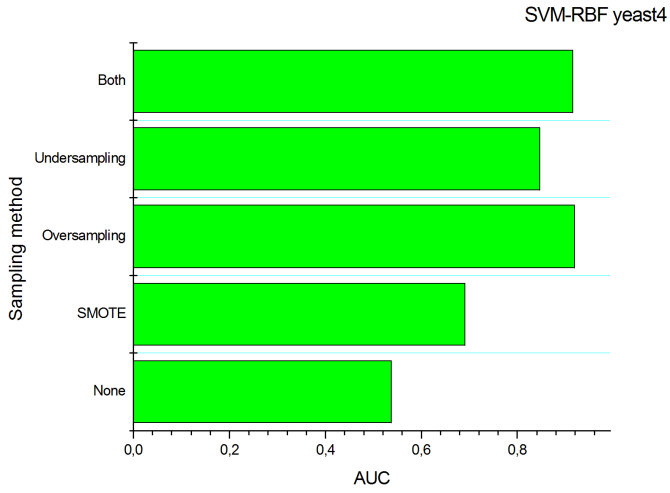

Fig. 5.5: Performance of SVMs-RBF classifer after applying sampling techniques for $yeast4$ dataset

5.2 Logistic Regression

For Logistic Regression, two cases have to be considered:

1. An estimate of new priors is known.
2. No estimate of new priors is known.

A full algorithm for classification in both cases consists of the following steps:

1. Preprocessing (e.g., oversampling); output: new priors.
2. Running Logistic Regression; output: a posteriori probabilities.
3. Adjustment of outputs (e.g., using iterative procedure if no estimate of new priors is known).

Let us demonstrate how these steps work on the example of $yeast3$ dataset.

Initially, the dataset includes 1484 observations with the following number of positive and negative instances:
$negative/positive$: 1321/163

Thus, initial priors for a chosen dataset are as follows:
$negative/positive$: 0.89/0.11

All 1484 observations have been split into 5 folds (for 5-fold CV) with the number and distribution of instances in each fold presented in Table 5.3:

Tab. 5.3: Number and distribution of instances in the folds

	$Fold1$	$Fold2$	$Fold3$	$Fold4$	$Fold5$
$total$	297	298	296	296	297
$negative$	264	265	264	264	264
$positive$	33	33	32	32	33

After apllying a preprocessing technique (oversampling), the number of instances in the dataset has increasd up to 2642 with the following numbers of positive and negative instances:
$negative/positive$: 1057/1585

Thus, the new priors are as follows:
$negative/positive$: 0.4/0.6

The following procedure will be shown on the example of observation #216. After running Logistic Regression it has a posteriori probabilty of belonging to a positive class equal to 0.8828.

Then, we can apply the following formula to calculate adjusted a posteriori probability:

$$\hat{p}(\omega_i \mid \mathbf{x}) = \frac{\frac{\hat{p}(\omega_i)}{\hat{p}_t(\omega_i)}\hat{p}_t(\omega_i|\mathbf{x})}{\sum_{j=1}^{n}\frac{\hat{p}(\omega_j)}{\hat{p}_t(\omega_j)}\hat{p}_t(\omega_j|\mathbf{x})}$$

Here, $\hat{p}(\omega_i \mid \mathbf{x})$ is an adjusted a posteriori probabiluty of belonging to class ω_i, $\hat{p}(\omega_i)$ is initial a priori probability of belonging to class ω_i, $\hat{p}_t(\omega_i)$ is a new a priori probability (in our case, after oversampling), $\hat{p}_t(\omega_i \mid \mathbf{x})$ is a posteriori probability (in our case, the output of Logistic Regression prior to adjustment).

In our example, for the observation #216, $\hat{p}(\omega_1) = 0.89$, $\hat{p}(\omega_2) = 0.11$ ($\omega_1 = negative$, $\omega_2 = positive$), $\hat{p}_t(\omega_1) = 0.4$, $\hat{p}_t(\omega_2) = 0.6$, $\hat{p}_t(\omega_1 \mid \mathbf{x}) = 0.1172$, $\hat{p}_t(\omega_2 \mid \mathbf{x}) = 0.8828$.

The adjusted a posteriori probability $\hat{p}(\omega_i \mid \mathbf{x})$ for the observation #216 is now equal to 0.3857. It means that after adjustment this observation should be labeled as belonging to a negative class ($0.3857 < 0.5$).

Let us now assume we have no estimates of new priors. Then, the iterative procedure have to be performed to estimate new priors. It is useful to understand how many iterations would be sufficient to achieve convergence. The condition for convergence is as follows:

$$\sum_{i=1}^{n} \mid \hat{p}^{(s+1)}(\omega_i) - \hat{p}^{(s)}(\omega_i) \mid < 0.0001$$

For $yeast3$ dataset, Fig. 5.6 shows that convergence has been achieved after six iterations.

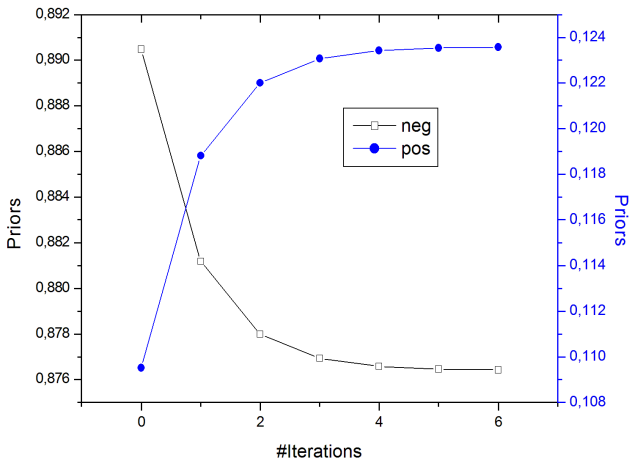

Fig. 5.6: New priors and number of iterations

The performance of Logistic Regression after using 4 techniques to balance $yeast3$ dataset is presented in Fig. 5.7, for $yeast4$ - in Fig. 5.8, for $yeast5$ - in Fig. 5.9, for $yeast6$ - in Fig. 5.10. LR denote Logistic Regression without adjustment for priors, TP means "True priors" (it corresponds to the case when priors are known),

$Log - EM$ corresponds to the case with not known priors. The results are also available in Tables 5.4 - 5.8.

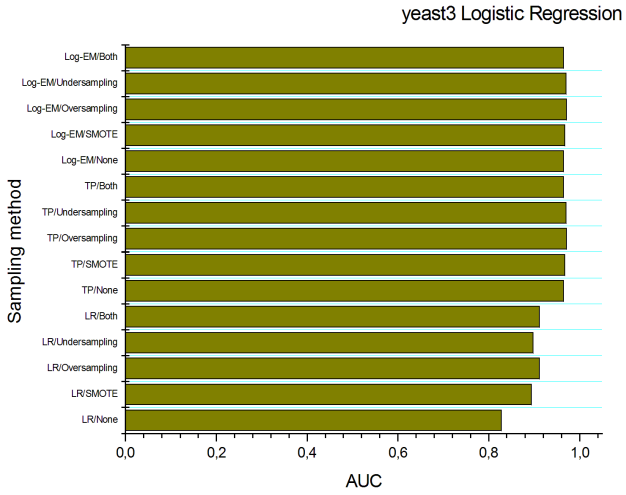

Fig. 5.7: Performance of Logistic Regression after applying sampling techniques and adjustment of priors for $yeast3$ dataset

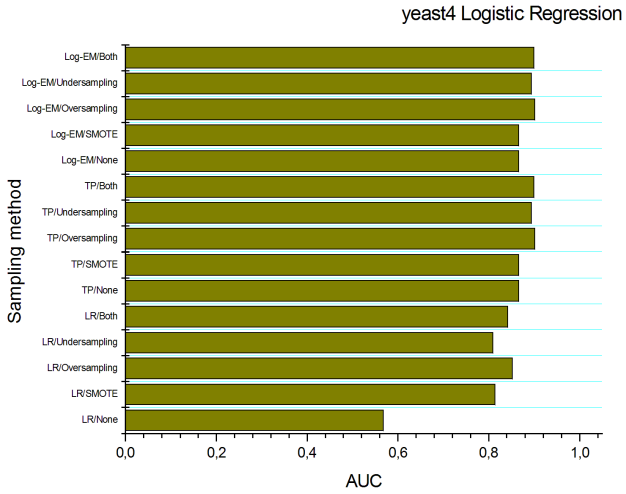

Fig. 5.8: Performance of Logistic Regression after applying sampling techniques and adjustment of priors for $yeast4$ dataset

5.3 Results for All Considered Classification Models and Preprocessing Techniques

The results for $yeast3$ dataset for all considered classification models and preprocessing techniques are presented in Table 5.4. $AUC(Test)$ means that the alghoritm was trained with the examples contained in the remaining folds and then tested with

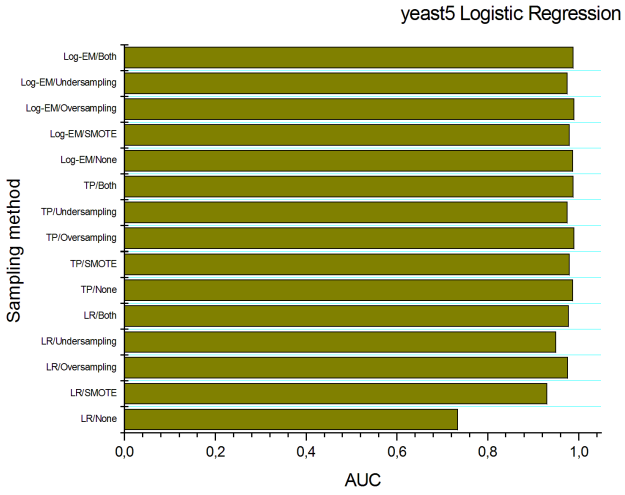

Fig. 5.9: Performance of Logistic Regression after applying sampling techniques and adjustment of priors for $yeast5$ dataset

the current fold to which the accuracy measure AUC is applied.

Tab. 5.4: Results for $yeast3$ dataset (after 5-fold CV)

yeast3	Logistic Regression	True priors	Log-EM (priors are not known)	Random Forest	kNN	SVM-RBF
Preprocessing	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)
None	0.8286	0.9644	0.9644	0.8422	0.8458	0.8642
SMOTE	0.8935	0.967	0.967	0.9189	0.9126	0.9056
Oversampling	0.9112	0.9715	0.9715	**0.9829**	0.9692	0.9272
Undersampling	0.8974	0.9703	0.9703	0.9695	0.9344	0.9153
Both	0.9113	0.9648	0.9648	0.9724	0.9438	0.9148

SMOTE: perc.over = 100, perc.under = 200; Oversampling: N=2642; Undersampling: N=326; Both: N=1000, p=0.5

For $yeast3$ dataset, a ROC curve after running Logistic Regression with adjustment for not known priors is presented in Fig. 5.11. For $yeast4$, $yeast5$, $yeast6$ datasets ROC curves are similar.

Similar analysis was performed for $yeast4$ dataset (Table 5.5), $yeast5$ dataset (Table 5.6), $yeast6$ dataset (Table 5.7).

Tab. 5.5: Results for $yeast4$ dataset (after5-fold CV)

yeast4	Logistic Regression	True priors	Log-EM (priors are not known)	Random Forest	kNN	SVM-RBF
Preprocessing	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)
None	0.5682	0.8669	0.8669	0.5677	0.6868	0.538
SMOTE	0.8144	0.867	0.8669	0.8257	0.8752	0.6909
Oversampling	0.8523	0.9013	0.9013	0.9193	0.9304	0.9193
Undersampling	0.8098	0.8941	0.8941	0.9199	0.8917	0.8468
Both	0.8427	0.8995	0.8994	0.9381	**0.9479**	0.9151

SMOTE: perc.over = 100, perc.under = 200; Oversampling: N=2866; Undersampling: N=102; Both: N=1000, p=0.5

Table 5.8 shows the average AUC results where the best average values per algorithm group are highlighted in bold. From this table, one may conclude the goodness of the use of this type of solution for imbalanced data, as there is a significant difference with respect to the results obtained with the original data.

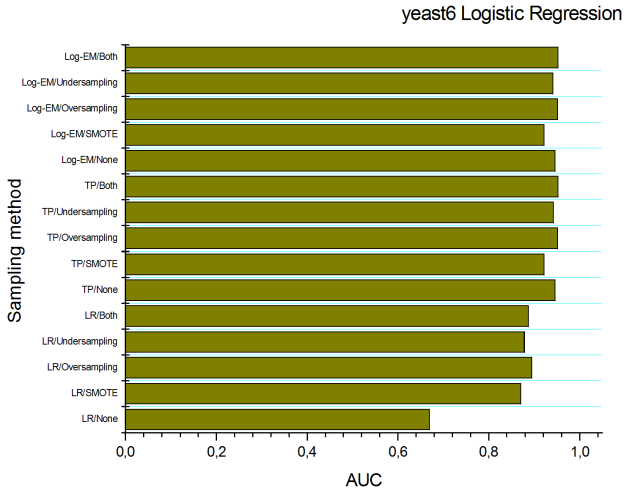

Fig. 5.10: Performance of Logistic Regression after applying sampling techniques and adjustment of priors for $yeast6$ dataset

Tab. 5.6: Results for $yeast5$ dataset (after 5-fold CV)

yeast5	Logistic Regression	True priors	Log-EM (priors are not known)	Random Forest	kNN	SVM-RBF
Preprocessing	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)
None	0.7337	0.9871	0.987	0.7799	0.844	0.7424
SMOTE	0.9299	0.9795	0.9795	0.9434	0.9597	0.9545
Oversampling	0.9757	**0.9894**	**0.9894**	0.9868	0.9878	0.9319
Undersampling	0.95	0.9751	0.9751	0.9691	0.9712	0.9708
Both	0.9777	0.9883	0.9883	0.9826	0.9812	0.9316

SMOTE: perc.over = 100, perc.under = 200; Oversampling: N=2880; Undersampling: N=88; Both: N=1000, p=0.5

5.4 Wilcoxon Signed-rank Test

In order to compare the outcomes of different preprocessing techniques and/or classification models, the Wilcoxon signed-rank test is performed. It is a non-parametric statistical hypothesis test used when comparing two related samples, matched samples, or repeated measurements on a single sample to assess whether their population mean ranks differ (i.e. it is a paired difference test). It can be used to determine whether two dependent samples were selected from populations having the same distribution. (Lowry, 1998).

Tab. 5.7: Results for $yeast6$ dataset (after 5-fold CV)

yeast6	Logistic Regression	True priors	Log-EM (priors are not known)	Random Forest	kNN	SVM-RBF
Preprocessing	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)
None	0.6694	0.9461	0.9461	0.7126	0.7826	0.6547
SMOTE	0.871	0.9218	0.9218	0.8736	0.8812	0.8297
Oversampling	0.895	0.9513	0.9513	0.9547	0.9537	0.9279
Undersampling	0.8788	0.9419	0.9416	0.9276	0.9	0.876
Both	0.8875	0.9521	0.952	0.9509	**0.9593**	0.9265

SMOTE: perc.over = 100, perc.under = 200; Oversampling: N=2898; Undersampling: N=70; Both: N=1000, p=0.5

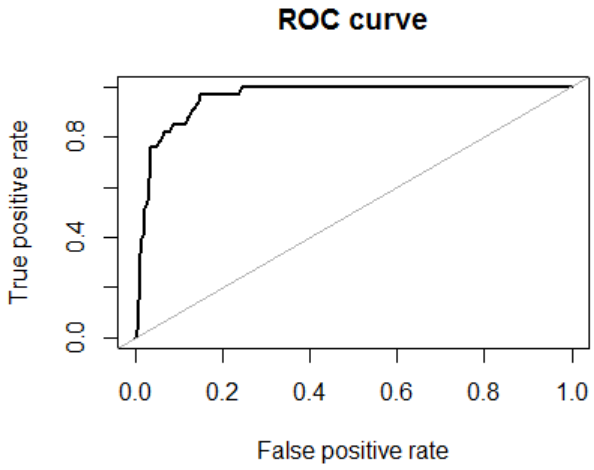

Fig. 5.11: ROC curve for $yeast3$ dataset after running Logistic Regression with adjustment for not known priors

Tab. 5.8: Average AUC results for all studied datasets

yeast6	Logistic Regression	True priors	Log-EM (priors are not known)	Random Forest	kNN	SVM-RBF
Preprocessing	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)	AUC (*Test*)
None	0.7000 ± 0.1859	0.9411 ± 0.0887	0.9411 ± 0.0886	0.7256 ± 0.2000	0.7898 ± 0.1268	0.699825 ± 0.234085
SMOTE	0.8772 ± 0.0821	0.9338 ± 0.0865	0.9338 ± 0.0865	0.8904 ± 0.0881	0.9072 ± 0.0656	0.845175 ± 0.195124
Oversampling	**0.9086 ± 0.0869**	**0.9534 ± 0.0646**	**0.9534 ± 0.0646**	0.9609 ± 0.0530	**0.9603 ± 0.0413**	**0.926575 ± 0.008951**
Undersampling	0.88840 ± 0.0983	0.9454 ± 0.0631	0.9453 ± 0.0631	0.9465 ± 0.0450	0.9243 ± 0.0616	0.902225 ± 0.091057
Both	0.9048 ± 0.0956	0.9521 ± 0.0638	0.9511 ± 0.0639	**0.9610 ± 0.0343**	0.9581 ± 0.0285	0.922 ± 0.014264

Let N be the sample size, i.e., the number of pairs. Thus, there are a total of $2N$ data points. For pairs $i = 1, ..., N$, let $x_{1,i}$ and $x_{2,i}$ denote the measurements. H_0 and H_1 denote the hypotheses tested.

H_0: difference between the pairs follows a symmetric distribution around zero.
H_1: difference between the pairs does not follow a symmetric distribution around zero.

An obtained value of Wilcoxon's test statistic W is calculated as follows:

$$W = \sum_{i=1}^{N} [sgn(x_{2,i} - x_{1,i}) \cdot R_i]$$

R_i denote the rank. $W - value$ should be compared to the critical value in the table (see Appendix I) taking into account the number of subjects N. The obtained value is statistically significiant if it is equal to or smaller than the value in the table.

For instance, suppose an obtained value is 22, and there were 15 participants. The critical value in the table is 25: the obtained value is smaller than this, and so one can conclude that the difference between the two conditions in the study was unlikely to occur by chance ($p < 0.05$ two-tailed test, or $p < 0.025$, one-tailed test).

For two methods (Log-EM without preprocessing (Method 1) vs. Log-EM with oversampling (Method 2)) Table 5.9 shows the results of Wilcoxon signed-rank test. $f1$, $f2$, $f3$, $f4$, $f5$ denote CV folds, Abs is an absolute value of difference between the outcomes of Method 1 and Method 2, $Sign$ shows if Method 1 is superior than Method 2 ($Sign = 1$) or not ($Sign = -1$).

Tab. 5.9: Results (Wilcoxon)

	$Method1$	$Method2$	$Sign$	Abs	R	$SignR$
$yeast3 - f1$	0.9776472	0.9799516	-1	0.0023	9	-9
$yeast3 - f2$	0.9327365	0.959596	-1	0.0269	18	-18
$yeast3 - f3$	0.9774372	0.9793529	-1	0.0019	7	-7
$yeast3 - f4$	0.9657746	0.9701493	-1	0.0044	12	-12
$yeast3 - f5$	0.9580189	0.960908	-1	0.0029	10	-10
$yeast4 - f1$	0.8947552	0.9374126	-1	0.0427	19	-19
$yeast4 - f2$	0.9229965	0.9304878	-1	0.0075	14	-14
$yeast4 - f3$	0.7857597	0.9203751	-1	0.1346	20	-20
$yeast4 - f4$	0.8482578	0.8489547	-1	0.0007	3	-3
$yeast4 - f5$	0.8829268	0.871777	1	0.0111	16	16
$yeast5 - f1$	0.9870756	0.9878472	-1	0.0008	4	-4
$yeast5 - f2$	0.9789738	0.9801312	-1	0.0012	5	-5
$yeast5 - f3$	0.9921875	0.9915365	1	0.0007	2	2
$yeast5 - f4$	0.9886188	0.9940201	-1	0.0054	13	-13
$yeast5 - f5$	0.9884259	0.9886188	-1	0.0002	1	-1
$yeast6 - f1$	0.8984182	0.9087988	-1	0.0104	15	-15
$yeast6 - f2$	0.9926108	0.9889163	1	0.0037	11	11
$yeast6 - f3$	0.9568966	0.958867	-1	0.002	8	-8
$yeast6 - f4$	0.9184729	0.9337438	-1	0.0153	17	-17
$yeast6 - f5$	0.9642857	0.9655172	-1	0.0012	6	-6

The $W - value$ is 29. The critical value of W for $N = 20$ at $p \leq 0.05$ is 52 (see Appendix I). Therefore, the result is significant at $p \leq 0.05$ and Method 2 therefore revealed a higher accuracy than Method 1.

Similarly, other hypotheses can be tested. In this work, Random Forest (Method 1) and Log-EM (Method 2) were also compared. It turned out that Log-EM outperformed Random Forest (the corresponding $W - value$ is 1) (both methods were compared without applying sampling techniques). However, when oversampling is applied, it is impossible to make a conclusion whether this method performs better

than Random Forest with oversampling ($W - value$ is 71 that is higher than critical value).

5.5 Dependence of Number of Iterations on Accuracy

In the paper (Gao and Sebastiani, 2016) the authors obtained the results showing that iterations beyond the first, which were carried out in order to obtain convergence, were actually detrimential, instead of benificial, to the accuracy. The authors do not mention which accuracy measure (AUC or classification accuracy based on confusion matrix) should be used. In order to theck this statement, the dependence of number of iterations on accuracy has been studied on $yeast3$ and $yeast5$ datasets. Both AUC and classification accuracy based on confusion matrix were calculated.

For $yeast3$, AUC revealed the same value (0.9644) for both 1 iteration and multiple iterations (i.e., until convergence has been obtained) whereas AUC value was a bit higher for $yeast5$ with 1 iteration (0.9871) than for the same dataset with miltiple iterations (0.9870).

In contrast, the classification accuracy (based on confusion matrix) was lower for $yeast3$ with 1 iteration (0.9474) than for $yeast3$ with multiple iterations (0.9488) when convergence was achieved and higher for $yeast5$ with 1 iteration (0.9771) than for $yeast5$ with multiple iterations (0.9744). Sampling techniques were not applied. Thus, the impact of number of iterations on accuracy is ambiguous and additional studies are required to better understand the dependence of given accuracy measure (e.g., AUC) against number of iterations.

5.6 Real-world Dataset

The dataset taken from Kaggle dataset repository (Kaggle dataset repository, 2016) contains transactions made by credit cards in September 2013 by European cardholders. The dataset has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection. This dataset presents transactions that occurred in two days, where 492 frauds out of 284,807 transactions were detected. The dataset is highly imbalanced, the positive class

(frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a principal component analysis (PCA) transformation (James and Hastie, 2015). Unfortunately, due to confidentiality issues, the authors cannot provide the original features and more background information about the data.

PCA is an unsupervised approach that refers to the process by which principal components are computed, and the subsequent use of these components in understanding the data. It finds a low-dimensional representation of a dataset that contains as much as possible of the variation. The idea is that each of the N observations lives in p-dimensional space, but not all of these dimensions are equally interesting.

Features $V1$, $V2$, ... $V28$ (see Fig. 5.12) are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction amount. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

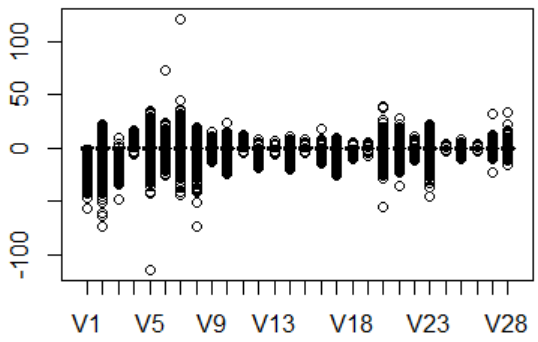

Fig. 5.12: PCA features of 'Credit Card Fraud Detection' dataset

The observations have been split into 3 folds (for 3-fold CV) with the number and distribution of instances in each fold presented in Table 5.10:

Tab. 5.10: Number and distribution of instances in the folds for 'Credit Card Fraud Detection' dataset

	$Fold1$	$Fold2$	$Fold3$
$total$	94936	94935	94936
$negative$	94772	94771	94772
$positive$	164	164	164

Given the class imbalance ratio, the accuracy has been measured using the Area Under the ROC curve (AUC). Figure 5.13 shows the ROC curve after running Logistic Regression without adjustment for priors. AUC is equal to 0.8007 that demonstrates rather poor prediction accuracy of the algorithm.

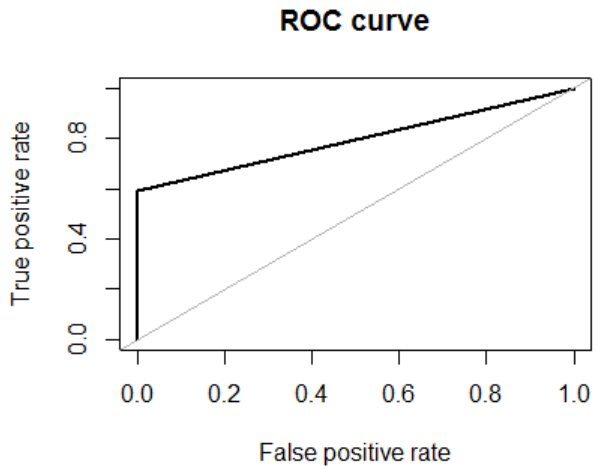

Fig. 5.13: ROC curve for 'Credit Card Fraud Detection' dataset after running Logistic Regression without adjustment for priors

Figure 5.14 shows the ROC curve after undersampling and running Logistic Regression with adjustment for not known priors (Log-EM). AUC is equal to 0.9861.

Obviously, confusion matrix accuracy is not meaningful for imbalanced classification but for reference one can show the confusion matrix for the model Log-EM/Undersampling (Table 5.11). It gives the accuracy of 0.9943.

Tab. 5.11: Confusion matrix for the model Log-EM/Undersampling

Log-EM/Undersampling		Predicted classes	
		0	1
True classes	0	282785	1530
	1	78	414

The performance of classification models after applying four techniques to balance 'Credit Card Fraud Detection' dataset is presented in Fig. 5.15. The performance

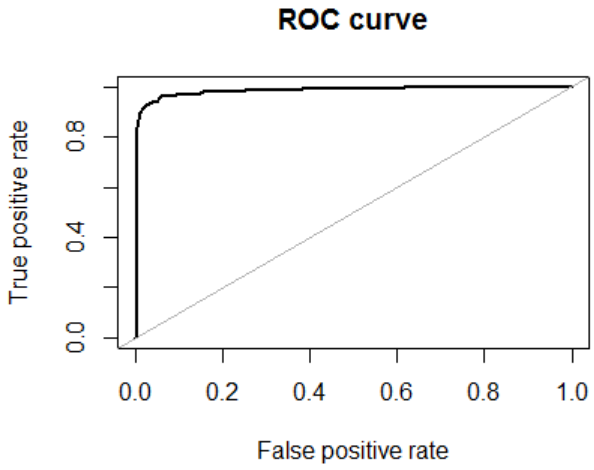

Fig. 5.14: ROC curve for 'Credit Card Fraud Detection' dataset after undersampling and running Logistic Regression with adjustment for not known priors (Log-EM)

was calculated after 3-fold CV. Note that for Random Forest model the parameter $ntree$ was set equal to 10 to reduce computational time. For the same reason, oversampling was not performed for Random Forest and k-NN models. Figure shows that for studied dataset the highest accuracy demonstrated Log-EM and TP models with different sampling techniques. The second best model is the Random Forest with undersampling.

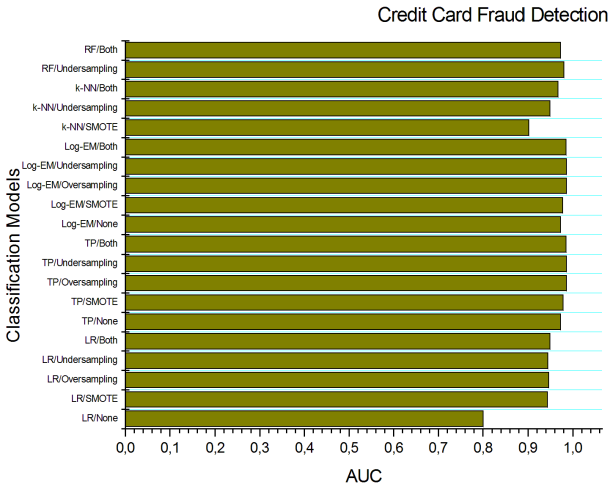

Fig. 5.15: Performance of classification models and sampling techniques

It is important to note that for any dataset with large number of observations, CPU time required for the implementation of algorithm plays crucial role. The comparison of CPU time of Intel Core i3-4030U @ 1.90GHz for studied methods is presented in

Fig.5.16. The combination of undersampling and Logistic Regression revealed the shortest CPU time that can be considered as an advantage of this approach. On the other hand, Logistic Regression with oversampling revealed the longest CPU time (more than 450 s).

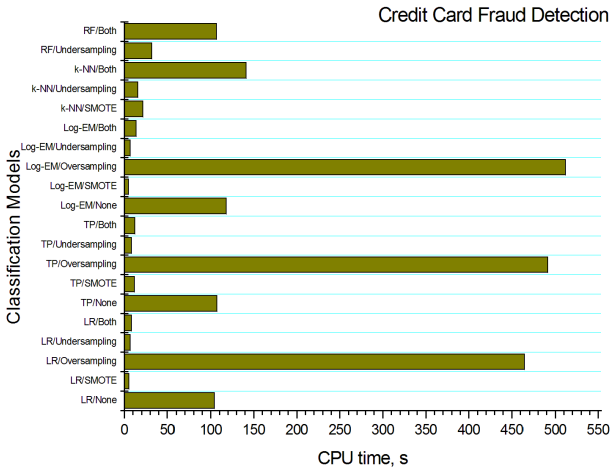

Fig. 5.16: CPU time

In most of cases, eight iterations are enough for the convergence of Log-EM algorithm (Fig. 5.17).

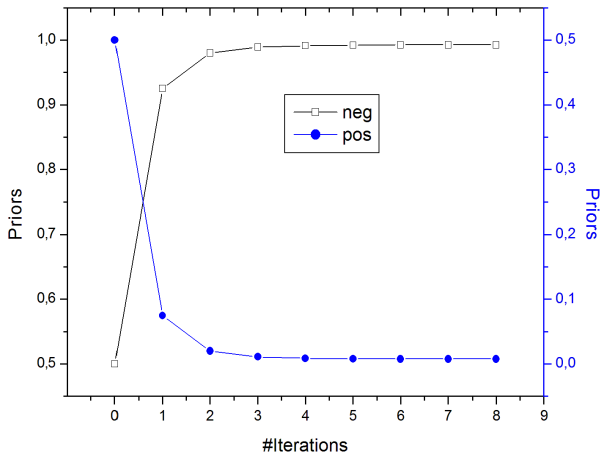

Fig. 5.17: Convergence of Log-EM algorithm

5.7 Conclusion

1. For all the studied $yeast$ datasets, oversampling or the combination of both oversampling and undersampling techniques have revealed the highest accuracy for imbalanced datasets. Undersampling is less competitive when the number of instances is small (as for $yeast3$, $yeast4$, $yeast5$, $yeast6$ datasets) and otherwise for large number of instances (as for 'Credit Card Fraud Detection').

2. Logistic Regression model demonstrates rather poor accuracy compared to other classification models if additional adjustment procedure for priors is not applied.

3. Logistic Regression with additional adjustment of priors is competitive in terms of accuracy with more sophisticated models such as Random Forest. Note that for studied datasets there is only slight difference in performance of Logistic Regression when priors are known and not known.

4. The best performance for imbalanced datasets have been shown for the combination of both undersampling and oversampling methods as a preprocessing technique and Log-EM or Random Forest as a classification model.

5. In terms of CPU time, oversampling revealed the worst result whereas undersampling and SMOTE worked an order of magnitude faster.

6 Conclusion

6.1 Main conclusions

In the master's thesis, imbalanced classification problems have been studied. Different classification models have been evaluated on $yeast$ datasets and real-world dataset 'Credit Card Fraud Detection'. Namely, Random Forest, k-NN, and SVMs-RBF models have been analyzed as well as preprocessing methods such as oversampling, undersampling, SMOTE. In all the cases, the performance of clssification models have been considerably improved after applying preprocessing techniques. More specifically, for all the studied $yeast$ datasets and classification models, oversampling or the combination of both oversampling and undersampling techniques have revealed the highest accuracy. Undersampling seems to be less competitive when the number of instances is small (as for $yeast3$, $yeast4$, $yeast5$, $yeast6$ datasets) and otherwise for large number of instances (as for 'Credit Card Fraud Detection').

A special focus have been given to the investigation of some ways of dealing with imbalanced datasets based on Logistic Regression. Different adjustment procedures for priors depending on whether the information on estimates of priors was available (TP model) or not (Log-EM model) have been studied. One can conclude that Logistic Regression model demonstrated rather poor accuracy compared to other classification models when additional adjustment procedure for priors was not applied. However, with additional adjustment for priors it was competitive with more sophisticated models such as Random Forest.

The best performance for imbalanced datasets have been shown for the combination of both undersampling and oversampling methods as a preprocessing technique and Log-EM or Random Forest as a classification model.

6.2 Future Work

For the future work one may propose the following topics:

1. Study on some extensions of preprocessing methods. For instance, some extensoins of SMOTE technique could be considered: SMOTE+ENN (Batista and Prati, 2004), Bordeline-SMOTE (Border-SMOTE) (H.Han and Wang, 2005), Safe-Level SMOTE (SL-SMOTE) (Bunkhumpornpat and Sinapiromsaran, 2009). In all the cases a level of balance in the training data near to 50:50 distribution should be obtained.

2. Additional study of adjustment procedure for the priors of Logistic Regression. For instance, it is important to clarify a question regarding the impact of number of iterations on the overall performance of Logistic Regression. Furthermore, it is not completely clear the reason behind significant improvement of the performance of Logistic Regression when adjustment procedure is applied. One should study comprehensively the behavior of bias term which might be responsible for that.

3. Study on the cost-sensitive algorithms. It could be beneficial to carry out an analysis regarding cost-sensitive classifiers. Namely, "Weighted-Classifier" (CS-Weighted) (Barandela and Sanchez, 2003; Ting, 2002) and CostSensitive Classifier (CS-Classifier) from the Weka environment (Hall and Frank, 2009). In the first case, the base classifiers are modified usually by wieghting the instances of the dataset to take into account the a priori probabilities, according to the number of samples in each class. In the two latter casses, one can use an input cost-matrix.

4. Algorithmic modification can also be important for imbalanced datasets. There were several possible ways of performance improvement when dealing with Random Forest and SVMs-RBF but new techniques or modifications of well-known old techniques should be studied.

Bibliography

Barandela, R. and J.S. Sanchez (2003). *Strategies for learning in class imbalance problems*. Pattern Recognition 36 (3) 849-851 (cit. on p. 40).

Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press 1st ed. (cit. on p. 12).

Batista, G.E.A.P.A. and R.C. Prati (2004). *A study of the behavior of several methods for balacing machine learning training data*. SIGKDD Explorations 6 (1), 20-29 (cit. on p. 40).

Batuwita, R. and V. Palade (2010). *Efflcient resampling methods for training support vector machines with imbalanced datasets*. Proceedings of the International Joint Conference on Neural Networks, pp. 1–8, IEEE Press (cit. on p. 12).

Bunkhumpornpat, C. and K. Sinapiromsaran (2009). *Safe-level-SMOTE: Safe-levelsynthetic minority over-sampling technique for handling the class imbalanced problem*. Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining PAKDD'09, 2009, pp. 475-482 (cit. on p. 40).

Chawla, Nitesh and Kevin Bowyer (2002). *SMOTE: synthetic minority over-sampling technique*. Journal of Artiflcial Intelligence Research 16 (2002) 321–357 (cit. on pp. 7, 9).

Chen, Chao and Andy Liaw (2004). *Using random forest to learn imbalanced data*. Report ID: 666 (cit. on pp. 2, 13).

Dempster, A. and N. Laird (1977). *The elements of statistical learning: data mining, inference and prediction*. Journal of the Royal Statistical Society, 39, 1-38 (cit. on p. 18).

Egan, J.P. (1975). *Signal Detection Theory and ROC Analysis*. Series in Cognition and Perception, AcademicPress, NewYork (cit. on p. 9).

Fernandez, J. and S. Salcedo-Sanz (2015). *Signficant wave height and energy flux range forecast with machine learning classfiers*. Eng. Appl. Artif. Intell. 2015, 43, 44–53 (cit. on p. 1).

Gao, Wei and Fabrizio Sebastiani (2016). *From classification to quantification in tweet sentiment analysis*. Soc. Netw. Anal. Min (2016)6:19 (cit. on p. 32).

Ha, T.M. and H. Bunke (1997). *Off-line, handwritten numeral recognition by perturbation method*. Pattern analysis and machine intelligence, 19/5, 535–539 (cit. on pp. 5, 7).

Hall, M. and E. Frank (2009). *The WEKA data mining software: an update*. SIGKDD Explorations 11 (1), 10-18 (cit. on p. 40).

Hastie, T. and R. Tibshirani (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer (cit. on pp. 12, 15).

He, Haibo and Yunqian Ma (2013). *Imbalanced Learning: Foundations, Algorithms, and Applications*. IEEE Press Wiley (cit. on p. 12).

H.Han and W.Y. Wang (2005). *Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning*. Lecture Notes in Computer Science, 3644, 878-887 (cit. on p. 40).

James, Gareth and Trevor Hastie (2015). *An Introduction to Statistical Learning, 6th ed.* Springer (cit. on pp. 11, 33).

Johnson, B. and R. Tateishi (2012). *Using geographically weighted variables for image classification*. Remote Sensing Letters, 3(6), 491-499 (cit. on p. 12).

Lopéz, Victoria and Albert Fernández (2013). *An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics*. Information Sciences 250 (2013) 113-141 (cit. on pp. 2, 5, 8).

Lowry, Richard (1998). *Concepts & Applications of Inferential Statistics* (cit. on p. 29).

Panigrahi, B. and P. Dash (2009). *Hybrid signal processing and machine intelligence techniques for detection, quantfication and classfication of power quality disturbances*. Eng. Appl. Artif. Intell. 2009, 22, 442–454 (cit. on p. 2).

Pérez-Ortiz, María and Silvia Jiménez-Fernández (2016). *A review of classiflcation problems and algorithms in renewable energy applications*. Energies 2016, 9, 607 (cit. on pp. 1, 11).

Saerens, Marco (2001). *Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure*. Neural Computstion 14, 21-41 (2001) (cit. on pp. 15–18).

Ting, K.M. (2002). *An instance-weighting method to induce cost-sensitive trees*. IEEE Transactions on Knowledge and Data Engineering 14 (3), 659-665 (cit. on p. 40).

Wang, Fei and Zhao Zhen (2015). *Solar irradiance feature extraction and support vector machines based weather status pattern recognition model for short-term photovoltaic power forecasting*. Energy and Buildings 86 (2015) 427-438 (cit. on p. 1).

Woods, K. and C. Doss (1993). *Comparative evaluation of pattern recognition techniques for detection of microcalciflcations in mammography*. Pattern Recognition and Artiflcial Intelligence,7(6):1417–1436 (cit. on p. 8).

Websites

Analytics Vidhya Content Team (2016). *Practical Guide to deal with Imbalanced Classification Problems in R*. URL: `https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/` (visited on May 2, 2017) (cit. on p. 6).

Benyamin, D (2012). *A gentle introduction to random forests, ensembles, and performance metrics in a commercial system*. URL: `https://datascienceplus.com` (visited on May 2, 2017) (cit. on p. 13).

Kaggle dataset repository (2016). *Credit Card Fraud Detection*. URL: `https://www.kaggle.com/dalpozz/creditcardfraud` (visited on Feb. 2, 2017) (cit. on p. 32).

KEEL dataset repository (2017). *Imbalanced datasets*. URL: http://sci2s.ugr.es/keel/imbalanced.php (visited on Feb. 2, 2017) (cit. on p. 20).

List of Figures

List of Tables

Place des Doyens, 1 bte L2.01.01, 1348 Louvain-la-Neuve, Belgique **www.uclouvain.be/lsm**

UCL
Université
catholique
de Louvain