Evaluating the influences of various class balancing methods on imbalanced business data

Biratal Wagle
Quantitative Biomedical Sciences
Dartmouth College
Hanover, NH
biratal.r.wagle.gr@dartmouth.edu

Kevin Yang
Quantitative Biomedical Sciences
Dartmouth College
Hanover, NH
zhihan.yang.gr@dartmouth.edu

1 Introduction

The rapid proliferation of machine learning (ML) applications across diverse domains has highlighted the critical need for robust predictive models that perform effectively even under challenging data conditions. One of the most pervasive challenges in real-world data analysis is the issue of imbalanced datasets, where the distribution of classes is highly skewed. This imbalance often leads to biased model predictions, where the minority class, despite its potential importance, is frequently overlooked in favor of the majority class. The Bank Marketing dataset, available from the UCI Machine Learning Repository, provides an ideal testbed for exploring these challenges. This study utilizes bank telemarketing data, comprising 52,944 phone contacts from a Portuguese retail bank collected between May 2008 and June 2013. The dataset is imbalanced, with a 12.38% success rate in selling long-term deposits. It includes 150 attributes, blending telemarketing, product, client, and socio-economic factors for comprehensive analysis.The inherent imbalance within the target classes makes this dataset particularly suitable for examining the effectiveness of various optimization techniques tailored to imbalanced data scenarios. Our objective is to evaluate how different techniques work on neural network based model and non-neuron network based model.

2 Data Preprocessing

A data preprocessing pipeline was designed to enusre that the raw bank marketing dataset was transformed into a format suitable for model training, while perserving the integrity and representativeness of the original data. The preprocessing procedure comprised the following steps:

2.1 Data Loading and Target Encoding

The features and target labels were extracted from the bank marketing dataset. The target variable, originally provided in a multidimensional format, was flattened to a one-dimensional array. Subsequently, a label encoder was applied to transform categorical target values (e.g., “yes”/“no”) into numerical indices, thereby facilitating their use in subsequent predictive modeling.

2.2 Identification of Feature Types:

The dataset was partitioned into categorical and numerical features by examining the data types. Columns with data types indicative of objects or categorical variables were classified as categorical, whereas those with numeric types (e.g., integers or floats) were designated as numerical. This distinction enabled the application of tailored preprocessing techniques for each group.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

2.3 Preprocessing Pipelines for Feature Transformation:

Two distinct pipelines were constructed using scikit-learn's Pipeline and ColumnTransformer:
• Numerical Pipeline: Missing values in numerical features were imputed using the mean value of each feature. Thereafter, a standardization procedure was applied to scale the features to have zero mean and unit variance. This standardization is critical for ensuring that the model training is not unduly influenced by features with larger numeric ranges.
• Categorical Pipeline: For categorical features, missing values were imputed using the most frequent value in each column. Subsequently, one-hot encoding was employed to transform categorical variables into binary indicator variables.

3 Methods

Three different machine learning algorithms were utilized in this study, including feedforward neural network model, LSTM model and Random Forest classifier to assess their performance on imbalanced classification tasks.

3.1 Transformation and Data Splits

The integrated preprocessing pipeline was then fitted to the entire feature set and used to transform the raw data into a numerical matrix, denoted as $X_{\text{preprocessed}}$. To maintain the validity of subsequent model evaluations, the preprocessed data was partitioned using stratified sampling. Initially, the data was divided into a training/validation set (comprising 80% of the data) and a test set (20%). The training/validation subset was further segmented into a training set (70% of the total data) and a validation set (10% of the total data). This hierarchical splitting strategy ensured that the test set remained entirely unseen during model development and hyperparameter tuning.

3.2 SMOTE

SMOTE [1]is an advanced oversampling method designed to address the challenges faced by imbalanced datasets. SMOTE generates synthetic samples by interpolating between existing minority examples and their nearest neighbor in the feature space. This process creates new, plausible instances that enrich the minority class distribution, thereby migating the risk of overfitting often associated with simple replication method. By enhancing the representation of the minority class, SMOTE helps in training more robust and generalizable models.

Algorithm 1 SMOTE Algorithm

Input: Minority class samples S, number of nearest neighbors k, oversampling factor N.
Output: Synthetic samples set S_{syn}.
for each sample $x_i \in S$ **do**
  Find the k nearest neighbors of x_i in S.
  for $n = 1$ to N **do**
    Randomly select a neighbor x_{nn} from the k nearest neighbors.
    Compute the difference vector: diff $= x_{\text{nn}} - x_i$.
    Generate a random scalar $\lambda \sim U(0, 1)$.
    Create synthetic sample: $x_{\text{new}} = x_i + \lambda \times \text{diff}$.
    Add x_{new} to S_{syn}.
  end for
end for
return S_{syn}.

[1]Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority oversampling technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953

3.3 Weighting Mechanism

In imbalanced classification problems, another common strategy to counteract the bias toward the majority class is to re-weight the loss function. In the implementation, the weight assigned to each class is determined as follows:

$$weight_i = \frac{N}{2 \times count_i}$$

Where N is the total number of samples and $count_i$ denotes the number of samples belonging to class i. This formulation is designed to allocate an equal total weight of $\frac{N}{2}$ to each class, regardless of its frequency in the data. In a binary classification setting, if class 0 represent the majority and class 1 represent the minority, this method inherently assigns a higher weight to class 1. For example, consider a dataset with N = 9043 samples, where class 0 has 7985 samples and class 1 has 1058 samples. The corresponding weight would be

$$weight_0 = \frac{9043}{2 \times 7985} = 0.566$$

$$weight_1 = \frac{9043}{2 \times 1058} = 4.277$$

Thus the loss contribution for a misclassified sample from class 1 is approximately 7.5 times higher than that for class 0. When incorporating these weights into the loss function, the overall loss is modified to

$$Loss = \frac{1}{N} \Sigma_{i=1}^{N} weight_{y_1} * L(p(y_i|x_i))$$

Where $L(p(y_i|x_i)$ represent the standard loss (cross entropy) for the ith sample. This weighted loss function ensures that the minority class exerts an influence on the gradient updates comparable to that of the majority class, thereby encourageing the model to allocate more capacity to correctly classifying underrepresented instance.

3.4 Neural Network

A feedforward neural network was selected as the baseline model due to its capacity to approximate complex, non-linear relationships between input features and target variable. This network consisted of three hidden layers, each containing 128 and sigmoid activation function as illustrated in Figure 1

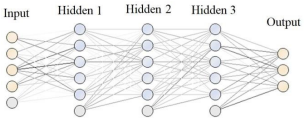

Figure 1: Feedforward Neural Network

However, the initial results revealed that the baseline neural network exhibited substantial bias toward the majority class ("no") , demonstrating poor predictive performance on the minority class ("yes"). Specially, the model achieved high precision but low recall for the minority class Table. The confusion matrix further highlights this issue, indicating the need for specialized method to address class imbalance.

Label	Precision	Recall	F1-Score	Support
no	0.90	0.98	0.94	7,985
yes	0.59	0.21	0.31	1,058
Macro Avg	0.75	0.60	0.63	9,043
Weighted Avg	0.87	0.89	0.87	9,043

Table 1: Classification Report

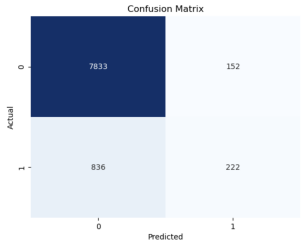

Figure 2: Confusion Matrix for Baseline model

One common approach for addressing imbalanced class distributions involves employing sampling techniques, while another popular strategy is to use weighting mechanisms that penalize the misclassifications of the minority class.

After employing SMOTE, the model performance significantly improved, especially for the minority class. THe recall for class "yes" notably increased from 0.21 to 0.83. The SMOTE techniques clearly enhanced the model's ability to classify minority class instance effectively.

Label	Precision	Recall	F1-Score	Support
0	0.97	0.82	0.89	7,985
1	0.38	0.83	0.52	1,058
Macro Avg	0.68	0.82	0.71	9,043
Weighted Avg	0.90	0.82	0.85	9,043

Table 2: Classification Report with SMOTE

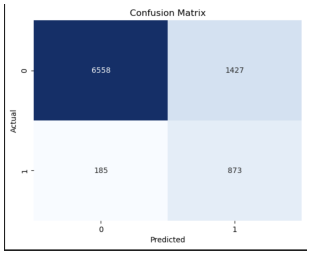

Figure 3: Confusion Matrix with SMOTE

Label	Precision	Recall	F1-Score	Support
no	0.92	0.97	0.94	7,985
yes	0.64	0.38	0.48	1,058
Macro Avg	0.75	0.60	0.63	9,043
Weighted Avg	0.87	0.89	0.87	9,043

Table 3: Classification Report with Weights

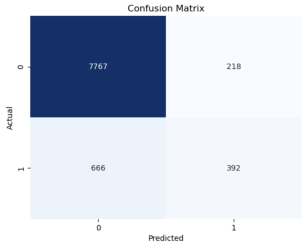

Figure 4: Confusion Matrix with Weights

Table 3 presents the classification performance obtained using the weighted loss mechanism. Although overall accuracy remains high the introduction of class weights has led to an improvement in recall for the minority class increasing its detection rate as evidenced by corresponding increase in the F1-score. The model was trained for 500 epochs, and even though the improvement in recall was modest, it contributed to a more balance performance between precision and recall, ultimately enhancing the model's capability to indentify positive instances.

3.5 Random Forest

Random Forest was chosen as the baseline model due to its inherent robustness in handling imbalanced datasets. Unlike single decision trees, which tend to be highly sensitive to class distribution, Random Forest leverages an ensemble of multiple trees trained on different subsets of data, reducing the risk of overfitting to the majority class. This bagging-based approach helps in capturing diverse decision boundaries, which is particularly useful when learning from an imbalanced dataset where the minority class is underrepresented.

The parameter grid used in our setup was designed to establish a strong baseline while maintaining flexibility. With 100 estimators, the model effectively aggregates multiple decision boundaries, reducing the likelihood of the majority class dominating predictions. By setting `max_depth=None`, each tree was allowed to grow fully, ensuring that even complex patterns in the minority class were captured. The `min_samples_split=2` allowed the model to create splits as soon as any variance was detected, enabling it to identify subtle differences in class distributions.

Label	Precision	Recall	F1-Score	Support
no	0.91	0.98	0.95	7952
yes	0.72	0.32	0.44	1091
Macro Avg	0.82	0.65	0.69	9043
Weighted Avg	0.89	0.9	0.89	9043

Table 4: Classification Report with Weights for RF

Label	Precision	Recall	F1-Score	Support
no	0.94	0.95	0.94	7952
yes	0.58	0.55	0.57	1091
Macro Avg	0.76	0.75	0.75	9043
Weighted Avg	0.90	0.90	0.90	9043

Table 5: Classification Report with Weights + SMOTE for RF

From our results, we see that Random Forest works better with just weights than it does with SMOTE. While SMOTE helps balance the dataset by creating synthetic examples of the minority class, it doesn't actually change how the model learns or how it treats different classes. Random Forest with class weighting, on the other hand, directly adjusts the model's focus by making misclassifications of the minority class more costly. This means the model naturally pays more attention to underrepresented instances without relying on artificially generated data. Using only SMOTE can sometimes introduce unrealistic or repetitive patterns, which might not generalize well to new data. Class weighting, however, works within the model itself, improving its ability to recognize and correctly classify minority class instances in a way that's more reliable and adaptable. We can see that reflected in our results

3.6 LSTM

A Long Short-Term Memory (LSTM) network was chosen as a model due to its ability to capture long-term dependencies in sequential data. Unlike traditional feedforward networks or simple recurrent neural networks (RNNs), LSTMs incorporate memory cells that regulate the flow of information using gating mechanisms. This structure allows LSTMs to retain relevant patterns over extended time steps while mitigating the vanishing gradient problem, making them particularly well-suited for tasks involving temporal dependencies.

Label	Precision	Recall	F1-Score	Support
no	0.93	0.97	0.95	7985
yes	0.64	0.46	0.53	1058
Macro Avg	0.78	0.71	0.74	9043
Weighted Avg	0.90	0.91	0.90	9043

Table 6: Classification Report for LSTM

Label	Precision	Recall	F1-Score	Support
no	0.98	0.84	0.91	7985
yes	0.42	0.85	0.56	1058
Macro Avg	0.70	0.85	0.73	9043
Weighted Avg	0.91	0.85	0.87	9043

Table 7: Classification Report LSTM with Weights

Label	Precision	Recall	F1-Score	Support
no	0.97	0.85	0.91	7985
yes	0.43	0.83	0.56	1091
Macro Avg	0.70	0.84	0.74	9043
Weighted Avg	0.91	0.85	0.87	9043

Table 8: Classification Report LSTM with Weights + SMOTE

From the table and the data, we can see that Introducing class weighting significantly shifts the model's behavior, improving recall for the minority class from 0.46 to 0.85. This indicates that the model is now better at identifying "yes" instances. However, this improvement comes at the cost of reduced precision for the minority class ($0.64 \rightarrow 0.42$), meaning the model is now generating more false positives. The F1-score for the minority class increases slightly ($0.53 \rightarrow 0.56$), and the macro average recall also improves ($0.71 \rightarrow 0.85$), showing that class weighting successfully rebalances the learning process.

Combining class weighting with SMOTE leads to further refinement. The recall for the minority class remains high (0.83), and its precision increases slightly ($0.42 \rightarrow 0.43$) compared to using class weights alone. The F1-score for the minority class remains steady at 0.56. Notably, the precision for the majority class slightly decreases ($0.98 \rightarrow 0.97 \rightarrow 0.97$), but this is a reasonable trade-off for achieving better minority class recognition. The overall macro F1-score improves from 0.73 to 0.74, suggesting a slight gain in balanced performance.

Overall, we can see that using both SMOTE and class weighting improves the performance of the LSTM.

4 Conclusion

From our results, we see that in most of our methods, a combination of using the class weights and and SMOTE improves the recall of the "Yes" Label. This implies that the class imbalance is succesfully being considered in our approaches. All the code for this project can be found in this GitHub Repository. For future directions, we would like to apply these methods to different datasets with different distributions of class imbalances. This would allow us to evaluate how generalizable these methods are.

5 Reference

Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P. (2002). Smote: Synthetic minority over- sampling technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953

6 Appendix

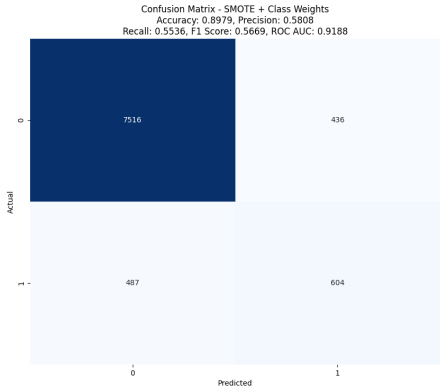

Figure 5: Confusion Matrix for LSTM with SMOTE + Weights

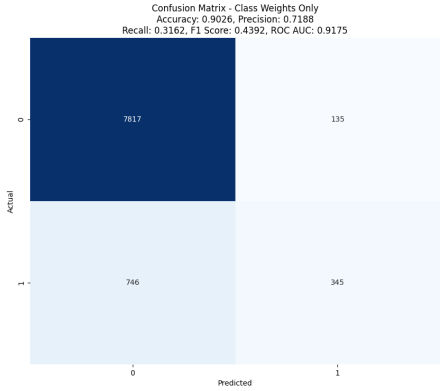

Figure 6: Confusion Matrix For RF with Weights