Universität Rostock Traditio et Innovatio

DISSERTATION

zur

zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)

Improved imbalanced classification through convex space learning

Promotionsgebiet Systembiologie und Bioinformatik

Fakultät für Informatik und Elektrotechnik

UNIVERSITÄT ROSTOCK

vorgelegt von

SAPTARSHI BEJ

geboren am 24. Juli, 1991, Bardhaman, West Bengal, Indien

Betreuer: Professor Olaf Wolkenhauer
Universität Rostock

Rostock, August 2021

https://doi.org/10.18453/rosdok_id00003503

Gutachter: Prof. Olaf Wolkenhauer
Lehrstuhl Systembiologie & Bioinformatik,
Universität Rostock
email: olaf.wolkenhauer@uni-rostock.de

Prof. Jan Baumbach
Zentrum für Bioinformatik,
Universität Hamburg
email: jan.baumbach@uni-hamburg.de

Prof. Carsten Ullrich
Steinbeis Hochschule, Berlin,
Direktor Künstliche Intelligenz,
CENTOGENE GmbH
email: carsten.ullrich@centogene.com

Verteidigung: Dezember 16, 2021

“ Stay hungry, Stay foolish ”

– Steve Jobs

This thesis is dedicated to my father Dr. Debkumar Bej,

who has always motivated me to enrich myself with knowledge

AND

to my mother Usha Rani Sen,

who has always cared for my well-being, ahead of everything

Acknowledgements

First, I would like to express my heartfelt gratitude to my supervisor, Prof. Olaf Wolkenhauer. Olaf, thank you for teaching me so much about research starting from the very minute details, investing a lot of your time in lengthy discussions. Starting from kindling my interest in data science, you have guided me through every small step, yet providing me with ample freedom and motivation to find my own path. I hope, I can keep on learning from you for a long time to come to enrich myself as a researcher and more importantly, as a person.

Next, I would like to thank my colleagues Markus Wolfien, Shailendra Gupta, Suchi Smita, Ali-Salehzadeh Yazdi, Krishna Pal Singh, Faiz Muhammad Khan, Martin Schram, Kristian Schulz, Andrea Bagnacani, Mariam Nassar, Peggy Sterling, Tom Gebhardt, Narek Davtyan, Julia Scheel, David Brauer and Matti Hoch from the Department of Systems Biology and Bioinformatics (SBI), for creating a fantastic environment for research, continuously learning through discussions and sharing. Discussions with Markus were especially helpful at several stages of the thesis. Thank you for the tedious proof-readings and valuable suggestions for our articles. Shailendra, Andrea, Suchi, Krishna, Marin, Faiz and Ali have also been kind enough to discuss several enjoyable topics, academic or otherwise, on many occasions. I also had the pleasure of working with David and Narek during their masters thesis, which proved to be good learning opportunities for me. I want to thank Tom for helping me out with several documents and bureaucracy-related issues. Thank you, Kristian, for participating in my research and helping me with coding. Thank you, Peggy for always helping me with all the official queries that I had. Finally, special thanks to all those colleagues of me, who lent me their office keys whenever I forgot mine (must be at least equal to the number of pages in this thesis).

Part of the work was supported by the EU Social Fund (*ESF/14-BM-A55-0024/18, ESF/14-BM-A55-0027/18, ESF/14-BM-A55-0028/18*), the "Deutsche Forschungsgemeinschaft" - DFG (*DA1296/6-1*), the German Heart Foundation *(F/01/12*), the FORUN Program of Rostock Medical University (*889001/889003*), the Josef and Käthe Klinz Foundation (*T319/29737/2017*), the Damp Foundation (2016-11), and the Federal Ministry of Education and Research - BMBF (*VIP+00240, 01ZX1709, BMBF FK 031L0106C*). I acknowledge support by the German Network for Bioinformatics (de.NBI) & de.STAIR.

I would like to thank my collaborators from University of Kiel, Prof. David Ellinghaus, Sören Mucha, Florian Uellendahl-Werth and Eike Matthias Wacker for providing me with the opportunity to learn a little biology. My heartfelt thanks to Prof. Ria Baumgrass and her PhD student Yen Hoang, discussions with whom have a very important role in this thesis. I acknowledge to my old friend Dr. Jit Sarkar and his supervisor Dr. Partha Chakrabarty for building a fruitful and enjoyable collaboration with me. I thank my Masters Thesis guide Prof. Anirban Banerjee and Prof. Satyaki Mazumder from my *alma mater* for encouraging and supporting me during my pre-PhD days.

I would like to thank my friends Vicky S. Ekanthalu, Anju Mohanan for inviting me over many delicious dinners and providing me with company. I wish you a lifetime of happiness together. My friends Igor Pokhotelov, Sankha Subhra Pal, Michael Schubert, Atefeh Habibpourmoghadam, and Sourodeep Biswas, thank you for spending several enjoyable moments with me. Thank you, my old friends Gonnabattula Siddartha and Prashant Srivastava for giving me company with good food, nice discussions, and gaming. Looking back on more than ten years of knowing her, I would like to take this opportunity to express my heartfelt gratitude to Shukla Sarkar, who has always supported me at a very personal level during my most difficult times. I hope you know what a delightful person you are.

Finally, I want to acknowledge my parents, who have always advised me to enrich myself in terms of knowledge. Thank you for always showing me the way, being great teachers, and always prioritising my education and well-being over anything else.

Abstract

Imbalanced datasets are abundant in several real-life classification problems where Machine Learning (ML) finds its application. Such problems are characterised by classes with an unequal distribution of samples over several classes. For imbalanced datasets during the training process, the ML models encounter a larger number of majority class samples and, thus, tends to become biased towards the majority class.

One of the principal disadvantages of the SMOTE algorithm is its tendency to over-generalise the minority class. This causes the associated classifiers to misclassify the majority class data points. Moreover, evidence from recent comparative studies reveals that the performance of the extensions of the SMOTE algorithm vary depending on the classifiers they are implemented with.

In this thesis, these limitations of SMOTE-based oversampling algorithms are addressed through the novel idea of *convex space learning*. In an analytical explanation behind the idea, we show that SMOTE-based oversampling algorithms generate synthetic samples with high variance in a minority class data neighbourhood. I developed the LoRAS algorithm that can model the convex space of the minority class using multiple convex combinations of shadowsamples in a minority class neighbourhood.

To address the issue of classifier dependence of SMOTE-based oversampling algorithms, I proposed the ProWRAS algorithm. By controlling the variance of the synthetic samples, as well as a proximity-weighted clustering system of the minority class data, the ProWRAS algorithm improves the performance, compared to algorithms that generate synthetic samples through modelling high dimensional convex spaces of the minority class. Most importantly, the performance of ProWRAS with proper choice of oversampling schemes, is independent of the classifier used.

The success of the proposed algorithms has been demonstrated using rigorous benchmarking studies for over thirty publicly available datasets. The most challenging datasets are chosen with several well-defined criteria such as high imbalance, high dimensionality, and high absolute imbalance, ensuring impartiality in the benchmarking studies. From the benchmarking studies, where the proposed algorithms are compared with over ten SMOTE-based algorithms including some models with best known performance, for diverse performance measures, it is comprehensively established that the proposed algorithms can out-perform state-of-the-art algorithms.

Zusammenfassung

Unausgewogene Datensätze sind in mehreren realen Klassifizierungsproblemen, in denen Maschinelles Lernen (ML) seine Anwendung findet, häufig anzutreffen. Solche Probleme sind durch Klassen mit einer ungleichen Verteilung der Proben auf mehrere Klassen gekennzeichnet. Bei unausgewogenen Datensätzen treffen die ML-Modelle während des Trainingsprozesses auf eine grösSere Anzahl von Stichproben der Mehrheitsklasse und neigen daher dazu, in Richtung der Mehrheitsklasse verzerrt zu werden.
Einer der Hauptnachteile des SMOTE-Algorithmus ist seine Tendenz zur Übergeneralisierung der Minderheitenklasse. Dies führt dazu, dass die zugehörigen Klassifikatoren Datenpunkte der Mehrheitsklasse falsch klassifizieren. Darüber hinaus zeigen aktuelle Vergleichsstudien, dass die Leistung der Erweiterungen des SMOTE-Algorithmus in Abhängigkeit von den Klassifikatoren, mit denen sie implementiert werden, variiert. In dieser Arbeit werden diese Einschränkungen der SMOTE-basierten Oversampling-Algorithmen durch die neuartige Idee des konvexen Raumlernens angegangen. In einer analytischen Erklärung hinter der Idee zeigen wir, dass SMOTE-basierte Oversampling-Algorithmen synthetische Stichproben mit hoher Varianz in einer Minderheitenklassen-Datenumgebung erzeugen. Ich habe den LoRAS-Algorithmus entwickelt, der den konvexen Raum der Minderheitenklasse mit mehreren konvexen Kombinationen von Schattensamples in einer Minderheitenklassen-Nachbarschaft modellieren kann.
Um das Problem der Klassifikatorabhängigkeit von SMOTE-basierten Oversampling-Algorithmen zu lösen, habe ich den ProWRAS-Algorithmus vorgeschlagen. Durch die Steuerung der Varianz der synthetischen Stichproben sowie eines näherungsgewichteten Clustersystems der Daten der Minderheitenklasse verbessert der ProWRAS-Algorithmus die Leistung im Vergleich zu Algorithmen, die synthetische Stichproben durch Modellierung hochdimensionaler konvexer Räume der Minderheitenklasse erzeugen. Am wichtigsten ist jedoch, dass die Leistung von ProWRAS bei richtiger Wahl der Oversampling-Schemata unabhängig vom verwendeten Klassifikator ist. Der Erfolg der vorgeschlagenen Algorithmen wurde durch strenge Benchmarking-Studien für über dreiSSig öffentlich verfügbare Datensätze nachgewiesen. Die anspruchsvollsten Datensätze werden nach mehreren wohldefinierten Kriterien ausgewählt, wie z. B. hohe Ungleichheit, hohe Dimensionalität und hohe absolute Ungleichheit, um die Unparteilichkeit in den Benchmarking-Studien zu gewährleisten. Aus den Benchmarking-Studien, in denen die vorgeschlagenen Algorithmen mit mehr als zehn SMOTE-basierten Algorithmen, einschlieSSlich einiger Modelle mit der besten bekannten Leistung, für verschiedene LeistungsmaSSe verglichen werden, wird umfassend festgestellt, dass die vorgeschlagenen Algorithmen die State-of-the-Art-Algorithmen übertreffen können.

Theses

- The reason behind over-generalization of the synthetic minority class by SMOTE is the high variance in the SMOTE generated synthetic samples.
- Controlling the variance while generating synthetic samples can improve classification. To establish a mathematical rigour, I provided a theoretical proof for the same.
- Using UMAP as a dimension reduction step to assign minority neighbourhoods improves classifier performances for high dimensional datasets.
- LoRAS improves the minority classification, compromising less on the majority classification.
- LoRAS can be used to automatize supervised annotation of rare-cell types from single-cell data. The more overlap between the clusters, the more profitable it is, to use synthetic oversampling.
- ProWRAS algorithm to customize synthetic sample generation based not only on the input data but also, on the classifier of choice, using convenient combinations of parameter choices. This leads to data and classifier specific sample generation and thereby reduces benchmarking efforts.
- An empirical measure of classifier independence is defined subjective to a benchmarking study, can measure the relative superiority of an oversampling algorithm considering its overall performance over all classifiers and datasets.
- Rigorous modelling of the minority class to construct synthetic samples with appropriate variance depending on input data and classifier of choice; termed as convex space learning can improve imbalance classification.

Contents

Acknowledgements v

Abstract vii

Zusammenfassung ix

Theses xi

List of Figures xx

List of Tables xxiii

List of Abbreviations xxv

Summary and outline of the thesis 1

1 Classification problems with imbalanced datasets 5

1.1 Challenges of imbalanced classification in real-life scenarios 5

1.1.1 Imbalanced datasets in bio-medicine 6

1.1.2 Imbalanced datasets in security and business management 7

1.1.3 Imbalanced datasets in engineering and technology 8

1.2 Common machine learning approaches for imbalanced classification problems 8

1.2.1 Data level approaches 9

1.2.2 Algorithm level approaches 12

2 Performance measures for imbalanced datasets 15

2.1 Preliminary performance measures 15

2.2 Common performance measures for imbalanced datasets 17

2.2.1 Precision, Recall and F1-Score 17

2.2.2 Other popular performance measures 20

2.3 Receiver operating characteristic curve and precision recall curve 22

2.4 Wilcoxon's signed rank test for model comparison 25

3 Oversampling Techniques for convex space modelling **29**

3.1 SMOTE algorithm and its limitations . . . 29

3.1.1 Criticisms of the SMOTE algorithm . . . 30

3.2 Some early extensions of the SMOTE algorithm . . . 31

3.2.1 Borderline based oversampling . . . 31

3.2.2 Weighting minority class samples . . . 33

3.3 Integration of SMOTE with unsupervised learning . . . 35

3.3.1 SMOTE with clustering algorithms . . . 35

3.3.2 SMOTE with dimension reduction algorithms . . . 37

3.3.3 Oversampling technique integrating multiple approaches . . . 39

3.4 Comparative studies between oversampling algorithms . . . 40

3.4.1 Some state-of-the-art extensions of SMOTE . . . 41

4 Improving convex space modelling with the LoRAS algorithm **43**

4.1 Modelling the convex space of minority class . . . 43

4.1.1 Geometric interpretation of convex space modelling . . . 43

4.1.2 Analytical explanation for convex space modelling . . . 44

4.2 LoRAS algorithm . . . 46

4.3 Benchmarking studies for LoRAS algorithm . . . 47

4.3.1 Datasets used . . . 47

4.3.2 Study protocols . . . 49

4.4 Improving classifier performance using LoRAS . . . 52

4.5 Significance of the LoRAS algorithm . . . 55

4.6 Integrating LoRAS with the UMAP algorithm . . . 57

4.7 Benchmarking studies for LoRAS UMAP algorithm . . . 57

4.8 Improved performance of LoRAS UMAP algorithm . . . 59

5 Automated annotation of rare cell populations **63**

5.1 Applying LoRAS in a biological context . . . 63

5.1.1 Using single-cell technology for the identification of rare cells . . . 63

5.1.2 Using machine learning algorithms to generate cell types *in silico* . . . 64

5.2 Datasets and methodologies . . . 65

5.2.1 Use case preparation . . . 65

5.2.2 sc-SynO: Transferring the LoRAS algorithm to single-cell data . . . 67

5.3 sc-SynO can detect rare cell types . . . 68
5.3.1 sc-SynO can detect extremely rare glial cells . . . 70
5.3.2 Sc-SynO achieves a low FN rate for the identification of proliferative cardiomyocytes . . . 71
5.3.3 Sc-SynO can detect rare-cell populations from large-scale datasets . . . 73
5.4 Importance and applicability of sc-SynO . . . 74

6 Classifier-independent oversampling using the ProWRAS algorithm 77
6.1 Classifier dependence of oversampling models . . . 77
6.1.1 Study protocols . . . 77
6.1.2 Pilot study confirming classifier dependence of oversampling . . . 81
6.2 ProWRAS algorithm . . . 84
6.3 Classifier independent performance of ProWRAS . . . 89
6.4 Interpretations and applicability of the ProWRAS oversamping approach . . . 91

Concluding remarks 97

Bibliography 99

A Implementation for the integrated LoRAS algorithms 109

B Implementation for the ProWRAS algorithm 113

C Complexity of the ProWRAS algorithm 121
C.1 Assumptions . . . 121
C.2 Estimations . . . 122

D Curriculum Vitae 125

E Publications 129
E.1 Articles published/accepted in peer-reviewed journals relevant to this thesis 129
E.2 Other published/accepted articles in peer reviewed journals . . . 130
E.3 Published/Accepted articles in conference proceedings . . . 131
E.4 Articles in peer-review and other independent articles . . . 131

Declaration of Authorship 137

List of Figures

1.1 Figure showing general working principle of a machine learning algorithm . 6

1.2 Figure showing the difference between two data level approaches: Undersampling and Oversampling 9

2.1 Figure showing several combinations of precision and recall to calculate A: Harmonic mean and B: Arithmetic mean. 19

2.2 Figure showing several combinations of precision and recall to calculate F_β-Score, for A: $\beta = 5$ and B: $\beta = 1/5$. 20

2.3 Figure showing A: Receiver Operating Characteristic (ROC) curve and B: Precision Recall (PR) curve for an imbalanced test dataset for which a hypothetical classifier performance described in Table 2.1. The region shaded in blue in each diagram represents the Area Under the Curve (AUC). The arrows indicate the direction in which the AUC (region shaded in blue) would expand in case of a better classifier. 24

2.4 Figure showing A: Receiver Operating Characteristic (ROC) curve and B: Precision Recall (PR) curve for a balanced subset of an imbalanced test dataset for which a hypothetical classifier performance described in Table 2.1. Note that the ROC-AUC in the balanced case is significantly lower than the imbalanced case shown in Figure 2.3. The PR-AUC however changes very little compared to the imbalanced case and is thus a more robust performance measure on imbalanced datasets. 26

3.1 Figure demonstrating minority class over generalisation by the SMOTE algorithm, which is considered to be one of the key limitations of SMOTE . . 30

4.1 Figure showing the idea of LoRAS to control the variance of the synthetic samples generated from the minority class. Compared to SMOTE, LoRAS can generate low-variance synthetic samples which can be intuitive interpreted as synthetic samples generated closer to the average point or centroid in a minority class data neighbourhood. LoRAS thus prevent classifiers from confusing them to majority class samples. 46

4.2 Figure showing the philosophies of SMOTE and LoRAS oversampling algorithms. While smote generates synthetic samples from convex combination of two close minority class samples, LoRAS generates synthetic samples from random convex combination of multiple shadowsamples in a minority class data neighbourhood. 48

4.3 Figure showing for Principal Component Analysis plot of ozone dataset for baseline data and oversampled data with several oversampling strategies for the ozone_level dataset. The boxed region in each subplot shows a neighbourhood of outliers and how each oversampling strategy generates synthetic samples in that neighbourhood. 56

5.1 Visualization of the workflow, demonstrating a step-by-step explanation for a sc-SynO analysis. **a)** Several or one snRNA-Seq or scRNA-Seq fastq datasets can be used as an input. Here, cell population of interest are identified and provide raw or normalized read counts of this specific population. This can be done with any single-cell analysis workflow, e.g., Seurat. **b)** Further information are extracted for cluster annotation that serve as improved input for the subsequent training with sc-SynO. **c)** Based on the data input, the underlying LoRAS [Bej 2021] synthetic oversampling algorithm of sc-SynO is utilized to generate new cells around the former origin of cells to increase the size of the minority sample. **d)** The trained Machine Learning classifier is validated on the trained, pre-annotated dataset to evaluate the performance metrics of the actual model. The sc-SynO model is now ready to identify the learned rare-cell type in novel data. This figure was solely created by the authors. 64

5.2 Comparison of the Allen Brain Atlas mice data of the whole dataset from (`https://celltypes.brain-map.org/`) and the reanalysis. The 119_Pvalb Vipr2 cluster, consisting in total of 1,720 cells, was chosen as a rare-cell type of interest. The sc-SynO input was 624 cells of this cell type obtained from the first 300,000 cells in the data. 66

5.3 Figure showing a comparison between the distribution of synthetic cells (purple) generated by sc-SynO and original input cells (blue) for model training. 70

5.4 Comparison of the baseline classification and sc-SynO visualized as mean outcome for the used quality parameter: F1-Score (grey), Precision (blue), and Recall (purple). Detailed results can be obtained in Table 5.3. I observe that in every case, sc-SynO improves the recall compared to the Baseline model (see dotted boxes in the figure). This ensures that oversampling with sc-SynO improves the detection rate of rare-cell types. 72

5.5 Validation of the sc-SynO model for the first use case of cardiac glial cell annotation. **a)** UMAP representation of the manually clustered Bl6 dataset of Wolfien *et al.* [Wolfien *Cardiovascular Research* 2020]. Predicted cells of sc-SynO are highlighted in blue, cells not chosen are grey. **b)** UMAP representation of the manually clustered dataset of Vidal *et al.* [Vidal 2019]. Predicted cells of sc-SynO are highlighted in blue, cells not chosen are grey. **c)** Average expression of the respective top five cardiac glial cell marker genes for both validation sets, including the predicted clusters and those in proximity. 72

5.6 Validation of the sc-SynO model for the second use case of proliferative cardiomyocyte annotation. **a)** UMAP representation of the manually clustered single-nuclei dataset of Linscheid *et al.* [Linscheid 2019]. Predicted cells of sc-SynO are highlighted in blue (based on top 20 selected features in the training model), red (based on top 100 selected features in the training model) cells not chosen are grey. **b)** UMAP representation of the manually clustered dataset of Vidal *et al.* [Vidal 2019]. Predicted cells of sc-SynO are highlighted in blue (based on top 20 selected features in the training model), red (based on top 100 selected features in the training model) cells not chosen are grey. **c)** Average expression of the respective top five proliferative cardiomyocyte marker genes for both validation sets, including the predicted clusters and those in proximity. 73

6.1 Illustration of the working principle of the ProWRAS algorithm. ProWRAS used a proximity based partitioning system to find clusters in the minority class. For each cluster, it then uses one of four oversampling schemes shown in the figure. 78

6.2 Figure showing results for the pilot study. Every heatmap for a respective classifier shows the number of datasets for which the oversampling model in the *i*-th row performs equally or better (by F1-Score) than the model in the *j*-th column. For example, for the gradient boosting classifier LoRAS performs equally or better than SMOTE for 10 out of 14 datasets. Note that, none of the oversampling models perform consistently well for all the classifiers. 82

6.3 Illustration of the working principle of the ProWRAS algorithm. ProWRAS used a proximity based partitioning system to find clusters in the minority class. For each cluster, it then uses one of four oversampling schemes shown in the figure. The key to success of the ProWRAS algorithm is its ability to rigorously model the convex space through controlling the variance of the synthetic samples. 87

6.4 Figure showing results for the final study. Every heatmap for the respective classifier shows the number of datasets for which the oversampling model in the *i*-th row performs equally or better (by F1-Score) than the model in the *j*-th column. Note that, ProWRAS performs consistently well for all the classifiers. 89

6.5 Summarising oversampling schemes/strategies used by the investigated oversampling models and their respective influence on the classifier performance. For example, SMOTE generates samples with a "High local variance" scheme and works well for Gradient Boosting and Random Forest. Since the ProWRAS algorithm has access to all four oversampling schemes, its performance can be made independent of the chosen classifier. 93

List of Tables

2.1 Table showing an example of predictions of a hypothetical classifier on 10 data points and thereby calculating TPR, FPR, precision and recall from the same. The column Predicted Prob shows the predicted probabilities of belonging to a positive class for 10 data points by some hypothetical classifier. 23

4.1 Table showing some statistics for the datasets studied in for the benchmarking of LoRAS. For each dataset, the feature of the dataset that led us to its choice for this study is marked in bold. 50

4.2 This table shows the details of the parameter settings for the oversampling algorithms used by us for the experiment. The second column is the size of the oversampling neighbourhood, and the same size is chosen for all the oversampling models for each dataset in the analysis. The last three columns are specific to LoRAS parameters. 51

4.3 Table showing balanced accuracy/F1-Score for several oversampling strategies (Baseline, SMOTE, SVM-SMOTE, Borderline1 SMOTE, Borderline2 SMOTE, ADASYN and LoRAS column-wise respectively) for all 14 datasets of interest for ML learning models producing the best average F1 score over all oversampling strategies and baseline training for respective datasets. . . . 53

4.4 Table showing the average balanced accuracy/F1-Score of the selected models for datasets with the highest imbalance ratios and high-dimensional datasets separately 53

4.5 Table showing p-values for comparison of LoRAS against the other oversampling algorithms, in terms of both the performance measures used: F1-Score and balanced accuracy. 54

4.6 Table showing $W_+/W_-/R$ for comparison of LoRAS against the other oversampling algorithms, in terms of both the performance measures used: F1-Score and balanced accuracy. 55

4.7 Table showing balanced accuracy/F1-Score for several oversampling strategies (Baseline, SMOTE, SVM-SMOTE, MOT2LD, DBSMOTE, CURE SMOTE, SOMO, LoRAS t-SNE, and LoRAS-UMAP) for all 14 benchmarking datasets 60

4.8 Table showing the results of Wilcoxon's signed rank test for comparison of LoRAS-UMAP with other oversampling algorithms. 61

5.1 Key statistics of the datasets used during this study. The column 'Oversampling nbd' shows the number of nearest neighbours considered for each minority class data points to generate synthetic samples 67

5.2 Table showing comparisons among several feature preselection scenarios in terms of run time and efficiency in detection of glial cells for two different validation datasets (VD 1 & 2) 69

5.3 Table showing F1-Scores/Precision/Recall for sc-SynO against baseline classification for the two ML classifiers (LR and kNN) and for several numbers of pre-selected features (Marker genes). 119_Pvalb represents a small subpopulation of the Allen Brain atlas. 71

6.1 Table showing the ProWRAS oversampling scheme used for every dataset and for every classifier. HGV: High global variance, LGV: Low global variance, HLV: High local variance, LLV: Low local variance. Column 2-5 show the oversampling scheme for which ProWRAS works best for respective datasets and classifiers. Furthermore, the table shows some statistics for the datasets. The last six datasets form Set II. 81

6.2 Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, SOMO, LoRAS) for 14 Set-I benchmarking datasets for GB classifier. 83

6.3 Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, SOMO, LoRAS) for 14 Set-I benchmarking datasets for RF classifier. 83

6.4 Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, SOMO, LoRAS) for 14 Set-I benchmarking datasets for kNN classifier. 83

6.5 Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, SOMO, LoRAS) for 14 Set-I benchmarking datasets for LR classifier. 83

6.6 Table showing the $\mathscr{I}$-scores for different oversampling algorithms for the pilot study. 84

6.7 Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, LoRAS, ProWRAS) for all 20 benchmarking datasets for Gradient Boosting classifier. The column on the right shows the performance of the ProWRAS algorithm over all datasets. Observe in the last row that the average performance of ProWRAS is superior to all other oversampling algorithms. 90

6.8 Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, LoRAS, and ProWRAS) for all 20 benchmarking datasets for Random Forest classifier. The column on the right shows the performance of the ProWRAS algorithm over all datasets. Observe that, in the last row that the average performance of ProWRAS is superior to all other oversampling algorithms. 90

6.9 Table showing F1-score/κ- score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE-SMOTE, LoRAS, and ProWRAS) for all 20 benchmarking datasets for k-Nearest neighbours classifier. The column on the right shows the performance of the ProWRAS algorithm over all datasets. Observe that, in the last row that the average performance of ProWRAS is superior to all the other oversampling algorithms. 91

6.10 Table showing F1-score/κ- score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE-SMOTE, LoRAS, and ProWRAS) for all 20 benchmarking datasets for Logistic Regression classifier. The column on the right shows the performance of the ProWRAS algorithm over all datasets. Observe that, in the last row that the average performance of ProWRAS is superior to all the other oversampling algorithms. 92

6.11 Table showing the results of Wilcoxon's signed rank test for comparison of ProWRAS with other oversampling algorithms. The p-values in the table quantify the statistical significance of the improvement achieved by ProWRAS over each oversampling model. Higher value of W_+, shows that how superior performance of ProWRAS for higher number of datasets. Higher value of R shows better reliability of the results. 92

6.12 Table showing the $\mathscr{I}$-scores for different oversampling algorithms for the final benchmarking experiments. 92

List of Abbreviations

TP	**T**rue **P**ositive
TN	**T**rue **N**egative
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
TPR	**T**rue **P**ositive **R**ate
FPT	**F**alse **P**ositive **R**ate
MCC	**M**atthews **C**orrelation **C**oefficient
ROC	**R**eceiver **O**perating **C**urve
AUC	**A**rea **U**nderthe **C**urve
PR	**P**recision **R**ecall
SMOTE	**S**ynthetic **M**inority **O**versampling **TE**chnique
ADASYN	**Ada**ptive **Syn**thetic Oversampling
ProWSYN	**Pro**ximity **W**eighted **S**ynthetic Oversampling
SOMO	**S**elf **O**rganising **M**ap **O**versampling
MOT2LD	**M**inority **O**versampling **T**echniquebasedon **L**ocal **D**ensities in Low-Dimensional Space
MWMOTE	**M**ajority **W**eighted **M**inority **O**versampling **T**echnique
DBSMOTE	**D**ensity **B**ased **SMOTE**
LoRAS	**Lo**calized **R**andom **A**ffine **S**hadowsampling
ProWRAS	**Pro**ximity **W**eighted **R**andom **A**ffine **S**hadowsampling
sc-SynO	**s**ingle **c**ell- **Syn**thetic **O**versampling
ML	**M**achine **L**earning
GB	**G**radient **B**oosting
LR	**L**ogistic **R**egression
kNN	**k**-**N**earest **N**eighbours
RF	**R**andom **F**orest
AB	**A**da**B**oost
NB	**N**aïve **B**ayes
SVM	**S**upport **V**ector **M**achine
UMAP	**U**niform **M**anifold **AP**proximation
t-SNE	**t-S**tochastic **N**eighbourhood **E**mbedding
SOM	**S**elf **O**rganizing **M**ap
CURE	**C**lustering **U**sing **RE**presentatives
DBSCAN	**D**ensity **B**ased **S**patial **C**lustering of **A**pplications with **N**oise
HGV	**H**igh **G**lobal **V**ariance
HLV	**H**igh **L**ocal **V**ariance
LGV	**L**ow **G**lobal **V**ariance
LLV	**L**ow **L**ocal **V**ariance

Summary and outline of the thesis

Summary

Imbalanced datasets are abundant in several real-life classification problems where Machine Learning (ML) finds its application. Such problems are characterised by classes with an unequal distribution of samples over several classes. For imbalanced datasets during the training process, the ML models encounter a larger number of majority class samples and, thus, tends to become biased towards the majority class. Starting from the vastly popular SMOTE algorithm, rigorous research over the past two decades has resulted in a family of oversampling algorithms that rely on modelling the convex space of the minority class, generate synthetic samples from the modelled convex space and improve classifier performances on imbalanced datasets.

One of the principal disadvantages of the SMOTE algorithm is its tendency to over-generalise the minority class. This causes the associated classifiers to misclassify the majority class data points. Moreover, evidence from recent comparative studies reveals that the performance of the extensions of the SMOTE algorithm vary depending on the classifiers they are implemented with.

In this thesis, these limitations of SMOTE-based oversampling algorithms are addressed through the novel idea of *convex space learning*. In an analytical explanation behind the idea, we show that SMOTE-based oversampling algorithms generate synthetic samples with high variance in a minority class data neighbourhood. I developed the LoRAS algorithm that can model the convex space of the minority class using multiple convex combinations of shadowsamples in a minority class neighbourhood. The LoRAS algorithm also learns the latent manifold structure of the minority class data manifold, to identify precise minority class data neighbourhoods, using several state-of-the-art manifold learning techniques.

To address the issue of classifier dependence of SMOTE-based oversampling algorithms, I proposed the ProWRAS algorithm. By controlling the variance of the synthetic samples, as well as a proximity-weighted clustering system of the minority class data, the ProWRAS algorithm improves the performance, compared to algorithms that generate synthetic samples through modelling high dimensional convex spaces of the minority class. ProWRAS is multi-schematic, employing four oversampling schemes, each of which has its unique way to model the variance of the generated data. The proximity weighted clustering approach of ProWRAS allows one to generate low variance synthetic samples only in borderline clusters to avoid overlap with the majority class. Most importantly, the performance of ProWRAS with proper choice of oversampling schemes, is independent of the classifier used.

The success of the proposed algorithms has been demonstrated using rigorous benchmarking studies for over thirty publicly available datasets. The most challenging datasets are chosen with several well-defined criteria such as high imbalance, high dimensionality, and high absolute imbalance, ensuring impartiality in the benchmarking studies. From the benchmarking studies, where the proposed algorithms are compared with over ten SMOTE-based algorithms including some models with best known performance, for diverse performance measures, it is comprehensively established that the proposed algorithms can out-perform state-of-the-art algorithms.

In addition to benchmark datasets, I applied the algorithms to experimental biological data, for which I have collaborated with research groups in the life sciences. In summary, the convex space modelling based algorithms, LoRAS and ProWRAS, can be highly effective tools for handling imbalanced classification problems.

Outline of the thesis

Chapter 1 addresses several challenges related to imbalanced data classification. The chapter first provides instances of real-life scenarios from different fields, where imbalanced classification problems are relevant. With discussions over several popular approaches that are used to solve the challenges relevant to imbalanced classification, such as data level and algorithm level approaches, the chapter is concluded. The next chapter is therefore dedicated to understand and justify proper choice of performance measures for the studies presented in this thesis.

Chapter 2 explains, why performance measures used in regular classification problems, are often irrelevant for imbalanced classification problems and then proceeds with the discussion of popular choices of metrics for imbalanced classification problems and describes several performance measures apt for imbalanced datasets, along with analytical explanations for the applicability of these measures. The chapter closes with a discussion on the relevance of the Wilcoxon's signed rank test. The next chapter presents developments in the field of synthetic oversampling through convex space modelling in the context of imbalanced classification problems.

Numerous SMOTE-based algorithms over the past two decades has resulted in a family of oversampling algorithms that rely on modelling the convex space of the minority class, generate synthetic samples from the modelled convex space and improve classifier performances on imbalanced datasets. **Chapter 3** presents relevant algorithms from this family of oversampling algorithms that have been instrumental in this research, starting from the SMOTE algorithm. Describing early extensions of the SMOTE algorithm in the beginning, the chapter covers extensions of SMOTE which integrate the algorithm with other unsupervised ML algorithms. Finally, the chapter is concluded with recent comparative studies on SMOTE extensions. The next chapter discusses my research work on improving synthetic sample generation through convex space modelling by introducing the LoRAS algorithm.

Chapter 4 introduces the LoRAS algorithm. The algorithm relies on a more precise modelling of the convex space minority class data compared to its predecessors. Firstly

provides a geometric interpretation and an analytical explanation behind the rationale of the approach adopted by LoRAS for convex space modelling, followed by presenting a detailed description and pseudocode of the algorithm. Next, I discuss the integration of the LoRAS algorithm with the state-of-the art dimension reduction algorithm UMAP. This leads to the LoRAS UMAP algorithm. The protocols and datasets used for the benchmarking studies for LoRAS and LoRAS UMAP algorithms are then described. The next chapter is focused on a bioinformatics application of the LoRAS algorithm.

Chapter 5 presents the LoRAS-based, sc-SynO tool designed for automated annotation of rare cell populations from single-cell data. The chapter first discusses briefly the single-cell technology. Next, the chapter describes the preprocessing techniques for preparation of the data for the study, followed by detailed protocol of the study. Results of data-based experiments in the effectiveness of the tool in detecting and thereby annotating two rare cell populations follow. Finally, the chapter is concluded through discussions on the applicability and importance of sc-SynO. This problem of classifier dependence of oversampling algorithms is addressed in the next chapter, which introduces the ProWRAS oversampling algorithm.

Chapter 6 presents the ProWRAS algorithm, a multi-schematic and classifier-independent extension of the LoRAS algorithm. Firstly, the chapter discusses an independent benchmarking study showing that performance of oversampling algorithms are indeed classifier dependent. Given numerous oversampling algorithms and classifiers to choose from, a proper choice of an oversampling algorithm and classifier for an imbalanced dataset is challenging. Analysing the philosophies of several oversampling algorithms, it is possible to identify generic oversampling schemes followed by common oversampling algorithms. The ProWRAS algorithm is designed to integrate such oversampling schemes under a single umbrella. I demonstrate through rigorous benchmarking studies that the ProWRAS algorithm, with proper choice of parameters, can adapt to classifier specific oversampling schemes and thereby perform in a classifier-independent way. After this chapter, the concluding remarks follow.

Chapter 1

Classification problems with imbalanced datasets

This introductory chapter addresses several aspects and challenges related to imbalanced data classification. Firstly, I provide instances of real-life scenarios from different fields, where imbalanced classification problems are relevant. Thereby, the chapter establishes the motivation behind the research presented in this thesis. Finally, with some discussions over several popular approaches that are used to solve the challenges relevant to imbalanced classification such as data level and algorithm level approaches, the chapter is concluded.

1.1 Challenges of imbalanced classification in real-life scenarios

Imbalanced datasets are abundant in real-life classification problems, where Machine Learning (ML) finds its application. Such problems are characterised by classification problem-based datasets with an unequal distribution of samples over several classes. The class(es) with a higher number of samples are called *majority class(es)* and the class(es) with a lower number of samples are called *minority class(es)*. The ratio between the number of instances in majority and the minority class is known as the *imbalance ratio*.

Imbalanced datasets are known to adversely affect ML based classifiers [Chawla 2002]. These classifiers are designed to learn patterns from data. When trained with sufficiently large volume of data, such classifiers, based on the choice of parameters, construct hypothetical inter-class decision boundaries that help them distinguish between the classes present in the data. Every classifier has a different strategy of constructing the decision boundaries. Moreover, the decision boundary depends on the chosen values of parameters for the classifier. The decision boundary is usually constructed near the *borderline* of the classes. The borderline is the hypothetical region in the data space which is shared by samples from two classes. Intuitively, the more well-defined the borderline region is, in terms of separation among classes, the easier it is for a classifier to construct a decision boundary. After a classification model is trained, it is tested on unseen data. The model decides on the classes of test data, depending on which decision region (based on its training) the test data points find themselves in. Thus, one may conclude that, the better the training experience of the model, the better is its ability to make decisions.

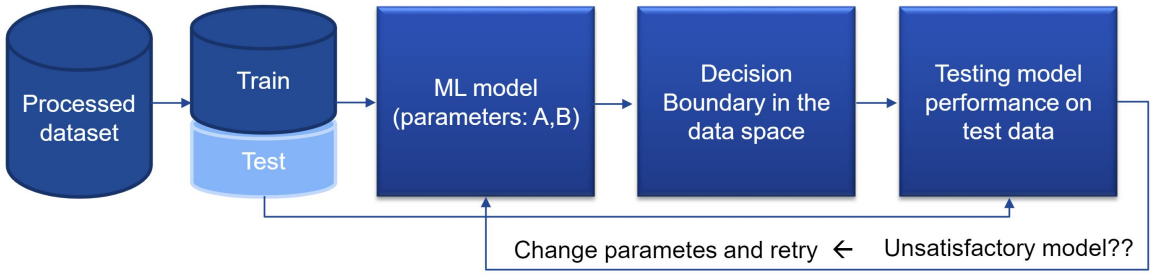


FIGURE 1.1: Figure showing general working principle of a machine learning algorithm

The training experience, however, largely depends on the volume of data used during training. Naturally, the more data used for training, the more patterns are detected by the classifiers and the better is its decision-making ability. For imbalanced datasets during the training process, the ML models encounter a larger number of majority class samples and, thus, tends to become biased towards the majority class. This limits the classification models to form a proper decision boundary, resulting to misclassify minority class samples as the majority class for unseen test data. Every real-life data is likely to have at least a little imbalance in it. However, the bigger the imbalance ratio, the more challenging it becomes for the classifiers to learn patterns from that data. To quote Krawczyk: *"Despite intense works on imbalanced learning over the last two decades there are still many shortcomings in existing methods and problems yet to be properly addressed"* [Krawczyk 2016]. Here, we provide several instances of imbalanced classification problems arising in several real-life research domains that have been discussed in [Fernández 2018].

1.1.1 Imbalanced datasets in bio-medicine

Imbalanced datasets are abundant in several subdomains of biomedical research, such as diagnosis, prognosis, and monitoring, and thereby, digital health in general [Fernández 2018].

Among several instances of imbalanced datasets used in diagnostic research, is a research work on lung nodule detection from computer tomography images. Pulmonary nodules, an important indicator for early-stage lung cancer diagnosis, are present in only a few images, thus creating an imbalanced classification problem [Cao 2014]. Another research classifies malign and benign thyroid nodules instead of lung nodules. Only a few patients have malign nodules, making the classification problem an imbalanced one [Acharya 2016]. One more such instance is based on breast cancer detection from thermography images [Krawczyk 2015]. Several more of such examples can be found in Fernandez *et al.* on diverse diagnostic studies based on polyp detection from colonoscopic images, breast cancer detection from MRI images, detection of diabetes and chronic kidney diseases [Fernández 2018].

As an example of imbalanced dataset in context of prognosis, one might consider a research work dedicated to screening of patents who may be suffering from osteoporosis [Bach 2017]. Another example is a research work to build a donor-receptor allocation system for possible liver transplantation. The model predicts graft survival after transplantation

is conducted, thus providing a decision support on whether the transplantation can be successful [Pérez-Ortiz 2017].

In the front of general digital health, for example, a research work predicts postoperative life expectancy of lung cancer patients. This kind of research can act as a decision support system for clinicians to decide on whether it is safe to consider a patient for a surgery [Zięba 2014]. Another work developed a system to predict the duration of admission of a patient to an emergency department of a hospital. Patients staying for a longer duration consume more resources. Such a decision support system can improve resource management of the hospital [Azari 2015].

In the field of bioinformatics, a research work by Yang *et al.* investigates the applicability of their algorithm on several bioinformatic problems such as: miRNA identification, protein localization prediction, promoter identification from DNA sequences and kinase substrate prediction from protein phosphorylation profiling [Yang 2014]. Another instance in this research domain deals with contact map prediction, a part of protein structure prediction where there are very few positive class instances [Triguero 2015]. Protein classification problem was addressed by Dai *et al.* , where a multi-class classification problem was reduced to multiple binary class problems to obtain multiple imbalanced datasets for classification [Dai 2015]. One more example from bioinformatic research where imbalance classification problem arises is based on cell-recognition problem, where mitotic cells are detected in HEp-2 images obtained after an indirect immunofluorescent assay. Most cells are in the interphase state, making mitosis detection an imbalanced classification problem [Iannello 2014].

1.1.2 Imbalanced datasets in security and business management

Imbalanced datasets are also abundant in several security-related and business management contexts. The has been a lot of research in the field of fraud detection. Lucas *et al.* states: "credit card transactions data suffer from a strong imbalance regarding the class labels, which needs to be considered either from the classifier perspective or from the data perspective (less than 1% of the transactions are fraudulent transactions). There is a significant difference in fraud ratio between e-commerce and face-to-face transactions. We observe in the Belgian credit card transactions dataset that there are 4 fraudulent transactions for 10.000 face-to-face transactions and 3 fraudulent transactions for 1000 e-commerce transactions. The class imbalance is 17 times stronger for the face-to-face transactions than for the e-commerce transactions. E-commerce and face-to-face transactions are very different paradigms and therefore are studied separately in this work, moreover classifier efficiencies are measured on the two types of transactions [Lucas 2019]."

Another research work analyses a fraud detection problem in a car insurance company, also based on an imbalanced dataset. The authors worked with a car company to acquire a dataset for their study. They describe the problem as: "One of the factors affecting the price of car insurance policies is the large amount of fraudulent reports that a company is not able to detect is. The company has to assume all the increase in costs produced by this fraud, and, as a consequence, the insurance policies become more expensive. The

experts in the companies think that at least 10% or 15% of the produced reports are fraudulent, and, however, about 5% of them is detected. So the databases in insurance companies have the following characteristics: the examples labelled as fraudulent belong to the minority class (class imbalance) and, on the other hand, they are the only 100% reliable data, because among the examples labelled as not fraudulent there are some fraudulent examples that the company has not been able to detect. Therefore the information provided to the algorithm is not correct which makes the machine learning problem difficult to solve [Pérez 2005]." Besides the security reason, imbalanced datasets are relevant to the field of business management in other contexts, such as customer churn prediction. Effendy *etal.* states "Customer churn is a major problem that is found in the telecommunications industry because it affects the company's revenue. At the time of the customer churn is taking place, the percentage of data that describes the customer churn is usually low. Unfortunately, the churn data is the data which have to be predicted earlier. The lack of data on customer churn led to the problem of imbalanced data. The imbalanced data caused difficulties in developing a good prediction model."

1.1.3 Imbalanced datasets in engineering and technology

In the field of engineering, imbalanced datasets are abundant in fault detection. There are usually fewer instances of faulty goods and in several contexts detecting the faulty products are important. For example, there are several research articles on the detection of defects in semiconductors from image datasets or consideration of power system short-term voltage stability assessment [Fernández 2018].

Another such example is wind turbine failure prediction. The importance of the task lies in the fact that most of the operational costs of wind farms are due to their maintenance and the distance between farms and industrial areas. Automatically monitoring, diagnosing and predicting the state of wind turbines effectively reduce maintenance costs [Fernández 2018].

Moreover, imbalanced data prevails in the research domain of software fault detection and network data analysis. For network data analysis, the problem of differentiating P2P botnet network traffic from normal traffic. The problem was imbalance because there were more normal traffic examples than abnormal (botnet) traffic ones [Fernández 2018].

Imbalanced classification problems are also common in wireless sensor networks. Patel *et al.* confirms that "Very few works have concentrated on handling of imbalanced data in WSNs. When the data are generated by wide range of sensors, continuously, there is every chance that the data generated from some of these sensors may be discrete; thereby, generated data from those sensors can be sparse. This makes the dataset generated from these sensors imbalanced [Patel 2020]."

1.2 Common machine learning approaches for imbalanced classification problems

Broadly, machine learning approaches that are used as remedies to imbalanced classification problems can be categorised into two types: data level approaches and

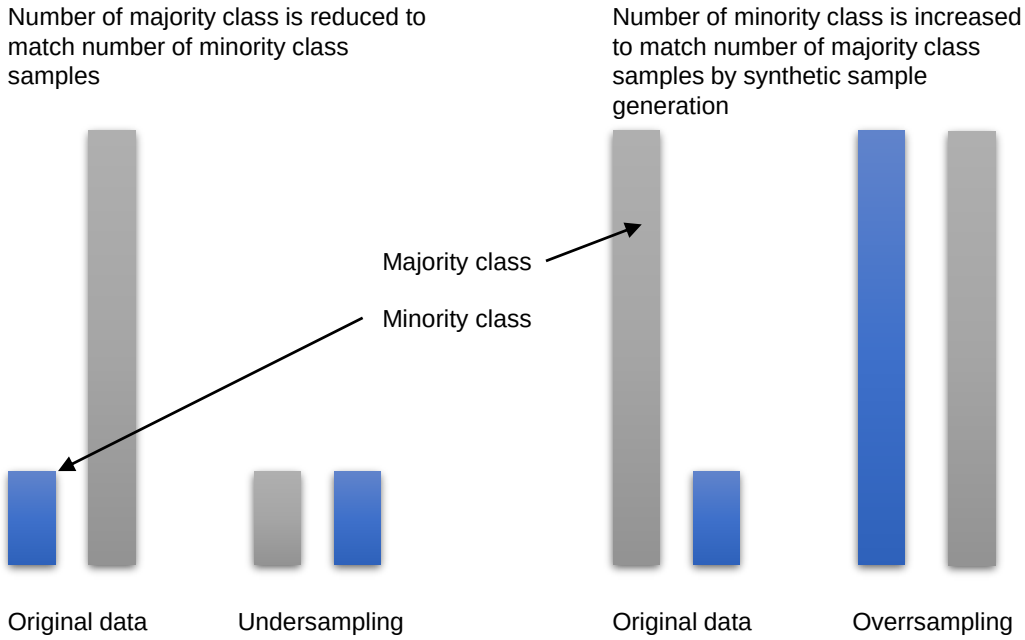


FIGURE 1.2: Figure showing the difference between two data level approaches: Undersampling and Oversampling

algorithm level approaches [Fernández 2018]. This section provides a brief overview and some examples for both types of approaches.

1.2.1 Data level approaches

The strategy of data level approach is to cleverly manipulate the data to learn from imbalanced datasets. It can again be categorised into two broad categories: oversampling and undersampling.

Oversampling approaches are characterised by synthetic minority class sample generation to improve the learning experience of the classifiers. One of the first oversampling approaches used to add Gaussian noise to minority class samples to generate synthetic samples [Lee 2000]. More and more complex approaches gradually came into being. One prominent direction of synthetic oversampling that is most relevant to this thesis is the convex space modelling of the minority class. This direction of research was pioneered by the SMOTE algorithm [Chawla 2002], which is still probably the most popular algorithm for synthetic sample generation. Since this thesis is closely related to these categories of algorithms, they are discussed in detail in Chapter 3.

A more recent trend in the research on imbalanced datasets is to generate synthetic samples, aiming to approximate the latent data manifold of the minority class data space. In [Bellinger 2018], a general framework for manifold-based oversampling, especially for high-dimensional datasets, is proposed for synthetic oversampling. The method has been successfully applied in [Bellinger 2016] to deal with gamma-ray spectra classification. It produces a synthetic set S of n instances in the manifold-space by randomly sampling n instances from the PCA-transformed reduced data space. In order to produce unique

samples on the manifold, they apply i.i.d. additive Gaussian noise to each sampled instance prior to adding it to the synthetic set S, controlling the distribution of the noise through the Gaussian distribution parameters. The synthetic Gaussian instances are then mapped back to the feature space to produce the final synthetic samples [Bellinger 2018]. Another scheme, using auto-encoders to oversample from an approximated manifold, has also been discussed in [Bellinger 2018]. This approach selects random minority class samples by adding Gaussian noise to them, and using the auto-encoder framework first maps them non-orthogonally off the manifold and then maps them back orthogonally on the manifold [Bellinger 2018].

Generative Adversarial Networks (GANs) can be considered as a breakthrough in the field of synthetic sample generation. Introduced in 2014, this generative network can be viewed as a competition among a generator NN and a discriminator NN [Goodfellow 2014]. The generator network creates synthetic samples from random noise, whereas the discriminator network tries to detect whether a sample is synthetic. During training steps, feedback from the discriminator network is fed into the generator network, helping it to produce more convincing synthetic samples. Instances of combining *Variational AutoEncoder* (VAE) and GANs can be found [Blum 2018]. State-of-the-art models applying GANs to oversample for imbalanced datasets are Generative Adversarial Minority Oversampling [Mullick 2019] and Conditional GAN (CGAN) [Douglas 2018].

GANs, on the other hand, also come with some limitations [Li 2018]. In addition to being a complicated model, GANs are infamous for being difficult to train and sensitive to small changes in hyperparameters [Mescheder 2018]. A typical source of instability is the discriminator rapidly overpowering the generator (overpowering effect) which leads to problems such as vanishing gradients or mode collapse [Sajjadi 2018]. The problem of vanishing gradient corresponds to the inability to model to effectively tune parameters by gradient descent while training. Complex models suffer from this problem, where in the initial hidden layers, the weights and biases of the Neural Network (NN) stop getting optimised during the training process. The mode collapse problem is the tendency of GANs not to generate synthetic data uniformly from the learnt latent space. Because of these problems, the use of GANs is still limited to mostly image-based datasets in imbalanced classification. For feature-based tabular datasets, mostly researchers rely on conventional oversampling methods, because of the simplicity of the approach and its effectiveness in training predictive models even with comparatively less amount of data.

Undersampling is a very popular data level approach of dealing with imbalanced datasets. An imbalanced dataset has excess majority class instances, which makes it difficult for classifiers to learn patterns from the data. The strategy of undersampling approach is to reduce the imbalance by removing excess data points from the majority class. This approach may be suitable when a sufficient number of minority class data points are already available. The most obvious way of doing this is, of course, randomly deleting majority class instances to rebalance the data. This technique is called Random Under Sampling (RUS). However, the problem with this is, there is always a risk of deleting data points that could be useful or important for the classification. Thus, it might be valuable to determine which data points to retain and which to remove during an undersampling process. Several algorithms and techniques have been developed using the approach of undersampling. Some of them

decide which data points to retain after undersampling, while some of them determine which data points to delete.

The Nearest Neighbour rule (NN rule) is to assign a label to an unlabelled data based on the label of its nearest neighbour. However, to decide on the label of an unlabelled data point, one must search the whole training set, which makes it computationally expensive. This motivates the Condensed Nearest Neighbour rule (CNN rule). The CNN rule retains the core philosophy of the NN rule, but searches for the nearest neighbour of an unlabelled data point in a subset of the whole training set. This subset is chosen such that it can still assign a correct label for all unlabelled data points. Several approaches have been explored to solve the problem of finding such a subset.

In the context of undersampling, the goal is to find such a subset S' of the actual training set S, so that the overall performance of a classifier improves. This is precisely the rationale behind the US-CNN algorithm. In this algorithm, firstly, all minority class data points and a single majority class data point are moved to the set S'. Now it considers the remaining data points in S as unlabelled data points and the data points in S' are training data points. The algorithm then uses the NN rule to label the data points in S. Since the labels of the all the members of S are actually known, one can verify whether they are correctly labelled after this process. The misclassified data points from S are then moved to S'. The rationale behind this step is that, for the correctly classified data points in S (which were all majority class data points), the S' used for training was enough. All correctly classified data point in S, thus already had a majority class data point as its nearest neighbour and moving some more majority class data points from S to S' will still keep a majority class data point as their nearest neighbour. Ultimately, the selected data points in S' are enough to label all members of S correctly. So this undersampling approach decides which data points to retain from the majority class. Now we look at an approach that decides, which majority class data points to delete.

Imbalanced classification problems often encounter datasets, such that there is a large overlap between the majority and minority class. The more the overlap between the majority and minority class, the more difficult it gets for a classifier to construct a decision boundary. This leads to several misclassifications on application of the trained classifier to the test data. The Tomek-links are data points that are typically lie in these overlapping regions between the minority and the majority class [So 1976].

Definition 1. *A Tomek-link (in the context of a binary classification problem) is defined as, a pair of data points x and y, such that x and y are not both from C_{maj} or C_{min} and there is no other data point z for which $d(x,z) < d(x,y)$ or $d(y,z) < d(x,y)$.*

Thus, a Tomek-link typically represents data points that are either in the borderline region of minority or majority classes or data points that are surrounded by more instances of the opposite class. Removing such instances are known to clean up the boundary region of the two classes, helping a classifier to create a cleaner decision boundary [So 1976].

Based on the concept of Tomek-Links, researchers have also constructed more complex data preprocessing schemes. For example, one such study focuses on not only on removing the samples in the borderline but also the outliers and redundant data points [Devi 2017].

The outliers in this context are the majority class instances that are close to many minority class instances. Redundant data points, on the other hand, are the majority class instances that are not outliers and that are too similar to each other and thus may contain redundant information. To determine redundancy, pairwise similarity among data points is calculated using three different similarity measures. For a redundant pair of data points, the one with the lesser contribution factor is eliminated. The contribution factor is calculated from likelihood of a data point to belong to its class [Devi 2017].

To further improve the model in a follow-up research, the concept of boosting has also been integrated to Tomek-link-based undersampling [Kumar 2019]. Boosting is a common concept in ML, used frequently in the context of imbalanced datasets, to improve classifier performances by using an ensemble of weak learners. The weak learners to learn in steps, trying to correct its mistakes made in a past step. For this model, the boosting algorithm AdaBoost is used. Note that the concept of Boosting is also used along with the RUS method (RUS-Boost) [Seiffert 2008].

1.2.2 Algorithm level approaches

Algorithm level approaches are characterised by clever manipulation of the classifiers themselves to improve classifier performance for imbalanced datasets. One of the prominent directions of research in this domain is **cost sensitive learning**. Such algorithms are sensitive to different costs associated to several entities of the classification problem. For example, given a dataset, costs can be associated with features or classes of the relevant dataset. A cost imposed on features could ensure that classifiers are able to utilise the best possible set of features for learning, while the cost imposed on classes could ensure that the classifier will focus more on avoiding misclassification of samples belonging to classes with higher weights [Fernández 2018].

In the context of imbalanced classification, cost sensitive algorithms impose costs on classes. Higher costs are imposed on minority classes to ensure that classifiers avoid misclassification of minority class data points. Intuitively, for regular classification problems, loss functions assign a value of zero to correctly classified instances and a value of one to incorrectly classified instances while training. The goal of the training procedure is to minimise the overall loss. However, this loss function assigns the same cost for the wrong classification of instances from every class. When the data is highly imbalanced and the classifier has a lot of training examples from the majority class, this loss function gets minimised easily, hence biased towards the majority class instances and unable to classify the minority class instances [Fernández 2018].

Cost sensitive learning employs a modified loss function with different costs associated to each class. Minority classes are assigned with higher penalisation cost compared to the majority classes. Then the loss cannot be minimised easily without a better learning of the minority class by the classifier. The costs assigned to the classes can be decided by domain experts. However, it might be difficult to use cost-sensitive algorithms in the case of multi-label problems, regression, or time series analyses. There have been instances where cost sensitive learning is adopted for several classifiers such as Support Vector Machines (SVM), Artificial neural networks, etc. [Fernández 2018]

Another prominent algorithm level approach is **ensemble learning**. The main philosophy of these kinds of approaches is to improve classification by constructing several classifiers or weak learners, whose combinations help in the final decision-making. The two most prominent algorithms of this type are *bagging (bootstrap aggregating)* and *boosting* algorithms. Bagging algorithms consist of several classifiers with different bootstrapped replicas of the original training dataset. To train each classifier in an ensemble of classifiers, a new dataset is created by randomly drawing instances from the dataset with replacement. Each classifier in the ensemble, thus, gets trained on a different set of data so that together, they can capture the diversity, present in the dataset. When an instance unseen by the model is classified, all these classifiers vote on which class it should belong, and the class with the majority of votes is chosen to be the class of the unseen instance.

The principal idea behind boosting is to train a series of weak classifiers sequentially, such that a certain weak classifier tries to rectify the misclassifications done by the previous weak classifier during training. A commonly used boosting algorithm is AdaBoost. The weak learners in the AdaBoost classifier are formed by decision trees. The first decision tree is trained with the training data of samples all carrying equal weight. It first trains on a pool of data. Then it tests itself on the same training data, trying to determine how much of the training data it can classify correctly already from its experience. Then, the misclassified data points are identified and weights are assigned to these data points. After this, the next decision tree is trained on this weighted data, to ensure that the weak classifier gets more efficient in classifying the weighted data. This process is repeated iteratively to enhance the efficiency of the weak classifier. AdaBoost picks its weak learners t in such a fashion that each newly added weak learner is able to infer something new about the data. To implement this idea, AdaBoost maintains a weight distribution D among all data points. Each data point x_i is assigned a weight $D(x_i)$ indicting its importance. When measuring a weak learners performance, AdaBoost takes into consideration each data points weight. A misclassified high-weight point will contribute more to the overall weighted error than a misclassified low-weight data point. To get a low weighted error, a weak learner must focus on high-weight data points and try to predict them correctly. The final classification decision of AdaBoost algorithm is achieved by assigning to each weak learner a confidence value β_t, which depends on the weak learners performance on its assigned task and then consider a weighted vote of the weak classifiers. Sometimes, for handling imbalanced classification problems, combinations of data level and algorithm level approaches are also used [Chawla 2003].

Besides the inefficiency of classifiers to learn from imbalanced data as discussed in this chapter, deciding upon a proper performance measure to interpret classifier performances is also a challenge in this research domain. The next chapter is therefore dedicated to understand and justify proper choice of performance measures for the studies presented in this thesis.

Chapter 2

Performance measures for imbalanced datasets

Deciding on proper performance metrics is one of the cornerstones of building reliable imbalanced classification models. This chapter explains, why performance measures used in regular classification problems, are often irrelevant for imbalanced classification problems and then proceeds with the discussion of popular choices of metrics for imbalanced classification problems. In addition to describing the metrics, I present my interpretation and analysis on the pros and cons of these metrics in this chapter. Finally, the chapter closes with a discussion on the relevance of the Wilcoxon's signed rank test, a statistical test used in the research presented in this thesis, for comparing performance of multiple models.

2.1 Preliminary performance measures

A ML model gains experience from training data by "learning" underlying patterns present in the training data. After the training, the model is tested on unseen data (test/validation data), to see how well it performs based on the experience gathered during the training step. But how can one quantify the performance of ML models? Note that, for supervised learning models, the labels or ground truths are always known to us. Therefore, for such problems, it is possible to compare the model predictions with the known labels to get an idea about how efficient the model is. For classification problems based on imbalanced datasets, especially, these performance measures have context-specific significance, which I will explore in this chapter.

ML model predictions can come in several forms. For example, in the case of k-Nearest Neighbour, the outputs are directly in the class of the nearest neighbour. For Naïve Bayes (NB) model or the Logistic regression (LR) model obtained for each test data point, the probabilities of their association to each class. Some models such as linear regression might associate a numerical score with the test data point instead of associating it directly with a class. However, ultimately, these predictions can all be transferred to predicting classes. For example, consider applying a NB model to a binary class problem (with classes A and B). While testing the model, for some test data points, its probability of belonging to class A and class B can be obtained. The sum of these probabilities will always be one. One can, thus, always choose the class A or B, such that the test data point has a higher probability of

belonging to that class. If the probability of belonging to both classes is equal, then one can choose a class randomly for the test data point. For a classification problem, a very useful way to visualise the test results is in the form of a Confusion Matrix [Olson 2008].

Definition 2. *For a classifier dealing with a classification problem with n classes, one can write the results of the classification task on the test data in form of a $n \times n$ matrix C such that:*

1. *The diagonal elements of the matrix contain the number of correct prediction for classes $1, \ldots, n$, respectively*
2. *An arbitrary non-diagonal element C_{ij} of C, is the number of test data points whose label(ground truth) is class i, but predicted wrongly by the classifier as class j.*

For a binary class problem, the confusion matrix is 2×2. Usually, in a binary classification problem, one class is labelled as a positive class and the other one a negative class. In such a case, the confusion matrix is viewed as:

$$C = \begin{pmatrix} \text{True negative } (TN) & \text{False positive } (FP) \\ \text{False negative } (FN) & \text{True positive } (TP) \end{pmatrix} \tag{2.1}$$

Note that, the terms False Negative and False Positive can be confusing. To be clear, False Negatives are the test data points who actually belong to the positive class but are falsely predicted as the negative class. In other words, the negative predictions of the classifier which are actually false. The False positives are just the opposite. From now on, the abbreviations mentioned in Matrix 2.1 will be used for all these terms. Note that, in statistical hypothesis testing, FP and FN are associated with Type I and Type II errors, respectively. Depending on the confusion matrix C, several quantities can be defined that can measure the model performances in several contexts. The most obvious and widely used of these quantities is the accuracy measure.

Definition 3. *Given a confusion matrix C, by applying a certain classifier on a test dataset relevant to a classification problem, the accuracy of the classifier in general is defined as:* $\frac{\sum_i C_{ii}}{\sum_{i,j} C_{ij}}$ *[Olson 2008].*

In other words, the accuracy measure is just the ratio of correct predictions made by the classifier. The accuracy measure is often written as a percentage. While in general, this measure is widely used to measure the efficiency of a classifier, it is usually unsuitable for imbalanced datasets.

To understand why an accuracy measure is unsuitable for an imbalanced dataset, let us consider an example. Consider an imbalanced dataset with the ratio of negative class instances to positive class instance is $9:1$. Since the test dataset is a random subset sampled from these datasets, let us assume ideally that the ratios of the positive and negative classes are maintained in the test dataset as well. Let us assume that the test dataset has 100 data points such that, the positive class has 10 data points while the negative class has 90 data points. Now a classifier f can learn the patterns in the highly abundant negative class much better from the training data. Usually, in such a case, it is observed that the classifier f can predict all 90 negative class data points from the test data precisely, but hardly predicts the 10

positive class data points from the test data correctly. However, predicting the 90 negative class data points correctly produces an accuracy score for the classifier to be at least 90%. Thus, even though the classification for the positive class is an extremely poor the classifier f, it can achieve a misleadingly high accuracy. For this reason, several other performance measures become relevant in case of imbalanced datasets. The most rudimentary remedy for this problem, would be a measure, popularly known as balanced accuracy [García 2009].

Definition 4. *Given a confusion matrix C, by applying a certain classifier on a test dataset relevant to a classification problem, the balanced accuracy of the classifier is defined as:* $\frac{1}{n}\Sigma_{i=1}^{n}\frac{C_{ii}}{\Sigma_j C_j}$ *, where n is the number of classes in the dataset.*

In other words, the balanced accuracy is the average of the ratios of correct predictions for every individual class. Just like the accuracy measure, the balanced accuracy is also often written as a percentage. For the example considered before, if it is assumed that all 90 of the majority class examples are correctly predicted by the classifier and none of the 10 minority class data points are correctly classified, then the balanced accuracy is 50%. Note that this is because for the majority class the proportion of correct predictions is 1 and for the minority class the proportion of correct predictions is 0, which makes the average proportion 0.5. However, balanced accuracy is still not among the most popular performance measures for imbalanced datasets.

2.2 Common performance measures for imbalanced datasets

Recall the confusion matrix described in Equation 2.1. For imbalanced datasets, several other performance measures can be described using the confusion matrix. I will describe these performance measures and their significance in this section. For the sake of better understanding, I will calculate each performance measure based on a simple example confusion matrix.

$$C' = \begin{pmatrix} 75 & 15 \\ 5 & 5 \end{pmatrix}$$

From the confusion matrix, it is evident that $TN = 75$, $TP = 5$, $FP = 15$ and $FN = 5$ from Equation 2.1. Just to recapitulate, the Accuracy of a classification model producing a confusion matrix C' is 80% and the balanced accuracy is 66.66%.

2.2.1 Precision, Recall and F1-Score

Definition 5. *Given a confusion matrix C, by applying a certain classifier on a test dataset relevant to a classification problem, the True Positive Rate or sensitivity or recall of the classifier is defined as:* $\frac{TP}{TP+FN}$ *[Olson 2008].*

In simpler words, sensitivity or recall or True Positive Rate (TPR) is just the ratio of correct predictions in positive class to the total positive class instances. A high recall score for a certain classifier means that fewer data points from the class c are wrongly classified. However, the recall score says nothing about how many data points from other classes were

incorrectly classified as belonging to class c. For the proposed example confusion matrix C', the recall would be $5/(5+5) = 0.5$.

Definition 6. *Given a confusion matrix C, by applying a certain classifier on a test dataset relevant to a classification problem, the precision of the classifier is defined as:* $\frac{TP}{TP+FP}$ *[Olson 2008].*

In simpler words, precision is the ratio of correct predictions in positive class to the false positive instances [Olson 2008]. Note that, the false positive instances are the negative instances which are erroneously classified as positive instances by the particular classifier. A high precision score for a class c in a classification model indicates that, most data points that are classified as class c were originally labelled as class c by a certain classifier. However, it says nothing about the number of items from the class c that were not classified correctly. For the proposed example confusion matrix C', the precision would be $5/(5+15) = 0.25$.

It is clear from the discussion so far, that neither precision nor recall alone is enough to quantify the performance of a classifier in a wholesome way. They however complement each other and are intricately connected. For a class c, precision and recall together can quantify not only how well the classification of the data points originally labelled as c are, but also can quantify how well the classifier performs in avoiding classification of a data point with label anything but c, as c. For this reason, a particular measure called F-measure was designed. This is alternatively known as F-Score and F1-Score. The term F1-Score for discussions henceforth.

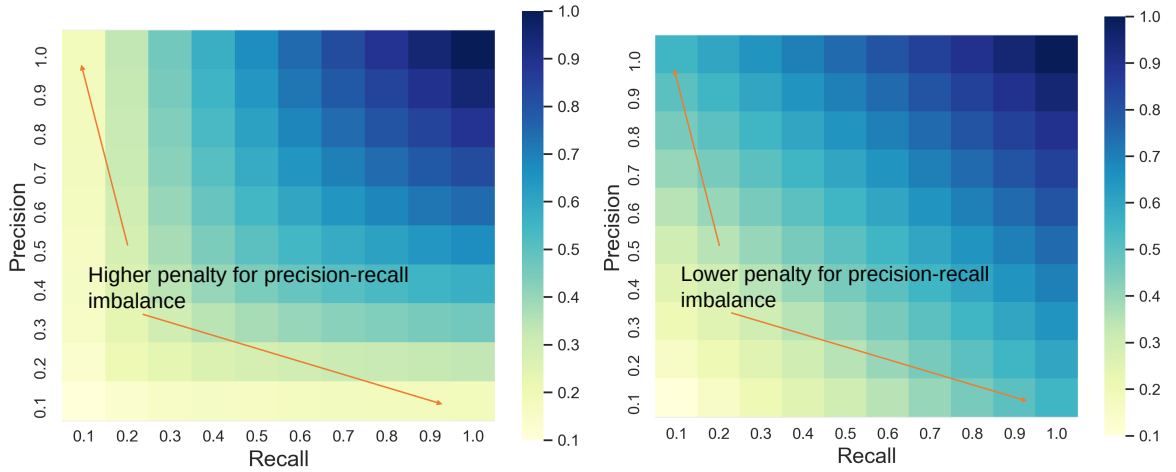
Definition 7. *F1-Score is defined as the harmonic mean of the precision score and the recall score of a classifier [Olson 2008].*

$$\text{F1-Score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Given the example confusion matrix C', the F1-Score is thus 0.33. An interesting question one can ask here is, while arithmetic mean is much more commonly used. To understand the reason, let us take a simple example. Let us assume that for a certain classification problem, the precision of a classifier is 0, but the recall is 1. Note that the arithmetic mean of the precision and recall in this case is 0.5. However, the Harmonic mean of the precision and recall is 0.

This means that if one of the measures, precision or recall, has a very high value and the other, a very low value, then the arithmetic mean overestimates the performance. In the considered example, the precision is 0, meaning that all the other data points not labelled as class c are wrongly classified as class c. This makes the point of classification meaningless. Thus, the harmonic mean of precision and recall produces a higher score for the model performance when the precision and recall are both high. Thus, F1-Score is a more realistic performance measure than the arithmetic mean of the precision and recall. Figure 2.1, demonstrates the phenomenon by taking several combinations of precision and recall and calculating both the Harmonic mean (see Figure 2.1(A)) and the Arithmetic mean (see Figure 2.1(B)).

Observe from Figure 2.1(A) that F1-Score puts equal weights on precision and recall. This is because, for example, a model with precision 0.7 and recall 0.9 will have the same F1-Score



(A) (B)

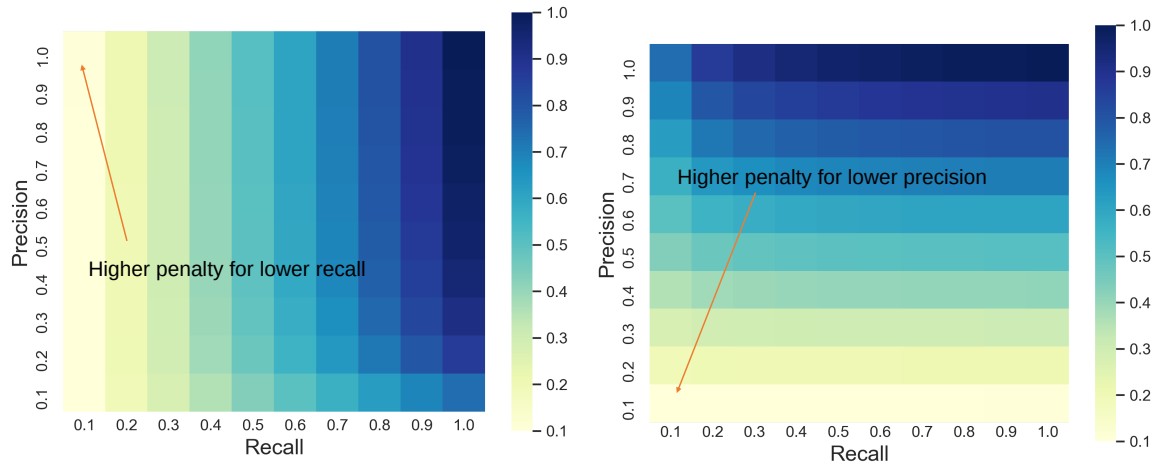
FIGURE 2.1: Figure showing several combinations of precision and recall to calculate A: Harmonic mean and B: Arithmetic mean.

as a model with precision 0.9 and recall 0.7. However, whether precision or recall is more important is entirely problem dependent. For example, the problem of detecting credit fraud is considered, the recall of the model would be more important. This is because detecting fraudulent transactions is more important here, and consequently one would prefer the FNs to be reasonably low. However, in certain biological applications the scenario could be entirely different. For example, suppose one seeks to detect from some data whether a certain therapy will work on some patients. In this case, precision would be a more important measure. The reason is, it can be dangerous for a patient if a therapy is not likely to work for that patient and yet, the classifier predicts that it will work. Thus, the FPs are desired to be reasonably low in this case.

To design a problem-specific performance measure related to precision and recall, one might thus consider a generalised version of the F1-Score called the F_β-Score. F_β-Score uses a positive real factor β chosen such that recall is β times as important as precision and is a generalised version of the F1-Score (special case with β=1) [Powers 2011]. Formally, the F_β-Score is defined as:

$$F_\beta\text{-Score} = \frac{(1+\beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$$

Figure 2.2, shows how the F_β-Score behaves depending on several combinations of precision and recall scores for $\beta = 5$ (see Figure 2.2(A)) and $\beta = \frac{1}{5}$ (see Figure2.2(B)). Note that, by definition, a choice of $\beta = 5$, implies that recall is 5 times more important than precision. In Figure 2.2(A), observe that when precision is 0.1 and recall is 1, the F_β-Score is higher than when precision is 1 and recall is 0.1. This gives an indication that a higher recall score gets more preference to a higher precision score. Note that, the converse situation is presented in Figure 2.2(B).



(A) (B)

FIGURE 2.2: Figure showing several combinations of precision and recall to calculate F_β-Score, for A: $\beta = 5$ and B: $\beta = 1/5$.

2.2.2 Other popular performance measures

Another performance measure that is also commonly used for imbalanced datasets, called the G-Mean, or the geometric mean. The G-Mean is defined as the geometric mean of precision and recall [García 2010].

$$\text{G-Mean} = \sqrt{\text{precision} \times \text{recall}}$$

It is thus evident that G-Mean $\geq$ F1-Score. Note that for the considered example confusion matrix C', the G-mean is $\sqrt{0.25 \times 0.5} = .353$. Compared to the arithmetic mean of precision and recall, the F1-Score has the tendency to penalise a model more if there is a high imbalance between precision and recall, that is if either of them is quite high. The G-mean is likely to find a balance between the two, which means G-means will not penalise the model as much as F1-Score for poor precision-recall balance but will not be as lenient as the arithmetic mean as well. This follows from the well-known inequality:

$$\text{Arithmetic Mean (AM)} \geq \text{Geometric Mean (GM)} \geq \text{Harmonic Mean (HM)}$$

Another performance measure relevant to imbalanced datasets that is gaining popularity is the Matthews correlation coefficient (MCC) [Matthews 1975]. Unlike the measures discussed so far, the range of this measure is $[-1, 1]$. The scores of 1 and -1 are achieved in the scenarios of perfect classification and perfect misclassification for all data points, respectively. A MCC score of 0 indicates that the classifier performance is as good as a random classification, where the probability of a data point belong to a class is equal.

Definition 8. *Given a confusion matrix C, by applying a certain classifier on a test dataset relevant to a classification problem, the balanced accuracy of the classifier is defined as [Matthews 1975]:*

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$$

The correlation between MCC and the F1-Score increases with the size of the dataset. A recent study claims 'The Matthews correlation coefficient (MCC), which is a more reliable statistical rate which produces a high score only if the prediction obtained good results in all four confusion matrix categories (true positives, false negatives, true negatives, and false positives), proportionally both to the size of positive elements and the size of negative elements in the dataset' [Chicco 2020]. The experiment they have conducted to come to this conclusion, involves a given hypothetical dataset with at most 500 samples and all possible combinations of TPs, TNs, FPs and FNs to form all possible confusion matrices. From these confusion matrices, they have calculated all possible values of F1-Scores and MCC scores. Interestingly, it was found that there was a certain confusion matrix for which, the MCC score was found to be -0.04, but the F1-Score was 0.95 [Chicco 2020]. For this certain case, number of positive instances were 91 as opposed to 9 negative instances. The TPs, TNs, FPs, FNs for this case were 90, 0, 0, 1 respectively, which clearly indicates a poor classification for the negative class. However, it generates a very high F1-Score as the class with fewer samples is assumed to be the negative class, which in practice this is not usually the case [Chicco 2020]. When the class with fewer samples is assumed to be the positive class, there is no strong evidence that the F-1 Score is misleading. Thus, F1-Score can still be used safely if the minority class is assumed to be the positive class, that is, assume that it is more important to detect the minority class in the context of a given problem. Note that for the considered example confusion matrix, C' the MCC measure is 0.25.
Another performance measure, called the Cohen's Kappa measure (κ), is also sometimes used for evaluating the performance of imbalanced datasets [Jeni 2013]. The κ measure was originally proposed by J.A. Cohen in the field of psychology to compare the judgements of two evaluators [Cohen 1960]. The κ measure, similar to the MCC measure, can assume values in the range $[-1, 1]$.

Definition 9. *Given a classification problem, the κ measure is formally defined as:*

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

where, P_o is the measure of the observed agreement among the chosen classifier and the ground truths of the classification problem. P_e is the measure of agreement by chance, among a chosen classifier and the ground truths of the classification problem.

To get a better feel of how to calculate the κ measure, let us calculate it for the proposed example confusion matrix C'. Note that P_o is always equivalent to the accuracy measure, which is 0.8, considering confusion matrix C'. Now, calculating P_e is the tricky part. Note that the first evaluator, the considered classifier, judges $20\%(15 + 5 = 20)$ of all the samples to be positive. The second evaluator, the ground truth, judges $10\%(5 + 5 = 10)$ of all the samples to be positive. Thus, the probability that both evaluators will agree on the positive classes by chance is $0.2 \times 0.1 = 0.02$. Similarly, the first evaluator, the considered classifier, judges $80\%(75 + 5 = 80)$ of all the samples to be negative and the second evaluator, the ground truth, judges $90\%(75 + 15 = 90)$ of all samples to be negative. Thus, the probability that both evaluators will agree on the positive classes by chance is $0.8 \times 0.9 = 0.72$. Thus, the total probability of agreement of the evaluators on the positive class and the negative

class is $0.72 + 0.02 = 0.74$. Thus, the κ, measure for the considered example matrix C', is $(0.8 - 0.74)/(1 - 0.74) = 0.230$. However, there is no universally set convention on what value of κ, which indicates a good enough classification. Just like the F_β-Score, the κ measure also has a weighted version, but in case of imbalanced datasets, it has been used quite rarely [Cohen 1968].

All the performance measures discussed so far have been subject to some criticism in particular contexts. Due to their simplicity, easy interpretation, F1-Score and G-Mean are more popularly used in comparative or benchmark studies. However, F1-Score and G-Mean however concentrate on how good the classification is for a single class. They are popular because mostly imbalanced classification problems are binary class problems. For multi-class problems, possibly with multiple minority classes, the κ measures would be more suitable.

2.3 Receiver operating characteristic curve and precision recall curve

Classification models are first trained on labelled data and then tested on test data. When a trained classification model is provided with the test data, ideally, it can predict the probability of each data point belonging to each class. Consider a binary classification problem. For each test data point, a classification model will estimate the probability of the data point to be belonging to the positive class. Conventionally, if this probability is greater than 0.5, then the data point can be classified as a positive class instance. For example, consider a k-Nearest Neighbour classifier, for a particular test data point. If k' of its k nearest neighbours from the training data are from positive class, then the probability of the test data point to belong to the positive class can be quantified as k'/k. Note that, this makes the confusion matrix dependent on the conventional probability threshold 0.5. Which means, if this threshold varies, a data point classified as a positive instance, which may end up being classified as a negative instance. Note that, all performance measures defined so far, are based on the confusion matrix where the quantities TP, TN, FP and FN derived from the conventional threshold, that is 0.5.

Let us assume now that some trained classifier produces probability scores of belonging to the positive class for all test data points, such that all scores are all in the range of $[0.4, 0.6]$. Now, if a decision threshold of 0.3 to decide on the positive class (i.e. a predictive probability greater than 0.3 for a test data point implies that data point is predicted to be in the positive class by the classifier) is considered for constructing a confusion matrix, all test data points will be predicted as positive class instances. Similarly, if the decision threshold is 0.7, all the test data points will belong to the negative class. If the classifier predicted the probabilities of its training experience better, that is, all the positive class samples would be assigned a high predicted probability (say, more than 0.95) of belonging to the positive class and all the negative class samples would be assigned a low predicted probability of belonging to the positive class (say, more than 0.95), then independent of the choice of the decision threshold (for probability of belonging to the positive class), the classifier would do a reasonably good job. Thus, ideally, a classifier should be able to learn from the training data irrespective

Predicted Prob	Labels	Threshold	TPR/Recall	FPR	Precision
0.625	1	0.625	0.333	0.00	1
0.483	0	0.483	0.333	0.142	1
0.475	0	0.475	0.333	0.285	0.5
0.412	0	0.412	0.333	0.428	0.333
0.358	1	0.358	0.666	0.428	0.4
0.305	0	0.305	0.666	0.571	0.333
0.275	1	0.275	1	0.571	0.428
0.205	0	0.205	1	0.714	0.3
0.187	0	0.187	1	0.857	0.3
0.125	0	0.125	1	1	0.3

TABLE 2.1: Table showing an example of predictions of a hypothetical classifier on 10 data points and thereby calculating TPR, FPR, precision and recall from the same. The column Predicted Prob shows the predicted probabilities of belonging to a positive class for 10 data points by some hypothetical classifier.

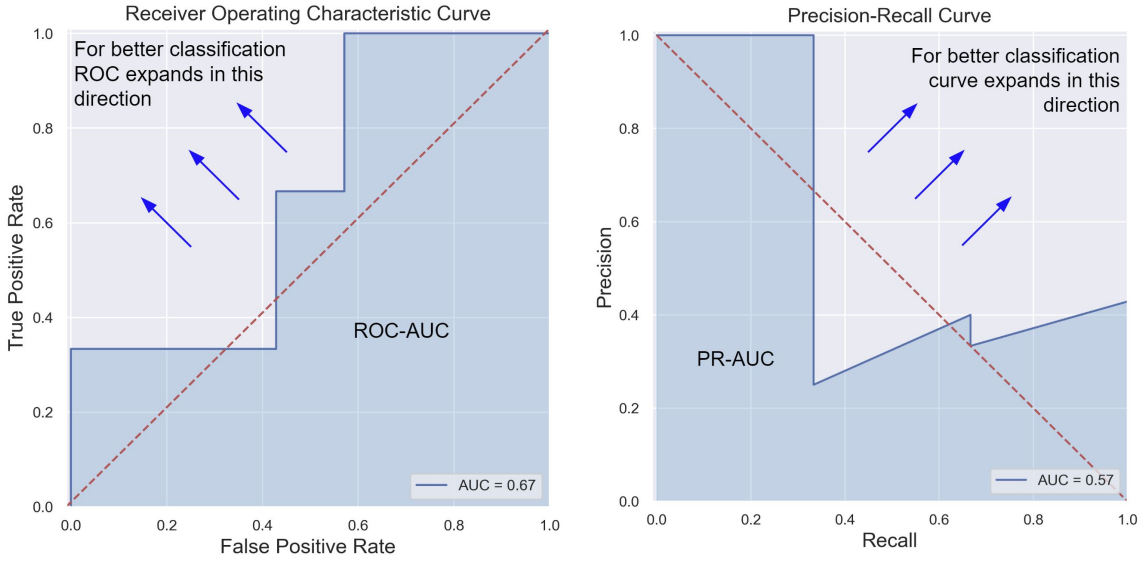
of the choice of the decision threshold. To evaluate how well a classifier achieved this, is the motivation behind defining the very popular performance measure called Receiver Operating Characteristic curve (ROC curve) [Fawcett 2006].

To understand the ROC curve, two performance measures based on which the curve is constructed are relevant. One is the sensitivity or recall or True Positive Rate (TPR) that has been already discussed is Definition 5. The other one is called specificity.

Definition 10. *Given a confusion matrix C, by applying a certain classifier on a test dataset relevant to a classification problem, the True Negative Rate or specificity of the classifier is defined as:* $\frac{TN}{TN+FP}$.

In simpler words, specificity quantifies the ratio of negative class data points that are correctly classified by the classifier. Note that for the proposed example confusion matrix C', the specificity would be, $75/(75+15) = 0.833$. Now that the motivation is clear and the foundation behind ROC curve is established, it is time to discuss the construction of a ROC curve. A ROC curve is a plot with False Positive Rate (FPR) on the x-axis and TPR on the y axis. Note that FPR is the same as $1 -$ specificity and TPR is sensitivity. Given a confusion matrix, one can thus derive the TPR and FPR. But given a decision threshold for the probability of a test data point belonging to the positive class, one can obtain only one confusion matrix. Thus, given one such decision threshold, one obtains only one value for TPR and FPR. Now, if one calculates the confusion matrices for several decision thresholds ranging from $[0, 1]$, one will obtain an ordered pair of (FPR, TPR) for each of the decision thresholds. The ROC curve is obtained by plotting all the ordered pairs of (FPR, TPR) obtained from different decision thresholds and drawing a curve by linear interpolation among the plotted points.

To demonstrate the idea of a ROC curve, a hypothetical example is presented with the prediction results of a hypothetical classifier on 10 data points. The predicted probability (hypothetical) of belonging to the positive class for each of these data points is shown in the column 'Predicted Prob' of the Table 2.1. The column 'Labels' show the actual labels for the dataset. Note that out of 10, there are only 3 data points from the positive class from the example, labelled as 1. For each of the 'Threshold's, one can compute a confusion matrix given the predicted probabilities and labels. For example, with a threshold of 0.625, $\text{TP} = 1$, $\text{TN} = 7$, $\text{FP} = 0$ and $\text{FN} = 2$. With these values, one can easily calculate the



(A) (B)

FIGURE 2.3: Figure showing A: Receiver Operating Characteristic (ROC) curve and B: Precision Recall (PR) curve for an imbalanced test dataset for which a hypothetical classifier performance described in Table 2.1. The region shaded in blue in each diagram represents the Area Under the Curve (AUC). The arrows indicate the direction in which the AUC (region shaded in blue) would expand in case of a better classifier.

quantities TPR and FPR for the first threshold. This can be repeated for all the assumed thresholds to finally calculate the TPRs and FPRs of the respective thresholds as shown in Table 2.1. The respective ROC curve is plotted in Figure 2.3(A). The red dotted line in Figure 2.3(A) signifies the expected ROC curve for a random classifier (a classifier that predicts randomly). Once the ROC curve is obtained, the Area Under the Curve (AUC) is calculated. The area under the ROC curve (ROC-AUC) for the example is 0.67. Here, I avoid further discussion on the computational details of how this area is calculated, but rather focus on what it represents. In Figure 2.3 (A), this is the area of the blue shaded region, and it quantifies how well the classifier has learnt on the training data, such that it can make reasonably good predictions on the test data independent of decision thresholds. In the proposed example, observe that the classifier performance is not very good. Ideally, for better classification, the blue shaded area can extend in the direction of the arrows towards the top left corner. For a perfect classifier, the blue shaded region will thus cover the whole area of the graph and attain a ROC-AUC $= 1$. Note that this translates to the situation when the TPR is very high, independent of the value of FPR. Usually when the decision threshold is low, most samples TPR will be quite high because most positive samples will be correctly classified with the low threshold. But at the same time, most negative samples are likely to be wrongly classified, which make the FPR also quite high for low thresholds. However, the better the classification models can maintain the high TPR with increasing decision thresholds, the better will be its ROC-AUC.

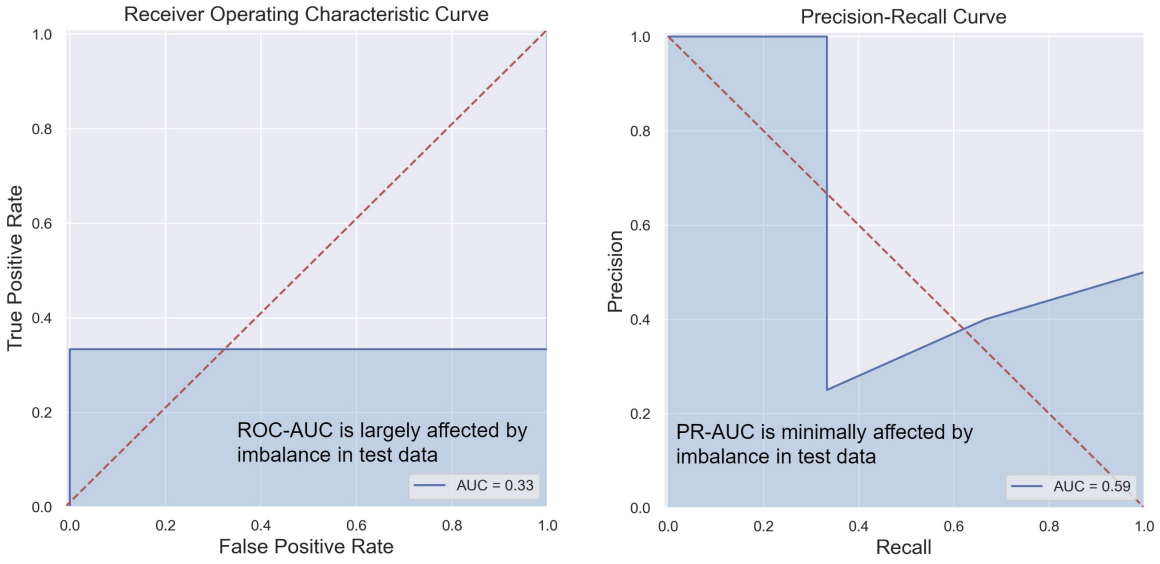
Another curve, very similar to the ROC curve, is especially relevant for imbalanced datasets. It is called the Precision-Recall curve (PR curve) [Saito 2015]. The Precision-Recall curve plots recall(TPR) in the x-axis and precision in the y-axis for several decision

thresholds. Similar to ROC-AUC, the AUC of the PR curve (PR-AUC) quantifies how good the predictions of the classifier are. In the example, note that the PR-AUC attains a value of 0.57. The PR curve is shown in Figure 2.3(B) along with the AUC in blue shaded region. The red dotted line in Figure 2.3(B) signifies the expected PR curve for a random classifier. A good PR-AUC score indicates that the precision of the classifier is always good, irrespective of the recall. The interpolation used to construct the curve itself from several (recall, precision) ordered pairs is nonlinear, in contrast to that of the ROC curve, which follows a linear interpolation.

There are debates on the applicability of the performance measures ROC-AUC and PR-AUC in the context of imbalanced datasets. Many studies use ROC-AUC as their evaluation metric since it is one of the most popular performance measures in general (for example [Bellinger 2018]). However, some data scientists believe that the PR-AUC is more informative than ROC-AUC in the case of imbalanced datasets [Saito 2015]. This essentially means that, if the model performance is tested on the balanced subset, in some cases it is observed that the ROC-AUC is lesser than when it is calculated over the imbalanced test set. Thus, using the ROC-AUC may lead one to overestimate the model performance if the test set is imbalanced (which naturally occurs for imbalanced datasets). The PR-AUC does not show or demonstrate this behaviour, and thus is considered more robust to imbalance in the test set [Saito 2015]. To visualise this at least heuristically, 6 data points are extracted out of the 10 data points shown in Table 2.1. 3 of these are of course the minority class data points with labelled 1. The three other data points are the majority class data points, which can be roughly described as poorly learned. These are the majority class data points with 'Predict prob' values of 0.475, 0.483 and 0.412. With this balanced dataset, once can create the ROC and PR plots as shown in Figure 2.4. Observe that for the balanced test dataset, there is a dramatic decrease in the ROC-AUC value, which is now 0.33. Even more interestingly, the PR-AUC value 0.59 remains almost unchanged compared to that of the imbalanced case, where it was 0.57. It can thus be concluded that if the test dataset is imbalanced, which is more likely to occur during handling imbalanced datasets, it is advisable to use the more robust measure PR-AUC.

2.4 Wilcoxon's signed rank test for model comparison

Until now, I have discussed what kinds of performance measures are appropriate to calculate classifier performance. However, the field of ML is now so dynamic that every year new algorithms emerge that claim to be better than the state-of-the-art. Usually, the new algorithms are benchmarked against existing algorithms on several publicly available datasets. Then the average performance of the algorithms are compared over all datasets to decide whether the new algorithm beats the state-of-the-art in terms of chosen performance measures. However, with so many alternative ML algorithms available for almost every particular problem, now it is not good enough just to show that the new algorithm is better than the state-of-the-art. It is also important to show that the new algorithm beats the existing state-of-the-art by a statistically significant margin. For this, one can take the help of statistical hypothesis testing. A statistical hypothesis is an assertion on a test statistic (a



(A) (B)

FIGURE 2.4: Figure showing A: Receiver Operating Characteristic (ROC) curve and B: Precision Recall (PR) curve for a balanced subset of an imbalanced test dataset for which a hypothetical classifier performance described in Table 2.1. Note that the ROC-AUC in the balanced case is significantly lower than the imbalanced case shown in Figure 2.3. The PR-AUC however changes very little compared to the imbalanced case and is thus a more robust performance measure on imbalanced datasets.

function of the data, such as mean or standard deviation), that may or may not be true. A hypothesis test is associated with a null (H_0) and an alternative hypothesis (H_1). The goal of a statistical test is to decide whether to reject the null hypothesis. For this, the value of the test statistic is calculated from the data. Usually the probability distribution of the test statistic is known and one can calculate a p-value or the probability that of observing a test statistic as extreme as the obtained value, assuming the null hypothesis is true. If the p-value is lower than a certain threshold, the null hypothesis is rejected. There are many statistical tests that are applicable to several types of situations.

The Wilcoxon's signed rank test can be effectively used to compare an ML model performance against other ML models, in terms of any number of performance measures [Wilcoxon 1945, Tarawneh 2020]. This test is effective for comparative studies because it is safer than parametric tests because it refrains from assuming homogeneity or normal distribution of data. Therefore, it can be applied to any classifier evaluation measure. The Wilcoxon test aims to find if a null hypothesis is true or not. The null hypothesis H_0 assumes that there is no significant difference between the classification results (observations) obtained from two different methods. The null hypothesis is rejected if the p-value of the Wilcoxon test is less than $\alpha = 0.05$ [Tarawneh 2020].

The p-value alone is informative enough and does not provide information about the relationship strength between variables [Tarawneh 2020]. The p-values do not reveal whether the results are significantly different in favour of an ML model (that is being benchmarked against others) or against it. For that, it is recommended to use the metrics W_+, W_- and R that are calculated using the following steps [Tarawneh 2020]:

1. For each data pair of model predictions, the difference between both predictions is calculated and stored in a vector D, excluding the zero difference values.

2. The signs of the difference is recorded in a sign vector S.

3. The entries in $|D|$ are ranked, forming a vector R'. In case of tied ranks, an average ranking scheme is adopted. This means, after ranking the entries of $|D|$ are ranked using integers and then, in case of tied entries, the average of the integer ranks are considered as the average rank for all the respective tied entries with a specific tied value.

4. The component-wise product of S and R' provides us with the vector W, the vector of the signed ranks. The sum of absolute values of the positive entries in W is W_+ and the sum of absolute values of the negative entries in W is W_-. After this define, $W_R = \min\{W_+, W_-\}$

5. Then the test statistic Z is calculated by the equation

$$Z = \frac{W_R - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24} - \frac{\Sigma t^3 - \Sigma t}{48}}} \tag{2.2}$$

   where n is the number of components in D and t is the number of times some i-th entry occurs in R', summed over all such repeated instances.

6. Finally, R is calculated using $R = \frac{|Z|}{\sqrt{N}}$, where N is the total number of datasets compared.

Note that a higher value W_+ for an ML model that is being benchmarked against other models, indicates towards a superior performance of that particular ML model and the value of R indicates towards how superior (with a higher W_+)/ inferior (with a higher W_-) the performance of the ML model of interest is, compared to the other models. Some researchers have considered ranges of $R \leq 0.1$, $0.1 < R \leq 0.5$ and $R > 0.5$ to be indicators for small, medium and high degree of change (improvement or deterioration) in the predictive performance respectively [Tarawneh 2020].

Now that the preliminaries and motivations of the research are established, it is reasonable to proceed to the state-of-the-art relevant for my research. The next chapter presents the history and relevance of synthetic oversampling through convex space modelling in the context of imbalanced classification problems.

Chapter 3

Oversampling Techniques for convex space modelling

Synthetic oversampling has been one of the most popular approaches to address the problem of imbalanced classification. Rigorous research over the past two decades has resulted in a family of oversampling algorithms that rely on modelling the convex space of the minority class, generate synthetic samples from the modelled convex space and improve classifier performances on imbalanced datasets. This chapter presents some relevant algorithms from this family of oversampling algorithms that have been instrumental in my research, starting from the SMOTE algorithm. Starting from some early extensions of the SMOTE algorithm, the chapter proceeds to discuss some extensions of SMOTE which integrate the algorithm with other unsupervised ML algorithms. Finally, the chapter is concluded with some SMOTE extensions based on classifier specific frameworks and some additional state-of-the-art SMOTE extensions.

3.1 SMOTE algorithm and its limitations

I discussed in Chapter 1.2.1 about the popularity of oversampling and undersampling in handling of the problem of imbalance. Followed the attempt of Lee to generate oversamples using Gaussian noise in 2000, Chawla *et al.* proposed with the idea of **Synthetic Minority Oversampling Technique (SMOTE)** in 2002 [Lee 2000, Chawla 2002]. SMOTE has been widely used for handling imbalanced datasets over the last decade. Till date, it is a popular choice of data scientists to tackle the problem of imbalance. There has also been an immense amount of research on attempts to improve this technique. To make discussions simpler for now, imbalanced datasets for binary classification problems are considered, that is, imbalanced datasets having only a minority and a majority class.

The SMOTE algorithm generates synthetic minority class samples by linear interpolation of the minority class. The synthetic sample generation approach is quite generic in its construct and focuses on the feature space of a given dataset. It has been described by [Chawla 2002] using the following approach:

Let us assume that x_1 is an arbitrary minority class sample in an imbalanced dataset C. Let us denote the set of k the nearest minority class neighbours of x_1 by $N_k^{C_{min}}(x_1)$. Let $N_k^{C_{min}}(x_1)$ be the neighbourhood of the minority class data point x_1. Note that, for the sake

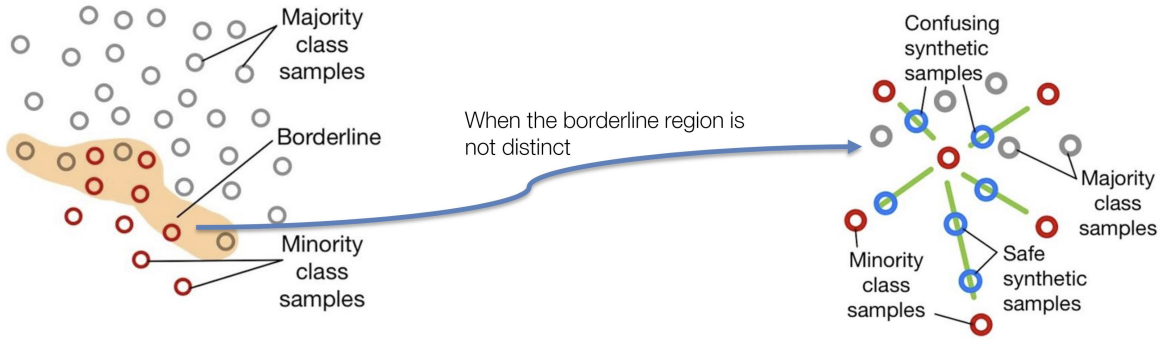


FIGURE 3.1: Figure demonstrating minority class over generalisation by the SMOTE algorithm, which is considered to be one of the key limitations of SMOTE

of consistency, these notations are maintained throughout the chapter. Let $x_2 \in N_k^{C_{min}}(x_1)$, $x_2' \neq x_1$ be a random minority class data point. A newly generated synthetic sample is described by,

$$S = x_1 + u(x_2 - x_1) \tag{3.1}$$

where $0 < u < 1$. More samples can be generated from $N_k^{C_{min}}(x_1)$ simply by choosing separate random neighbours of x_1 within $N_k^{C_{min}}(x_1)$. Ultimately, this process can be repeated over all data points in the minority class to generate a population of synthetic samples from the minority class. Equation 3.1 happens to be the heart and soul of most of the convex-combination based oversampling techniques existing today.

3.1.1 Criticisms of the SMOTE algorithm

Soon after the SMOTE algorithm was proposed in 2002, researchers started to point out several of its limitations. There have been research works solely dedicated to rigorous analysis of the limitations of SMOTE [Blagus 2013]. One of the main limitations of the SMOTE algorithm is its tendency to over generalise the minority class [Puntumapon 2012]. This means, that the synthetic samples generated by SMOTE often makes classifiers biased towards the minority class. This improves the minority class classification, of course, but at a cost of higher number of misclassification of the majority class samples. This jeopardises the balance of the classification models.

Researchers over the years have provided many explanations for such behaviour of SMOTE oversampling. For example, while oversampling, SMOTE considers only the distribution of the minority class and not the majority class. Several algorithms have been proposed where the minority class distribution is considered relative to the distribution of the majority class while oversampling [Han 2005, Barua 2013]. Another explanation for the minority class over-generalisation by SMOTE is that SMOTE treats all the minority class samples with equal importance. This direction of thought proposes to add weights to the minority class samples such that they quantify the importance of the samples in context to the oversampling process.

In their research Blagus *et al.* points out at an interesting direction in this context. In this article, they calculate the variance of a SMOTE generated synthetic sample. Following this research, I construct an analytical framework in Chapter 4.1.2, where I argue about my interpretation for minority class over-generalisation of the SMOTE algorithm. Intuitive explanation of the analytical result is that, in imbalanced classification scenarios where the decision boundaries are not too well-defined, SMOTE-generated synthetic samples having a high degree of local variance tends to interfere with the majority class. Figure 3.1 shows a demonstration of the problem.

One more important limitation of SMOTE and SMOTE-based algorithms is their classifier dependent performance. This means some oversampling algorithms performs well only with some classifiers. This is evident from recent comparative studies in Blagus *et al.* and Kovács *et al.* [Blagus 2013, Kovács 2019]. I also propose a solution to this problem in the form of the multi-schematic classifier-independent ProWRAS algorithm in Chapter 6.

3.2 Some early extensions of the SMOTE algorithm

3.2.1 Borderline based oversampling

Borderline-SMOTE is an early attempt to improve on the SMOTE algorithm [Han 2005]. Classifiers, while dealing with classification problems, try to create a decision boundary, a hypothetical boundary that helps a classifier to separate several classes in some labelled data. Samples lying close to the decision boundary are in the regions that are harder to learn for the classifier. For imbalanced datasets, generating a decision boundary is even more difficult to begin with, due to the scarcity of data points in the minority class. SMOTE empowers a classifier, to some extent, by generating synthetic minority class data points, helping the classifier to gather a better experience of the minority class.

The key idea of the Borderline-SMOTE algorithm is to emphasise particularly the 'confusing' minority class data points, which are termed as 'borderline' samples. These constitute of the minority class data points that are closer to the majority class data points and thus, can be more confusing for the classifiers to identify. Unlike SMOTE, which creates synthetic samples from all the minority class C_{min} of a dataset, Borderline-SMOTE creates synthetic samples from only the borderline minority class data points. Now, let us discuss how the Borderline-SMOTE detects the borderline samples.

Borderline-SMOTE uses a simple definition for the borderline samples. For a minority class data point $x \in C_{\text{min}}$, let $N_l^C(x)$ denote the l-nearest neighbours of x is the whole dataset C. Let $m(x)$ denote the number of members of $N_l^C(x)$ that are from the majority class C_{maj}, formally speaking $m(x) = |N_l^C(x) \cap C_{\text{maj}}|$. Borderline-SMOTE algorithm considers $x \in C_{\text{min}}$ a borderline minority class data point if,

$$\frac{l}{2} \leq m(x) < l \tag{3.2}$$

that is, at least half of the l-neighbours must be majority class data points to qualify as a x a borderline sample. If $m(x) = l$, the sample x, is assumed to be an outlier or noise by the

algorithm and therefore, the algorithm does not involve it in the oversampling procedure. Moreover, if $m(x) < \frac{l}{2}$, then it is assumed, that the sample x is likely to be far away enough from the decision boundary or the so-called borderline, and thus can be excluded from the process of generating oversamples.

After the borderline samples are identified through Equation 3.2, the synthetic samples are generated by applying SMOTE only on the Borderline samples. Note here that, while applying SMOTE on the borderline samples a different neighbourhood size k can be chosen, which is completely unrelated to the neighbourhood size l, chosen to decide on the borderline samples themselves. Also, while oversampling the borderline samples, the neighbourhoods are chosen with respect to the minority class only, as witnessed for SMOTE.

It is noteworthy that, Borderline-SMOTE algorithm has a second variant as well, called Borderline-SMOTE2, which adopts a sightly different strategy. It not only generates synthetic samples from each sample in the borderline set and its nearest neighbours in C_{min}, but also does that from its nearest neighbours in C_{maj}. For a nearest neighbour $y \in C_{\text{maj}}$ of a borderline data point x, an oversample is generated similarly as described in Equation 3.1. The only difference is that the randomly chosen u is constrained to be $0 < u \leq 0.5$, to ensure that this the synthetic sample is located closer to the minority class data point x itself rather than the majority class data point y.

The **Safe-level SMOTE** algorithm proposed in [Bunkhumpornpat 2009], philosophically, is the opposite of the Borderline-SMOTE algorithm. However, instead of generating oversamples from the borderline regions, as done in Borderline-SMOTE, Safe-level SMOTE generates synthetic samples from relatively 'safe' regions. To determine how 'safe' a minority class data point $x \in C_{\text{min}}$ is, the algorithm uses a quantity called safe-level ratio for x, which is denoted as $r(x)$. First, another quantity called, safe-level $l(x)$, is defined for an arbitrary $x \in C_{\text{min}}$ as $l(x) = |\{N_m^C(x) \cap C_{\text{min}}\}|$. This means that, $l(x)$ quantifies the number of minority class samples among m-nearest neighbours of x in the whole imbalanced dataset C. Let $y \in C_{\text{min}}$ be a randomly selected sample such that $y \in N_m^{C_{\text{min}}}(x)$. The quantity $r(x)$, for some $x \in C_{\text{min}}$ is defined as

$$r(x) = \frac{l(x)}{l(y)} \tag{3.3}$$

Note that, if $l(y) = 0$, then $r(x)$ becomes undefined as per Equation 3.3 and so it is assumed that $r(x) = \infty$. For $x, y \in C_{\text{min}}$ and $y \in N_k^{C_{\text{min}}}(x)$, a SMOTE generated synthetic sample is described by Equation 3.1. For SMOTE, the random variable u determines a random point in the line joining x and y, which is chosen to be a synthetic sample S. In case of Safe level SMOTE, depending on values of $r(x)$, $l(x)$ and $l(y)$, the value of u is determined.

Firstly, if $r(x) = \infty$ (implying $l(y) = 0$) and $l(x) = 0$, then it is evident that both x and y are closer to majority class data points in C. For this reason, x is assumed to be an outlier or noise and is not used to generate oversamples.

Secondly, if $r(x) = \infty$ (implying $l(y) = 0$) and $l(x) \neq 0$, then it means that x has minority class data points in its k-neighbourhood in C, but some random minority class neighbour y of x has no minority class data points in its k-neighbourhood. In this case, u is assumed to be 0. Thus, no new synthetic samples are generated in this case as well.

Thirdly, if $r(x) = 1$ (implying $l(x) = l(y)$), u is chosen in exactly the same fashion as done in SMOTE, that is, u is randomly chosen such that $0 < u < 1$. Note that, in this, there is no clear effort of restricting the samples to the safe region since it is not deterministic whether x or y is a safer minority class data point. In the fourth case, if $r(x) > 1$ (implying $l(x) > l(y)$), indicating that x is surrounded by more minority class neighbours compared to y, and thus x is more 'safe'. In this, u is chosen randomly such that $0 < u \leq \frac{1}{r(x)}$. This ensures that the synthetic sample S is generated closer to x, since, x is more safe.

Finally, if $r(x) < 1$ (implying $l(x) < l(y)$), indicating that y is surrounded by more minority class neighbours compared to x, and thus y is more 'safe'. In this, u is chosen randomly such that $1 - r(x) \leq u < 1$. This ensures that the synthetic sample S is generated closer to x.

3.2.2 Weighting minority class samples

ADASYN is also one of the earlier attempts to develop the SMOTE algorithm [He 2008]. The Borderline-SMOTE algorithm concentrates on oversampling from the minority class samples that are too close to the majority class samples or are in "borderline" between the majority and minority class. This might cause the algorithm to ignore other important portions of the minority class data space. This causes the oversampled data distribution of the minority class to be too dense in the borderline regions, while too sparse in the other regions. The key idea of ADASYN algorithm is to "adaptively" decide the number of synthetic samples to be generated from each minority class data point, depending on their importance in improving the learning capability of a classifier. This importance factor is again determined by the number of nearest neighbour a given minority class data points that are actually from the majority class.

The number of synthetic samples to be generated is decided by the balancing factor β. The number of synthetic samples to be generated is thus defined by equation.

$$S_{\text{min}} = (C_{\text{maj}} - C_{\text{min}})\beta \tag{3.4}$$

After this, a weight distribution $D(x)$ is defined for every $x \in C_{\text{min}}$. To define $D(x)$, first a quantity r is defined by the equation

$$r(x) = \frac{\delta}{k} \tag{3.5}$$

such that $\delta = |\{y : y \in N_k(x) \cap C_{\text{maj}}\}|$, or simply the number of majority class samples among the k-nearest neighbours of x. Once for all $x \in C_{\text{min}}$, the quantity $r(x)$ is calculated, $r(x)$ is normalised to give it the form of a distribution, thereby creating the distribution $D(x)$. Formally speaking,

$$D(x) = \frac{r(x)}{\Sigma_{x \in C_{\text{min}}} r(x)} \tag{3.6}$$

The distribution $D(x)$ quantifies the importance of a given minority class sample x. Finally, how many synthetic samples are to be generated from a given minority class data point x is

given by

$$g(x) = D(x)S_{\min} \tag{3.7}$$

Finally using the principle of SMOTE, ADASYN generates $g(x)$ synthetic samples for each $x \in C_{\min}$.

The SMOTEBoost Algorithm, proposed in [Chawla 2003], was one of the earlier follow-ups of the SMOTE algorithm. It involves combining the idea of SMOTE to standard boosting procedure. The component of SMOTE oversampling in this model is aimed at enriching the minority class data distribution by oversampling. The boosting component of this model is aimed at improving the overall accuracy that is sometimes compromised by SMOTE.

I discussed the core idea behind boosting as previously discussed in Section 1.2.2. The idea of Boosting involves a weak classifier that iteratively learns to make better decisions by correcting wrong decisions made by its predecessor classifier in the iteration. **SMOTEBoost** particularly uses AdaBoost classifier as the underlying Boosting algorithm.

For imbalanced data however, there is a problem with a standard boosting algorithm, that has been addressed by Chawla *et al.* A boosting algorithm samples for training data from a pool of data that mostly majority class data points, since the dataset is imbalanced. AdaBoost algorithm treats False positives and False negatives equally. AdaBoost algorithm is known to reduce both bias and variance of the fit in the final ensemble. However, due to the imbalance in the dataset, there is a strong learning bias towards the majority class. Thus, the classifier might be biased in learning the majority class properly and deficient in learning the minority class [Chawla 2003].

The goal of SMOTEBoost is to reduce the biased inherent to the learning procedure due to the class imbalance by increasing the sample weights of the minority class data points. The logic behind this is that, introducing SMOTE generated oversamples at every stage of boosting, will enable the classifier to sample more from the minority class. This, in turn, helps the classifier to learn better and broader decision regions of the minority class [Chawla 2003].

Let us now go into the details of the algorithm. The SMOTEBoost algorithm starts with a labelled training set $S = \{(x_i, y_i)\}$, $i \in \{1, \ldots, m\}$ and $x_i \in X$, $y_i \in Y$ as input. $C_{\min}$ be the minority class. First, a weight distribution is initialised for every training sample carrying equal weight $D_1(x_i) = \frac{1}{m}$, for every i. A total of T weak classifiers $t_j : j \in \{1, \ldots, T\}$ are to be sequentially trained. Before training an arbitrary weak classifier t_j, the minority class $C_{\min}$ is oversampled and the distribution D_j is modified. Recall, that a SMOTE-generated oversample is given by $O = x_m + u(x_n - x_m)$, for some minority class samples x_m and x_n and u is some constant such that $0 < u < 1$. Thus, the weight of a newly generated oversample O can be calculated as $uD_j(x_m) + (1-u)D_j(x_n)$. Thus, the weights of a weight distribution is then defined over the oversampled training distribution S'. Initially, a weak classifier t_1 is then trained on the training dataset with the initial weight distribution D_1 with all training samples bearing equal weights, to build a weak hypothesis $h_1 : X \times Y \to [0, 1]$. For every weak classifier t_j, the weak hypothesis $h_j : X \times Y \to [0, 1]$, can be thought of as

the prediction of the classifier t_j on the weighted training set. The hypothesis is then tested against the ground to truth to identify the set of misclassified training data B_j. The pseudo loss for t_j is then calculated by the equation [Chawla 2003]:

$$\epsilon_{t_j} = \Sigma_{x_i \in B_j} D_j(x_i) \left(1 - h_1(x_i, y_i) + h_1(x_i, \hat{y}_i) \right) \tag{3.8}$$

where $\hat{y}_i$ is the predicted class of x_i by weak classifier t_j. The predictive confidence value of the weak classifier is then calculated as $\beta_{t_j} = \frac{\epsilon_{t_j}}{1-\epsilon_{t_j}}$. The distribution D_{j+1} is then updated by the equation [Chawla 2003],

$$D_{j+1}(x_i) = D_j(x_i) \beta_j^{\frac{1-h_1(x_i,y_i)+h_1(x_i,\hat{y}_i:\hat{y}_i \neq y_i)}{2}} \tag{3.9}$$

The updated D_{j+1} is then normalised and used to train weak learner t_{j+1}. The final hypothesis takes into consideration the decisions of the weak learners to finally make its predictions [Chawla 2003].

$$H_f(x) = \arg\max_{y \in Y} \Sigma_{t=1}^{T} h_t(x, y) \log \frac{1}{\beta_t} \tag{3.10}$$

3.3 Integration of SMOTE with unsupervised learning

3.3.1 SMOTE with clustering algorithms

One of the problems that is very common while dealing with imbalanced datasets is related to the spacial distribution of the minority class samples. Sometimes it might occur that the minority class data points are distributed over several clusters. Majority class data points may lie between these minority class clusters. In such cases, considering the distributions of these clusters might be important. Thus, some extensions of SMOTE have been built with the integration of several clustering techniques with SMOTE. In this section, I will discuss some of these algorithms.

K-means SMOTE, as implied from the nomenclature itself, is an integration of the K-means clustering algorithm with SMOTE oversampling [Douzas 2018]. The K-means SMOTE algorithm is realised through three steps.

1. **Clustering:** The clustering step clusters the whole input data. This divides the data into K clusters, where K is a pre-specified number defined by the user.

2. **Filtering:** The filtering step is used to identify the clusters with a high proportion(by default more than 50%) of minority class data points. It then distributes the number of synthetic minority class samples to be generated over such identified clusters. The distribution is done such that more synthetic samples would be generated for the clusters with sparse distribution of minority class samples.

3. **Oversampling:** Finally, SMOTE is applied to generate predefined synthetic samples from each of the clusters prioritised by the filtering step.

Note that, in the filtering step, for determining the distribution of the synthetic samples to be generated, the filtered clusters are assigned sampling weights between zero and one. A higher value of sampling weight for a certain sample corresponds to a low density of minority class samples in the cluster containing that sample. The density of minority class samples in a filtered cluster is measured relative to the average density of minority class samples over all filtered clusters. The computation of the sampling weights is realised through the following steps.

1. For a filtered cluster f, the Euclidean distance matrix is calculated for the minority class samples from that clusters only.
2. The mean distance among samples is then calculated by taking the mean of all non-diagonal elements in this distance matrix. Let us denote this by M_f
3. The density of the cluster f is then defined as the ratio of number of minority class data points in f to the quantity $(M_f)^m$, where m is the number of features. The inverse of density is termed as the sparsity of the cluster f.
4. Finally, the sampling weight of a cluster f is defined as the ratio of the sparsity measure of f to the sum of the sparsity measures of all the filtered clusters.

Note that, the last step ensures that the sum of the sampling weights of all the filtered clusters is always one. Thus, the sampling weight can be multiplied by the overall synthetic samples to be generated to obtain the desired number of synthetic samples to be generated from each cluster.

DBSMOTE is another extension of SMOTE that explores the integration of a clustering algorithm DBSCAN with SMOTE. To understand the algorithm, it is crucial to understand the DBSCAN algorithm discussed in [Bunkhumpornpat 2012]. DBSMOTE algorithm relies on a data structure called *Directly density reachable graph*. Let x be a data point and $N_\epsilon(x)$ be a neighbourhood of x with radius ϵ. If the $N_\epsilon(x)$ contains at least k points, where k is some positive integer and $y \in N_\epsilon(x)$, then y is said to be directly density reachable from x, with respect to the chosen values of ϵ and k. For some cluster C in the data, a Directly density reachable graph is an undirected but weighted simple graph constructed such that the nodes are the data points of the cluster, for any pair of directly density reachable nodes, there is an edge in the graph such that the edge weight is the distance between the two nodes. The DBSMOTE algorithm is realised through the following steps:

1. First, the DBSCAN algorithm is used to cluster the minority class into m disjoint clusters $C_1, \ldots, C_m$, depending on some choice of k and ϵ. Note that DBSCAN clustering has a noise filtering mechanism.
2. From each of the selected clusters C_i, $1 \leq i \leq m$, a directly density-reachable graph is constructed depending upon the chosen ϵ and k.
3. The *pseudo centroid* c_i of the cluster C_i is calculated. It is the nearest data point to the centroid of the cluster C_i.

4. For every point p except for c_i in C_i the shortest path between p and c_i is identified using Dijkstra's algorithm.

5. Random edges are chosen from this shortest paths, and the node data points attached to the corresponding edges are chosen to generate synthetic samples using the SMOTE algorithm.

6. The process is then repeated over all clusters to balance the classes.

The CURE clustering algorithm is used by **CURE SMOTE** to identify clusters among the minority class samples [Ma 2017, Guha 2001]. The CURE algorithm uses a hierarchical clustering framework, which is also able to identify outliers in the minority class. The algorithm identifies centre points for each cluster after removing the noisy data points, in the process of clustering. It then generates synthetic data by applying SMOTE on each representative minority data point and the cluster centres [Ma 2017].

Notably, a similar algorithm has been proposed using the Affinity propagation clustering method instead of the K means clustering [Laureano 2019]. The proper choice of K is often considered as a limitation of K-means clustering algorithm. However, Affinity propagation does not require the number of clusters to be specified *a priori*.

Also, there is another algorithm called cluster-SMOTE that is quite similar to the K-means SMOTE algorithm. However, it uses K-means clustering only on the minority class and then oversamples from each cluster using the SMOTE algorithm.

3.3.2 SMOTE with dimension reduction algorithms

I have discussed algorithms that integrate clustering methods with SMOTE. Most of these clustering methods are dependent on their parameter values and might not be robust. Dimension reduction methods are different from clustering methods in the sense that they do not deliberately try to find clusters in the data, but only attempt to reduce the data to lower dimensions. If there is a strong pattern occurring in a subset of the data, usually visualising the dimension-reduced data helps us identify such subsets as clusters occurring in lower dimensions.

Isomap hybrid resampling is an algorithm integrating Isomap with SMOTE [Gu 2009]. Isomap is a dimension reduction method, which differs from its predecessor PCA on a key aspect. PCA uses Euclidean distance as a distance measure between two points, while Isomap uses the approximated geodesic distance as a distance measure between two points. This enables Isomap to represent the nonlinearity in data in lower dimensions with more efficiency. Isomap hybrid resampling uses the Isomap algorithm to embed the original data into a lower dimensional space where it is linearly separable. The SMOTE algorithm is then used to generate oversamples in the reduced data space. The oversampled samples are then under-sampled using Neighbourhood Cleaning Rule (NCR) to generate balanced low-dimensional data sets.

Another instance of integrating dimension reduction techniques with SMOTE is the Minority Oversampling based on Local Densities in Low-Dimensional Space, the **MOT2LD**

algorithm. Notably, this algorithm uses both dimension reduction and clustering approach [Xie 2015]. It uses t-SNE, a popular algorithm used for dimension reduction maintaining the underlying manifold structure in a sense that, in a lower dimension, t-SNE can cluster points, that are close enough to the latent high-dimensional manifold. The MOT2LD algorithm is realised through the following steps:

1. The first step involves dimension reduction of the training data. A high dimensional data point is embedded in some lower dimension. This is done using the t-SNE algorithm. Notably, the t-SNE algorithm was proposed after Isomap and was a significant improvement over the latter.

2. The second step is about finding clusters in the low dimensional embedding of the data. A density based clustering algorithm, DPCluster is employed here. This clustering algorithm determines the number of clusters automatically, unlike K-Means Clustering. The motivation of the clustering step is to generate synthetic samples within the clusters rather than between the clusters.

3. The third step involves outlier detection and filtering. A minority class sample can be treated as an outlier if its local minority density is zero. This means that none of its nearest neighbours are minority class samples.

4. In the fourth step, the minority class samples are assigned some weights, indicative of their importance in context of the oversampling process. The importance of a minority class sample is measured by the product of local majority count and inverse of local minority density.

5. The fifth step is about synthetic sample generation. For this, the importance scores of the minority class data points are first normalised to obtain a probability distribution. Then a minority samples is drawn randomly following this probability distribution. Another random sample is drawn that lies in the same cluster as the previous one. Using these two samples, a new synthetic sample is generated following the SMOTE algorithm.

The Self Organising Map Oversampling **SOMO** algorithm uses a Self Organising Map (SOM) [Yin 2008, Douzas 2017], a dimension reduction approach, to learn the latent distribution of the minority class data points. The imbalance ratio within each cluster is then calculated, followed by identification of clusters with an imbalance ratio of less than 1. For each of these identified clusters a density factor quantifying the density of minority class data points in the respective cluster is calculated and then, for every pair of topologically neighbouring clusters, another density factor, is calculated between the pairs. Both of the density factors are normalised to form a weight distribution. The weight distribution arising from the first distribution governs the amount of intra-cluster synthetic samples to be generated from each detected cluster, while the weight distribution arising from the second distribution governs the amount of inter-cluster synthetic samples to be generated between a pair of topologically close clusters [Douzas 2017]. The synthetic samples are generated using the SMOTE algorithm for both cases.

3.3.3 Oversampling technique integrating multiple approaches

The algorithm MWMOTE has been designed with the objective of improving the selection of relevant minority class samples and using them for oversampling [Barua 2014]. It is a hybrid approach that uses weight distribution to prioritise minority class samples that are beneficial for synthetic data generation as seen in the ADASYN algorithm and also uses a clustering algorithm to cluster the minority class data points. The philosophy behind the strategy is similar to its predecessor Borderline-SMOTE. However, it uses a more intricate mechanism to select the Borderline minority class samples. There are six input parameters for the MWMOTE algorithm C_{maj}, C_{min}, S_{min}, k_1, k_2, k_3. C_{maj} and, C_{min} as usual, denote the majority and minority class, respectively. Through the parameter S_{min} the user can determine the number of synthetic samples to be generated. k_1, k_2 and k_3 are the parameters used to calculate data subsets defined as filtered minority set, borderline majority set, and informative minority set respectively.

The filtered minority set is defined as $C_{\text{fmin}} = C_{\text{min}} - \{x \in C_{\text{min}} : N_{k_1}(x) \cap C_{\text{min}} = \phi\}$, where $N_{k_1}(x)$ denotes the k_1 nearest neighbours of x. The filtered minority set identifies the relatively safe minority class data points. This step is designed to filter out the outliers in the minority class.

The borderline majority set is defined as $C_{\text{bmaj}} = \{y \in C_{\text{maj}} : \exists x \in C_{\text{fmin}}, y \in N_{k_2}^{C_{\text{maj}}}(x)\}$, where $N_{k_2}^{C_{\text{maj}}}(x)$ denotes the k_2 nearest majority class neighbours of x. This step is designed to identify majority class data points that are in close vicinity to some minority class sample that is not an outlier.

The informative minority set is defined $C_{\text{imin}} = \{y \in C_{\text{fmin}} : \exists x \in C_{\text{bmaj}}, y \in N_{k_3}^{C_{\text{min}}}(x)\}$, where $N_{k_3}^{C_{\text{min}}}(x)$ denotes the k_3 nearest minority class neighbours of x. This step is designed to identify minority class data points that are not an outlier, but lie in the close vicinity of some majority class data point.

After the three sets C_{fmin}, C_{bmaj} and C_{imin} are identified, a weight distribution is defined the minority class samples of C_{imin}. The weighing mechanism adopted by MWMOTE is based on three key assumptions.

1. Samples close to the decision boundary contain more information than those far from the boundary.
2. Minority samples in sparse clusters are more important than those in dense clusters.
3. Minority samples near dense majority clusters are more important than those near sparse majority clusters.

A weight distribution D_{imin}^{w} is calculated by taking the borderline majority set C_{bmaj} into consideration. Each data point $x \in C_{\text{bmaj}}$ contributes to the weight of each minority class data point $p \in C_{\text{imin}}$. The weight imposed by $x \in C_{\text{bmaj}}$ on $p \in C_{\text{imin}}$ is denoted by $I_w(x, y)$, named as information weight imposed by x on p. For a minority class data point $p \in C_{\text{imin}}$, the selection weight is defined by the equation

$$D_{\text{imin}}^{w}(p) = \Sigma_{x \in C_{\text{bmaj}}} I_w(x, p) \tag{3.11}$$

After all selection weights are determined, they are normalised to form a probability distribution over the set C_{imin}. The quantity $I_w(x, p)$ for a given $x \in C_{\text{bmaj}}$ and $p \in C_{\text{imin}}$ are calculated depending on two factors called closeness factor and density factor.

The closeness factor C_f of two points $x \in C_{\text{min}}$ and, $p \in C_{\text{min}}$ with respect to a cut-off function f, is assumed to be inversely proportional to the Euclidean distance between points x and p. Let $d_e(x, p)$ denote the Euclidean distance between x and p. Let $d(x, p) = \frac{d_e(x,p)}{|F|}$, where $|F|$ is the dimension of the feature space. The closeness factor is then defined by the equation:

$$C_f(x, p) = \frac{f(d(x, p)^{-1})}{C_f(t)} M \tag{3.12}$$

Where t and M are user defined quantities and f is a cut-off function.

The density factor is designed to support the second assumption 'Minority samples in sparse clusters are more important than those in dense clusters', keeping in mind that the first assumption is not contradicted. For points x and, p the density factor is defined as:

$$D_f(x, p) = \frac{C_f(x, p)}{\Sigma_{q \in C_{\text{min}}} C_f(x, q)} \tag{3.13}$$

This is the normalised version of the closeness factor. The density factor ensures that a large weight is assigned to the members of the sparse clusters.

Finally, depending on the closeness factor from Equation 3.12 and density factor from Equation 3.13, the quantity $I_w(x, p)$ is calculated as the product of $C_f(x, p)$ and $D_f(x, p)$.

After the probability distribution over C_{min} is finalised, MWMOTE proceeds to a clustering step to cluster C_{min} using average linkage agglomerative clustering, a hierarchical clustering process. Agglomerative clustering does not require the number of clusters to be fixed *a priori*.

Finally, the synthetic samples are generated by first randomly choosing a data point $x \in C_{\text{imin}}$ based on the probability distribution defined over C_{imin}. A second sample y is randomly chosen from the cluster which contains the data points x. For the choice of y, a uniform probability distribution is considered over the cluster. Using x and y a synthetic data is generated by the Equation 3.1.

3.4 Comparative studies between oversampling algorithms

A recent study conducted an empirical comparison of 85 such extensions or variants of the SMOTE algorithm proposed until 2018 [Kovács 2019]. This shows that there is still scope for improvement in convex space based oversampling and that the research field is still quite dynamic. The comprehensive study benchmarked these algorithms on over a hundred imbalanced datasets using different classifiers, including Support Vector Machine (SVM), Decision Tree (DT), k-Nearest Neighbours (kNN), and Multi Layered Perceptron (MLP), and investigated the best performing algorithms among the 85 SMOTE extensions. I briefly introduce two algorithms with the best ranking according to Kovács *et al.* : Polynom-fit SMOTE and ProWSyn.

3.4.1 Some state-of-the-art extensions of SMOTE

The **Polynom-fit SMOTE** (pf-SMOTE) algorithm was proposed in 2008 [Gazzah 2008]. This algorithm has different oversampling schemes based on the underlying network topologies of the minority class. The pf-SMOTE algorithm proposes four different network topologies to generate synthetic samples from minority classes, depending on the latent data distribution. These are: star topology, polynomial curve topology, bus topology, and mesh topology. The star topology generates synthetic samples along straight lines joining the mean of the minority class data points and each minority class data point, forming a star-like silhouette for the synthetic data. For the polynomial curve topology, each feature is fit to a polynomial, the synthetic samples are generated feature-wise along the curve of these polynomials. For the bus topology, a path connecting one minority data to its nearest neighbour is formed using straight lines. Synthetic samples are sampled from this path. For the mesh topology, synthetic samples are generated along straight lines connecting each minority data point to the rest of the minority data points. The authors suggest that the star topology has proven to be the most effective [Gazzah 2008].

The **ProWSyn** algorithm partitions the minority classes by their proximity to the majority class. The partitions are treated as clusters. The clusters are weighted as per their proximity to the majority class, such that clusters closer to the majority class have higher weights. The precise method for this is documented in Algorithm 4. The weights decide how many samples are to be generated from each cluster. Synthetic samples are generated following the approach of SMOTE, that is, taking a convex combination of two arbitrary samples in a cluster [Barua 2013].

This chapter reviews the state-of-the-art relevant to synthetic oversampling based on convex space modelling. The next chapter discusses my research work on improving synthetic sample generation through convex space modelling by introducing the LoRAS algorithm.

Chapter 4

Improving convex space modelling with the LoRAS algorithm

This chapter introduces the LoRAS algorithm. The algorithm relies on a more precise modelling of the convex space minority class data compared to its predecessors. Firstly, this chapter provides a geometric interpretation and an analytical explanation behind the rationale of the approach adopted by LoRAS for convex space modelling, followed by a section presenting a detailed description and pseudocode of the algorithm. Then the chapter discusses the integration of the LoRAS algorithm with the state-of-the art dimension reduction algorithm UMAP, to introduce the LoRAS UMAP algorithm. The protocols and datasets used for my benchmarking studies for LoRAS and LoRAS UMAP algorithms are then described. I show, from the results of the benchmarking studies, some unique aspects of LoRAS in terms of improving classifier performance and present the statistical significance of the results to analyse the reliability of the results.

4.1 Modelling the convex space of minority class

4.1.1 Geometric interpretation of convex space modelling

This section describes the strategy of LoRAS to approximate the data manifold, given a dataset. A typical dataset for a supervised ML problem consists of a set of *features* $F = \{f_1, f_2, \dots\}$, that are used to characterise patterns in the data, and a set of *labels* or ground truth. Ideally, the number of instances or samples should be significantly greater than the number of features. In order to maintain the mathematical rigour of LoRAS consider the following definition for a *small dataset*.

Definition 11. *Consider a class or the whole dataset with n samples and $|F|$ features. If* $\log_{10}(\frac{n}{|F|}) < 1$, *then the dataset is called, a small dataset.*

The LoRAS algorithm is designed to learn from a dataset by approximating the underlying data manifold. Assuming that F is the best possible set of features to represent the data and all features are equally important, one can think of a data oversampling model to be a function $g : \prod_{i=1}^{l} R^{|F|} \rightarrow R^{|F|}$, that is, g uses l parent data points (each with $|F|$ features) to produce an oversampled data point in $R^{|F|}$.

Definition 12. *Define a random affine combination of some arbitrary vectors as the affine linear combination of those vectors, such that the coefficients of the linear combination are chosen randomly. Formally, a vector* v, $v = \alpha_1 u_1 + \cdots + \alpha_n u_m$, *is a random affine combination of vectors* $u_1, \ldots, u_m$, $(u_j \in R^{|F|})$ *if* $\alpha_1 + \cdots + \alpha_m = 1$, $\alpha_j \in R^+$ *and* $\alpha_1, \ldots, \alpha_m$ *are the coefficients of the affine combination chosen randomly from a Dirichlet distribution.*

The simplest way of augmenting a data point would be to take the average (or random affine combination with positive coefficients as defined in Definition 12) of two data points as an augmented data point. But, when there are $|F|$ features, one can assume that the hypothetical manifold on which the data lies is $|F|$-dimensional. A $|F|$-dimensional manifold can be locally approximated by a collection of $(|F|–1)$-dimensional planes.

4.1.2 Analytical explanation for convex space modelling

I constructed a mathematical framework to prove that LoRAS is a more effective oversampling technique, since it provides a better estimate for the mean of the underlying local data distribution of the minority class data space. Let $X = (X_1, \ldots, X_{|F|}) \in C_{\min}$ be an arbitrary minority class sample. Let N_k^X be the set of the k-nearest neighbours of X, which will consider the neighbourhood of X. Both SMOTE and LoRAS focus on generating augmented samples within the neighbourhood N_k^X at a time. Assume that a random variable $X \in N_k^X$ follows a shifted t-distribution with k degrees of freedom, location parameter μ, and scaling parameter σ. Note that here σ is not referring to the standard deviation but sets the overall scaling of the distribution [Jackman 2009], which can be chosen to be the sample variance in the neighbourhood of X. A shifted t-distribution is used to estimate population parameters, if there are less number of samples (usually, ≤ 30) and/or the population variance is unknown. Since in SMOTE or LoRAS generates samples from a small neighbourhood, one can argue in favour of the assumption that locally, a minority class sample X as a random variable, follows a t-distribution. Following [Blagus 2013], one can assume that if $X, X' \in N_k^X$ then X and X' are independent. For $X, X' \in N_k^X$, also assume:

$$\begin{aligned} \mathbf{E}[X] &= \mathbf{E}[X'] \\ &= \mu = (\mu_1, \ldots, \mu_{|F|}) \\ \mathrm{Var}[X] &= \mathrm{Var}[X'] \\ &= \sigma^2\Big(\frac{k}{k-2}\Big) \\ &= \sigma'^2 = (\sigma_1'^2, \ldots, \sigma_{|F|}'^2) \end{aligned} \tag{4.1}$$

where, $\mathbf{E}[X]$ and $\mathrm{Var}[X]$ denote the expectation and variance of the random variable X respectively. However, the mean has to be estimated by an estimator statistic (i.e., a function of the samples). Both SMOTE and LoRAS can be considered as an estimator statistic for the mean of the t-distribution that $X \in C_{\min}$ follows locally.

Theorem 1. *Both SMOTE and LoRAS are unbiased estimators of the mean* μ *of the t-distribution that* X *follows locally. However, the variance of the LoRAS estimator is less than the variance of SMOTE given that* $|F| > 2$.

Proof. A shadowsample S is a random variable $S = X + B$ where $X \in N_k^X$, the neighbourhood of some arbitrary $X \in C_{\min}$ and B follows $\mathcal{N}(0, \sigma_B)$.

$$\begin{aligned} \mathbf{E}[S] &= \mathbf{E}[X] + \mathbf{E}[B] \\ &= \mu \\ \text{Var}[S] &= \text{Var}[X] + \text{Var}[B] \\ &= \sigma'^2 + \sigma_B^2 \end{aligned} \tag{4.2}$$

assuming S and B are independent. Now, a LoRAS sample $L = \alpha_1 S^1 + \cdots + \alpha_{|F|} S^{|F|}$, where $S^1, \ldots, S^{|F|}$ are shadowsamples generated from the elements of the neighbourhood of X, N_k^X, such that $\alpha_1 + \cdots + \alpha_{|F|} = 1$. The affine combination coefficients $\alpha_1, \ldots, \alpha_{|F|}$ follow a Dirichlet distribution, with all concentration parameters assuming equal values of 1 (assuming all features to be equally important). For arbitrary $i, j \in \{1, \ldots, |F|\}$,

$$\begin{aligned} \mathbf{E}[\alpha_i] &= \frac{1}{|F|} \\ \text{Var}[\alpha_i] &= \frac{|F| - 1}{|F|^2(|F| + 1)} \\ \text{Cov}(\alpha_i, \alpha_j) &= \frac{-1}{|F|^2(|F| + 1)} \end{aligned}$$

where $\text{Cov}(A, B)$ denotes the covariance of two random variables A and B. Assuming α and S to be independent,

$$\mathbf{E}[L] = \mathbf{E}[\alpha_1]\mathbf{E}[S^1] + \cdots + \mathbf{E}[\alpha_{|F|}]\mathbf{E}[S^{|F|}] = \mu \tag{4.3}$$

Thus L is an unbiased estimator of μ. For $j, k, l \in \{1, \ldots, |F|\}$,

$$\begin{aligned} \text{Cov}[\alpha_k S_j^k, \alpha_l S_j^l] &= \mathbf{E}[\alpha_k S_j^k \alpha_l S_j^l] - \mathbf{E}[\alpha_k S_j^k]\mathbf{E}[\alpha_l S_j^l] \\ &= \mathbf{E}[\alpha_k \alpha_l]\mu_j^2 - \frac{\mu_j^2}{|F|^2} \\ &= \Big[\text{Cov}(\alpha_k, \alpha_l) + \frac{1}{|F|^2}\Big]\mu_j^2 - \frac{\mu_j^2}{|F|^2} = \mu_j^2 \text{Cov}(\alpha_k, \alpha_l) \end{aligned} \tag{4.4}$$

since $\alpha_k \alpha_l$ is independent of $S_j^k S_j^l$. For an arbitrary j, j-th component of a LoRAS sample L_j

$$\begin{aligned} \text{Var}(L_j) &= \text{Var}(\alpha_1 S_j^1 + \cdots + \alpha_{|F|} S_j^{|F|}) \\ &= \text{Var}(\alpha_1 S_j^1) + \cdots + \text{Var}(\alpha_{|F|} S_j^{|F|}) + \Sigma_{k=1}^{|F|} \Sigma_{l=1, l \neq k}^{|F|} \text{Cov}(\alpha_k S_j^k, \alpha_l S_j^l) \\ &= \frac{\mu_j^2(|F| - 1) + 2(\sigma_j'^2 + \sigma_{Bj}^2)|F|}{|F|(|F| + 1)} - \frac{\mu_j^2(|F| - 1)}{|F|(|F| + 1)} \\ &= \frac{2(\sigma_j'^2 + \sigma_{Bj}^2)}{(|F| + 1)} \end{aligned} \tag{4.5}$$

For LoRAS, a convex combination of $|F|$ shadowsamples is considered as a synthetic samples and SMOTE considers a convex combination of two minority class samples.

Note that since a SMOTE generated oversample can be interpreted as a random convex combination of two minority class samples, one can consider $|F| = 2$ for SMOTE, independent of the number of features. Moreover, from Equation 4.3, this implies that SMOTE is an unbiased estimator of the mean of the local data distribution. Thus, the variance of a SMOTE generated sample as an estimator of μ would be $\frac{2\sigma'^2}{3}$ (since $B = 0$ for SMOTE). However, for LoRAS as an estimator of μ, when $|F| > 2$, the variance would be less than that of SMOTE. □

This implies that, locally, LoRAS can estimate the mean of the underlying t-distribution better than SMOTE.

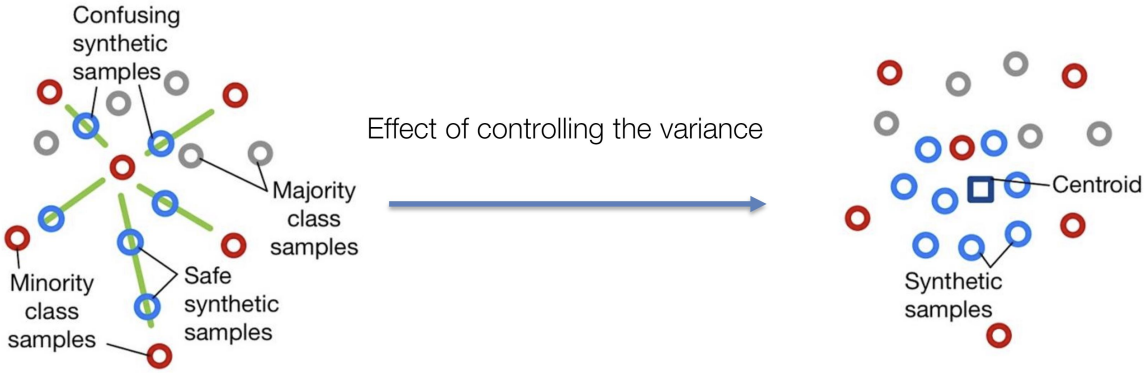


FIGURE 4.1: Figure showing the idea of LoRAS to control the variance of the synthetic samples generated from the minority class. Compared to SMOTE, LoRAS can generate low-variance synthetic samples which can be intuitive interpreted as synthetic samples generated closer to the average point or centroid in a minority class data neighbourhood. LoRAS thus prevent classifiers from confusing them to majority class samples.

4.2 LoRAS algorithm

Given $|F|$ sample points, it is possible to exactly derive the equation of a unique $(|F|-1)$-dimensional plane containing these $|F|$ sample points. Note that, a small neighbourhood of a dataset can itself be considered as a small dataset. A small neighbourhood of k points around a data point in a dataset, given sufficiently small k, satisfies Definition 11, that is k and $|F|$ satisfies, $\log_{10}(\frac{k}{|F|}) < 1$. Thus, considering k to be sufficiently small, it can be assumed that this small neighbourhood is a small dataset. To enrich this small dataset, LoRAS creates *shadow data points* or *shadowsamples* from the k parent data points in the minority class data point neighbourhood. Each shadow data point is generated by adding noise from a normal distribution, $\mathcal{N}(0, h(\sigma_f))$ for all features $f \in F$, where $h(\sigma_f)$ is some function of the sample variance σ_f for the feature f. For each of the k data points, LoRAS can generate m shadow data points such that, $k \times m \gg |F|$. Now it is possible for us to choose $|F|$ shadow data points from the $k \times m$ shadow data points even if $k < |F|$. LoRAS chooses $|F|$ shadow data points as follows: it first chooses a random parent data point p and then restrict the domain of choice to the shadowsamples generated by the parent data points in N_k^p.

For high-dimensional datasets, choosing k-nearest neighbours of a data point using simple Euclidean, Manhattan or general Minkowski distance measures can be misleading in terms of approximating the latent data manifold. To avoid this, LoRAS adopts a manifold learning-based strategy. Before choosing the k-nearest neighbours of a data point, LoRAS performs a dimension reduction on the data points of the minority class using the well-known dimension reduction and manifold learning technique t-SNE [van der Maaten 2008]. Once a two-dimensional t-embedding of the minority class data is obtained, LoRAS chooses the k-nearest neighbours of a particular data point consistent with its k-nearest neighbours (measured as per usual distance metrics) in the 2-dimensional t-SNE embedding of the minority class.

Once the neighbourhoods are decided upon and the shadowsamples are generated, LoRAS takes a random affine combination with positive coefficients (Convex combination) of the $|F|$ chosen shadowsamples to create one augmented Localized Random Affine Shadowsample or a LoRAS sample as defined in Definition 12. Considering the arbitrary low variance that LoRAS can choose for the Normal distribution from which it draws the shadowsamples, one can assume that the shadowsamples lie in the latent data manifold itself. It is a practical assumption, considering the stochastic factors leading to small measurement errors. Now, there exists a unique $(|F|-1)$-dimensional plane, that contains the $|F|$ shadowsamples, which is assumed to be an approximation of the latent data manifold in that small neighbourhood. Thus, a LoRAS sample is an artificially generated sample drawn from a $(|F|-1)$-dimensional plane, which locally approximates the underlying hypothetical $|F|$-dimensional data manifold. It is worth mentioning here, that the effective number of features in a dataset is often less than $|F|$. In high dimensional data, there are often correlated features or features with low variance. Thus, for practical use of LoRAS, one might consider generating convex combinations of the effective number of features, which might be less than $|F|$.

4.3 Benchmarking studies for LoRAS algorithm

For testing the potential of LoRAS as an oversampling approach, I designed benchmarking experiments on a total of 14 datasets which are either highly imbalanced, high dimensional or with a few data points. With this number of diverse case studies, it is possible to have a comprehensive idea of the advantages of LoRAS over the other oversampling algorithms of interest.

4.3.1 Datasets used

This section provides a brief description of the datasets and sources that have been used for the study.

Scikit-learn imbalanced benchmark datasets: The `imblearn.datasets` package is complementing the `sklearn.datasets` package. It provides 27 preprocessed datasets, which are imbalanced. The datasets span a large range of real-world problems from several fields such as business, computer science, biology, medicine, and technology. This collection

Algorithm 1 Localized Random Affine Shadowsample (LoRAS) Oversampling

Inputs:

`C_maj`:	Majority class parent data points
`C_min`:	Minority class parent data points

Parameters:

`k`:	Number of nearest neighbours to be considered per parent data point (default value : 30 if $\|C_{min}\| >= 100$, 5 otherwise)
$\|S_p\|$:	Number of generated shadowsamples per parent data point $(\text{default value : max}\,(\text{ceil}(\frac{2\|F\|}{k}), 40))$
L_σ:	List of standard deviations for normal distributions for adding noise to each feature (default value : $[0.005, \ldots, 0.005]$)
N_{aff}:	Number of shadow points to be chosen for a random affine combination (default value : $\|F\|$)
N_{gen}:	Number of generated LoRAS points for each nearest neighbours group $(\text{default value : } \frac{\|C_{maj}\| - \|C_{min}\|}{\|C_{min}\|})$
`embedding`:	Type of Embedding used to choose minority class neighbourhood (regular or t-embedding) (default value : 'regular')
`perplexity`:	Perplexity of t-embedding (applicable only if `embedding`='t-embedding') (default value : 30)

Constraint:

$N_{aff} < k * |S_p|$

Initialize `loras_set` as an empty list
For each minority class parent data point `p` in `C_min` **do**
  `neighbourhood` ← calculate `k`-nearest neighbours of `p`, as per selected `Embedding` parameter and append `p`
  Initialize `neighbourhood_shadow_sample` as an empty list
  For each parent data point `q` in `neighbourhood` **do**
    `shadow_points` ← draw $|S_p|$ shadowsamples for `q` drawing noises from normal distributions with corresponding standard deviations L_σ containing elements for every feature
    Append `shadow_points` to `neighbourhood_shadow_sample`
  endfor
  Repeat
    `selected_points` ← select N_{aff} random shadow points from `neighbourhood_shadow_sample`
    `affine_weights` ← create and normalize random weights for `selected_points`
    `generated_LoRAS_sample_point` ← `selected_points` · `affine_weights`
    Append `generated_LoRAS_sample_point` to `loras_set`
  Until N_{gen} resulting points are created;
endfor
Return resulting set of generated LoRAS data points as `loras_set`

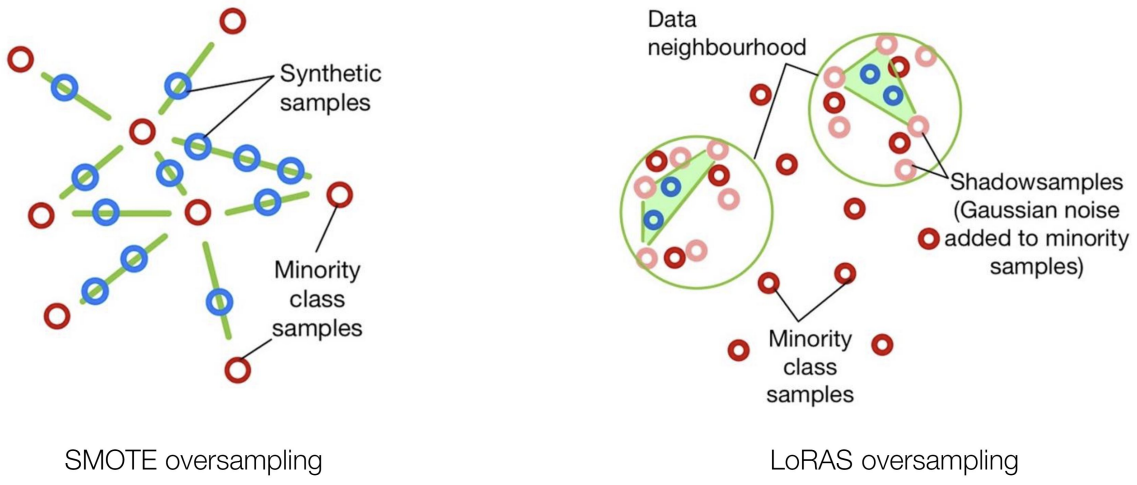


FIGURE 4.2: Figure showing the philosophies of SMOTE and LoRAS oversampling algorithms. While smote generates synthetic samples from convex combination of two close minority class samples, LoRAS generates synthetic samples from random convex combination of multiple shadowsamples in a minority class data neighbourhood.

of datasets was proposed in the `imblearn.datasets` Python library by [Lemaître 2017] and benchmarked by [Ding 2011]. Many of these datasets have been used in various research articles on oversampling approaches [Ding 2011, Sáez 2016]. A statistically reliable benchmarking analysis of all 27 datasets in a stratified cross-validation framework involves

a lot of computational effort. 11 datasets are thus chosen out of these two, depending on two criteria:

- **Highly imbalanced:** Datasets with imbalance ratio more than 25:1. This category includes abalone_19, letter_image, mammography, ozone_level, webpage, wine_quality, yeast_me2 datasets.
- **High dimensional:** Datasets with more than 100 features. This category includes arrhythmia, isolet, scene, webpage and yeast_ml8.

Note that the `webpage` dataset is common in both the criteria, giving us a total of 11 datasets. These two categories are chosen because they are of special interest in research related to imbalanced datasets and have received extensive attention in this research area [Anand 2010, Hooda 2018, Jing 2021, Blagus 2013].

Credit card fraud detection dataset: The description of this dataset from the website is: `https://www.kaggle.com/mlg-ulb/creditcardfraud`. "The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where there are 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.00172 percent of all transactions. The dataset contains only numerical input variables, which are the result of a PCA transformation. Feature variables $f_1, \ldots, f_{28}$ are the principal components obtained with PCA, the only features that have not been transformed with PCA are the 'Time' and 'Amount'. The feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' consists of the transaction amount. The labels are encoded in the 'Class' variable, which is the response variable and takes value 1 in case of fraud and 0 otherwise" [Dal Pozzolo 2018].

Small datasets: It was also an interesting venture to check the performance of LoRAS on small datasets. Two such datasets were obtained: ar1, ar3. Both of these datasets have very few data points and less than 10 points in the minority class.

Thus, in total, the LoRAS oversampling algorithm was benchmarked against the existing algorithms on a total of 14 datasets. Please note the relevant statistics on these datasets in Table 4.1.

4.3.2 Study protocols

For every dataset analysed, a consistent workflow has been used. Given a dataset, for every machine learning model, the model performances are judged based on a 5×10-fold stratified cross-validation framework. However, for the two small datasets ar1 and ar3 use, a 5×3-fold stratified cross validation framework is employed, since there are less than 10 samples in the minority class. First, the dataset is randomly scuffled. A given dataset is first split into 10 folds, each one distinct from the other, maintaining the imbalance ratio for every fold. Then machine learning models are trained on the dataset without any oversampling with 10-fold cross validation. This means that they are trained and tested 10 times, each time considering a fold as a test fold and the rest 9 folds as training folds. However, while training

TABLE 4.1: Table showing some statistics for the datasets studied in for the benchmarking of LoRAS. For each dataset, the feature of the dataset that led us to its choice for this study is marked in bold.

Dataset	Imbalance ratio	Number of samples	Number of features
abalone_19	**130:1**	4177	10
arrythmia	17:1	452	**278**
isolet	12:1	7797	**617**
letter-img	**26:1**	20000	16
mammography	**42:1**	11183	6
scene	13:1	2407	**294**
ozone_level	**34:1**	2536	72
webpage	**33:1**	34780	**300**
wine-quality	**26:1**	4898	11
yeast-me2	**28:1**	1484	8
yeast-ml8	13:1	2417	**103**
credit fraud	**577:1**	284807	28
ar1	12.44:1	**121**	30
ar3	6.8:1	**63**	30

the ML models on oversampled data, the oversampling was done only on the training folds and the test folds were left as they are for each training session. For each dataset, the whole process is repeated five times to avoid the stochastic effects as much as possible.

For all the oversampling algorithms, for a given dataset, the same neighbourhood size for every oversampling model is chosen. If there were less than 100 data points in the minority class, the neighbourhood size was chosen to be 5. Otherwise, the neighbourhood size is considered to be 30. Given numerous datasets analysed, for every dataset, this was not customised, and the above-mentioned general rule is maintained throughout. For LoRAS oversampling, however, a preliminary study was performed to find out the customised parameter values for every dataset, since the LoRAS algorithm is highly parametrized in nature. Several combinations of parameters $\mathtt{N_{aff}}$, `embedding` and `perplexity` were tried employing random grid search. For the initial study involving the parameter optimisation of LoRAS, given a dataset, a simple train-test split of the dataset (1:1 train-test split ratio) was done, and then LoRAS was applied with parameter grids on the training data to oversample and test the classifier performances on the test data. The training set is kept relatively small, so that the classifier does not gain much experience on the data while parameter estimation and gets prone to overfitting. This study was kept completely independent of the main cross-validation based results so that the samples from the test sets of cross validation have minimum effect on parameter tuning. For parameter, $\mathtt{N_{aff}}$ the grid interval is $[2, |F|]$, $|F|$ being the number of features. Five numbers were chosen while forming a search grid from this interval. Three of them are randomly chosen and the numbers 2 and $|F|$ are always included in this set of 5 numbers. For parameter, `embedding` the grid values are the two possible entries that the parameter adopts. For the `perplexity` parameter, grid values $[0.01, 0.1, 1, 10, 30, 100]$ were used.

For all the algorithms including LoRAS, for a given dataset, the neighbourhood size for every oversampling model is fixed. For every oversampling model considered, the neighbourhood size for the oversampling model is the parameter that the model is highly sensitive to, since it contributes the most in determining the distribution of the oversampled minority class. For LoRAS, there are three (out of seven parameters in total) parameters designed to better model/approximate the minority class data manifold (for example: the ones involving the t-SNE on the minority class), which are tuned to show the applicability

of manifold approximation to improve convex combination-based oversampling. However, as suggested, all parameters related to the original distribution of the minority class, for all oversampling models are fixed for all comparisons.

However, considering the philosophy of LoRAS and the comparatively large number of parameters it uses, the other parameters for LoRAS are tuned, since the other parameters are the key to a proper approximation or modelling of the minority class data manifold, which is the key factor behind the success of LoRAS.

For LoRAS oversampling, every dataset a unique value for $\mathtt{N}_{\mathtt{aff}}$ is used, as presented in Table 4.2. For individual ML models, different settings for the LoRAS parameters `embedding` and `perplexity` are implemented (See supplementary materials in [Bej 2021]). To ensure fairness of comparison, the oversampling was done such that the total number of augmented samples generated from the minority class was as close as possible to the number of samples in the majority class as allowed by each oversampling algorithm. Speaking of other parameters of the LoRAS algorithm, for $\mathtt{L}_{\sigma}$, w list consisting of a constant value of .005 is considered for each dataset and for the parameter $\mathtt{N}_{\mathtt{gen}}$ the value $\frac{|C_{\text{maj}}|-|C_{\text{min}}|}{|C_{\text{min}}|}$ is chosen. A detailed list of parameter settings used for the oversampling algorithms is presented in Table 4.2

TABLE 4.2: This table shows the details of the parameter settings for the oversampling algorithms used by us for the experiment. The second column is the size of the oversampling neighbourhood, and the same size is chosen for all the oversampling models for each dataset in the analysis. The last three columns are specific to LoRAS parameters.

Dataset	Minority samples	Oversampling nbd	LoRAS $\mathtt{N}_{\mathtt{aff}}$
abalone19	32	5	10
arrythmia	25	5	100
isolet	600	30	179
letter-img	734	30	16
mammography	260	30	6
scene	177	30	2
ozone_level	73	5	10
webpage	981	30	94
wine-quality	183	30	2
yeast-me2	51	5	2
yeast-ml8	178	30	3
credit fraud	492	30	30
ar1	9	3	30
ar3	8	3	10

To choose ML models for the study, first a pilot study was conducted with ML classifiers such as k-nearest neighbours (kNN), Support Vector Machine (SVM) (linear kernel), Logistic regression (LR), Random forest (RF), and Adaboost (AB). As inferred in [Blagus 2013], kNN was found to be quite effective for the datasets used. LR and SVM performed better compared to RF and ab in most cases. Thus, kNN, SVM, and LR were considered for the final studies. The lbfgs solver was used for the LR model and a linear kernel for the SVM model. For the kNN model, 10 nearest neighbours were chosen as parameter settings if there are less than 100 samples in the minority class and 30 nearest neighbours otherwise. For 'arrhythmia', 'abalone-19', 'ar1' and 'ar3' however only 5 nearest neighbours for the kNN model are considered since it has only 25, 32, 9 and 8 minority class samples respectively.

This parameter is chosen to be consistent with the neighbourhood size of all oversampling models, since the neighbourhood size directly influences the distribution of the training data and hence the model performance.

In the analysis, special notice of the credit card fraud detection dataset is taken. This dataset is not included in the `imblearn.datasets` Python library. However, the main reason why a special attention was bestowed upon this dataset is that, it is by far the most imbalanced popular publicly available dataset. The extreme imbalance ratio of 577:1 is incomparable to any of the datasets in `imblearn.datasets`. Also, this dataset has received special attention of researchers attempting to use ML in Credit fraud detection [Varmedja 2019]. LR and RF are known to have good prediction accuracies for the dataset. Thus, these two ML models are considered for the credit fraud dataset. [Varmedja 2019] has also not provided cross-validated analysis of their models, while the models have been trained and tested with the usual 10-fold cross-validation framework as discussed before. In addition, for two small datasets with a critically small minority class, only kNN and LR classifiers were used, with parameter settings as specified before. The reason is, for all 12 other datasets, SVM did not stand out to be the best performer in terms of F1-Score in any of them. As performance measures, F1-Score and balanced accuracy have been used for classifier evaluation in this study. For a given classifier, F1-Score quantifies the classification performance of the minority class, whereas balanced accuracy quantifies the overall classification performance.

For computational coding, I used the `scikit-learn (V 0.21.2)`, `numpy (V 1.16.4)`, `pandas (V 0.24.2)`, and `matplotlib (V 3.1.0)` libraries in `Python (V 3.7.4)`.

4.4 Improving classifier performance using LoRAS

From the experiments, one can observe an interesting behaviour of oversampling models in terms of their average F1-Score and balanced accuracy. Once experiment results are presented, I will discuss why considering both F1-Score and balanced accuracy can give us a clearer idea about model performance.

Selected model performances for all datasets: Detailed results of the experiments for all machine learning models are provided as supplementary material in Bej *et al*. To be precise, for every combination of datasets, ML models and oversampling strategies, the mean and variance of the 10-fold cross validation process over 5 repetitions are presented. For judging the performance of the oversampling models the following scheme is followed:

- First, for a given dataset, the ML model trained on that dataset that provides the highest average F1-Score over all the oversampling models and training without oversampling is chosen. The F1-Score reflects the balance between precision and recall and considered as a reliable metric for imbalanced classification tasks.

- Both the balanced accuracy and F1- score of the chosen model are considered as an evaluation of how well the oversampling model performs on the considered dataset. Following this evaluation scheme, the results are presented in Table 4.3.

TABLE 4.3: Table showing balanced accuracy/F1-Score for several oversampling strategies (Baseline, SMOTE, SVM-SMOTE, Borderline1 SMOTE, Borderline2 SMOTE, ADASYN and LoRAS column-wise respectively) for all 14 datasets of interest for ML learning models producing the best average F1 score over all oversampling strategies and baseline training for respective datasets.

Dataset	ML	Baseline	SMOTE	Bl-1	Bl-2	SVM	ADASYN	LoRAS
abalone19	kNN	.534/.000	.644/.054	.552/.044	.552/.044	.556/.045	.571/.055	**.675**/**.059**
arrythmia	LR	.679/.37	.666/.345	.672/.352	**.709**/.307	.679/.350	.667/.362	.694/**.380**
isolet	LR	.900/**.826**	.898/.806	.899/.802	.906/.693	**.911**/.799	.898/.806	.904/.809
letter-img	kNN	.927/**.915**	.988/.781	.984/.768	.977/.687	.986/.724	.985/.732	**.989**/.833
mammography	kNN	.703/**.549**	**.911**/.413	.909/.414	.899/.326	.909/.467	.905/.353	.896/.511
scene	LR	.551/.168	.616/.222	.619/.230	**.620**/.223	.616/**.235**	**.620**/.224	.616/.226
ozone_level	LR	.517/.062	.800/.190	.777/.212	.781/.183	.738/**.215**	.803/.192	**.809**/.207
webpage	kNN	.805/**.711**	.906/.267	.901/.274	.903/.287	.904/.267	.903/.264	**.923**/.613
wine-quality	LR	.517/.067	.718/.179	.715/.182	.711/.171	.712/**.216**	.721/.180	**.734**/.197
yeast-ml8	kNN	.500/.000	.558/.152	.561/.153	.563/.153	**.572**/**.158**	.558/.151	.559/.152
yeast-me2	kNN	.523/.074	.834/.331	.797/.373	.79/.304	.785/**.388**	.825/.315	**.842**/.354
credit fraud	RF	.669/.775	.922/.359	.919/.645	.919/.556	.913/.741	**.923**/.350	.904/**.820**
ar1	kNN	.340/.071	.561/.306	.549/.298	**.594**/.338	.550/.324	.583/.320	.563/**.349**
ar3	RF	.634/.259	.810/.531	.809/**.584**	.819/.582	.755/.479	781/.457	**.823**/.563
Average	-	.636/.338	.775/.352	.764/.380	.771/.346	.759/.386	.777/.340	**.783**/**.433**
Average rank	-	6.53/4.64	3.57/4.75	4.35/3.46	3.39/5.10	4.07/3.17	3.5/4.71	**2.57**/**2.14**

Calculating average performances over all datasets, LoRAS has the best balanced accuracy and F1-Score. As expected, SMOTE improved balanced accuracy compared to model training without any oversampling. Surprisingly, it lags behind in F1-Score, for quite a few datasets with high baseline F1-Score such as letter_image, isolet, mammography, webpage and credit fraud. Interestingly, the oversampling approaches SVM-SMOTE and Borderline1 SMOTE also improved the average F1-Score compared to SMOTE, but compromised for a lower balanced accuracy. On the other hand, applying ADASYN increased the balanced accuracy compared to SMOTE, but again compromises on the F1-Score. In contrast, the LoRAS approach produces the best balanced accuracy on average by maintaining the highest average F1-Score among all oversampling techniques. Even considering stochastic factors, LoRAS can improve both the balanced accuracy and F1-Score of ML models significantly compared to SMOTE, which makes it unique.

Datasets with high imbalance ratio: To verify the performance of LoRAS on highly imbalanced datasets, the average of the selected model performances for the datasets with the highest imbalance ratios (among the ones tested) is presented in Table 4.4.

TABLE 4.4: Table showing the average balanced accuracy/F1-Score of the selected models for datasets with the highest imbalance ratios and high-dimensional datasets separately

Average	Baseline	SMOTE	Bl-1	Bl-2	SVM	ADASYN	LoRAS
Highly imbalanced datasets	.662/.381	.840/.321	.819/.364	.817/.319	.814/.382	.841/.305	**.846**/**.449**
High dimensional datasets	.687/.415	.728/.358	.730/.362	**.740**/.332	.736/.361	.729/.361	.739/**.436**

From the results, note that LoRAS oversampling can significantly improve model performance for highly imbalanced datasets. LoRAS provides the highest F1-Score and balanced accuracy among all oversampling models. The results here show similar properties for SMOTE, Borderline-1 SMOTE, SVM SMOTE, ADASYN and LoRAS as discussed before. Note that, for the credit fraud dataset, which is the most imbalanced

among all, LoRAS has significant success over the other oversampling models in terms of balanced accuracy. For the webpage dataset as well, it improves the balanced accuracy significantly, compromising minimally on the baseline F1-Score. The same trend follows for the letter_image dataset. Notably, these three datasets have the highest number of overall samples as well, implying that with more data, LoRAS can significantly outperform compared convex combination-based oversampling models.

High dimensional datasets: It is also of interest to us to check how LoRAS performs on high dimensional datasets. Therefore, five datasets with the highest number of features are selected among the tested datasets and the performances of the selected ML methods are presented in Table 4.4. From the results for high dimensional datasets, observe that LoRAS produces the best F1-Score and second best balanced accuracy on average among all oversampling models as Borderline-2 SMOTE beats LoRAS marginally. SMOTE improves both balanced accuracy with respect to the baseline score here. Borderline-1 SMOTE and SVM SMOTE further increased SMOTE's performance both in terms of F1-Score and balanced accuracy. Borderline-2 SMOTE, although improves the balanced accuracy of SMOTE, compromises on the F1-Score. Note that, even excluding the webpage dataset, where LoRAS has an overwhelming success, LoRAS still has the best average F1-Score and the third highest balanced accuracy marginally behind SVM-SMOTE and Borederline-2 SMOTE. One can thus conclude, that for high dimensional datasets LoRAS can outperform the compared oversampling models in terms of F1-Score, while compromising marginally for balanced accuracy.

Small datasets: For the two small datasets (with less than 10 samples in the minority class) explored, one can observe that LoRAS performs reasonably well. For 'ar1', LoRAS produces the best F1-Score and the third best balanced accuracy. For the 'ar2' dataset, LoRAS produces the best balanced accuracy and the third best F1-Score. Note that LoRAS performs quite well for the 'abalone' and 'arrhythmia' datasets, which also have a few data points in the minority class.

The Wilcoxon's signed rank test is implemented to compare LoRAS with other convex-combination based oversampling algorithms, in terms of both the performance measures used: F1-Score and balanced accuracy.

TABLE 4.5: Table showing p-values for comparison of LoRAS against the other oversampling algorithms, in terms of both the performance measures used: F1-Score and balanced accuracy.

Measure	Baseline	SMOTE	Bl-1	Bl-2	SVM	ADASYN
F1-Score	0.0303	0.0009	0.0479	.0035	0.0479	0.0009
balanced accuracy	0.0009	0.0354	0.0258	0.5095	0.0382	0.1670

From Table 4.5, observe that the p-values for all paired tests are less than 0.05 for the F1-Score, and therefore, the H_0 is rejected for all the paired tests in case of the F1-Score. Thus, the F1-Scores LoRAS produce have a big enough difference compared to the other compared algorithms, to be statistically significant. For balanced accuracy, the algorithms Borderline-2 SMOTE and ADASYN do not show significant statistical difference to LoRAS. However, since F1-Score is a more reliable and widely used metric for imbalanced datasets,

one can conclude that overall results generated by LoRAS are significantly different from the compared oversampling algorithms.

Recall that a higher value W_+ for LoRAS indicates towards a superior performance of LoRAS and the value of R indicates towards how superior(with a higher W_+)/ inferior(with a higher W_-) the performance of LoRAS is, compared to the other oversampling model for the tested datasets. [Tarawneh 2020] have considered ranges of $R \leq 0.1$, $0.1 < R \leq 0.5$ and $R > 0.5$ to be indicators for small, medium and high degree of change (improvement or deterioration) in the predictive performance respectively.

TABLE 4.6: Table showing $W_+/W_-/R$ for comparison of LoRAS against the other oversampling algorithms, in terms of both the performance measures used: F1-Score and balanced accuracy.

Measure	Baseline	SMOTE	Bl-1	Bl-2	SVM	ADASYN
F1-Score	95/10/.713	105/0/.880	90/15/.629	102/3/.830	80/15/.629	105/0/.880
balanced accuracy	105/0/.880	102/3/.830	95/10/.715	69/36/.286	95/10/.722	95/10/.837

From Table 4.6, note that, LoRAS has a higher W_+ value for both F1 Score and balanced accuracy in comparison to each of the other convex combination-based oversampling methods in consideration. Moreover, for the F1 Score measure, the R value is also more than 0.5, indicating a high degree of improvement in F1-Score for LoRAS, over the considered oversampling models. Similarly, for balanced accuracy, a high degree of improvement for LoRAS is observed over all considered oversampling models except the Borderline-2 SMOTE, for which there is a medium degree of improvement. Overall, it is safe to conclude that LoRAS provides a significant improvement in performance over the compared convex combination-based oversampling methods.

4.5 Significance of the LoRAS algorithm

To visualise the key aspects of LoRAS oversampling, the PCA plots for oversampled data from the ozone_level dataset are generated for several studied oversampling methods, in Figure 4.3. From Figure 4.3 one can observe that SMOTE and ADASYN oversamples highly on the neighbourhood of the outliers, depicted by a blue box in each subplot. While this is somewhat controlled in Borderline1-SMOTE and SVM SMOTE, they still generate some synthetic samples in this neighbourhood. LoRAS on the other hand, refrains to do so, leveraging on its strategy to produce a better estimate for the local mean of the underlying local data distribution. This enables LoRAS to ignore the outliers and to oversample more uniformly, resulting in a better approximation of the data manifold. Note that the average F1-Scores of the oversampling models as presented in Table 4.3 has a direct correlation with how the oversampling strategy oversamples in this neighbourhood. SMOTE and ADASYN generate the lowest F1-Scores and show a tendency of oversampling excessively in this neighbourhood. Borderline-SMOTE and SVM improve the F1-Score compared to SMOTE and ADASYN, again, consistent to their behaviour of oversampling lesser in this neighbourhood. LoRAS, has the highest average F1-Score and oversampling very sparsely in this neighbourhood.

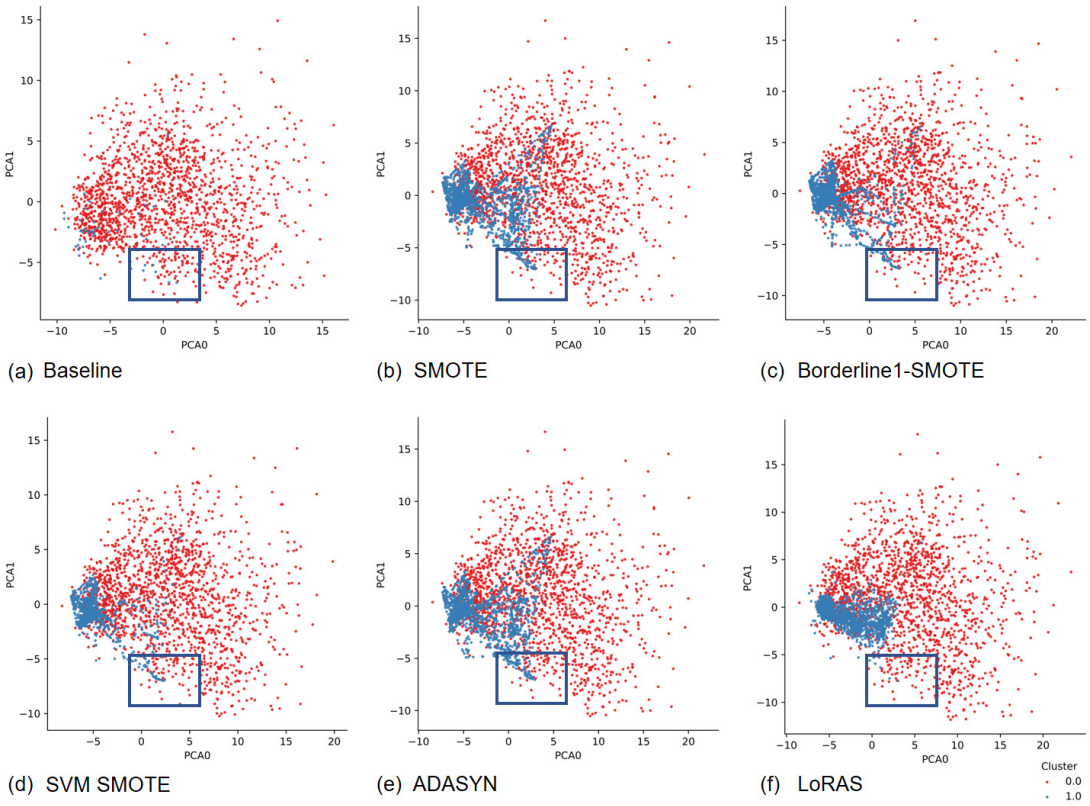


FIGURE 4.3: Figure showing for Principal Component Analysis plot of ozone dataset for baseline data and oversampled data with several oversampling strategies for the ozone_level dataset. The boxed region in each subplot shows a neighbourhood of outliers and how each oversampling strategy generates synthetic samples in that neighbourhood.

Oversampling with LoRAS produces comparatively balanced ML model performances on average, in terms of balanced accuracy and F1-Score among the compared convex-combination-strategy-based oversampling techniques. This is due to the fact that, in most cases, LoRAS produces lesser misclassifications on the majority class with a reasonably small compromise for misclassifications on the minority class. From the study I infer that for tabular high dimensional and highly imbalanced datasets, the LoRAS oversampling approach can better estimate the mean of the underlying local distribution for a minority class sample (considering it a random variable) and can improve the balanced accuracy and F1-Score of ML classification models. However, the scope of such convex combination based strategies, including LoRAS, might be limited for heterogeneous image-based imbalanced datasets.

The distribution of both the minority and majority class data points is considered in the oversampling techniques such as Borderline1 SMOTE, Borderline2 SMOTE, SVM-SMOTE, and ADASYN. SMOTE and LoRAS are the only two techniques, among the oversampling techniques I explored, that deal with the problem of imbalance just by generating new data points, independent of the distribution of the majority class data points. Thus, comparing LoRAS and SMOTE gives a fair impression about the performance of the novel LoRAS algorithm as an oversampling technique, without considering any aspect of the distribution of the minority and majority class data points and relying just on resampling. Other

extensions of SMOTE such as Borderline1 SMOTE, Borderline2 SMOTE, SVM-SMOTE, and ADASYN can also be built on the principle of LoRAS oversampling strategy. According to the analyses, LoRAS already reveals great potential for a broad variety of applications and evolves as a true alternative to SMOTE, while processing highly unbalanced datasets.

4.6 Integrating LoRAS with the UMAP algorithm

LoRAS generates synthetic samples from convex combinations of multiple *shadowsamples* from a neighbourhood. The shadowsamples are nothing but Gaussian noise added over the original samples. The approach of LoRAS enables synthetic data generation such that the local variance of a synthetic sample can be controlled in a minority class data neighbourhood. Moreover, the LoRAS algorithm integrates the philosophy of convex space modelling with prior learning of the minority class manifold using the t-SNE algorithm [Bej 2021].

However, the state-of-the-art manifold learning technique UMAP is known to preserve the global data structure better than t-SNE, while performing significantly faster than the t-SNE algorithm [McInnes 2018]. Both UMAP and t-SNE thrives on creating a higher and lower dimensional representation of data, and iteratively try to make the two representations as similar as possible [McInnes 2018]. Interestingly, for high-dimensional datasets UMAP models represent the data using a weighted neighbourhood graph construction approach in contrast to other algorithms, including t-SNE, which suffer from the curse of dimensionality [McInnes 2018]. McInnes *et al.* describe the algorithm as: "UMAP uses local manifold approximations and patches together with their local fuzzy simplicial set representations to construct a topological representation of the high-dimensional data. Given some low-dimensional representation of the data, a similar process can be used to construct an equivalent topological representation. UMAP then optimises the layout of the data representation in the low-dimensional space, to minimise the cross-entropy between the two topological representations" [McInnes 2018].

Considering the advantages of UMAP over t-SNE, I integrated UMAP in the LoRAS algorithm. Instead of t-SNE, I use the UMAP algorithm to learn the latent manifold of the minority class data.

4.7 Benchmarking studies for LoRAS UMAP algorithm

In terms of benchmarking study design, I have followed similar protocols as the original article on LoRAS [Bej 2021]. 14 datasets characterised by either of high imbalance ratio, high dimensionality and high absolute imbalance were selected for the benchmarking study, following the work of [Bej 2021].

Classification model selection for each dataset: The extensive benchmarking study for LoRAS, compared three classification models, namely k-Nearest neighbours (kNN), Logistic Regression (LR), and Support Vector Machine (SVM) [Bej 2021]. For each of the 14 benchmarking datasets, which are also used in this study, I have compared the F1-Scores of

Algorithm 2 LoRAS-UMAP Oversampling (GitHub link)

Inputs:

`C_maj`: Majority class parent data points
`C_min`: Minority class parent data points

Parameters:

`k`: Number of nearest neighbours to be considered per parent data point
$|S_p|$: Number of generated shadowsamples per parent data point
L_σ: List of standard deviations for normal distributions for adding noise to each feature
N_{aff}: Number of shadow points to be chosen for a random affine combination
N_{gen}: Number of generated LoRAS points for each nearest neighbour group
`n_neighbours`: Number of neighbours for UMAP embedding

Constraint:

$N_{aff} < k * |S_p|$

Initialise `loras_set` as an empty list
For each minority class parent data point `p` in `C_min` **do**
  `neighbourhood` ← calculate `k`-nearest neighbours of `p`, as per after dimension reduction by `UMAP(n_neighbours)`
  Initialise `neighbourhood_shadow_sample` as an empty list
  For each parent data point `q` in `neighbourhood` **do**
    `shadow_points` ← draw $|S_p|$ shadowsamples for `q` drawing noises from normal distributions with corresponding standard deviations L_σ containing elements for every feature
    Append `shadow_points` to `neighbourhood_shadow_sample`
  endfor
  Repeat
    `selected_points` ← select N_{aff} random shadow points from `neighbourhood_shadow_sample`
    `affine_weights` ← create and normalize random weights for `selected_points`
    `generated_LoRAS_sample_point` ← `selected_points` · `affine_weights`
    Append `generated_LoRAS_sample_point` to `loras_set`
  Until N_{gen} resulting points are created;
endfor
Return resulting set of generated LoRAS data points as `loras_set`

the classification models kNN, LR, and SVM. In the current study, for each of benchmarking datasets the classification model that has provided the highest F1-Score for each respective dataset is used [Bej 2021]. For the credit fraud dataset the Random forest model (RF) is used as it produces a better F1-Score compared to LR, SVM, and kNN.

As discussed in Section 3.3, there are several recent extensions of SMOTE that use diverse clustering and manifold learning techniques to learn the latent manifold of the minority class data. I have benchmarked the LoRAS-UMAP algorithm against four such algorithms: DBSMOTE, MOT2LD, SOMO, and CURE SMOTE. I choose the four algorithms in particular for the benchmarking study due to the following arguments:

- All of these algorithms use some unsupervised approach (either clustering or dimension reduction) for learning the minority class data manifold.
- The DBSMOTE algorithm was one of the first such algorithms.
- CURE SMOTE and SOMO are proposed recently in 2017.
- MOT2LD, proposed in 2015 uses the t-SNE for manifold learning of the minority class, same as the originally proposed LoRAS algorithm.

Choosing parameters for oversampling algorithms: Each of the oversampling algorithms used has their respective set of parameters. Notably, SMOTE, MOT2LD and LoRAS has one common parameter. This parameter accounts for deciding the oversampling neighbourhood around a minority class data point. For each dataset, I used the same value for this parameter for all oversampling algorithms. However, the datasets vary in size. For datasets, with less than 10 samples in the minority class, with more than 10 but less than 100 and with more than 100 a fixed value of 3, 5, and 30 is defined as the oversampling neighbourhood for

each algorithm. All other parameters of all other oversampling algorithms have been set to its default value. However, the philosophy of oversampling with LoRAS is different. Unlike SMOTE based algorithms, which model the convex space of the minority class using convex combinations of two close enough minority class samples, LoRAS models the convex space more rigorously considering convex combinations of multiple close enough shadowsamples (Gaussian noise added to minority class samples). For each of the 14 benchmarking datasets I already obtained an optimised parameter value for the number of shadowsamples to be chosen for a convex combination obtained by a random grid search (see column 4 in Table 3 in Bej *et al.*) [Bej 2021]. In the current study, I have used the same parameter value for this particular parameter for each dataset as obtained in the benchmarking study of LoRAS. For the rest of the parameters of LoRAS default values were used to keep this study simple.

Training and validation of classification models: Given a dataset and a classification model, I used a specific protocol to train and test the classification model. First, I used 5×10-fold stratified cross validation for most of the chosen datasets. However, for datasets with less than 30 samples in the minority class (arrhythmia, ar1, and ar3), I have used 5×3-fold stratified cross validation. For each cross-validation fold, only the training set is used for oversampling. The oversampled training fold is used to train the classification model. The trained model is then tested using the test fold, which was not oversampled. I have used the F1-Score and balanced accuracy as performance measures following Bej *et al.* [Bej 2021]. F1-Score, the harmonic mean of the precision and recall measures, plays a important role to measure how accurate the classification model perceives the minority class. balanced accuracy, the average class-wise accuracy, on the other hand, measures how accurate the model classifies both classes on average and thus takes the accuracy of the classification accuracy of the majority class into account. In Bej *et al.*, observe that depending on the oversampling model used, the classification models tend to compromise on either the balanced accuracy or the F1-Score. Models with higher F1-Score tend to produce lower balanced accuracy and vice versa. The LoRAS algorithm performed best in terms of producing the highest F1-Score, while compromising minimally on the balanced accuracy. For coding, I used the `scikit-learn (V 0.21.2)`, `numpy (V 1.16.4)`, `pandas (V 0.24.2)`, and `matplotlib (V 3.1.0)` libraries in `Python (V 3.7.4)`.

4.8 Improved performance of LoRAS UMAP algorithm

LoRAS-UMAP produces the best F1-Score, while compromising minimally on the balanced accuracy: Table 4.7 presents the results of benchmarking study . SMOTE, improves both the F1-Score and the balanced accuracy compared to the Baseline training (training classification models without oversampling), producing an average F1-Score and balanced accuracy of 0.372 and 0.785 respectively. Interestingly, the DBSMOTE algorithm improves neither of F1-Score and balanced accuracy compared to SMOTE. For most of the datasets I have used for the benchmarking study, the minority class is quite sparse. Thus, a density-based clustering algorithm as employed by DBSMOTE fails to detect significant clusters and since it focuses on producing between-cluster samples only, there are hardly any significant synthetic samples produced in significant regions that can affect

TABLE 4.7: Table showing balanced accuracy/F1-Score for several oversampling strategies (Baseline, SMOTE, SVM-SMOTE, MOT2LD, DBSMOTE, CURE SMOTE, SOMO, LoRAS t-SNE, and LoRAS-UMAP) for all 14 benchmarking datasets

Dataset	ML	Baseline	SMOTE	MOT2LD	DBSMOTE	CURE SMOTE	SOMO	LoRAS t-SNE	LoRAS-UMAP
abalone19	kNN	.500/.000	.663/.052	.506/.0166	.538/.050	.607/.056	.500/.000	.527/.035	.631/.058
arrythmia	LR	.320/.341	.665/.361	.547/.178	.656/.340	.664/.353	.656/.340	.667/.353	.671/361
isolet	LR	.844/.802	.94/.808	.902/.806	.901/.802	.904/.803	.901/.802	.902/.801	.904/.805
mammography	kNN	.724/.580	.911/.469	.892/.594	.859/.532	.866/.505	.734/.589	.886/.523	.875/.548
ozone_level	LR	.265/.053	.804/.193	.728/.208	.515/.057	..785/.191	.515/.092	.807/.205	.798/.209
scene	LR	.190/.184	.61/.217	.612/.225	.562/.188	.611/.223	.558/.184	.612/.220	.618/.228
yeast-ml8	kNN	.500/.002	.559/.153	.562/.154	.5004/.002	.575/.1587	.500/.002	.565/.154	.57/.156
yeast-me2	kNN	.534/.114	.834/.329	.797/.404	.590/.221	.791/.350	.534/.114	.835/.335	.844/.366
wine-quality	LR	.259/.068	.726/.183	.698/.171	.705/.174	.706/.221	.519/071	.704/.189	.701/.169
letter-img	kNN	.979/.971	.995/.920	.995/.921	.981/.960	.992/.963	.979/.961	.994/.960	.992/.959
webpage	kNN	.855/.776	.924/.394	.928/.510	.856/.776	.925/.495	.858/.632	.903/.725	.900/.733
ar1	kNN	.500/.000	.690/.283	.581/.183	.500/.000	.586/.210	.498/.000	..581/.185	.627/.25
ar3	RF	..500/.000	.747/.481	.533/.079	.500/.000	.500/.000	.531/.099	.766/.488	.758/.469
credit fraud	RF	.773/.665	.923/.359	.906/.772	.759/.641	.906/.805	.773/.665	.904/.818	.902/.819
Average	-	.553/.325	.785/.372	.727/.373	.673/.339	.744/.381	.647/.326	.761/.428	.771/.438
Average rank	-	7.5/5.85	2.25/4.5	3.85/4.10	6.10/5.64	3.60/3.71	6.75/5.71	3.10/3.89	2.82/2.57

the decision boundary. That is why its performance is not much of an improvement over the baseline case. The MOT2LD algorithm also uses a density-based algorithm along eith the manifold learning technique t-SNE. It marginally improves the average F1-Score of SMOTE but compromises heavily on the balanced accuracy. The SOMO algorithm also takes into account the density factor of clusters during oversampling. However, it generates both between-cluster and within-cluster samples. It thus performs slightly better than DBSMOTE. The CURE algorithm improves the F1-Score compared to SMOTE to a minor extent, compromising relatively less on the balanced accuracy. The LoRAS algorithm integrated with t-SNE improves the F1-SCORE compared to CURE, and compromises minimally on the balanced accuracy achieved by SMOTE. The LoRAS-UMAP algorithm outperforms the LoRAS algorithm integrated with t-SNE in terms of both F1-Score and balanced accuracy.

LoRAS-UMAP is consistently better than the LoRAS t-SNE for high-dimensional imbalanced datasets: In the benchmarking experiments, there are five high-dimensional datasets (arrhythmia, isolet, scene, yeast-ml8, and webpage), with the number of features more than 100. It is noteworthy that for all these high-dimensional datasets, LoRAS-UMAP outperforms the LoRAS algorithm combined with t-SNE, in terms of F1-Score. In terms of balanced accuracy, LoRAS-UMAP outperforms the LoRAS algorithm combined with t-SNE for four out of five such datasets (except for the webpage dataset).

LoRAS-UMAP is faster compared to LoRAS t-SNE: A disadvantage of the rigorous convex space learning approach of LoRAS algorithm is that with the growing number of features the computational complexity gets higher. The use of the UMAP algorithm for manifold learning can thus make the algorithm faster compared to using t-SNE. With LoRAS-UMAP, the manifold learning step takes less time as UMAP and is significantly faster than t-SNE. For datasets with a high number of minority class samples, this makes LoRAS-UMAP marginally faster than LoRAS t-SNE. I demonstrate this by recording the data generation times for three datasets (over the entire datasets), isolet, letter image, and webpage, which have also a relatively high number of minority class samples (600, 734, and 981 respectively). Note that, for these three datasets, the data generation process with LoRAS-UMAP is approximately 14, 7, and 45 seconds faster than LoRAS t-SNE using a standard PC (Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 1992 Mhz, 4 Core(s), 8 Logical

Processor(s), 16 GB RAM).

TABLE 4.8: Table showing the results of Wilcoxon's signed rank test for comparison of LoRAS-UMAP with other oversampling algorithms.

Dataset	Baseline	SMOTE	MOT2LD	DBSMOTE	CURE SMOTE	SOMO	LoRAS t-SNE
p-value (F1 Score)	0.011	0.084	0.055	0.011	0.030	0.003	0.055
p-value (Balanced acc.)	0.000	0.078	0.055	0.001	0.064	0.000	0.593
W_+/W_- (F1 Score)	99/6	95/10	95/10	99/6	99/6	102/3	99/6
W_+/W_- (Balanced acc.)	105/0	60/45	95/10	104/1	90/15	105/0	77/28
R (F1 Score)	0.78	0.713	0.629	0.780	0.780	0.830	0.780
R (Balanced acc.)	0.88	0.12	0.713	0.864	0.663	0.888	0.411

Interpretation of results: From Table 4.8, it is evident that p-values of the Wicoxon's test alone do not reflect the potential of the proposed algorithm fully. Although LoRAS-UMAP in comparison to SMOTE, MOT2LD, and LORAS t-SNE does not show a p-value less than 0.05 for both balanced accuracy and F1-Score, observe that for all models $W_+ > W_-$ in favour of LoRAS-UMAP, for both balanced accuracy and F1-Score. Thus, if each individual oversampling model is compared against LoRAS-UMAP over 14 benchmarking datasets, the newly proposed extension performs better. Note that in terms of F1-Score, the MOT2LD algorithm also performs well even though it is not reflected in the average scores in Table 4.7. In terms of F1-Score, LoRAS-UMAP does not improve the balanced accuracy over SMOTE much, but in terms of F1-Score one can observe an improvement on R value (0.713). However, UMAP, being able to preserve the global as well as local structure, proves to be more effective than t-SNE used by both MOT2LD and LoRAS t-SNE for the purpose of manifold learning. Moreover, the global structure preservation property of the UMAP algorithm also results in a better dimension reduction for high-dimensional datasets, which improves the performance of LoRAS algorithm on high-dimensional datasets. The results are also supporting this finding, where LoRAS-UMAP performs better than all compared algorithms in terms of F1-Score and is only worse than SMOTE in terms of balanced accuracy when compared over all benchmarking datasets. The superior performance of LoRAS t-SNE and LoRAS-UMAP indicates that using a manifold learning algorithm to learn the latent data manifold is an effective approach to learn from imbalanced datasets.

An implementation of the LoRAS algorithm for binary classification problems using `Python (V 3.7.4)` and several examples `Jupyter Notebooks` from the benchmarking study is provided in the GitHub repository: `https://github.com/COSPOV/LoRAS`.

After rigorous analysis of the performance of the LoRAS algorithm on publicly available benchmarking datasets, the next chapter proceeds to an application of the LoRAS algorithm. In the next chapter, a LoRAS-based tool, sc-SynO, for automated detection of rare cell types from single cell data is introduced.

Chapter 5

Automated annotation of rare cell populations

This chapter presents a real-life application of the LoRAS algorithm in the form of the sc-SynO tool designed for automated annotation of rare cell populations from single-cell data. Single-cell data has been chosen for the study because of its vast potential in biomedical research. Firstly, the chapter discusses briefly the single-cell technology, followed by the idea of *in silico* cell generation using synthetic oversampling. The next part of the chapter then describes the preprocessing techniques for preparation of the data for the study. Next, the detailed protocol of the sc-SynO tool is discussed. I then demonstrate through data-based experiments the effectiveness of the tool in detecting and thereby annotating two rare cell populations. Finally, the chapter is concluded through discussions on the applicability and importance of sc-SynO.

5.1 Applying LoRAS in a biological context

Single-cell RNA-sequencing (scRNA-Seq), as well as single-nuclei RNA-sequencing (snRNA-Seq), open up a transcriptome-wide gene expression measurement at single-cell level, enabling cell-type cluster identification, the arrangement of populations of cells according to novel hierarchies, and the identification of cells transitioning between individual states [Lähnemann 2020]. This facilitates the investigation of underlying structures in tissue, organism development, and diseases, as well as the identification of unique subpopulations in cell populations that were so far perceived as homogeneous.

5.1.1 Using single-cell technology for the identification of rare cells

Classifying cells into cell types or states is essential for many biological analyses [Lee 2020]. For example, investigating gene expression changes within a cell type or cell subpopulation can be of high interest across different biological conditions, time-points, or in patient samples. To be able to compare these different cell types, reliable reference systems, especially in sparse-cell states, are necessary. However, the lack of markers for rare-cell types motivates the use of unsupervised clustering approaches. Method development for such unsupervised clustering of cells has already reached a certain level of

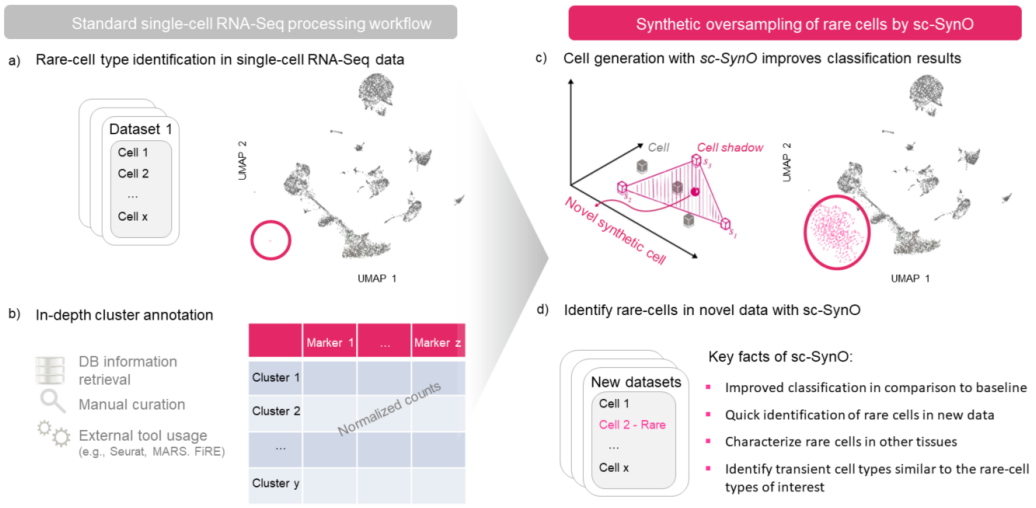


FIGURE 5.1: Visualization of the workflow, demonstrating a step-by-step explanation for a sc-SynO analysis. **a)** Several or one snRNA-Seq or scRNA-Seq fastq datasets can be used as an input. Here, cell population of interest are identified and provide raw or normalized read counts of this specific population. This can be done with any single-cell analysis workflow, e.g., Seurat. **b)** Further information are extracted for cluster annotation that serve as improved input for the subsequent training with sc-SynO. **c)** Based on the data input, the underlying LoRAS [Bej 2021] synthetic oversampling algorithm of sc-SynO is utilized to generate new cells around the former origin of cells to increase the size of the minority sample. **d)** The trained Machine Learning classifier is validated on the trained, pre-annotated dataset to evaluate the performance metrics of the actual model. The sc-SynO model is now ready to identify the learned rare-cell type in novel data. This figure was solely created by the authors.

maturity [Duo 2018, Freytag 2018, Kiselev 2019]. Furthermore, many studies are interested in specialised cells (e.g., cancer cells, cardiac pacemaker cells) with an occurrence of less than 1 in 1000. The identification of such clusters, solely based on unsupervised clustering of a single dataset, remains very challenging [Jindal 2018]. For this reason, almost every cell clustering characterisation approach is driven by manual cluster annotation, which is time-consuming and involves a bias of the annotating domain expert, thus limiting the reproducibility of results. One possible solution requires a so-called cell atlas, as a curated reference system that systematically captures cell types and states, either tissue-specific or across different tissues [Zhang 2018].

Here, it is shown how the limitation of identifying already annotated rare-cell types in newly generated scRNA-Seq data can be overcome, by using a synthetic oversampling approach (sc-SynO). sc-SynO is able to automatically identify rare-cell types in an unbiased and precise manner in novel data.

5.1.2 Using machine learning algorithms to generate cell types *in silico*

Machine learning (ML) algorithms are widely used to deal with classification problems and, thus, are used here to automate the annotation of rare-cell types from single-cell or nuclei RNA-Seq data. However, the scarce number of these cells within samples (less than 1 out of 1000 cells) often results in highly imbalanced data. An imbalanced dataset is a type of dataset where one or more classes have a significantly less number of samples compared

to other classes (e.g., sinus node cells in the heart). A class having such a low number (minority class) is difficult to detect for unsupervised clustering approaches or classification algorithms in general [Jindal 2018].

The reason behind this is the inability of ML algorithms to perceive or learn underlying patterns from the minority class due to the scarcity of samples and thereby failure of these algorithms to classify them properly [Satoso 2017].

To overcome the problem of imbalance, oversampling techniques have been an area of research in the field of ML for more than a decade. Among several approaches proposed to deal with such issues is the approach of synthetic oversampling [Weiss 2007]. The philosophy of generating synthetic samples is to impute minority class instances, here cell types, in an attempt to enhance the capacity of an ML algorithm to learn. The idea of oversampling is thus commonly used to rebalance the classes [Satoso 2017]. I have discussed oversampling in much more details in previous chapters.

In this study, the ML-based annotations of rare cells were compared and benchmarked with no oversampling and the most commonly used oversampling algorithm SMOTE [Chawla 2002], as well as sc-SynO approach based on the LoRAS algorithm [Bej 2021]. LoRAS integrates the idea of approximating the minority class data manifold with a more comprehensive modelling of the convex data space of the minority class, while generating synthetic minority class samples, resulting in more balanced model performance in terms of precision and recall. To the best of my knowledge, this is the first time that an oversampling approach is applied to single-cell RNA-Seq data for rare-cell detection improvement. The workflow can be obtained in Fig.5.1 and is available on GitHub (`https://github.com/COSPOV/sc-SynO`). For more details, please see the Section 5.2.

5.2 Datasets and methodologies

5.2.1 Use case preparation

To evaluate the potential of synthetic oversampling to precisely annotate cell populations in newly generated data, three use cases were generated, by utilising already published single-cell and nuclei RNA-Seq datasets. Normalised read counts were processed with Seurat [Butler 2018] (any other normalisation method is also applicable). These are then used as an input to generate the synthetic samples and train the different ML classifiers. In addition, the influence of using all transcripts for a classification were tested, or only the top 20, 50, or 100 preselected ones (basic feature selection function of Seurat 3 was used). This helps us to investigate the influence of further downstream information obtained from standard feature selection workflows that are usually applied during scRNA-Seq analysis.

The first use case identifies cardiac glial cells in snRNA-Seq data (17 nuclei out of 8,635) [Wolfien *Cells* 2020], which were used as a training set. The trained sc-SynO ML-classifier was subsequently applied to the independently generated snRNA-Seq data sets of Wolfien *et al.* [Wolfien *Cardiovascular Research* 2020] and Vidal *et al.* [Vidal 2019] to automatically detect the cardiac glial cells (Glial cells). This use case was designed to take a larger imbalance ratio (~1 to 500) into account and only uses single-nuclei data.

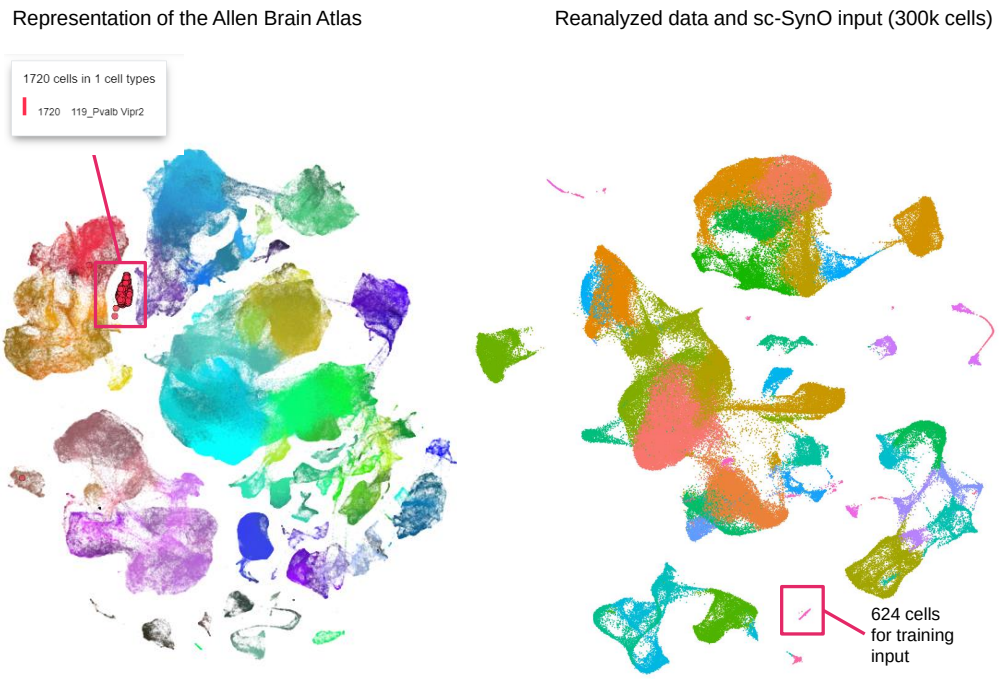


FIGURE 5.2: Comparison of the Allen Brain Atlas mice data of the whole dataset from (https://celltypes.brain-map.org/) and the reanalysis. The 119_Pvalb Vipr2 cluster, consisting in total of 1,720 cells, was chosen as a rare-cell type of interest. The sc-SynO input was 624 cells of this cell type obtained from the first 300,000 cells in the data.

In contrast to this, the second use case was designed to jointly use snRNA-Seq data and scRNA-Seq on a lower imbalance ratio (~1 to 26) for the training step to likewise investigate the potential of the algorithm to consider both single cell capture procedures and the impact of "less" rare-cell types. In particular, studies of Galow *et al.* [Galow 2020] (snRNA-Seq), and Linscheid *et al.* [Linscheid 2019] (scRNA-Seq) were used to identify prolifertive cardiomyocytes (Prl cardio).

The third data set refers to the murine data of the Allen Brain Atlas (https://celltypes.brain-map.org/), which serves as an example for a large dataset. Here, the expression of the first 300,000 cells has been used as a model training input to identify cells in the 119_Pvalb Vipr2 cluster (Figure 5.2). Only models based on previously selected features were used for a downstream comparison because training on such a large dataset including all available transcriptomic features demands excessive computational resources. The validation of the trained model was performed on additional 300,000 cells of the murine Allen Brain Atlas dataset.

For validation purposes, all datasets have also been analysed traditionally using common data analysis approaches, such as the Seurat workflow, as already described elsewhere [Wolfien *Cardiovascular Research* 2020]. These additional experiments have been conducted to manually evaluate the identified cells from sc-SynO via traditional clustering. The computational scripts for data preprocessing in R and sc-SynO model generation in Python can be retrieved from the FairdomHub instance (https://fairdomhub.org/assays/

1368) and GitHub (https://github.com/COSPOV/sc-SynO). Key statistics, such as the imbalance ratio (Imb. ratio), number of minority samples, cross-validation folds, and oversampling neighbours of the use cases, are presented in Table 5.1.

TABLE 5.1: Key statistics of the datasets used during this study. The column 'Oversampling nbd' shows the number of nearest neighbours considered for each minority class data points to generate synthetic samples

Dataset	Imb. ratio	minority samples	CV folds	Oversampling nbd
Glial cells	506.94	17	3	3
Prl cardio	26.21	625	10	30
Brain atlas	348.5	624	10	30

5.2.2 sc-SynO: Transferring the LoRAS algorithm to single-cell data

The LoRAS algorithm is designed to create a better approximation of the underlying data manifold by a rigorous modelling of the convex data space compared to pre-existing algorithms, like SMOTE and several of its already presented extensions. A brief outline of the sample generation of sc-SynO approach, as well as the resulting benefits, are shown in Figure 5.1. To generate a synthetic sample, the algorithm first considers the k-nearest neighbours of a minority class data point from a two-dimensional embedding of the minority class achieved by using t-SNE (this is an optional step). When there are enough data points in the minority class, this provides the algorithm a better approximation of the local data manifold for the minority class.

Once the k-nearest neighbours are decided for a minority class data point p and thereby the neighbourhood of p is identified, Gaussian noise is added to all the data points in the neighbourhood of p. The pseudo data points generated by the Gaussian noise are called shadowsamples. A random convex combination of multiple shadowsamples is used to create a Synthetic LoRAS sample. A mathematical explanation of the algorithm asserts that, using a convex combination of multiple shadowsamples in LoRAS, we can produce a better estimate of the local mean considering the synthetic samples generated in a neighbourhood are random variables [Bej 2021].A brief outline of the sample generation of sc-SynO approach, as well as the resulting benefits, are shown in Figure 5.1. The details of the LoRAS algorithm is discussed in Chapter 4.

ML model description: For the benchmark study, the k-nearest neighbours (kNN) and logistic regression model (LR) were chosen as ML classifiers. The reason behind choosing kNN is that this model is known to perform well for imbalanced datasets, especially while using oversampling algorithms [Blagus 2013]. The LR model was used because of its effectiveness in other benchmarking studies using different imbalanced datasets, is performing well jointly with the LoRAS oversampling algorithm [Bej 2021]. The kNN model was used with $k = 30$ parameter value. After oversampling, there are almost equal data points in the majority and the minority class. For the kNN classifier model, a k value of thirty was chosen to ensure that the classifier's decision is made on a statistically significant number of samples. The LR model was used with default parameter settings.

Given the proliferative cardiomyocytes dataset, for every ML model, a 5×10-fold stratified cross-validation framework was implemented to judge model performances. For the cardiac glial cell dataset, due to the minuscule minority class of only 17 cells, a 5×3-fold stratified cross-validation was implemented. First, the dataset was scuffled randomly. The dataset was divided into k-folds depending on the dataset as described above. The folds are kept distinct, maintaining approximately the same imbalance ratio in each fold. After training and testing the models using stratified cross validation, an appropriate model for a given dataset based on the F1-score and balanced accuracy was identified. The selected model was then trained over the whole dataset and is then used to detect rare cells in two corresponding validation datasets.

Oversampling procedure: Although there are several other oversampling strategies, convex combination-based oversampling can work particularly well when there are too few data points in the minority class due to a lesser chance of overfitting.

For every test fold, I oversample only on the training fold, so that the test fold is completely unseen to the classifiers. I specify the most important parameter values of the oversampling model to ensure the full reproducibility of the models. For the proliferative cardiomyocytes dataset having 625 minority class samples, I choose 30 of the nearest neighbours of a minority class sample, as the oversampling neighbourhood for sc-SynO. sc-SynO has some additional parameters, such as N_{aff}, L_{σ}, and N_{gen}, enabling a better approximation of the minority class data manifold. For the Glial cell dataset, with only 17 minority class samples, I use three of the nearest neighbours of a minority class sample, as well as the oversampling neighbourhood. The *num_afcomb* parameter is chosen to be 23 and 100 for the two cases studies of the proliferative cardiomyocytes dataset with 23 and 100 prioritized marker-genes, respectively. For the Glial cell dataset, *num_afcomb* is chosen to be 50 in both case studies. For detailed parameter values, please see the code published on FairdomHub (`https://fairdomhub.org/assays/1368`).

Choosing proper performance metrics are also often a challenge for imbalanced datasets. The usual performance measures, such as accuracy or area under the curve (AUC) of receiver operating characteristic (ROC) might be unreliable in this scenario [Saito 2015]. In this study, I used three performance measures, precision, recall, and F1-Score (Harmonic mean of precision and recall). Here, a high precision model indicates a low number of FN cells, relative to the number of TPs and a high-recall model indicates a higher proportion of TP cells. These two measures can provide in combination a fair understanding of a classifier performance on the underlying datasets. However, in the specific case of detecting rare cells, the primary priority can be user assigned to either prefer a high recall or precision score by appropriately choosing above-mentioned parameters during the training step. The F1-Score determines, how well the model is balanced in terms of precision and recall. In combination, these scores can indicate how appropriate the classification model has performed.

5.3 sc-SynO can detect rare cell types

Based on analysed data of all three use cases, this study shows that the preselection of features (e.g., marker gene identification via Seurat or ML-based or manually driven)

is an important preprocessing step for rare-cell type detection. Preselection of features not only results in faster classification models, but also produces more reliable results than using all possible features. Table 5.2 shows the comparison of runtimes for several preselection scenarios using a kNN model. One can observe a much higher runtime without preselection of features. Moreover, the performance of the predictive model on both validation datasets without preselection is highly unreliable. In the validation dataset VD1 and VD2, there are only five and three cardiac glial cells, respectively, as per expert annotations. In contrast, preselection of features based on prior marker identification per cluster yields much more accurate results [Vidal 2019, Wolfien *Cells* 2020]. Without preselection the predictive model uses numerous features leading to an overfitted model. For example, without feature selection, the predicted number of glial cells in VD1 is 0 without any oversampling and 2423 using sc-SynO. For this reason, I recommend using the workflow based on preselected features obtained from manually curated feature selection methods in the in-depth comparisons in this work.

TABLE 5.2: Table showing comparisons among several feature preselection scenarios in terms of run time and efficiency in detection of glial cells for two different validation datasets (VD 1 & 2)

Dataset	Pre-selection	Pre-processing	Data generation	Training	Detected/Actual cells
VD1	All features	None	-	11min 56s	0/5
VD2	All features	None	-	2min 19s	0/3
VD1	All features	sc-SynO	4min 3s	28min 24s	2423/5
VD2	All features	sc-SynO	4min 19s	5min 8s	299/3
VD1	50 features	None	-	2.4 s	4
VD2	50 features	None	-	408 ms	2
VD1	50 features	sc-SynO	1.23 s	1.94 s	5/5
VD2	50 features	sc-SynO	1.12 s	484 ms	3/3
VD1	20 features	None	-	706 ms	4
VD2	20 features	None	-	240 ms	1
VD1	20 features	sc-SynO	1.12 s	679 ms	6/5
VD2	20 features	sc-SynO	1.11 s	236 ms	3/3

For simplicity, I have chosen the in-build marker gene identification method from Seurat as a feature input in sc-SynO. Other commonly available methods, such as Random Forest or any other feature selection methods to derive important transcripts for the rare-cell type of interest are also suitable. Analyses of population-specific marker genes are commonly performed for all single-cell pipelines and can contribute to the biological explainability of the model. For this reason, I have shown different case studies with different amounts of pre-selected features (e.g., 20, 50, and 100).

However, I deliberately refrain from recommending a fixed number of features for using sc-SynO because every dataset has different characteristics, and it is a common practice to tune model parameters in a dataset-specific way in machine learning. To fully understand the specific sc-SynO synthetic cell generation, users should test different parameters, including the best amount of features by using methods, such as random grid search. Here, the results show that even with less than 100 features, the models are able to detect rare cells successfully.

In general, I observed that the synthetic cells were generated close to the original minority class data using UMAP visualizations (Figure 5.3). To show the newly introduced cells by sc-SynO, the generated cells are highlighted for the proliferative CM cluster (50, 500, 1000, and 2000 cells respectively) and the original cells. I observed for the increased amount of *in silico* generated cells in a stretch of the cell cluster because with larger amounts of synthetic cells the space stretches to maintain the assumption of uniformity of data distribution in UMAP.

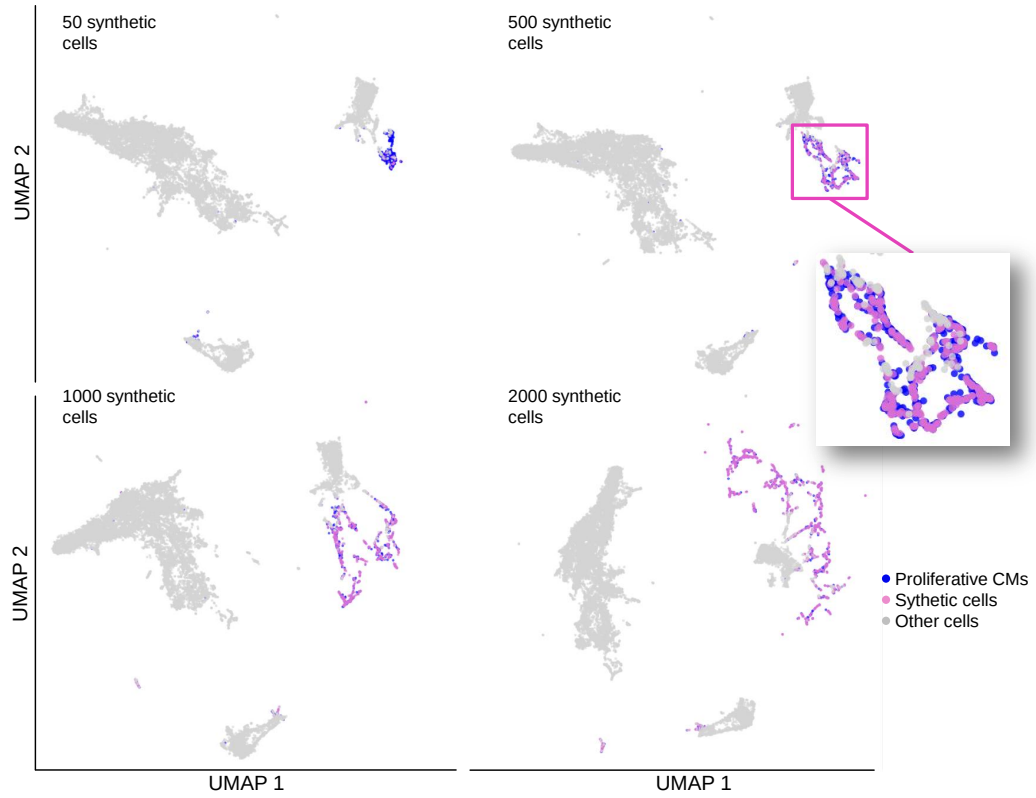


FIGURE 5.3: Figure showing a comparison between the distribution of synthetic cells (purple) generated by sc-SynO and original input cells (blue) for model training.

The results for all model training cases, including pre-specified cellular markers and 5-fold stratified cross validation, are presented in Figure 5.4 and more detailed in Table 5.3.

5.3.1 sc-SynO can detect extremely rare glial cells

Training data: For the cardiac glial cell dataset, all models except for the kNN-Baseline model produce an F1-Score of more than 90 percent. For the LR model with 20 features, one can observe that the recall is 1 irrespective of the model used. The reason behind the superior performance of all models in this case is that, even though the glial cell cluster is extremely rare, it is also very well separated from the rest of the clusters, making it easy for machine learning models to detect cells.

TABLE 5.3: Table showing F1-Scores/Precision/Recall for sc-SynO against baseline classification for the two ML classifiers (LR and kNN) and for several numbers of pre-selected features (Marker genes). 119_Pvalb represents a small subpopulation of the Allen Brain atlas.

Dataset	ML	Features	Baseline	sc-SynO
Glial cells	LR	20	.96/.94/1	.96/.94/1
Glial cells	kNN	20	.97/1/.95	.94/.90/1
Glial cells	LR	50	.90/.94/.88	.94/.94/.95
Glial cells	kNN	50	.89/1/.81	.94/.90/1
Prl cardio	LR	20	.80/.87/.74	.72/.62/.86
Prl cardio	kNN	20	.86/.91/.81	.85/.79/.92
Prl cardio	LR	100	.86/.88/.84	.84/.81/.87
Prl cardio	kNN	100	.86/.95/.79	.79/.68/.95
119_Pvalb	LR	50	.43/.45/.41	.65/.49/.99
119_Pvalb	LR	100	.45/.48/.42	.65/.49/.99

Validation: I tested the baseline case (without oversampling) against sc-SynO using the LR model with 20 features, which was trained on snRNA-Seq normalized read count data of two independent snRNA-Seq data sets. Both, the baseline model and sc-SynO, identified four out of five cardiac glial cells in the first validation set of Wolfien *et al.* [Wolfien *Cells* 2020] (Figure 5.5A). For the second validation dataset from Vidal *et al.* [Vidal 2019] (Figure 5.5B) sc-SynO was able to detect three out of three glial cells, whereas the baseline model was able to detect only one. Figure 5.5C shows the average gene expression of particular cardiac glial cell markers that are highly expressed in the identified clusters and weakly in other clusters. Although sc-SynO was effective in finding rare cells, as I have observed in this case study, the case study itself does not deterministically prove the advantage of oversampling over the baseline case. The reason behind this is, as I have discussed before, that the cluster of glial cell is already well-separated within the dataset. That is why to appreciate the effectiveness of the tool, I provide the next case study on proliferative cardiomyocyte detection. Although this cell type is not as rare as the glial cells, it is a transient cell type that is not well separated from the neighbouring cluster and, thus, can appropriately show the variation in performance of the considered models.

5.3.2 Sc-SynO achieves a low FN rate for the identification of proliferative cardiomyocytes

Training data: For the proliferative cardiomyocytes dataset, I, notice that the performance of the classifiers clearly improves with including more marker genes as features (Figure 5.6A, 5.6B). Note that, for all baseline models in this case, the precision is quite high. In contrast, the recall in turn is low. Due to the high precision, a high F1-Score is also maintained for the baseline models. However, given that the goal is to detect rare cells, a low recall means that the baseline models are not very effective to execute this task, even though they produce low FNs. Interestingly, sc-SynO in turn, improves the recall, which facilitates the detection of rare cells, compromising on the precision of the classifiers. Clearly, in this case, the kNN classifiers produce a higher recall and F1-score compared to

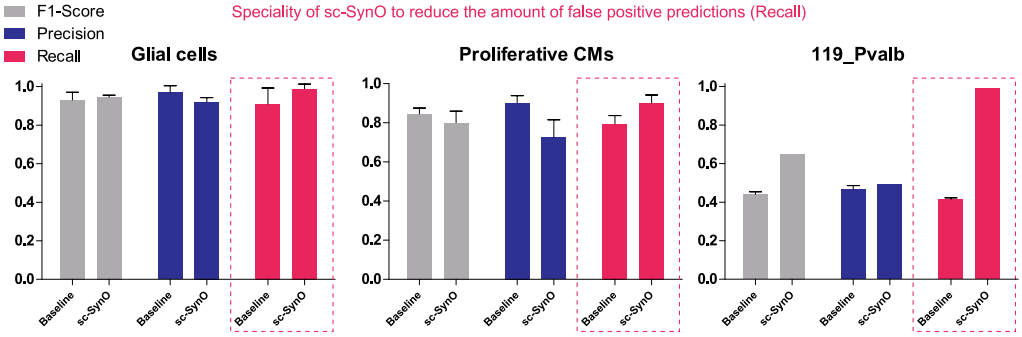


FIGURE 5.4: Comparison of the baseline classification and sc-SynO visualized as mean outcome for the used quality parameter: F1-Score (grey), Precision (blue), and Recall (purple). Detailed results can be obtained in Table 5.3. I observe that in every case, sc-SynO improves the recall compared to the Baseline model (see dotted boxes in the figure). This ensures that oversampling with sc-SynO improves the detection rate of rare-cell types.

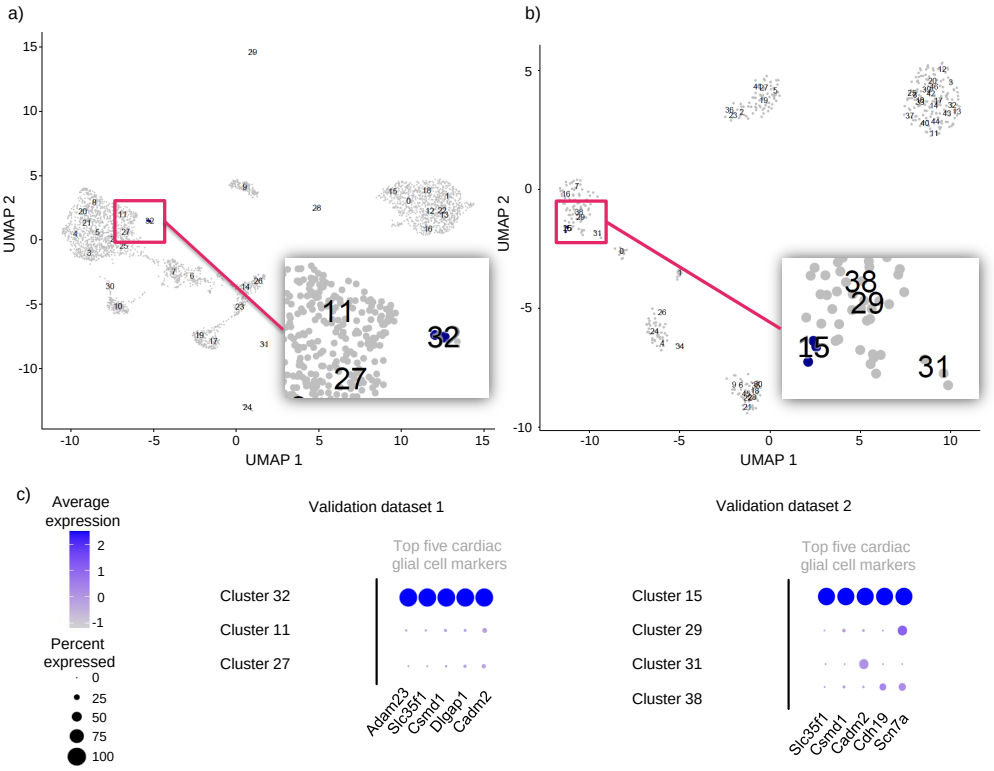


FIGURE 5.5: Validation of the sc-SynO model for the first use case of cardiac glial cell annotation. **a)** UMAP representation of the manually clustered Bl6 dataset of Wolfien *et al.* [Wolfien *Cardiovascular Research* 2020]. Predicted cells of sc-SynO are highlighted in blue, cells not chosen are grey. **b)** UMAP representation of the manually clustered dataset of Vidal *et al.* [Vidal 2019]. Predicted cells of sc-SynO are highlighted in blue, cells not chosen are grey. **c)** Average expression of the respective top five cardiac glial cell marker genes for both validation sets, including the predicted clusters and those in proximity.

the LR classifiers. With 20 features using kNN for sc-SynO an improved recall to 92% can be observed, while the F1-Score remains comparable to baseline.

Validation: I applied the baseline model and the sc-SynO algorithm on two validation datasets for proliferative cardiomyocytes, using the kNN classifier with 20 features. While

sc-SynO was able to identify 10 out of 11 cells, the baseline case could not detect any cells from the first validation dataset. Interestingly, the model using the top 100 genes identifies 48 cells, including all 9 cells from the top 20 model, which may imply that this higher set of transcripts can detect a larger, yet similar, set of cells that are closely related to the cells of investigation. Since the second use case was about a transient cell type, the assigned cells of the model might indicate related cells that have already been or closely to enter the actual state of a proliferative cardiomyocyte. The second validation set assigned 40 cells out of 67 correctly (top 20 features). By using 100 features, the amount of correctly assigned cells increased further. In both cases, the capability of the baseline model to detect numerous cells was limited (None with 20 features and only 29 with 100 features).

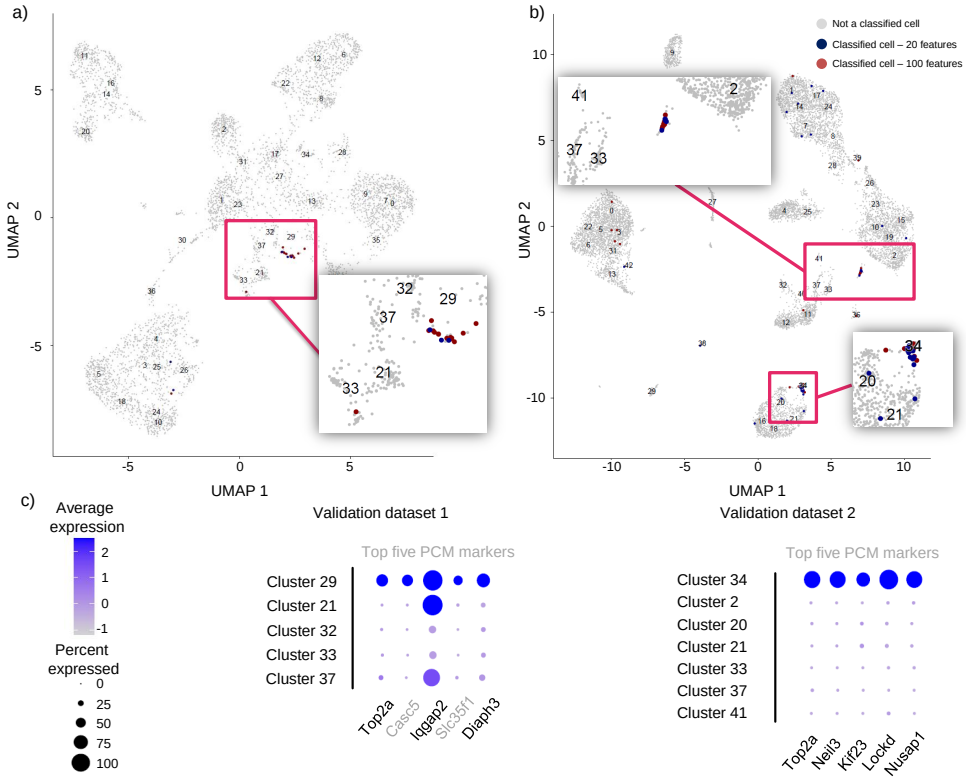


FIGURE 5.6: Validation of the sc-SynO model for the second use case of proliferative cardiomyocyte annotation. **a)** UMAP representation of the manually clustered single-nuclei dataset of Linscheid *et al.* [Linscheid 2019]. Predicted cells of sc-SynO are highlighted in blue (based on top 20 selected features in the training model), red (based on top 100 selected features in the training model) cells not chosen are grey. **b)** UMAP representation of the manually clustered dataset of Vidal *et al.* [Vidal 2019]. Predicted cells of sc-SynO are highlighted in blue (based on top 20 selected features in the training model), red (based on top 100 selected features in the training model) cells not chosen are grey. **c)** Average expression of the respective top five proliferative cardiomyocyte marker genes for both validation sets, including the predicted clusters and those in proximity.

5.3.3 Sc-SynO can detect rare-cell populations from large-scale datasets

Training data: To test the effectiveness of sc-SynO on large-scale datasets, the approach was performed on the murine data of the Allen Brain Atlas (`https://celltypes.brain-map.org/`), which includes more than 1,000,000 cells. In the third case study, a small population of cells was chosen (119_Pvalb Vipr2, see Figure 5.2) and tested the effectiveness

of baseline classifiers and sc-SynO on detecting this rare cell-population. The rare-cell population of interest has an imbalance ratio of 348.5. Since the dataset was very large, I performed a pilot study to notice that LR is not only the fastest among the models, but performs significantly better than kNN in this case (dns). Thus, I used only the LR model for the classification task. I noticed that sc-SynO significantly improves the classifier performance. With 20 features, the precision, recall, and F1-Score for the baseline model where 0.486, 0.424, and 0.450 respectively, while with 50 features the values were 0.456, 0.416, and 0.433. For sc-SynO, with 20 features the precision, recall, and F1-Score were 0.492, 0.988, and 0.655 respectively, while with 50 features the values were 0.496, 0.990, and 0.665.

Validation: For validation, I tested the baseline and sc-SynO oversampled trained model for an additional 300,000 cells of the Allen Brain Atlas dataset. A manual analysis of the dataset served as the ground truth and resulted in ~1,500 cells of the target cluster 119_Pvalb Vipr2. Using the top 10 features, the baseline model identified 388 cells correctly in comparison to 491 correctly assigned cells of sc-SynO (375 cells in common). In addition, sc-SynO mis-classified around 200 fewer cells in comparison to the baseline model (915 vs. 734 cells).

5.4 Importance and applicability of sc-SynO

sc-SynO tool is the first oversampling approach to identify and annotate rare-cell populations from scRNA-Seq and snRNA-Seq data. The LoRAS algorithm has been benchmarked against other popular oversampling techniques like SMOTE or Borderline-SMOTE, and presents LoRAS as the underlying algorithm of sc-SynO as a robust algorithm for a broad set of applications in terms of F1-score and balanced accuracy [Bej 2021]. For the baseline models trained without oversampling, I observe a clear limitation on the validation datasets. Since the identification of rare cells in new unseen data is a key requirement, a high recall, as obtained from oversampling approaches, would be essential. Moreover, oversampling with sc-SynO produces comparatively balanced ML model performances on average, in the sense that, in most cases, the algorithm produces less mis-classifications on the majority class with a reasonably small compromise for mis-classifications on the minority class. This is why I would suggest using the algorithm for rare-cell detection instead of the baseline model without oversampling.

I also investigated the similarity of the original cells in comparison to the synthetic cells. I visualized the results using UMAP plots represented in Figure 5.3. I noticed that plotting the synthetic samples of the minority class data space along with the synthetic samples looks stretched. I assume this phenomenon can be explained by investigating the mathematical assumptions underlying the UMAP algorithm. Note that, the full name of UMAP is 'Uniform Manifold Approximation and Projection'. The word 'Uniform' is essential in this context. The UMAP algorithm relies on building a neighbourhood graph in the process of clustering, and the basic assumption behind this construction is the *uniformly* distributed data over the whole data space, even though it is not the reality. Now what happens when one oversamples on a very small population is that, within a very small volume of the data space, one can synthetically generate numerous data points. Whereas, during clustering,

one can assume that they are all uniformly distributed. Since there are a lot of data points, congested in a small volume, under the assumption of uniform distribution, this volume in the space itself stretches. That is why, in the 2-D plot, the distribution of synthetic data points looks stretched and spread over a lot of the space. In summary, with a higher density of points in a data subspace, it will be more stretched to satisfy the assumption of uniformity. Thus, if one ignores the stretching effect caused by UMAP due to a high density of synthetic samples in a small data neighbourhood, one can observe that the synthetic samples are indeed quite similar to the original minority class samples as shown in Figure 5.3.

sc-SynO facilitates the identification of very similar cells for smaller sets of feature genes and biologically related cells for larger sets of genes. The initial clustering of the training data plays an essential role, in which I observed that smaller clusters with a distinct border to other clusters are better suited for an analysis in comparison to larger cell populations with transient borders. However, the algorithm still has high accuracies in identifying those cells.

In comparison to other current tools, such as cscGAN [Marouf 2018], scANVI [Xu 2021], MARS [Brbi 2020], FiRE [Jindal 2018], and ELSA [Wang 2009], sc-SynO uses synthetic oversampling of previously, manually curated cell populations to identify such rare cells of interest in novel unseen data. In addition, sc-SynO is easily applicable and only requires a single, well-curated dataset of any size, including only a few cells of interest, to be able to achieve already high predictive accuracy. Likewise, sc-SynO can be used on integrated datasets (scRNA-Seq and snRNA-Seq) as well, which commonly represent the underlying biological heterogeneity of the sample in an improved manner [Wolfien *Cardiovascular Research* 2020].

sc-SynO can be seamlessly incorporated in any single-cell and single-nuclei data analysis workflow after the identification and annotation of cell populations on raw or normalized read count data and important transcripts per cluster. As a potential perspective, it might be even possible to generate synthetic cells/samples out of homogeneous bulk-RNA-Seq data by treating a single sample as one single cell that needs to be identified within a single-cell dataset. Once a rare-cell population has been identified and carefully checked by a domain expert, sc-SynO can be used on this highly curated dataset to train the specific cell type. Based on experience with the application of oversampling models, one would see individual models for single rare-cell types as the preferred solution, rather than embedding multiple minority class cell types in one data augmentation model. Applying sc-SynO on a novel dataset to identify the same rare-cell type is magnitude less time-consuming than manually curated data processing and annotation of scRNA-Seq data. This facilitated cell enrichment can be used for more in-depth downstream analyses on the cell type of interest, without reanalysing all datasets. Such a scenario can be of high interest for single-cell identification in cancer, hypothesis testing on larger cell sets, cell homology search across tissues, or further individual applications.

All computational scripts can be obtained from the FairdomHub instance (`https://fairdomhub.org/assays/1368`), or the algorithm itself (`https://github.com/narek-davtyan/LoRAS`), and the current integration for sc-SynO on GitHub (`https://github.com/COSPOV/sc-SynO`). Single-cell RNA-Seq data utilized during

this study are already publicly available at the Single Cell Expression Atlas via ArrayExpress (E-MTAB-7869, E-MTAB-8751, E-MTAB-8848), the Allen Brain Atlas (https://celltypes.brain-map.org/), and GEO (GSE130710).

Chapter 6

Classifier-independent oversampling using the ProWRAS algorithm

This chapter presents the ProWRAS algorithm, a multi-schematic and classifier-independent extension of the LoRAS algorithm. Firstly, the chapter discusses an independent benchmarking study showing that performance of oversampling algorithms are indeed classifier dependent. Given numerous oversampling algorithms and classifiers to choose from, a proper choice of an oversampling algorithm and classifier for an imbalanced dataset is challenging. Analysing the philosophies of several oversampling algorithms, it is possible to identify some generic oversampling schemes followed by common oversampling algorithms. The ProWRAS algorithm is designed to integrate such oversampling schemes under a single umbrella. I demonstrate through rigorous benchmarking studies that the ProWRAS algorithm, with proper choice of parameters, can adapt to classifier specific oversampling schemes and thereby perform in a classifier-independent way.

6.1 Classifier dependence of oversampling models

6.1.1 Study protocols

For the pilot study and the final benchmarking study, two sets of public datasets were selected. The first set (Set-I) is a subset of the 104 publicly available imbalanced datasets used for benchmarking studies in Kovács *et al.* [Kovács 2019]. The second set (Set-II), is a subset of 27 publicly available datasets in the `imblearn.datasets` Python library. The datasets in Set-I and Set-II are selected, based on the following three criteria. Set-I has 14 datasets and Set-II has 6 datasets. Note that all datasets that follow all the three criteria are selected to ensure that the choice of datasets for the studies are impartial. The criteria are:

- Datasets with an imbalance ratio of at least 15 : 1 were chosen. This is to ensure that the performances of the compared oversampling algorithms are tested, particularly on datasets with high imbalance.
- Datasets with a minimum of 35 minority class samples were chosen. Datasets with very fewer minority class samples, classifier performances that are often affected by high stochasticity and the results are often statistically unreliable. Setting this condition for the choice of datasets enhances the reliability of results.

- Datasets with at most 5000 samples (only 4 datasets do not satisfy this condition) were chosen. Given that the studies involve 4 classifiers, 8-different oversampling models and 4-oversampling schemes of the ProWRAS algorithm, this constraint is considered to limit the computational effort.

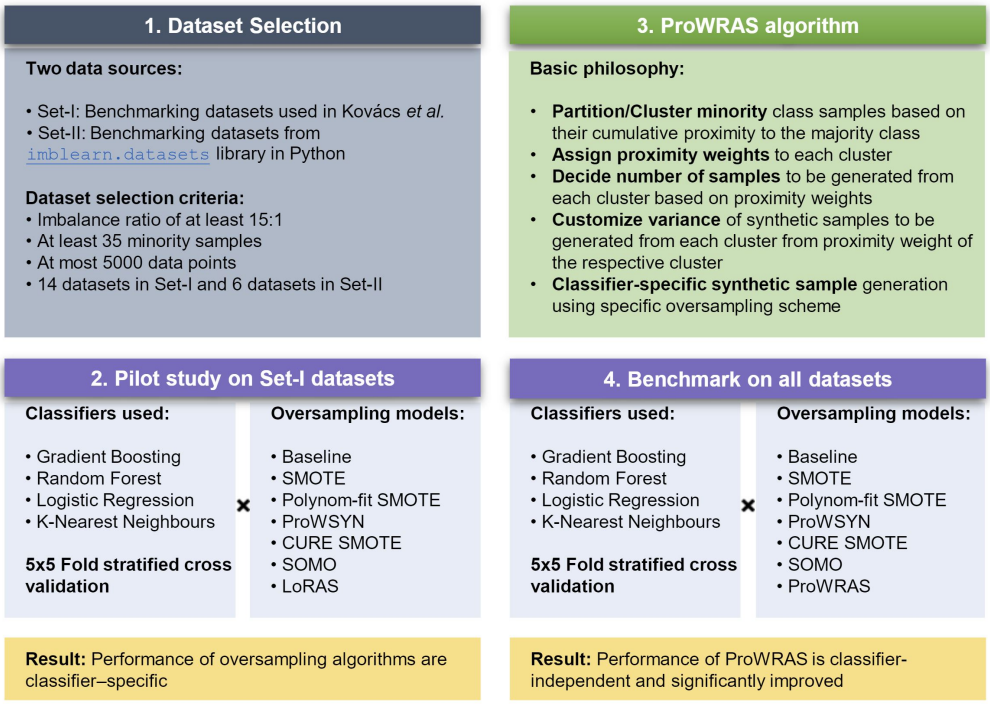


FIGURE 6.1: Illustration of the working principle of the ProWRAS algorithm. ProWRAS used a proximity based partitioning system to find clusters in the minority class. For each cluster, it then uses one of four oversampling schemes shown in the figure.

Choosing classification models and parameters: The classification models GB, RF, kNN, and LR were used for the benchmarking studies. GB and RF were used, since they are powerful classifiers that use ensemble approaches of boosting and bagging, respectively. kNN and LR were also seen to perform well for imbalanced datasets in the benchmarking study of Bej *et al.* [Bej 2021]. Both in the pilot study and in the final benchmarking study, default parameters were used for all the classifiers, as recommended in `scikit-learn (V 0.21.2)` documentation.

Choosing oversampling models and parameters: In the final benchmarking study, five benchmarking algorithms are compared against ProWRAS. These algorithms were chosen in particular for the following reasons:

- SMOTE, of course, is the pioneer of all algorithms and still, widely used because of its simplicity and applicability.
- Polynom-fit SMOTE, ProWSyn are the top two oversampling algorithms by overall performance, from the detailed benchmarking study by Kovács [Kovács 2019].
- ProWSyn, CURE-SMOTE, and SOMO also use the idea of using clustering approaches on the minority class to learn the distribution of the minority class better and take

advantage of it during synthetic sample generation, a philosophy they share with ProWRAS algorithm. Moreover, both CURE-SMOTE and SOMO are proposed fairly recently(in 2017). SOMO has been used only in the pilot study. It was excluded in the main study because of its low $\mathscr{J}$-score in the pilot experiment (See Table 6.6).

- I chose LoRAS because, evidently, ProWRAS is an extension of the LoRAS algorithm.

For the pilot study, default parameters for SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, and SOMO algorithms were used. For LoRAS, parameter values of $\texttt{k} = 5$, $\texttt{|S\_p|} = 100$, $\texttt{L}_\sigma = 5 \times 10^{-8}$ and regular `embedding` were implemented. For the parameter $\texttt{L}_\sigma$, random search among the values $\{2, 10, 30, \texttt{dim(data)})\}$ were performed. This random search is based on a training and testing done on a randomly chosen 250 percent and 20 percent mutually disjoint subset of the respective dataset, chosen such that the imbalance ratio is maintained in the randomly chosen subsets.

For the final benchmarking study, default parameters were used for SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, and LoRAS algorithms. For ProWRAS parameter values of $\texttt{max\_levels} = 5$, $\texttt{n\_neighbours\_max} = 5$, $\texttt{num\_samples\_to\_generate} = |C_{\text{max}}| - |C_{\text{min}}|$, $\theta = 1$, $\texttt{shadow} = 100$ and $\sigma = 10^{-6}$. To access the four oversampling schemes four combinations of values for the parameters `max_conv`, `neb_conv` were considered.

- High global variance (HGV)
  ($\texttt{max\_conv} = 2, \texttt{neb\_conv} = 1000$)
- Low global variance (LGV)
  ($\texttt{max\_conv} = \texttt{dim(data)}, \texttt{neb\_conv} = 1000$)
- High local variance (HLV)
  ($\texttt{max\_conv} = 2, \texttt{neb\_conv} = 5$)
- Low local variance (LLV)
  ($\texttt{max\_conv} = \texttt{dim(data)}, \texttt{neb\_conv} = 5$)

Table 6.1, presents for every classifier and every dataset, which oversampling scheme was used by ProWRAS to obtain the best performance. Note that, as previously discussed, for accessing the global oversampling scheme for a certain cluster, `neb_conv` must be at least the size of the cluster. Since all the chosen datasets have minority class size of less than 1000, a choice of $\texttt{neb\_conv} = 1000$ to access the global oversampling scheme works for all datasets.

Performance measures: Choice of performance measures are an important aspect of studies with imbalanced datasets. For the study, two performance measures, F1-Score and Cohen's κ -Score, were used. F1-Score is the harmonic mean of precision and recall and is a good measure for how good the classification is for the minority class. Given a classification problem, the κ measure is formally defined as:

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

where, P_o is the measure of the observed agreement among the chosen classifier and the ground truths of the classification problem. P_e is the measure of agreement by chance,

among a chosen classifier and the ground truths of the classification problem. The κ-Score gives a quantification of how good the classification is considering both the majority and the minority class.

Quantification of classifier independence: The basis of this work is the observation that the performance of existing variants of the well-known SMOTE oversampling method for imbalanced classification problems are "classifier dependent". This is quite natural, as it is widely appreciated that, for machine learning, no single best method will exist with respect to all possible classification problems (the so-called no free-lunch theorem). Analogously, it is unlikely that there is a single best oversampling scheme, over all possible classifiers. However, this still poses problems. Since there are more than a hundred of such SMOTE variants and tens of ML based classifiers, given an imbalanced dataset, it is difficult to choose an appropriate oversampling algorithm from such a large pool of algorithms. I address this problem with the development of an oversampling approach, which offers good classification performance on an average, irrespective of the classifier used. In practice, this avoids laborious benchmarking experiments on numerous oversampling algorithms.

Some ambiguity may arise regarding the term 'classifier independence' since an oversampling algorithm that always leads to worse than the others, could also be considered classifier independent, if its ranking is 'stably' low. To establish some formalization about the term 'classifier independence', I therefore propose a quantitative measure for classifier independence here.

Definition 13. *Given a set of oversampling algorithms O, a set of classifiers C and a set of benchmarking datasets D, for a given oversampling algorithm $o \in O$, classifier independence of o is defined as,*

$$\mathscr{I}(o) = \sqrt[|C|]{\prod_{c \in C} \left(\frac{1}{|O| - 1} \sum_{\substack{o' \in O \\ o' \neq o}} \frac{\mathscr{F}(o', o)}{|D|} \right)} \tag{6.1}$$

where, $\mathscr{F}(o', o)$, denotes the number of datasets for which the oversampling algorithm $o \in O$, performs equally or better than another oversampling algorithm $o' \in O$.

In other words to measure the classifier independence of an oversampling algorithm o relative to some other oversampling algorithms, given a set of datasets and a set of classifiers, one can calculate for each classifier, the average proportion of datasets for which o outperforms other oversampling algorithms. When one calculates this for all classifiers, one can take the geometric mean over all classifiers to obtain $\mathscr{I}(o)$. Note that, I choose a geometric mean over all classifiers in Equation 6.1, to keep the measure strictly sensitive to classifier-specific performance of the oversampling algorithms, since the geometric mean is always less than the arithmetic mean. The value of $\mathscr{I}$, will always range between 0 and 1, making it a conveniently interpretable measure. $\mathscr{I}$ thus, measures not only how consistent the performance of an oversampling algorithm is over a set of classifiers, but also how well the oversampling model performs compared to other such models, overall. I, however, would like to emphasize that, while I do provide a measure of classifier-independence,

the conclusion will nevertheless be empirical, dependent on the datasets, oversampling algorithms, and classifiers used.

For coding, I used the `scikit-learn (V 0.21.2)`, `numpy (V 1.16.4)`, `pandas (V 0.24.2)`, and `matplotlib (V 3.1.0)` libraries in `Python (V 3.7.4)`.

Table 6.1 shows the statistics for the relevant datasets. The datasets of Set-I are used for the pilot study, comparing Baseline classification with oversampling models SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, SOMO, and LoRAS for four respective classifiers GB, RF, kNN, and LR. For the final benchmarking study, all 20 datasets from Set I and Set II were utilised, comparing Baseline classification with oversampling models SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, LoRAS, and ProWRAS for the four classifiers GB, RF, kNN, and LR. For both the pilot study and for the main benchmarking study, 5×5 stratified cross-validation as a validation protocol was implemented. While training models, oversampling algorithms were applied only on the training data for each fold. Also, normalised datasets were used for training and testing.

TABLE 6.1: Table showing the ProWRAS oversampling scheme used for every dataset and for every classifier. HGV: High global variance, LGV: Low global variance, HLV: High local variance, LLV: Low local variance. Column 2-5 show the oversampling scheme for which ProWRAS works best for respective datasets and classifiers. Furthermore, the table shows some statistics for the datasets. The last six datasets form Set II.

Dataset	GB	RF	kNN	LR	Imbalance ratio	Minority samples	Total samples
abalone9-18	HGV	HGV	LGV	HGV	16.40	42	731
abalone_17_vs_7_8_9_10	HGV	HGV	LGV	LLV	39.31	58	2338
car-vgood	LLV	HLV	LGV	LGV	25.58	65	1728
car_good	LGV	HLV	LLV	LGV	24.04	69	1728
flare-F	HGV	HGV	LLV	LGV	23.79	43	1066
hypothyroid	LGV	LGV	LLV	LGV	19.95	151	3163
kddcup-guess_passwd_vs_satan	HLV	HGV	LLV	LGV	29.98	53	1642
kr-vs-k-three_vs_eleven	LGV	HGV	LGV	LGV	35.23	81	2935
kr-vs-k-zero-one_vs_draw	LGV	LGV	LGV	LGV	26.63	105	2901
shuttle-2_vs_5	HGV	HGV	LLV	LGV	66.67	49	3316
winequality-red-4	HLV	HGV	HLV	LGV	29.17	53	1599
yeast4	HLV	HGV	LGV	LGV	28.10	51	1484
yeast5	HLV	HGV	LGV	LLV	32.73	44	1484
yeast6	LGV	HLV	LGV	LLV	41.40	35	1484
oil	HGV	HGV	LGV	LGV	22.85	41	937
ozone_level	HGV	HGV	LLV	LGV	34.73	73	2536
solar_flare_m0	HLV	HLV	LLV	LGV	20.42	68	1389
thyroid_sick	LGV	HLV	LLV	LGV	16.32	231	3772
wine_quality	HLV	HLV	HLV	LGV	26.76	183	4898
yeast_me2	LLV	HLV	LLV	LGV	29.09	51	1484

6.1.2 Pilot study confirming classifier dependence of oversampling

Figure 6.2, shows a comparative plot for the performance by F1-Score of all classifiers over all oversampling models. Figure 6.2 has a heatmap for every classifier. For a given classifier, the number in an arbitrary cell in the i-th row and j-th column, of the heatmap shows the number of datasets for which the oversampling model corresponding to the i-th row performs equally or better than the oversampling model corresponding to the j-th column. Since the pilot study was performed on only the datasets of Set-I (with 14 datasets), all diagonal elements are 14. Notably, the performance by the κ -Score follows a similar trend.

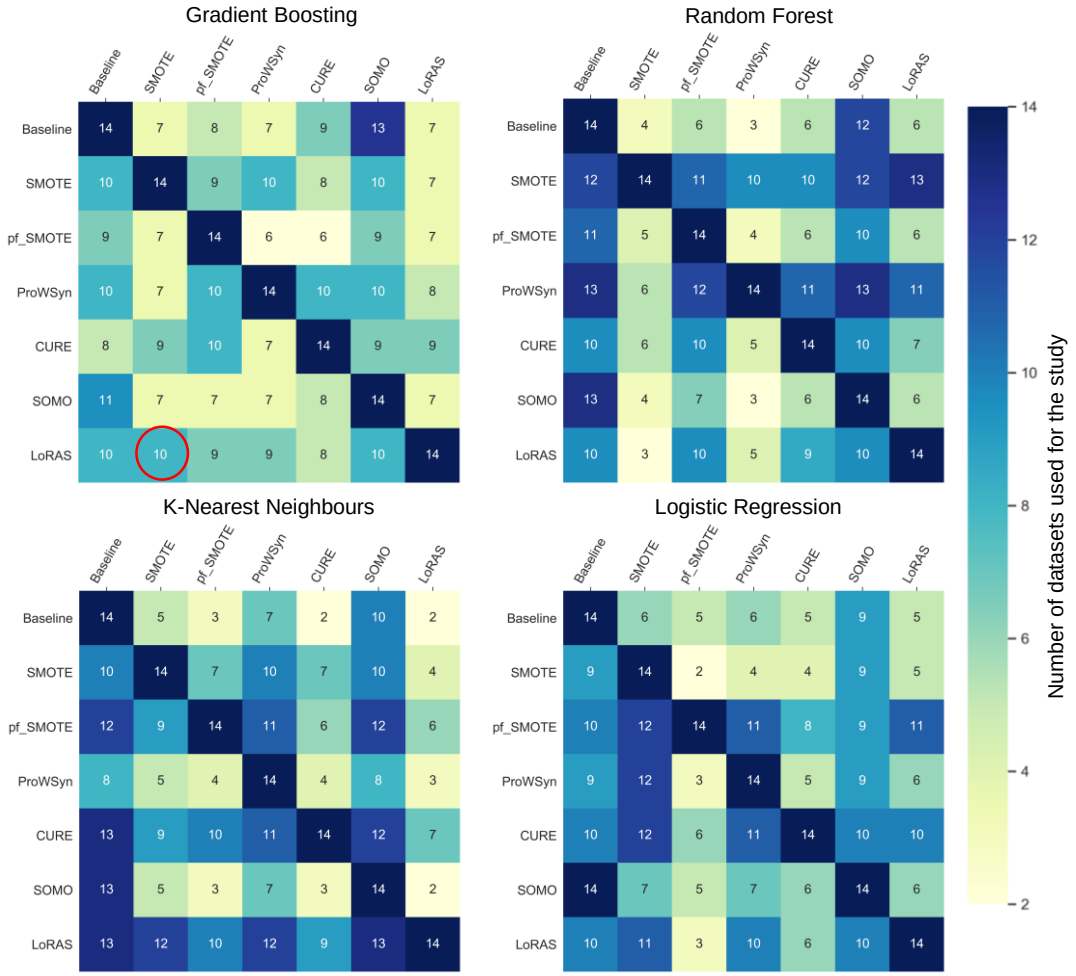


FIGURE 6.2: Figure showing results for the pilot study. Every heatmap for a respective classifier shows the number of datasets for which the oversampling model in the i-th row performs equally or better (by F1-Score) than the model in the j-th column. For example, for the gradient boosting classifier LoRAS performs equally or better than SMOTE for 10 out of 14 datasets. Note that, none of the oversampling models perform consistently well for all the classifiers.

Observe from Figure 6.2, that for kNN classifier, LoRAS, CURE SMOTE, and Polynom-fit SMOTE are the best performers. For LR classifier, CURE SMOTE and Polynom-fit SMOTE are ahead of the other oversampling models. For the RF classifier, SMOTE and ProWSyn are the best performers. For GB classifier, ProWSyn, LoRAS, and SMOTE generate better F1-Scores.

Moreover, observe that the average F1-Score and κ-Score for all classifiers is comparatively better for the ensemble based classifiers RF and GB.

I also provide the $\mathscr{I}$-score for every oversampling model used in the pilot study in Table 6.6. Observe that, CURE-SMOTE, Polynom-fit SMOTE and LoRAS produce the best scores, while the score of SOMO is quite close to the baseline. Thus, I excluded the SOMO algorithm in the final benchmarking studies.

TABLE 6.2: Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, SOMO, LoRAS) for 14 Set-I benchmarking datasets for GB classifier.

Datasets	Baseline	SMOTE	Polynom-fit SMOTE	ProWSyn	CURE	SOMO	LORAS
abalone9-18	0.394/0.367	0.408/0.362	0.3/0.275	0.39/0.345	0.38/0.336	0.394/0.367	0.441/0.401
abalone_17_vs_7_8_9_10	0.283/0.272	0.318/0.292	0.309/0.298	0.349/0.324	0.346/0.323	0.283/0.272	0.324/0.298
car-vgood	0.968/0.967	0.957/0.956	0.968/0.967	0.975/0.974	0.966/0.964	0.963/0.961	0.985/0.984
car_good	0.904/0.9	0.813/0.805	0.86/0.855	0.839/0.832	0.867/0.862	0.893/0.888	0.867/0.862
flare-F	0.14/0.121	0.315/0.281	0.197/0.181	0.281/0.254	0.154/0.132	0.164/0.143	0.124/0.099
hypothyroid	0.806/0.797	0.781/0.769	0.81/0.801	0.776/0.763	0.792/0.782	0.806/0.797	0.793/0.782
kddcup-guess_passwd_vs_satan	1.0/1.0	1.0/1.0	0.994/0.994	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
kr-vs-k-three_vs_eleven	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
kr-vs-k-zero-one_vs_draw	0.979/0.979	0.956/0.954	0.973/0.972	0.97/0.969	0.977/0.977	0.975/0.974	0.967/0.966
shuttle-2_vs_5	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
winequality-red-4	0.048/0.033	0.156/0.115	0.083/0.046	0.141/0.096	0.11/0.076	0.048/0.033	0.08/0.043
yeast4	0.347/0.332	0.399/0.37	0.309/0.292	0.372/0.34	0.291/0.27	0.347/0.332	0.386/0.361
yeast5	0.625/0.615	0.738/0.729	0.685/0.676	0.713/0.704	0.691/0.682	0.625/0.615	0.708/0.699
yeast6	0.436/0.426	0.481/0.465	0.531/0.521	0.46/0.442	0.531/0.521	0.436/0.426	0.486/0.472
Average	0.638/0.629	0.666/0.65	0.644/0.634	0.662/0.646	0.65/0.637	0.638/0.629	0.654/0.641

TABLE 6.3: Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, SOMO, LoRAS) for 14 Set-I benchmarking datasets for RF classifier.

Datasets	Baseline	SMOTE	Polynom-fit SMOTE	ProWSyn	CURE	SOMO	LORAS
abalone9-18	0.28/0.449	0.389/0.347	0.313/0.29	0.336/0.287	0.336/0.298	0.28/0.267	0.368/0.331
abalone_17_vs_7_8_9_10	0.106/0.353	0.339/0.319	0.167/0.157	0.32/0.296	0.278/0.261	0.106/0.103	0.31/0.29
car-vgood	0.969/0.995	0.988/0.988	0.954/0.952	0.974/0.973	0.947/0.945	0.969/0.967	0.932/0.93
car_good	0.795/0.964	0.861/0.856	0.734/0.727	0.817/0.81	0.66/0.651	0.797/0.791	0.664/0.654
flare-F	0.08/0.232	0.182/0.151	0.025/0.003	0.196/0.169	0.045/0.024	0.027/0.004	0.086/0.06
hypothyroid	0.789/0.862	0.77/0.758	0.789/0.78	0.77/0.758	0.773/0.763	0.789/0.779	0.758/0.748
kddcup-guess_passwd_vs_satan	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
kr-vs-k-three_vs_eleven	0.992/1.0	0.997/0.997	0.992/0.992	0.995/0.995	0.999/0.999	0.992/0.992	0.995/0.995
kr-vs-k-zero-one_vs_draw	0.95/0.997	0.943/0.941	0.95/0.948	0.955/0.953	0.951/0.949	0.953/0.951	0.939/0.937
shuttle-2_vs_5	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
winequality-red-4	0.0/0.172	0.143/0.115	0.031/0.01	0.164/0.127	0.054/0.037	0.0/-0.0	0.07/0.046
yeast4	0.23/0.408	0.413/0.391	0.264/0.252	0.372/0.342	0.287/0.27	0.23/0.219	0.341/0.319
yeast5	0.637/0.752	0.724/0.715	0.707/0.699	0.728/0.719	0.689/0.68	0.637/0.629	0.715/0.705
yeast6	0.428/0.53	0.493/0.481	0.496/0.489	0.46/0.445	0.528/0.519	0.428/0.42	0.504/0.494
Average	0.59/0.694	0.66/0.647	0.602/0.593	0.649/0.634	0.61/0.6	0.586/0.58	0.62/0.608

TABLE 6.4: Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, SOMO, LoRAS) for 14 Set-I benchmarking datasets for kNN classifier.

Datasets	Baseline	SMOTE	Polynom-fit SMOTE	ProWSyn	CURE	SOMO	LORAS
abalone9-18	0.125/0.119	0.345/0.286	0.368/0.345	0.404/0.356	0.332/0.279	0.125/0.119	0.394/0.346
abalone_17_vs_7_8_9_10	0.11/0.105	0.295/0.267	0.316/0.3	0.335/0.309	0.286/0.259	0.11/0.105	0.298/0.271
car-vgood	0.594/0.585	0.88/0.875	0.84/0.833	0.836/0.828	0.84/0.834	0.583/0.574	0.898/0.894
car_good	0.409/0.399	0.857/0.851	0.572/0.546	0.509/0.478	0.737/0.725	0.423/0.413	0.84/0.832
flare-F	0.095/0.079	0.28/0.234	0.28/0.234	0.275/0.227	0.305/0.265	0.1/0.083	0.293/0.251
hypothyroid	0.61/0.595	0.589/0.562	0.646/0.625	0.578/0.551	0.668/0.651	0.61/0.595	0.619/0.595
kddcup-guess_passwd_vs_satan	0.99/0.99	0.99/0.99	0.99/0.99	0.99/0.99	0.99/0.99	0.99/0.99	0.99/0.99
kr-vs-k-three_vs_eleven	0.936/0.935	0.949/0.948	0.949/0.948	0.936/0.934	0.937/0.935	0.939/0.937	0.945/0.943
kr-vs-k-zero-one_vs_draw	0.892/0.889	0.879/0.873	0.904/0.9	0.879/0.874	0.92/0.917	0.898/0.895	0.91/0.906
shuttle-2_vs_5	0.998/0.998	1.0/1.0	1.0/1.0	0.992/0.991	1.0/1.0	0.998/0.998	1.0/1.0
winequality-red-4	0.0/-0.001	0.066/0.009	0.054/-0.005	0.074/0.015	0.073/0.016	0.0/-0.001	0.078/0.023
yeast4	0.139/0.126	0.323/0.285	0.329/0.292	0.292/0.25	0.357/0.325	0.139/0.126	0.383/0.353
yeast5	0.679/0.67	0.67/0.656	0.67/0.657	0.627/0.611	0.688/0.676	0.679/0.67	0.686/0.673
yeast6	0.562/0.553	0.332/0.306	0.367/0.343	0.307/0.28	0.49/0.474	0.562/0.553	0.361/0.338
Average	0.51/0.503	0.604/0.582	0.592/0.572	0.574/0.55	0.616/0.596	0.511/0.504	0.621/0.601

TABLE 6.5: Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, SOMO, LoRAS) for 14 Set-I benchmarking datasets for LR classifier.

Datasets	Baseline	SMOTE	Polynom-fit SMOTE	ProWSyn	CURE	SOMO	LORAS
abalone9-18	0.559/0.694	0.524/0.485	0.57/0.538	0.518/0.477	0.563/0.529	0.559/0.538	0.569/0.536
abalone_17_vs_7_8_9_10	0.357/0.436	0.307/0.278	0.363/0.339	0.323/0.295	0.342/0.315	0.357/0.347	0.324/0.297
car-vgood	0.122/0.353	0.377/0.337	0.397/0.36	0.381/0.342	0.368/0.327	0.132/0.117	0.385/0.346
car_good	0.0/0.078	0.095/0.024	0.103/0.033	0.099/0.028	0.108/0.039	0.022/0.012	0.094/0.024
flare-F	0.208/0.352	0.263/0.212	0.329/0.287	0.265/0.214	0.273/0.224	0.252/0.23	0.28/0.23
hypothyroid	0.335/0.444	0.368/0.317	0.401/0.355	0.381/0.331	0.388/0.343	0.335/0.316	0.376/0.326
kddcup-guess_passwd_vs_satan	0.996/0.989	1.0/1.0	0.996/0.996	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
kr-vs-k-three_vs_eleven	0.96/0.997	0.947/0.946	0.948/0.947	0.953/0.951	0.949/0.948	0.96/0.959	0.946/0.944
kr-vs-k-zero-one_vs_draw	0.853/0.934	0.732/0.719	0.795/0.786	0.74/0.728	0.766/0.756	0.862/0.857	0.764/0.753
shuttle-2_vs_5	1.0/1.0	1.0/1.0	0.983/0.983	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
winequality-red-4	0.007/0.155	0.127/0.072	0.132/0.078	0.129/0.074	0.157/0.106	0.007/0.004	0.138/0.084
yeast4	0.212/0.415	0.255/0.211	0.275/0.233	0.259/0.215	0.247/0.203	0.212/0.2	0.26/0.217
yeast5	0.552/0.709	0.599/0.582	0.631/0.616	0.607/0.59	0.622/0.607	0.552/0.541	0.598/0.581
yeast6	0.437/0.512	0.3/0.272	0.345/0.32	0.295/0.266	0.398/0.377	0.437/0.428	0.323/0.296
Average	0.471/0.576	0.492/0.461	0.519/0.491	0.496/0.465	0.513/0.484	0.478/0.468	0.504/0.474

TABLE 6.6: Table showing the $\mathscr{I}$-scores for different oversampling algorithms for the pilot study.

	Baseline	SMOTE	Polynom-fit SMOTE	ProWSyn	CURE-SMOTE	SOMO	LoRAS
$\mathscr{I}$	0.442	0.58	0.592	0.562	**0.649**	0.478	0.647

6.2 ProWRAS algorithm

Recall that the **LoRAS** oversampling approach proposes to model the convex space more rigorously. Instead of generating synthetic samples by taking a convex combination of only two samples from a neighbourhood (as done by SMOTE and a majority of its extensions), LoRAS proposes to generate synthetic samples by taking convex combinations of multiple shadowsamples (Gaussian noise added to the original minority class samples) in a minority class data neighbourhood [Bej 2021]. Bej *et al.* analytically calculates the variance of a LoRAS-generated synthetic sample (considered as a random variable) as:

$$\text{Var}(L_j) = \frac{2(\sigma_j'^2 + \sigma_{Bj}^2)}{(|F| + 1)} \tag{6.2}$$

where, L_j is the j-th component of a LoRAS-generated sample L, $|F|$ is the number of shadowsamples considered for a convex combination to generate L, $\sigma_j'^2$ is the original variance of the minority class samples in a neighbourhood and σ_{Bj}^2 is the variance of the noise added to the original minority class samples to generate shadowsamples (σ_{Bj}^2 can be chosen to be arbitrarily small) [Bej 2021]. Moreover, the LoRAS algorithm uses manifold learning technique t-SNE to learn the minority class data neighbourhoods.

Algorithm 3 ProWRAS oversampling algorithm (GitHub link)

Inputs:

`data` Data points.

Parameters:

`max_conv` (> 0) Weight for number of generated samples per layer.
`num_samples_to_generate` (> 0) Maximal count of generated samples in the output.

Function `ProWRAS_oversampling(data)` **begin**
  `clusters` $\leftarrow$ `partition_info(data)` (See Algorithm 4)
  `weight_max` $\leftarrow$ $max(\{$`weight` : (`cluster`, `weight`) $\in$ `clusters`$\})$
  Initialize *synth_samples* with an empty set.
  For (`cluster, weight`) $\in$ `clusters` **do**
    `num_samples` $\leftarrow \lceil$ `num_samples_to_generate` $\cdot$ `weight` $\rceil$
    `num_convcomb` $\leftarrow \left\lceil \frac{\texttt{max\_conv}\cdot\texttt{weight}}{\texttt{weight\_max}} \right\rceil$
    `synth` $\leftarrow$ `generate_points(cluster, num_samples, num_convcomb)` (See Algorithm 5)
    `synth_samples` $\leftarrow$ `synth_samples` $\cup$ `synth`
  endfor
  Return resulting set of generated data points as `synth_samples`.
end

The ProWRAS algorithm is a multi-schematic oversampling algorithm, which integrates several aspects of the LoRAS and ProWSyn algorithms. The ProWRAS algorithm can be realised by the following steps:

Partition/Cluster minority class samples: The algorithm takes labelled imbalanced data as input. As a first step, it creates a partition of the minority class. The partition is

Algorithm 4 Proximity weighted minority class data partitioning

Inputs:

`data`	Data points.

Parameters:

`max_levels`	(≥ 1)	Maximal repeat of bordersearch.
`n_neighbours_max`	(≥ 1)	Number of neighbours considered for the majority class data points while constructing minority class partitions.
θ	(> 0)	Scaling for weights.
`num_feats`	$(= dim(x_1))$	Number of features.

Function `partition_info(data)` **begin**

- `X_maj` $\leftarrow$ Data points in `data` with label for major class.
- `X_min` $\leftarrow$ Data points in X with label for minor class.
- $L =$ `max_levels`
- Initialize `clusters` as empty set.
- **For** $i = 1, 2, \ldots, L-1$ **do**
  - **If** $|$`X_min`$| = 0$ **then**
    - break
  - **endif**
  - `weight` $= \exp(-\theta \cdot (i-1))$
  - $k \leftarrow \min(|$`X_min`$|,$ `n_neighbours_max`$])$
  - `cluster` $\leftarrow$ All neighbours in k-Neighbourhoods from `X_maj` in `X_min`
  - `clusters` $=$ `clusters` $\cup \{($`cluster`$,$ `weight`$)\}$
  - `X_min` $\leftarrow$ `X_min` $\setminus$ `cluster`
- **endfor**
- **If** $|$`X_min`$| > 0$ **then**
  - `weight` $= \exp(-\theta \cdot (L-1))$
  - `clusters` $=$ `clusters` $\cup \{($`X_min`$,$ `weight`$)\}$
- **endif**
- `weight_sum` $\leftarrow sum(\{$`weight` $:$ `(cluster, weight)` $\in$ `clusters`$\})$
- `clusters` $\leftarrow \left\{\left(\texttt{cluster}, \frac{\texttt{weight}}{\texttt{weight\_sum}}\right) : (\texttt{cluster}, \texttt{weight}) \in \texttt{clusters}\right\}$
- Returns pairs of clusters and normalised weights as `clusters`.

end

Algorithm 5 Cluster-wise oversampling schemes

Inputs:

`cluster`	Data points.
`num_samples`	Number of generated shadowsamples per parent data point.
`num_convcomb`	Number of convex combinations for each new sample.

Parameters:

`neb_conv`	(≥ 1)	Number of data points used in affine combination for new samples.
`shadow`	(≥ 1)	Number of generated shadowsamples per parent data point.
`sigma`	(≥ 0)	List of standard deviations for normal distributions for adding noise to each feature.

Function `generate_points(cluster, num_samples, num_convcomb)` **begin**

- Initialize `generated_data` with empty set.
- **If** $|$`cluster`$| >$ `neb_conv` **then**
  - `neb_list` $\leftarrow$ set of all k-Neighbourhoods in `cluster`
- **else**
  - `neb_list` $\leftarrow \{$`cluster`$\}$
- **endif**
- **If** `num_convcomb` $<$ `num_feats` **then**
  - $k \leftarrow 2$
- **else**
  - $k \leftarrow$ `num_convcomb`
- **endif**
- **For** $i = 1, 2, \ldots$ `num_samples` **do**
  - `neighbourhood` $\leftarrow$ a random neighbourhood in `neb_list`
  - **If** `num_convcomb` $<$ `num_feats` **then**
    - `data_shadow` $\leftarrow$ `neighbourhood`
  - **else**
    - Initialize `data_shadow` with empty set.
    - **For** $v \in$ *neighborurhood* **do**
      - `data_shadow` $\leftarrow$ `data_shadow` $\cup \{$`shadow` random vectors around v with normal distribution. $\}$
    - **endfor**
  - **endif**
  - $u = (u_1, \ldots, u_k) \leftarrow k$ random vectors $\in$ `data_shadow`
  - $w = (w_1, \ldots, w_k) \leftarrow$ a random vector with positive values and $w_1 + w_2 + \ldots + w_k = 1$
  - `generated_data` $\leftarrow$ `generated_data` $\cup \{w \cdot u\}$
- **endfor**
- Returns new points as `generated_data`.

end

done as per the proximity of the minority class data points from the majority class. The maximum number of desired partitions can be predefined by the user using a parameter `max_levels` (recommended value of 5). The first partition P_1 is determined by the union of `n_neighbours_max` (recommended value of 5) number of minority class nearest neighbours of all the majority class data points. The parameter `n_neighbours_max` can also be adjusted by the user. Once the first partition P_1 is ready, the process is repeated for the remaining minority class data points (if any left) that are not in P_1. This procedure is repeated for $L-1$ steps to obtain partitions $P_1, \ldots, P_{L-1}$. The minority class data points that are not included in any of the partitions $P_1, \ldots, P_{L-1}$, form the partition P_L. Thus, for $i < j$, P_i is closer to the majority class compared to P_j, $i, j \in \{1, \ldots, L\}$. The partitions thus formed are treated as clusters in the minority class. This clustering technique is adopted from ProWSyn, a very effective oversampling technique, described in Section 3.4.1 [Barua 2013]. An advantage of this type of partitioning/ clustering of data is that the clustering process considers the distribution of the minority class with respect to the majority class, which is not considered in other clustering algorithms.

Assigning proximity weights to clusters: The next step is to assign proximity weights to each cluster P_i, $i \in \{1, \ldots, L\}$, such that clusters closer to the majority class have more weights. This is done to make the decision boundary stronger as the minority class data points that are closer to the majority class cause more confusion for the classifiers to create a decision boundary. This is achieved for $i \in \{1, \ldots, L\}$, by assigning weight w_i to the cluster P_i, following the equation:

$$w_i = e^{-\theta \cdot (i-1)} \tag{6.3}$$

The weights are then normalised. The parameter θ (recommended value 1) can be used to control the rate of decay of weights. The pseudocode for this process can be found in Algorithm 4.

Deciding the number of synthetic samples to generate from each cluster: After the clusters and their respective normalised weights are obtained, ProWRAS decides the number of synthetic samples to be generated from each cluster. The total number of synthetic samples to be generated is taken as a user input using the parameter `num_samples_to_generate` (a recommended value for this parameter is the difference between the number of majority and minority class samples). The normalised weights of respective clusters are multiplied to `num_samples_to_generate`, to determine the number of samples to be generated from those clusters. Until this point, the ProWRAS algorithm follows the same steps as the ProWSyn algorithm [Barua 2013].

Customise variance for each cluster: Bej *et al.* in the article on the LoRAS algorithm pointed out that customising the variance of the synthetic samples can be important for an improved modelling of the convex space of the minority class. In contrast to the ProWSyn algorithm, the ProWRAS algorithm uses an approach to rigorously model the convex space of the minority class by controlling the variance of the synthetic samples generated. There are two aspects of the algorithm that can help one achieve this, which are described below.

A) Choice of proper neighbourhood size: For each of the identified clusters, ProWRAS can generate synthetic samples using different minority class neighbourhood sizes. This can lead to a local oversample generation scheme by choosing a small minority class

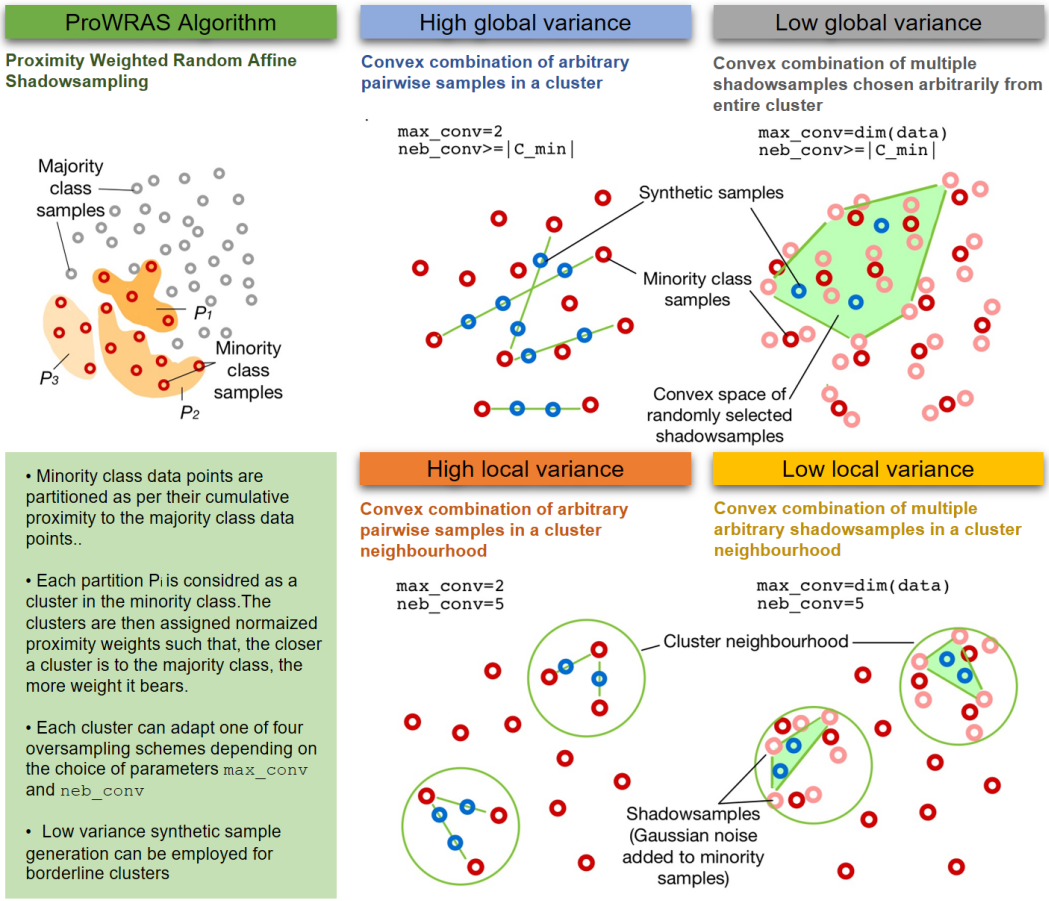


FIGURE 6.3: Illustration of the working principle of the ProWRAS algorithm. ProWRAS used a proximity based partitioning system to find clusters in the minority class. For each cluster, it then uses one of four oversampling schemes shown in the figure. The key to success of the ProWRAS algorithm is its ability to rigorously model the convex space through controlling the variance of the synthetic samples.

neighbourhood (similar to the approach of SMOTE or LoRAS) or a global oversample generation scheme, which considers the entire cluster as a neighbourhood (similar to the approach of CURE-SMOTE). The global or local oversampling schemes can be accessed using proper choice of `neb_conv` parameter. If the choice of `neb_conv` is less than the size of the cluster itself, then ProWRAS will employ a local oversampling scheme, otherwise the whole cluster will be considered as a neighbourhood and the global oversampling scheme is employed.

B) Convex space modelling: ProWRAS can also control the variance of the generated synthetic samples using rigorous convex space modelling. This is achieved using the `max_conv` parameter. If `max_conv` $= 2$, ProWRAS generates SMOTE-like synthetic samples by taking convex combinations of any two minority class samples. Let's call this, a high variance oversampling scheme, since this leads to high variance of the synthetic samples (see Equation 6.2). If `max_conv` > 2, ProWRAS generates LoRAS-like synthetic samples by taking convex combinations of multiple numbers of shadowsamples. LoRAS-like sample generation of course requires two more parameters to be added to the algorithm: σ (recommended value of 0.001), for deciding the variance of the normal distribution to

draw the noise for creating the shadowsamples, `shadow` (recommended value of 100), for deciding how many shadowsamples need to be created per minority class sample. Let's call this a low variance oversampling scheme. The number of convex combinations is decided by the normalised proximity weight of the respective cluster. First, the normalised proximity weights of all clusters are scaled, dividing them by the maximum normalised proximity weight obtained. Note that, this produces scaled weights for every cluster, such that each cluster has a weight between 0 and 1. The scaled weights are then multiplied with the `max_conv` parameter to obtain the number of appropriate convex combinations of shadowsamples for each cluster.

Note that clusters with higher scaled weights are closer to the majority class. Taking a convex combination of multiple samples for such clusters will help to keep the variance of the synthetic samples low, which will prevent them from interfering with the majority class. Clusters with lower scaled weights are far away from the majority class, and hence one can choose to create high variance synthetic samples from them. This also reduces the computational costs of the LoRAS algorithm.

Based on the points mentioned above, one can identify four oversampling schemes for the ProWRAS algorithm that can be accessed by different combinations of the two parameters `max_conv` and `neb_conv`. They are:

- High global variance (HGV)
  (`max_conv` $= 2$, `neb_conv` $\geq |C_{\text{min}}|$)
- Low global variance (LGV)
  (`max_conv` $=$ `dim(data)`, `neb_conv` $\geq |C_{\text{min}}|$)
- High local variance (HLV)
  (`max_conv` $= 2$, `neb_conv` $= 5$)
- Low local variance (LLV)
  (`max_conv` $=$ `dim(data)`, `neb_conv` $= 5$)

where C_{min} is the minority class and `dim(data)` is the number of features in the dataset. Note that the global oversampling scheme is employed for a cluster, if the chosen value of `neb_conv` is greater than the size of the cluster. Choosing the value of `neb_conv` $\geq |C_{\text{min}}|$ ensures that for all clusters the global oversampling scheme is employed. Moreover, if `max_conv` $= 2$ then automatically scales cluster weights to determine the number of convex combinations for the shadowsample generation in any arbitrary cluster becomes 2 (See Algorithm 5), leading ProWRAS into the high variance data generation scheme. A graphical representation of the ProWRAS algorithm is shown in Figure 6.3.

To sum up, the ProWRAS algorithm, which has eight adjustable parameters: `max_levels`, `n_neighbours_max`, `num_samples_to_generate`, θ, σ, `shadow`, `max_conv`, `neb_conv`. Among these, only two parameters `max_conv`, `neb_conv`, affects the oversampling process significantly by changing the use of one of the four oversampling schemes.

Classifier-specific synthetic sampling: Finally, to take full advantage of the ProWRAS algorithm given a dataset and a classifier of choice, a user can choose to train the classifier

using all four oversampling schemes and finally select the best one. In Section 6.3, where I discuss the results of this research, I show that classifier-specific choice of oversampling schemes helps ProWRAS to perform better, independently of the classifier used.

6.3 Classifier independent performance of ProWRAS

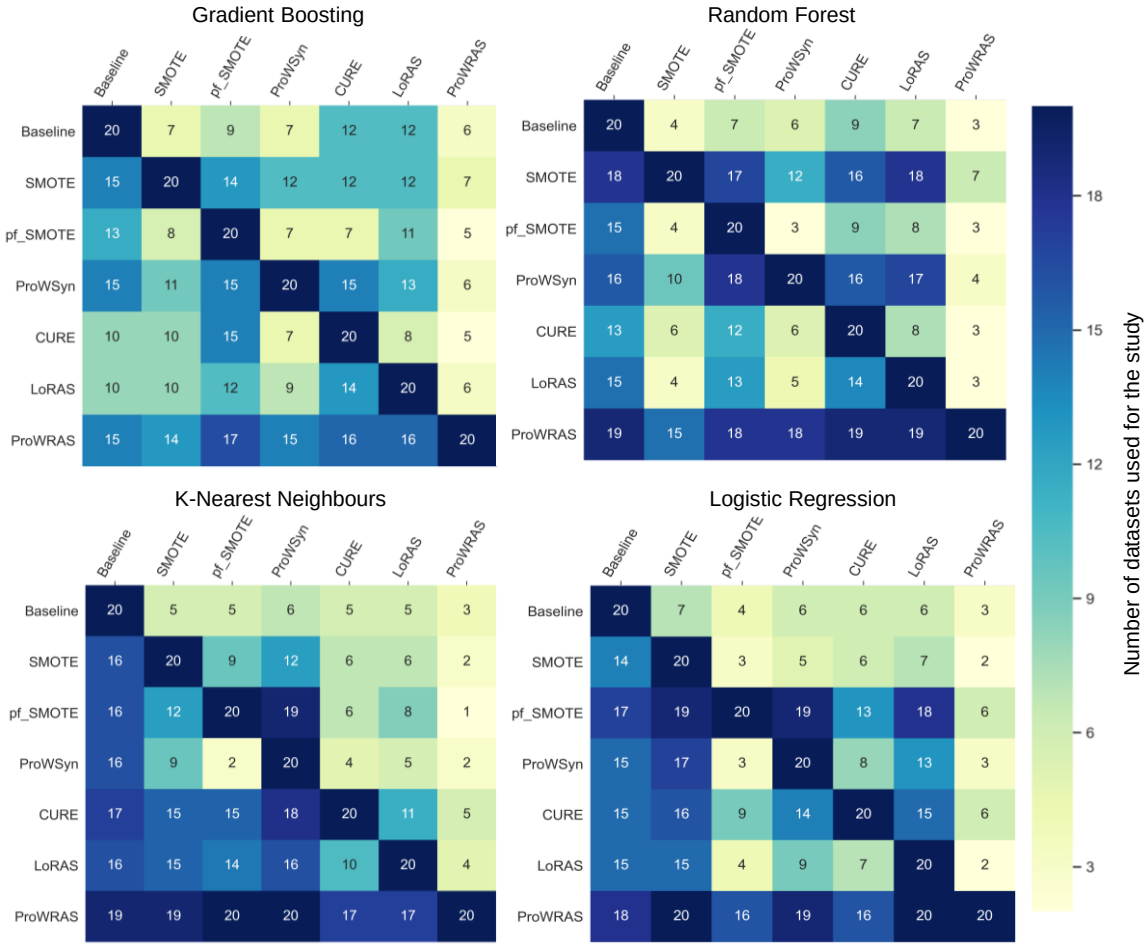


FIGURE 6.4: Figure showing results for the final study. Every heatmap for the respective classifier shows the number of datasets for which the oversampling model in the i-th row performs equally or better (by F1-Score) than the model in the j-th column. Note that, ProWRAS performs consistently well for all the classifiers.

Details of the final benchmarking studies are shown in Table 6.7, 6.8, 6.9, 6.10. Observe that the ensemble models on an average perform quite well in comparison to kNN or LR. Even the baseline model for GB has better F1 and κ-Scores comparing the oversampled classifiers for kNN and LR. For classifiers GB, RF, and LR, ProWRAS produces better average F1-Score and κ-Score over all models. In the case of kNN, CURE SMOTE does marginally better than ProWRAS. Figure 6.4 shows the heatmap plots for each chosen classifier, showing comparisons of the oversampling algorithms among each other in terms of their performance on the 20. Observe that the oversampling models that performed well in the pilot study for the respective classifiers continue to do well. For example,

for RF, SMOTE and ProWSYN still perform quite well. Interestingly, ProWRAS performs consistently well for all the classifiers. Thus, the classifier and dataset specific synthetic data generation of ProWRAS makes its performance classifier-independent.

I have also quantified the classifier independence of the compared oversampling algorithms using the $\mathscr{I}$-score (6.1). I observed that, CURE-SMOTE, LoRAS and ProWSYN still maintains comparatively high degree of classifier independence. However, ProwRAS significantly outperforms other algorithms with a score of 0.833.

Thus, I conclude that, classifier and dataset specific synthetic data generation of ProWRAS makes its performance classifier-independent.

TABLE 6.7: Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, LoRAS, ProWRAS) for all 20 benchmarking datasets for Gradient Boosting classifier. The column on the right shows the performance of the ProWRAS algorithm over all datasets. Observe in the last row that the average performance of ProWRAS is superior to all other oversampling algorithms.

Dataset	Baseline	SMOTE	Polynom-fit SMOTE	ProWSyn	CURE-SMOTE	LoRAS	**ProWRAS**
abalone9-18	0.342/0.319	0.359/0.312	0.259/0.236	0.381/0.338	0.317/0.271	0.319/0.294	0.385/0.341
abalone_17_vs_7_8_9_10	0.277/0.265	0.333/0.308	0.294/0.282	0.359/0.336	0.337/0.314	0.236/0.226	0.335/0.310
car-vgood	0.981/0.980	0.946/0.943	0.966/0.964	0.968/0.967	0.960/0.959	0.968/0.966	0.959/0.957
car_good	0.900/0.896	0.850/0.843	0.819/0.812	0.855/0.849	0.84/0.833	0.868/0.863	0.863/0.857
flare-F	0.172/0.155	0.321/0.287	0.174/0.156	0.271/0.241	0.143/0.123	0.171/0.149	0.320/0.291
hypothyroid	0.805/0.796	0.762/0.748	0.798/0.788	0.776/0.763	0.788/0.778	0.796/0.787	0.803/0.794
kddcup-guess_passwd_vs_satan	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	0.998/0.998
kr-vs-k-three_vs_eleven	0.993/0.992	0.993/0.993	0.995/0.995	0.993/0.993	0.995/0.995	0.995/0.995	0.995/0.995
kr-vs-k-zero-one_vs_draw	0.969/0.968	0.949/0.947	0.969/0.968	0.958/0.957	0.962/0.96	0.961/0.96	0.969/0.967
shuttle-2_vs_5	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
winequality-red-4	0.056/0.045	0.154/0.112	0.083/0.044	0.150/0.105	0.132/0.093	0.143/0.105	0.156/0.115
yeast4	0.304/0.287	0.351/0.32	0.307/0.290	0.339/0.305	0.321/0.299	0.214/0.197	0.335/0.303
yeast5	0.697/0.689	0.739/0.730	0.727/0.719	0.734/0.725	0.738/0.731	0.703/0.695	0.735/0.726
yeast6	0.454/0.444	0.456/0.44	0.505/0.496	0.462/0.445	0.507/0.496	0.554/0.545	0.514/0.501
oil	0.482/0.464	0.525/0.503	0.482/0.466	0.537/0.517	0.527/0.507	0.549/0.533	0.587/0.568
ozone_level	0.166/0.154	0.341/0.317	0.221/0.207	0.332/0.306	0.233/0.221	0.269/0.254	0.329/0.303
solar_flare_m0	0.106/0.085	0.155/0.115	0.110/0.088	0.129/0.109	0.121/0.101	0.098/0.078	0.194/0.139
thyroid_sick	0.865/0.856	0.852/0.841	0.858/0.849	0.845/0.834	0.845/0.836	0.856/0.847	0.873/0.864
wine_quality	0.232/0.216	0.278/0.234	0.22/0.178	0.247/0.201	0.225/0.189	0.238/0.193	0.290/0.248
yeast_me2	0.329/0.313	0.361/0.33	0.319/0.303	0.339/0.305	0.207/0.192	0.329/0.241	0.225/0.328
Average	0.556/0.546	0.586/0.566	0.555/0.542	0.584/0.565	0.564/0.549	0.560/0.546	**0.600/0.580**

TABLE 6.8: Table showing F1-Score/κ- Score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE SMOTE, LoRAS, and ProWRAS) for all 20 benchmarking datasets for Random Forest classifier. The column on the right shows the performance of the ProWRAS algorithm over all datasets. Observe that, in the last row that the average performance of ProWRAS is superior to all other oversampling algorithms.

Dataset	Baseline	SMOTE	Polynom-fit SMOTE	ProWSyn	CURE-SMOTE	LoRAS	**ProWRAS**
abalone9-18	0.211/0.198	0.342/0.3	0.266/0.245	0.327/0.278	0.311/0.272	0.325/0.293	0.38/0.335
abalone_17_vs_7_8_9_10	0.17/0.166	0.338/0.318	0.25/0.241	0.324/0.3	0.242/0.228	0.247/0.233	0.328/0.303
car-vgood	0.959/0.958	0.974/0.973	0.937/0.935	0.955/0.953	0.922/0.919	0.943/0.942	0.972/0.971
car_good	0.78/0.773	0.8/0.793	0.713/0.705	0.768/0.76	0.723/0.715	0.6/0.59	0.869/0.863
flare-F	0.087/0.066	0.147/0.117	0.091/0.07	0.183/0.156	0.075/0.055	0.1/0.08	0.21/0.181
hypothyroid	0.786/0.777	0.755/0.742	0.791/0.782	0.756/0.743	0.783/0.773	0.785/0.775	0.787/0.778
kddcup-guess_passwd_vs_satan	0.998/0.998	0.998/0.998	1.0/1.0	0.998/0.998	0.998/0.998	0.998/0.998	0.998/0.998
kr-vs-k-three_vs_eleven	0.989/0.989	0.995/0.995	0.989/0.989	0.991/0.991	0.991/0.991	0.991/0.990	0.996/0.996
kr-vs-k-zero-one_vs_draw	0.961/0.96	0.949/0.947	0.955/0.953	0.958/0.956	0.956/0.954	0.941/0.939	0.961/0.959
shuttle-2_vs_5	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
winequality-red-4	0.007/0.007	0.113/0.086	0.048/0.025	0.168/0.131	0.037/0.022	0.104/0.078	0.158/0.119
yeast4	0.238/0.23	0.351/0.326	0.26/0.249	0.352/0.321	0.267/0.251	0.245/0.235	0.357/0.325
yeast5	0.655/0.647	0.751/0.743	0.723/0.716	0.739/0.73	0.695/0.687	0.723/0.716	0.754/0.746
yeast6	0.435/0.428	0.478/0.465	0.464/0.456	0.474/0.459	0.482/0.473	0.531/0.523	0.518/0.507
oil	0.391/0.379	0.52/0.503	0.429/0.415	0.586/0.57	0.441/0.427	0.421/0.408	0.58/0.564
ozone_level	0.014/0.012	0.301/0.284	0.086/0.081	0.326/0.304	0.114/0.109	0.177/0.17	0.33/0.307
solar_flare_m0	0.07/0.047	0.105/0.066	0.058/0.036	0.109/0.079	0.067/0.045	0.081/0.058	0.168/0.117
thyroid_sick	0.843/0.833	0.875/0.867	0.839/0.829	0.85/0.84	0.843/0.834	0.85/0.841	0.868/0.859
wine_quality	0.292/0.282	0.378/0.355	0.304/0.283	0.331/0.296	0.275/0.258	0.307/0.282	0.368/0.344
yeast_me2	0.181/0.173	0.406/0.384	0.223/0.211	0.359/0.328	0.242/0.229	0.268/0.257	0.389/0.365
Average	0.504/0.496	0.579/0.563	0.521/0.511	0.578/0.56	0.524/0.519	0.531/0.521	**0.600/0.582**

Statistical significance of results: The higher the value of W_+ the better the performance of ProWRAS with respect to the compared algorithms, whereas a higher value of W_- implies

TABLE 6.9: Table showing F1-score/κ- score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE-SMOTE, LoRAS, and ProWRAS) for all 20 benchmarking datasets for k-Nearest neighbours classifier. The column on the right shows the performance of the ProWRAS algorithm over all datasets. Observe that, in the last row that the average performance of ProWRAS is superior to all the other oversampling algorithms.

Dataset	Baseline	SMOTE	Polynom-fit SMOTE	ProWSyn	CURE-SMOTE	LoRAS	**ProWRAS**
abalone9-18	0.206/0.197	0.285/0.223	0.345/0.295	0.316/0.257	0.335/0.279	0.309/0.249	0.384/0.35
abalone_17_vs_7_8_9_10	0.104/0.1	0.276/0.247	0.339/0.316	0.324/0.297	0.295/0.269	0.297/0.271	0.347/0.328
car-vgood	0.827/0.822	0.74/0.728	0.737/0.723	0.703/0.688	0.765/0.753	0.741/0.728	0.818/0.81
car_good	0.581/0.571	0.645/0.625	0.607/0.585	0.491/0.458	0.645/0.626	0.645/0.626	0.641/0.622
flare-F	0.203/0.183	0.273/0.228	0.257/0.212	0.255/0.208	0.284/0.244	0.311/0.273	0.301/0.263
hypothyroid	0.437/0.422	0.479/0.444	0.509/0.48	0.487/0.454	0.546/0.523	0.545/0.519	0.553/0.528
kddcup-guess_passwd_vs_satan	1.0/1.0	0.996/0.996	0.994/0.994	1.0/1.0	1.0/1.0	0.994/0.994	1.0/1.0
kr-vs-k-three_vs_eleven	0.902/0.899	0.918/0.915	0.925/0.923	0.911/0.908	0.907/0.904	0.92/0.917	0.94/0.938
kr-vs-k-zero-one_vs_draw	0.852/0.847	0.867/0.861	0.879/0.874	0.872/0.866	0.868/0.863	0.875/0.87	0.9/0.896
shuttle-2_vs_5	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
winequality-red-4	0.039/0.036	0.132/0.083	0.13/0.081	0.13/0.079	0.145/0.099	0.13/0.08	0.141/0.093
yeast4	0.219/0.21	0.261/0.217	0.286/0.244	0.267/0.223	0.292/0.25	0.257/0.212	0.308/0.268
yeast5	0.69/0.682	0.631/0.616	0.655/0.642	0.614/0.598	0.677/0.665	0.661/0.648	0.714/0.704
yeast6	0.574/0.565	0.314/0.288	0.363/0.339	0.301/0.274	0.507/0.493	0.4/0.379	0.588/0.579
oil	0.329/0.319	0.426/0.39	0.475/0.444	0.456/0.423	0.476/0.445	0.492/0.464	0.545/0.531
ozone_level	0.165/0.155	0.202/0.161	0.202/0.162	0.2/0.159	0.23/0.192	0.216/0.177	0.218/0.179
solar_flare_m0	0.052/0.033	0.226/0.168	0.207/0.146	0.2/0.138	0.208/0.155	0.225/0.175	0.228/0.177
thyroid_sick	0.5/0.48	0.527/0.487	0.531/0.495	0.528/0.49	0.555/0.525	0.556/0.524	0.556/0.526
wine_quality	0.104/0.096	0.246/0.2	0.226/0.177	0.22/0.168	0.235/0.19	0.232/0.185	0.25/0.204
yeast_me2	0.255/0.245	0.309/0.27	0.287/0.248	0.266/0.223	0.306/0.272	0.329/0.295	0.339/0.305
Average	0.452/0.443	0.488/0.457	0.498/0.469	0.477/0.446	0.514/0.487	0.506/0.479	**0.538/0.515**

the opposite. The value of R quantifies the degree of improvement of ProWRAS compared to the other oversampling models. Table 6.11 provides the p-values, as well as the W_+, W_- and R measures for ProWRAS with other oversampling models for all classifiers used in the study.

For the classifiers RF and LR, ProWRAS significantly improves the classifier performance, both F1-Score and κ-Score, compared to all other oversampling algorithms. For GB, ProWRAS significantly improves the classifier performance, both F1-Score and Balanced accuracy, compared to all other compared oversampling algorithms except for SMOTE. ProWRAS still improves the κ-Score significantly compared to SMOTE for the GB classifier. Moreover, SMOTE does better than ProWRAS in only six out of twenty datasets. Only for the kNN classifier, CURE SMOTE marginally outperformed ProWRAS, looking at the average performance from Table 6.9 and the p-value from Table 6.11. However, Figure 6.4 shows that ProWRAS performs better than CURE SMOTE for ten datasets, worse than CURE SMOTE for eight datasets and equally well for two datasets. Thus, one can conclude that using ProWRAS has been as effective as CURE-SMOTE for the kNN classifier.

6.4 Interpretations and applicability of the ProWRAS oversamping approach

Observe that the distribution of oversampling schemes used by ProWRAS used by LoRAS is vastly different for different classifiers. For KNN, the LLV oversampling scheme has been largely effective for LR, the LGB scheme has worked out for most datasets. For RF, the oversampling schemes HGV and HLV have worked out to be the best. For GB, which has produced the best average F1-Score and κ-Score among all classifiers over all oversampling models, interestingly, have a more uniform distribution for the ProWRAS oversampling schemes.

TABLE 6.10: Table showing F1-score/κ- score for several oversampling strategies (Baseline, SMOTE, Polynom-fit SMOTE, ProWSyn, CURE-SMOTE, LoRAS, and ProWRAS) for all 20 benchmarking datasets for Logistic Regression classifier. The column on the right shows the performance of the ProWRAS algorithm over all datasets. Observe that, in the last row that the average performance of ProWRAS is superior to all the other oversampling algorithms.

Dataset	Baseline	SMOTE	Polynom-fit SMOTE	ProWSyn	CURE-SMOTE	LoRAS	**ProWRAS**
abalone9-18	0.463/0.448	0.46/0.413	0.488/0.447	0.481/0.437	0.447/0.401	0.464/0.418	0.488/0.446
abalone_17_vs_7_8_9_10	0.266/0.259	0.301/0.271	0.333/0.307	0.3/0.271	0.309/0.281	0.312/0.284	0.315/0.287
car-vgood	0.086/0.075	0.378/0.338	0.403/0.365	0.388/0.35	0.37/0.33	0.382/0.343	0.407/0.371
car_good	0.0/0.0	0.099/0.028	0.1/0.029	0.099/0.028	0.102/0.033	0.096/0.025	0.103/0.033
flare-F	0.213/0.198	0.268/0.217	0.313/0.268	0.272/0.222	0.298/0.253	0.268/0.218	0.341/0.3
hypothyroid	0.356/0.339	0.386/0.338	0.42/0.376	0.39/0.342	0.423/0.381	0.412/0.367	0.446/0.407
kddcup-guess_passwd_vs_satan	0.99/0.99	0.99/0.99	0.99/0.99	0.99/0.99	0.99/0.99	0.99/0.99	0.99/0.99
kr-vs-k-three_vs_eleven	0.921/0.919	0.877/0.873	0.924/0.922	0.898/0.895	0.864/0.86	0.881/0.878	0.928/0.926
kr-vs-k-zero-one_vs_draw	0.816/0.81	0.692/0.677	0.796/0.787	0.727/0.714	0.716/0.703	0.713/0.699	0.853/0.847
shuttle-2_vs_5	0.966/0.966	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
winequality-red-4	0.007/0.007	0.113/0.086	0.048/0.025	0.168/0.131	0.037/0.022	0.104/0.078	0.158/0.119
yeast4	0.219/0.21	0.261/0.217	0.286/0.244	0.267/0.223	0.292/0.25	0.257/0.212	0.308/0.268
yeast5	0.511/0.5	0.571/0.552	0.615/0.599	0.584/0.566	0.601/0.584	0.585/0.567	0.591/0.574
yeast6	0.383/0.375	0.309/0.281	0.33/0.304	0.298/0.269	0.388/0.366	0.316/0.289	0.326/0.3
oil	0.524/0.507	0.474/0.44	0.518/0.49	0.506/0.475	0.484/0.454	0.475/0.444	0.535/0.511
ozone_level	0.191/0.178	0.262/0.226	0.287/0.253	0.263/0.226	0.314/0.282	0.311/0.279	0.347/0.319
solar_flare_m0	0.112/0.102	0.196/0.124	0.216/0.151	0.204/0.133	0.206/0.142	0.192/0.119	0.25/0.195
thyroid_sick	0.644/0.626	0.511/0.465	0.652/0.624	0.534/0.492	0.611/0.579	0.535/0.492	0.57/0.532
wine_quality	0.083/0.077	0.178/0.121	0.195/0.14	0.188/0.133	0.225/0.175	0.169/0.111	0.196/0.143
yeast_me2	0.22/0.21	0.255/0.211	0.28/0.238	0.262/0.218	0.291/0.25	0.266/0.222	0.295/0.254
Average	0.399/0.39	0.429/0.393	0.46/0.428	0.441/0.406	0.448/0.417	0.436/0.401	**0.472/0.441**

TABLE 6.11: Table showing the results of Wilcoxon's signed rank test for comparison of ProWRAS with other oversampling algorithms. The p-values in the table quantify the statistical significance of the improvement achieved by ProWRAS over each oversampling model. Higher value of W_+, shows that how superior performance of ProWRAS for higher number of datasets. Higher value of R shows better reliability of the results.

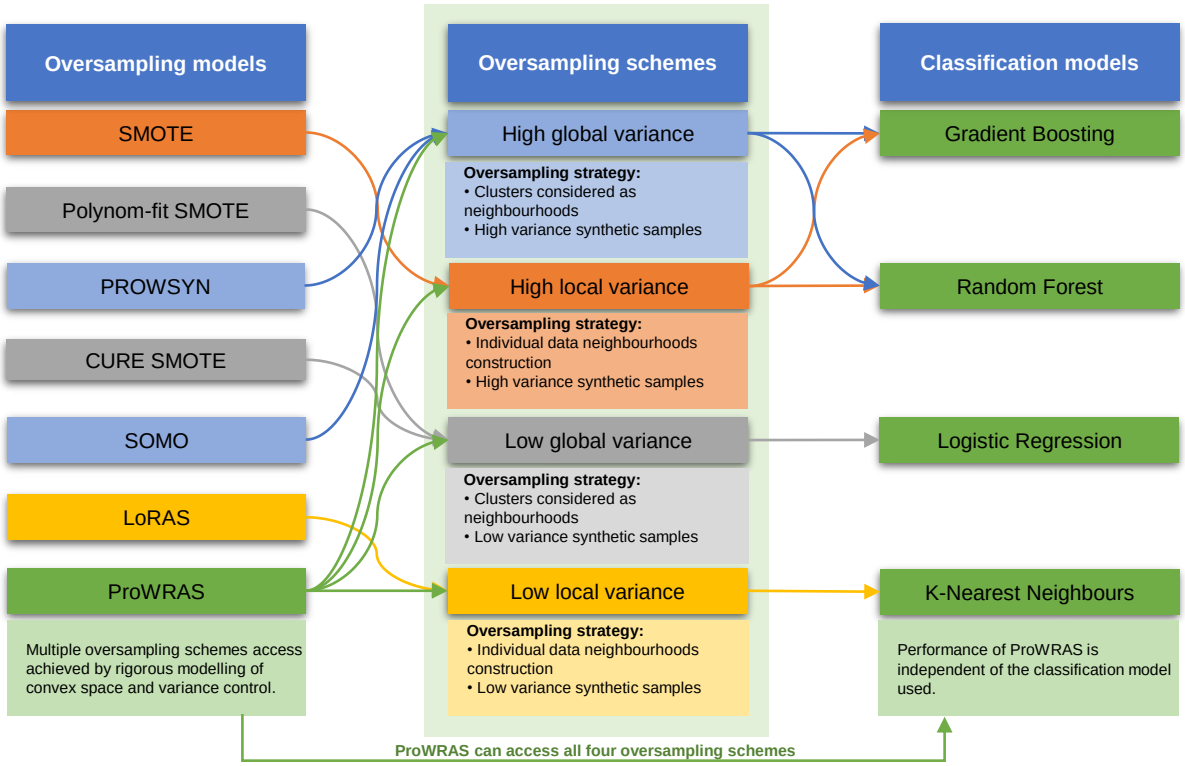
Dataset	classifier	Baseline	SMOTE	Polynom-fit SMOTE	ProWSYN	CURE-SMOTE	LoRAS
p-value (F1 score)	GB	0.003	0.024	0.000	0.012	0.001	0.003
p-value (κ-score)	GB	0.006	0.017	0.000	0.014	0.001	0.003
W_+/W_- (F1 score)	GB	189/15	182/21	195/6	189/15	195/10	189/10
W_+/W_- (κ-score)	GB	189/15	189/15	195/6	189/15	195/10	189/10
R (F1 score)	GB	0.860	0.795	0.925	0.860	0.914	0.878
R (κ-score)	GB	0.860	0.860	0.925	0.860	0.914	0.878
p-value (F1 score)	RF	0.000	0.013	0.000	0.001	0.000	0.000
p-value (κ-score)	RF	0.000	0.019	0.000	0.001	0.000	0.000
W_+/W_- (F1 score)	RF	204/1	182/15	204/3	200/3	207/0	207/1
W_+/W_- (κ-score)	RF	204/1	174/21	204/3	200/3	207/0	207/1
R (F1 score)	RF	0.983	0.820	0.989	0.960	0.995	1
R (κ-score)	RF	0.983	0.750	0.989	0.960	0.995	1
p-value (F1 score)	kNN	0.000	0.000	0.000	0.000	0.001	0.000
p-value (κ-score)	kNN	0.000	0.000	0.000	0.000	0.001	0.000
W_+/W_- (F1 score)	kNN	204/1	207/1	209/0	207/0	195/6	200/6
W_+/W_- (κ-score)	kNN	204/1	207/1	209/0	207/0	195/6	204/3
R (F1 score)	kNN	0.983	1	1	0.995	0.925	0.957
R (κ-score)	kNN	0.983	1	1	0.995	0.925	0.984
p-value (F1 score)	LR	0.001	0.000	0.047	0.000	0.034	0.003
p-value (κ-score)	LR	0.007	0.000	0.047	0.000	0.034	0.000
W_+/W_- (F1 score)	LR	204/3	207/0	189/10	204/1	189/10	207/0
W_+/W_- (κ-score)	LR	200/6	207/0	182/15	204/1	189/10	207/ 10
R (F1 score)	LR	0.989	0.995	0.878	0.983	0.878	0.995
R (κ-score)	LR	0.957	0.995	0.820	0.983	0.878	0.995

TABLE 6.12: Table showing the $\mathscr{I}$-scores for different oversampling algorithms for the final benchmarking experiments.

	Baseline	SMOTE	Polynom-fit SMOTE	ProWSyn	CURE-SMOTE	LoRAS	ProWRAS
$\mathscr{I}$	0.301	0.484	0.403	0.505	0.524	0.496	**0.833**

For the kNN model, the LoRAS algorithm that produces synthetic samples with low

FIGURE 6.5: Summarising oversampling schemes/strategies used by the investigated oversampling models and their respective influence on the classifier performance. For example, SMOTE generates samples with a "High local variance" scheme and works well for Gradient Boosting and Random Forest. Since the ProWRAS algorithm has access to all four oversampling schemes, its performance can be made independent of the chosen classifier.



local variance has also proved to be the most successful for the pilot study. LoRAS uses LLV strategy, LoRAS both reduces the variance of synthetic samples and generate them from neighbourhoods of each minority class data point. Note that the ProWRAS algorithm is also quite successful for the kNN classifier using the LLV scheme.

For RF and GB classifiers except for ProWRAS, SMOTE, and ProWSyn also perform well. Note that SMOTE uses synthetic samples with high local variance (SMOTE generates synthetic samples from neighbourhoods of minority class data points and does not control the variance of the synthetic samples, hence high local variance), while ProWSyn, within each cluster adopts the high global variance philosophy of synthetic sample generation (ProWSyn generates synthetic samples from the entire clusters of minority class data points and does not control the variance the synthetic samples, hence high global variance). Note that, for the ProWRAS algorithm, for most datasets, HGV and HLV strategies are successful for both GB and RF. For LR, except for ProWRAS, Polynom fit-SMOTE also performs well.

For the star topology, Polynom fit-SMOTE controls the variance of the synthetic samples by choosing a convex combination of a minority sample with the centroid of the minority class, rather than choosing a convex combination of two minority samples. Moreover, Polynom fit-SMOTE does not generate synthetic samples from individual data neighbourhoods. Thus, it follows a global oversampling scheme. Although the oversampling strategy of Polynom fit-SMOTE does not follow any of the strategies

considered for ProWRAS exactly, the use of the star topology (the default topology used by Polynom fit-SMOTE, hence used in benchmarking studies), is quite similar to the LGV strategy, which again is the successful strategy used by ProWRAS for most datasets for LR.

To sum up, the results of the benchmarking study show the ProWRAS improve classifier performance for all the chosen classifiers, with proper choice of oversampling scheme compared to the other state-of-the-art oversampling algorithms. Even though LoRAS is not included in the final benchmarking study, comparing LoRAS and ProWRAS on the 14 datasets that are common among the pilot study and the final benchmarking study, it is easy to see that ProWRAS convincingly outperforms LoRAS. Moreover, the significance of the improvement induced by ProWRAS is quantified using the Wilcoxon's signed rank test, which proves the improvement induced by using ProWRAS is statistically significant.

Observe that, given a dataset and a classifier, it is hard to predict which oversampling scheme of ProWRAS would be most effective for that particular dataset and classifier. To obtain the best results, it is highly recommended using all four oversampling schemes and choose the scheme that is most effective. However, from the results, some patterns are observed on: which oversampling scheme works better for which classifiers. For example, for GB and RF, oversampling schemes HGV and HLV are likely to be more effective; for kNN, the scheme LLV is likely to be more effective and for LR, the scheme LGV is likely to be more effective.

Of course, oversampling models based on modelling the convex space of the minority class are more effective for datasets where the convex space of the data can add some meaningful information to the training experience of the classifiers. ProWRAS also relies on convex space modelling of the minority class and hence can be widely applicable to regular tabular data that are homogeneous in nature, that is the features are well-defined with respect to the classification problem.

Observations confirm that different classifiers adapt differently to the different approaches of oversampling algorithms to generate synthetic samples. ProWRAS on the other hand, provides a straightforward and unbiased way to generate synthetic samples using different oversampling schemes and, thus, can adapt to different classifiers. The proposed ProWRAS approach can model the convex space of the minority class more rigorously than the LoRAS algorithm by controlling the variance of the synthetic samples better. ProWRAS achieves this through four unique oversampling schemes, as well as a proximity-weighted clustering system of the minority class data. The oversampling scheme of ProWRAS depends on two factors controlling the variance of the synthetic samples: neighbourhood size and convex space modelling. Moreover, ProWRAS allows generating low variance synthetic samples only in borderline clusters to avoid an overlap with the majority class, making the synthetic sample generation computationally cheaper compared to LoRAS. Using the multi-schematic approach of oversampling, ProWRAS significantly improves performance of classifiers in terms of both F1-Score and κ-Score compared to state-of-the-art oversampling models. ProWRAS is highly flexible to different classifiers and can find broad applicability in solving classification problems for real-world imbalanced datasets.

An implementation of the algorithm using `Python (V 3.7.4)` and several `Jupyter`

`Notebooks` from the benchmarking study is provided in the GitHub repository: https://github.com/COSPOV/ProWRAS.

With detailed documentation of the research work presented in this thesis, it is time to proceed to the final concluding remarks in the next chapter. The next chapter presents a concise account of my research, thereby highlighting its importance and indicating towards potential applicability.

Concluding remarks

Convex-space-based oversampling algorithms are a popular choice to improve imbalanced classification problems. This thesis focuses on solving tow primary limitations of convex space based oversampling algorithms. The first problem is the over-generalisation of minority classes by SMOTE and many of its extensions, and the second problem is the classifier-dependent performance of the SMOTE based oversampling algorithms.

The two proposed algorithms in this thesis, LoRAS and ProWRAS, successfully solved these problems. The key philosophy behind both algorithms is a rigorous modelling of the convex space that is lacking in the SMOTE algorithm and its extensions. The LoRAS algorithm forms the basis of rigorous convex space modelling. The ProWRAS algorithm, an extension of the LoRAS algorithm, to the best of my knowledge, is the first instance of a classifier independent algorithm, that is an oversampling algorithm whose performance is independent of the classifier used. This is achieved through a multi-schematic framework also unique to the algorithm.

The thesis provides an analytical explanation of this philosophy, which is consistent with the results of the data-based experiments. Moreover, the applicability of the philosophy is also tested in a real-life scenario through the development of the sc-SynO tool for automated annotation of rare cells from single-cell data. These algorithms have been tested over multiple datasets arising from diverse domains of research. They would thus be applicable to imbalanced classification problems arising from any research domain.

Oversampling algorithms based on convex space modelling, proposed in this thesis, are applicable to datasets where the convex space of the datasets is meaningful. This typically includes feature-based tabular datasets, where features are consistent through data points. Future directions of research can investigate how meaningful such methods can be, when integrated with deep learning approaches such as contrastive learning or generative networks. This could extend the applicability of such methods to more heterogeneous image and text datasets, where latent representations of the data are more important due to the lack of consistent and interpretable features in the data. Application-wise, the algorithms can be useful in real-life problems such as fraud detection, rare disease detection, software fault detection, anomaly detection, etc. Given that interpretable classical machine learning models on feature-based tabular datasets are still highly sought-after in many research fields such as finance, biomedicine, epidemiology, and given that imbalance naturally occurs in such datasets, the algorithms proposed in the thesis can add significant value to future research in such applicative fields.

Bibliography

[Krawczyk 2016] Krawczyk B. *Learning from imbalanced data: open challenges and future directions* Prog Artif Intell vol **5,** 221232 (2016). https://doi.org/10.1007/s13748-016-0094-0.

[Fernández 2018] Fernández A., García S., Galar M., Prati R.C., Krawczyk B., Herrera F.*Learning from Imbalanced Data Sets* Springer International Publishing, October 2018 ISBN: 978-3-31-998073-7, https://doi.org/10.1007/978-3-319-98074-4

[Zięba 2014] Zięba M., Tomczak J.M., Lubicz M., Świątek J. *Boosted SVM for extracting rules from imbalanced data in application to prediction of the post-operative life expectancy in the lung cancer patients, Applied Soft Computing* vol **14(A),** 2014, pp. 99-108, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2013.07.016.

[Azari 2015] Azari A., Janeja V.P. and Levin S. *Imbalanced learning to predict long stay Emergency Department patients* IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Washington, DC, USA, 2015, pp. 807-814. https://doi.org/10.1109/BIBM.2015.7359790

[Cao 2014] Cao P., Yang J., Li W., Zhao D., Zaiane O., *Ensemble-based hybrid probabilistic sampling for imbalanced data learning in lung nodule CAD* Computerized Medical Imaging and Graphics, vol **38(3),** 2014, pp 137-150, ISSN 0895-6111, https://doi.org/10.1016/j.compmedimag.2013.12.003.

[Acharya 2016] Acharya U.R., Chowriappa P., Fujita H., Bhat S., Dua S., Koh J.E.W, Eugene L.W.J., Kongmebhol P., Ng K.H. *Thyroid lesion classification in 242 patient population using Gabor transform features from high resolution ultrasound images* Knowledge-Based Systems, vol **107,** 2016, Pages 235-245, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2016.06.010.

[Krawczyk 2015] Krawczyk B., Schaefer G., Woniak M. *A hybrid cost-sensitive ensemble for imbalanced breast thermogram classification* Artificial Intelligence in Medicine, vol **65(3),** 2015, Pages 219-227, ISSN 0933-3657, https://doi.org/10.1016/j.artmed.2015.07.005.

[Bach 2017] Bach M., Werner A., Żywiec J., Pluskiewicz W. *The study of under- and over-sampling methods utility in analysis of highly imbalanced data on osteoporosis* Information Sciences, vol **384,** 2017, Pages 174-190, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2016.09.038.

[Pérez-Ortiz 2017] Pérez-Ortiz M, Gutiérrez P.A., Ayllón-Terán M.D., Heaton N., Ciria R., Briceño J., Hervás-Martínez C. *Synthetic semi-supervised learning in imbalanced*

domains: Constructing a model for donor-recipient matching in liver transplantation Knowledge-Based Systems, vol **123,** 2017, Pages 75-87, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2017.02.020.

[Yang 2014] Yang P., Yoo P. D., Fernando J., Zhou B.B., Zhang Z. and Zomaya A. Y., *Sample Subset Optimization Techniques for Imbalanced and Ensemble Learning Problems in Bioinformatics Applications,* in IEEE Transactions on Cybernetics, vol. **44(3)**, pp. 445-455, March 2014. https://doi.org/10.1109/TCYB.2013.2257480.

[Triguero 2015] Triguero I, del Río S., López V., Bacardit J., Benítez J.M., Herrera F. *ROSEFW-RF: The winner algorithm for the ECBDL14 big data competition: An extremely imbalanced big data bioinformatics problem* Knowledge-Based Systems, vol **87,** 2015, Pages 69-79, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2015.05.027.

[Dai 2015] Dai H-L. *Imbalanced Protein Data Classification Using Ensemble FTM-SVM* IEEE Trans Nanobioscience. 2015 Jun; vol **14(4)**:350-359. doi: https://doi.org/10.1109/TNB.2015.2431292.

[Iannello 2014] Iannello G., Percannella G., Soda P., Vento M. *Mitotic cells recognition in HEp-2 images, Pattern Recognition Letters* vol **45,** 2014, Pages 136-144, ISSN 0167-8655, https://doi.org/10.1016/j.patrec.2014.03.011.

[Lucas 2019] Lucas Y. *Credit card fraud detection using machine learning with integration of contextual knowledge* Artificial Intelligence [cs.AI]. Université de Lyon; Universität Passau (Deutscheland), 2019.https://tel.archives-ouvertes.fr/tel-02951477/document

[Pérez 2005] Pérez J.M., Muguerza J., Arbelaitz O., Gurrutxaga I., Martín J.I. (2005) *Consolidated Tree Classifier Learning in a Car Insurance Fraud Detection Domain with Class Imbalance* In: Singh S., Singh M., Apte C., Perner P. (eds) Pattern Recognition and Data Mining. ICAPR 2005. Lecture Notes in Computer Science, vol **3686,** Springer, Berlin, Heidelberg. https://doi.org/10.1007/11551188_41.

[Effendy 2014] Effendy V., Adiwijaya and Baizal Z.K.A., *Handling imbalanced data in customer churn prediction using combined sampling and weighted random forest* 2nd International Conference on Information and Communication Technology (ICoICT), Bandung, Indonesia, 2014, pp. 325-330, https://doi.org/10.1109/ICoICT.2014.6914086.

[So 1976] So X. *Two Modifications of CNN* in IEEE Transactions on Systems, Man, and Cybernetics, vol. **SMC-6**, no. 11, pp. 769-772, Nov. 1976, https://doi.org/10.1109/TSMC.1976.4309452.

[Devi 2017] Devi D., Biswas S.K., Purkayastha B., *Redundancy-driven modified Tomek-link based undersampling: A solution to class imbalance* Pattern Recognition Letters, vol **93,** 2017, Pages 3-12, ISSN 0167-8655, https://doi.org/10.1016/j.patrec.2016.10.006.

[Kumar 2019] Kumar, S., Biswas, S.K., Devi, D. *TLUSBoost algorithm: a boosting solution for class imbalance problem* Soft Comput 23, 1075510767 (2019). https://doi.org/10.1007/s00500-018-3629-4.

[Seiffert 2008] Seiffert C., Khoshgoftaar T.M., Van Hulse J. and Napolitano A., *RUSBoost: Improving classification performance when training data is skewed* 2008 19th International Conference on Pattern Recognition, Tampa, FL, 2008, pp. 1-4, https://doi.org/10.1109/ICPR.2008.4761297.

[Patel 2020] Patel H., Rajput D., Gadekallu T Iwendi C., Bashir A., Jo O. *A review on classification of imbalanced data for wireless sensor networks* International Journal of Distributed Sensor Networks. vol **16(4),** 10.1177/1550147720916404.

[Bellinger 2018] Bellinger C., Drummond C. Japkowicz N. *Manifold-based synthetic oversampling with manifold conformance estimation* Mach Learn vol **107,** 605637 (2018). https://doi.org/10.1007/s10994-017-5670-4

[Bellinger 2016] Bellinger C., Drummond C., Japkowicz N. *Beyond the Boundaries of SMOTE* In: Frasconi P., Landwehr N., Manco G., Vreeken J. (eds) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2016. Lecture Notes in Computer Science, vol **9851,** Springer, Cham. https://doi.org/10.1007/978-3-319-46128-1_16

[Goodfellow 2014] Goodfellow I.J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., and Bengio Y. *Generative adversarial nets* In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14). MIT Press, Cambridge, MA, USA, 26722680 https://arxiv.org/abs/1406.2661.

[Blum 2018] Blum O., Brattoli B., Ommer, B. *X-GAN: Improving Generative Adversarial Networks with ConveX Combinations* GCPR(2018) https://doi.org/10.1007/978-3-030-12939-2_15

[Mullick 2019] Mullick S.S., Datta S. and Das S., *Generative Adversarial Minority Oversampling* 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 1695-1704, https://doi.org/10.1109/ICCV.2019.00178.

[Douglas 2018] Douzas G., Bacao F., *Effective data generation for imbalanced learning using conditional generative adversarial networks* Expert Systems with Applications, vol **91,** 2018, Pages 464-471, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2017.09.030.

[Li 2018] Li J., Madry A., Peebles J. Schmidt, L. (2018) *On the Limitations of First-Order Approximation in GAN Dynamics* Proceedings of the 35th International Conference on Machine Learning, in PMLR vol **80**:3005-3013 http://proceedings.mlr.press/v80/li18d.html

[Mescheder 2018] Mescheder L.M., Geiger A., Nowozin S. (2018). *Which Training Methods for GANs do actually Converge?* Proceedings of the 35 th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. https://ei.is.mpg.de/uploads_file/attachment/attachment/424/paper.pdf

[Sajjadi 2018] Sajjadi M.S.M., Parascandolo G., Mehrjou A. Schölkopf B.. (2018). *Tempered Adversarial Networks* Proceedings of the 35th International Conference on Machine Learning, in PMLR vol**80**:4451-4459 http://proceedings.mlr.press/v80/sajjadi18a.html

[Tarawneh 2020] Tarawneh A.S., Hassanat A.S.A., Almohammadi K., Chetverikov D. and Bellinger C. *SMOTEFUNA: Synthetic Minority Over-Sampling Technique Based on Furthest Neighbour Algorithm* in IEEE Access, vol. **8,** pp. 59069-59082, 2020, https://doi.org/10.1109/ACCESS.2020.2983003.

[Powers 2011] Powers D.M.W. *Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation* Journal of Machine Learning Technologies. vol **2(1)** 3763, 2011 https://doi.org/10.1016/0005-2795(75)90109-9

[Matthews 1975] Matthews B.W. *Comparison of the predicted and observed secondary structure of T4 phage lysozyme* Biochimica et Biophysica Acta (BBA) - Protein Structure. vol **405(2)** 442451, 1975, https://doi.org/10.1016/0005-2795(75)90109-9.

[Olson 2008] Olson D.L. and Delen D. *Advanced Data Mining Techniques* Springer, 1st edition (February 1, 2008), page 138

[Jeni 2013] Jeni L.A., Cohn J.F., De La Torre F. *Facing Imbalanced Data Recommendations for the Use of Performance Metrics* Int Conf Affect Comput Intell Interact Workshops. 2013: 245-251. https://doi.org/10.1109/ACII.2013.47.

[Cohen 1960] Cohen, J. *A coefficient of agreement for nominal scales* Educational and Psychological Measurement. vol **20(1)** 3746. 1960 https://doi.org/10.1177/001316446002000104.

[Cohen 1968] Cohen J. *Weighed kappa: Nominal scale agreement with provision for scaled disagreement or partial credit* Psychological Bulletin. vol **70(4)**: 213220. 1968, https://doi.org/10.1037/h0026256.

[García 2010] García V., Mollineda R.A. and Sánchez J.S., Theoretical Analysis of a Performance Measure for Imbalanced Data, 2010 20th International Conference on Pattern Recognition, Istanbul, 2010, pp. 617-620, https://doi.org/10.1109/ICPR.2010.156.

[García 2009] García V., Mollineda R.A., Sánchez J.S. *Index of Balanced Accuracy: A Performance Measure for Skewed Class Distributions* In: Pattern Recognition and Image Analysis. IbPRIA 2009. Lecture Notes in Computer Science, vol **5524,** Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-02172-5_57.

[Chicco 2020] Chicco, D., Jurman, G. *The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation* BMC Genomics vol **21(6)** (2020). https://doi.org/10.1186/s12864-019-6413-7.

[Wilcoxon 1945] Wilcoxon F. *Individual comparisons by ranking methods* Biometrics Bulletin. vol1(6) 8083. 1945, https://doi.org/10.2307/3001968.

[Saito 2015] Saito T., Rehmsmeier M. *The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets* PLoS ONE 10(3): e0118432. 2015, https://doi.org/10.1371/journal.pone.0118432.

[Bellinger 2018] Bellinger, C., Drummond, C. & Japkowicz, N. *Manifold-based synthetic oversampling with manifold conformance estimation* Mach Learn 107, 605637 (2018). https://doi.org/10.1007/s10994-017-5670-4.

[Fawcett 2006] Fawcett, T. *An Introduction to ROC Analysis* Pattern Recognition Letters. vol **27(8)**: 861874. 2006, https://doi.org/10.1016/j.patrec.2005.10.010.

[Blagus 2013] Blagus R., Lusa L. *SMOTE for high-dimensional class-imbalanced data* BMC Bioinformatics vol **14,** 106. https://doi.org/10.1186/1471-2105-14-106.

[Jackman 2009] Jackman S. (2009), *Bayesian Analysis for the Social Sciences* John Wiley & Sons, Ltd.

[Lee 2000] Lee S. (2000). *Noisy replication in skewed binary classificatio* Computational Statistics & Data Analysis vol **34,** 165-191. https://doi.org/10.1016/S0167-9473(99)00095-X.

[Chawla 2002] Chawla N., Bowyer K., Hall L., Kegelmeyer W. *SMOTE: Synthetic Minority Over-sampling Technique* J. Artif. Intell. Res. (JAIR). vol **16,** 321-357. https://doi.org/10.1613/jair.953.

[Han 2005] Han H., Wang WY., Mao BH. *Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning* In: Huang DS., Zhang XP., Huang GB. (eds) Advances in Intelligent Computing. ICIC 2005. Lecture Notes in Computer Science, vol **3644,** Springer, Berlin, Heidelberg. https://doi.org/10.1007/11538059_91.

[Bunkhumpornpat 2009] Bunkhumpornpat C., Sinapiromsaran K., Lursinsap C. *Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem* In: Theeramunkong T., Kijsirikul B., Cercone N., Ho TB. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2009. Lecture Notes in Computer Science, vol **5476.** Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-01307-2_43.

[Chawla 2003] Chawla N.V., Lazarevic A., Hall L.O., Bowyer K.W. *SMOTEBoost: Improving Prediction of the Minority Class in Boosting* Knowledge Discovery in Databases: PKDD 2003. Lecture Notes in Computer Science, vol **2838,** Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-39804-2_12.

[He 2008] He H., Bai Y., Garcia E.A. and Li S., *ADASYN: Adaptive synthetic sampling approach for imbalanced learning* IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 2008, pp. 1322-1328, https://doi.org/10.1109/IJCNN.2008.4633969.

[Douzas 2018] Douzas G., Bacao F., Last F., *Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE* Information Sciences, vol **465,** 2018, pp. 1-20, https://doi.org/10.1016/j.ins.2018.06.056.

[Bunkhumpornpat 2012] Bunkhumpornpat C., Sinapiromsaran K. & Lursinsap C. *DBSMOTE: Density-Based Synthetic Minority Over-sampling TEchnique* Appl Intell vol **36**, 664684 (2012). https://doi.org/10.1007/s10489-011-0287-y.

[Gu 2009] Gu Q., Cai Z., Zhu L. *Classification of Imbalanced Data Sets by Using the Hybrid Re-sampling Algorithm Based on Isomap* Advances in Computation and Intelligence. ISICA 2009. Lecture Notes in Computer Science, vol **5821** Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04843-2_31.

[Xie 2015] Xie Z., Jiang L., Ye T., Li X. *A Synthetic Minority Oversampling Method Based on Local Densities in Low-Dimensional Space for Imbalanced Learning* Database Systems for Advanced Applications. DASFAA 2015. Lecture Notes in Computer Science, vol **9050** Springer, Cham. https://doi.org/10.1007/978-3-319-18123-3_1.

[Douzas 2017] Douzas G., Bacao F., *Self-Organizing Map Oversampling (SOMO) for imbalanced data set learning* Expert Systems with Applications, vol **82,** 2017, Pages 40-52, https://doi.org/10.1016/j.eswa.2017.03.073.

[Laureano 2019] Laureano L.B., Sison A.M., and Medina R.P.. *Handling Imbalanced Data through Affinity Propagation and SMOTE* In Proceedings of the 2nd International Conference on Computing and Big Data (ICCBD 2019). Association for Computing Machinery, New York, NY, USA, 2226. https://doi.org/10.1145/3366650.3366665.

[Barua 2014] Barua S., Islam M.M., Yao X. and Murase K. *MWMOTE–Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning* IEEE Transactions on Knowledge and Data Engineering vol **26** (2014): 405-425 https://doi.org/10.1109/TKDE.2012.232

[Barua 2013] Barua S., Islam M.M., Murase K. *ProWSyn: Proximity Weighted Synthetic Oversampling Technique for Imbalanced Data Set Learning* Advances in Knowledge Discovery and Data Mining. PAKDD 2013. Lecture Notes in Computer Science, vol **7819,** Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37456-2_27.

[Gazzah 2008] Gazzah S., Amara N.E.B. *New Oversampling Approaches Based on Polynomial Fitting for Imbalanced Data Sets* The Eighth IAPR International Workshop on Document Analysis Systems, Nara, 2008, pp. 677-684, https://doi.org/10.1109/DAS.2008.74.

[Kovács 2019] Kovács G., *An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets.* Applied Soft Computing, vol **83,** 105662, ISSN 1568-4946, 2019, https://doi.org/10.1016/j.asoc.2019.105662.

[van der Maaten 2008] van der Maaten L. and Geoffrey H. *Visualizing Data using t-SNE* Journal of Machine Learning Research vol **9** (2008): 2579–2605. https://www.jmlr.org/papers/v9/vandermaaten08a.html

[Lemaître 2017] Lemaître G., Nogueira F., and Aridas C.K. *Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning* J. Mach. Learn. vol **18,** 1 (January 2017), 559563. https://jmlr.org/papers/v18/16-365.html

[Ding 2011] Ding. Z *Diversified ensemble classifiers for highly imbalanced data learning and its application in bioinformatics* Ph.D. Dissertation. Georgia State University, USA. Advisor(s) Yan-Qing Zhang. 2011 Order Number: AAI3486649. https://scholarworks.gsu.edu/cgi/viewcontent.cgi?article=1060&context=cs_diss

[Sáez 2016] Sáez J.A., Krawczyk B., Woniak M., *Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets* Pattern Recognition, vol **57**, pp. 164-178, https://doi.org/10.1016/j.patcog.2016.03.012.

[Anand 2010] Anand A., Pugalenthi G., Fogel G.B., Suganthan P.N. *An approach for classification of highly imbalanced data using weighting and undersampling* Amino Acids vol **39(5)**:1385-91. https://doi.org/10.1007/s00726-010-0595-2.

[Hooda 2018] Hooda N., Bawa S., Rana P.S., *B2FSE framework for high dimensional imbalanced data: A case study for drug toxicity prediction* Neurocomputing, vol **276**, 2018, Pages 31-41, https://doi.org/10.1016/j.neucom.2017.04.081.

[Jing 2021] Jing X.Y., Zhang X., Zhu X., Wu F. et al., *Multiset Feature Learning for Highly Imbalanced Data Classification* in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol **43,** no. 1, pp. 139-156, 1 Jan. 2021, https://doi.org/10.1109/TPAMI.2019.2929166.

[Dal Pozzolo 2018] Dal Pozzolo A., Boracchi G., Caelen O., Alippi C. and Bontempi G., *Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy,* in IEEE Transactions on Neural Networks and Learning Systems, vol **29,** no. 8, pp. 3784-3797, Aug. 2018, https://doi.org/10.1109/TNNLS.2017.2736643.

[Varmedja 2019] Varmedja D., Karanovic M., Sladojevic S., Arsenovic M. and Anderla A., *Credit Card Fraud Detection - Machine Learning methods* 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 2019, pp. 1-5, https://doi.org/10.1109/INFOTEH.2019.8717766.

[McInnes 2018] McInnes L., Healy J., Melville J. (2018). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.* ArXiv e-prints. arXiv:1802.03426, https://arxiv.org/pdf/1802.03426.pdf.

[Bej 2021] Bej S., Davtyan N., Wolfien M., Nassar M., Wolkenhauer O. *LoRAS: an oversampling approach for imbalanced datasets.* Machine Learning, vol **110,** 279301 (2021), https://doi.org/10.1007/s10994-020-05913-4.

[Lähnemann 2020] Lähnemann D., Köster J., Szczurek E., McCarthy D.J.*et al. Eleven grand challenges in single-cell data science* Genome Biol vol 21, 31 (2020) https://doi.org/10.1186/s13059-020-1926-6.

[Lee 2020] Lee J., Hyeon D., and Hwang D., *Single-cell multiomics: technologies and data analysis methods*, Experimental & Molecular Medicine, vol **52,** pp. 1428–1442, Sep 2020. https://doi.org/10.1038/s12276-020-0420-2

[Duo 2018] Duò A., Robinson M., and Soneson C., *A systematic performance evaluation of clustering methods for single-cell rna-seq data (version 2; peer review: 2 approved)*," F1000Research, vol **7,** no. 1141, 2018. https://doi.org/10.12688/f1000research.15666.3

[Freytag 2018] Freytag S., Tian L., Lönnstedt I., Ng M., and Bahlo M. *Comparison of clustering tools in r for medium-sized 10x genomics single-cell rna-sequencing data (version 2; peer review: 3 approved)* F1000Research, vol **7,** no. 1297, 2018 https://doi.org/10.1038/s41576-018-0088-9

[Kiselev 2019] Kiselev V.Y., Andrews T.S., and Hemberg M. *Challenges in unsupervised clustering of single-cell rna-seq data* Nature reviews. Genetics, vol**20,** p. 273282, May 2019. https://doi.org/10.1038/s41576-018-0088-9

[Jindal 2018] Jindal A., Gupta P., Jayadeva, and Sengupta D. *Discovery of rare cells from voluminous single cell expression data* Nature Communications, vol **9,** 12 2018 https://doi.org/10.1038/s41467-018-07234-6

[Zhang 2018] Zhang F., Lehallier B., Schaum N., and Li T.Q. *Single-cell transcriptomics of 20 mouse organs creates a tabula muris Nature*, vol **562,** pp. 367–372, 10 2018 https://doi.org/10.1038/s41586-018-0590-4.

[Satoso 2017] Santoso B., Wijayanto H., Notodiputro K.A., and Sartono B. *Synthetic over sampling methods for handling class imbalanced problems : A review* IOP Conference Series: Earth and Environmental Science, vol **58,** pp. 012–031, Mar 2017. https://doi.org/10.1088/1755-1315/58/1/012031.

[Weiss 2007] Weiss G., McCarthy K., and Zabar B. *Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs?* in DMIN, pp. 35–41, 01 2007.

[Butler 2018] Butler A., Hoffman P., Smibert P., Papalexi E., and Satija R. *Integrating single-cell transcriptomic data across different conditions, technologies, and species* Nature biotechnology, vol **36,** 05 2018 https://doi.org/10.1038/nbt.4096.

[Wolfien *Cells* 2020] Wolfien M., Galow A.M., Müller P., Bartsch M., Brunner R.M., Goldammer T., Wolkenhauer O., Hoeflich A., and David R. *Single-nucleus sequencing of an entire mammalian heart: Cell type composition and velocity* Cells, vol **9(2),** 2020 https://doi.org/10.3390/cells9020318.

[Wolfien *Cardiovascular Research* 2020] Wolfien M., Galow A.M., Müller P., Bartsch M., Brunner R.M., Goldammer T., Wolkenhauer O., Hoeflich A., and David R. *Single nuclei sequencing of entire mammalian hearts: strain-dependent cell-type composition and velocity*, Cardiovascular Research, vol **116,** pp. 1249–1251, 04 2020. https://doi.org/10.1093/cvr/cvaa054. .

[Vidal 2019] Vidal R., Wagner J.U.G., Braeuning C., Fischer C., Patrick R., Tombor L., Muhly-Reinholz M., John D., Kliem M., Conrad T., Guimarães-Camboa N., Harvey R., Dimmeler S., and Sauer S. *Transcriptional heterogeneity of fibroblasts is a hallmark of*

the aging heart JCI Insight, vol **4,** 11 2019 https://doi.org/10.1172/jci.insight.131092.

[Galow 2020] Galow A.M., Wolfien M., Müller P., Bartsch M., Brunner R., Hoeflich A., Wolkenhauer O., David R., and Goldammer T. *Integrative cluster analysis of whole hearts reveals proliferative cardiomyocytes in adult mice* Cells, vol **9,** p. 1144, 05 2020. https://doi.org/10.3390/cells9051144.

[Linscheid 2019] Linscheid N., Logantha S.J.R.J., Poulsen P.C., Zhang S., Schrölkamp M., Egerod K.L., Thompson J.J., Kitmitto A., Galli G., Humphries M.J., Zhang H., Pers T.H., Olsen J.V., Boyett M., and Lundby A. *Quantitative proteomics and single-nucleus transcriptomics of the sinus node elucidates the foundation of cardiac pacemaking* Nature Communications, vol **10(1),,** p. 2889, 2019 https://doi.org/10.1038/s41467-019-10709-9.

[Marouf 2018] Marouf M., Machart P., Magruder S., Bansal V., Kilian C., Krebs C., and Bonn S. *Realistic in silico generation and augmentation of single cell rna-seq data using generative adversarial neural networks* Nature Communications volume, vol **11,** p. 166, 08 2018 https://doi.org/10.1038/s41467-019-14018-z.

[Xu 2021] Xu C., Lopez R., Mehlman E., Regier J., Jordan M., and Yosef N., *Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models* Molecular Systems Biology, vol **17,** 1 2021 https://doi.org/10.15252/msb.20209620.

[Brbi 2020] Brbi M., Zitnik M., Wang S., Pisco A., Altman R., Darmanis S,, and Leskovec J. *Mars: discovering novel cell types across heterogeneous single-cell experiments* Nature Methods, vol **17,** pp. 1–7, 10 2020 https://doi.org/10.1038/s41592-020-00979-3.

[Wang 2009] Wang L., Catalan F., Shamardani K., Babikir H., and Diaz A. *Ensemble learning for classifying single-cell data and projection across reference atlases* Bioinformatics, vol **36,** pp. 3585–3587, 02 2020 https://doi.org/10.1093/bioinformatics/btaa137.

[Saito 2015] Saito T. and Rehmsmeier M., *The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets*, PLOS ONE, vol. **10,** pp. 1–21, 03 2015. doi: https://doi.org/10.1371/journal.pone.0118432.

[Hu 2009] Hu S., Liang Y., Ma L., He Y. *MSMOTE: Improving Classification Performance When Training Data is Imbalanced.* 2009 Second International Workshop on Computer Science and Engineering, Qingdao, 2009, 13-17, https://doi.org/10.1109/WCSE.2009.756.

[Puntumapon 2012] Puntumapon K., Waiyamai K. *A Pruning-Based Approach for Searching Precise and Generalized Region for Synthetic Minority Over-Sampling.* Advances in Knowledge Discovery and Data Mining. PAKDD 2012. Lecture Notes in Computer Science, vol **7302,** Springer, Berlin, Heidelberg, https://doi.org/10.1007/978-3-642-30220-6_31.

[Hastie 2009] Hastie, T., Tibshirani, R., Friedman, J. H. *The elements of statistical learning: Data mining, inference, and prediction.* Springer series in statistics. Springer, 2009, New York, 2nd edition, https://link.springer.com/book/10.1007/978-0-387-84858-7.

[Yin 2008] Yin H. *The Self-Organizing Maps: Background, Theories, Extensions and Applications.* Computational Intelligence: A Compendium. Studies in Computational Intelligence, vol **115,** Springer, Berlin, Heidelberg, https://doi.org/10.1007/978-3-540-78293-3_17.

[Ma 2017] Ma, L., Fan, S. *CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests.* BMC Bioinformatics 2017, vol **18,** 169, https://doi.org/10.1186/s12859-017-1578-z.

[Guha 2001] Guha S., Rastogi R., Shim K. *Cure: an efficient clustering algorithm for large databases.* Information Systems, 2001, vol **26(1),** 35-58, https://doi.org/10.1016/S0306-4379(01)00008-4.

[Brown 2014] Brown R.A. *Building a Balanced k-d Tree in O(kn log n) Time, Journal of Computer Graphics Techniques (JCGT)*, vol **4(1),** 50-68, 2015 http://jcgt.org/published/0004/01/03/

Appendix A

Implementation for the integrated LoRAS algorithms

This appendix provides an implementation of the integrated LoRAS algorithms (LoRAS and LoRAS UMAP) with possible manifold learning options with both t-SNE and UMAP. The implementation is for binary classifications only.

- `convex_neighbourhood` corresponds to the parameter k from the LoRAS algorithm. The standard value of this parameter should be 5 to 10. However for larger datasets (minority class size more than 500) it can be extended to 20 to 30.
- `umap_neighbourhood` corresponds to the parameter `n_neighbours` from the LoRAS algorithm. A standard value of 5 would be fine for this parameter.
- `shadow` corresponds to the parameter $|S_p|$ from the LoRAS algorithm. The standard value of this parameter is to be kept around 100.
- `sigma` corresponds to the standard deviation for a Gaussian distribution to draw noise from. The standard value of sigma can be taken as 0.005. Although in the LoRAS algorithm it is proposed that a list of feature-wise sigma values is taken as input, we have noticed that a standard value of small enough sigma works quite well.
- `num_RACOS` corresponds to the parameter $\mathtt{N_{gen}}$ from the LoRAS algorithm. The standard value for this parameter is $int(\frac{|C_{maj}|-|C_{min}|}{|C_{min}|})$, where $|C_{maj}|$ and $|C_{min}|$ are the number of majority and minority instances in the training dataset, that is provided as input to the `LoRAS_gen` function and the *int* function refers to the greatest integer floor function in case $\frac{|C_{maj}|-|C_{min}|}{|C_{min}|}$ is a float value.
- `num_convcomb` corresponds to the parameter $\mathtt{N_{aff}}$ from the LoRAS algorithm.

```
import numpy as np
import umap.umap_ as umap
from sklearn.manifold import TSNE
from  numpy.random  import  randint , normal , seed
from  sklearn.neighbors  import  NearestNeighbors

def Neb_grps(data,near_neb):
    'Function calculating nearest near_neb neighbours
```

```
    (among input data points), for every input data point'
    from sklearn.neighbors import NearestNeighbors
    nbrs = NearestNeighbors(n_neighbors=near_neb,
    algorithm='ball_tree').fit(data)
    distances, indices = nbrs.kneighbors(data)
    neb_class=[]
    for i in (indices):
        neb_class.append(i)
    return(np.asarray(neb_class))

def LoRAS(data, convex_neighbourhood, shadow, sigma, num_RACOS, num_convcom):
    'Function creating LoRAS samples for one
    minority data point neighbourhood'
    np.random.seed(42)
    data_shadow=([])
    for i in range (convex_neighbourhood):
        c=0
        while c<shadow:
            data_shadow.append(data[i]+np.random.normal(0,sigma))
            c=c+1
    data_shadow==np.asarray(data_shadow)
    data_shadow_lc=([])
    for i in range(num_RACOS):
        idx = np.random.randint(shadow*convex_neighbourhood,
        size=num_convcom)
        w=np.random.randint(100, size=len(idx))
        aff_w=np.asarray(w/sum(w))
        data_tsl=np.array(data_shadow)[idx,:]
        data_tsl_=np.dot(aff_w, data_tsl)
        data_shadow_lc.append(data_tsl_)
    return(np.asarray(data_shadow_lc))

def LoRAS_gen(data, labels, convex_neighbourhood, shadow,
sigma, num_RACOS, num_convcom, embedding):
    'Main LoRAS function performing preferred dimension reduction'
    'For UMAP use embedding="umap"'
    'For t-SNE use embedding="tsne"'
    'For regular use embedding="regular"'
    import numpy as np
    import umap.umap_ as umap

    features_1_trn=data[np.where(labels==1)]
    features_0_trn=data[np.where(labels!=1)]
    n_feat=features_0_trn.shape[1]
    if embedding == 'umap':
        umap_neighbourhood = int(input("Enter umap_neighbourhood:"))
        data_embedded_min = umap.UMAP(n_neighbors=umap_neighbourhood,
        min_dist=0.00000001, n_components=2,
        metric='euclidean', random_state=42).fit_transform(features_1_trn)
        nb_list=Neb_grps(data_embedded_min,  convex_neighbourhood)
        print('UMAP for minority class: Done\n')
    elif embedding == 'tsne':
        perp = int(input("Enter perplexity:"))
        data_embedded_min = TSNE(n_components=2, perplexity=perp,
```

```
        random_state=42)
        .fit_transform(features_1_trn)
        nb_list=Neb_grps(data_embedded_min,  convex_neighbourhood)
        print('t-SNE for minority class: Done\n')
    elif embedding == 'regular':
        data_embedded_min = features_1_trn
        nb_list=Neb_grps(data_embedded_min,  convex_neighbourhood)
    else:
        print('Enter embedding="tsne" or "umap" or "regular"')

    RACOS_set=[]
    for i in range (len(nb_list)):
        RACOS_i= LoRAS(features_1_trn[nb_list[i]],convex_neighbourhood,
        shadow,sigma,num_RACOS,num_convcom)
        RACOS_set.append(RACOS_i)
    LoRAS_set=np.asarray(RACOS_set)
    LoRAS_1=np.reshape(LoRAS_set,(len(features_1_trn)*num_RACOS,n_feat))
    features_1_trn=np.concatenate((LoRAS_1,features_1_trn))
    print('Data generation: Done\n')
    return(np.concatenate((features_1_trn,features_0_trn)),
    np.concatenate((np.zeros(len(features_1_trn))+1,
    np.zeros(len(features_0_trn)))))
```

Appendix B

Implementation for the ProWRAS algorithm

```
r"""
ProWRAS:
Generates sample data points for imbalanced data sets.
"""

# List of public functions.
__all__ = ["ProWRAS_gen"]

# See Style guide: https://www.python.org/dev/peps/pep-0008/
# Importing libraries

import warnings
import time

import numpy as np
from numpy.random import randint, normal, seed
from sklearn.neighbors import NearestNeighbors


def minority_class(data, labels):
    """Returns all data points that are in the minority class.
    data: numpy array of data points.
    labels: numpy array of labels. (== 1: minority class, != 1: other class)
    """
    return data[np.where(labels == 1)]


def majority_class(data, labels):
    """Returns all data points that are not in the minority class.
    data: Numpy array of data points.
    labels: Numpy array of labels. (== 1: minority class, != 1: other class)
    """
    return data[np.where(labels != 1)]


def random_subset(data, size):
    """Return a subset of the array data. The returned data points
    are selected
```

```
    by random.
    data: Numpy array of data points.
    size: Number of values in the returned array.
    """
    return data[randint(len(data), size=size)]


def random_item(data):
    """Returns a random item in the given array.
    data: Numpy array of data points.
    """
    return data[randint(len(data))]


def concatArray(listOfListOfThings):
    """Changes the type of the given array to Numpy array.
    Concatenates all aubarrays.

    listOfListOfThings: Array of arrays.
    Example:
    [[1,2,3],[4,5,6,7],[8,9]] will become
    array([1,2,3,4,5,6,7,8,9])
    [[[1,2],[3,4]],[[5,6],[7,8]]] will become array([[1,2],[3,4],[5,6],[7,8]])
    """
    return np.concatenate(np.array(listOfListOfThings))


def random_vector_with_sum_one(size):
    """Rerurns an array of the given size. The array has random non negative
    values. The sum of all Values is one.
    size: Number of the expected values in the array.
    """
    w = [0]
    sum_w = 0
    while sum_w == 0:
        w = randint(100, size=size)
        sum_w = sum(w)

    return np.array(w / sum_w)


# Defining ProWRAS function

class ProWRASHelper:
    """
    A collection of helpfull functions for the ProWRAS algorithm.
    """

    def __init__(
        self,
        convex_nbd,
        shadow,
        sigma,
        n_jobs,
```

```
94          num_feats):
95          """Constructor of the helper class. Checks some values for validity.
96          Sets the parameters for the helping functions.
97          convex_nbd: Number of points in the neighbourhoods. If a cluster has
98              more than this amount of points. It will be divided in
99              neighbourhoods of this size.
100         shadow: Number of shadow point to create for each point in cluster.
101         sigma: Used to generate random shadow samples with normal deviation.
102
103         n_jobs: Maximal count of processor cores, used during the calculation.
104
105         num_feats: Number of features for each sample.
106         """
107         assert convex_nbd >= 1
108         assert shadow >= 1
109         assert n_jobs >= 1
110         assert num_feats >= 1
111
112         self.convex_nbd_size = int(convex_nbd)
113         self.shadow_size = int(shadow)
114         self.sigma = sigma
115         self.n_jobs = int(n_jobs)
116         self.num_feats = int(num_feats)
117         self.isDebugEnabled = False
118
119     def random_vector(self, point):
120         """Returns a random vector with normal deviation
121         around the given point.
122         point: Data point (Numpy array of float values.)
123         returns: Numpy array of float values.
124         """
125         return point + [normal(0, self.sigma) for k in range(len(point))]
126
127     def shadow_for_point(self, point):
128         """Creates random shadow points for a given point.
129         point: Data point (Numpy array of float values.)
130         returns: Numpy array of points.
131         """
132         return [
133             self.random_vector(point)
134             for _c in range(self.shadow_size)
135             ]
136
137     def split_into_neighbourhoods(self, cluster):
138         """If the given cluster has more points than the
139         given convex_nbd_size,
140         it will be divided into neighbourhood of size convex_nbd_size.
141         It will be returned an array of arrays of indices.
142
143         cluster: Numpy array of data points.
144         """
145         if len(cluster) > self.convex_nbd_size:
146             self.debug('local')
147             return self.neb_grps(cluster)
```

```
148
149         self.debug('global')
150         return np.array([np.array(range(len(cluster)))])
151
152     # Defining ProWRAS function
153
154     def neb_grps(self, data):
155         """
156         Function calculating nearest convex_nbd neighbours
157         (among input data points), for every input data point.
158         data: Numpy array of data points.
159         """
160         nbrs = NearestNeighbors(
161                 n_neighbors=self.convex_nbd_size,
162                 n_jobs=self.n_jobs)
163         nbrs = nbrs.fit(data)
164         _distances, indices = nbrs.kneighbors(data)
165         return np.asarray(indices)
166
167     def partition_info(self, data, labels, max_levels, n_neighbors, theta):
168         """Divides the given array of data points into weighted layers.
169         data: Numpy array of data points.
170         labels: numpy array of labels. (1: minority class, != 1: other class)
171         max_levels: Maximal number of returned layers.
172         n_neighbours: Number of neighbours in a layer from the minority class
173             for points in majority class.
174         theta: Scaling factor for the weights.
175         """
176
177         def proximity_level(level):
178             """Calculates the weight for a layer level."""
179             return np.exp(-theta * (level - 1))
180
181         # Step 2
182         P = np.where(labels == 1)[0]
183         data_maj = majority_class(data, labels)
184
185         Ps = []
186         weights = []
187
188         # Step 3
189         for i in range(1, max_levels):
190             if len(P) == 0:
191                 break
192             # Step 3 a
193             n_neighbors = min([len(P), n_neighbors])
194             nn = NearestNeighbors(n_neighbors=n_neighbors, n_jobs=self.n_jobs)
195             nn.fit(data[P])
196             _distances, indices = nn.kneighbors(data_maj)
197
198             # Step 3 b
199             P_i = np.unique(np.hstack(indices))
200
201             # Step 3 c - proximity levels are encoded in the Ps list index
```

```
202             Ps.append(P[P_i])
203             weights.append(proximity_level(i))
204 
205             # Step 3 d
206             P = np.delete(P, P_i)
207 
208         if len(P) > 0:
209             # Step 4
210             Ps.append(P)
211 
212             # Step 5
213             weights.append(proximity_level(i - 1))
214 
215         # Step 6
216         weights = np.array(weights)
217 
218         # weights is the probability distribution of sampling in the
219         # clusters identified
220         weights = weights / np.sum(weights)
221         return (np.array(Ps), weights)
222 
223     def limit_neighbourhood(self, neighbourhood):
224         """Limits a neighbourhood to convex_nbd_size amount of values."""
225         if len(neighbourhood) > self.convex_nbd_size:
226             return neighbourhood[:self.convex_nbd_size]
227         return neighbourhood
228 
229     def generate_points(self, cluster, num_of_points_to_create, num_convcomb):
230         """
231         Function creating samples for one minority data point neighbourhood.
232         cluster: Numpy array of data points.
233         num_of_points_to_create: Size of the returned array with new created
234             points.
235         num_convcomb: Number of points used for creating one new point.
236         """
237 
238         assert num_convcomb >= 1
239 
240         generated_data = []
241 
242         isLowVariance = True
243         num_convcomb = int(num_convcomb)
244         if num_convcomb < self.num_feats:
245             isLowVariance = False
246             num_convcomb = 2
247             self.debug('high')
248         else:
249             self.debug('low')
250 
251         neb_list = self.split_into_neighbourhoods(cluster)
252 
253         for _i in range(int(num_of_points_to_create)):
254             random_neighbourhood = cluster[random_item(neb_list)]
255 
```

```
256             if isLowVariance:
257                 data_shadow = concatArray([
258                     self.shadow_for_point(x)
259                     for x in self.limit_neighbourhood(random_neighbourhood)
260                     ])
261             else:
262                 data_shadow = random_neighbourhood
263 
264             shadowPoints = random_subset(data_shadow, num_convcomb)
265             aff_w = random_vector_with_sum_one(num_convcomb)
266 
267             generated_data.append(np.dot(aff_w, shadowPoints))
268 
269         return np.array(generated_data)
270 
271     def debug(self, message):
272         """Prints a message, if debug is enabled."""
273         if self.isDebugEnabled:
274             print(message)
275 
276     def enableDebug(self):
277         """Prints a message, if debug is enabled."""
278         self.isDebugEnabled = True
279 
280 def ProWRAS_gen(
281         data,
282         labels,
283         max_levels,
284         convex_nbd,
285         n_neighbors,
286         max_concov,
287         num_samples_to_generate,
288         theta,
289         shadow,
290         sigma,
291         n_jobs,
292         enableDebug=False):
293     """Calculates shadow points for a minority class of data points.
294     data: numpy array of data points.
295     labels: numpy array of labels. (== 1: minority class, != 1: other classes)
296     max_levels: Maximal number of layers, used in the algorithm.
297     convex_nbd: Number of points in the neighbourhoods. If a layer has more
298         than this amount of points. It will be divided in neighbourhoods of
299         this size.
300     n_neighbours: Number of neighbours in a layer from the minority class
301         for points in majority class. This parameter is used for generating
302         the layers.
303     max_concov:
304     num_of_samples_to_generate: Number of new generated points in the minority
305         class of data points.
306     theta: Scaling factor for the weights of the layers.
307     shadow: Number of shadow point to create for each point.
308     sigma: Used to generate random shadow samples with normal deviation.
309 
```

```
    n_jobs: Maximal count of processor cores, used during the calculation.
    """

    warnings.filterwarnings("ignore")
    seed(int(time.time() * 1000) & 0x0fffffff)

    features_1_trn = minority_class(data, labels)
    features_0_trn = majority_class(data, labels)

    num_feats = data.shape[1]

    helper = ProWRASHelper(convex_nbd, shadow, sigma, n_jobs, num_feats)
    if enableDebug:
        helper.enableDebug()

    clusters, weights = helper.partition_info(
        data, labels, max_levels, n_neighbors, theta)

    num_samples_each_cluster = np.ceil(num_samples_to_generate * weights)
    num_convcomb_each_cluster = np.ceil((weights / max(weights)) * max_concov)

    sample_params = zip(
        clusters,
        num_samples_each_cluster,
        num_convcomb_each_cluster
        )

    synth_samples = concatArray([
        helper.generate_points(data[cluster], num_samples, num_convcomb)
        for (cluster, num_samples, num_convcomb) in sample_params
        ])

    prowras_train = np.concatenate((
        synth_samples,
        features_1_trn,
        features_0_trn
        ))

    minority_class_size = len(synth_samples) + len(features_1_trn)
    prowras_labels = np.concatenate((
        np.ones(minority_class_size),
        np.zeros(len(features_0_trn))
        ))

    return(prowras_train, prowras_labels)
```

Appendix C

Complexity of the ProWRAS algorithm

C.1 Assumptions

- N, M are big
- $1 \leq s, L$
- $4 \leq d$
- $2 \leq m \leq s \cdot n$
- $1 \leq c, n \leq N$
- The complexity of the k-Neighbourhood search was estimated by Brown [Brown 2014], We will use:

$$O(\texttt{k-Nextneighbourhood}) = O(d \cdot N \cdot \log(N))$$

Where

- $N = |\texttt{data}|$ is the number of datasets.
- d is the number of features
  $(\forall x \in \texttt{data} : \texttt{dim}(x) = d)$
- $L = \texttt{max\_levels}$ is the maximum number of created layers in the `partition_info` function.
- $M = \texttt{num\_samples\_to\_generate}$ is the number of new samples `ProWRAS_oversampling` will generate.
- $m = \texttt{max\_conv}$ is the number of shadow samples, used in affine combination to create a new datapoint.
- $n = \texttt{neb\_conv}$ is the maximum size of a neighbourhood, used to create shadow samples for the affine combination.

- $s =$ `shadow` is number of shadow samples, that are created for each sample in a neighbourhood.
- $c =$ `n_neighbours_max` is number maximal neighbourhood size.

C.2 Estimations

The computation complexity as estimated in complexity analysis of Algorithms 7, 8 and 6 results in:

$$O(\texttt{ProWRAS\_oversampling}) = O\Big(L \cdot d \cdot \big(N \cdot \log(N) + c \cdot \log(c) + M(s \cdot n + m)\big)\Big)$$

As c is constant or $c \leq N$ we have:

$$O\Big(L \cdot d \cdot \big(N \cdot \log(N) + M(s \cdot n + m)\big)\Big)$$

For every sensible choice of m, it is true, that $m \leq s \cdot n$. So we get:

$$O\big(L \cdot d \cdot (N \cdot \log(N) + M \cdot s \cdot n)\big)$$

For HLV and LLV it is suggested to use $n = 5$. For HGV and LGV we select $n =$ `neb_conv` $\geq |C_{min}|$. As no neighbourhood can be greater than N, the complexity simplifies to:

$$O\big(L \cdot d \cdot N \cdot (\log(N) + M \cdot s)\big)$$

Algorithm 6 ProWRAS oversampling algorithm (Complexity Analysis)

Inputs:

`data` Data points.

Parameters:

`max_conv`	(> 0)	Weight for number of generated samples per layer.
`num_samples_to_generate`	(> 0)	Maximal count of generated samples in the output.

Function `ProWRAS_oversampling(data)` **begin**

- `clusters` $\leftarrow$ `partition_info(data)` (See Algorithm 7) ..... $O(\texttt{partition\_info}) = O(L \cdot d \cdot N \cdot \log(N))$
- `weight_max` $\leftarrow max(\{$`weight` $:$ `(cluster, weight)` $\in$ `clusters`$\})$ ..... $O(L)$
- Initialize *synth_samples* with an empty set. ..... $O(1)$
- **For** `(cluster, weight)` $\in$ `clusters` **do**
  - `num_samples` $\leftarrow \lceil$`num_samples_to_generate` $\cdot$ `weight`$\rceil$ ..... $O(1)$
  - `num_convcomb` $\leftarrow \left\lceil \frac{\texttt{max\_conv} \cdot \texttt{weight}}{\texttt{weight\_max}} \right\rceil$ ..... $O(1)$
  - `synth` $\leftarrow$ `generate_points(cluster, num_samples, num_convcomb)` (See Algorithm 8) ..... $O(\texttt{generate\_points}) = O(d(c \log(c) + M(s \cdot n + m))$
  - `synth_samples` $\leftarrow$ `synth_samples` $\cup$ `synth` ..... $O(1)$
- **endfor**

..... $O(\texttt{for} \ldots \texttt{endfor}) = O\Big(L \cdot \big(3O(1) + O(d \cdot (c \cdot \log(c) + M(s \cdot n + m)))\big)\Big) = O\Big(L \cdot d \cdot \big(c \cdot \log(c) + M(s \cdot n + m)\big)\Big)$

- Return resulting set of generated data points as `synth_samples`.

end

$$\begin{aligned} O(\textit{ProWRAS\_oversampling}) &= O(L \cdot d \cdot N \cdot \log(N)) + O(L) + O(1) + O\Big(L \cdot d \cdot \big(c \cdot \log(c) + M(s \cdot n + m)\big)\Big) \\ &= O(L \cdot d \cdot N \cdot \log(N)) + O\Big(L \cdot d \cdot \big(c \cdot \log(c) + M(s \cdot n + m)\big)\Big) \\ &= O\Big(L \cdot d \cdot \big(N \cdot \log(N) + c \cdot \log(c) + M(s \cdot n + m)\big)\Big) \end{aligned}$$

Algorithm 7 Proximity weighted minority class data partitioning (Complexity Analysis)

Inputs:

`data` Data points.

Parameters:

`max_levels`	(≥ 1)	Maximal repeat of bordersearch.
`n_neighbours_max`	(≥ 1)	Number of neighbours considered for the majority class data points while constructing minority class partitions.
θ	(> 0)	Scaling for weights.
`num_feats`	$(= dim(x_1))$	Number of features.

Function `partition_info(data)` **begin**

- `X_maj` $\leftarrow$ Data points in `data` with label for major class. ..... $O(N)$
- `X_min` $\leftarrow$ Data points in X with label for minor class. ..... $O(N)$
- $L =$ `max_levels` ..... $O(1)$
- Initialize `clusters` as empty set. ..... $O(1)$
- **For** $i = 1, 2, \ldots, L-1$ **do**
  - **If** $|$`X_min`$| = 0$ **then**
    - break
  - **endif**
  - ..... $O(1)$
  - `weight` $= \exp(-\theta \cdot (i-1))$ ..... $O(1)$
  - $k \leftarrow \min(|$`X_min`$|,$ `n_neighbours_max`$)$ ..... $O(1)$
  - `cluster` $\leftarrow$ All neighbours in k-Neighbourhoods from `X_maj` in `X_min` ..... $O(d \cdot N \cdot \log(N))$
  - `clusters` $=$ `clusters` $\cup \{($`cluster`$,$ `weight`$)\}$ ..... $O(1)$
  - `X_min` $\leftarrow$ `X_min` $\setminus$ `cluster` ..... $O(N)$
- **endfor**
- ..... $O(\texttt{for}\ldots\texttt{endfor}) = O\Big(L \cdot (4O(1) + O(N) + O(d \cdot N \cdot \log(N)))\Big) = O(L \cdot d \cdot N \cdot \log(N))$
- **If** $|$`X_min`$| > 0$ **then**
  - `weight` $= \exp(-\theta \cdot (L-1))$ ..... $O(1)$
  - `clusters` $=$ `clusters` $\cup \{($`X_min`$,$ `weight`$)\}$ ..... $O(1)$
- **endif**
- ..... $O(1)$
- `weight_sum` $\leftarrow sum(\{$`weight` $:$ `(cluster, weight)` $\in$ `clusters`$\})$ ..... $O(L)$
- `clusters` $\leftarrow \left\{\left(\texttt{cluster}, \frac{\texttt{weight}}{\texttt{weight\_sum}}\right) : (\texttt{cluster}, \texttt{weight}) \in \texttt{clusters}\right\}$ ..... $O(L)$
- Returns pairs of clusters and normalized weights as `clusters`.

end

..... $O(\texttt{begin}\ldots\texttt{end}) = O\Big(2O(N) + 4O(1) + 2O(L) + O\big(L \cdot d \cdot N \cdot \log(N)\big)\Big) = O\big(L \cdot d \cdot N \cdot \log(N)\big)$

Algorithm 8 Cluster-wise oversampling schemes (Complexity Analysis)

Inputs:

`cluster`	Data points.
`num_samples`	Number of generated shadowsamples per parent data point.
`num_convcomb`	Number of convex combinations for each new sample.

Parameters:

`neb_conv`	(≥ 1)	Number of data points used in affine combination for new samples.
`shadow`	(≥ 1)	Number of generated shadowsamples per parent data point.
`sigma`	(≥ 0)	List of standard deviations for normal distributions for adding noise to each feature.

Function `generate_points(cluster, num_samples, num_convcomb)` **begin**

Initialize `generated_data` with empty set. ..... $O(1)$

If $|$`cluster`$|$ > `neb_conv` **then**
  `neb_list` ← set of all k-Neighbourhoods in `cluster` ..... $O(d \cdot c \cdot \log(c))$
else
  `neb_list` ← {`cluster`} ..... $O(1)$
endif
..... $O(\texttt{if}\ldots\texttt{endif}) = O(d \cdot c \cdot \log(c))$

If `num_convcomb` < `num_feats` **then**
  $k \leftarrow 2$
else
  $k \leftarrow$ `num_convcomb`
endif
..... $O(\texttt{if}\ldots\texttt{endif}) = O(1)$

For $i = 1, 2, \ldots$ `num_samples` **do**
  `neighbourhood` ← a random neighbourhood in `neb_list` ..... $O(1)$
  If `num_convcomb` < `num_feats` **then**
    `data_shadow` ← `neighbourhood` ..... $O(1)$
  else
    Initialize `data_shadow` with empty set.
    For $v \in$ *neighborurhood* **do**
      `data_shadow` ← `data_shadow` ∪ {`shadow` random vectors around v with normal distribution. } ..... $O(d \cdot s)$
    endfor
    ..... $O(\texttt{for}\ldots\texttt{endfor}) = O(d \cdot s \cdot n)$
  endif
  ..... $O(\texttt{if}\ldots\texttt{endif}) = O(d \cdot s \cdot n)$
  $u = (u_1, \ldots, u_k) \leftarrow k$ random vectors ∈ `data_shadow` ..... $O(m)$
  $w = (w_1, \ldots, w_k) \leftarrow$ a random vector with positive values and $w_1 + w_2 + \ldots + w_k = 1$ ..... $O(m)$
  `generated_data` ← `generated_data` ∪ $\{\sum_{i=1}^{k} w_i \cdot u_i\}$ ..... $O(d \cdot m)$
endfor

$$O(\texttt{for}\ldots\texttt{endfor}) = O\Big(M \cdot \big(O(1) + O(d \cdot s \cdot n) + O(d \cdot m + 2 \cdot m)\big)\Big) = O\big(M \cdot (d \cdot s \cdot n + d \cdot m)\big) = O\big(d \cdot M \cdot (s \cdot n + m)\big)$$

.....

Returns new points as `generated_data`.

end

$$O(\texttt{begin}\ldots\texttt{end}) = O\Big(O(d \cdot c \cdot \log(c)) + 2O(1) + O\big(d \cdot M \cdot (s \cdot n + m)\big)\Big) = O\Big(d \cdot \big(c \cdot \log(c) + M \cdot (s \cdot n + m)\big)\Big)$$

.....

Appendix D

Curriculum Vitae

Saptarshi Bej

Date of birth:	24th July 1991
Place of Birth:	Bardhaman
Nationality:	Indian

Academic Background

2003-2007	Secondary education, RKMV Narendrapur, Kolkata, India
2008-2009	Higher secondary education, CMS high school, Burdwan, India
2009-2014	Integrated Bachelors and Masters degree (Mathematics), (with specialization in Graph and Network theory), Indian Institute of Science Education and Research, Kolkata, India (with KVPY Scholarship)
May-Jul, 2012	Internship on Number theory, Center For Excellence in Basic Sciences, Mumbai, India

Professional Experience

2014-2016	Project intern, Indian Institute of Science Education and Research, Kolkata, India
2016-2017	Research assistant, Paderborn Center for Advanced Studies, University of Paderborn, Germany
2018-present	Research assistant and PhD student, Department of Systems Biology and Bioinformatics, University of Rostock, Germany
2021-present	Guest scientist, Leibniz-Institute for Food Systems Biology, at the Technical University of Munich, Germany

Scholarships and Awards

2009-2014	KVPY scholarship, For highly motivated students pursuing basic science research, Awarded by the Department of Science and Technology, Government of India
2020	DAAD-Prize, for outstanding achievement of a foreign student, awarded by the University of Rostock

Academic experiences

Conference presentations:

- e-Med Meeting (Sept. 2018)
- International Conference on Systems Biology (Sept. 2018)
- Internation Joint Conference on Neural Networks (July 2021)

Project contributions:

- GB-X Map: Assessing gut-brain-cross-diseases risk (BMBF funded) *Performed RNA-Seq analysis and machine learning analysis for 500 patients suffering from Ulcerative Colitis and Schizophrenia*

- iRhythmics: Programming pacemaker cells for *in vitro* drug testing (ESF funded) *Created machine learning models for automatic annotation of rare cells from single cell data*

Teaching assistance:

- Bio-systems modelling and simulation *for 5 semesters, with a class of 30 students*
- Data science seminar with Python *for 3 semesters, with a class of 30 students, was responsible for designing, planning and evaluation of the course*

Thesis and project supervision:

- Narek Davtayn (Masters Thesis) *LoRAS: An oversampling approach for imbalanced datasets*
- Krithika Sayar Chand (Masters Thesis) *Advanced Image Analysis with Deep Learning*
- David Brauer (Masters Thesis) *ML-Assisted Construction of a Disease-Specific molecular Interaction Network from Human Blood RNA-Seq Data*
- Prashant Srivastava (project supervision) *Knowledge-graph creation through Biological Literature Mining*
- Cagri Kuzulu (Erasmus project supervision) *Comparative study on alternative methods of imbalanced learning*

Appendix E

Publications

E.1 Articles published/accepted in peer-reviewed journals relevant to this thesis

1. **Bej S.**, Davtyan N., Wolfien M., Nassar M., Wolkenhauer O. *LoRAS: an oversampling approach for imbalanced datasets* Mach Learn vol 110, 279301 (2021). https://doi.org/10.1007/s10994-020-05913-4

   Contributions: I developed the concept behind this study. I built the analytical results and developed the algorithm. I also decided upon the protocols of the benchmarking studies and performed the benchmarking experiments. I wrote the manuscript for this study and am the first author. This study is directly related to this thesis.

2. **Bej S.**, Schultz K., Srivastava P., Wolfien M., Wolkenhauer O. *A multi-schematic classifier-independent oversampling approach for imbalanced datasets*, IEEE Access, vol. 9, pp. 123358-123374, 2021 https://doi.org/10.1109/ACCESS.2021.3108450

   Contributions: I developed the concept behind this study. I developed the algorithm. I also decided upon the protocols of the benchmarking studies and performed the benchmarking experiments. I wrote the manuscript for this study and am the first author. This study is directly related to this thesis.

3. **Bej S.**, Galow A-M., David R., Wolfien M., Wolkenhauer O. *Automated annotation of rare-cell types from single-cell RNA-sequencing data through synthetic oversampling*. BMC Bioinformatics 22, 557 (2021) https://doi.org/10.1186/s12859-021-04469-x

   Contributions: The concept of this study was developed by Markus Wolfien. He along with Anne-Marie Galow and Robert David generated and pre-processed the data. I used the data to construct and test the sc-SynO tool. I wrote significant portions of the manuscript. This study is directly related to this thesis.

E.2 Other published/accepted articles in peer reviewed journals

1. Mucha S., Baurecht H., Novak N., **Bej S.** *et al. Protein-coding variants contribute to the risk of atopic dermatitis and skin-specific gene expression* J Allergy Clin Immunol. 2020; **145(4),** 1208-1218. https://doi.org/10.1016/j.jaci.2019.10.030

   Contributions: I participated in this study as a part of the GB-X Map project. I constructed gene-protein-interaction networks from genes and proteins identified by experts. Moreover, I used standard network analysis techniques to identify important genes and proteins from the constructed network.

2. Banerjee A., **Bej S.** *On extension of regular graphs* Journal of Discrete Mathematical Sciences and Cryptography, (2018) Vol **21:1,** 13-21, https://doi.org10.1080/09720529.2015.1085740

   Contributions: I developed the concept behind this study. I built the analytical results and also wrote the manuscript under the supervision of Prof. Banerjee.

3. Kok J., **Bej S.** *Coloring sums of extensions of certain graphs* Journal of Algebra Combinatorics Discrete Structures and Applications, (2016) Vol **5(1)** 19-27, http://dx.doi.org/10.13069/jacodesmath.349383

   Contributions: Johan Kok developed the concept behind this study. I built some of the analytical results and also wrote parts of the manuscript.

4. **Bej S.**, Steffen E. *Factors of edge-chromatic critical graphs: a brief survey and some equivalences* Lecture Notes of Seminario Interdisciplinare di Matematica, (2017) Vol **14** 37-48, http://dimie.unibas.it/site/home/in-evidenza/articolo3004582.html

   Contributions: Eckhard Steffen developed the concept behind this study. I performed the literature survey for the review portion of the article and also participated in writing the manuscript.

5. **Bej S.** *Hamiltonian cycles in annular decomposable Barnette graphs* https://arxiv.org/abs/2008.06671 (accepted for publication in the Journal of Discrete mathematical Sciences and Cryptography)

   Contributions: I developed the concept behind this study. I developed the theorems and proofs for showing some conditions such that ADB-AC graphs are Hamiltonian. This work is driven by my long term interest in graph theory, especially the Barnette's conjecture. The manuscript was also written by me.

6. **Nguinkal J., Bej S., *et al.*** *Comprehensive Characterization of Multitissue Expression Landscape, Co-Expression Networks and Positive Selection in Pikeperch* Cells. 2021; 10(9):2289, https://doi.org/10.3390/cells10092289

   Contributions: I contributed with the idea of using unsupervised ML approaches in the data analysis workflow and related consultations.

7. **Srivastava P., Bej S., *et al.*.** *Self-Attention-Based Models for the Extraction of Molecular Interactions from Biological Texts.* Biomolecules 2021, 11, 1591 https://doi.org/10.3390/biom11111591

   Contributions: This is a review article. I wrote a major portion of the publication.

E.3 Published/Accepted articles in conference proceedings

1. **Bej S.**, Srivastava P., Wolfien M., Wolkenhauer O. *Combining uniform manifold approximation with localized affine shadowsampling improves classification of imbalanced datasets* 2021 International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1-8, https://ieeexplore.ieee.org/document/9534072

   Contributions: I developed the concept behind this study. I developed the algorithm. I also decided upon the protocols of the benchmarking studies and performed the benchmarking experiments. I wrote the manuscript for this study and am the first author. This study is directly related to this thesis.

E.4 Articles in peer-review and other independent articles

1. **Bej S.**, Sarkar J., Biswas S., Mitra P., Chakrabarti P., Wolkenhauer O. *Identication and epidemiological characterization of Type-2 Diabetes sub-population using an unsupervised machine learning approach* (submitted to Nutrition and Diabetes (Nature))

   Contributions: The concept of this study was developed by Jit Sarkar and myself. Jit Sarkar along with his collaborators generated and pre-processed the data. I used the data to construct a feature-distributed clustering method that could identify significant patient clusters. I wrote significant portions of the manuscript.

2. **Bej S.**, Wolkenhauer O. *The timing of contact restrictions and pro-active testing balances the socio-economic impact of a lockdown with the control of infections* https://doi.org/10.1101/2020.05.08.20095596

   Contributions: The concept of this study was developed by Olaf Wolkenhauer. I helped in building Python based models to simulate and visualise several scenarios.

Moreover, I helped in the interpretation of the results. I also wrote significant portions of the manuscript.

3. Uellendahl-Werth F., Maj C., Borisov O., Matthias Wacker E., **Bej S.**, Wolkenhauer O., Ellinghaus D., *et al. Cross-tissue transcriptome-wide association studies of 885,176 individuals and seven diseases of the gut-brain axis identify susceptibility genes shared between schizophrenia and inflammatory bowel disease*

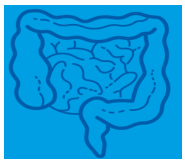
   Contributions: I contributed to the workflow of using unsupervised learning, supervised learning and feature selection techniques in transcriptomic analysis.

Presented poster in E-Med meeting 2018, Berlin

GB-X Map: Causal Pathways and Motifs Common in IBD and Schizophrenia

David Ellinghaus[1], Saptarshi Bej[2], Carlo Maj[3], Markus Wolfien[2], Oleg Borisov[3], Sören Mucha[1], Per Hoffmann[4,5,6,7], Franziska Degenhardt[6,7], Andrea Bagnacani[2], Stefan Schreiber[1], Peter Michael Krawitz[3], Markus Nöthen[6,7], Olaf Wolkenhauer[2]

(1) Institute of Clinical Molecular Biology, Christian-Albrechts-University of Kiel, Kiel, Germany, (2) University Rostock, Institute of Computer Science, Department of Systems Biology and Bioinformatics, Ulmenstraße 69, 18057 Rostock, Germany, (3) Institute for Genomic Statistics and Bioinformatics, University Hospital Bonn, Rheinische Friedrich-Wilhelms-Universität Bonn, Bonn, Germany, (4) Human Genomics Research Group, Department of Biomedicine, University of Basel, Basel, Switzerland, (5) Institute of Medical Genetics and Pathology, University Hospital Basel, Basel, Switzerland, (6) Institute of Human Genetics, University of Bonn, Bonn, Germany, (7) Department of Genomics, Life & Brain Center, University of Bonn, Bonn, Germany



Abstract: Psychiatric comorbidity in inflammatory bowel disease (IBD) is well known, with a higher incidence of schizophrenia (SCZ) in IBD cohorts compared with controls (incidence rate ratio [IRR] = 1.64) [1]. An overlap for common GWAS loci has been observed, in particular for the MHC complex on chromosome 6p21. The objective of the **GB-XMAP (Gut-brain cross-disease map)** Vernetzungsfonds project is to decipher the mechanisms of action of disease-predisposing GWAS loci shared between ulcerative colitis (UC) and SCZ. The e:Med consortia "Sysinflame" and "IntegraMent" have generated a wealth of genome-wide genotyping and gene expression data that are used for exploration.

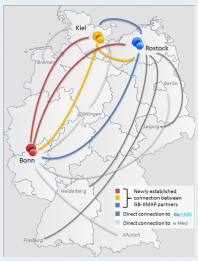
Method and data

Genome-wide association studies have identified >300 risk loci for IBD and SCZ. The GB-XMAP project, a new strategic alliance between two e:Med centres in Bonn and Kiel as well as one de.NBI node in Rostock, uses transcriptome (RNA-seq and array experiments) of whole-blood samples of 500 UC, 500 SCZ patients and 1,500 healthy controls in combination with GWAS SNP array data availabel from >20,000 UC and >30,000 SCZ patients and >79,000 healthy controls.

• Objective 1: We estimate the genetically regulated component of expression of potential risk genes of established UC and SCZ GWAS loci for a wide range of tissues by means of tissue specific cis-eQTL models followed by transcriptome wide association studies (TWAS).

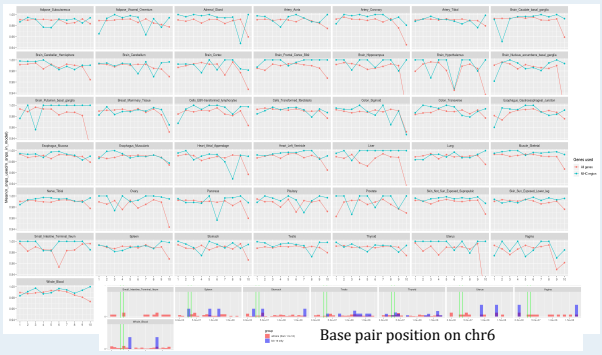
• Objective 2: We construct a single cross-disease interaction map, derive a common regulatory core, and predict disease gene signatures using in silico model simluation to create a multidimensional model.

	Genome-wide data		Transcriptome-wide data	
Technology	Immunochip	GWAS	RNA-Seq	HumanHT-12 v4
Ulcerative Colitis	14,513	6,945	500	-
Schizophrenia	1,000	>34,000	500	-
Cases total	15,513	>40,945	1000	-
Controls Kiel+Bonn	>34,000	>45,000	250+250	250+750



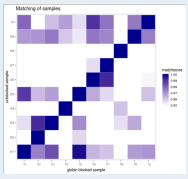
TWAS Benchmark

Expression imputation works for non-MHC/MHC genes

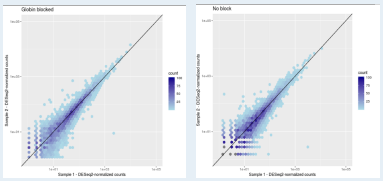


Improved RNA-Seq Analysis

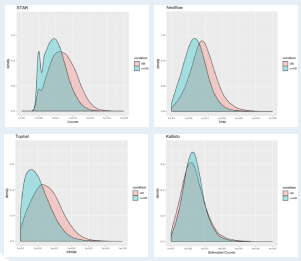
50-80% of seq. reads map to globin RNA in whole blood samples. Efficient gene-expression analysis of whole-blood samples with globin block (GB) oligos from Lexogen:



Heatmap of normalized Pearson's rho values (matchscore) shows that globin blocking (GB) RNA-seq has no impact on the correlation of "technical" replicates (GB versus non-GB). RNA-seq GB and non-GB samples from same individuals perfectly match, shown for a subset of 10 individuals.



Correlation plots showing the strong correlation of counts normalized by DESeq2 from two healthy individuals. GB does not significantly increase the variance (biological variability) or occurence of outliers in the data.

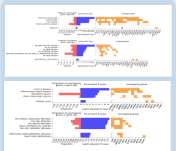


Most common RNA-seq alignment tools profit from reduced globin transcripts: More transcripts with increased counts for differential expression analysis. STAR: raw counts; Nextflow: STAR&stringTie; Tophat: FPKM (Fragments Per Kilobase Million); Kallisto: Pseudoalignment.

Network Analysis

• GWAS Network for IBD
• 970 Nodes
• 3802 Edges
• All feed forward loops and feedback loops involving at least two of the input genes included
•Tool used: Bisogenet [3]

• Sub-networks extracted by MCODE Algorithm
• MCODE Algorithm extracts relatively dense motifs from a larger network
• After integrating the RNA-sequencing data with the GWAS data the networks will be updated



•Functional Enrichment of the motifs extracted by MCODE
• Tool used for Enrichment Analysis: FUMA-GWAS [2]

References:
[1] Bernstein et al. Increased Burden of Psychiatric Disorders in Inflammatory BoweDisease. Inflamm Bowel Dis. (2018). July 7 doi: 10.1093 ePub ahead
[2] Watanabe, K., Taskesen, E., van Bochoven, A., & Posthuma, D. (2017). Functional mapping and annotation of genetic associations with FUMA. Nature Communications, 8(1), 1826
[3] Martin, A., Ochagavia, M. E., Rabasa, L. C., Miranda, J., Fernandez-de-Cossio, J., & Bringas, R. (2010). BisoGenet: a new tool for gene network building, visualization and analysis. BMC Bioinformatics, 11, 91. https://doi.org/10.1186/1471-2105-11-91

Presented poster in LSB Munich (2021)

S3

Classifier Independent of Oversampling for the Classification of Imbalanced Datasets

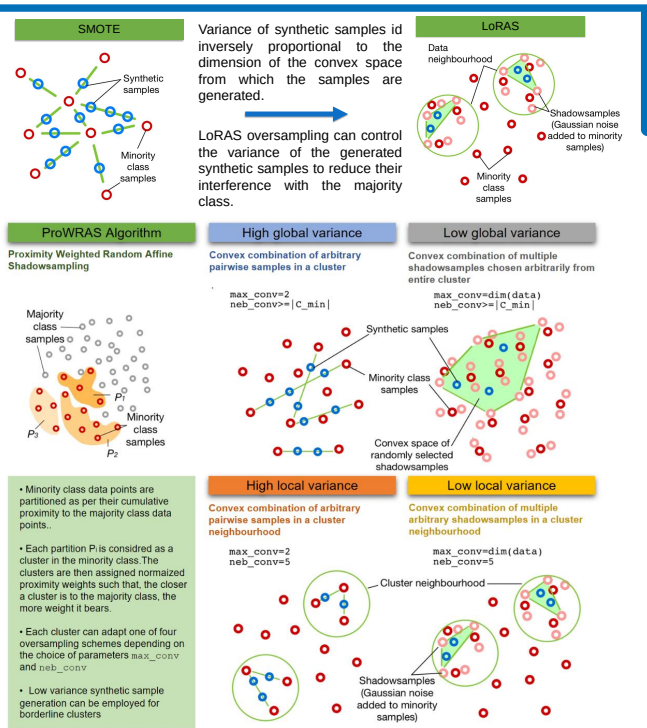
S. Bej[1,2] , P. Srvivastava[1,2], and O. Wolkenhauer[1,2]

[1]Leibniz Institute for Food Systems Biology at TUM, [2]University of Rostock

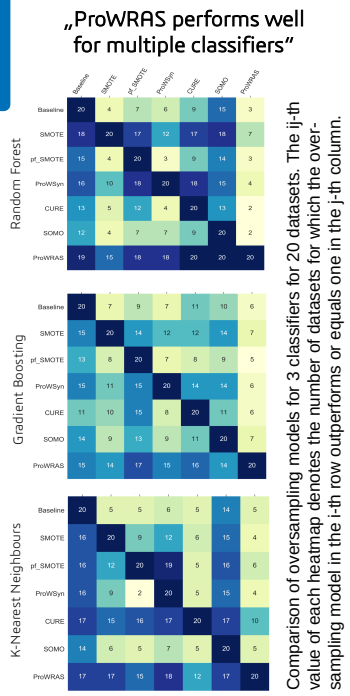
Research Question

Oversampling is a data pre-processing approach for handling imbalanced datasets. However, more than a hundred oversampling approaches perform differently when integrated with different machine learning classifiers. Given a dataset and a classifier, how do we make sure that a chosen oversampling model is effective?

Method



Results



Comparison of oversampling models for 3 classifiers for 20 datasets. The ij-th value of each heatmap denotes the number of datasets for which the over-sampling model in the i-th row outperforms or equals one in the j-th column.

Conclusion

Our ProWRAS algorithm outperforms state of the art oversampling algorithms in a classifier independent manner. Our application based studies on automated rare cell type annotation is promising. With method firmly established, we are using it in different applications.

S.Bej et al. LoRAS: an oversampling approach for imbalanced datasets. Machine Learning (2021) 110:279-301
S.Bej et al. A multi-schematic classifier-independent oversampling approach for imbalanced datasets. Accepted for publication in IEEE Access
S.Bej et al. Automated annotation of rare-cell types from single-cell RNA-sequencing data through synthetic oversampling – submitted (available on bioRxiv)

LSB · SYSTEMS BIOLOGY BIOINFORMATICS ROSTOCK · Universität Rostock Traditio et Innovatio · Leibniz Gemeinschaft · Technische Universität München

Presented poster in CTNR Summer School (2021)

Universität Rostock Traditio et Innovatio

SYSTEMS BIOLOGY BIOINFORMATICS ROSTOCK

iRhythmics

Explainable transcriptome (?) analyses

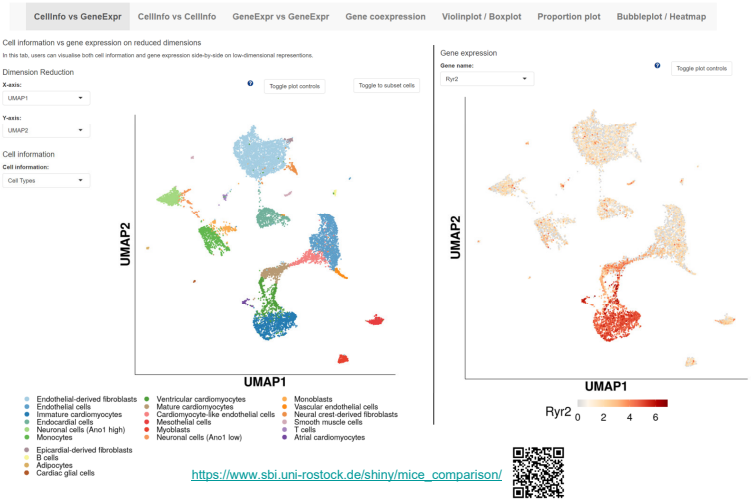
Make patterns and relationships visible through bioinformatics

Computer-aided investigations are irreplaceable in research. Within the Chair of Systems Biology and Bioinformatics, there are different approaches in this regard, for example based on artificial intelligence or network and pathway modeling, in order to simplify the explainability and interpretation of data. Not only are the results of the CTNR community presented, but important fundamentals of these different approaches are also illustrated using interactive examples. With the help of this poster you can actively experience and discover the most modern, computer-aided techniques. All you need is a device with a QR code reading function and you're ready to go.

Explore single-cell RNA sequencing data

Single-cell transcriptome analyzes enable an improved degree of resolution within cell type characterization.[1,2] However, the analysis is very complex and multi-layered, which is why easy access to the interpretation of the data is of great importance. RShiny is used as a well-suited solution. Explore yourself!

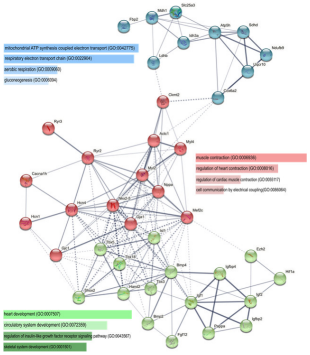
Mice strain comparison



https://www.sbi.uni-rostock.de/shiny/mice_comparison/

Deep dive in networks

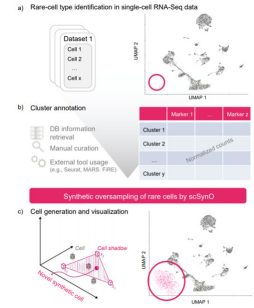
Interactions between proteins or RNA transcripts can be compactly visualized and analyzed with the help of networks. In addition, they can serve as a knowledge base for a broad variety of data. Exemplarily, these methods are demonstrated via AIR[3], which is based on Minerva.



https://air.elixir-luxembourg.org/minerva/

Influences of clustering

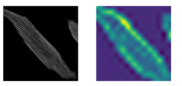
Clustering cells into certain groups is an essential part of many analyzes. Here, we present LoRAS[4,5] an algorithm to adjust for imbalance of data and improve accuracy.

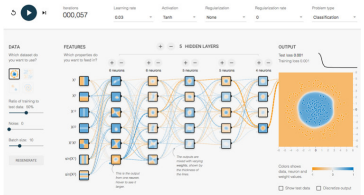


https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Train neurons with data

Image analysis using deep learning has become an indispensable part of research. But how do these neural networks actually learn, and more importantly, what exactly do they learn? Here, we link to a showcase of TensorFlow, which is a software framework used as a basis for our sarcomere studies.

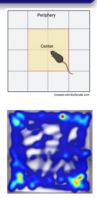




playground.tensorflow.org

Finding mice in a maze

Behavioral analysis has made great progress through the use of deep learning technologies, especially in the field of video analysis. With the help of DeepLabCut, which is an AI-based analysis of different parameters in an unbiased manner. Try to label images yourself!



https://contrib.deeplabcut.org/

Literature

1 Wolfien M, Galow AM, David R. 2021. Single-cell RNA-Sequencing procedures and data analysis. *Exon Publications*.
2 Wolfien M, Galow AM, *et al.* 2020. Single-Nuclei Sequencing of entire Mammalian Hearts: Cell Type Comparison and Velocity. *Cardiovascular Research*
3 Serhan C, Gupta SK, *et al.* 2020. The Atlas of Inflammation Resolution (AIR). *Molecular Aspects of Medicine*.
4 Bej S, Davtyan N, *et al.* 2021. LoRAS: An oversampling approach for imbalanced datasets. *Machine Learning*.
5 Bej S, Galow AM, *et al.* 2021. Automated anno -tation of rare cell types from sc-RNA-Seq data through synthetic oversampling. *BioRxiv*.

Tools available via: github.com/COSPOV

Markus Wolfien, Maximilian Hillemanns, Daniel Jenderny, Saptarshi Bej, Olaf Wolkenhauer
www.sbi.uni-rostock.de

MV tut gut.

Künstliche Intelligenz KI tut gut

EUROPÄISCHE UNION Europäischer Sozialfonds
Europäische Fonds EFRE, ESF und ELER in Mecklenburg-Vorpommern 2014-2020
Funded by:
European Social Funds
(ESF)-14-BM-A55-0027/18

Declaration of Authorship

I hereby declare that this thesis was independently composed and authored by myself.

All content and ideas drawn directly or indirectly from external sources are indicated as such. All sources and materials that have been used are referred to in this thesis.

The thesis has not been submitted to any other examining body and has not been published.

Signed: Saptarshi BEJ
Date: February 21, 2022
Place: Rostock