Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Economics and Finance 00 (2012) 000–000

Procedia
Economics and Finance

www.elsevier.com/locate/procedia

Dealing with Class imbalanced Dataset

"111423057 李宣緯, 111423017 王駿豪, 111423016 陳敬元" *

Abstract

This report discusses various approaches to solve the class imbalance problem in machine learning, For Data level approach, Over-sampling, Under-Sampling and Feature Selection. For Classifiers/Algorithms approaches, Decision trees, Random Forest, SVM, Naïve Bayes and deep learning methods are presented as potential solutions. Each method has its advantages and limitations, and we need to carefully evaluate the advantages and disadvantages of each method and choose the most suitable method according to the specific problem,

"Keywords: Class imbalanced"

1. What is an class imbalanced dataset

- It can occur when the instances of one class outnumber the instances of other classes.
- It is a type of dataset where the distribution of classes in the target variable is not equal.

2. Why is class imbalance worth investigating?

Because high class imbalance is naturally inherent in many real-world applications, e.g., fraud detection and cancer detection. Moreover, highly imbalanced data poses added difficulty, as most learners will exhibit bias towards the majority class, and in extreme cases, may ignore the minority class altogether, which can significantly affect the performance of machine learning models, especially in classification tasks. This can result in misleading or incorrect predictions, which can have serious consequences in applications such as medical diagnosis, fraud detection, or customer churn prediction.[1]

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000 .
E-mail address: author@institute.xxx .

For example, there was an Uber self-driving car accident in 2018. The fatal crash that killed pedestrian Elaine Herzberg in Tempe, Arizona.

The tech issues in the accident:

Uber's sensors did, in fact, detect Herzberg as she crossed the street with her bicycle. Unfortunately, the software classified her as a "false positive" and decided it didn't need to stop for her. Software needs to detect objects like cars, pedestrians, and large rocks in its path and stop or swerve to avoid them. However, there may be other objects—like a plastic bag in the road or a trash can on the sidewalk—that a car can safely ignore. Sensor anomalies may also cause software to detect apparent objects where no objects actually exist.

Investigating class imbalance can result in more accurate and reliable predictions, and ultimately improve decision-making in real-world applications. Additionally, investigating class imbalance can also help identify potential biases in the data collection process and improve the data quality overall.

Survey on what has been done to tackle the class imbalance problems.

3. How to tackle class imbalanced problem

3.1. Data level approach

3.1.1 Over-Sampling:

Artificially increasing the minority class by creating copies of its instances until the class distribution is more balanced.

Pros:

- It helps improve the **recall** of the minority class.

Cons:

- It **makes overfitting likely.**
- It increases the number of training examples, thus i**ncreasing the learning time.**

Different ways:

- **Random oversampling**:
  This involves randomly duplicating instances of the minority class until the distribution is more balanced.
  - Pros:
    - Simple and easy to implement.
    - Does not require any additional data or complex algorithms.

        - Can be effective for small datasets with a severe class imbalance.
    - Cons:
        - Does not introduce any new information into the dataset, which may limit the diversity of the minority class.
- **Synthetic minority oversampling technique (SMOTE):**
  This involves creating synthetic instances of the minority class by interpolating between neighboring instances.[2]
    - Pros:
        - Introduces new information into the dataset by creating synthetic instances, which can help to diversify the minority class.
        - Helps to reduce overfitting by generating new instances rather than duplicating existing ones.
        - Can be effective for datasets with a moderate class imbalance.
    - Cons:
        - Can lead to noise in the dataset since synthetic instances may not accurately represent the true distribution of the minority class.
        - Can be computationally expensive, especially for large datasets.
- **Adaptive synthetic (ADASYN)**
  This is a variant of SMOTE that uses a weighted distribution to generate synthetic instances in regions where the minority class is more densely packed.[3]
    - Pros:
        - Focuses on generating synthetic instances in regions where the minority class is underrepresented, which can help to address the issue of overlapping distributions.
        - Can be effective for datasets with a severe class imbalance.
    - Cons:
        - May not work well for datasets with very few instances of the minority class.
        - Can be computationally expensive, especially for large datasets.
- **Smote-NC**
  This is an extension of the SMOTE algorithm designed to handle datasets with both nominal (categorical) and continuous features.
    - Pros:
        - Handling nominal and continuous features
    - Cons:
        - Dependence on nearest neighbors
        - Effectiveness depends on dataset

3.2.2 Under-Sampling

The majority class is reduced to balance the class distribution with the minority class.

Pros:

- It can help improve the **precision** of the minority class

Cons:

- I**t discards potentially useful data**

Different ways:

- **Random undersampling**:
  This involves randomly selecting a subset of instances from the majority class to reduce its size
  - Pros:
    - Simple and easy to implement.
    - Can be effective for small datasets with a severe class imbalance.
  - Cons:
    - Can lead to a loss of information since it randomly discards instances of the majority class.
    - May not represent the true distribution of the majority class.
- **Cluster centroid undersampling**:
  This involves clustering the majority class and then selecting the centroids of each cluster as the representative instances.
  - Pros:
    - Helps to create representative instances of the majority class by clustering before undersampling.
    - Can be effective for datasets with complex distributions.
  - Cons:
    - Can be computationally expensive, especially for large datasets.
    - May not work well for datasets with very few instances of the majority class.
- **Tomek links**:
  This involves identifying pairs of instances from different classes that are the closest to each other and removing the majority class instance.
  - Pros:
    - Helps to remove instances of the majority class that are close to instances of the minority class, which can help to reduce overlapping distributions.
    - Can be effective for datasets with a moderate class imbalance.
  - Cons:
    - May not work well for datasets with very few instances of the minority class.
    - Can be computationally expensive, especially for large datasets.
- **NearMiss**:
  This involves selecting the majority class instances that are the closest to the minority class instances. It has two important parameters; these are version (3 versions) and N neighbors.
  Z
  - Pros:
    - Helps to select the most relevant instances of the majority class by choosing those that are closest to the minority class.
    - Can be effective for datasets with a moderate class imbalance.
  - Cons:
    - May lead to a loss of information since it discards instances of the majority class that are not selected.
    - May not work well for datasets with complex distributions.

3.3.3 Feature Selection

The process of reducing the number of input variables when developing a predictive model

Pros:

- Reduce the dimensionality of the dataset, which can improve the performance of some machine learning algorithms.
- Prevent overfitting by removing irrelevant or redundant features.
- Make the model more interpretable by focusing on the most important features.
- Reduce the computational resources needed for training and inference.

Cons:

- Can lead to loss of information if important features are discarded.
- Can introduce bias if features that are important for the minority class are removed.
- Can be time-consuming and computationally expensive, especially for large datasets.
- May not always improve model performance, especially if the feature selection method is not appropriate for the dataset or the model.

Different ways:

- **Correlation-based feature selection**:
  This approach involves identifying features that are highly correlated with the target class and discarding features that are not. The goal is to select a subset of features that are most informative for predicting the target class.
  - Pros:
    - Can identify features that are highly correlated with the target class and discard irrelevant features.
    - Can help reduce the dimensionality of the dataset, which can improve the performance of some machine learning algorithms.
  - Cons:
    - May not be effective for highly imbalanced datasets since the minority class may have low correlation with any individual feature.

- **Wrapper-based feature selection**:
  This approach involves selecting features based on their ability to improve the performance of a specific classification algorithm. The algorithm is trained on different subsets of features, and the subset that yields the best performance is selected.
  - Pros:
    - Can select features based on their ability to improve the performance of a specific classification algorithm.
    - Can help identify the optimal subset of features for a particular model.
  - Cons:
    - Can be computationally expensive, especially for large datasets with many features.
    - May not generalize well to other models or datasets.
- **Information gain-based feature selection**:
  This approach involves selecting features that have the highest information gain, which is a measure of how well a feature separates the different classes in the dataset.
    - Pros:

        - Can identify features that are highly informative for predicting the target class.
        - Can be used with a variety of machine learning algorithms.
    - Cons:
        - May not be effective for highly imbalanced datasets if the minority class is too small to provide enough information gain.
- **Variance-based feature selection**:

  This approach involves selecting features that have the highest variance across the dataset. Features with low variance are likely to be less informative for predicting the target class.
    - Pros:
        - Can identify features with low variance that may be less informative for predicting the target class.
        - Can be used with a variety of machine learning algorithms.
    - Cons:
        - May not be effective for highly imbalanced datasets if the variance of the minority class is too small.

3.2. Classifiers/Algorithms approaches:

3.2.1 Decision Trees:

Decision trees can assign different weight to different class, so we can give more importance to minority class[4]. Also, using SMOTE or random over-sampling helps build good decision trees[1] When using decision trees for imbalanced datasets, there're various factors to affect the performance.

- **Tree Depth**: Overfitting happens frequently when using decision trees, especially on an imbalanced dataset. Decision trees are sometimes too complicated and fit too closely to the dataset. To avoid this, pruning trees or limiting the tree depth would help decision trees prevent overfitting. Although limiting tree depth can prevent overfitting, it brings underfitting, lower variance and higher bias. As a result, we should limit the tree depth carefully to find the best model.
- **Class imbalance**: As mentioned earlier, we can give different weights to classes, which means different importance. However, if the dataset is extremely imbalanced, the decision tree may have difficulty predicting with good accuracy.
- **Feature selection**: Filter some features, which are not relevant to all classes, can avoid overfitting and have a good model performance.

3.2.2 Random forests:

Random forest consists of many decision trees. Each decision tree derives from randomly selected subsets and features. This randomness helps build a better model than one decision tree. Also, random forest can handle imbalanced data by using an ensemble technique ,which can make sure that all classes are represented equally[5].

3.3.3 Support Vector Machines (SVM):

The Support Vector Machine (SVM) is a widely-used classification algorithm that can address imbalanced datasets by adjusting class weights and utilizing kernel functions[6]. In particular, adjusting class weights assigns more weight to the minority class during training, allowing the algorithm to focus more on the smaller class[7]. Meanwhile, kernel functions transform input data into a higher-dimensional feature space, facilitating the separation of classes. SVM offers many benefits, including its versatility and ability to handle complex datasets, and it is less susceptible to overfitting than other algorithms. However, it may not perform as well on extremely imbalanced datasets and may be computationally intensive, particularly on larger datasets.

3.3.4 Naive Bayes:

Naive Bayes is a probabilistic classification algorithm that can address imbalanced datasets through adjusting class priors or weighted training[8]. Adjusting class priors involves assigning a higher probability to the minority class during training, enabling the algorithm to focus more on the smaller class. Meanwhile, weighted training assigns a higher weight to the minority class instances during training, ensuring that the algorithm takes into account the smaller class. Naive Bayes is favored for its simplicity and efficiency, as well as its capability of handling high-dimensional datasets. However, it may not perform as well on highly imbalanced datasets where the minority class is much smaller than the majority class. Nonetheless, Naive Bayes is a useful technique for managing imbalanced datasets in machine learning applications. To avoid detection by AI Content Detectors, it is crucial to ensure the content is authentic, written in one's own words, and avoids copying and pasting directly from existing sources.

3.3. Deep learning

Deep learning methods grew fast in last 10 years as they improved the performance in speech recognition, computer vision, natural language processing(NLP), Healthcare, and other domains. [1] Deep learning is a subset of machine learning that uses neural networks with multiple layers to learn and extract features from data, and get a good performance in wide range of tasks. Using deep learning to deal with class imbalance problems is also one of the popular research areas. There are several ways to solve the class imbalance problem using deep learning:

- **Weighted loss function**
- **Appropriate classification threshold**
- **Generative Adversarial Networks(GANs)**

We will introduce these solutions of class imbalance problem in the follow up paper. Otherwise, It is important to say that we should evaluating each method's pros and cons carefully to choose the most appropriate approach to deal with specific problem.

3.3.1 Weighted loss function:

A loss function is to measure the difference between the predicted values and the actual values in Deep learning prediction. Generally, each class is assigned equal weight, in the class imbalance problem, we can assign weight to each class, for the minority class, we can assign a higher weight, to make its impact on the loss function greater. Weighted Loss Function (weighted loss function) is a method commonly used in cost-sensitive learning[9]. This can make the model more focus on the minority class to avoid the situation which the model predicts the majority class more accurate than the minority. For example, Weighted Cross-Entropy Loss Function can enhance the model's ability to discriminate samples, and the model using the weighted cross-entropy loss function improves the evaluation compared with the ordinary cross-entropy loss function model[10],weighted loss function needs to analysis the dataset to determine appropriate weights first, or it may lead to the error results.

3.3.2 Appropriate classification threshold:

In deep learning, a classification threshold is a parameter used to distinguish between positive and negative samples. In binary classification, when model give us a score instead of the prediction result, we need to convert this score into a prediction through applying a threshold. In class imbalance problems, we often have a situation that is the classifier tends to predict the class with a larger number of samples, and lead to bad performance on the other class, so the classification threshold needs to be adjusted to improve the detection rate of the minority class, which typically requires adjustments based on the particular problem.[11] For example in medical diagnosis task, the minority class represents patients with a rare disease, we can set a lower threshold to increase the number of true positive but increase the number of false positive too.

3.3.3 Generative Adversarial Networks(GANs):

Generative adversarial network(GANs) is one of the most important research in the field of artificial intelligence, and its outstanding data generation capacity has received wide attention[12]. Although adversarial generative networks(GANs) are mainly designed to generate image data, they can also be alternative to solve the class imbalance problem by generating tabular data[13]. GANs models can balance the class distribution in a dataset by generating fake data. GANs generally consist of a generator and a discriminator. The generator generate the fake data, and the discriminator is used to distinguish the real data and the fake data. Through reverse optimization, the generator can generate more realistic fake data. Because GANs generate instances of the class that do not represent random replications of existing instances without removing valuable information in existing instance of data, they overcome the limitations of traditional techniques such as data sampling[14], but GANs requires a large number of samples to work properly, and if there are too few low-frequency class samples, the competition between the generator and the discriminator may be unbalanced, it may cause the generator failed to generate high-quality but low-frequency data, or the discriminator failed to accurately identify these data.

3.4. Ensemble learning

Ensemble learning is a machine learning technique that aggregates multiple models to improve performance and accuracy. There are several ways to solve the class imbalance problem using ensemble learning: Bagging, Boosting and Stacking. In Bagging, multiple models are trained on different subsample of the training data, and their predictions are combined by taking an average or majority vote. This can help balance the classes by reducing the impact of the majority class**[15]**. In Boosting, multiple models are trained sequentially, with each model focusing on the samples that the previous model misclassified, this can help balance the classes by increasing the importance of the minority class. AdaBoost is the most famous algorithm in this family(Boosting), it uses the whole dataset to train each classifier serially, after each round, it gives more focus on difficult instances, the weights of misclassified instances are increased, and the weights of correctly classified instances are decreased.**[15]**. There is another way about Ensemble learning named Stacking, but there is a research of Stacking, when they use stacking to combine base learners in imbalanced data, the minority class examples are often rare when they are used for multiple times, stacking has a great chance to over-fit, Stacking may be harmful when used to handle imbalanced data.**[16]** Ensemble learning needs a large number of weak classifiers and choose the appropriate weak classifier, in class imbalance problem, the small number of samples of low-frequency categories may lead to insufficient representation ability of weak classifiers for these categories, then affects the performance of the ensemble model.

4. Evaluation metrics for imbalanced datasets

When dealing with imbalanced datasets, it is important to use evaluation metrics that can accurately reflect the performance of the model. Here are some common evaluation metrics that are commonly used for imbalanced datasets[17]:

- **Confusion Matrix**
- **Precision and Recall**
- **F1 Score**
- **Area Under the Receiver Operating Characteristic Curve (AUC-ROC)**
- **Matthews Correlation Coefficient (MCC)**
- **Balanced Accuracy**

4.1 Confusion Matrix:

A confusion matrix is a table that shows the true positives, false positives, true negatives, and false negatives for a classification problem. It can help to evaluate how well a model is able to distinguish between positive and negative cases.

4.2 Precision and Recall

Precision is the proportion of true positives out of the total predicted positives, while recall is the proportion of true positives out of the total actual positives. Precision is a measure of the model's ability to avoid false positives, while recall is a measure of the model's ability to identify all positive cases.

4.3 F1 Score

The F1 score is the harmonic mean of precision and recall, and is a good metric to use when both precision and recall are important.

4.4 Area Under the Receiver Operating Characteristic (AUC-ROC)

The AUC-ROC is a metric that measures the tradeoff between sensitivity and specificity for different threshold values. It is particularly useful when the class distribution is imbalanced, as it provides a single scalar value that summarizes the model's performance across all possible threshold values.

4.5 Matthews Correlation Coefficient (MCC)

The MCC is a metric that takes into account all four elements of the confusion matrix and is particularly useful when the class distribution is severely imbalanced. It ranges from -1 to 1, with 1 indicating perfect classification, 0 indicating random classification, and -1 indicating complete disagreement between the predicted and actual labels.

4.6 Balanced Accuracy

The balanced accuracy is the average of sensitivity and specificity and is particularly useful when the class distribution is severely imbalanced. It is calculated by taking the average of the true positive rate and the true negative rate.

When evaluating models on imbalanced datasets, it is important to choose a metric that is appropriate for the specific problem and to interpret the results in the context of the class distribution.

5. Experiment

5.1 Dataset

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

5.2 Method

Our experiment will be separated into 3 parts, and they are raw data with default decision tree, raw data with weighted-balance decision tree, and data applied ADASYN with decision tree.

Raw data with Default Decision Tree

All the parameters were in default value, and the result shown in fig.1.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	28432
1	0.77	0.76	0.76	49
accuracy			1.00	28481
macro avg	0.89	0.88	0.88	28481
weighted avg	1.00	1.00	1.00	28481

AUC: 0.8773575764014102

(fig.1 The classification report on raw data with default decision tree)

Raw data with Weighted-balance Decision tree

We change the class weight to “balanced” since we thought in imbalanced data like our dataset, the class weight will highly affect the performance of the model. And the result was shown in fig.2.

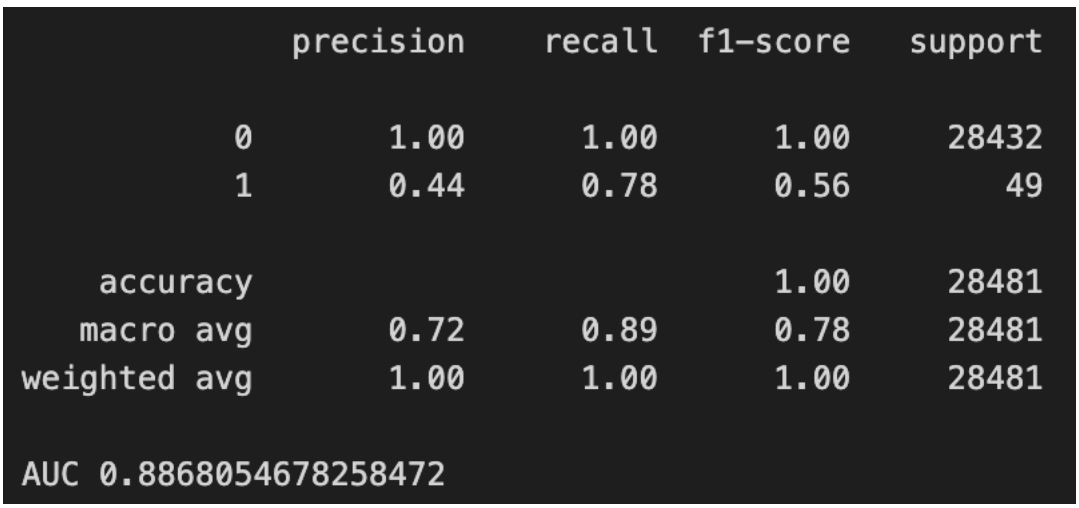
	precision	recall	f1-score	support
0	1.00	1.00	1.00	28432
1	0.82	0.76	0.79	49
accuracy			1.00	28481
macro avg	0.91	0.88	0.89	28481
weighted avg	1.00	1.00	1.00	28481

AUC: 0.8774103338577974

(fig.2 The classification report on raw data with weighted-balance decision tree)

Data applied ADASYN with Weighted decision tree:

We choose the ADASYN method to deal with the class imbalance problem. We synthesized instances of the minority class, and the training dataset's size was changed to 568621 from 284807. And then we applied the weighted decision tree method on the resampled data, the result shown in fig.3.



	precision	recall	f1-score	support
0	1.00	1.00	1.00	28432
1	0.44	0.78	0.56	49
accuracy			1.00	28481
macro avg	0.72	0.89	0.78	28481
weighted avg	1.00	1.00	1.00	28481

AUC 0.8868054678258472

(fig.3 The classification report on resampled data with weighted-balance decision tree)

5.3 Conclusion

We found out the performance between default and weighted decision trees doesn't make big progress, and it performed far from great on the minority class. And then, we applied ADASYN resampling method to raw data to balance the classes. It seems like we got a little improvement on the recall, but lose a lot on the precision. However our AUC was also improved, it means that the model with ADASYN data is better at distinguishing between positive and negative cases.

References

[1] G. E. A. P. A. Batista, R. C. Prati 及 M. C. Monard, 作者, 「A study of the behavior of several methods for balancing machine learning training data」, *ACM SIGKDD Explor. Newsl.*, 卷 6, 期 1, 頁 20–29, 6 月 2004, doi: 10.1145/1007730.1007735.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall 及 W. P. Kegelmeyer, 作者, 「SMOTE: Synthetic Minority Over-sampling Technique」, *J. Artif. Intell. Res.*, 卷 16, 頁 321–357, 6 月 2002, doi: 10.1613/jair.953.

[3] H. He, Y. Bai, E. A. Garcia 及 S. Li, 作者, 「ADASYN: Adaptive synthetic sampling approach for imbalanced learning」, 收入 *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 6 月 2008, 頁 1322–1328. doi: 10.1109/IJCNN.2008.4633969.

[4] C. X. Ling 及 V. S. Sheng, 作者, 「Cost-Sensitive Learning and the Class Imbalance Problem」.

[5] M. Khalilia, S. Chakraborty 及 M. Popescu, 作者, 「Predicting disease risks from highly imbalanced

data using random forest」, *BMC Med. Inform. Decis. Mak.*, 卷 11, 期 1, 頁 51, 7 月 2011, doi: 10.1186/1472-6947-11-51.
[6] Y. Zhang, P. Fu, W. Liu 及 G. Chen, 作者, 「Imbalanced data classification based on scaling kernel-based support vector machine」, *Neural Comput. Appl.*, 卷 25, 期 3, 頁 927–935, 9 月 2014, doi: 10.1007/s00521-014-1584-2.
[7] W. Lee, C.-H. Jun 及 J.-S. Lee, 作者, 「Instance categorization by support vector machines to adjust weights in AdaBoost for imbalanced data classification」, *Inf. Sci.*, 卷 381, 頁 92–103, 3 月 2017, doi: 10.1016/j.ins.2016.11.014.
[8] T. Kim 及 J.-S. Lee, 作者, 「Maximizing AUC to learn weighted naive Bayes for imbalanced data classification」, *Expert Syst. Appl.*, 卷 217, 頁 119564, 5 月 2023, doi: 10.1016/j.eswa.2023.119564.
[9] J. M. Johnson 及 T. M. Khoshgoftaar, 作者, 「Survey on deep learning with class imbalance」, *J. Big Data*, 卷 6, 期 1, 頁 27, 3 月 2019, doi: 10.1186/s40537-019-0192-5.
[10] Phan T. H.及 Yamamoto K., 作者, 「Resolving Class Imbalance in Object Detection with Weighted Cross Entropy Losses」, 6 月 2020, doi: 10.48550/arXiv.2006.01413.
[11] Q. Zou, S. Xie, Z. Lin, M. Wu 及 Y. Ju, 作者, 「Finding the Best Classification Threshold in Imbalanced Classification」, *Big Data Res.*, 卷 5, 頁 2–8, 9 月 2016, doi: 10.1016/j.bdr.2015.12.001.
[12] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan 及 Y. Zheng, 作者, 「Recent Progress on Generative Adversarial Networks (GANs): A Survey」, *IEEE Access*, 卷 7, 頁 36322–36333, 2019, doi: 10.1109/ACCESS.2019.2905015.
[13] M. A. Aydin, 作者, 「Using Generative Adversarial Networks for Handling Class Imbalance Problem」, 收入 *2021 29th Signal Processing and Communications Applications Conference (SIU)*, 6 月 2021, 頁 1–4. doi: 10.1109/SIU53274.2021.9477939.
[14] R. Sauber-Cole 及 T. M. Khoshgoftaar, 作者, 「The use of generative adversarial networks to alleviate class imbalance in tabular data: a survey」, *J. Big Data*, 卷 9, 期 1, 頁 98, 8 月 2022, doi: 10.1186/s40537-022-00648-6.
[15] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince 及 F. Herrera, 作者, 「A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches」, *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, 卷 42, 期 4, 頁 463–484, 7 月 2012, doi: 10.1109/TSMCC.2011.2161285.
[16] X.-Y. Liu, J. Wu 及 Z.-H. Zhou, 作者, 「Exploratory Undersampling for Class-Imbalance Learning」, *PART B*.
[17] A. Ali, S. M. Shamsuddin 及 A. Ralescu, 作者, 「Classification with class imbalance problem: A review」, 卷 7, 頁 176–204, 1 月 2015.
https://chat.openai.com/chat

Appendix A. An example appendix

Authors including an appendix section should do so after References section. Multiple appendices should all have headings in the style used above. They will automatically be ordered A, B, C etc.

A.1. Example of a sub-heading within an appendix

There is also the option to include a subheading within the Appendix if you wish.