INSTITUTE OF BUSINESS ADMINISTRATION

PROJECT REPORT – IMBALANCE DATASETS

Machine Learning I (CSE 602)

Submitted by:

HINA FARHEEN – 29402

UMMUL WARA - 29407

ABSTRACT

Imbalanced data is a common challenge in machine learning where the distribution of target classes is significantly uneven. In real-world scenarios, such as fraud detection, medical diagnosis, and anomaly detection, imbalanced data is ubiquitous. However, traditional machine learning algorithms often struggle to effectively learn from imbalanced datasets, leading to biased models that favor the majority class. In this project, we explore the impact of various class imbalance techniques on the performance of different machine learning algorithms using three imbalanced datasets.

INDEX

Table of Contents

A. OBJECTIVE.....3

B. DATASET 1: CUSTOMER CHURN.....5

C. DATASET 2: INSURANCE.....14

D. DATASET 3: STROKE.....22

A. OBJECTIVE

The primary goal of this project is to investigate and compare the effectiveness of various class imbalance techniques in enhancing the performance of machine learning models on imbalanced datasets. Imbalanced datasets are common in many real-world applications, where the number of instances in one class significantly outweighs those in the other class. This imbalance can lead to biased models that perform poorly on the minority class. To address this, our specific aims are:

1. **Exploration of Class Imbalance Techniques**: We will explore a range of techniques designed to handle class imbalance. These include:
   - Oversampling Methods: Techniques such as SMOTE (Synthetic Minority Over-sampling Technique), ADASYN (Adaptive Synthetic Sampling), and random oversampling.
   - Undersampling Methods: Techniques such as random undersampling, Tomek Links, and cluster-based undersampling.
   - Class Imbalance Algorithms**:** Methods that integrate class imbalance handling within the algorithm itself, such as class weighting and cost-sensitive learning.

2. **Implementation on Imbalanced Datasets**: These techniques will be applied to several imbalanced datasets, implementing various machine learning algorithms to assess their impact. The algorithms will include:
   - Logistic Regression: A linear model used for binary classification.
   - Decision Trees: A non-linear model that splits the data based on feature values.
   - Random Forests: An ensemble method that builds multiple decision trees and merges their results.
   - Gradient Boosting Machines**:** An ensemble technique that builds trees sequentially to reduce prediction errors.
   - K-Nearest Neighbors: A non-parametric method that classifies instances based on their proximity to neighbors.
   - Naïve Bayes: A probabilistic classifier based on Bayes' theorem with strong independence assumptions.
   - Multilayer Perceptron: A class of feedforward artificial neural networks.

3. **Performance Evaluation**: The performance of these models will be rigorously evaluated using multiple metrics to provide a comprehensive assessment. The metrics include:
   - Accuracy: The ratio of correctly predicted instances to the total instances.
   - Precision: The ratio of true positive predictions to the sum of true positives and false positives.

- Recall (Sensitivity): The ratio of true positive predictions to the sum of true positives and false negatives.
- F1-Score: The harmonic mean of precision and recall, providing a balance between the two.
- AUC-ROC (Area Under the Receiver Operating Characteristic Curve): A measure of the model's ability to distinguish between classes.
- AUC-PR (Area Under the Precision-Recall Curve): A measure that provides a more informative view of an imbalanced dataset by focusing on the performance of the minority class.

4. **Identification of Effective Techniques:** The ultimate aim is to identify the most effective class imbalance techniques that improve the performance and generalizability of machine learning models on imbalanced datasets. This involves comparing the performance of models with and without these techniques and determining the optimal approach for different types of datasets and domains.

The analysis will be performed on real-world imbalanced datasets obtained from Kaggle, covering diverse domains such as finance, healthcare and cybersecurity. These domains are chosen because they often encounter significant class imbalances, making them ideal for this study.

By applying and comparing different class imbalance techniques across these varied datasets and domains, we aim to derive insights that are broadly applicable and can guide practitioners in selecting the most suitable techniques for their specific needs. This comprehensive approach will ensure that the findings are robust and generalizable, providing valuable contributions to the field of machine learning and data science.

B. DATASET 1: CUSTOMER CHURN

Predicting customer churn is a critical task for companies, as it allows them to effectively retain their existing customer base. The cost of acquiring new customers is significantly higher than the cost of retaining existing ones. Therefore, large corporations are increasingly focusing on developing models that can predict which customers are likely to churn. By identifying these customers early, companies can implement targeted retention strategies to reduce churn rates and maintain a stable revenue stream.

In this project, we aim to develop robust machine learning models to predict the churn rate of existing customers for an IT company. This will involve analyzing customer data from the past year to identify patterns and indicators that are predictive of churn.

The dataset provided a comprehensive study of an IT company, encompassing various customer attributes such as demographic information, service usage patterns, customer support interactions, billing information, and any previous instances of churn. To gain a thorough understanding of the dataset and its metadata, we first conducted exploratory data analysis (EDA). This step was crucial for uncovering the characteristics of the data, identifying any missing values, and addressing data quality issues. Through data cleaning, normalization, and transformation, we ensured that the data was prepared adequately for model building.

Following the EDA, we checked for class imbalance in the data. Given that predicting customer churn often involves dealing with a significant disparity between the number of customers who churn and those who do not, this step was essential. We then proceeded to establish a baseline by executing five different algorithms: XGBoost, logistic regression, decision tree, random forest, and multilayer perceptron (MLP). This initial execution provided a benchmark for evaluating the effectiveness of the subsequent class imbalance techniques.

To address the class imbalance, we applied various techniques designed to balance the dataset. These included oversampling the minority class, undersampling the majority class, and implementing algorithms with built-in mechanisms for handling class imbalance. With a balanced dataset, we moved on to feature selection. By identifying and selecting the most relevant features, we aimed to improve the performance and efficiency of our models.

After feature selection, we re-applied the machine learning models to the dataset and evaluated their performance using multiple metrics. This comprehensive approach allowed us to assess the impact of the class imbalance techniques and the feature selection process on the predictive power of the models. By iterating through these steps, we aimed to develop robust models capable of accurately predicting customer churn, thereby providing valuable insights for the IT company to enhance their customer retention strategies.

SECTION 1: CLASS IMBALANCE TECHNIQUES

Below is the overview of the specific class imbalance techniques I utilized and the reasons behind their selection, providing a foundation for understanding their impact on the dataset and model performance.

Since the ratio of churning customers is generally significantly lower than the non-churning ones, this dataset is also highly imbalanced and therefore needs to be handled accordingly. For this purpose, we have applied two resampling techniques to balance the dataset and a class imbalance algorithm to handle the imbalance data.

1.1 Random Oversampling

Random oversampling is a class imbalance technique used to address the issue of imbalanced datasets by artificially increasing the number of instances in the minority class. The rationale behind random oversampling is to provide the machine learning model with more representative examples from the minority class, thereby mitigating the imbalance and improving the model's ability to learn patterns from both classes effectively.

In this project, random oversampling was implemented by randomly duplicating examples from the minority class in the training dataset until the class distribution became balanced. This involved randomly selecting instances from the minority class and adding duplicates to the dataset, effectively increasing the number of minority class instances.

To facilitate this process, the RandomOverSampler class from the imbalanced-learn library in Python was utilized. This class provides a convenient and efficient way to perform random oversampling by automatically handling the duplication of minority class instances.

Upon applying random oversampling, the size of the dataset grew significantly. Initially, the dataset contained around 7000 rows, with a substantial class imbalance, where the minority class was underrepresented. However, after applying random oversampling, the number of rows increased to approximately 10000, with around 5000 rows for each class. This balanced distribution ensured that the machine learning models had sufficient examples from both classes to learn from, thus improving their performance in predicting customer churn accurately.

By implementing random oversampling in this manner, we were able to address the imbalance in the dataset and provide the models with a more balanced and representative training dataset, ultimately enhancing their ability to make accurate predictions on customer churn.

1.2 Cluster-Based Undersampling

Cluster-based undersampling, such as the NearMiss technique, is a class imbalance technique employed to alleviate the imbalance in datasets by reducing the number of instances in the majority class. This approach involves identifying clusters within the majority class and subsequently removing instances from these clusters to rebalance the dataset. The rationale behind cluster-based undersampling is to strategically select a subset of examples from the majority class that are closest to the instances in the minority class, thus minimizing the imbalance while maintaining the distribution of the minority class.

In this project, we utilized the NearMiss technique to implement cluster-based undersampling. This technique involves selecting a subset of examples from the majority class that are "near" to the instances in the minority class, effectively reducing the imbalance between classes. To facilitate this process, we employed the NearMiss class from the imbalanced-learn library in Python. This class provides a convenient and efficient way to perform undersampling on the training dataset using the NearMiss technique.

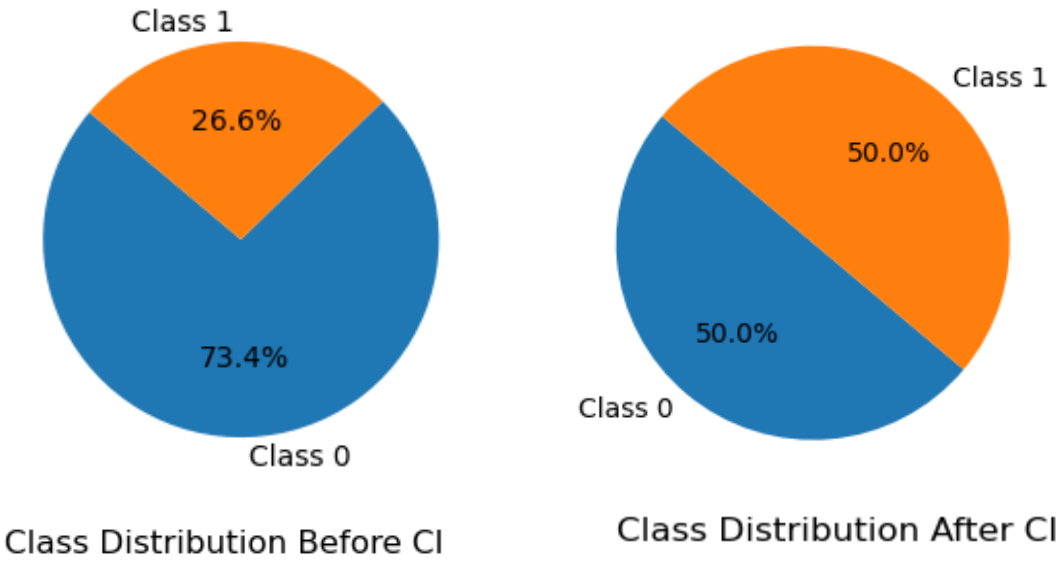
Upon applying cluster-based undersampling using NearMiss, the size of the dataset was significantly reduced. Initially, the dataset contained a larger number of rows, with a considerable class imbalance where the majority class predominated. However, after implementing NearMiss, the number of rows in the dataset decreased to approximately 3700. This reduction in dataset size was a result of selectively removing instances from the majority class clusters to rebalance the dataset.

Despite the reduction in dataset size, the distribution of the minority class was preserved, ensuring that the models were trained on a representative subset of examples from both classes. By strategically selecting instances from the majority class that are close to the minority class, NearMiss effectively minimized the class imbalance and provided the models with a more balanced training dataset, ultimately enhancing their ability to accurately predict customer churn.

The diagram below illustrates the distribution of positive and negative classes before and after implementing Random Oversampling and Cluster-based UnderSampling. Remarkably, both techniques resulted in identical distributions. This symmetrical distribution post-application of both class imbalance (CI) techniques unequivocally indicates that the datasets have been successfully balanced.

This balance in distribution is pivotal as it ensures that the machine learning models are trained on datasets that contain an equitable representation of both classes. Consequently, the models are less likely to be biased toward the majority class and more capable of accurately predicting

customer churn across both classes. Furthermore, the symmetrical distribution affirms the efficacy of both Random Oversampling and Cluster-based UnderSampling in mitigating the class imbalance issue within the dataset.



1.3 Class Weighting

Class weighting is a valuable technique employed to address class imbalance by adjusting the contribution of each class to the overall loss function during model training. Unlike data-level techniques such as oversampling or undersampling, class weighting is an algorithmic approach that does not alter the dataset itself. Instead, it ensures that the model pays more attention to the minority class by assigning higher weights to less frequent classes and lower weights to more frequent classes. In this project, we utilized the Balanced class weights method, which calculates the ratio of the number of samples in the most populous class to the number of samples in each other class.

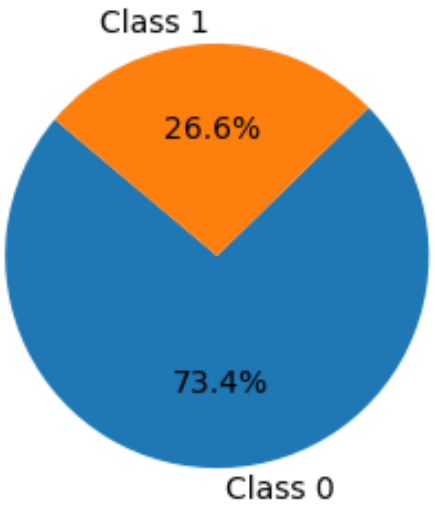
Before applying class weights to the machine learning algorithms, we also employed the Tomek Links undersampling technique. Tomek Links are pairs of instances from different classes that are nearest neighbors of each other. By removing such pairs, Tomek Links can improve the separability between classes and aid in enhancing the performance of machine learning models on imbalanced datasets.

The combination of class weights and Tomek Links can further enhance the performance of machine learning models on imbalanced datasets. While Tomek Links improve the separability between classes by removing pairs of instances that are nearest neighbors of each other and belong to different classes, class weights ensure that the model gives more importance to minority class instances during training. This synergistic approach can lead

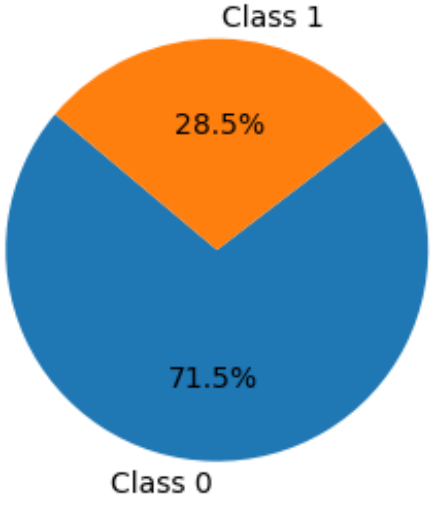
to better generalization and improved performance of machine learning models on imbalanced datasets, ultimately resulting in more accurate predictions of customer churn. By employing both techniques, we aimed to leverage their complementary strengths to address the challenges posed by class imbalance and enhance the overall effectiveness of the predictive models.

The diagram below illustrates the distribution of positive and negative classes before and after the application of Tomek Links. Surprisingly, the distribution indicates that Tomek Links did not significantly address the class imbalance issue, as the dataset remains largely imbalanced. Despite the implementation of Tomek Links, the proportion of the positive class only marginally increased from 26.6% to 28.5%.

This observation suggests that Tomek Links may not be effective in sufficiently rebalancing the dataset in this context. While Tomek Links aim to improve the separability between classes by removing pairs of instances that are nearest neighbors of each other and belong to different classes, their impact on rebalancing the dataset appears limited. As a result, additional techniques or alternative approaches may be necessary to achieve a more balanced distribution and enhance the performance of machine learning models on imbalanced datasets. For this very reason we applied class weights with Tomek links as well.



Class Distribution Before CI



Class Distribution After CI

SECTION 2: IMPACT OF CI TECHNIQUES

This section explores the impact of each CI technique on the performance of the models. This is done by comparing the performance of the baseline execution of the models with the performance after applying Class Imbalance techniques.

In this project, five machine learning models were created using Gradient Boosting Machine, Logistic Regression, Decision Trees, Random Forest, and Multilayer Perceptron. Initially, these models were trained on the imbalanced dataset, and a baseline execution was conducted. Subsequently, the models were retrained on balanced datasets after applying each class imbalance (CI) technique. The impact of each technique on the performance of machine learning algorithms was then evaluated.

2.1 Oversampling:

The models exhibited improved performance after oversampling the data compared to the baseline execution. Particularly noteworthy was the increase in AUC-PR scores, which is a crucial metric for evaluating data imbalance. The higher AUC-PR scores of the models trained on balanced data indicated better handling of the data by the algorithms. While the individual performance of the models varied, the overall performance was enhanced with balanced data.

2.2 Clustering UnderSampling:

Utilizing clustering undersampling, however, resulted in generally poorer performance across models. Most models demonstrated diminished performance compared to the baseline. Although AUC scores showed slight improvement, suggesting relatively better handling of imbalance, the overall model performance was significantly inferior to the baseline.

2.3 Class Weighting:

Implementing class weighting led to a significant improvement in both generalizability and performance across models, with better AUC scores compared to the baseline. While some models performed relatively poorly with lower scores than the baseline, the AUC scores were generally higher. Therefore, even if the overall performance of a model may have been unsatisfactory, it cannot be denied that the handling of data was superior in balanced datasets.

In summary, Random Oversampling emerged as the most potent technique for our dataset, yielding the best scores and significantly improving model performance. Conversely, Cluster UnderSampling using NearMiss proved to be the least effective class imbalance technique,

with minimal impact on the models' performance. One possible reason for this could be the small size of our data, which may have resulted in the loss of information during undersampling, thus leading to the poor performance of Cluster-based Undersampling.

SECTION 3: PERFORMANCE OF ALGORITHMS

This section delves into the variance in performance observed across different algorithms with the application of class imbalance (CI) techniques.

Our initial baseline analysis revealed that despite the dataset's imbalance, certain algorithms such as Logistic Regression and Random Forest exhibited relatively good performance. However, further examination showed that these high scores did not necessarily indicate good models, as evidenced by consistently low AUC-PR scores.

The performance of models with different Class Imbalance technique also varied throughout as

described below:

3.1 Gradient Boosting Machine (GBM):

In the baseline execution, GBM exhibited signs of overfitting, with high training scores but notably lower scores on validation and testing datasets. Upon applying oversampling, GBM's generalizability improved, albeit with a compromise on performance. Conversely, clustering UnderSample resulted in poor performance characterized by overfitting and low scores on testing and validation datasets. However, utilizing class weighting led to consistent performance during cross-validation. Notably, GBM demonstrated optimal performance with oversampling, achieving significantly higher AUC scores compared to the baseline.

3.2 Logistic Regression (LR):

LR showcased relatively strong performance in the baseline execution, with high scores across metrics except for the AUC-PR score, indicating mishandling of imbalance data. Upon oversampling, LR's generalizability improved, albeit with slightly reduced performance, while AUC-PR scores improved significantly, reflecting enhanced data handling. Conversely, clustering UnderSample led to poor performance, with scores hovering around 60%. The utilization of class weighting resulted in well-generalized and improved performance, with higher AUC scores compared to the baseline.

3.3 Decision Trees (DT):

DT displayed high training scores in the baseline execution but significantly lower scores on validation and testing datasets, indicating overfitting. Despite improvements with oversampling, overfitting persisted. However, clustering UnderSample yielded poor model

performance, exacerbating overfitting. Class weighting did not provide reliable results, with continued overfitting. DT demonstrated proficiency in handling imbalance data with both resampling techniques, with class weighting offering marginally improved performance over the baseline.

3.4 Random Forest (RF):

RF exhibited behavior similar to DT in handling data imbalance, with comparable AUC-PR scores. However, RF's overall performance surpassed that of DT. RF performed well with oversampling and class weighting but showed limited effectiveness with clustering UnderSampling. Overfitting remained consistent across all CI techniques and the baseline execution, only mitigated through feature selection.

3.5 Multilayer Perceptron (MLP):

MLP demonstrated better performance compared to DT and RF, with higher generalizability despite lower scores. However, AUC scores across all CI techniques were notably low, suggesting a bias towards the majority class. Overall, logistic regression emerged as the top performer, exhibiting consistent performance improvements with CI techniques, except for clustering UnderSampling, which proved ineffective across all models. Conversely, decision trees and multilayer perceptron emerged as the weakest learners, failing to adequately address data imbalance and favoring the majority class.

In summary, logistic regression emerged as the best-performing model, showcasing well-rounded performance improvements with CI techniques techniques (except for cluster UnderSampling in which none of the models performed well), while decision trees and multilayer perceptron displayed weaknesses in handling data imbalance. These findings are further summarized in the table on the subsequent page which shows the performance of different ML models with each CI technique.

3.6 Performance Variance

Metrics	Type	Gradient Boosting	Logistic Regression	Decision Trees	Random Forest	Multilayer Perceptron
ACCURACY	Baseline	78	80	73	79	80
	CI 1	85	76	84	83	76
	CI 2	68	68	63	71	66
	CI 3	78	77	75	82	80
PRECISION	Baseline	77	80	74	78	80
	CI 1	86	76	85	84	76
	CI 2	68	70	63	73	69
	CI 3	80	82	76	82	80
RECALL	Baseline	78	81	73	79	80
	CI 1	85	75	84	83	76
	CI 2	68	68	63	71	66
	CI 3	78	77	75	82	80
F1	Baseline	77	80	73	79	80
	CI 1	85	75	84	83	76
	CI 2	68	68	63	71	65
	CI 3	79	78	75	82	80
AUC ROC	Baseline	81	85	67	83	84
	CI 1	91	83	84	90	84
	CI 2	74	76	63	78	76
	CI 3	85	87	70	87	85
AUC PR	Baseline	60	66	56	64	65
	CI 1	89	82	87	87	82
	CI 2	70	71	70	73	71
	CI 3	70	73	62	73	68

C. DATASET 2: INSURANCE

Insurance companies that sell life, health, and property and casualty insurance are using machine learning (ML) to drive improvements in customer service, fraud detection, and operational efficiency. The data is provided by an insurance company, providing Health Insurance to its customers, which is not excluded from other companies in getting advantage of ML.

Just like medical insurance, there is vehicle insurance where every year customer needs to pay a premium of certain amount to insurance provider company so that in case of unfortunate accident by the vehicle, the insurance provider company will provide a compensation (called 'sum assured') to the customer.

In this project we aim to develop models to predict whether the policyholders (customers) from past year will also be interested in Vehicle Insurance provided by the company. Building a model to predict whether a customer would be interested in Vehicle Insurance is extremely helpful for the company because it can then accordingly plan its communication strategy to reach out to those customers and optimize its business model and revenue.

The dataset contains information on customer attributes, including their ID, gender, age, region code whether they have driving license, they are previously insured, their vehicle's age, if their vehicle is damaged, their Annual premium package, the policy sales channel, their vintage and how their previous response.

To prepare the data for modeling, we performed data cleaning to address inconsistencies in the data. There were no missing values, so they need not to be handled. Exploratory data analysis (EDA) was also done to understand data characteristics and identify patterns, and normalization and transformation was done to ensure numerical features were on a similar scale and categorical variables were appropriately formatted. Feature engineering was also conducted to create new predictive features from existing attributes. These steps ensured the dataset was ready for effective model training and accurate prediction of customer interest in vehicle insurance.

However, the most important step was to handle the imbalance in dataset which was done by applying several Class Imbalance techniques mentioned in detail in the next section.

SECTION 1: CLASS IMBALANCE TECHNIQUES:

The dataset we utilized exhibited a significant class imbalance, as the ratio of customers who responded positively to the customers who did not was markedly low. Specifically, the distribution of the positive class (customers who responded) was approximately 16%, while the negative class (customers who did not respond) constituted about 84%. This pronounced disparity highlights the highly imbalanced nature of the data, which poses challenges for building accurate and reliable machine learning models.

To address this issue, it is crucial to apply class imbalance techniques prior to model development. These techniques help to balance the dataset, thereby improving the model's ability to correctly predict both classes, rather than being biased towards the majority class.

In our analysis, we employed three different class imbalance techniques to balance the dataset:

1.1 Random UnderSampling

The first technique employed to handle class imbalance was random undersampling. Random undersampling is a method where a subset of the majority class is randomly selected to match the number of instances in the minority class. This technique effectively reduces the size of the dataset by removing excess examples from the majority class, thereby creating a balanced distribution between the classes.

Our original dataset was quite large, containing approximately 400,000 rows. Applying random undersampling not only helped in balancing the data but also significantly reduced its size. This reduction in size made computations faster and streamlined the overall modeling process, adhering to the principle of simplicity advocated by Occam's Razor, which suggests that the simplest solution is often the best.

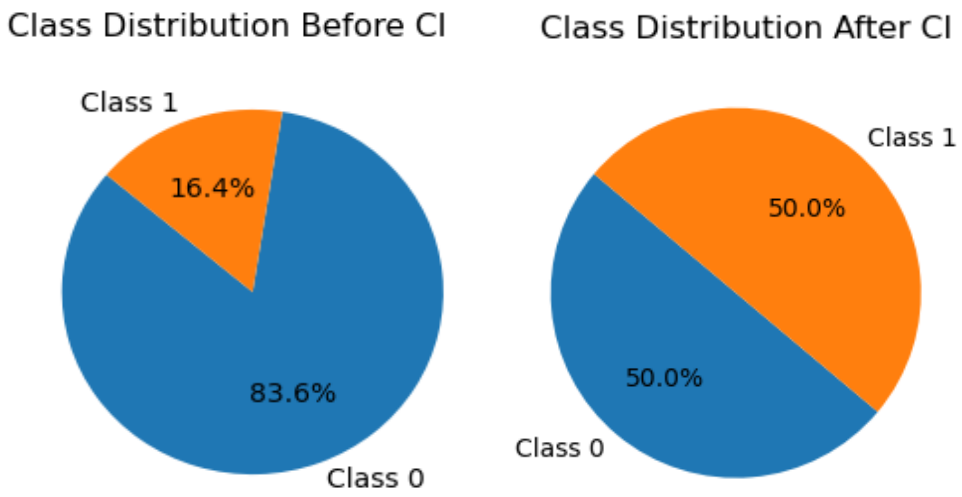
After applying random undersampling, the size of the dataset was reduced to around 125,000 rows, resulting in a balanced dataset with an equal number of instances for both classes. This balanced dataset facilitated more effective training of machine learning models, as the models could now learn equally from both the majority and minority classes, thereby improving their ability to generalize and make accurate predictions on new data.

1.2 Cluster based Oversampling using SMOTE:

The second technique employed to handle class imbalance was cluster-based oversampling using SMOTE (Synthetic Minority Over-sampling Technique). SMOTE works by generating synthetic samples for the minority class through interpolation between existing instances, rather than merely duplicating them. To enhance this process, we first clustered the minority class instances using the K-Means algorithm, which grouped similar instances together based on their features. Within each cluster, SMOTE was applied to generate new synthetic samples, ensuring that these samples accurately represented the local structure and patterns

of the minority class. This method increased the size of our dataset from around 400,000 to approximately 600,000 rows, thereby adding complexity but achieving a balanced dataset. The balanced data enabled our machine learning models to learn from a more diverse and representative set of examples, ultimately improving their performance and generalizability.

The diagram below illustrates how both of the techniques were able to handle the highly imbalanced dataset in question, by distributing the majority and minority classes equally.



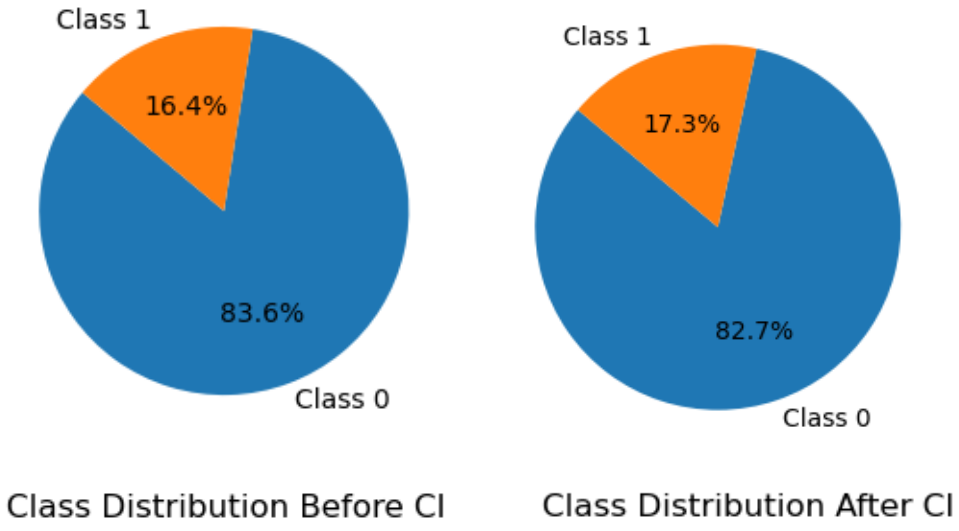
1.3 Class Weighting

The third technique we employed to handle class imbalance is class weighting. Unlike data-level techniques such as random undersampling or cluster-based oversampling, class weighting is an algorithmic approach that integrates directly with the machine learning algorithm to address imbalances without altering the dataset itself. This technique works by adjusting the contribution of each class to the overall loss function during model training. Specifically, higher weights are assigned to the minority class, ensuring that the model gives more attention to these less frequent instances, while lower weights are assigned to the majority class.

Before implementing class weighting, we applied the Tomek Links undersampling technique to further refine the dataset. Tomek Links identifies pairs of instances from different classes that are nearest neighbors. By removing these pairs, we aimed to improve the separability between the classes, thereby enhancing the performance of our models. This preprocessing step helps to create a clearer distinction between classes, which is particularly beneficial when combined with class weighting.

However, this approach did not seem to be very useful for our dataset as illustrated in the diagram below which shows the distribution of positive and negative classes before and after

applying Tomek Links. Tomek Links contributed only a slight increase in the proportion of the positive class and the dataset remained largely imbalanced, moving from 16.4% to 17.3% only. This observation suggests that while Tomek Links improve class separability, their impact on overall class balance might be limited in certain contexts. Thus, the integration of class weighting became crucial to effectively handle the remaining imbalance and ensure the model's focus on the minority class during training.



SECTION 2: IMPACT OF CI TECHNIQUES

In this project, we began by executing a baseline evaluation without applying any class imbalance techniques. We assessed the performance of several machine learning models to establish a reference point. The algorithms used to create these models included K-Nearest Neighbors (KNN), Logistic Regression, Decision Trees, Random Forest, and Multilayer Perceptron (MLP).

These initial evaluations revealed significant overfitting across most of the algorithms due to the highly imbalanced nature of the dataset, which favored the majority class. This overfitting was evident from the extraordinarily high accuracy scores. However, these accuracy scores were misleading because they did not reflect the models' true performance on the minority class.

For this purpose, we examined the Area Under the Precision-Recall Curve (AUC-PR) scores. The AUC-PR metric is particularly useful in the context of imbalanced datasets as it focuses on the performance with respect to the minority class. Unfortunately, we observed extremely low AUC-PR scores, indicating that the models performed poorly in identifying the minority class despite their high overall accuracy.

After establishing the baseline performance, we applied various class imbalance techniques to address the skewed distribution of the dataset and reevaluated the performance of the models on each of these techniques.

1. Random Under Sampling

Random under-sampling proved effective in addressing class imbalance across most models. Some models like KNN, Decision Tree and Random Forest showed a significant drop in performance from training to testing, highlighting overfitting concerns. However, the AUC-PR scores improved for almost all the algorithms, indicating better handling of imbalance using this technique which was further improved by feature selection.

2. Cluster based Over Sampling

Cluster-based over-sampling using SMOTE, significantly improved the performance of most models by handling imbalance in the dataset quite well. Models consistently showed high performance metrics, including accuracy, precision, recall, and F1 scores across training, validation, and testing datasets, indicating good generalization. However, the Random Forest model exhibited signs of overfitting, as evidenced by exceptionally high training performance metrics that did not translate as well to validation and testing sets. Despite this, the AUC-PR scores remained consistently high across all models, highlighting effective handling of class imbalance. Feature selection with Wrapper's methods, further enhanced performance and reduced overfitting, though methods like Variance Thresholding and PCA were less effective and sometimes even worse.

3. Class Weighting

The combination of class weighting and Tomek links yielded mixed results across different models. While this technique aimed to improve class balance, many models continued to show signs of imbalance, reflected in lower AUC-PR scores. Logistic Regression and Decision Trees particularly struggled with low accuracy, recall, and F1 scores, indicating poor learning from the minority class. Although precision and AUC-ROC were reasonable, the low AUC-PR scores highlighted ineffective imbalance handling. Overfitting was a concern in some models, but Forward Feature Selection helped improve generalization and performance, producing slightly higher AUC-PR scores. Other feature selection methods, such as VT and PCA, failed to enhance performance and often worsened it.

Overall, cluster-based oversampling using SMOTE proved to be the most effective technique in this study. It delivered well-rounded performance across all models, which further improved with the application of feature selection. The models showed high accuracy, precision, recall, and F1 scores, and consistently high AUC-PR scores, indicating successful handling of class imbalance. In contrast, class weighting did not perform as expected. None of the models using class weighting produced sufficiently high AUC-PR scores to demonstrate effective handling of the

dataset's imbalance. This suggests that while class weighting can be beneficial in some contexts, it was not as effective as cluster-based oversampling in this case.

SECTION 3: PERFORMANCE OF ALGORITHMS

3.1 KNN

For KNN, the baseline model had high accuracy but low AUC-PR scores, indicating poor handling of class imbalance. Random under-sampling improved AUC-PR scores, with Recursive Feature Elimination (RFE) reducing overfitting and enhancing performance, while VT, FFS, and PCA decreased performance. Cluster-based oversampling with SMOTE led to high performance metrics (around 90%), though there was a slight drop on testing data. Cross-validation confirmed robustness, and Wrapper's method further improved generalizability and AUC-PR scores, whereas VT and PCA did not perform well. Class weighting with Tomek links resulted in low AUC-PR scores and precision below 75%, indicating ineffective handling of class imbalance across all feature selection methods.

3.2 Logistic Regression

The baseline Logistic Regression model had good accuracy, precision, and recall but moderate AUC-PR scores, showing some imbalance issues. Random under-sampling showed no signs of overfitting, with high performance metrics and Wrapper's method improving AUC scores from 80% to 86%, while VT and PCA failed to enhance performance. With cluster-based oversampling (SMOTE), the model showed high and consistent performance across datasets. Wrapper's method significantly improved AUC-PR scores, but VT and PCA did not contribute much to performance enhancement. Using class weighting, Logistic Regression displayed low accuracy, recall, and F1 scores, with reasonable precision and AUC-ROC but low AUC-PR (49%). Feature selection methods did not significantly improve these scores.

3.3 Decision Trees

The baseline Decision Tree model exhibited high overfitting with training accuracy at 99% and testing at 76%, and moderate AUC-PR scores, indicating poor imbalance handling. Random under-sampling improved performance with Wrapper's methods, resulting in consistent 82% scores across training, validation, and testing datasets, and high AUC-PR scores after feature selection, while VT and PCA worsened performance. Cluster-based oversampling (SMOTE) initially showed overfitting but good validation and testing performance (88%). Cross-validation supported robustness, and Wrapper's methods reduced overfitting and improved AUC-PR scores, though VT and PCA failed to improve the model. With class weighting, Decision Tree continued to show overfitting and low validation and

testing performance, with low AUC-PR scores. However, Forward Feature Selection improved performance and AUC-PR scores, while other feature selection methods did not enhance performance.

3.4 Random Forest

The baseline Random Forest model had high performance metrics but significant overfitting. Random under-sampling, coupled with Recursive Feature Elimination (RFE), reduced overfitting and maintained high AUC-PR scores, indicating effective imbalance handling, while VT and PCA did not enhance performance. Using cluster-based oversampling (SMOTE), Random Forest showed high performance but signs of overfitting. Recursive Feature Elimination improved generalization and AUC-PR scores, though VT and PCA did not perform well. Class weighting led to overfitting with low AUC-PR scores. Cross-validation revealed low positive precision and F1 scores, but Forward Feature Selection produced a well-generalized model with high AUC-PR scores, whereas other feature selection methods performed poorly.

3.5 Multilayer Perceptron

The baseline MLP model showed good performance metrics, but potential imbalance issues indicated by lower AUC-PR scores. Random under-sampling resulted in consistent performance across datasets, with Wrapper's methods improving precision and AUC scores, while VT generalized well but performed worse, and PCA failed. Cluster-based oversampling (SMOTE) led to high and consistent performance across datasets, with Wrapper's methods improving precision and AUC scores, though VT generalized well but performed worse, and PCA failed. Class weighting yielded good overall performance but low AUC-PR scores. Cross-validation showed lower precision, but Wrapper's method increased precision and improved overall model performance, while other feature selection methods did not significantly enhance performance.

Overall, by looking at the combined performance of each model, it is observed that Multilayer Perceptron has performed the best since it produced the most well generalized model having good overall scores which got even better with the right selection of features. It also handled imbalance well specially with first two CI techniques.

The Random Forest and Decision trees algorithms however, showed clear signs of overfitting.

In handling imbalance, Logistic Regression proved to be the weakest model.

These findings are further summarized in the table on the subsequent page which shows the performance of different ML models with each CI technique.

3.6 Performance Variance

Metrics	Type	KNN	Logistic Regression	Decision Trees	Random Forest	Multilayer Perceptron
ACCURACY	Baseline	83	83	81	84	83
	CI 1	80	81	76	82	83
	CI 2	86	82	88	90	84
	CI 3	86	74	83	86	85
PRECISION	Baseline	82	77	82	82	84
	CI 1	81	86	76	83	84
	CI 2	87	86	88	90	85
	CI 3	85	78	83	85	85
RECALL	Baseline	83	83	81	84	83
	CI 1	80	81	76	82	83
	CI 2	86	82	88	90	84
	CI 3	86	74	83	86	85
F1	Baseline	82	76	81	83	84
	CI 1	80	80	76	82	83
	CI 2	86	81	88	90	84
	CI 3	86	77	83	85	85
AUC ROC	Baseline	83	85	67	88	88
	CI 1	86	85	76	88	88
	CI 2	92	86	88	97	90
	CI 3	88	88	70	91	91
AUC PR	Baseline	45	43	49	49	49
	CI 1	82	78	82	82	82
	CI 2	91	79	91	97	84
	CI 3	60	49	55	61	58

D. DATASET 3: STROKE

A stroke, also known as a cerebrovascular accident (CVA), is a severe medical condition where part of the brain loses its blood supply, leading to the loss of function in the part of the body that the affected brain cells control. This disruption can manifest in two primary forms: ischemic and hemorrhagic strokes.

Using machine learning techniques to analyze this dataset can help in developing predictive models that identify individuals at high risk of stroke. Such models can assist healthcare professionals in early intervention and personalized treatment plans. However, due to the imbalance, special techniques like resampling, cost-sensitive learning, and advanced algorithms such as ensemble methods are often employed to improve model performance and ensure accurate predictions.

Understanding strokes, their types, and the critical need for prompt treatment underscores the importance of datasets aimed at predicting stroke risk. Accurate predictions can lead to better preventive measures and timely medical interventions, ultimately reducing the burden of stroke-related morbidity and mortality.

SECTION 1: CLASS IMBALANCE TECHNIQUES:

We have applied several class imbalance (CI) techniques to assess their results and impact on our dataset. These techniques are crucial for balancing the classes in data, which is essential for achieving more accurate and reliable outcomes in machine learning models. By addressing class imbalance, we ensure that our models are not biased towards the majority class and can effectively learn to recognize and predict instances of the minority class. This process involves various methods, such as resampling, advanced algorithms and Gaussian Mixture Models (GMM). Each technique is evaluated to determine its effectiveness in improving model performance and its overall impact on the predictive accuracy of the dataset. By comparing the results of these CI techniques, we can identify the most suitable approach for balancing our specific dataset, leading to more robust and equitable machine learning solutions.

1.1 Random oversampling:

Random oversampling is a powerful technique to address class imbalance in datasets, which is particularly useful for medical datasets like the "Cerebral Stroke Prediction" dataset. This method involves increasing the number of instances in the minority class (stroke cases) by randomly replicating existing samples until the class distribution is balanced. By doing so, it helps to ensure that the machine learning model pays equal

attention to both classes during training. This process can significantly enhance the model's ability to correctly identify stroke cases, which are typically underrepresented in the dataset, thus improving its overall performance and reliability.

1.2Gaussian Mixture Model (GMM):

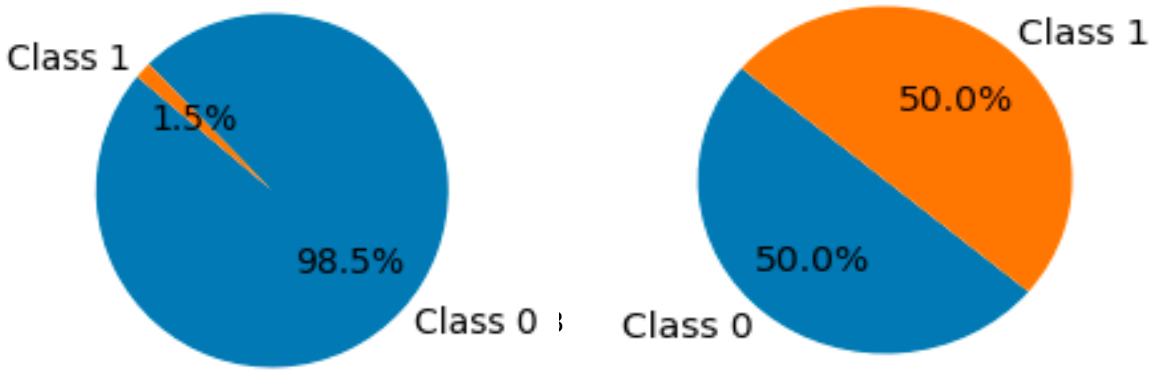
Gaussian Mixture Model (GMM) is an advanced clustering technique used to address class imbalance in datasets, such as the "Cerebral Stroke Prediction" dataset. Class imbalance occurs when there are significantly fewer instances of the minority class (stroke cases) compared to the majority class (non-stroke cases). GMM helps identify underlying clusters within the data, making it relatively robust to outliers and suitable for high-dimensional datasets. By fitting a GMM to the minority class, we can predict the probability of each instance belonging to different components of the mixture model, thereby generating more minority class instances to balance the dataset.

The implementation of GMM involves several steps. First, the dataset is divided into majority and minority classes. A GMM is then fitted to the features of the minority class to capture its distribution. The model predicts probabilities for each minority instance belonging to the identified clusters. By setting a threshold on these probabilities, additional instances are labeled as the minority class. This approach increases the number of minority class instances without introducing synthetic samples. Finally, these instances are duplicated to match the number of majority class instances, resulting in a balanced dataset. This method ensures a balanced class distribution, improving the model's ability to predict stroke cases effectively.

1.3 Algorithmic Methods:

Algorithmic methods alone can’t fully balance or equalize both classes but they do have a significant impact on the data. These methods typically adjust the learning process to better handle the skewed distribution of classes, thereby improving the model's ability to recognize minority class instances. Recognizing the limitations of purely algorithmic approaches, we initially balanced the dataset using a simple sampling technique to create equal representation for both classes. After achieving a balanced dataset, we applied algorithmic techniques, including the addition of class weights and ensemble methods, to further enhance the model's performance.

We experimented with five different algorithms, incorporating class weights to give more importance to the minority class during the learning process.



SECTION 2: IMPACT OF CI TECHNIQUES

2.1 Random oversampling:

Effectively balances class distribution by increasing number of instances and change the data by making duplicates. Classification is more impactful and algorithms learn the data more accurately. Random forest makes a straightforward solution for addressing class imbalance.

In the application of random oversampling to the stroke dataset, the number of stroke cases was increased from a minority count to match the 41,295 instances of non-stroke cases, resulting in a balanced dataset with 41,295 instances in each class. This balance is crucial as it reduces the model's bias towards the majority class and helps in achieving better evaluation metrics such as precision, recall, and F1-score for the minority class. However, it's important to be cautious of potential drawbacks like overfitting, where the model might learn the noise in the replicated samples rather than general patterns. Despite these considerations, random oversampling remains an effective strategy for mitigating class imbalance and enhancing predictive performance in imbalanced datasets.

2.2 Gaussian Mixture Models (GMM):

The balanced distribution indicates that the minority class instances have been successfully increased and duplicated to match the number of majority class instances. This balance is critical for training machine learning models, as it reduces the model's bias towards the majority class and enhances its ability to detect and predict stroke cases accurately. By ensuring an equal representation of both classes, the predictive model can achieve better performance metrics and provide more reliable and equitable healthcare predictions.

Although it has balanced our dataset, but this technique is not a powerful one. In this data distribution within each cluster follows a Gaussian distribution, which may not hold true for all datasets, leading to biased results.

We can’t find good results by this technique, mostly algorithms give overfitting results of 99% to 100%.

2.3 Algorithmic Methods:

Algorithmic methods play a crucial role in addressing class imbalance problems in machine learning. These techniques aim to adjust the learning process to handle the skewed distribution of classes more effectively, ultimately improving the model's ability to accurately predict instances from both classes. Methods such as adjusting class

weights, utilizing ensemble techniques like Random Forest and Gradient Boosting, and implementing cost-sensitive learning algorithms help mitigate the biases introduced by imbalanced datasets.

Despite the range of algorithmic methods available for addressing class imbalance, some techniques may have limited impact on the dataset. However, one standout performer in this regard is Naïve Bayes, which consistently demonstrates effectiveness in learning from imbalanced data points. Naïve Bayes is renowned for its simplicity and efficiency, particularly in scenarios where class imbalance is prevalent.

As second Algorithm CI1 and CI2 is applied on Logistic Regression and CI3 is applied on Balanced Bagging.

SECTION 3: PERFORMANCE OF ALGORITHMS

3.1 KNN

K-Nearest Neighbors (KNN) as an algorithmic method utilized for addressing class imbalance. However, KNN models may suffer from overfitting, evident from high testing accuracy rates like 93%. Cross-validation's impact may be minor, indicating potential generalization issues. To combat overfitting and enhance performance, feature selection methods such as recursive and forward feature selection are employed. Recursive feature selection yields a testing accuracy of 81%, while forward feature selection achieves 86%, signifying superior fit. Evaluation metrics like precision, recall, and F1 score consistently measure the model's ability to identify positive instances, suggesting balanced performance. Additionally, high AUC-ROC (96%) and AUC-PR (96%) values demonstrate the model's strong discriminative ability and effectiveness in ranking positive instances, respectively. Overall, KNN, when coupled with feature selection and robust evaluation, serves as a valuable algorithmic tool for mitigating class imbalance and improving prediction accuracy.

3.2 Balanced Bagging/ Logistic Regression

Balanced Bagging, a class imbalance (CI) technique, exhibits overfitting with a high testing accuracy of 98% and minimal impact from Cross Validation. Despite applying feature selection, the testing accuracy decreases only marginally by 1%. However, precision, recall, and F1 score remain consistent across training, validation, and test sets, suggesting poor generalization and unsuitability for real-world scenarios. Although AUC-ROC and AUC-PR scores are high at 0.99, indicating excellent discriminative ability and ranking of positive instances, the model's performance on unseen data remains questionable.

Logistic Regression gives better fitting results after applying Feature Selection methods.

3.3 Decision Tree

Decision Tree, as a class imbalance (CI) technique, exhibits signs of overfitting, evident from its perfect performance on the training data (100%). Although the model demonstrates excellent test precision, recall, F1 score, and accuracy, hovering around 97.5%, it raises concerns regarding its generalization to unseen data. The minimal impact of Decision Tree on handling class imbalance suggests that it may not effectively address the imbalance issue in the dataset. Despite achieving high AUC-ROC (97%) and AUC-PR (98%) scores, indicating good discriminative ability and ranking of positive instances, the model's susceptibility to overfitting implies potential shortcomings in real-world performance.

3.4 Naïve Bayes

Naive Bayes, demonstrates consistent performance on the given data with balanced results across training (76%), validation (76%), and testing (76%). Despite the reduction in accuracy to 74% after applying recursive feature elimination, indicating improved generalization, Naive Bayes still maintains a satisfactory fit. Precision, recall, and F1 score remain consistent across all datasets, indicating the model's ability to correctly identify positive instances regardless of the data subset.

3.5 Random Forest

Random Forest, employed as a class imbalance (CI) technique, demonstrates excellent performance in learning patterns from the data, resulting in overfitting with perfect training accuracy (100%) and high validation accuracy (96%). While the testing accuracy slightly increases to 96%, indicating good generalization, it still suggests the presence of overfitting. Despite applying feature selection techniques, such as variance thresholding, which yields a minor decrease in accuracy to 93% testing, overfitting persists.

Precision, recall, and F1 score remain consistent across training, validation, and testing sets, indicating the model's ability to correctly identify positive instances. However, the observed overfitting suggests that Random Forest may not perform well in live testing scenarios, as it struggles to generalize beyond the training data.

3.6 Performance Variance

Metrics	Type	KNN	Logistic Regression/Bagging	Decision Tree	Naïve Bayes	Random Forest
ACCURACY	Baseline	98	98	96	91	98
	CI 1	76	77	79	76	79
	CI 2	80	77	96	73	94
	CI 3	81	97	96	74	93
PRECISION	Baseline	97	97	97	97	97
	CI 1	76	78	80	76	80
	CI 2	80	78	96	74	94
	CI 3	81	97	96	75	93
RECALL	Baseline	98	98	96	91	98
	CI 1	76	77	79	73	79
	CI 2	80	77	96	73	94
	CI 3	81	97	96	74	93
F1	Baseline	97	97	96	94	97
	CI 1	76	77	79	73	79
	CI 2	80	77	96	73	94
	CI 3	81	97	96	74	93
AUC ROC	Baseline	57	86	51	-	76
	CI 1	83	85	87	-	87
	CI 2	88	85	96	-	98
	CI 3	88	99	96	-	-
AUC PR	Baseline	03	06	04	-	04
	CI 1	81	82	84	-	84
	CI 2	88	80	97	-	98
	CI 3	88	99	97	-	-