# Unbalanced Datasets

Matthew Rui

December 2024

# 1 Unbalanced Datasets

## 1.1 Introduction

In many classification tasks in Deep Learning, there is an assumption that the classes are roughly evenly represented in the data [4]. Many model architecture and frameworks are built upon this assumption. For example, the Bayes classifier (label $y = 1$ if $p > 0.5$) implicitly assumes equal class priors.

However, real-world datasets often exhibit imbalanced class distributions, where some classes (majority classes) have a large number of samples, while others (minority classes) are underrepresented. This imbalance is particularly problematic when the minority class holds critical importance, such as identifying cancer cases in a large dataset of patient records. For this project, a dataset of credit card transaction details was studied, where the minority class (fraudulent transactions) only represented 0.17% of the dataset.

## 1.2 Alternative Performance Metric

Unlike normal classification tasks, unbalanced datasets generally require the usage of a performance metric other than accuracy. This is because the "baseline" accuracy of a deterministic classification scheme is much higher than 0.5 (for binary classification). For the credit card fraud dataset, a model that always classified features as "no fraud" would be accurate 99.83% of the

time while being entirely useless.

The most common alternative notions of performance are precision and recall. Precision is the proportion of true positive instances out of all predicted positive instances. Recall is the proportion of correctly predicted positive instances out of all actual positive instances. These metrics intuitively capture what usually matters most for unbalanced classification: accurately predicting as much of the minority class as possible. The Precision-Recall Curve (PRC) demonstrates the tradeoff varying the classification threshold (the cutoff value that specifies how instances are assigned labels) has between precision and recall.

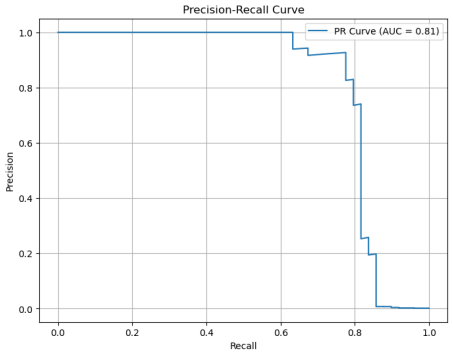


Figure 1: Sample Precision-Recall Curve (PRC)

The AUPRC (Area under PRC) provides a metric for evaluating model performance [2]. A better model will have a better AUPRC, as it would indicate better pairs of precision and recall values can be obtained.

## 1.3 Undersampling

In a undersampling scheme, we construct a training dataset containing only a proper subset of the majority instances. These methods attempt to even out the class sizes, which focuses the classifier to pay more attention to the minority class. The most naive approach is *random undersampling*, where a random subset of majority instances are kept in the training dataset [6]. However, this method may result in signals of the majority class to be removed from the training dataset.

A more targeted approach is given by undersampling with *tomek links*. A tomek link is a pair of points, one in each class, such that each point is the nearest neighbor of each other. In this approach, the majority instance of all such tomek links are removed from the dataset [1]. Tomek links can be intuitively thought as instances right on the classification boundary and thus might make the decision boundry too noisy. Removing these links would thus "clear up" the boundary, enhancing model performance.

This approach can be generalized to the KNN-based approach [8]. For each point of the majority class, we remove them from the training dataset if more than $t$ of its $k$ nearest neighbors are in the minority class, where $t$ and $k$ are hyperparameters.

## 1.4 Oversampling

Undersampling methods all risk potentially dropping important features and signals from the training dataset. Oversampling methods also attempt to even out the class imbalance, but by increasing the number of minority instances, avoiding the potential downsides of undersampling. The most naive oversampling approach is to duplicate minority instances, but this often lead to overfitting issues and provides no new information to the model.

The alternative approach is to generate synthetic minority instances. In Synthetic Minority Over-sampling Technique (SMOTE), we create new minority instances by interpolating between minority points and their nearest minority neighbors. For some minority instance $x$, we first find its $k$ nearest minority neighbors $x_1 ... x_k$ and randomly choose one of the neighbors $x_i$. We can then interpolate these two points so that $x_{new} = x + \lambda(x_i - x)$ where $\lambda \sim \text{Uniform}(0, 1)$ [3].

In practice, oversampling techniques are often used in conjunction with undersampling. This is because for extreme class imbalances, we can only generate so many synthetic minority instances and likely will be unable to even out the class imbalance solely through SMOTE. Since over/undersampling deal solely with the underlying dataset, they can be widely applied to any classifier.

### 1.5 Focal Loss

For Deep Learning classifiers utilizing cross-entropy loss, extremely unbalanced datasets pose a unique problem as easily classifiable majority instances will represent the majority of the loss. Thus the minority instances might be overlooked when solely focused on minimizing cross-entropy loss.

Following [7], let $p_t = p$ if $y = 1$ and $p_t = 1 - p$ if $y = 0$, where $p$ is the probability that a instance is labeled 1. We define $\alpha_t$ analogously: $\alpha_t = \alpha$ if $y = 1$ and $\alpha_t = 1 - \alpha$ if $y = 0$, where $\alpha$ is a fixed hyperparameter. We then rescale cross entropy loss, so that $FL = -\alpha_t(1 - p_t)^\gamma \log(p_t)$, where $\gamma$ is another hyperparameter. If $p_t$ is large (the instance is easily classifiable), the loss contribution of that instance is minimzed by the $(1 - p_t)^\gamma$ term. However if $p_t$ is small, the loss contribution will be roughly maintained. $\gamma$ can be thought of as how strong we want to reduce the loss contributions of easily classification instances. $\alpha_t$ is a normalization constant.

## 2 Experiment

### 2.1 Data Processing

For this project, we applied various unbalanced classification techniques on a dataset of credit card transactions labeled fraud/no fraud [5]. This dataset contains $284,807$ instances with only 492 instances labeled as fraudulent. Each instance contains numerical values from 28 PCA dimensions, as well as two additional values for the amount of the transaction and time elapsed between each transaction and the very first transaction in the dataset.

Before any over/undersampling scheme was applied, a stratified train/test split was applied, meaning that the proportion of minority instances is maintained to be roughly equal in the resulting datasets. Any over/undersampling was then conducted only on the train dataset, and a further stratified split was preformed to obtain the train and validation datasets. It is important for the over/undersampling to occur after the train/test split, to ensure that our models are evaluated on the unbalanced dataset, but before the train/validation split, to ensure the validation dataset resembles the train dataset. Finally, all three datasets were normalized.

## 2.2 Results

We trained a Neural Network binary classifier with 3 hidden layer using Population-Based Training and Adam with Weight Decay. While these classifiers were ultimately evaluated using the AUPRC on the test dataset, the training and validation metrics still used accuracy. This was done because using AUPRC as the train/validation metric was significantly less efficient without providing better results. We first used cross-entropy loss to test the various over/undersampling methods.

| **Method** | **Parameters** | **AUPRC** |
|---|---|---|
| Baseline | – | **0.8028** |
| Random Undersampling | $R = 1$ | 0.5551 |
| | $R = 2$ | 0.5921 |
| | $R = 3$ | 0.6470 |
| | $R = 4$ | 0.6943 |
| | $R = 5$ | 0.7188 |
| | $R = 6$ | 0.7110 |
| | $R = 7$ | 0.6973 |
| | $R = 8$ | **0.7639** |
| Tomek Links | – | **0.8158** |
| KNN | $k = 50$ | 0.8165 |
| | $k = 100$ | **0.8169** |
| | $k = 150$ | 0.8007 |
| | $k = 200$ | 0.8124 |
| SMOTE | $N = 2$ | 0.8127 |
| | $N = 3$ | 0.8121 |
| | $N = 4$ | 0.8206 |
| | $N = 5$ | 0.8172 |
| | $N = 6$ | 0.8194 |
| | $N = 7$ | 0.8216 |
| | $N = 8$ | **0.8220** |
| | $N = 9$ | 0.8172 |
| | $N = 10$ | 0.8203 |

Table 1: Over/undersampling

The baseline model is the classifer with no over/undersampling methods applied. The parameter $R$ for random sampling represents the size of the under-sampled majority class, where $R$ is the ratio of the majority to minority class in the sample dataset. $k$ is defined as before for the KNN method (we use Euclidean distance for all nearest-neighbor calculations) and $t = 1$ was used for all experiments. The parameter $N$ for SMOTE refers to the output minority instances being $(N * 100)\%$ the size of the original minority instances. For example, $N = 2$ means we generate an equal amount of synthetic instances as original instances. $k = 5$ was used for all experiments [3].

It's surprising that random undersampling performs much worse than the baseline model, especially for the smaller $R$ values. However, as there only exists roughly 400 minority instances in the training dataset, random undersampling with even $R = 10$ results in a dataset of less than 10000 values. This underscores the difficulty and potential downsides in applying undersampling methods. The tomek link and KNN approaches were much more successful, resulting in moderate boosts to overall model AUPRC compared with the baseline. The tomek link approach only removed 71 majority instances and the KNN approach with $k = 200$ only removed roughly $50,000$ of the roughly $250,000$ majority instances, and thus avoided the pitfalls of the random undersampling approach.

SMOTE was also successful, as all the models exhibited a moderate performance boost. In particular, models with parameters $N = 6$ to $N = 10$ received the largest boost. While this suggests that larger parameter values may improve our model performance, we follow the current literature and don't apply $N$ values that are too large to prevent possible model overfitting [3].

As both undersampling and oversampling seem to boost model performance, we now train classifiers using both SMOTE and undersampling methods. For SMOTE with Random Undersampling, the optimal $N$ parameter of SMOTE was tuned for each $R$. The optimal $N$ was likewise tuned for SMOTE with Tomek Links and for each $k$ value of SMOTE with KNN. We note that for all three models SMOTE is conducted before undersampling. This means that the denominator in the $R$ ratio for SMOTE with Random Undersampling is the ratio of the size of the sampled majority instances to the size of the original and synthetic minority instances. For SMOTE with

Tomek Links and SMOTE with KNN undersampling, the synthetic minority instances are included in the KNN calculations.

| **Method** | **Parameters** | **AUPRC** |
|---|---|---|
| SMOTE with Random Undersampling | $R = 5, N = 10$ | 0.8259 |
| | $R = 6, N = 10$ | **0.8279** |
| | $R = 7, N = 7$ | 0.8228 |
| | $R = 8, N = 8$ | 0.8181 |
| SMOTE with Tomek Links | $N = 6$ | **0.8202** |
| SMOTE with KNN | $K = 50, N = 7$ | 0.8254 |
| | $K = 100, N = 9$ | **0.8255** |
| | $K = 150, N = 10$ | 0.8190 |
| | $K = 200, N = 7$ | 0.8183 |

Table 2: SMOTE with Undersampling (Optimal $N$)

Every model in Table 2 outperforms its equivalent in Table 1. This is unsurprising for SMOTE with Random Undersampling where the effect is the strongest, as the SMOTE alleviates the previous issue of a small training dataset. SMOTE with Tomek Links and SMOTE with KNN had smaller increases, demonstrating that the combination of techniques indeed provides an improvement. However, this effect is much smaller, likely because they do not benefit from a much larger training dataset than before.

| **Method** | **Parameters** | **AUPRC** |
|---|---|---|
| Baseline | $\gamma = 0.1, \alpha = 0.75$ | 0.8116 |
| | $\gamma = 0.2, \alpha = 0.25$ | **0.8199** |
| | $\gamma = 0.5, \alpha = 0.25$ | 0.8194 |
| | $\gamma = 1.0, \alpha = 0.25$ | 0.8197 |
| | $\gamma = 2.0, \alpha = 0.5$ | 0.8070 |
| | $\gamma = 5.0, \alpha = 0.5$ | 0.7908 |

Table 3: Focal Loss Tuning (Optimal $\alpha$)

We now move to experimenting with Focal Loss. Using the baseline model (no sampling methods), we first tuned the optimal hyperparameter values for

$\gamma$ and $\alpha$. For each $\gamma$, the optimal $\alpha$ was tuned and performance is displayed in Table 3. Unlike in [3] where the authors found $\gamma = 2.0, \alpha = 0.25$ to be best, for this dataset we find that $\gamma = 0.2, \alpha = 0.25$ is best, with the smaller $\gamma$ values generally performing better. This and the poor performance of the cross-entropy based Random Undersampling models suggests that predicting majority instances is not entirely trivial and that we should not discount the model's attention to the majority instance too much (either through undersampling or increasing $\gamma$). For the optimal parameters, the Focal Loss model has a sizable improvement over the baseline model trained with cross-entropy loss. For our remaining analysis, we will use $\gamma = 0.2, \alpha = 0.25$.

| **Method** | **Parameters** | **AUPRC** |
|---|---|---|
| Random Undersampling (Focal Loss) | $R = 1$ | 0.6079 |
| | $R = 2$ | 0.6902 |
| | $R = 3$ | 0.6522 |
| | $R = 4$ | 0.7062 |
| | $R = 5$ | 0.7108 |
| | $R = 6$ | 0.7115 |
| | $R = 7$ | 0.7133 |
| | $R = 8$ | **0.7172** |
| Tomek Links (Focal Loss) | – | **0.8092** |
| KNN (Focal Loss) | $K = 50$ | 0.8062 |
| | $K = 100$ | **0.8109** |
| | $K = 150$ | 0.8070 |
| | $K = 200$ | 0.8006 |
| SMOTE (Focal Loss) | $N = 2$ | 0.7980 |
| | $N = 3$ | 0.8054 |
| | $N = 4$ | 0.7961 |
| | $N = 5$ | 0.8115 |
| | $N = 6$ | 0.8097 |
| | $N = 7$ | 0.8026 |
| | $N = 8$ | 0.8048 |
| | $N = 9$ | **0.8127** |
| | $N = 10$ | 0.7920 |

Table 4: Over/undersampling with Focal Loss

Compared to the baseline, all of these models utilizing over/undersampling methods preform worse. While counterintuitive, this is what we expect, as the tuning of focal loss should completely resolve the class imbalance problem. Thus, any additional undersampling removes valuable signals from the dataset while oversampling introduces potential model overfitting. As focal loss was originally introduced as an alternative to sampling methods [3], it is not common practice to use them together. Our results further support this guideline.

This project does have some limitations. We were only able to extract a moderate performance gain using unbalanced learning methods, likely due to the extreme imbalance of the dataset. For less extreme imbalances, the undersampling methods would likely be more viable and could likely further model performance. More robust model tuning could also improve both over/undersampling and focal loss performance.

# References


[1] Elhassan At, Aljourf M, Al-Mohanna F, and Shoukri M. Classification of imbalance data using tomek link(t-link) combined with random undersampling (rus) as a data reduction method. *Global Journal of Technology and Optimization*, 1:1–11, 2016.

[2] K. Boyd, K.H. Eng, and C.D. Page. Area under the precision-recall curve: Point estimates and confidence intervals. In H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, editors, *Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2013*, volume 8190 of *Lecture Notes in Computer Science*, pages 451–466. Springer, Berlin, Heidelberg, 2013.

[3] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, January 2002.

[4] W. Chen, K. Yang, Z. Yu, et al. A survey on imbalanced learning: latest research, applications and future directions. *Artificial Intelligence Review*, 57:137, 2024.

[5] Kaggle. Credit card fraud detection dataset. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/code, 2024. Accessed: 2024-11-20.

[6] M. Kim and K.B. Hwang. An empirical evaluation of sampling methods for the classification of imbalanced data. *PLoS One*, 17(7):e0271260, 2022.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, February 2020.

[8] J.P. Zhang and I. Mani. Knn approach to unbalanced data distributions: A case study involving information extraction. In *Proceedings of the International Conference on Machine Learning (ICML 2003), Workshop on Learning from Imbalanced Data Sets*, Washington DC, August 21 2003.