Literature Review for Imbalanced Data

Brad Burkman

7 March 2022

Contents

Index **3**

1 To Do **4**
1.1 Format . . . 4
1.2 Topics Remaining . . . 4

2 Methods **5**
2.1 Algorithm Level Approaches . . . 5
2.1.1 Some Papers . . . 5
2.1.2 Genetic Algorithms . . . 5
2.1.3 Subspace Model . . . 5
2.2 Loss Functions . . . 6
2.2.1 Class Weights and α-weighted Loss . . . 6
2.2.2 Oversampling . . . 7
2.2.3 Naive Oversampling . . . 7
2.2.4 Class Weights v/s Naive Oversampling: They're the Same . . . 7
2.2.5 Focal Loss . . . 7
2.2.6 Optimizing F_β . . . 8
2.2.7 Tree-Based Methods . . . 8
2.3 Data Level Methods . . . 8
2.3.1 Imbalanced Cleaning: Tomek and Condensed Nearest Neighbor . . . 8
2.3.2 Tomek's Links . . . 9
2.3.3 Cleaning Multiclass Data . . . 10
2.3.4 Oversampling . . . 10
2.3.5 Underssampling . . . 10
2.3.6 SMOTE and its Flavors . . . 11
2.3.7 Train/Test Split . . . 12
2.3.8 Feature Selection . . . 12
2.4 Bagging and Boosting . . . 12

3 Datasets **13**
3.1 Repositories 13
3.2 Useful Datasets 13
3.3 Articles using Multiple Datasets 13

4 Interesting Papers **14**
4.1 Review Papers 14
4.1.1 Chawla 14
4.1.2 Chabbouh 2019 15
4.1.3 Mahmudah 2021 15

Bibliography **16**

Index

F_β, 8
α-Weighted Loss, 6

Class Weights, 6
Condensed Nearest Neighbor, 8

Focal Loss, 7

Imbalanced-Learn, 8

SMOTE, 11

Tomek Links, 8

Chapter 1

To Do

1.1 Format

- Introduction
  - Problem
  - Rationale
- Lit Review
  - For each topic relevant to the problem, summarize the topic so people who haven't read the paper can follow.
  - For non-so-relevant topics, just give a brief review.
- Dataset
  - Overview
  - Properties
- Research Plan
  - What I've done
  - Plan

1.2 Topics Remaining

- Worked through Brad's Reports through 2/14/22. Need to do 2/21.
- Find where it mentions multiple Tomek runs.
- Work through this tutorial, which will explain Condensed Nearest Neighbor. https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/
- Figure out what Rough Sets Theory and Fuzzy Sets are.
- Learn to use ROC curves.
- Review different types of models and how to implement them in Keras.

Chapter 2

Methods

2.1 Algorithm Level Approaches

2.1.1 Some Papers

- Recognition-based: Learning from one class rather than discrimination-based, doing unsupervised learning on the minority class. [4]
- Fuzzy rule-based classification systems (what is this?) [2] [6] [9] [18] [20]
- In decision trees, using evolutionary/genetic methods instead of greedy search [2] [16]
- Clustering and Subspace Modeling [5]

2.1.2 Genetic Algorithms

In this short 2000 paper, Weiss [16] used a genetic algorithm to predict rare events. Borrowing from simulated annealing, they varied the relative importance of precision and recall at each step of the genetic algorithm.

2.1.3 Subspace Model

Chen 2011 [5]

This was fascinating and entirely different from anything I've seen.

1. Separate the training data Tr into negative (majority) and positive (minority) classes TrN and TrP.
2. Let K be the ratio of negative to positive samples, in my case about 100, so that if you divide the majority class TrN into K groups, each will have about the same number of samples as the minority class.
3. Use K-means clustering to separate the negative (majority) class TrN into K groups; each of the groups is a cluster of the negative (majority) class.
4. For each of the K groups TrN_i:

- Combine the negative elements of the group with the entire positive (minority) class TrP to form a balanced subspace.
- Train the model for the subspace

5. Recombine the K subspace models with a model trained on the entire data set to build an integrated model.

2.2 Loss Functions

Let's say we have an imbalanced data set with 100 negative samples for each positive sample.

For binary classification, the first three (class weights, weighted loss function, and naive oversampling) are effectively the same in the training phase. The cross-entropy loss function,

$$loss = \sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

for binary classification is

$$loss = \sum_{y_i=1} \log(p_i) + \sum_{y_i=0} \log(1 - p_i)$$

which is the sum of the logs of the errors in predictions for the negative class plus the sum of the logs of the errors in predictions for the positive class.

2.2.1 Class Weights and α-weighted Loss

If the classes are imbalanced, like there are 100 times as many samples with $y = 0$ as samples with $y = 1$, then the loss function is mostly summing how bad the predicting probability is for the majority class and largely ignoring the minority class. Both the class weights parameter and a weighted loss function fix this by multiplying one or the other by some compensating factor.

$$loss = 100 \times \sum_{y_i=1} \log(p_i) + \sum_{y_i=0} \log(1 - p_i)$$

This multiple gives the two classes equal weight in the loss.

In the α-weighted cross entropy,

$$loss = \sum_{i=1}^{n} \alpha y_i \log(p_i) + (1 - \alpha)(1 - y_i) \log(1 - p_i)$$

let $\alpha = \frac{100}{100+1}$ and you'll get the same thing, within a positive constant multiple.

$$loss = \sum_{i=1}^{n} \frac{100}{101} y_i \log(p_i) + \frac{1}{101}(1 - y_i) \log(1 - p_i)$$

$$loss = \frac{1}{101}\sum_{i=1}^{n} 100y_i \log(p_i) + (1-y_i)\log(1-p_i)$$

The only difference I can ascertain between the class weights parameter and a weighted loss function is that the class weights aren't used with the validation set.

2.2.2 Oversampling

2.2.3 Naive Oversampling

Naive oversampling would be to create 99 copies of each of the positive samples, so that the two sets are balanced. That would have exactly the same effect on the loss function, because there would now be 100 times as many samples with $y_i = 1$.

2.2.4 Class Weights v/s Naive Oversampling: They're the Same

I had an insight on why these things are the same. Let's say you have an imbalanced data set, with 100 times as many negative samples as positive samples.

In Naive Oversampling, you make 100 copies of each of the positive samples and run regular cross-entropy loss.

In weighted Class Entropy, you multiply the positive-class losses by 100.

$$loss = 100 \times \sum_{y_i=1} \log(p_i) + \sum_{y_i=0} \log(1-p_i)$$

These two approaches different in execution but the same in result because, as I often remind my students, multiplying something by 100 is the same as adding it to itself 100 times.

2.2.5 Focal Loss

In the focal loss function,

$$loss = \sum_{i=1}^{n} \alpha(1-p_i)^{\gamma_1} y_i \log(p_i) + (1-\alpha)p_i^{\gamma_2}(1-y_i)\log(1-p_i)$$
$$loss = \sum_{y_i=1} \alpha(1-p_i)^{\gamma_1} \log(p_i) + \sum_{y_i=0} (1-\alpha)p_i^{\gamma_2} \log(1-p_i)$$

if $\gamma_1 = \gamma_2 = 0$, then it's the same as the α-weighted loss function.

In the original focal loss paper by Lin [8], γ_1 and γ_2 are the same.

For samples with $y_i = 1$, the minority class, here are values of $(1-p_i)^{\gamma_1}\log(p_i)$ for different values of p_i and different values of γ_1. I got the range of values of $\gamma_1 \in \{0, 0.5, 1, 2, 5\}$ from Lin's 2018 paper that proposed focal loss.

$(1-p_i)^{\gamma_1}\log(p_i)$		γ_1 0	0.5	1	2	5
	0.1	-3.32	-3.15	-2.99	-2.69	-1.96
	0.3	-1.74	-1.45	-1.22	-0.85	-0.29
p_i	0.5	-1	-0.71	-0.5	-0.25	-0.03
	0.7	-0.51	-0.28	-0.15	-0.05	0
	0.9	-0.15	-0.05	-0.02	0	0

If $\gamma_1 > 0$, then for samples in the positive class, the loss is negligible for good predictions (p_i close to 1), so it focuses the loss on poor predictions.

Yu applied focal loss in the crash literature.[17]

2.2.6 Optimizing F_β

Loss functions for gradient-based learning need to be differentiable (?), and the F_β score is not differentiable, so this 2021 article by Lee [7] proposes a differentiable surrogate loss function that optimizes the F_β score.

With imbalanced data, using a loss function that optimized F_β instead of accuracy would let you balance precision and recall, fixing one aspect of the imbalance problem.

$$F_\beta = \frac{(1+\beta^2)\cdot \text{Precision}\cdot \text{Recall}}{(\beta^2\cdot \text{Precision}) + \text{Recall}} = \frac{1}{\frac{\lambda_\beta}{\text{Recall}} + \frac{1-\lambda_\beta}{\text{Precision}}}, \qquad \lambda_\beta = \frac{\beta^2}{1+\beta^2}$$

The article takes a deep dive into loss functions. I should master it.

2.2.7 Tree-Based Methods

Pendault [11] has a 2000 article on insurance risk modeling that incorporates "a domain-specific optimization criterion... to identify suitable splits during tree building." It assigns different weights to *claim* and *nonclaim* records. Because that strategy helps but does not entirely solve the imbalanced data problem, they also have a split criterion that prevents splits of really small branches, "splinter groups," that are unlikely to contain any elements of the minority class because the minority class is so sparse.

2.3 Data Level Methods

2.3.1 Imbalanced Cleaning: Tomek and Condensed Nearest Neighbor

Batista [1] uses two imbalanced cleaning method called *Tomek links* and *Condensed Nearest Neighbor*. If examples from the majority and minority class are close to each other, it deletes the majority samples. One could think of it as targeted undersampling of the majority set.

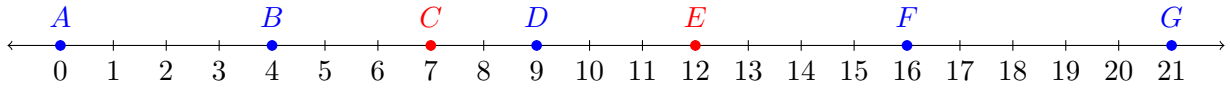
Imbalanced-Learn, an add-on to Scikit-Learn, has these algorithms read to use. Tomek and Wilson's papers introducing these algorithms are from the 1970's.

2.3.2 Tomek's Links

This method undersamples the majority-class samples, eliminating ones that are too close to minority-class samples, presuming them to be noise, and helping clarify clusters of minority samples.

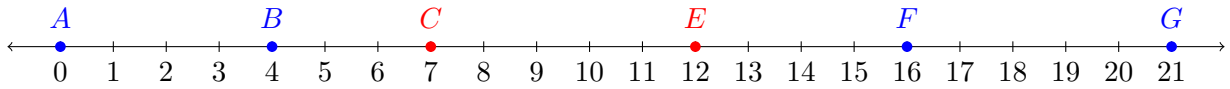
A pair of samples are a *Tomek link* if one is majority and one minority, and they are each other's nearest neighbors. To use Tomek's inks as an undersampling strategy for imbalanced data, delete the positive sample in each Tomek's link. Other cleaning strategies (for balanced sets) would eliminate both the positive and negative.

It is possible to iterate Tomek's several times. Here's an example of how it works in one round and in a second round. The blue samples are from the majority set and the red are from the minority. Assume that these seven points are a small part of a large dataset, but these are the only points in this region.

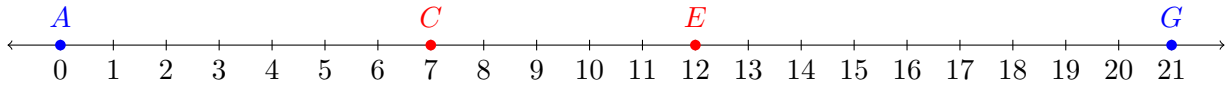


In the original dataset, C and D are each other's nearest neighbors, C from minority and D from majority, so they are a Tomek link. On the other hand, D is the nearest neighbor to E, but E is not D's nearest neighbor, so they are not a Tomek link.

Eliminate sample D.



Now the pairs (B, C) and (E, F) are Tomek links, so if we ran Tomek undersampling a second time, we would remove samples B and F.



Now C and E are each other's nearest neighbors and of the same (minority) class, so this part of the dataset would not change under another run of Tomek.

The idea of Tomek assumes that the minority samples should cluster, and any majority samples in or near those clusters must be noise, so we can eliminate them. We now have a clear cluster of two minority samples with no close majority samples.

I saw multiple runs of Tomek mentioned [somewhere] in my reading, so I tried it on the crash data, running it up to five times, and saw that it converged, with fewer positive samples eliminated in each round. I had conjectured that a negative sample in a Tomek link in a later round must have been a negative sample in a Tomek link in an earlier round, digging itself out of a field of positive-class dust, but I suspected that there might be (perhaps unusual) cases where one minority-class sample (C in the example above) created a Tomek link, and eliminating the majority-class sample in that link (D above) allowed a Tomek link for a different minority-class sample (E above). I then played with it until I found a counterexample to my conjecture, so the conjecture, that a

minority-class sample in a Tomek link in a later round of Tomek undersampling must have been in a Tomek link in every previous round of the Tomek undersampling, is false.

If the conjecture had been true, then we could greatly speed up subsequent rounds of Tomek undersampling by only considering the minority samples in Tomek links in the previous round. That would not be thorough, but this approach would.

Algorithm for Repeated Application of Tomek's Links

For the first round of Tomek undersampling, one has to consider each element of the minority class. In the Tomek's links, call the minority-class elements $\{A_1, A_2, \ldots, A_{n1}\}$, and the majority-class elements $\{B_1, B_2, \ldots, B_{n1}\}$. Tomek undersampling for minority classes eliminates all of $\{B_1, B_2, \ldots, B_{n1}\}$.

In the second round of Tomek undersampling, we only need to consider as possible Tomek links the nearest neighbors of $\{A_1, A_2, \ldots, A_{n1}\}$ and any element of the minority class that had one of $\{B_1, B_2, \ldots, B_{n1}\}$ as its nearest neighbor.

In subsequent rounds, consider the minority-class samples from the Tomek's links from the previous round, and the elements of the minority class that had as their nearest neighbor an element of the majority class in the Tomek's links.

In theory there could be more Tomek's links in one round than in the previous round, but in practice they go to zero and the set converges to a set with no Tomek's links.

2.3.3 Cleaning Multiclass Data

Wei (2021) [15] uses something similar to Tomek's links for a multi-class problem with a majority class and multiple minority classes.

- Splits an imbalanced multi-class problem with $n + 1$ classes (n of them being minority) into n imbalanced binary problems for data cleaning.
- Uses cleaning undersampling (similar to Tomek's Links) to remove noisy spots in the data.

2.3.4 Oversampling

Naive oversampling would be to create 99 copies of each of the positive samples, so that the two sets are balanced. That would have exactly the same effect on the loss function, because there would now be 100 times as many samples with $y_i = 1$.

2.3.5 Underssampling

Undersampling would erase 99% of the negative samples so that the classes would be balanced. That seems like a bad idea, because you would lose a lot of information about the majority class.

2.3.6 SMOTE and its Flavors

SMOTE, or Synthetic Minority Oversampling TEchnique, [3] creates extra samples of the minority class, but rather than making exact copies, it finds two similar samples and creates more samples "between" them, with feature values between the values of the two samples. SMOTE only works for continuous features, not for categorical features. Almost all of my features are categorical.

I got this list of flavors of SMOTE from a 2021 review by Mahmudah. [9] I've investigated some of them and given some flesh to some parts of this skeleton.

- SMOTE: Synthetic Minority Oversampling TEchnique [3]
  Uses k-nearest neighbors to find two close positive (minority) samples, and creates a synthetic sample between them. Works on continuous data, not on categorical or binary data.
- ADASYN: ADAptive SYNthetic sampling approach for imbalanced learning. [9]
  Creates synthetic samples based on the level of difficulty in learning the samples of the minority class. A positive samples is "difficult" if it has more negative samples as its nearest neighbors. The more difficult a sample is, the more synthetic copies of that sample ADASYN creates.
- Borderline SMOTE [9]
  Generates synthetic positive samples along the border between the positive and negative classes. Brad's Question: This assumes you know where the border is. I suppose you could do it iteratively.
- Safe-level SMOTE [9]
  When SMOTE finds the nearest positive-class neighbors of a positive sample, it ignores the negative (majority-class) neighbors. [I think this is what it means:] Creating synthetic positive-class samples in a neighborhood with lots of negative samples just makes more of a mess, so this is not considered a "safe" place to make synthetic samples. Safe-level SMOTE creates synthetic positive samples only in majority-positive neighborhoods.
- Relocating-safe-level SMOTE (RSLS) [9]
  Avoids creating synthetic positive samples near negative samples.
- Density-based SMOTE (DBSMOTE) [9]
  Integration of DBSCAN and SMOTE. DBSCAN, Density-Based Spatial Clustering of Application with Noise, discovers clusters with an arbitrary shape (?) DGSMOTE creates synthetic samples at the pseudo-centroids of the clusters of positive samples.
- Adaptive Neighbor SMOTE (ANS) [9]
  Focuses not on -where- to generate synthetic samples, but on -how many- samples to generate in a particular neighborhood.
- D2GAN
  This 2020 article by Zhai [20] builds on the Dual Discriminator Generative Adversarial Nets (D2GAN) paper from 2017 by Nguyen [10]. They want to do better oversampling, comparing D2GAN with SMOTE. I don't understand what this is, but they say SMOTE has three drawbacks:

1. Ignores the probability distribution of minority class samples.
2. Synthetic examples lack diversity.
3. Interating SMOTE many times will give synthetic samples with significant overlap.

This 2022 article by Zhai [19] slightly modifies Zhai's claims against SMOTE.

1. Does not extend the training field of positive samples.
2. Synthetic examples lack diversity.
3. Does not accurately approximate the probability distribution of minority class samples.

The authors propose two new methods of diversity oversampling by generative models, one based on "extreme machine learning autoencoder," and the other based on generative adversarial networks (GAN).

2.3.7 Train/Test Split

The application in Sharififar's 2019 article [12] is digital mapping of farmland, categorizing areas by soil type. Some soil types are rare but significant. This is the first article I've seen that, at the beginning, says that making sure each minority class appears in appropriate distribution in the validation and test sets is an important challenge. They explicitly say that they split 30% for the validation set by taking 30% of each class.

2.3.8 Feature Selection

This 2012 article by Tan [13] introduces a feature selection model specifically for imbalanced data sets. I haven't dug in yet.

2.4 Bagging and Boosting

- Boosting and Bagging [1] [2] [6] [9] [12]

Chapter 3

Datasets

3.1 Repositories

UCI Machine Learning Repository

`https://archive.ics.uci.edu/ml/about.html`

3.2 Useful Datasets

3.3 Articles using Multiple Datasets

Zheng 2021 [21]

- Oversampling, undersampling, and hybrid methods use random sampling ratios. [What? How? I thought the user set the sampling ratios.]
- This paper proposes three algorithms to automatically set the sampling ratios using genetic algorithms.
- Used fourteen datasets, some of which may be useful benchmark datasets.

Wang 2021 [14]

- Uses seven benchmark imbalanced datasets from the UCI machine learning repository
- Implicit regularization for dynamic ensemble selection of classifiers.

Chapter 4

Interesting Papers

4.1 Review Papers

4.1.1 Chawla

Chawla [4] gives an overview of the state of the field in 2004.

- Data Methods
  - Random Oversampling with Replacement
  - Random Oversampling
  - Directed Oversampling
    No new examples are created, but the choice of which ones to replace is informed rather than random.
  - Directed Undersampling
  - Oversampling with informed generation of new samples
  - Combinations of the above
- Algorithmic Methods
  - Adjusting class costs
  - Adjusting the probabilistic estimate at the tree leaf (for tree methods)
  - Recognition-based methods (learning from one class) rather than discrimination-based.
- Issues at 2000 Conference
  -
- Issues at 2003 ICML Conference
  - Probabilistic estimates
  - Pruning
  - Threshold adjusting
  - Cost-matrix adjusting.
- Interesting Topics at 2003 ICML Conference

  - Selective sampling based on query learning (Abe)

- Overlapping Problems
  - Class Imbalance
  - Small Disjunct Problem (?)
  - Rare Cases
  - Data Duplication
  - Overlapping Classes

By 2003, the field started to mature.

4.1.2 Chabbouh 2019

This article [2] has a nice table classifying existing work in imbalanced classification; however, I think much of the information was old in 2019, particularly C4.5, an early decision tree base classifier that may not be used much anymore.

4.1.3 Mahmudah 2021

This article [9] is really a review of current methods. They have some datasets, most public benchmark sets, and throw every combination of tools at them. The "methods" section is really an overview of current methods.

Has a section on techniques for feature extraction (feature engineering?) by dimensionality reduction, not particularly related to imbalanced data.

Bibliography

[1] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. "A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data". In: *SIGKDD Explor. Newsl.* 6.1 (June 2004), pp. 20–29. ISSN: 1931-0145. DOI: 10.1145/1007730.1007735. URL: https://doi.org/10.1145/1007730.1007735.

[2] Marwa Chabbouh et al. "Multi-objective evolution of oblique decision trees for imbalanced data binary classification". In: *Swarm and Evolutionary Computation* 49 (2019), pp. 1–22. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2019.05.005. URL: https://www.sciencedirect.com/science/article/pii/S2210650218305054.

[3] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (June 2002), pp. 321–357. ISSN: 1076-9757. DOI: 10.1613/jair.953. URL: http://dx.doi.org/10.1613/jair.953.

[4] Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. "Editorial: Special Issue on Learning from Imbalanced Data Sets". In: *SIGKDD Explor. Newsl.* 6.1 (June 2004), pp. 1–6. ISSN: 1931-0145. DOI: 10.1145/1007730.1007733. URL: https://doi.org/10.1145/1007730.1007733.

[5] Chao Chen and Mei-Ling Shyu. "Clustering-based binary-class classification for imbalanced data sets". In: *2011 IEEE International Conference on Information Reuse Integration*. 2011, pp. 384–389. DOI: 10.1109/IRI.2011.6009578.

[6] Damien Dablain, Bartosz Krawczyk, and Nitesh V. Chawla. "DeepSMOTE: Fusing Deep Learning and SMOTE for Imbalanced Data." In: (2021).

[7] Namgil Lee, Heejung Yang, and Hojin Yoo. *A surrogate loss function for optimization of F_β score in binary classification with imbalanced data*. 2021. arXiv: 2104.01459 [cs.LG].

[8] T. Lin et al. "Focal Loss for Dense Object Detection." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence, Pattern Analysis and Machine Intelligence, IEEE Transactions on, IEEE Trans. Pattern Anal. Mach. Intell* 42.2 (2020), pp. 318–327. ISSN: 0162-8828.

[9] Kunti Robiatul Mahmudah et al. "Classification of Imbalanced Data Represented as Binary Features". In: *Applied Sciences* 11.17 (2021). ISSN: 2076-3417. DOI: 10.3390/app11177825. URL: https://www.mdpi.com/2076-3417/11/17/7825.

[10] Tu Dinh Nguyen et al. *Dual Discriminator Generative Adversarial Nets*. 2017. arXiv: 1709.03831 [cs.LG].

[11] Edwin P. D. Pednault, Barry K. Rosen, and Chidanand Apté. "Handling Imbalanced Data Sets in Insurance Risk Modeling". In: AAAI Workshop on Learning from Imbalanced Data Sets, 2000.

[12] Amin Sharififar et al. "Addressing the issue of digital mapping of soil classes with imbalanced class observations." In: *Geoderma* 350 (2019), pp. 84–92. ISSN: 0016-7061.

[13] Ding-Wen Tan et al. "A feature selection model for binary classification of imbalanced data based on preference for target instances". In: *2012 4th Conference on Data Mining and Optimization (DMO)*. 2012, pp. 35–42. DOI: 10.1109/DMO.2012.6329795.

[14] Chen Wang et al. "Adaptive ensemble of classifiers with regularization for imbalanced data classification". In: *Information Fusion* 69 (2021), pp. 81–102. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2020.10.017. URL: https://www.sciencedirect.com/science/article/pii/S1566253520303869.

[15] Jianan Wei et al. "New imbalanced bearing fault diagnosis method based on Sample-characteristic Oversampling TechniquE (SCOTE) and multi-class LS-SVM". In: *Applied Soft Computing* 101 (2021), p. 107043. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2020.107043. URL: https://www.sciencedirect.com/science/article/pii/S1568494620309819.

[16] Gary Weiss. "Learning to Predict Extremely Rare Events". In: *AAAI Workshop on Learning from Imbalanced Data Sets* (May 2000).

[17] Rongjie Yu et al. "Convolutional neural networks with refined loss functions for the real-time crash risk analysis." In: *Transportation Research Part C* 119 (2020). ISSN: 0968-090X.

[18] JUN-HAI ZHAI et al. "A Three-stage Method for Classification of Binary Imbalanced Big Data". In: *2020 International Conference on Machine Learning and Cybernetics (ICMLC)*. 2020, pp. 207–212. DOI: 10.1109/ICMLC51923.2020.9469568.

[19] Junhai Zhai, Jiaxing Qi, and Chu Shen. "Binary imbalanced data classification based on diversity oversampling by generative models". In: *Information Sciences* 585 (2022), pp. 313–343. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2021.11.058. URL: https://www.sciencedirect.com/science/article/pii/S0020025521011804.

[20] Junhai Zhai, Jiaxing Qi, and Sufang Zhang. "Binary Imbalanced Data Classification Based on Modified D2GAN Oversampling and Classifier Fusion". In: *IEEE Access* 8 (2020), pp. 169456–169469. DOI: 10.1109/ACCESS.2020.3023949.

[21] Ming Zheng et al. "An automatic sampling ratio detection method based on genetic algorithm for imbalanced data classification". In: *Knowledge-Based Systems* 216 (2021), p. 106800. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.106800. URL: https://www.sciencedirect.com/science/article/pii/S0950705121000630.