ECE414: Combating Class Imbalance in a Clinical Dataset

Jongoh (Andy) Jeong
The Cooper Union, New York, NY 10003
jeong2@cooper.edu

Abstract

Class imbalance is an issue commonly observed in real-world data, particularly more often found in clinical datasets. In this investigation of such a frequent problem in a clinical setting, we compare and discuss the performances of such techniques to address the issue with a few classifiers for a binary classification task. We approach the task by exploiting the Parkinson's dataset from the UCI Machine Learning Repository to generate data of various mixture (imbalance) ratios, performing the classification task with Naive Bayes, Logistic Regression and SVM classifiers, and evaluate their performances relative to each other. We observe that applying a combined resampling algorithm (SMOTEENN) on an RBF-kernel SVM classifier yields the best prediction level.

1 Introduction

1.1 Background

In a typical machine learning classification problem, it is important that datasets are well-balanced in order to generate robust prediction models, especially in medical statistics fields. Conventional approaches to modelling realistic data do not take into account imbalance in a dataset and it tends to be the main cause for the reduced generalization of the model. When it comes to dealing with smaller subsets of data, the cost of predicting sub-optimally increases, and the resulting predictive model loses its robustness because the weights for the data points are treated almost as equally. Such cases are common in medical datasets where high-risk patients tend to be smaller subsets of data [6, 8].

In this regard, we take an openly-available clinical dataset from the UC Irvine Machine Learning Repository to apply and compare techniques to handle the imbalance problem by observing the performance levels for datasets of various proportions of positive and negative classes. The particular dataset, Parkinson's Disease Telemonitoring dataset, allows for thresholding the target labels (continuous) to formulate the problem as a binary classification problem. For the simplicity of observing the effects of the applied methods, we only consider for binary class cases.

There are a number of existent techniques to address the imbalance issue [4]. For instance, common approaches include under-sampling the majority class in which samples from the dominant subset of the data are taken, and over-sampling in which samples from the non-dominant subset of the data are taken, in order to ensure balance between the dominant and non-dominant groups. Each has advantages and disadvantages – under-sampling allows for reduced computation time, but there remains a high likelihood of selecting non-representative or biased subsamples from the dominant group, and over-sampling allows for all data points to be used but at a cost of expensive computation for a large dominant subset of data. Such techniques pose an interesting question of the degree of how well they contribute to performance gain in comparison to the baseline level. There are quite a number of use cases for one or two techniques to combat the class imbalance problem, but they tend to only discuss the particular technique that works best for their applications. In this

investigation, we seek to thoroughly observe how each of the available methods compares with one another, applied to the identical dataset. As these methods are known to apply differently for each specific application, it should be noted that the performance varies by the properties of the dataset [6].

1.2 Approach

We first approach the problem by pre-processing the obtained dataset. This dataset contains two attributes that can be designated as labels – motor and total UPDRS scores, which can be thresholded at an arbitrary value such that values greater than the criterion are re-labeled as positive class and the rest as negative. Using this approach, we were able to sweep through all ranges of values for both attributes and obtain imbalance ratios from 0 to 100%, where an imbalance ratio is defined as a proportion of minority to majority subsets in data represented as a percentage. The features are then all normalized by Equation 1, where each data point x is subtracted by the mean attribute value μ and divided by the standard deviation of the same attribute σ. After applying grid search for hyper-parameters (where applicable), we visualize the 18-dimensional feature space with principal component analysis (PCA) (N=2). For cross validation, we employed a stratified 10-fold cross validation in order to ensure all data points are tested from a model trained on the rest, and output the average of each evaluation metric over all folds.

$$z = (x - \mu)/\sigma \tag{1}$$

For handling the imbalance, we explore a number of available techniques with scikit-learn and imbalanced-learn Python libraries, as listed in Table 1. For commonly known under- and over-sampling methods, we explore specific algorithms that realize such resampling techniques, such as KMeans SMOTE and Cluster Centroids. Then we observe their performances in comparison to combined and decision-tree approaches, as well as some algorithmic variations to the conventional methods.

	Technique	Library	Type
1	Under-Sampling the Majority	imbalanced-learn	Resampling
2	Over-Sampling the Minority	imbalanced-learn	Resampling
3	Synthetic Sample Generation	imbalanced-learn	Resampling
4	Combined-sampling	imbalanced-learn	Resampling
5	Ensemble Classifier	imbalanced-learn	Ensemble Learning
6	Different Error Cost (SVM)	LIBSVM, scikit-learn	Algorithmic
7	One-Class Learning (SVM)	LIBSVM	Algorithmic

Table 1: General Imbalance Handling Approaches

2 Dataset and Features

The dataset used throughout the experiments is the Parkinson's Disease (PD) Telemonitoring dataset from the UCI Machiine Learning Repository, which consists of 5,875 speech recordings from 32 PD patients whose age ranges from 36 to 85, collected over 6-month period by six U.S. medical centers in the context of the Tsanas et al. study [11]. It consists of 18 attributes with 2 different attributes for labels, based on six sustained phonation of the vowel /a/ for each patient. The voice recordings of the subjects were obtained at weekly intervals for the 6-month duration of the study, while the motor and total UPDRS scores (labels) were assessed only three times (and thus the missing scores are estimated by linear interpolation [11]). This dataset contains timestamp for each row, but for the scope of this investigation we drop the temporal aspect. Multiple rows contain attribute values of the same individual since more than one recording are made, but we assume that each row is independent. The list of attributes are shown in Table 2. For pre-processing, we generate imbalanced datasets of various majority-to-minority proportions by thresholding, as in Table 6 in Appendix A. The rows with bolded imbalance ratio (%) indicates the datasets we used throughout the experiments for comparison – 99% indicates nearly perfectly-balanced, while 2.5% indicates severe disproportion between majority and minority subsets.

Attributes	age	sex	Jitter(%)
	Jitter(Abs)	Jitter:RAP	Jitter:PPQ5
	Jitter:DDP	Shimmer	Shimmer(dB)
	Shimmer:APQ3	Shimmer:APQ5	Shimmer:APQ11
	Shimmer:DDA	NHR	HNR
	RPDE	DFA	PPE
Labels	motor_UPDRS	total_UPDRS	

Table 2: Attributes and Labels for PD Telemonitoring Dataset

We work with one of the severe imbalanced cases by thresholding total UPDRS score attribute at the value of 8. The features are reduced to 2-dimensional space with principal component analysis for ease of visualization (see Figure 1). As marked by green and red colors, the dominant subset (green) outnumbers the minority such that red points are almost not present at all.

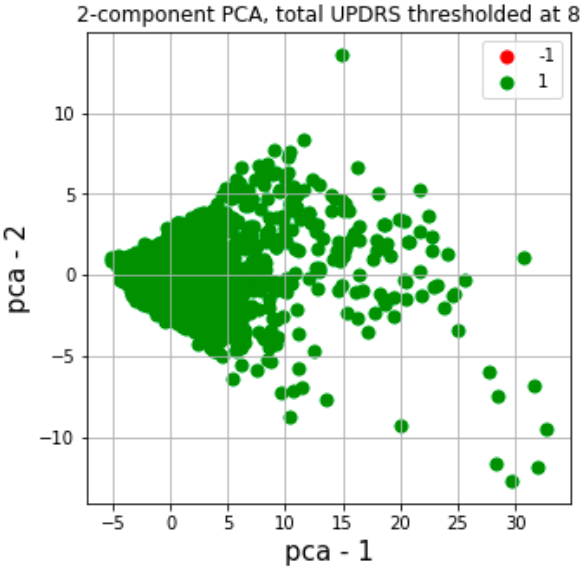


Figure 1: PCA - 2 component analysis

3 Methods

3.1 Classifiers

Because the performance of a classifier can vary by the type of classifier used and the evaluation metric, we compare the yields for three different ones, which tend to produce a good enough performance in general cases – Gaussian Naive Bayes, Logistic Regression and support vector machines (SVM).

3.1.1 Naive Bayes

A Naive Bayes model is a classifier based on Bayes' Theorem with the assumption that all features are independent of each other. Recall Bayes Theorem as stated in Equation 2. We predict the likelihood P(X|C_k) using this rule, where we exploit the independence assumption, as in Equation 3. We then use the relationship (posterior $\propto$ likelihood * prior) to estimate the posterior probability [3].

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)}, \text{for k = 1,2, ..., K classes} \tag{2}$$

$$P(X|C_k) = P(x_1, ..., x_n|C_k) = \prod_{i=1}^{n} P(x_i|C_k) \tag{3}$$

Naive Bayes, which involves class prior and class conditional probabilities, can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution ('Gaussian Naive Bayes') as it only requires mean and standard deviation estimation for determining its distribution.

3.1.2 Logistic Regression

Logistic regrssion is similar to linear regression, but with a logistic (sigmoid) function for activation. It makes a hypothesis about how likely it will be the target, given an input X and some parameter θ, such that it can represent in equations as shown below [10].

$$\begin{gathered} h_\theta = P(Y = 1|X; \theta) \\ P(Y = 1|X; \theta) + P(Y = 0|X; \theta) = 1 \\ P(Y = 0|X; \theta) = 1 - P(Y = 1|x; \theta) \end{gathered}$$

3.1.3 SVM

The objective of an SVM model is to maximize the marginal distance between the support vectors and the hyperplane in the feature dimensions. The general optimization problem is mathematically defined below [5].

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\ \Sigma_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}_i^T\phi(\mathbf{x_i}) + b) \geq 1 - \xi_i, \ \ \xi_i \geq 0 \end{aligned} \tag{5}$$

We also experiment with different error cost (DEC) approach, where the cost C is modified to weigh each class differently, as shown below. One-learning is an extension of this cost structure modification where one assigns zero cost for the majority class samples and $\frac{1}{N^+}$ to the minority samples, thereby only learning from the positive target class (minority) [1, 7, 9]. To determine the balanced weights for the two classes, compute_class_weight function from the scikit-learn library is used.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C^+\ \Sigma_{i|y_i=+1}^{l}\xi_i + C^-\ \Sigma_{i|y_i=-1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}_i^T\phi(\mathbf{x_i}) + b) \geq 1 - \xi_i, \ \ \xi_i \geq 0 \end{aligned} \tag{6}$$

3.2 Techniques

Resampling. We exploit the algorithms for resampling provided in 'imbalanced-learn' Python library. They support under-sampling with RandomUnderSample (randomly under-sampling the majority class without replacement), ClusteredCentroids (replacing majority clusters with KMeans centroids), NearMiss (3 versions; each is based on nearest neighbors algorithm with different heuristics), TomekLinks (using links between samples of different classes), EditedNearestNeighbours (based on nearest neighbors algorithm but also removes samples that don't belong in the 'neighborhood'), AllKNN (NN algorithm, but the number of neighbors is increased at each iteration), CondensedNearestNeighbour (using a 1-NN rule to iteratively decide if a sample should be removed or not) and OneSidedSelection (uses TomekLinks to remove noisy samples and 1-NN rule to all samples) algorithms. For over-sampling, we use RandomOverSampler, SMOTE (generate synthetic samples by linear interpolation), ADASYN (SMOTE, but generating samples next to the original samples that are misclassified using a k-NN classifier), BorderlineSMOTE (generating synthetic samples from borderline samples), SVMSMOTE (using SVM to make new samples), KMeansSMOTE (applying KMeans clustering before SMOTE), SMOTENC (for nominal and continuous features), and for

combined methods, SMOTEENN (over-sampling with SMOTE and cleaning with Edited Nearest Neighbours) and SMOTETomek (over-sampling with SMOTE and cleaning with Tomek links).

Ensemble Learning. Apart from resampling from the data, we approach differently by exploring other classifiers based on decision trees, which classifies effectively with feature selection. These available classifiers – BalancedBaggingClassifier, RUBoostClassifier, EasyEnsembleClassifier – all work in an ensemble, or multiple trees work to yield the best results by majority voting. BalancedBaggingClassifier combines a bagging classifier (buildling several estimators on different randomly selected subsets) with a random undersampler. EasyEnsembleClassifier is a group of AdaBoost learners trained on different balanced bootstrap samples, where balancing is from under-sampling, and RUBoostClassifier is similar but the class balancing is handled at each iteration of the boosting algorithm (AdaBoost).

4 Experiments

4.1 Parameters

Most parameters for the functions from the specified library (see Table 6) are set to default. For example, algorithms involving nearest neighbors use the default number of neighbors for computing the distances. For the classifiers, the gaussian naive bayes model sets the parameter 'var_smoothing' to 1e-9, which determines the portion of the largest variance of all features that is added to variances for calculation stability. The logistic regression model is set to use 'lbfgs' solver and use L2-nnorm penalty. For SVMs, the only parameters manipulated include the cost, regularization, type of SVM (C-SVC, kernel types (Linear or RBF), and weights (for DEC). Grid search is applied for C and gamma of range between 2^{-5} and 2^{6} and cross validation is performed with 10 folds, stratified where possible.

4.2 Evaluation Metrics

In a binary classification task in which a training set consists of different numbers of samples from either class, it is very common to observe that a classifier may be biased towards the majority class. When such a model are applied to a test set that is imbalanced in the same direction, it may yield an optimistic accuracy estimate, thereby increasing the accuracy metric as close to the proportions of the samples. In an imbalanced dataset, even with techniques of re-sampling and weighing each class differently in its cost, there still remains potential bias towards the majority and therefore there are no generic safeguards against reporting an optimistic accuracy estimate [2]. Since the conventional *accuracy* metric may not capture the performance of the classifier precisely, a different set of evaluation metrics need to be used to assess the classifier's performance, namely *balanced accuracy, F-1 score, geometric mean*, and *area under receiver operating characteristic curve (AUROC).*

Balanced accuracy, as opposed to accuracy $\left(= \frac{TP+TN}{TP+TN+FP+FN}\right)$, takes into account the recalls for each class (see Equation 7 for formula and Table 3 for confusion matrix). If the dataset is balanced well enough, balanced accuracy reduces to the conventional accuracy, while it is lower if imbalanced because the conventional accuracy exploits the imbalanced state in data. Using balanced version of accuracy for evaluation helps mitigate this issue because it considers each class symmetrically [2]. In addition, for binary classification task on an imbalanced dataset, sensitivitiy and specificity do not bear significance as sensitivity for one class equals specificity of the other, and vice versa for two classes [12].

$$\text{balanced accuracy} = \frac{1}{2}\left[\frac{TP}{P} + \frac{TN}{N}\right] \tag{7}$$

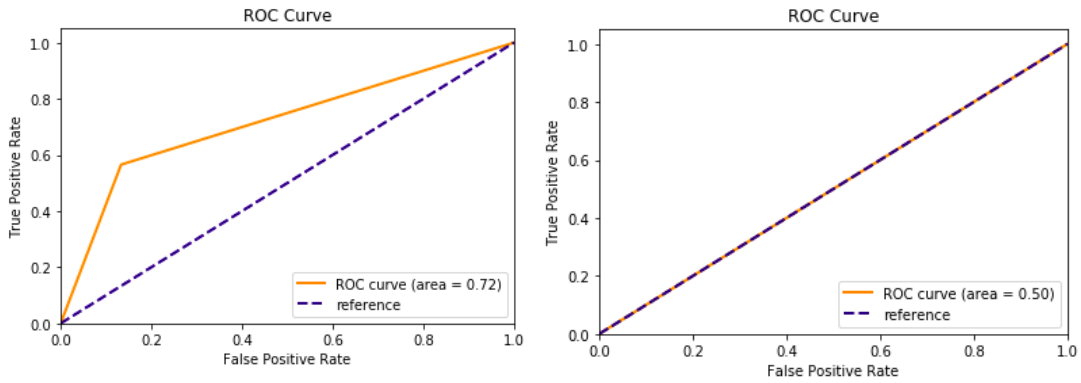
Often one way to improve the performance of a classifier is to change the performance metric. Since the common approach of using the conventional accuracy measure can sometimes be misleading, it is best to evaluate based on measures that take into account both classes equally. In this imbalanced data setting, we therefore compare the performance by balanced accuracy, F1, G-Mean, and AUROC (AUC) throughout the experiments.

		Predicted	Predicted
		Positive	Negative
Actual	Positive	**TP**	**FN**
Actual	Negative	**FP**	**TN**

Table 3: Confusion Matrix

4.3 Results

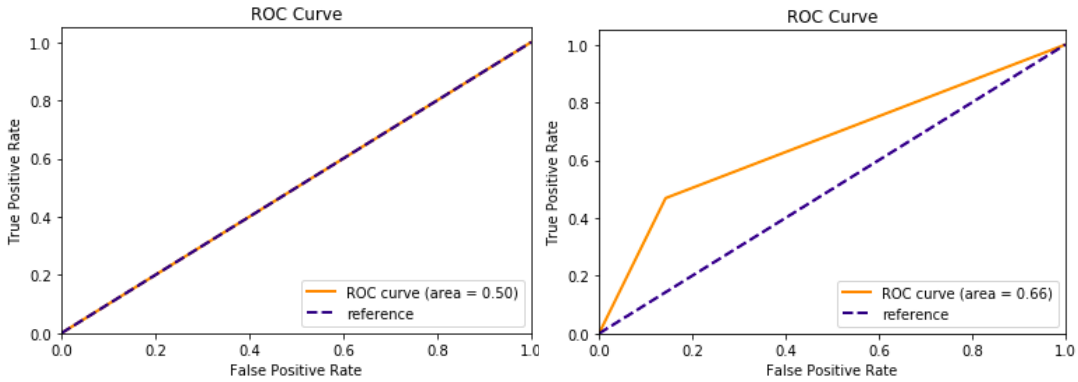
The results from applying the techniques listed in Tables 4 and 5 show that although various algorithms for resampling from the data can improve the performance from the baseline (Figures 2 3), it is heavily dependent upon which modification technique – classifier – evaluation metric combination one employs. The bolded results highlight ones that are significant for the particular evaluation metric. For under-sampling technique in general, this dataset was almost perfectly classified with AllKNN technique with RBF-kernel SVM evaluated on the balanced accuracy measure. For over-sampling, the random sampling method with RBF-kernel SVM yielded consistently good results. Because each metric can vary greatly due to whether it takes into account false positives and false negatives, such consistently high results are impressive to observe. For a more complex combined resampling technique,SMOTEENN applied to RBF-kernel SVM predicted all targets correctly for each of the fold (see Figure 5). This shows it may be desirable to use a combined resampling approach, although it may depend significant on the type of classifier. For decision tree-based ensemble learning, balanced bagging classifier was the best of the three explored on the AUC and G-Mean measures (see Figure 4). Tackling the error cost for each class (DEC) in SVMs was effective on the balanced accuracy and F1 score metrics, but not on the rest. DEC technique did not seem to be consistent due to the presence of imbalance in the data, and one-class learning approach was not able to reach good results.



(a) Gaussian Naive Bayes ROC

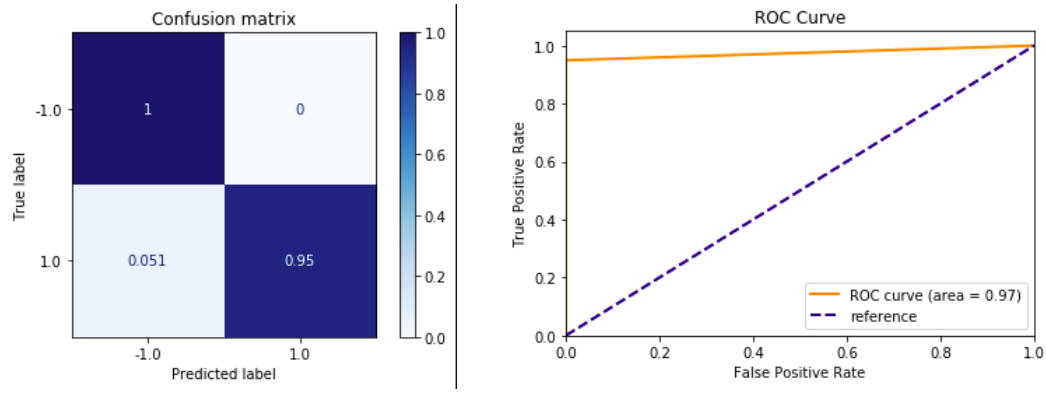
(b) Logistic Regression ROC

Figure 2: Baseline - 1



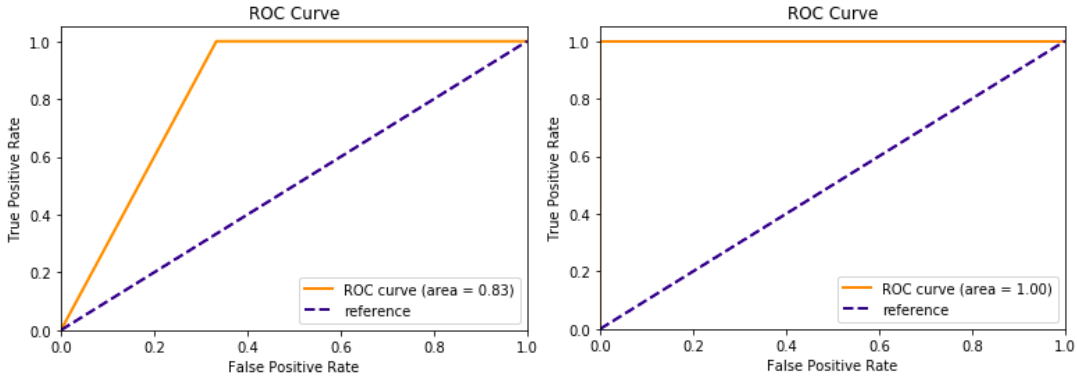
(a) SVM Linear Kernel ROC (b) SVM RBF Kernel ROC

Figure 3: Baseline - 2



(a) BalancedBaggingClassifer Confusion Matrix (b) BalancedBaggingClassifer ROC

Figure 4: Balanced Baggging Classifier Ensemble Learning (Highest AUC, G-Mean)



(a) AllKNN with SVM ROC (Highest balanced acc.) (b) SMOTEENN with SVM ROC (Highest metrics)

Figure 5: Best Performances with Resampling and SVM on PD dataset

Sampling\Classifier		SVM (Linear)	SVM (RBF)	Gaussian Naive Bayes	Logistic Regr.
Baseline	-	0.9762 0.5000 0.9880 0	0.9762 0.5000 0.9880 0	0.5229 0.7337 0.7366 0.7180	0.9796 0.5000 0.9880 0
Under-Sampling	RandomUnderSampler	0.7409 0.7274 0.6981 0.7165	0.7407 0.6657 0.6290 0.5969	0.7287 0.7003 0.6237 0.6694	0.7753 0.7642 0.8116 0.7596
	ClusteredCentroids	0.8317 0.8176 0.8043 0.8115	0.7726 0.6802 0.6859 0.6111	0.7933 0.7364 0.6563 0.6989	0.8413 0.8442 0.8615 0.8442
	NearMiss(1)	0.7075 0.6924 0.7189 0.6776	0.6735 0.6026 0.5794 0.5018	0.6765 0.6279 0.7035 0.5652	0.7646 0.7685 0.7667 0.7652
	NearMiss(2)	0.8806 0.8936 0.8716 0.8524	0.8655 0.7864 0.8345 0.7359	0.9232 0.9164 0.9798 0.9143	0.9231 0.8800 0.9167 0.8718
	NearMiss(3)	0.8408 0.8157 0.8336 0.8036	0.7148 0.6383 0.6499 0.5766	0.6377 0.6143 0.5202 0.5802	0.7786 0.7836 0.7869 0.7816
	TomekLinks	0.9753 0.5000 **0.9875** 0	0.9753 0.5000 **0.9875** 0	0.5323 0.7354 0.7354 0.7186	0.9692 0.5000 0.9843 0
	EditedNearesetNeighbours	0.9748 0.5000 0.9872 0	0.9746 0.5000 0.9871 0	0.5246 0.7432 0.7432 0.7258	0.9700 0.5000 0.9848 0
	AllKNN	0.9762 0.5000 0.9880 0	**0.9955** 0.8333 0.9955 0.8165	0.5247 0.7408 0.7432 0.7260	0.9698 0.5000 0.9847 0
	CondensedNearestNeighbour	0.7298 0.5000 0.8438 0	0.8314 0.7371 0.8919 0.7025	0.6126 0.6313 0.5760 0.5955	0.6542 0.5000 0.7910 0
	OneSidedSelection	0.9733 0.5000 0.9864 0	0.9036 0.7982 0.9927 0.7732	0.5229 0.7244 0.7155 0.7048	0.9697 0.5000 0.9846 0

Table 4: Under-sampling results for 2.5% Imbalance Ratio (total UPDRS score)
Balanced Accuracy / AUC / F1 / G-Mean in order

Sampling\Classifier		SVM (Linear)	SVM (RBF)	Gaussian Naive Bayes	Logistic Regr.
Over-Sampling	RandomOverSampler	0.7962	**0.9701**	0.7256	0.8094
		0.7951	**0.9686**	0.6863	0.8079
		0.7889	**0.9695**	0.6060	0.8018
		0.7946	**0.9682**	0.6537	0.8069
	SMOTE	0.8247	0.7744	0.7414	0.8119
		0.8213	0.5881	0.6996	0.8112
		0.8118	0.7086	0.6188	0.8075
		0.8197	0.4198	0.6665	0.8107
	ADASYN	0.7887	0.7513	0.7045	0.7972
		0.7860	0.5018	0.6629	0.7969
		0.7754	0.6690	0.5645	0.7938
		0.7845	0.0592	0.6231	0.7958
	BorderlineSMOTE	0.8374	0.7980	0.7518	0.8252
		0.8301	0.6606	0.7055	0.8229
		0.8166	0.7469	0.6249	0.8156
		0.8268	0.5667	0.6720	0.8214
	SVMSMOTE	0.8545	0.7825	0.8137	0.8583
		0.8527	0.6143	0.7670	0.8575
		0.8473	0.7220	0.7156	0.8545
		0.8519	0.4781	0.7454	0.8569
	KMeansSMOTE	0.9176	0.8405	0.8710	0.9159
		0.9111	0.7653	0.8363	0.9115
		0.9052	0.8102	0.8067	0.9065
		0.9090	0.7284	0.8222	0.9097
	SMOTENC	0.8285	0.7739	0.7499	0.8118
		0.8248	0.5864	0.7114	0.8112
		0.8151	0.7078	0.6409	0.8078
		0.8231	0.4157	0.6837	0.8107
Combined	SMOTEENN	0.8416	**1**	0.7477	0.8310
		0.8365	**1**	0.7039	0.8263
		0.8240	**1**	0.6254	0.8128
		0.8349	**1**	0.6737	0.8248
	SMOTETomek	0.8069	0.8585	0.7395	0.8261
		0.8037	0.8455	0.6978	0.8525
		0.7930	0.8293	0.6179	0.8212
		0.8020	0.8402	0.6657	0.8241
Ensemble Learning	BalancedBaggingClassifier	0.6386	-	-	-
		0.9757	-	-	-
		0.9740	-	-	-
		0.9743	-	-	-
	RUSBoostClassifier	0.6751	-	-	-
		0.8961	-	-	-
		0.9835	-	-	-
		0.8930	-	-	-
	EasyEnsembleClassifier	0.6176	-	-	-
		0.9684	-	-	-
		0.9474	-	-	-
		0.9679	-	-	-
Algorithmic	DEC (SVM)	0.9754	0.9754	-	-
		0.5000	0.5000	-	-
		0.9874	**0.9876**	-	-
		0	0	-	-
	One-class (SVM)	0.5119	0.5152	-	-
		0.6012	0.6629	-	-
		0.4376	0.6367	-	-
		0.5045	0.6338	-	-

Table 5: Over-sampling, combined, ensemble and algorithmic modification results for 2.5% Imbalance Ratio (total UPDRS score)

Balanced Accuracy / AUC / F1 / G-Mean in order

5 Conclusion

We have explored various available techniques for conventional approaches – under-sampling the majority and over-sampling the minority – as well as decision tree-based ensemble learning and algorithmic cost modifications. Our empirical results show that for this particular Parkinson's Disease Telemonitoring dataset, a combined resampling approach (over-sampling with SMOTE and cleaning with under-sampling) works well specifically with an RBF-kernel SVM classifier. In general, SMOTE may be good enough for adjusting the proportions; however, because we manipulated the target class such that the number of positive (minority) class is significantly less than that of the dominant, it was much more severe and may be more difficult to fix the problem. The evaluation of how good a classifier heavily depends on the evaluation metric, and it was sometimes hard to observe consistency and high numerical results simultaneously in some technique-classifier combinations. For this reason, some methods yielded results poorer than the baseline. As next steps to build upon this work, comparison of application of the same techniques on other datasets and other algorithmic modifications could be attempted.

References

[1] Rukshan Batuwita and Vasile Palade. "Class Imbalance Learning Methods for Support Vector Machines". In: vol. 83. June 2013, pp. 83–99. ISBN: 9781118074626. DOI: 10.1002/9781118646106.ch5.

[2] Kay Henning Brodersen et al. "The Balanced Accuracy and Its Posterior Distribution". In: *Proceedings of the 2010 20th International Conference on Pattern Recognition*. ICPR '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 3121–3124. ISBN: 978-0-7695-4109-9. DOI: 10.1109/ICPR.2010.764. URL: https://doi.org/10.1109/ICPR.2010.764.

[3] Shuzhan Fan. *Why is accuracy not the best measure for assessing classification models?* 2018. (Visited on 12/15/2019).

[4] Haibo He and Edwardo A. Garcia. "Learning from Imbalanced Data". In: *IEEE Trans. on Knowl. and Data Eng.* 21.9 (Sept. 2009), pp. 1263–1284. ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.239. URL: https://doi.org/10.1109/TKDE.2008.239.

[5] Chih-wei Hsu, Chih-chung Chang, and Chih-Jen Lin. "A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin". In: (Nov. 2003).

[6] Kerenaftali Klein, Stefanie Hennig, and Sanjoy Ketan Paul. "A Bayesian Modelling Approach with Balancing Informative Prior for Analysing Imbalanced Data". In: *PLOS ONE* 11.4 (Apr. 2016), pp. 1–12. DOI: 10.1371/journal.pone.0152700. URL: https://doi.org/10.1371/journal.pone.0152700.

[7] Adam Kowalczyk and Bhavani Raskutti. "One Class SVM for Yeast Regulation Prediction". In: *SIGKDD Explorations* 4 (Dec. 2002), pp. 99–100. DOI: 10.1145/772862.772878.

[8] Mostafizur Rahman and Darryl N. Davis. "Addressing the Class Imbalance Problem in Medical Datasets". In: *International Journal of Machine Learning and Computing* 3 (Apr. 2013), p. 224. DOI: 10.7763/IJMLC.2013.V3.307.

[9] Bhavani Raskutti and Adam Kowalczyk. "Extreme re-balancing for SVMs: A case study". In: *SIGKDD Explorations* 6 (June 2004), pp. 60–69. DOI: 10.1145/1007730.1007739.

[10] Saishruthi Swaminathan. *Logistic Regression — Detailed Overview*. 2018. (Visited on 12/15/2019).

[11] Athanasios Tsanas et al. "Accurate Telemonitoring of Parkinson's Disease Progression by Noninvasive Speech Tests". In: *IEEE transactions on bio-medical engineering* 57 (Nov. 2009), pp. 884–93. DOI: 10.1109/TBME.2009.2036000.

[12] *Why is accuracy not the best measure for assessing classification models?* 2017. (Visited on 12/15/2019).

Appendix A Contrived Class Proportions

Thresholded Attr.	Value	Majority		Minority	Minority/Majority (%)
Total_UPDRS_Score	8	5731	:	144	**2.5127**
	9	5713	:	163	2.8356
	10	5684	:	191	3.3603
	11	5656	:	219	3.8720
	12	5608	:	267	4.7611
	13	5569	:	306	5.4947
	14	5507	:	368	6.6824
	15	5368	:	507	9.4449
	16	5280	:	595	**11.2689**
	17	5077	:	798	15.7179
	18	4969	:	906	18.2330
	19	4855	:	1020	21.0093
	20	4599	:	127	27.7452
	21	4491	:	1384	**30.8172**
	22	4300	:	1575	36.6279
	23	4186	:	1689	**40.3488**
	24	4002	:	1873	46.8016
	25	3687	:	2188	**59.3436**
	26	3475	:	2400	69.0647
	27	3084	:	2791	90.4994
	28	3043	:	2832	**93.0661**
	29	3210	:	2665	**83.0218**
	30	3341	:	2534	**75.8456**
	31	3502	:	2373	67.7613
	32	3691	:	2184	59.1710
Motor_UPDRS_Score	6	5821	:	54	0.9277
	7	5707	:	168	2.9438
	8	5642	:	233	4.1297
	9	5500	:	375	6.8182
	10	5408	:	467	8.6354
	11	5275	:	600	11.3744
	12	5114	:	761	14.8807
	13	4789	:	1086	**22.6770**
	14	4603	:	1272	27.6342
	15	4411	:	1464	33.1898
	16	4115	:	1760	42.7704
	17	3885	:	1990	**51.2227**
	18	3583	:	2292	63.9687
	19	3292	:	2583	78.4629
	20	3115	:	2760	88.6035
	21	2951	:	2924	**99.0851**
	22	3130	:	2745	87.6997
	23	3271	:	2604	79.6087

Table 6: Imbalance Proportions by Thresholding