ENHANCING CLASSIFICATION PERFORMANCE ON IMBALANCED DATASETS: A COMPARATIVE ANALYSIS OF OVERSAMPLING TECHNIQUES

Contents

Project Overview

This project aims to address the challenge of class imbalance in machine learning classification tasks by evaluating the effectiveness of various oversampling techniques. The study focuses on the impact of ADASYN, KMeansSMOTE, and a Deep Learning Generator on classification performance across three real-world datasets. The following three datasets are selected for this project:

- **Title:** Hotel Reservation Classification Dataset
  Domain: Hospitality
  Number of Rows: 36275
  Number of Columns: 19
  Feature Type: Mixed
  Class Balance: 67:33 (Percentage)
  Source: https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset

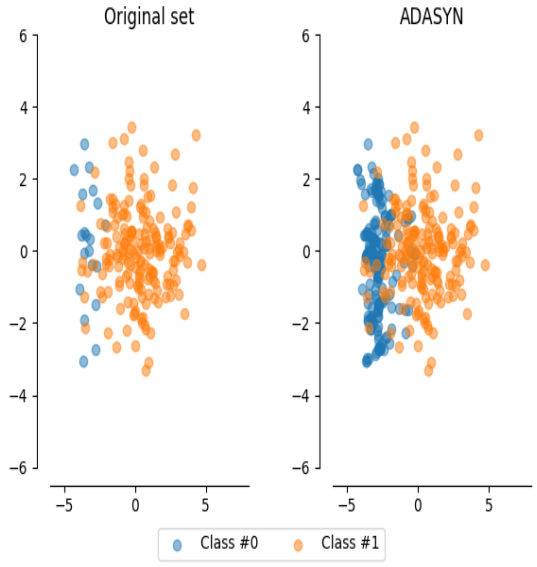
- **Title:** Churn Modelling
  Domain: Finance
  Number of Rows: 10000
  Number of Columns: 14
  Feature Type: Mixed
  Class Balance: 79:21 (Percentage)
  Source: https://www.kaggle.com/datasets/shrutimechlearn/churn-modelling

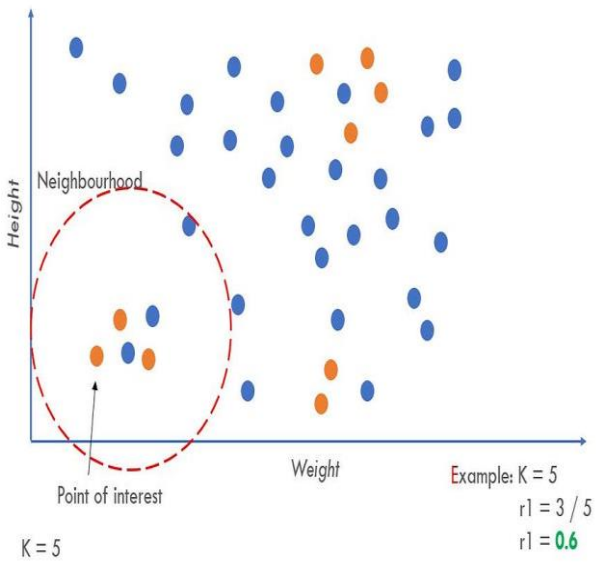
- **Title:** Lending Club Loan Data Analysis
  Domain: Finance
  Number of Rows: 9578
  Number of Columns: 14
  Feature Type: Mixed
  Class Balance: 84:16 (Percentage)
  Source: https://www.kaggle.com/datasets/urstrulyvikas/lending-club-loan-data-analysis/data

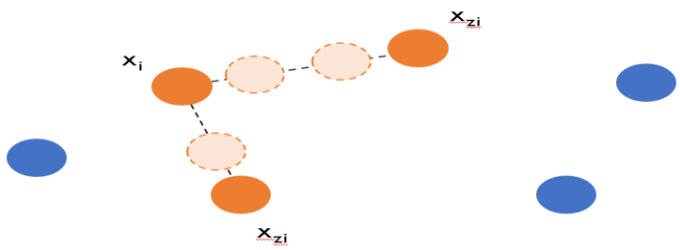
Class Imbalance Techniques Employed

ADASYN

ADASYN (Adaptive Synthetic Sampling) is an oversampling technique used to address class imbalance in datasets. It generates synthetic samples for the minority class, focusing on regions where the minority class is underrepresented. ADASYN identifies minority class instances in the dataset. It then computes the density of each minority class instance. Density is the imbalance ratio, indicating the representation of minority class instances among its k nearest neighbors. From the nearest neighbors, it randomly selects one neighbor to create synthetic sample. For the minority instance and randomly selected neighbor, it calculates a weighted average of the two for each feature and this gives a new synthetic data point and it then assigns it minority label and continues to do so until class imbalance is treated. ADASYN prioritizes regions with low imbalance ratios, meaning it creates more synthetic samples in the regions with less minority labelled neighbors.







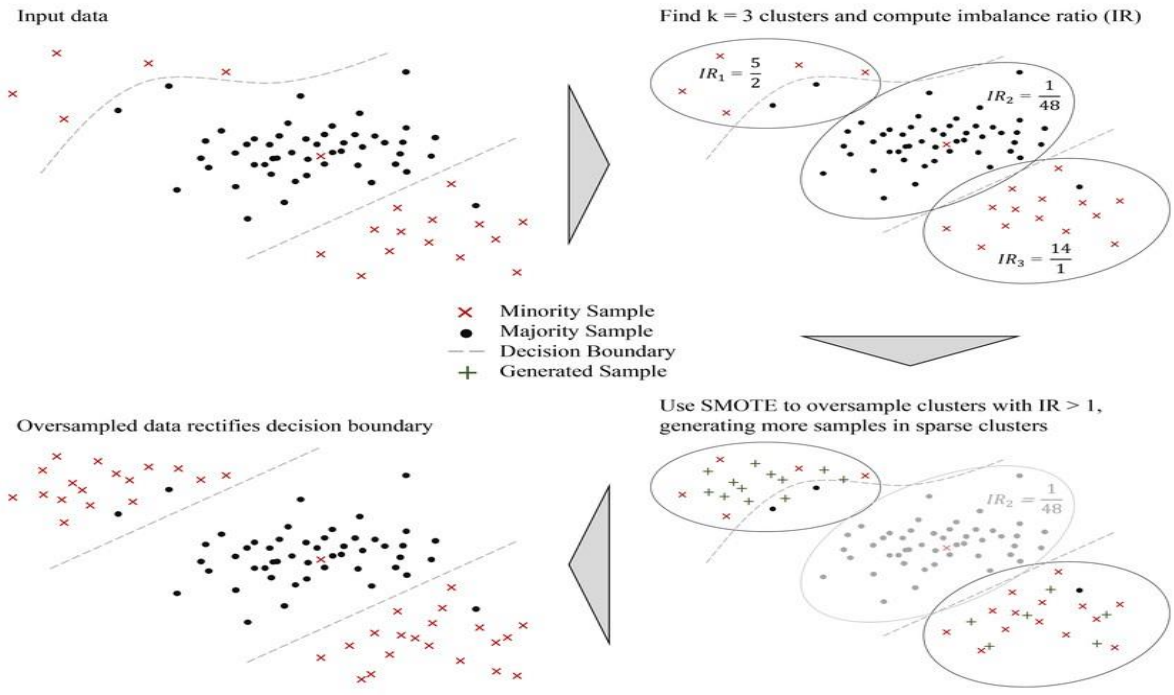
Why ADASYN?

In our datasets we got class imbalance of 84:16 at max so we could've gone with both ADASYN & SMOTE in this case. However, ADASYN is still the preferred choice. But because when visualized (through PCA) we found that for each dataset we're unable to identify distinct simple linear decision boundaries, ADASYN turns out to be a better choice than SMOTE.

Moreover, since our datasets are relatively small in size there's no practical difference between SMOTE & ADASYN in terms of computation and time cost (otherwise for large datasets SMOTE is better and major reason why one should pick SMOTE over ADASYN otherwise ADASYN is a better choice overall)

KMeansSMOTE

KMeansSMOTE is an extension over SMOTE oversampling technique where first it identifies clusters within the data using K-Means Clustering and then generates synthetic samples for minority instances within a cluster using SMOTE. Initially, k cluster centers (centroids) are selected and based on distance, instances closer to centroids are assigned to their respective clusters. Centroids are recalculated (mean is taken) and all minority instances are reassigned a cluster again based on distance and this continues until centroids don't change much and that's how we get our clusters. Then similar to ADASYN it generates synthetic samples using the neighbors of each minority instance but this time neighbors are restricted to clusters and unlike ADASYN number of synthetic samples to generate for each minority instance is not based on instances' density.



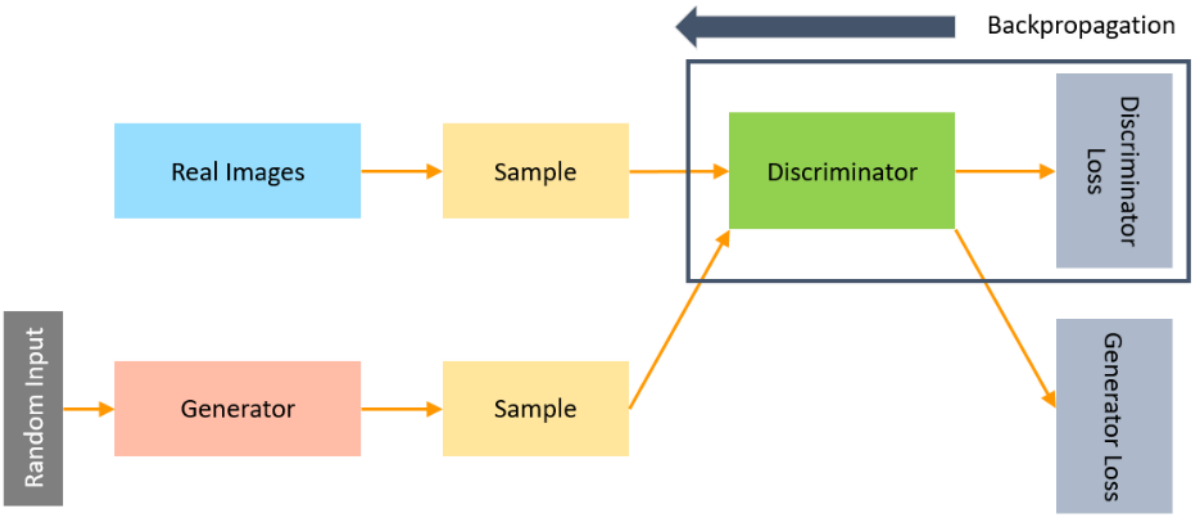
Why KMeansSMOTE?

When visualized (through PCA) we found that for each dataset we've complex non-linear decision boundaries (low correlation as well) so clustering-based oversampling methods like KMeansSMOTE is particularly beneficial as it captures the local structure of the data more effectively but only generating within clusters, leading to improved model generalization and performance. Sampling generation through this method is highly interpretable and also computationally efficient and scalable to large datasets. This scalability makes it suitable for datasets with a large number of rows and columns, such as ours with more than 36000 rows and about 19 columns.

Deep Learning Generator (Mostly AI)

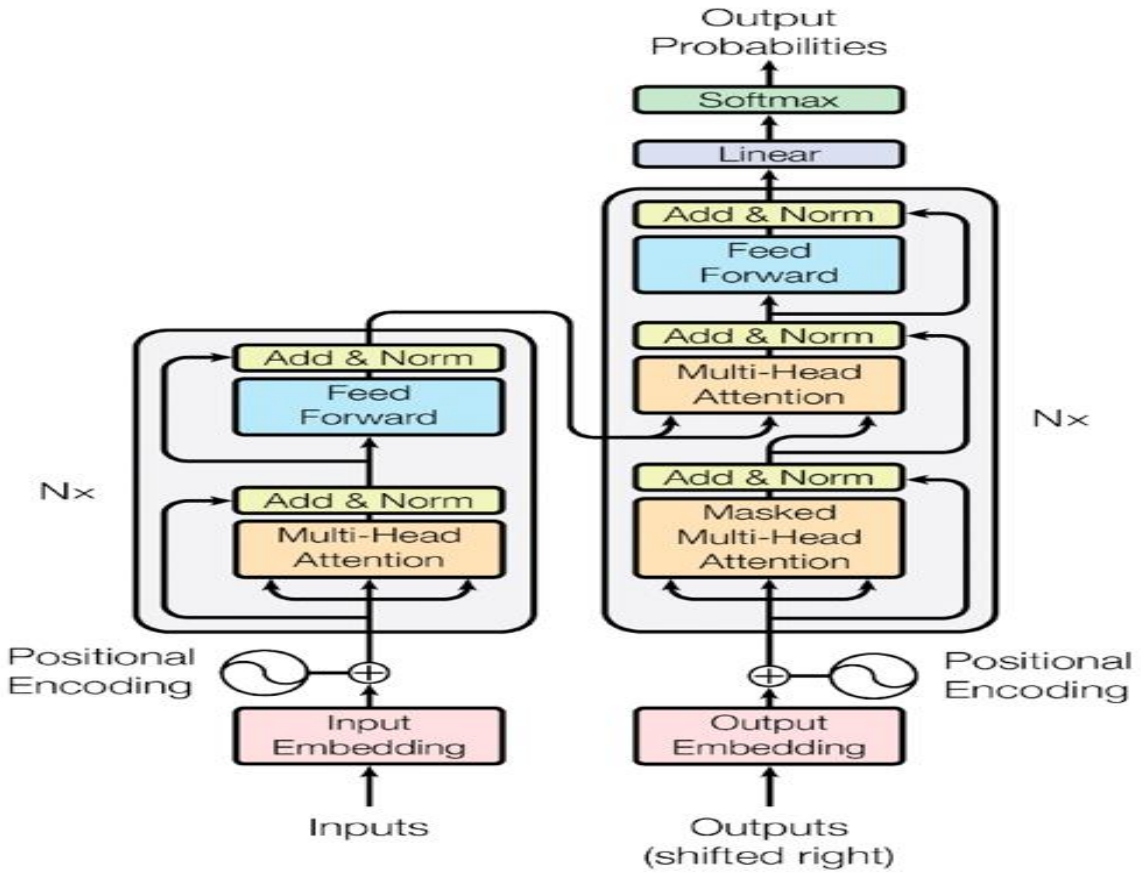
It is an online generator which uses a number of Deep Learning model architectures and approaches that have emerged to create synthetic data, including Transformers, GANs, Variational Autoencoders as well as Autoregressive Networks. It has and continue to actively research all of them, and have developed its own unique combination of techniques to provide the best possible results in terms of accuracy, privacy as well as flexibility.

- **GANs**
  GANs consists of two neural networks. There is a generator, G(x), and a discriminator, D(x). Both of them engage in a competing game. The generator's goal is to trick the discriminator by providing data that is similar to that of the training set. The discriminator will attempt to distinguish between fake and real data. Both work together to learn and train complicated data such as audio, video, and image files. The Generator network takes a sample and creates a simulated sample of data. The Generator is trained to improve the Discriminator network's likelihood of making errors.



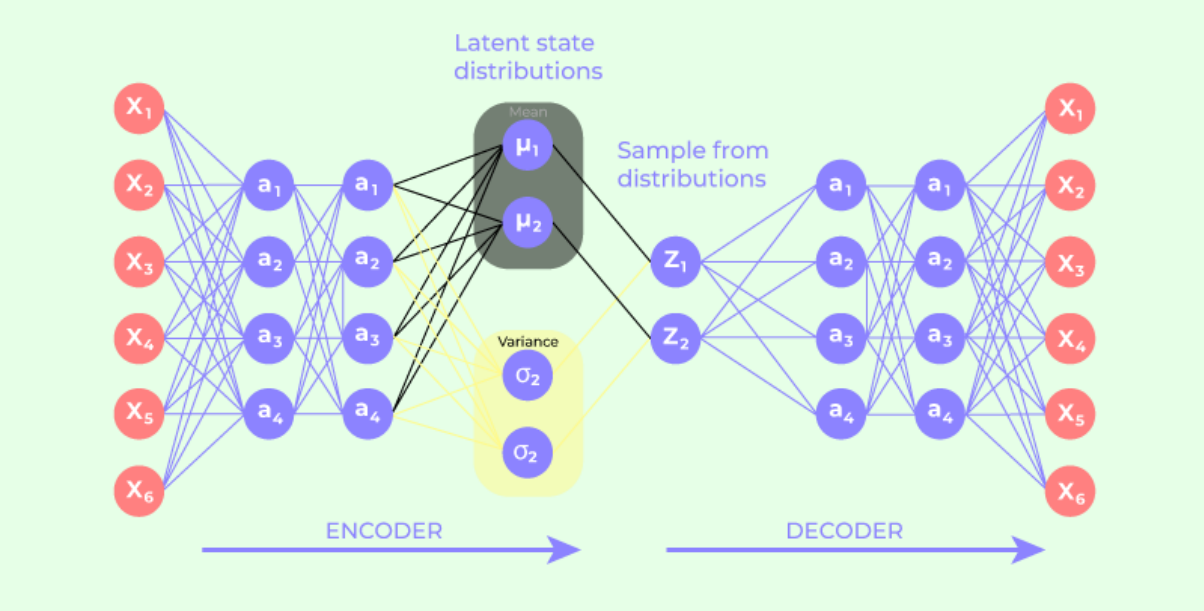
- **Transformers**

  Transformers are neural network architectures that convert or modify input sequences into output sequences. They accomplish this by acquiring context and tracking the relationships between sequence components. Consider the following input sequence: "What is the color of the sky?" The transformer model employs an internal mathematical representation to determine the relevance and relationship among the words color, sky, and blue. It uses that knowledge to produce the following output: "The sky is blue."

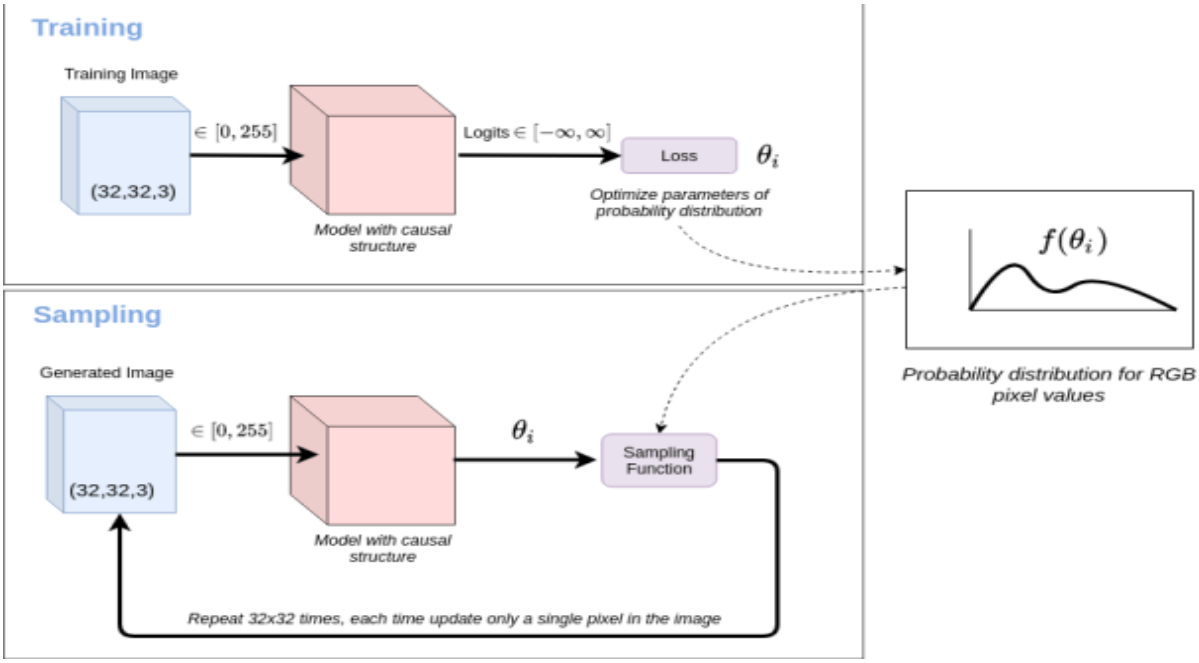


- **Variational Autoencoders**

  A variational autoencoder (VAE) uses probability to describe a latent space observation. Thus, instead of creating an encoder that outputs a single value to describe each latent state characteristic, It will create an encoder that describes a probability distribution for each latent property. It is used in a variety of applications, including data compression and synthetic data synthesis. Variational autoencoders differ from autoencoders in that they give a statistical method for representing the dataset's samples in latent space. As a result, the variational autoencoder generates a probability distribution in the bottleneck layer rather than a single output value.



- **Autoregressive Models**
  Autoregressive models are a class of machine learning (ML) models that automatically predict the next component in a sequence by taking measurements from previous inputs in the sequence.



Why Deep Learning Generator?

The AI tool leverages a combination of cutting-edge techniques, including Generative Adversarial Networks (GANs), Transformers, Variational Autoencoders, and Autoregressive models. These state-of-the-art methods enable the generation of synthetic data that closely

resembles the distribution of the original datasets. By employing these advanced techniques, we can create high-quality synthetic samples that effectively represent the minority class, thereby addressing the imbalance issue.

Results & Discussion

It has been observed for each CI solution, a drop in testing accuracy, which is to be expected in comparison to baseline as previously algorithms were biased and majorly predicting only major class and test set had mostly major class values so testing accuracy was going to be good, even if we don't train a model and simply predict major class to all data points. However, it was not a good model as we can see from minority class (1) precision, recall, and f1-score and AUC score. Detailed result description on each dataset can be found below.

Dataset 1: Lending Club Loan Dataset

Baseline Results:

1. The baseline models demonstrate significant class imbalance issues, especially in terms of precision, recall, and F1-score for the minority class (class 1).

2. Precision for class 1 is low across all models, indicating a high number of false positives in predicting the minority class.

3. Recall for class 1 is also low, indicating that the models are missing a large number of positive instances.

4. AUC scores are relatively low, indicating poor discriminative ability between the two classes.

CI Solution #1: ADASYN

1. ADASYN has improved the performance metrics for class 1 compared to the baseline.

2. Precision, recall, and F1-score for class 1 have increased, indicating better detection of positive instances.

3. However, there's still room for improvement as precision and recall for class 1 are relatively low.

4. AUC score is also slightly improved, indicating better overall discriminative ability, especially for KNN model.

CI Solution #2: KMeansSMOTE

1. KMeansSMOTE has also shown improvements compared to the baseline, especially in terms of F-1 score and recall for class 1.

2. Precision for class 1 has increased, indicating fewer false positives.

3. Recall for class 1 has also increased, indicating better detection of positive instances.

4. AUC score has also slightly improved compared to the baseline for most models but in KNN it's significant.

CI Solution #3: Deep Learning Generator

1. This solution has shown mixed results compared to the other approaches.

While recall for class 1 has improved in a similar drastic manner as shown in previous approach, precision has shown minuscule improvement and even decreased in Naive Bayes, indicating a better detection of positive instances and similar number of false negatives to baseline.

2. Overall, the performance metrics for class 1 are slightly better than the baseline but not as good as ADASYN and KMeansSMOTE

3. AUC score is comparable to ADASYN and KMeansSMOTE but slightly lower.

Dataset 2: Hotel Reservation Classification Dataset

Baseline Results:

Baseline models generally exhibit decent performance.

1. Precision (class 0): The precision for the majority class is relatively high across all models, indicating a low false positive rate.

2. Precision (class 1): Precision for the minority class varies but generally lower, suggesting a higher false positive rate for predicting the minority class.

3. Recall (class 0): Recall for the majority class is high, indicating a low false negative rate.

4. Recall (class 1): Recall for the minority class is generally lower, indicating a higher false negative rate.

5. Recall for class 0 is consistently higher than for class 1 across all models.

6. F1-scores for class 0 are notably higher compared to class 1, indicating better overall performance for the majority class.

CI Solution #1: ADASYN:

1. Precision for class 1 has generally decreased compared to the baseline for most models. This indicates that while ADASYN may have improved the model's ability to capture more instances of the minority class, it has also increased the number of false positives, which negatively impacts precision.

2. Precision (class 0): Precision for the previous majority class is high, indicating a low false positive rate.

3. Recall for class 1 has generally improved, suggesting better identification of positive cases after applying ADASYN.

4. F1-scores for class 1 have shown improvement in some models but might have dropped slightly in others.

5. Testing Accuracy: Overall accuracy has slightly decreased compared to the baseline, which is ok as in previously imbalance data set due to high bias towards majority class testing accuracy was higher

6. AUC: AUC values show improvement compared to the baseline for some models but not all. Specifically, the AUC has increased in two out of the five models (KNN and Decision Tree) after applying ADASYN. This suggests that ADASYN has effectively improved the discriminative ability of these models, leading to better overall performance in terms of AUC.

CI Solution #2: KMeansSMOTE:

1. Precision (class 0): Precision for the class 0 is high.

2. Precision (class 1): Precision for the class 1 is higher compared to ADASYN across all model besides naive bayes but precision for class 1 has also generally decreased compared to the baseline for most models after applying KMeansSMOTE. Similar to ADASYN, this suggests that while KMeansSMOTE may have effectively balanced the class distribution, it has also led to more false positive predictions for the minority class.

3. Recall for class 1 has generally improved, suggesting better identification of positive cases after applying KMeansSMOTE.

5. F1-scores for class 1 have shown improvement in some models but might have dropped slightly in others.

6. Testing Accuracy: Overall accuracy is comparable to ADASYN.

7. AUC: AUC values also show improvement compared to the baseline for some models but not all. Similar to ADASYN, the AUC has increased in two out of the five models (KNN and Decision Tree) after applying KMeansSMOTE. This indicates that KMeansSMOTE has effectively enhanced the discriminative ability of these models, resulting in better overall performance in terms of AUC.

CI Solution #3: Deep Learning Generator:

1. Precision (class 0): Precision for the majority class is high.

2. Precision for class 1 has also generally decreased compared to the baseline for most models after applying Deep Learning Generator. Similar to ADASYN & KMeansSMOTE this suggests that while Deep Learning Generator may have effectively balanced the class distribution, it has also led to more false positive predictions for the minority class.

3. Recall for class 1 has generally improved, suggesting better identification of positive cases after applying Deep Learning Generator.

4. F1-scores for class 1 have shown improvement in some models but might have dropped slightly in others.

5. Testing Accuracy: Overall accuracy is slightly lower compared to all Baseline, ADASYN and KMeansSMOTE.

6. AUC: Across the models, the AUC values have mostly decreased compared to the baseline. Additionally, in two models, the AUC values are almost equal to the baseline but still less. This indicates that the Deep Learning Generator's performance in terms of AUC is generally lower or comparable to the baseline across the various models tested.

Dataset 3: Churn Modelling Dataset

Baseline Results:

1. The baseline models demonstrate significant class imbalance issues, especially in terms of recall, and F1-score for the minority class (class 1). Precision for the minority class 1 varies but generally lower than majority 0 class, suggesting a higher false positive rate for predicting the minority class.

2. Recall for class 1 is low across all models, indicating that the models are missing a large number of positive instances.

3. F1-scores for class 0 are notably higher compared to class 1, indicating better overall performance for the majority class

4. Testing accuracy is decent which is to be expected as due to imbalance data almost 80% is majority class.

5. AUC scores are relatively good, indicating good discriminative ability between the two classes.

CI Solution #1: ADASYN:

1. Precision for class 1 has marginally decreased compared to the baseline for all models. This indicates that while ADASYN may have improved the model's ability to capture more instances of the minority class, it has also increased the number of false positives, which negatively impacts precision.

2. Recall for class 1 has drastically improved, suggesting much better identification of positive cases after applying ADASYN.

3. F-1 score for class 1 has generally improved.

4. Testing accuracy has declined slightly as that understandable because of imbalance data and 80% majority class in baseline.

5. AUC across all model shows improvement indicating better distinction between two classes after ADAYSN

CI Solution #2: KMeansSMOTE:

1. Precision for class 1 has marginally decreased compared to the baseline for all models which is similar to ADASYN. This indicates that while KMeansSMOTE may have improved the model's ability to capture more instances of the minority class, it has also increased the number of false positives, which negatively impacts precision.

2. Recall for class 1 has drastically improved same as ADASYN, suggesting much better identification of positive cases after applying KMeansSMOTE.

3. F-1 score for class 1 has generally improved same as ADASYN.

4. Testing accuracy has declined slightly as that understandable because of imbalance data and 80% majority class in baseline.

5. AUC across all model shows improvement indicating better distinction between two classes after KMeansSMOTE similar to ADASYN.

CI Solution #3: Deep Learning Generator:

1. Precision for class 1 has marginally decreased compared to the baseline for all models which is similar to ADASYN & KMeansSMOTE. This indicates that while Deep Learning Generator may have improved the model's ability to capture more instances of the minority class, it has also increased the number of false positives, which negatively impacts precision.

2. Recall for class 1 has drastically improved same as ADASYN & KMeansSMOTE, suggesting much better identification of positive cases after applying Deep Learning Generator.

3. F-1 score for class 1 has generally improved same as ADASYN & KMeansSMOTE.

4. Testing accuracy has declined slightly as that understandable because of imbalance data and 80% majority class in baseline.

5. AUC across all model shows improvement indicating better distinction between two classes after Deep Learning Generator similar to ADASYN & KMeansSMOTE.

Conclusion

For all 3 datasets, across majority of the models the classification performance definitely improves by deploying the CI solutions with improvement in the minority class recall and f1-score being seen across majority of the algorithms and CI solutions. Overall, ADASYN gave the best results among all 3 datasets on majority of the algorithms and understandably so because when visualized (through PCA) we found that for each dataset we're unable to identify distinct simple linear decision boundaries and so ADASYN is a better choice than SMOTE and therefore KMeansSMOTE. ADASYN even outperforms the Deep Learning Generators and interestingly Deep Learning Generators performed the worst. So, ADASYN gave the best performance, then KMeansSMOTE, and then Deep Learning Generators and all 3 approaches gave an improvement over the baseline across majority of algorithms and minority class relevant metrics, therefore providing sufficient evidence to conclude that yes ML performance does get improve by using CI solutions.

But does the performance of a CI solution get impacted significantly by the choice of different algorithms? Comparing all performance metrics of different CI solutions across different models (KNN, Logistic Regression, Gaussian Naive Bayes, Linear SVM, & Decision Trees) we saw that mostly KNN resulted in the most performance improvement even though it wasn't necessarily the best performing model on the baseline and on some datasets after CI solutions KNN turned out to be the best model, not just most improved model. Further, on ADASYN this impact was prominent and then dropped on KMeansSMOTE and dropped again on Deep Learning Generators (with KNN no longer being the best or most improved model anymore). If you check AUC it has been consistently improving across majority datasets for KNN model of ADASYN and KMeansSMOTE. This was expected as both ADASYN and KMeansSMOTE use KNN itself to generate synthetic data and so doing so it might inherently generate patterns which are natural for KNN to learn. Moreover, Naïve Bayes showed the best results only on Deep Learning Generators consistently. However, all algorithms improved more or less (in one dataset Decision Tree even outperformed KNN on ADASYN). On some datasets and CI solutions, majority of algorithms other than KNN showed fall in AUC score consistently and even sometimes drop in f1-score is observed. Logistic regression and linear SVM consistently gave similarly results and improvements, and Decision Tree works best on baseline but doesn't improve much for any CI technique. Therefore, we can conclude that we might not have conclusive evidence on Deep Learning Generators but we can conclude that ADASYN and KMeansSMOTE get significantly impacted by the choice of algorithm and work best with KNN particularly but we cannot conclude that in general the performance of a CI solution gets impacted significantly by the choice of different algorithms.

Limitations and Future Work

Limitations

- **Deep Learning Generator Performance:**
  Limitation: The Deep Learning generator's poor performance could be attributed to its methodology. The data generated by the Deep Learning generator was entirely synthetic, created based on learning from the original dataset with class imbalance addressed. In contrast, both ADASYN and KMeansSMOTE added synthetic data to the original data, preserving the patterns in the original dataset. The efficacy of the Deep Learning generator in preserving these patterns depends on the accuracy of its learning process.
  Recommendation: A better approach for future studies would be to generate synthetic data using Deep Learning generators and combine it with the original data to balance the classes. Additionally, the impact of varying the volume of synthetic data (from low to high) could be assessed.

- **Data Leakage in Training and Testing Splits:**
  Limitation: In this project, the original test set was retained for accurate comparison between the baseline and class imbalance (CI) solution techniques. However, when applying CI solution techniques before splitting the data, some test data might leak into the training set, leading to misleading results.
  Recommendation: A better approach for future studies would be to apply CI solution techniques to the training and validation sets only. After addressing the class imbalance, the data should be split into training and testing sets. The model should then be trained on the new training set and tested on the original test set to avoid data leakage.

Future Work

- **Extreme Imbalance Ratios and Advanced Classification Algorithms:**
  Exploration: Assess the impact of CI techniques on datasets with more extreme imbalance ratios, such as a dataset of COVID-19 patients with an imbalance ratio of 99:1.
  Expansion: Increase the number of classification algorithms used, incorporating ensemble methods and deep learning algorithms.

- **Additional Oversampling and Undersampling Techniques:**
  Exploration: Test a wider variety of oversampling techniques beyond those used in this study. Additionally, explore undersampling techniques or a combination of both approaches, such as SMOTEENN and SMOTETomek.

- **Balanced Ensemble Methods and Batch Generators:**
  Alternative Approaches: Instead of modifying the dataset, explore balanced ensemble methods and balanced batch generators. These techniques could provide a different perspective on addressing class imbalance without altering the original data distribution.

By addressing these limitations and exploring the proposed future work, the project can contribute more robust and comprehensive insights into the effectiveness of various class imbalance solution techniques in machine learning.