Boosting approaches with multi-label imbalanced data problem (Machine Learning 2024 Course)

Diana Koldasbayeva [1] **Aleksandr Kolomeitcev** [1] **Yekaterina Smolenkova** [1] **Lada Karimullina** [1]
Dmitriy Topchiy [1 2]

Abstract

In machine learning, handling imbalanced datasets poses a challenge for traditional algorithms designed for equal class distributions. Boosting algorithms offer a powerful solution by enhancing individual learners' performance through ensemble methods. This review focuses on boosting techniques for multi-class imbalanced datasets, evaluating CatBoost and LogitBoost as top performers. The study explores data and algorithm level interventions, assessing efficacy across various imbalanced datasets. It also proposes a novel criterion for metric comparison to identify optimal performance metrics.

Keywords: Boosting algorithms, Imbalanced data, Multi-class classifcation,ensemble learning

Github repo: Multi-class imbalanced data classifcation

1. Introduction

Learning on skewed datasets becomes very important since many real-world classification tasks are usually unbalanced. Multiclass unbalanced learning is even more challenging than binary scenarios and remains an open problem. Busting algorithms that improve model performance by combining underlying learners face difficulties in obtaining good predictions on large unbalanced multiclass datasets. The goal of this project is to conduct experiments aimed at improving the performance of Busting algorithms in multi-label classification tasks. The main contribution of this report is the experiments:

- Base-Line Implementation
- Studying the influence of base learners within the bousting procedure
- Studying the effect of preprocessing operators on the performance of bousting algorithms
- Implementation of combined ensemble methods using data-level sampling techniques

Our hypotheses:

- There is a significant difference in performance between different boosting algorithms when using different estimators.
- Algorithms that are more robust to imbalanced data, like RusBoost and SmoteBoost, should yield better results

2. Literature review

2.1. Imbalanced Multilabel data

"Multi-Label Classification (MLC)" is an extension of the standard single-label classification where each data instance is associated with several labels simultaneously (Adane Nega Tarekegn, 2021). MLC has gained much importance in recent years due to its wide range of application domains. However, the class imbalance problem has become an inherent characteristic of many multi-label datasets, where the samples and their corresponding labels are non-uniformly distributed over the data space. There are two well-known approaches for solving the MLC task: **problem transformation** and **algorithm adaptation**.

The **former transforms** the MLC task into one or more single-label classification (G. Tsoumakas, 2007), or label ranking (LR) tasks (J. Fürnkranz, 2008), while the latter aims to adapt the traditional machine learning algorithms to handle multi-label dataset (MLD) directly. The imbalance problem in an MLD can be viewed from three perspectives: **imbalance within labels**, **imbalance between labels**, and **imbalance among the label-sets**.

In the case of **imbalance within labels**, each label usually contains an extremely high number of negative samples and a very small number of positive samples (Y. Sun, 2009; Z. Sun, 2015; W.W.Y. Ng, 2016).

In the **imbalance between labels**, the frequency of individual labels in the MLD is considered where the number of 1's (positive class) in one label may be higher than the number of 1's in the other label (M. Fang, 2014; F. Charte, 2019). Since every instance of an MLD is associated with several outputs or labels, it is common that some of them are majority ones while others are minority labels, i.e., some labels have many more positive samples than others.

Imbalance among the label-sets occurs in MLD is the sparse frequency of label-sets (F. Charte, 2015). If a full label-set is taken into account, the proportion of positive to negative samples for each class may be associated with the most common label-sets. In MLDs, due to the label sparseness, there are usually more frequent label-sets and unique label-sets. This also implies that some of the label-sets may be considered majority and the remaining label-sets may be considered minority cases at the same time.

2.2. Approaches to deal with imbalanced data problem

Imbalanced data presents a challenge for machine learning, as it can lead to biased predictions favoring more common classes. Solutions fall into three categories: data-level, algorithm-level, and hybrids. Data-level strategies balance class distributions through techniques like under-sampling or over-sampling.[(A.J. Ferreira, 2012)] For multi-class scenarios, methods like one-versus-all or one-versus-one simplify classification.[(F. Charte, 2019)] Algorithm-level approaches adjust learning algorithms to reduce bias, often using cost-sensitive learning, where higher penalties are assigned to misclassifications of minority classes to promote accurate classification.

In this research, various boosting algorithms are scrutinized for their efficacy in handling multi-class imbalanced data. CatBoost, SMOTEBoost and LogitBoost emerge as prominent techniques amongst others due to their tailored approaches for imbalanced datasets. CatBoost, in particular, statistically outperforms AdaBoost, SAMME, MEBoost, and RUSBoost on 15 conventional datasets, maintaining superiority in terms of the G-mean, MAUC and MMCC criterions.[(J. Fürnkranz, 2008)]

CatBoost stands out for efficiently handling categorical features and robust performance, even on large datasets. Gradient-based boosting algorithms like CatBoost, GradientBoost, and XGBoost, along with SMOTEBoost's sampling technique, show superior performance over other methods, suggesting their integration could enhance performance on imbalanced datasets.

2.3. Validation about metrics (compare scores roc auc)

As it is known the area under the curve (AUC) of the receiver operating characteristic (ROC) is extremely used as an overall evaluation technique. The ROC curve shows all possible conflicts between true-positive rate (TPR) and false-positive rate (FPR) across various decision thresholds and AUC evaluation metric converts this curve to a value in the range of [0.5, 1], where the value 1 shows a perfect classifier and the lower value 0.5 means the classifier does not work better than random guess (T. Saito, 2015). Although extending the AUC to multi-class problems is an open research topic, the MAUC metric (L. Zhen, 2012) which averages the AUC value of all pairs of classes, are mostly used in the researches multi-class imbalanced data learning.

Algorithm 1 AdaBoost

```
Input: dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$,
Number of learning iterations $M$
Initialize equal weight to all training samples: $w_i = \frac{1}{N}$, $i = 1, 2, ..., N$
for $m = 1$ to $M$ do
    Train a base learner $G_m$ from $D$ using weights $w_i$
    Compute error of $G_m$ as $err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$
    Compute the weight of $G_m$ as $\alpha_m = \log\left(\frac{1-err_m}{err_m}\right)$
    Update weights: $w_i = w_i \cdot \exp\left(\alpha_m I(y_i \neq G_m(x_i))\right)$ for all $i$
end for
Output: The final hypothesis $G(x) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$
```

3. Algorithms and Models

3.1. AdaBoost

The main idea of AdaBoost is bolding the effect of misclassified samples by increasing their weights in the iterations of boosting. Thus, AdaBoost is well-suited for multilabel imbalanced data because it focuses on hard-to-classify examples by increasing their weights, ensuring underrepresented classes gain more attention in each iteration. This adaptive emphasis helps correct classification errors progressively, particularly benefiting minority classes. Additionally, AdaBoost's flexibility in combining multiple weak learners into a strong classifier helps improve generalization across all labels, making it effective against class imbalance. However, it is sensitive to noise and outliers. So necessitating careful preprocessing for optimal performance in imbalanced settings.

3.2. LogitBoost

LogitBoost is one of the modifications of the Adaboost algorithm, whose main difference is the use of logistic loss function instead of exponential loss function. Due to this, LogitBoost is less sensitive to noise and outliers.

This algorithm can be used for multi-class classification directly: taking into account the multi-class logistic loss. However, in our study, to follow the general pipeline, it was implemented as a binary classifier (see here). And it was used in multiclass classification through the One-to-Rest approach.

However, LogitBoost is poorly suited for data classification because it is based on conditional Bernoulli likelihood without prior probability in consideration, which leads to a high minority class prediction error.

3.3. MEBoost

MEBoost is specially designed to give the best performance with an unbalanced dataset. During each iteration, one of the two weak estimators is selected (in the original paper (Farshid Rayhan & Rahman, 2017) they are C4.5 and Extra Classifier). And it is added to the ensemble model only if, as a result, the score according to auROC on the test sample has become better. If no base estimator is added within a certain number of iterations, the stopping criterion is triggered. This implementation does not allow it to be directly implemented for multi classification, therefore, as in the case of LogitBoost, it is used as a binary classifier in the One-to-Rest technique.

The MEBoost algorithm is different from all the other boosting methods because it uses C4.5 and Extra tree classifier alternately instead of using only one of them. This allows to take advantage of both learner's characteristic and discard their individual weaknesses.

However, for the algorithm to work correctly, it is necessary to select parameters for each estimator separately. Otherwise, most often only one algorithm will appear in the resulting ensemble, or it is necessary to implement an additional rule that will equalize the number of base estimators.

3.4. RUSBoost

RUSBoost is an algorithm for solving class imbalance problems in data with discrete class labels. It uses a combination of RUS (random under-sampling) and the standard AdaBoost boosting procedure to better model the minority class by removing class samples.

RUSBoost is computationally cheaper than SMOTEBoost and has a much shorter model training time. This combination of simplicity, speed, and performance makes RUSBoost an excellent method for training on unbalanced data.

In our experiments, we used an algorithm implemented by our team, which can be found here.

3.5. GradientBoost

GradientBoost is particularly effective for imbalanced datasets due to its focus on misclassification. By iteratively adjusting for errors in predictions, especially favoring minority class predictions, it directly addresses the challenges of imbalanced data. Its flexibility with various base learners and the capability to capture non-linear relationships between features and labels enhance its performance on complex multi-label classification tasks. However, its effectiveness may be dampened by the risks of overfitting, especially in datasets with noise and outliers, and the computational intensity required for its sequential boosting process. This necessitates careful tuning and management to optimize its application in imbalanced datasets.

3.6. XGBoost

Extreme Gradient Boosting (XGBoost) is an ensemble learning technique that employs the gradient boosting technique over a set of decision trees. The XGBoost classifier presents strong advantages, including the ability to effectively classify instances by learning from previous mistakes, employ fine-tuning hyperparameters, adopt regularizing techniques, and handle imbalanced data

XGBoost has some amazing properties that make it extremely suitable for our data: It does a great job at handling severe imbalance of classes It can select the most important features needed during training and uses those for its tree building

In our experiments, we used ready code from the dmlc XGBoost.

3.7. CatBoost

CatBoost is a novel algorithm that is developed by Yandex researchers for gradient boosting on decision trees, which has the ability to handle the categorical features in the training phase. One of the important improvements of the CatBoost is doing unbiased gradient estimation in order to control the overfit. To this aim, in each iteration of the boosting, for truly estimating the gradient of each sample, it excludes that sample from the training set of the current ensemble model. The other improvement is the automatic transformation of categorical features to numerical features without any preprocessing phase. Also CatBoost is applicable for both binary and multi-class. CatBoost often requires careful tuning of its parameters. While CatBoost is designed to control overfitting, its performance might still be affected by noisy data or outliers. In cases where the dataset contains a lot of noise, additional preprocessing steps might be necessary to clean the data before training the model.

3.8. SMOTEBoost

SMOTEBoost directly addresses imbalances in datasets by generating synthetic examples of the minority class through the Synthetic Minority Over-sampling Technique (SMOTE). This strategy enhances minority class representation, improving model generalization and accuracy for these critical but underrepresented classes. Its versatility across various types of imbalances makes it a robust tool for multi-label classification challenges. However, the approach comes with challenges, including the potential for over-generalization from synthetic data and increased computational costs due to the additional steps in training, which may be significant for large datasets.

4. Experiments

4.1. Baseline

As a baseline all considered boosting algorithms were used with default hypo-parameters. To estimate the performance of each algorithm such metrics were used as: Precision, Recall, F1, G-mean (Geometric Mean of Sensitivity and Specificity), MMCC (Multi- class Matthews Correlation Coefficient), Kappa, Weighted Accuracy, PR Score, Balanced Accuracy. Precision and Recall measure the proportion of true positive predictions in the positive class predictions and the proportion of actual positives that are correctly predicted, respectively It's useful for understanding how many of the predicted positive instances are actually positive, which is especially important in imbalanced datasets where false positives for minority classes can be costly. F1 Score balances the trade-off between precision and recall, making it a valuable metric when it is necessary to consider both false positives and false negatives, which is often the case in imbalanced datasets. G-mean is useful in imbalanced datasets to ensure that the model does not favor the majority class. MMCC takes into account true and false positives and negatives, making it a robust metric for evaluating performance across all classes. Kappa is useful for multi-class problems, including imbalanced datasets, to assess the model's predictive accuracy beyond what is expected by chance. Weighted Accuracy adjusts traditional accuracy to account for the imbalance in class distribution by weighting the accuracy scores of each class by its prevalence. Balanced Accuracy ensures that each class contributes equally to the overall metric, making it especially suitable for evaluating performance on imbalanced datasets.

4.2. Searching base-estimators

In this experiment we used base estimators such as : Desicion Tree Regressor, Desicion Tree Classifier, Extra Tree Classifier, Support Vector Classifier and Logistic Regression. Decision trees are explainable and visualizable to business on the model's outcome. There is a high chance of overfitting. SVM works relatively well when there is a clear margin of separation between classes and more effective in high dimensional spaces. Logistic regression is easier to implement, interpret, and very efficient to train. It is less inclined to over-fitting but it can overfit in high dimensional datasets. We use Regularization (L1 and L2) techniques to avoid over-fitting.

For each algorithm, the base estimator with the best hyper parameters was selected using ParameterGrid.

4.3. Preprocessing

Preprocessing employs several boosting models, including 'LogitBoost', 'MEBoost', 'AdaBoost', 'RUSBoost', and 'GradientBoostingClassifier', to iteratively improve predictions by focusing on previous errors.

To complement these models, a range of preprocessing methods were utilized:

'StandardScaler', which standardizes features by removing the mean and scaling to unit variance. This is useful in algorithms that assume data to be normally distributed;
'MinMaxScaler',which transforms features by scaling each feature to a given range, often between zero and one. This scaling compresses all features to a specific range and is beneficial for models sensitive to variance in data;
'RobustScaler', which scales features using statistics that are robust to outliers. This scaler removes the median and scales the data according to the quantile range. It's ideal for data with outliers; 'Normalize', which normalizes individual samples to have unit norm. This process is useful when the magnitude of data vectors is significant;
'PowerTransformer', which applies a power transformation to each feature to make the data more Gaussian-like, which is useful for modeling issues related to heteroscedasticity or non-normal distributions;
'QuantileTransformer', which transforms the features to follow a uniform or a normal distribution. Therefore, it smoothens out unusual distributions and is less influenced by outliers than scaling methods. It spreads out the most frequent values and reduces the impact of (marginal) outliers;
'PolynomialFeatures', which generates polynomial and interaction features. It creates new features by raising existing features to an exponent or by multiplying features together. This can uncover relationships between features that are not observable in the original space;
'KBinsDiscretizer', Bins continuous data into intervals. This transformer discretizes features into bins, converting continuous features into discrete or categorical features, which can be useful for certain types of models that work better with categorical data.

4.4. Resampling

In the third experiment (link) we tried to improve the performance of boosting algorithms by balancing classes using various resampling techniques. Among the **oversampling methods** used in the experiment were the following:

- **Random oversampling.** Randomly repeats available samples from the minor class with replacements. SMOTE. Samples are randomly generated between randomly selected nearest neighbors of minor class samples belonging to the same class.
- **Borderline SMOTE.** The algorithm oversamples the samples which will be close to the decision boundary.

The following algorithms were used for **Undersampling**:

- **Random undersampling** involves randomly selecting a sample from the majority class, with or without replacement.
- **NearMiss-1.** The algorithm removes the majority-class sample with the smallest mean distance from the minority class to the nearest N sample.
- **CondensedNearestNeighbour.** The algorithm iteratively determines which major class samples should be removed using the 1-nearest neighbor rule. The result should be a subset representing the major class.
- **EditedNearestNeighbours.** Modification of CNN, 3-nearest neighbors instead of one is used with the same idea: remove samples that do not match their neighbors.
- **RepeatedEditedNearestNeighbours.** It extends the ENN by repeating the algorithm multiple times.
- **ALLKNN** extends RENN by increasing the number of nearest neighbors at each iteration.
- **OneSidedSelection** extends ENN by applying TLs and the 1-nearest neighbor rule to remove noisy samples.
- **NeighborhoodCleaningRule.** If a value is classified as a majority class using 3-nearest neighbors, it is deleted. However, if the value is classified as a minority class, the majority-class samples in the 3-nearest neighbors are deleted.

The experiment pipeline is structured as follows: for each dataset, algorithms are iterated in turn, resamplers are iterated for each algorithm, and metrics are collected through M-Fold training and testing.

Table 1. Baseline Top-Performing algorithms by F1 score.

Dataset	Top Algorithm	F1
Pen-Based	XGBoost	0.989 ± 0.001
Hayes	LogitBoost	0.891 ± 0.024
Wine	AdaBoost	0.983 ± 0.024
Contraceptive	RUSBoost	0.461 ± 0.061
Balance	AdaBoost	0.857 ± 0.025
Differentiated	LogitBoost	0.979 ± 0.015
Vertebral	AdaBoost	0.862 ± 0.016
Dermatology	AdaBoost	0.98 ± 0.017
Thyroid	CatBoost	0.962 ± 0.014
Glass	LogitBoost	0.744 ± 0.104
Heart	LogitBoost	0.561 ± 0.11
Car	XGBoost	0.974 ± 0.008
Yeast	AdaBoost	0.582 ± 0.039
Covertype	XGBoost	0.87 ± 0.0
Statlog	XGBoost	0.998 ± 0.0

5. Results

5.1. Baseline

For calculating metrics M folding was used. All considering boosting algorithms were implemented without tuning hypo-parameters. Obtained results of each algorithms for each metrics in all datasets it is possible to observe here.In general, based on the highest F1 score of algorithms in each datasets AdaBoost performs better than others more frequently. Also mean imbalanced ratio was calculated for each dataset and based on the highest F1 score top performing algorithms were selected in tested datasets. Further, results of F1 score that was received in baseline will be used for comparison with experiments.

5.2. Searching base-estimators

Based on the highest F1 score of the algorithms in each dataset, the Decision tree classifier performs the best more often than others, especially in combination with AdaBoost classifier. Also, according to the results of all experiments, the best results were obtained by selection the best estimators, including the searching of hyperparameters for the estimators.

5.3. Preprocessing

During the experiment the best combination of boosting model and a preprocessor was found for every dataset based on the highest F1 score. Obtained results that compare evry model to preprocessor for every dataset can be accessed here. And the best results for the best preprocessor for the best performing model for every datset can be accessed here

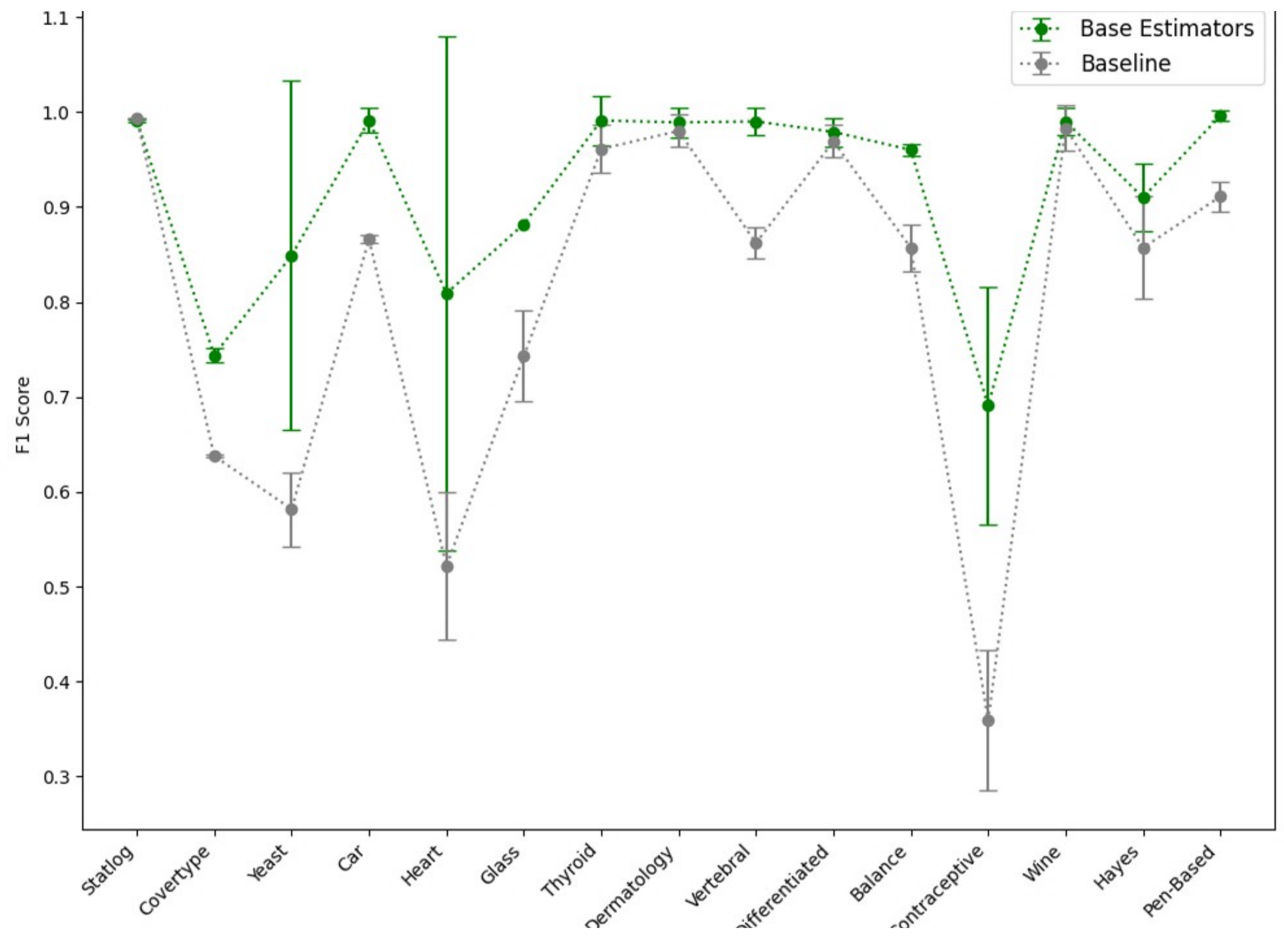

Figure 1. F1 Scores by experiment 1 for Datasets with Standard Deviation for ADABoost.

Table 2. Results comparison experiment 1 with baseline for the Vertebral dataset

TOP ALGORITHM	TOP ESTIMATOR	F1	F1_BASE
MEBOOST	DTREECL	0.847	0.753
ADABOOST	DTREECL	0.904	0.862
LOGITBOOST	DTREEREG	0.848	0.816
GRADIENTBOOST	DTREEREG	0.875	0.857
SMOTEBOOST	ETREECL	0.841	0.546
RUSBOOST	DTREEREG	0.844	0.739

Table 3. Results comparison experiment 2 with baseline for the Vertebral dataset

TOP ALGORITHM	TOP PREPROCESSOR	F1	F1_BASE
MEBOOST	NORMALIZER	0.784	0.753
ADABOOST	POLYNOMIAL	0.913	0.862
LOGITBOOST	POLYNOMIAL	0.927	0.816
GRADIENTBOOST	POLYNOMIAL	0.905	0.857
XGBOOST	NORMALIZER	0.924	0.906
CATBOOST	POLYNOMIAL	0.912	0.877
SMOTEBOOST	NORMALIZER	0.841	0.546
RUSBOOST	NORMALIZER	0.844	0.739

5.4. Resampling

During the experiment it was revealed that the correct selection of the algorithm depends more on the datasets themselves than on the boosting algorithms used or the base estimators used in them. It was also revealed that, in general, for the same dataset, the same resampling algorithm can be used for different boosting algorithms. For example, for the Vertebral dataset, most often the BorderlineSMOTE resampler showed the best performance (the results of the experiments can be seen in the tables on the git (link) for each experiment).

But on the other hand, for an optimally selected base esti-

Table 4. Results comparison experiment 3 with baseline for the Vertebral dataset

Top Algorithm	Top resampler	F1	F1_base
MEBoost	SMOTE	0.858	0.753
AdaBoost	BorderSMOTE	0.856	0.862
LogitBoost	BorderSMOTE	0.856	0.816
GradientBoost	OneSidedSelect	0.86	0.857
XGBoost	BorderSMOTE	0.912	0.906
CatBoost	BorderSMOTE	0.892	0.877

mator and preprocessing, the use of a resampler (the best without them) as a rule only worsens the performance.

6. Conclution

The best results were obtained by selection the best estimators, including the searching of hyperparameters for the estimators hence Hypothesis 1 is confirmed.

The conducted experiments have shown that the hypothesis about algorithms that are robust to imbalanced data should yield better results is inaccurate.

References

Adane Nega Tarekegn, Mario Giacobini, K. M. A review of methods for imbalanced multi-label classification. *Pattern Recognition*, 118(107965), 2021.

A.J. Ferreira, M. F. Boosting algorithms: a review of methods, theory, and applications. In *Ensemble Machine Learning*, pp. 35–85. Springer, Boston, 2012.

F. Charte, A.J. Rivera, M. d. J. F. H. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163:3–16, 2015.

F. Charte, A.J. Rivera, M. d. J. F. H. Dealing with difficult minority labels in imbalanced mutilabel data sets. *Neurocomputing*, 326-327:39–53, 2019.

Farshid Rayhan, Sajid Ahmed, A. M. M. R. J. S. S. D. M. F. and Rahman, C. M. Meboost: Mixing estimators with boosting for imbalanced data classification. *IEEE*, 2017.

G. Tsoumakas, I. V. Random k-labelsets: an ensemble method for multilabel classification. In *Machine Learning: ECML 2007, Springer Berlin Heidelberg*, pp. 406–417, Berlin, Heidelberg, 2007.

J. Fürnkranz, E. Hüllermeier, E. L. M. K. B. Multilabel classification via calibrated label ranking. *Machine Learning*, 73:133–153, 2008.

L. Zhen, L. Q. A new feature selection method for internet traffic classification using ml. *Phys Procedia*, 1(33):455–1338, 2012.

M. Fang, Y. Xiao, C. W. J. X. Multi-label classification: dealing with imbalance by combining labels. In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, Limassol, Cyprus, 2014. IEEE.

T. Saito, M. R. Mhe precision-recall plot is more informative than the roc plot when evaluating binary clas- sifiers on imbalanced datasets. *PLoS ONE*, 10(3), 2015.

W.W.Y. Ng, G. Zeng, J. Z. D. Y. W. P. Dual autoencoders features for imbalance classification problem. *Pattern Recognition*, 60:875–889, 2016.

Y. Sun, A.K.C. Wong, M. K. Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(4):687–719, 2009.

Z. Sun, Q. Song, X. Z. H. S. B. X. Y. Z. A novel ensemble method for classifying imbalanced data. *Pattern Recognition*, 48(5):1623–1637, 2015.

Table 5. Results comparison best experiments with baseline for all datasets

DATASET	MEANIR	TOP ALGORITHM	F1 BEST PERFORMANCE	F1 BASELINE
PEN-BASED	0.961	ADABOOST	0.995 ± 0.007	0.989 ± 0.001
HAYES	0.856	ADABOOST	0.908 ± 0.037	0.891 ± 0.024
WINE	0.836	ADABOOST	0.989 ± 0.015	0.983 ± 0.024
CONTRACEPTIVE	0.781	GRADIENTBOOST	0.681 ± 0.134	0.461 ± 0.061
BALANCE_SCALE	0.723	ADABOOST	0.956 ± 0.007	0.857 ± 0.025
DIFFERENTIATED	0.696	GRADIENTBOOST	0.987 ± 0.009	0.979 ± 0.015
VERTEBRAL	0.689	ADABOOST	0.904 ± 0.055	0.862 ± 0.016
DERMATOLOGY	0.538	ADABOOST	0.987 ± 0.016	0.98 ± 0.017
THYROID	0.478	ADABOOST	0.981 ± 0.018	0.962 ± 0.014
GLASS	0.469	ADABOOST	0.841 ± 0.079	0.744 ± 0.104
HEART	0.371	ADABOOST	0.824 ± 0.248	0.561 ± 0.11
CAR	0.357	ADABOOST	0.974 ± 0.008	0.974 ± 0.008
YEAST	0.321	ADABOOST	0.836 ± 0.202	0.582 ± 0.039
STATLOG	0.182	GRADIENTBOOST	0.998 ± 0.0	0.998 ± 0.0
COVERTYPE	0.293	GRADIENTBOOST	0.805 ± 0.005	0.87 ± 0.0

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Yekaterina Smolenkova (25% of work)

- Reviewing literate on the topic (2 papers)
- Coding RUSBoost algorithm
- Experimenting with base estimators for boosting algorithms
- Preparing the GitHub Repo
- Preparing the Sections 1-6 of this report
- Pre-preprocessing of all datasets

Aleksandr Kolomeitcev (25% of work)

- Reviewing literate on the topic (3 papers)
- Coding MEBoost, LogitBoost algorithm and MulticlassClassificationOvR
- Experimenting with resampling for boosting algorithms
- Preparing the Sections 1-6 of this report

Lada Karimullina (25% of work)

- Reviewing literate on the topic (2 papers)
- Coding ADABoost algorithm and def for plotting results
- Baseline performance
- Preparing the Sections 1-6 of this report
- Metrics exploring

Dmitriy Topchiy (25% of work)

- Reviewing literate on the topic (2 papers)
- Coding GradientBoost algorithm
- Experimenting with preprocessing for boosting algorithms
- Preparing the Sections 1-6 of this report
- Metrics exploring

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   General comment: We used SMOTEBoost, XGBoost and CatBoost from open source

   Students' comment: None

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   ☐ Yes.
   ☐ No.
   ☑ Not applicable.

   Students' comment: MFold was used

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: MFold was used

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

9. The exact number of evaluation runs is included.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.

    ☑ Yes.

- ☐ No.
- ☐ Not applicable.

Students' comment: None