MATH 191 TOPICS IN DATA SCIENCE
DIMENSIONALITY REDUCTION FOR IMBALANCED CLASSIFICATION

SHARON XU*, AKSHAT MAHAJAN*, RAYMOND OOI*

December 13, 2015

Abstract. We describe several novel approaches towards addressing imbalanced classification. The first class of techniques proposed involves the application of a dimensionality reduction technique, followed by classification in the reduced space. Using principal component analysis as our baseline, we investigate the random walk Laplacian in conjunction with radius nearest neighbours (rNN) and distance classification algorithms. Through cross-validation on both simulated and real-world datasets, we find that the Laplacian eigenmap and rNN perform the best in terms of the F-score metric, and that Laplacian eigenmap-based classification consistently outperforms principal component analysis-based classification. We extend our results with the proposal of a random walk Laplacian-based version of the SMOTE oversampling algorithm.

Key words. diffusion maps, SMOTE, random walk Laplacian, imbalanced classification

1. Introduction. The *marginal class imbalance* problem arises when one class in the dataset is severely underrepresented[1][4]. In recent years there has been increased interest in applying statistical methods to challenging real-world problems, many of which are characterized by imbalanced data. These applications include medical diagnosis, fraud detection, speech recognition, page retrieval, and more. However, most standard learning algorithms are not designed to cope with imbalanced class distributions, resulting in significant performance compromises on such datasets. We refer to the ratio of points in the minority class to the points in the majority class as the *imbalance ratio*.

In this work, we propose to extend the sphere of imbalanced classification techniques with *nonlinear* dimensionality reduction techniques. Specifically, we are interested in the performance of the random walk Laplacian as compared to principal component analysis. We implemented our own simulated dataset, as well as our own naive classification algorithms to work upon the datasets following dimensionality reduction. Performance was compared by F-score; dataset was cross-validated (five-fold for real-world, two-fold for synthetic) to avoid overfitting.

2. Related work. Traditionally, the class imbalance problem has been approached through three general types of methods: *informed sampling*, *cost-sensitive learning*, and *ensemble-based* techniques[1]. Informed sampling techniques aim to establish a more balanced class distribution, either through undersampling of the majority points or oversampling of the minority points. Informed undersampling methods generally remove majority instances based on three heuristics: noisy instances that are too close to the decision boundary (e.g., Tomek links[7]), redundant instances whose information can be explained through other observations (e.g. OSS[8]), instances that suffer from class-label noise (ENN), and instances distant from the decision border, which do not contribute as much to learning (CNN[9]). On the other hand, intelligent oversampling methods like SMOTE[2] aim to create synthetic minority points that generalize the feature space just enough to add predictive power, while minimizing the noise added. Meanwhile, cost-sensitive learning assigns a higher cost to the misclassification of minority points and minimizes the Bayes' conditional risk:

$$R(i|x) = \sum_j P(j|x)C(i,j),$$

where $P(j|x)$ is the the probability of class j for a given example x, and $C(i,j)$ represents the cost of predicting class i when the true class is j. Thus, the minority points will contribute more to the overall error (more commonly used with models such as neural networks and AdaBoost-based implementations). Finally, ensemble methods for imbalanced classification employ a set of classifiers, where the class distribution for each classifier is randomly sampled to be more evenly distributed. One such method is the balanced random forest[11], where each tree is trained on a different bootstrap sample of the original data.

More recent approaches have included dimensionality reduction techniques as well. In the unsupervised realm, Shyu *et al.*[5] implemented an anomaly detection scheme using principal component analysis, with applications by Zheng *et al.*[6] to real-world traffic analysis. The paper computed Mahalanobis distance of a test point to a distribution from the underlying PCA scores.

[1]UCLA, 520 Portola Plaza, Mathematical Sciences Building 6363, Los Angeles, CA 90095-1555

Explorations with a combination of SMOTE (synthetic minority oversampling) and locally linear embedding with LLE have been carried out by Wang *et al.*[4]. For each minority instance x_i, SMOTE randomly generates a synthetic minority point in the space between x_i and a minority class nearest neighbor. However, this method assumes that the space between two minority class neighbors belong solely to the minority class, which may not be the case for less separable data. The authors used LLE to find an embedding where the data was separable, performed SMOTE in that space, and then projected the new points back to the original space. We will outline a parallel methodology using Laplacian eigenmaps.

3. Our work. Here we explore the potential of the random walk Laplacian with regards to achieving good class separation with imbalanced datasets. In our methodology, we first apply dimensionality reduction techniques to the dataset as a whole (training and testing) and then use several classification algorithms to predict the class of out-of-sample test data. We assume that the underlying distribution is such that minority points are close among each other and further away from the majority points, and vice versa.

In this paper, we present our own fast implementation of two simple classification methods: a Euclidean distance-based method, and radius nearest neighbors. Although the diffusion map was the original choice for nonlinear dimensionality reduction, calculating and storing the dense distance matrix on massive datasets was extremely computationally expensive. We found that the R package **diffusionMap**[14] did not have sufficient controls to address this, so we implemented a sparse version of the technique - in other words, the random walk Laplacian. This is essentially a generalization of the diffusion map - it uses only the k-nearest neighbors to compute the distance matrix, whereas the diffusion map sets k to $N-1$, where N is the number of observations in the dataset.

Our implementation of the random walk Laplacian, *rwlaplacian()*, leverages sparse matrices in R for compact storage. It is able to scale much better than *diffuse()* from the **diffusionMap** package. Whereas the latter crashed R with 5000 observations, our random walk Laplacian implementation successfully handled 16,500 rows instances. Ultimately, our methods using the random walk Laplacian were able to outperform some highly popular methods for classification in the original space, including SVM and the nearest neighbors algorithm.

Metrics. In the imbalanced case, classification accuracy is not a useful metric - if the ratio is 1:99, we would obtain 99% accuracy just by classifying all points in the majority class. For this reason, we will take two metrics in evaluating our methods:

$$\text{Precision} = P(X_{\text{actual}} = 1 | X_{\text{pred}} = 1), \tag{3.1}$$

$$\text{Recall} = P(X_{\text{pred}} = 1 | X_{\text{actual}} = 1), \quad \text{where 1 is the minority class.} \tag{3.2}$$

Traditionally, the F-score is used as a compact representation of these two metrics:

$$\text{F-score} \;=\; 2\,\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3.3}$$

All methods were evaluated through cross-validation on several real-world datasets. In the following subsections, we describe the techniques we employed.

3.1. Dimensionality Reduction. Successful dimensionality reduction results in better class separation, reduced noise, and computational gains. In the first step we apply such a technique in order to better separate the two classes.

3.1.1. Baseline: Principal Component Analysis. As a *linear* dimensionality reduction technique, Principal component analysis (PCA) functions as our baseline aginst which we will compare the random walk Laplacian, a self-implemented non-linear dimensionality reduction technique. Given points $x_1, \ldots, x_n$ in $\mathbb{R}^d$, in PCA we derive a projection onto $\mathbb{R}^m$ by a linear transformation of the dataset.

Step 1. Compute the $d \times d$ correlation matrix.

Step 2. Compute the eigenvalues and eigenvectors of the correlation matrix.

Step 3. Select the top m eigenvectors to form our new m-dimensional basis, $m << d$. Given that the correlation matrix is square symmetric, these eigenvectors are guaranteed to be orthogonal.

Step 4. Project dataset onto the new basis by a linear transformation of the form $y = W^T x$.

Here, W is the $d \times m$ matrix formed by taking each eigenvector as a column. The resulting datapoint now has dimensions of $m \times 1$ and captures a large percentage of the variance in the original datapoint.

3.1.2. Reversible Random Walk Laplacian. Given points $x_1, \ldots, x_n$ in $\mathbb{R}^d$, we construct a weighted graph with n nodes, with the edges connecting each of the n points to its k nearest neighbors. The embedding map is then derived from computing the eigenvectors of the random walk Laplacian[12].

Step 1. Construct a symmetric adjacency graph.

k nearest neighbors, $k \in \mathbb{N}$. Connect nodes i and j with an edge if i is among the k nearest neighbors of j, or if j is among the k nearest neighbors of i. This gives us a sparse, scalable representation of the data.

Step 2. Weight the edges.

Heat kernel. Compute the sparse weight matrix W where:

$$W_{ij} = \begin{cases} e^{-\|x_i - x_j\|^2/\epsilon} & x_i \text{ and } x_j \text{ connected, } \epsilon \in \mathbb{R} \\ 0 & \text{otherwise} \end{cases}$$

Step 3. Derive the low-dimensional embedding.

Random walk Laplacian. Construct the random walk Laplacian computed by

$$L = D^{-1}W,$$

where the diagonal weight matrix D is given by $D_{ii} = \sum_j W_{ji}$. We solve the eigenvector problem

$$L\psi = \lambda\psi$$

We disregard the eigenvector ψ_0 corresponding to the trivial eigenvalue of 1, and use the next m eigenvectors as the basis for our embedding in m-dimensional Euclidean space. Interpreting the eigenvectors as functions over our dataset, the Laplacian eigenmap maps points from the original space to the space spanned by the first m eigenvectors: $\mathcal{L} : \mathbb{R}^d \mapsto \mathbb{R}^m$, $m << d$:

$$\mathcal{L}(x_i) = (\lambda_1\psi_1(i),\ \lambda_2\psi_2(i),\ \ldots,\ \lambda_m\psi_m(i))$$

.

3.2. Random Walk Laplacian-Based SMOTE (RWL-SMOTE). We oversample the minority class using the random walk Laplacian and the Synthetic Minority Oversampling Technique to obtain synthetic minority points S_i.

Step 1. Map points to embedding given by the Random Walk Laplacian.

Step 2. Oversample the minority points in the new low-dimensional space to obtain $\mathcal{L}(S_i)$:

SMOTE: Generates synthetic minority instances using the k nearest minority class neighbors of each minority point x_i. The new synthetic point is chosen randomly along the line segment between a neighbor and x_i. The detailed algorithmic procedure is outlined below.

For $i = 1, \ldots N$:

1. Compute the k nearest neighbors of x_i. Randomly select a nearest neighbor.
2. Take the difference between the feature vector x_i and the chosen nearest neighbor.
3. Multiply this difference by a random number between 0 and 1.
4. Add it to x_i to get the synthetic example.

Repeat depending on the amount of oversampling needed. For oversampling at N x 100%, repeat the process N times.

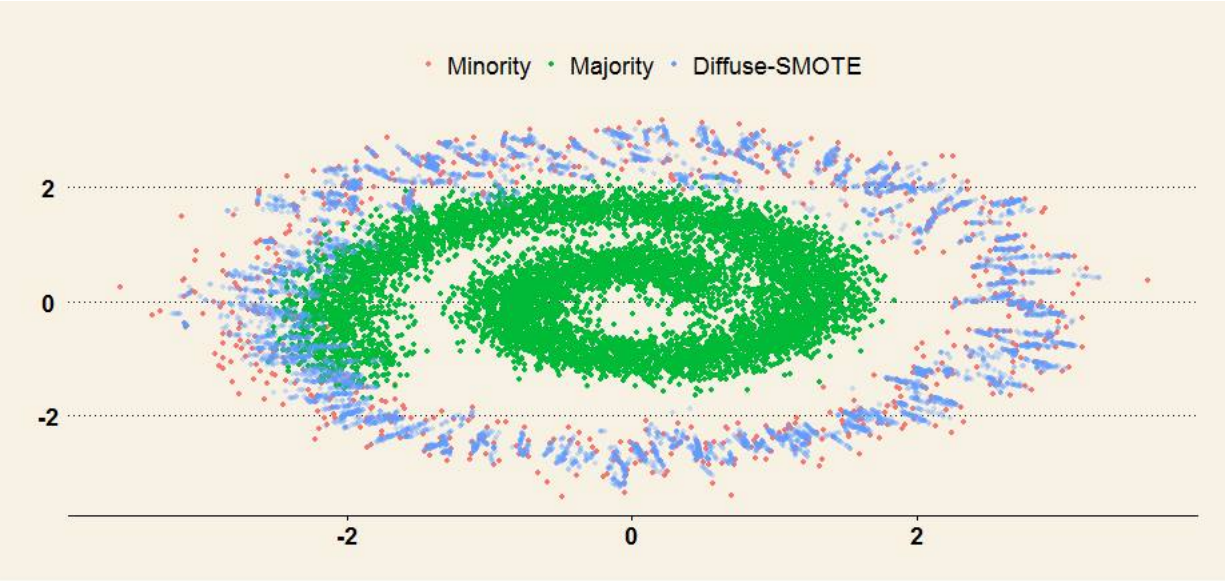


FIG. 3.1. *Result of SMOTE on synthetic dataset*

Step 3. Project $\mathcal{L}(S_i)$ back to the original space. Find the k nearest neighbors of S_i and their distance to S_i in the random walk Laplacian embedding:

$$D_{ij} = ||\mathcal{L}(S_i) - \mathcal{L}(x_j)||_2^2, \quad T_{ii} = \sum_{j=1}^{N} D_{ij}$$

Calculate the row-stochastic weight matrix $W = T^{-1}D$. The synthetic points are simply given by

$$S_i = \sum_{j=1}^{N} W_{ij} x_j$$

3.3. Classification. After obtaining a low-dimensional embedding through principal component analysis or the random walk Laplacian, we apply one of two classification methods:

3.3.1. Distance Classification.

Step 1. Compute the center of masses of the distribution of minority and majority class from the test data.

$$C_{maj} = \frac{1}{n} \sum_{i=0}^{n} x_i^{maj} \ , \ C_{min} = \frac{1}{m} \sum_{i=0}^{m} x_i^{min}$$

Step 2. For each point x_i in the training data, compute the Euclidean distance to these center of masses.

$$D_{ik} = ||x_i - C_{maj}||_2^2 \ , \quad i = 1, \ldots, N, \quad k = 0, 1$$

Step 3. Assign the point to the class for which the Euclidean distance is smaller; i.e., assign x_i to class k such that

$$k = \arg\min_k D_{ik}, \quad k = 0, 1$$

3.3.2. Radius Nearest Neighbors Algorithm.

Step 1. For a given radius r, compute the nearest neighbours of the test data set within r of a training datapoint.

Step 2. Compute the fraction of minority points within these nearest neighbours.

$$\text{minority fraction} = \frac{|\text{minority neighbors}|}{|\text{total neighbors}|}$$

Step 3. If the fraction exceeds a certain threshold, assign the point to the minority class; otherwise, assign it to the majority class.

The specific choice of what radius r to use is obtained through trial and error. For our purposes, we experimented with radius r as to maximize the F-score. This algorithm will sometimes fail to classify certain points due to radius size.

3.4. Support Vector Machine. The support vector machine[10] uses a kernel to map data to a high-dimensional space, where the classes can more readily be linearly separated. A good separation is achieved by the hyperplane that has the largest margin (the distance to the nearest data point of any class). We will use SVM as a classification method to evaluate SMOTE and RWL-SMOTE.

4. Numerical experiments on synthetic data. To create our simulated dataset, the following steps were employed. Majority points were presumed to have a spiral distribution generated by the parametric equations

$$x(t) = 0.5 \exp(0.15t) \cos(t) + \epsilon \tag{4.1}$$
$$y(t) = 0.5 \exp(0.15t) \sin(t) + \epsilon \tag{4.2}$$

where $\epsilon \sim N(0, 0.3)$ and $t \sim 10 \cdot U^{0.7}$, where $U \sim \text{Unif}[0, 1]$. Minority points were generated from a torus of

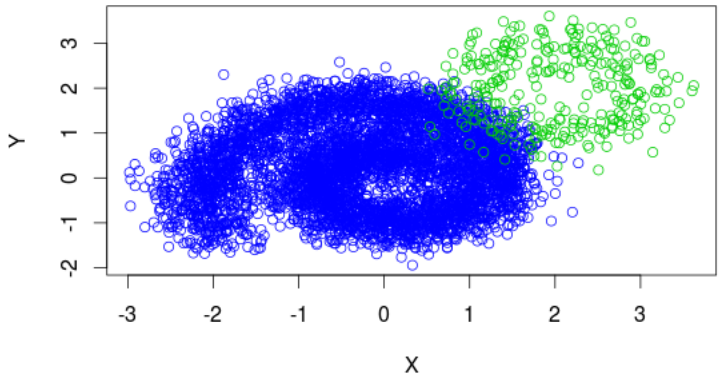
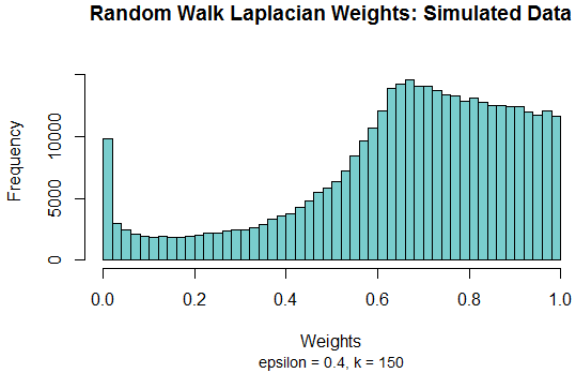
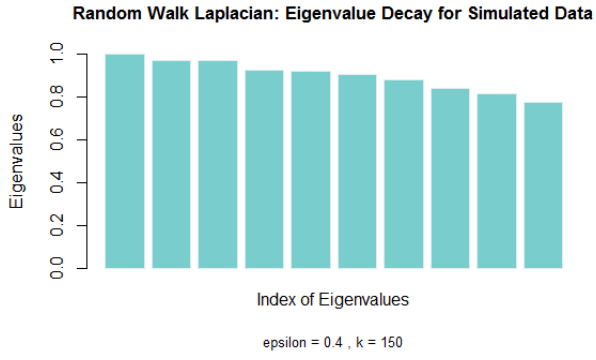


FIG. 4.1. *Our simulated dataset. Majority points are in blue; minority points are in green.*

radius $\rho = 2$, with random offsets $\epsilon \sim N(2, 0.4)$ for each coordinate. Our dataset consisted of 2000 points from the majority class and about 200 points from the minority class, possessing an imbalance ratio of $1 : 10$. Dataset was 2-fold cross-validated.





The following tables represent averaged summary statistics for five confusion matrices. The 2x2 Punnett squares at the top of each table were formed as the sum of the confusion matrices for all five of the confusion matrices

generated for each split; the precision, recall and F-score metrices presented here represent the mean ± standard deviation of the metrics across each individual split.

TABLE 4.1
Principal Component Analysis

(a) Distance Classification

	Actual Majority	Actual Minority
Predicted Majority	186	0
Predicted Minority	14	20
Precision	0.59 ± 0.00	
Recall	1.00 ± 0.00	
F-score	0.73 ± 0.00	

(b) Radius Nearest Neighbors Classification

	Actual Majority	Actual Minority
Predicted Majority	198	3
Predicted Minority	1	16
Precision	0.94 ± 0.00	
Recall	0.84 ± 0.00	
F-score	0.88 ± 0.00	

TABLE 4.2
Random Walk Laplacian

(a) Distance Classification

	Actual Majority	Actual Minority
Predicted Majority	199	4
Predicted Minority	1	16
Precision	0.94±0.00	
Recall	0.80±0.00	
F-score	0.86±0.00	

(b) Radius Nearest Neighbors Classification

	Actual Majority	Actual Minority
Predicted Majority	200	3
Predicted Minority	0	17
Precision	1.00±0.00	
Recall	0.85±0.00	
F-score	0.91±0.00	

TABLE 4.3
Random Walk Laplacian + SMOTE

(a) Support vector machine

	Actual Majority	Actual Minority
Predicted Majority	158	0
Predicted Minority	41	19
Precision	0.32 ± 0.00	
Recall	1.00 ± 0.00	
F-score	0.48 ± 0.00	

(b) Radius Nearest Neighbors Classification

	Actual Majority	Actual Minority
Predicted Majority	177	0
Predicted Minority	23	20
Precision	0.47 ± 0.00	
Recall	1.00 ± 0.00	
F-score	0.64 ± 0.00	

Overall, the random walk Laplacian applied together with the radius nearest neighbors algorithm had the best performance. *Note:* As this is a proof of concept, we did not record the radius r or threshold for which these values were recorded - we chose the best results returned from a range of possible radii. These values are displayed, however, for the real-world datasets below.

5. Numerical experiments on real data. We used 5-fold cross-validation to compare our techniques to several baseline methods. Tests were conducted on the following real-world datasets:

Loans: imbalance ratio of 20:1, which contains credit history for defaulters (the minority class) and non-defaulters (the majority class).

Yeast: An imbalanced version of the Yeast data set, where there are some classes with a small number of examples while other classes have a large number of examples. It consists of 1484 instances and 8 attributes, with an imbalance ratio of 1:23.

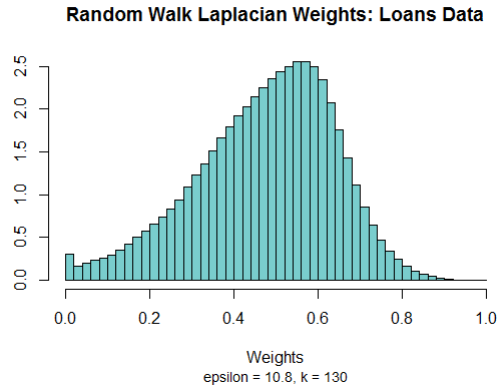
Additional Preprocessing: Although the random walk Laplacian $D^{-1}W$ was originally calculated with the full normalized dataset ($\sim$ 140,000 observations), *eigs()* in **rARPACK**[13] reached the maximum number of iterations before eigenvalue calculations converged (likely due to sparsity issues). Thus, our objective became to use the largest number of observations possible, subject to constraints on computational feasibility.

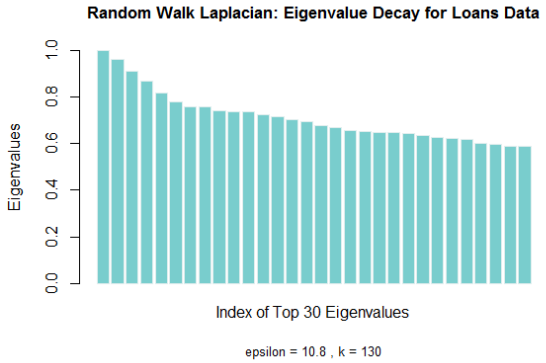
It soon became clear that additional preprocessing was needed in order to obtain the embedding. We experimented with various levels of undersampling and sparsity for the Laplacian. Ultimately, we applied two informed undersampling methods, one-sided selection (OSS), and the neighborhood cleaning rule (NCL). To achieve the

best F-score, we then retained all of the minority instances, and randomly undersampled the majority instances to 10% the original size, obtaining a dataset of about 17,000 rows. We used each instance's 130 nearest neighbors to compute the weight matrix.

The following tables represent averaged summary statistics for five confusion matrices. The 2x2 Punnett squares at the top of each table were formed as the sum of the confusion matrices for all five of the confusion matrices generated for each split; the precision, recall and F-score metrices presented here represent the mean ± standard deviation of the metrics across each individual split.

5.1. Results for Loans. .





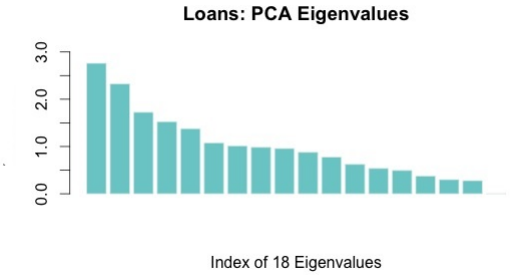


TABLE 5.1
Principal Component Analysis

(a) Distance Classification

	Actual Majority	Actual Minority
Predicted Majority	4947	7062
Predicted Minority	1984	2742
Precision	0.27 ± 0.01	
Recall	0.58 ± 0.01	
F-score	0.37 ± 0.01	

(b) rNN (threshold=0.35, r = 4.207)

	Actual Majority	Actual Minority
Predicted Majority	9735	2319
Predicted Minority	2212	2374
Precision	0.52 ± 0.01	
Recall	0.51 ± 0.02	
F-score	0.51 ± 0.01	

TABLE 5.2
Random Walk Laplacian

(a) Distance Classification

	Actual Majority	Actual Minority
Predicted Majority	6047	2471
Predicted Minority	5962	2255
Precision	0.27 ± 0.01	
Recall	0.47 ± 0.01	
F-score	0.34 ± 0.01	

(b) rNN (threshold=0.35, r=0.025)

	Actual Majority	Actual Minority
Predicted Majority	9565	2470
Predicted Minority	2440	2256
Precision	0.48 ± 0.01	
Recall	0.47 ± 0.00	
F-score	0.47 ± 0.00	

TABLE 5.3
Random Walk Laplacian - SMOTE (100% Oversampling, k = 30)

(a) Support Vector Machine

	Actual Majority	Actual Minority
Predicted Majority	N/A	N/A
Predicted Minority	N/A	N/A
Precision	N/A	
Recall	N/A	
F-score	N/A	

(b) rNN (threshold=0.5, r=30)

	Actual Majority	Actual Minority
Predicted Majority	11755	1487
Predicted Minority	253	3239
Precision	0.92 ± 0.01	
Recall	0.68 ± 0.01	
F-score	0.78 ± 0.01	

TABLE 5.4
SMOTE (100% Oversampling)

(a) Support Vector Machine

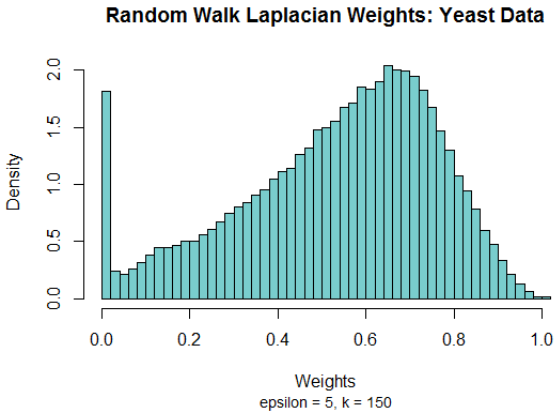
	Actual Majority	Actual Minority
Predicted Majority	9848	1787
Predicted Minority	2161	2939
Precision	0.58± 0.02	
Recall	0.62± 0.01	
F-score	0.60 ± 0.02	

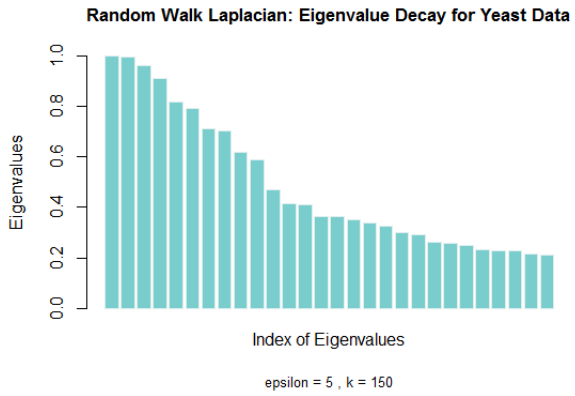
(b) rNN (threshold=0.9, r=30.136)

	Actual Majority	Actual Minority
Predicted Majority	9903	1793
Predicted Minority	2106	2933
Precision	0.582 ± 0.012	
Recall	0.620 ± 0.014	
F-score	0.600 ± 0.013	

Overall, the random walk Laplacian-based SMOTE method, in conjunction with the radius nearest neighbors classification, achieved the best performance.

5.2. Results for Yeast.





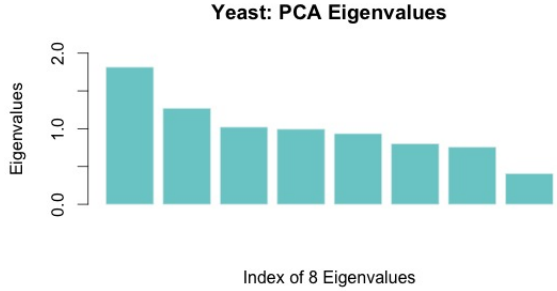


TABLE 5.5
Principal Component Analysis

(a) Distance Classification

	Actual Majority	Actual Minority
Predicted Majority	180	10
Predicted Minority	1268	25
Precision	0.02 ± 0.01	
Recall	0.69 ± 0.27	
F-score	0.04 ± 0.02	

(b) rNN (threshold=0.4, r=1.75)

	Actual Majority	Actual Minority
Predicted Majority	1397	13
Predicted Minority	13	22
Precision	0.66 ± 0.23	
Recall	0.62 ± 0.11	
F-score	0.63 ± 0.16	

TABLE 5.6
Random Walk Laplacian

(a) Distance Classification

	Actual Majority	Actual Minority
Predicted Majority	127	16
Predicted Minority	1321	19
Precision	0.01 ± 0.01	
Recall	0.55 ± 0.18	
F-score	0.03 ± 0.011	

(b) rNN (threshold=0.2, r=0.21)

	Actual Majority	Actual Minority
Predicted Majority	1380	20
Predicted Minority	67	15
Precision	0.20 ± 0.08	
Recall	0.44 ± 0.12	
F-score	0.27 ± 0.09	

TABLE 5.7
Random Walk Laplacian - SMOTE (300% Oversampling, k = 10)

(a) Support Vector Machine

	Actual Majority	Actual Minority
Predicted Majority	1447	27
Predicted Minority	1	8
Precision	0.93 ± 0.12	
Recall	0.22 ± 0.15	
F-score	0.42 ± 0.13	

(b) rNN (threshold = 0.55, r = 2.00)

	Actual Majority	Actual Minority
Predicted Majority	1413	19
Predicted Minority	12	16
Precision	0.61 ± 0.21	
Recall	0.45 ± 0.18	
F-score	0.50 ± 0.18	

TABLE 5.8
SMOTE

(a) Support Vector Machine (100% Oversampling)

	Actual Majority	Actual Minority
Predicted Majority	1342	6
Predicted Minority	106	29
Precision	0.22 ± 0.05	
Recall	0.83 ± 0.12	
F-score	0.35 ± 0.05	

(b) rNN (threshold=0.05, r=2.51, 300% Oversampling)

	Actual Majority	Actual Minority
Predicted Majority	934	3
Predicted Minority	480	32
Precision	0.064 ± 0.011	
Recall	0.922 ± 0.108	
F-score	0.119 ± 0.018	

No overall performance comparison can be made. Random walk Laplacian-based SMOTE performed best for SVM classification methods, but was outperformed by PCA for radius nearest neighbours.

6. Summary and conclusion. We conclude, from our detailed investigation of both synthetic and real-world data, that the random walk Laplacian is a viable dimension reduction technique in the context of imbalanced datasets. In particular, the random walk Lapacian-based SMOTE oversampling algorithm shows promise in generalizing the feature space of the minority class, resulting in higher predictive power.

In terms of classification methods, the distance algorithm was generally unsuccessful. The radius nearest neighbor method, while more flexible than distance classification and SVM, can fail to classify instances if the radius is not large enough. This limits choice of radius, and ultimately may decrease predictive power. k nearest neighbors, though less geometrically intuitive, would overcome this deficiency.

Possible future research could involve incorporating completely new test data, estimating the coordinates in the Laplacian embedding space using a weighted nearest neighbors method similar to the one used in RWL-SMOTE.

REFERENCES

[1] N. Chawla, *Data Mining for Imbalanced Datasets: An Overview*, Ch. 40 of *Data Mining and Knowledge Discovery Handbook, ed.* M. Oded and L. Rokach *(2010).*

[2] N.V. Chawla *et al.* SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research (2002): 321-357.*

[3] M. Belkin, P. Niyogi, Laplacian Eigenmaps for Dimensionality Reduction and Data Representation, *Neural computation 15.6 (2003): 1373-1396.*

[4] J. Wang, J. Zhang,M. Xu, H. Wang, Classification of Imbalanced Data by Using the SMOTE Algorithm and Locally Linear Embedding, *Signal Processing, 2006 8th International Conference on. Vol. 3. IEEE, 2007.*

[5] M.-L. Shyu *et al.* A novel anomaly detection scheme based on principal component classifier. *MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING, 2003.*

[6] C. Zheng *et al.* Using principal component analysis to solve a class imbalance problem in traffic incident detection. *Mathematical Problems in Engineering 2013 (2013).*

[7] G. Batista, A. L.C. Bazzan, M. C. Monard. Balancing Training Data for Automated Annotation of Keywords: a Case Study. *WOB. 2003.*

[8] M. Kubat, S. Matwin. Addressing the curse of imbalanced training sets: one-sided selection. *ICML. Vol. 97. 1997.*

[9] K.C. Gowda, G. Krishna."The condensed nearest neighbor rule using the concept of mutual nearest neighborhood. *IEEE Transactions on Information Theory 25.4 (1979): 488-490.*

[10] CJC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery 2.2 (1998): 121-167.*

[11] C. Chen, A. Liaw, L. Breiman. Using random forest to learn imbalanced data. *University of California, Berkeley (2004).*

[12] M. Cucuringu, *Slides on "Diffusion Map", Math 191 Fall 2015*

[13] Y. Qiu, J. Mei, *rARPACK: R wrapper of ARPACK for large scale eigenvalue/vector problems, on both dense and sparse matrices. = http://CRAN.R-project.org/package=rARPACK, version 0.7-0 (2014).*

[14] J. Richards, *diffusionMap: Diffusion map. = http://CRAN.R-project.org/package=diffusionMap, version 1.1-0 (2014).*