Dealing with class imbalance in medical datasets with Active Learning and SMOTE

Raul Felipe SENA ROJAS

Master 2 IASD: Artificial Intelligence, Systems, Data

Université Paris Sciences et Lettres

Acknowledgments

This project would not have been possible without the support of many people. I want to express my gratitude to my primary supervisor, Sana Ben HAMIDA MRABET, who guided me throughout this project. I would also like to thank my family and friends who supported me all along the way. Also, I would like to thank PSL – University, which has committed to applying the same tuition fees for all students, whether French or International, thus making it possible for me to study this Master.

Abstract

Class Imbalance occurs when one of the target variables in a classification problem is under-represented with respect to the other classes, for example, in medical diagnostics where sick persons are usually a smaller percentage compared to healthy people. This problem poses several difficulties in the learning process of a Machine Learning algorithm because most of these models were designed around the assumption of an equal number of examples for each class, and ignoring this assumption results in models with poor predictive performance, especially with the minority class. This poor performance is problematic because usually, this class is more important, which is critical in the case of medical diagnostics, as the consequences of wrongly classifying someone as not sick when they are sick are enormous.

One way to solve the previous issue is to create synthetic samples of the dataset with a technique called SMOTE, which has many variants, e.g., Borderline, Adasyn, and ADOMS. These techniques balance the classes and thus help the learning algorithm to learn more efficiently. However, creating synthetic samples comes with a cost; one of these costs is that we can generate synthetic data that has no sense in the real world, for example, a person with 0 glucose or weighing 500 kg. As a general rule, the more samples we generate with synthetic sampling, the more assumptions we make from our data distribution, thus increasing the probability of making a no-sense sample. Also, inputting synthetic data into our problem may result in a classifier without practical use in reality.

Generally, we generate as many synthetic samples, until we have a dataset with balance classes. That means if we have 90 samples in the majority class and 10 in the minority class, we will need to generate 80 synthetic minority class samples. However, is this always necessary? What if we can generate as few synthetic samples as possible and achieve the same results as if we balance the whole dataset? In doing so, we would avoid the cost of creating a lot of synthetic samples. The present work aims to achieve a state of the art results in medical datasets with fewer synthetic samples, and this is done with a novel technique that combines Active Learning and SMOTE.

Table of Contents

Chapter 1 Introduction 1

Chapter 2 Theoretical Framework 1

2.1. Cost-Sensitive Learning 1

2.2. Data Level Preprocessing Methods 4

2.2.1. Synthetic Minority Oversampling Technique (SMOTE) 6

2.2.2. Borderline – SMOTE 9

2.2.3. Adjusting the Direction of the Synthetic Minority Class Examples: ADOMS 12

2.2.4. ADASYN: Adaptive Synthetic Sampling Approach 14

2.2.5. ROSE: Random Oversampling Examples 16

2.3. Active Learning 17

2.3.1. Uncertainty Sampling 18

2.3.2. Diversity Sampling 20

2.3.3. Combining Uncertainty and Diversity Sampling 22

Chapter 3 Experimentation 22

3.1. Active Learning with SMOTE, algorithm description 22

3.2. Experiment description 23

3.2.1. Machine Learning models evaluated 23

3.2.2. Sampling techniques evaluated 24

3.3. PIMA Indians Diabetes Dataset 26

3.3.1. EDA PIMA 26

3.3.2. Preprocessing and Feature Engineering 30

3.3.3. Training results PIMA Active Learning with SMOTE and other SMOTE techniques 31

3.4. Breast Cancer Wisconsin Dataset 35

3.4.1. EDA Breast Cancer 35
3.4.2. Preprocessing and Feature Engineering Breast Cancer 37
3.4.3. Training results from Breast Cancer Active Learning with SMOTE and other SMOTE techniques 37
Chapter 4 Conclusions 43
Chapter 5 Recommendations and Future Research 43

Table of Figures

Figure 1 Cost Matrix for a two-class problem. Source: [1] 2
Figure 2 “Graphical example of Sampling Techniques.” Source: [2] 5
Figure 3 Chessboard Dataset. Source: [3] 6
Figure 4 “Illustration on how to create the synthetic data points with SMOTE” Source: [1] 7
Figure 5 Imbalanced chessboard treated with SMOTE. Source: [3] 9
Figure 6 An example of a Decision Boundary between two classes 10
Figure 7 Imbalanced chessboard dataset treated with Borderline-SMOTE1. Source: [3] 12
Figure 8 Imbalanced chessboard treated with ADASYN. Source: [3] 15
Figure 9 Imbalanced chessboard with ROSE (original points not included). Source: [3] 17
Figure 10 “JSON-encoded example of a prediction” 18
Figure 11 “Uncertainty sampling from different ML algorithms” Source:[6] 20
Figure 12 “Example of Diversity Sampling” Source: [6] 21
Figure 13 “Least-Confidence combine with clustering-based sampling” Source: [6] 22
Figure 14 “Hyperparameter space for the models” 24
Figure 15 “Data Distribution of PIMA dataset” 26
Figure 16 “Pairplot PIMA” 27
Figure 17 “Correlation between Features PIMA” 28
Figure 18 “Possible NA Values PIMA” 28
Figure 19 “Distribution of Diabetes between Age and Pregnancies” 29
Figure 20 “Distribution of Age and Pregnancies” 29
Figure 21 “Density Plot Glucose and Diab Pedigree” 29
Figure 22 “Boxplots Pima features” 30
Figure 23 “Educated Guess PIMA” 30
Figure 24 “Standard Scaler PIMA dataset” 31
Figure 25 “Active Learning SMOTE vs. Classical Methods with LogReg in PIMA” 31
Figure 26 “Active Learning SMOTE vs. Classical Methods with SVC in PIMA” 32

Figure 27 “Active Learning SMOTE vs. Classical Methods with Gradient Boost in PIMA” 33
Figure 28 “Active Learning SMOTE vs. Classical Methods with AdaBoost in PIMA” 34
Figure 29 “Active Learning SMOTE vs. Classical Methods with Random Forest in PIMA” 35
Figure 30 “Class Imbalance from Breast Cancer Dataset” 37
Figure 31 “Active Learning SMOTE vs. Classical Methods with LogReg Breast Cancer” 38
Figure 32 “Active Learning SMOTE vs. Classical Methods with SVM Breast Cancer” 39
Figure 33 “AL SMOTE vs. Classical Methods Gradient Boost Breast Cancer” 40
Figure 34 “AL SMOTE vs. Classical Methods Ada Boost Breast Cancer” 41
Figure 35 “AL SMOTE vs. Classical Methods Random Forest Breast Cancer” 42

Table of Algorithms

Algorithm 1 SMOTE. Source: [1] 8
Algorithm 2 “SMOTE with Active Learning Algorithm” 23

Chapter 1 Introduction

The problem with imbalanced datasets is that classification algorithms are frequently biased towards the majority classes (sometimes called “negative”). For this reason, there is a higher misclassification rate in the minority class (known as the “positive” class). Many solutions to this problem are suitable for standard machine learning algorithms and ensemble techniques. These solutions are categorized into three major groups:

1. **Cost-Sensitive Learning:** This solution changes the cost function of a given algorithm and assigns a higher cost for the misclassification of the positive class.
2. **Data Level Preprocessing Methods:** This procedure modifies the training instances in such a way as to produce a more adequate or balanced class distribution, and this helps the classifier achieve better performance.
3. **Algorithmic Modification:** This modification is oriented towards adapting based learning methods to be more attuned to imbalanced class issues.

Chapter 2 Theoretical Framework

2.1. Cost-Sensitive Learning

Cost-sensitive learning is an aspect of algorithm-level modifications for class imbalance. This modification refers to a specific set of algorithms sensitive to different costs associated with certain characteristics of considered problems. These costs can be learned during the classifier training phase or be provided by a domain expert. There exist two different distinct views on cost-sensitive learning in the literature. These are the following:

1. **Cost associated with classes:** This technique considers that making errors on instances coming from a particular class is associated with a higher cost. There are two views for this approach: A financial perspective (e.g., giving credit to a person with a bad credit score will potentially cause higher losses to a bank than declining credit to a person with a good score) or the other scenario priority/health/ethical issues (e.g., sending a cancer patient home is much more costly than assigning additional tests to a healthy person).

2. **Cost associated with features:** This method supposes that obtaining a particular feature is connected to a given cost, also known as test cost. We can view this from a monetary perspective (e.g., a feature is more expensive to obtain as it requires more resources) or other inconveniences (e.g., the measurement procedure is unpleasant, puts a person at risk, or is difficult to obtain). In other words, this approach aims at creating a classifier that obtains the best possible predictive performance while utilizing features that can be obtained at the lowest cost possible.

In the context of class imbalance, cost-sensitive learning can be seen as a specific type of algorithm-level approach. It assumes asymmetric misclassification costs between classes, defined in the form of a cost matrix. Standard machine learning methods commonly use the so-called 0–1 loss function, which allocates value 0 to a correctly classified instance and 1 to an incorrectly classified one. Then the training procedure aims at minimizing the overall cost, i.e., minimizing the number of wrong predictions. As the 0–1 loss function uses the exact cost associated with a wrong classification for all classes considered, it is highly susceptible to skewed class distributions. The 0–1 loss function over imbalanced data can be easily minimized by focusing on the negative class and overlooking (or, in extreme cases, even wholly ignoring) the positive class. This problem gets more prevalent when the imbalance ratio increases.

	True positive	True negative
Predicted positive	$C(0, 0)$	$C(0, 1)$
Predicted negative	$C(1, 0)$	$C(1, 1)$

Figure 1 Cost Matrix for a two-class problem. Source: [1]

Concretely, cost-sensitive learning aims to alleviate this problem by adopting a different loss function with different costs associated with each class. This cost can be seen as penalizing a class during a classifier training procedure, seeking to increase the importance of the minority class. By stronger penalizing errors in a given class, we push the classifier training procedure (whose purpose is to minimize the overall cost) to

concentrate on instances coming from this distribution. An example of such a cost matrix is given in Figure 1.

In the cost matrix of Figure 1, a new instance must be classified as belonging to a class described by the lowest expected cost. The previous statement is known as the minimum expected cost principle. The expected cost (conditional risk) $\mathrm{R}(i|\,x)$ of classifying instance x as belonging to the $i-th$ class can be represented as:

$$\mathrm{R}(i|\,x) = \sum_{j=1}^{M} P(j|\,x) \cdot C(i,j)$$

Equation 1

where $P(j|\,x)$ is the probability estimation of classifying instance x as belonging to a class j from a set of M classes.

For a standard two-class problem, a cost-sensitive classifier will classify given instance x as belonging to a positive class if and only if:

$$P(0|\,x) \cdot C(1,0) + P(1|\,x) \cdot C(1,1) \leq P(0|\,x) \cdot C(0,0) + P(1|\,x) \cdot C(0,1)$$

Equation 2

from which we can derivate the following:

$$P(0|\,x) \cdot (\,C(1,0) - C(0,0)\,) \leq P(1|\,x) \cdot (\,C(0,1) - C(1,1)\,)$$

Equation 3

Equation 3 proves that any cost matrix can work under the assumption that $C(0,0) = C(1,1) = 0$ (and analogically for multi-class problems). This reduces the number of cost parameters to be established, as one is only interested in misclassification costs among classes. Thanks to this assumption, a cost-sensitive classifier will classify given instance x as belonging to a positive class if and only if:

$$P(0|\,x) \cdot C(1,0) \leq P(1|\,x) \cdot C(0,1)$$

Equation 4

By following the fact that $P(0|\,x) = 1 - P(1|\,x)$, we can get a threshold $p*$ for classifying an instance x as belonging to a positive class if $P(1|\,x) \geq p*$, where:

$$p* = \frac{C(1,0)}{C(1,0) - C(0,1)}$$

Equation 5

Cost-sensitive algorithms can be divided into two main groups:

- **Meta-learning approaches:** This method does not modify the underlying training algorithm. Therefore, it is appropriate for almost any type of classifier. This procedure is based on modifying either the training data or the outputs of a classifier and can be used during two different steps of the classification process:
    - **Preprocessing:** Here, we seek to modify the training data similarly to data-level solutions. The most widespread technique includes weighting the instances according to a provided cost matrix, assigning higher importance to minority objects.
    - **Postprocessing:** This approach aims to modify the outputs of a classifier during the classification phase. We achieved this without any modification before or during training. The fundamental struggle is introducing the cost factor when deciding on a new instance.
- **Direct approaches:** In contrast with meta-learning, this strategy introduces the misclassification cost into the training procedure of a classifier. This directly resembles other algorithm-level approaches, with the difference in utilizing the cost matrix.

2.2. Data Level Preprocessing Methods

Data preprocessing methods consist of procedures to modify the imbalanced dataset to more adequate or balanced data distribution. This is helpful for many classifiers because rebalancing the dataset significantly improves their performance. This section will review the under-sampling and oversampling techniques such as SMOTE. These techniques are simple and easy to implement. However, no clear rule tells us which technique works best. Resampling techniques can be categorized into three groups:

- **Oversampling Methods:** This method replicates some instances or creates new instances from existing ones, thus creating a superset of the original dataset.
- **Under-sampling: Methods:** Create a subset of the original dataset by eliminating instances (usually negative class instances).
- **Hybrid method:** A combination of Oversampling and under-sampling techniques.

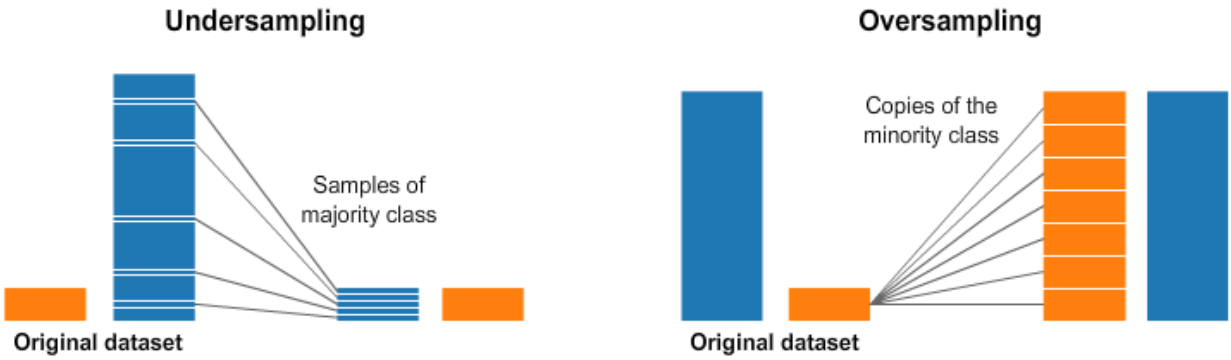

Figure 2 "Graphical example of Sampling Techniques." Source: [2]

There are many ways to implement the previous techniques, where the simplest preprocessing are non-heuristic methods like random under-sampling and random oversampling. Nevertheless, these techniques have some drawbacks. In the case of under-sampling, the major problem is that it can discard potentially valuable data that could be used in the training process, reducing our dataset's variability. On the other hand, our classifier can occur overfitting with random oversampling because it makes exact copies of existing instances.

To tackle the previous problems, more sophisticated methods have been proposed. The "Synthetic Minority Oversampling Technique" (SMOTE) has gained popularity among them. In short, its main idea is to overcome overfitting posed by random oversampling with the generation of new instances with the help of interpolating between the positive instances closer to each other.

The advantages and disadvantages of various sampling methods will be described in the following sections with the chessboard dataset proposed by [3]. The original data set is shown on the right side of Figure 3; it contains 1,000 data points and is 100%

balanced; hence, it has 500 examples per class. The borders and areas shown are done by a classification tree (specifically the CART algorithm). The CART algorithm is used to visualize the chessboard's shape and possible loss of information.

On the left side of Figure 3, we can constate the effects of the imbalanced learning problem as we can see that the shape of the graph is no longer a chessboard. This problem is because the class represented by white points (minority or positive class) is reduced to only 50 data points using a random sampling technique.

Concretely, the left dataset of Figure 3 will be the basis from which we will illustrate the behavior of other data-level techniques.

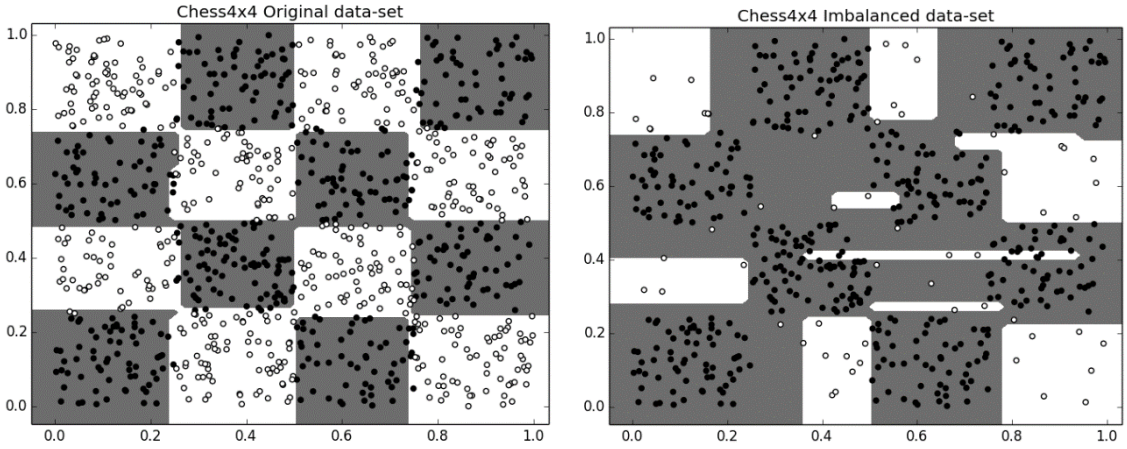

Figure 3 Chessboard Dataset. Source: [3]

2.2.1. Synthetic Minority Oversampling Technique (SMOTE)

As stated before, the problem of random oversampling is that because it replicates the exact copies of existing instances, no new information is added to the model's training; therefore, there is a high risk of overfitting. Here is where SMOTE comes to the rescue because instead of applying a simple replication of the minority class, the central idea of SMOTE is to generate new synthetic samples. This procedure focuses on the "feature space" rather than the "data space" since these new examples are created by interpolating several minority class instances closer. An example of this interpolation is shown in Figure 4.

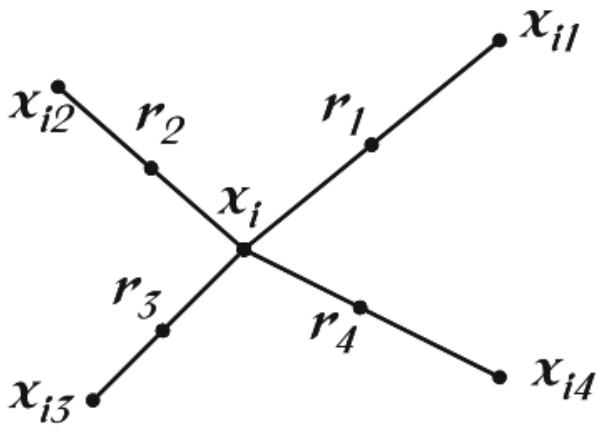

Figure 4 "Illustration on how to create the synthetic data points with SMOTE" Source: [1]

Concretely, in Figure 4, a minority class instance is denoted as x_i and is selected as a pivot point to create new synthetic data points. Then its nearest neighbors based on a distance metric are chosen from the training data set. In this example, these nearest neighbors are the points $x_{i1}\ to\ x_{i4}$ Lastly, a randomized interpolation is carried out to obtain new instances $r_1\ to\ r_4$.

Formally, the procedure works as follows. Initially, the total amount of oversampling N is set up. This integer value is often set to obtain a 1:1 class distribution. Next, an iterative process is executed that consists of the following steps:

1. A minority class instance is selected at random from the training set.
2. The K Nearest Neighbors (5 by the original paper) are obtained.
3. N of the K Nearest Neighbors of step 2 are randomly chosen to compute the new instances by interpolation. This process takes the difference between the feature vector (sample of step 1) and its K Nearest Neighbors. Finally, this difference is multiplied by a random number between 0 and 1 and added to the previous feature vector. The previous procedure creates a new point in the line segment between the features.

The previously described process is summarized in [1].

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE $N\%$; Number of nearest neighbors k
Output: $(N/100) * T$ synthetic minority class samples

(∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
if $N < 100$
  then Randomize the T minority class samples
    $T = (N/100) * T$
    $N = 100$
endif
$N = (int)(N/100)$ (∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
k = Number of nearest neighbors
$numattrs$ = Number of attributes
$Sample[\,][\,]$: array for original minority class samples
$newindex$: keeps a count of number of synthetic samples generated, initialized to 0
$Synthetic[\,][\,]$: array for synthetic samples
(∗ *Compute k nearest neighbors for each minority class sample only.* ∗)
for $i \leftarrow 1$ **to** T
    Compute k nearest neighbors for i, and save the indices in the $nnarray$
    Populate(N, i, $nnarray$)
endfor

Populate(N, i, $nnarray$) (∗ *Function to generate the synthetic samples.* ∗)
while $N \neq 0$
    Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
    for $attr \leftarrow 1$ **to** $numattrs$
      Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
      Compute: gap = random number between 0 and 1
      $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
    endfor
    $newindex{+}{+}$
    $N = N - 1$
endwhile
return (∗ *End of Populate.* ∗)
End of Pseudo-Code.

Algorithm 1 SMOTE. Source: [1]

In Figure 5, the imbalanced dataset is treated with SMOTE. We can see that the results are outstanding because the CART algorithm almost totally recovers the complete chessboard. Only three white squares are partially drawn, and all the black squares are

shown. However, the algorithm is not perfect, and many problems with the SMOTE algorithm can be shown in Figure 5. Specifically, we can assert the following problems:

1. SMOTE generates noisy points. We can identify these by the white points which fall into the black squares.
2. Many new synthetic points are generated in the same direction, following an artificial connecting line between diagonal squares. This problem produces borderline examples and thus complicates the modeling of decision surfaces.
3. Not so many white squares touch each other; this occurs because the minority class is distributed or oriented by a specific side of the square. Having the minority points on one side of the square causes the interpolations to create new synthetic points in the middle of the adjacent points. This creates the negative effect of keeping away when no points are closed in two adjacent diagonal squares. For this reason, there are three half-white squares modeled by CART.

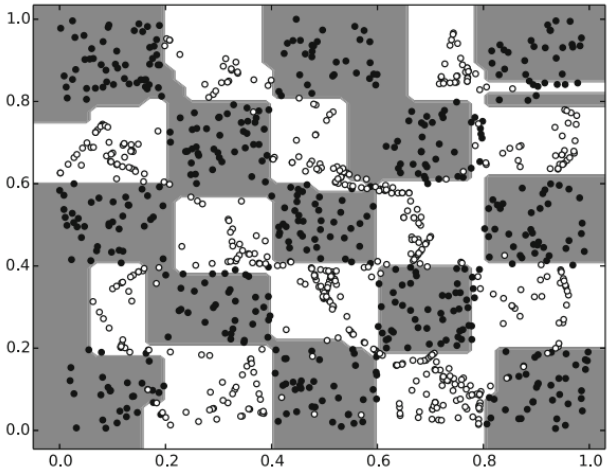

Figure 5 Imbalanced chessboard treated with SMOTE. Source: [3]

Influenced by the previously described drawbacks of SMOTE, many authors have proposed new improvements to the original SMOTE algorithm. The most relevant SMOTE extensions proposed in the literature will be described.

2.2.2. Borderline – SMOTE

Many Machine Learning algorithms like Logistic Regression and Support Vector Machines use the concept of decision boundary to decide whether an example belongs to one class or another. This decision boundary tries to learn the limits of each class as accurately as possible in the training process. Then when the decision boundary is set, if an example lies far away from it, there is a small probability that it will belong to the opposite class. It is as if the decision boundary divides the space into regions where each region belongs to one class; an example of this can be shown in Figure 6.

Based on the previous statement, the algorithms state that examples away from the borders may contribute little to classification. With this in mind, two new methods of oversampling minority class examples were proposed, Borderline-SMOTE1 and Borderline-SMOTE2, in which only the limited examples of the minority class will be oversampled (the ones close to the decision boundary). This method differs from the existing ones of oversampling, in which all minority examples or a random subset of the minority class are oversampled [2].

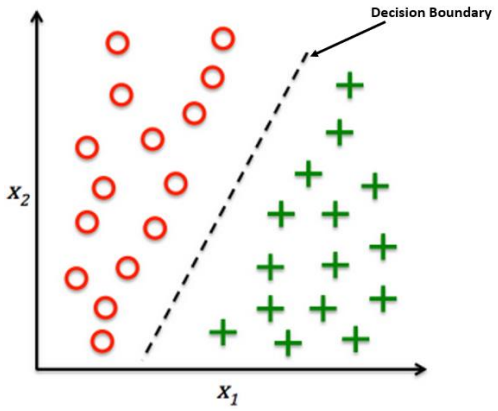

Figure 6 An example of a Decision Boundary between two classes

Let us define more formally the procedure of Borderline-SMOTE1. First, let us assume that TR denotes the whole training set, the set of the minority or positive class is P, and the set of majority examples is M. The number of minority and majority examples is denoted by N^+ and N^- respectively. Then we must observe the following steps:

1. For every p_i, $(i = 1, 2, \ldots, N^+)$ in P, we calculate its K Nearest Neighbors from the whole training set TR. The number of majority examples among the m Nearest Neighbors is denoted by m' $(0 \leq m' \leq m)$.
2. If $k' = k$, i.e., all the m nearest neighbors of p_i are majority examples, p_i is noise and is not operated in the following steps. If $m/2 \leq m' \leq m$, namely the number of p_i majority NNS is larger than the number of its minority ones, p_i is considered to be easily misclassified and put into a set $DANGER$. If $(0 \leq m' \leq m/2), p_i$ is safe and does not need to participate in the following steps.
3. The examples in $DANGER$ are the borderline data of the minority class P, and we can see that $DANGER \subset P$. We set $D = \{p'_1, p'_2, .., p'_D\}$, $0 \leq D \leq N^+)$. For each example in $DANGER$, We calculate its KNN from P.
4. In this last step, $D\ x\ s$ synthetic positive examples are generated from the data in $DANGER$, where s is an integer between 1 and k. For each p'_i, it randomly selects s Nearest Neighbors from its KNN in P. Firstly, it calculates the differences, dif_j, $(j = 1, 2, \ldots, s)$ between $p'_i\ and\ its\ s\ NNs\ from\ P$, then multiply dif_j by a random number r_j $(j = 1, 2, \ldots, s)$ between 0 and 1. In the end, s new synthetic minority examples are generated between p'_i and its Nearest Neighbors: $synthetic_j = p'_i + r_j \times dif_j$, $(j = 1, 2, \ldots, s)$.

The algorithm repeats the above mechanism for each p'_i in $DANGER$, and it can obtain $D\ x\ s$ synthetic examples.

The principal difference between the algorithm Borderline-SMOTE2 and Borderline-SMOTE1 is that the first one not only generates synthetic examples from each example in $DANGER$ and its positive NNs in P but also does that from its nearest negative neighbor in M. Then, the difference between it and its NNs is multiplied by a random number between 0 and 0.5; hence the new generated examples are closer to the minority class. Let us see how Borderline-SMOTE 1 performs in the chessboard dataset.

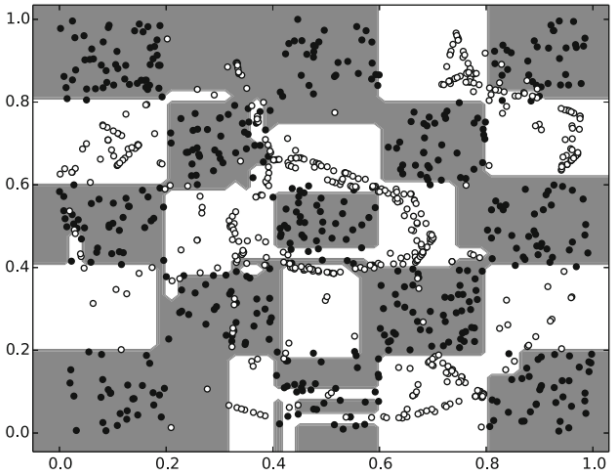

Figure 7 Imbalanced chessboard dataset treated with Borderline-SMOTE1. Source: [3]

In Figure 7, we can constate a distribution change of the synthetic white points generated concerning the original SMOTE. The newly generated points follow the lines across the diagonals of the squares, and some of them (like those in the center) form a circular shape. The denser an area, the higher the number of points generated around it. This demonstrates how some white squares located in external places have been skipped or reduced while learning the CART decision tree, producing less accurate chessboard modeling. It is also essential to state that the number of white dots generated in black squares is less than those generated by SMOTE.

2.2.3. Adjusting the Direction of the Synthetic Minority Class Examples: ADOMS

According to the authors [4], when exploring the SMOTE algorithm, the impact of the synthetic examples on the original feature space is restricted to the local space by interpolating the synthetic between the minor class example processing and one of its NNs. When the neighbor is distant from the center, which means that there are only a few examples in the local space near the center, and the actual underlying distribution of the class will be expressed as unreliable, the example of the synthetic minority class should be inserted in the space from the center to occupy the space comparatively better defined. In contrast, when the neighbor is near the center, there are already enough examples in the local space to finely express the underlying class distribution. The synthetic example must be inserted closer to the center to avoid disrupting the well-definite space. The previous generation mechanism of examples of synthetic minority classes sounds

reasonable; however, in SMOTE, only one of the center's neighbors is chosen randomly to represent the local space. The mechanism is performed only in one 1-dimension defined by the center and the selected neighbor. The mechanism must be performed in all space more appropriately.

In real-world data, the space can be isotropic, and the interpolation mechanism must be considered in all directions in the feature space. When a synthetic example is generated in the direction in which the projections of the local data are scarce, the projection of the synthetic must be distant from the center and; on the contrary, in the direction in which the projections of the local data are dense, the projection of the synthetic examples must be nearer to the center. [1]

The local space could be rebuilt using PCA. The maximum variance will be retained by the first axis of the main component, and the following axis of the main component will retain the largest of the remainder, and so on. Consequently, the synthetic example must be most distant from the center in the direction of the first axis of the main component. Thus, be closer in the direction in which its occupied variance is smaller, to the nearest to the center in the direction of the last axis of the main component. When the synthetic point is generated directly on the first axis of the main component through the center, its projection on the first axis of the main component must be itself and the most distant from the center. Then its projection will be closer to the center in which it has a minor variance. Hence, it is easy to see that the synthetic minority class example generation along the axis of the first significant component of the local data distribution would fit better with the interpolation mechanism. [1]

Thanks to the previous basis, the ADOMS algorithm can be expressed in the following steps:

1. Randomly select one of the minority class examples p_i in the original training data as the processing example.
2. Define the neighbor number k, (k = 1, 2, 3, . . .), and calculate the KNN of p_i in the feature space using the Euclidean distance.
3. Calculate the first principal component axis of local data distribution, which is composed of p_i and its k neighbors in the feature space.

4. Select one of its neighbors m_j randomly and calculate the Euclidean distance d between p_i and m_j, then scaling is obtained from Random(0, 1).
5. Generate a new synthetic minority class example p_i' in the feature space, where p_i' is generated along the direction from p_i to the projection of m_j on the first principal component axis through p_i and the Euclidean distance between p_i' and p_i is $scaling\ x\ d$.

2.2.4. ADASYN: Adaptive Synthetic Sampling Approach

ADASYN's [5] centered idea is to generate minority examples according to their distributions adaptively. In other words, more synthetic data is created for samples of minority classes that are more difficult to learn than samples of minorities that are easier to learn. The ADASYN approach can not only reduce the learning bias introduced by the original unbalance data distribution. However, it can also adaptively alter the decision boundary to concentrate on those samples that are challenging to learn. The ADASYN algorithm is defined down below.

Input of the ADASYN algorithm [1]: Training data set TR with N samples $\{x_i,\ y_i\}$, $i = 1, \dots, N$, where x_i is an instance in the m dimensional feature space M and $y_i \in \mathrm{C} = \{\mathrm{pos}, \mathrm{neg}\}$ is the class identity label associated with x_i. Define $N^+$and N^-as the number of minority class examples and the number of majority class examples, respectively. Therefore, $N^+ \leq N^-$and $N^+ + N^- = N$.

ADASYN's Procedure [1]

1. Calculate the degree of class imbalance as $d = N^+/N^-$, where $d \in (0, 1]$.
2. Set d_{th} a preset threshold for the maximum tolerated degree of class imbalance ratio. If $\mathrm{d} < d_{th}$:
    a. Calculate the number of synthetic data examples that need to be generated for the minority class as $G = (N^- - N^+) \times \beta$, where $\beta \in [0, 1]$ is a parameter that specifies the desired balance level after the synthetic data generation. β = 1 means a fully balanced data set is created after the generalization process

b. For each minority example p_i, find KNN based on the Euclidean distance in m dimensional space, and calculate the ratio r_i defined as $r_i = \frac{\Delta i}{k}$, $i = 1, \ldots, N^+$, where Δi is the number of examples in the KNN of p_i that belong to the majority class, therefore $r_i \in [0, 1]$.

c. Normalize r_i according to $\hat{r}_i = r_i / \sum_{i=1}^{N^+} r_i$, so that $\hat{r}_i$ is a density distribution (if $\sum_{i=1}^{N^+} r_i = 1$).

d. Calculate the number of synthetic data examples that need to be generated for each minority example p_i as $p_i = \hat{r}_i \times G$, where G is the total number of synthetic data examples that need to be generated for the minority class as defined in step 2a.

e. For each minority class data example p_i, generate g_i synthetic data examples according to the next loop. Make the Loop from 1 to g_i:

   i. Randomly choose one minority data example, p_{zi}, from the KNN for data p_i.

   ii. Generate the synthetic data example s_i with $s_i = p_i + (p_{zi} - p_i) \times \Lambda$, where $(p_{zi} - p_i)$ is the difference vector in the m dimensional space, and Λ is a random number $\Lambda \in [0, 1]$.

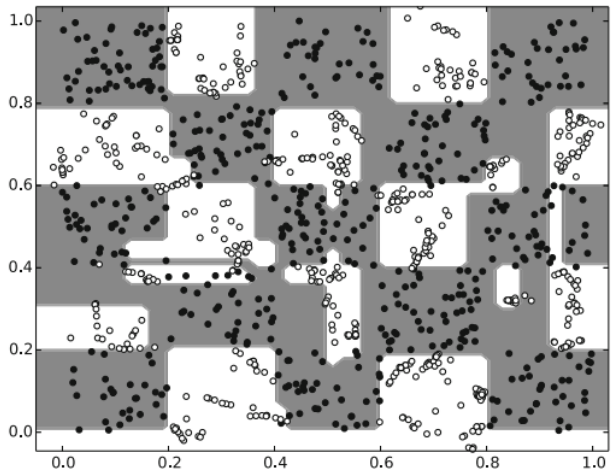

Figure 8 Imbalanced chessboard treated with ADASYN. Source: [3]

We can analyze in Figure 8 the result of applying ADASYN as described in the algorithm aforementioned on the imbalanced chessboard dataset. In this example, we

can appreciate that the synthetic points are generated with a more appropriate distribution. The first result that we can view is that the chessboard is almost wholly restored; nonetheless, the CART algorithm cannot connect all the white squares. Also, we can state that the behavior mentioned above of generating new points following the diagonals of connected squares is not present. A negative result is that some white points are created within the black squares, and on the other hand is evident to see the significant separation between the points of different classes.

2.2.5. ROSE: Random Oversampling Examples

ROSE supplies a unified framework for model estimation and accuracy evaluation in imbalanced learning. It achieves this, creating new artificial synthetic examples from the classes with a smoothed bootstrap strategy. [5]

Let us explore this new paradigm. First, let TR be our training data set with N samples x_i, y_i, $i = 1, \dots, N$, where x_i is an instance in the m dimensional feature space M and $y_i \in \mathrm{C} = \mathrm{Y}_0, \mathrm{Y}_1$. x_i are some related attribute supposed to be realizations of a random vector x defined on R^d with an unknown probability density function $f(x)$. Let N_j be the number of examples belonging to the class Y_j. ROSE consists of the following steps:

1. Select $y* = Y_j$ with probability π_j .
2. Select $\{x_i, y_i\} \epsilon\, TR$, such that $y_i = y*, with\ probability\ 1/N_j$.
3. Sample $x*$ from $K_{\boldsymbol{H}j}(\cdot, x_i)$, (·, xi), with $K_{\boldsymbol{H}j}$ a probability distribution centered at x_i and covariance matrix $\boldsymbol{H}_j$.

In other words, what ROSE does is that it draws from the training set an observation that belongs to one of the two classes and then generates a new example $x*, y*$ in its vicinity, where the shape of the neighborhood is determined by the shape of the contour sets of K and its width governed by $\boldsymbol{H}_j$.It is essential to mention that the generation of new examples from $\boldsymbol{Y}_j$, corresponds to the generation of data from the kernel density estimate of $\boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{Y}_j)$, with kernel K and smoothing matrix $\boldsymbol{H}_k$. The choices of $K\ and\ \boldsymbol{H}_k$ are solved by the literature on kernel density estimation.

The repetition of steps from 1 to 3 allows for generating as many new instances as wanted. This recurrence changes the value of the probability $\boldsymbol{\pi}_j$ associated with the class choice and allows the new data set to be balanced.

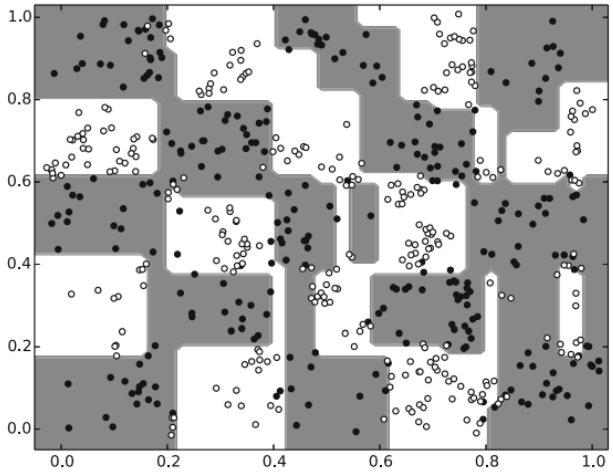

Figure 9 Imbalanced chessboard with ROSE (original points not included). Source: [3]

Figure 9 values of the optional shrink factor multiplied by the smoothing parameters to estimate the conditional kernel density of the majority and minority classes were set to 0.15. Also, this figure only shows the artificially generated points for both classes. We can see in Figure 9 that it is an excellent approximation of what ROSE does. This new data will allow us to sample new data from many possible configurations depending on the later DM algorithm.

2.3. Active Learning

Active Learning is a set of techniques that seeks to sample the most critical data for humans to review. The problem that Active Learning aims to solve is: if we have a massive dataset with unlabeled data, how do we choose which unlabeled data to label first? In other words, what unlabeled data will increase my algorithm performance once I label it? At first, this problem may sound disconnected from the imbalanced class problem. However, in our case, the question is: What are the points from the minority class that we need to choose first to generate the synthetic samples so we can finally have good model performance? How do we sample the most critical data for humans to review? To answer this, we have the following techniques:

1. Uncertainty Sampling: This type of sampling is based on the uncertainty of a model about a sample. There are many ways of measuring uncertainty, like least, margin or ratio of confidence, and entropy. All of this will be explained later.
2. Diversity Sampling: This type of sampling tackles the problem of identifying where the model might be confident but wrong due to undersampled or nonrepresentative data. It is based on various data sampling approaches helpful in identifying gaps in the model's knowledge, such as clustering, representative sampling, and methods that identify and reduce real-world bias in the models. Collectively, these techniques are known as diversity sampling.

2.3.1. Uncertainty Sampling

As we stated, the most straightforward strategy people use to make AI models smarter is for the machine learning models to tell humans when they are uncertain about a task. That means unlabeled data that confuses an algorithm is the most valuable when labeled and added to the training data. If the algorithm can already label an item with high confidence, it is probably correct. Nevertheless, it is not always easy to know when a model is uncertain and how to calculate that uncertainty. The different approaches to measuring uncertainty can produce vastly different results. Uncertainty sampling is the set of techniques for identifying unlabeled items near a decision boundary. We are going to explain the four approaches to uncertainty sampling with Figure 10 :


```
{
    "Object": {
        "Label": "Hypertension",
        "Scores": {
            "Hypertension": 0.91,
            "Hypotension": 0.015,
            "Aritmia": 0.05,
            "Healthy": 0.025
        }
    }
}
```


Figure 10 "JSON-encoded example of a prediction"

- Margin of confidence sampling: Difference between the two most confident predictions. In our example, if the model is most confident that the patient has Hypertension and second most confident that the patient has Aritmia, the margin of confidence is: 0.91 – 0.05 = 0.86.
- Least confidence sampling: The difference between the most confident prediction and 100% confidence. In our example, the model is most confident that the patient has Hypertension; thus: 100 – 0.91 = 0.99.
- Ratio of confidence sampling: Ratio between the two most confident predictions. In our example, the model is most confident that the patient has Hypertension and second most confident that the patient has Aritmia; thus, the ratio of confidence is 0.91 / 0.05 = 18.2.
- Entropy-based sampling: Difference between all predictions, as defined by information theory. In our example, entropy-based sampling would capture how much every confidence differed from every other. In the binary classification problem, entropy base sampling is the same as the margin of confidence, which is why it is not explained here.

How do we measure uncertainty for different types of machine learning models? That is how we got the score shown in Figure 10. Almost every machine learning library will return some scores for the algorithms, which can be used for uncertainty sampling. Sometimes these scores can be used directly, but in other cases, it is necessary to convert the scores to probability distributions using something like softmax. An example of determining uncertainty for different types of machine learning algorithms is shown in Figure 11.

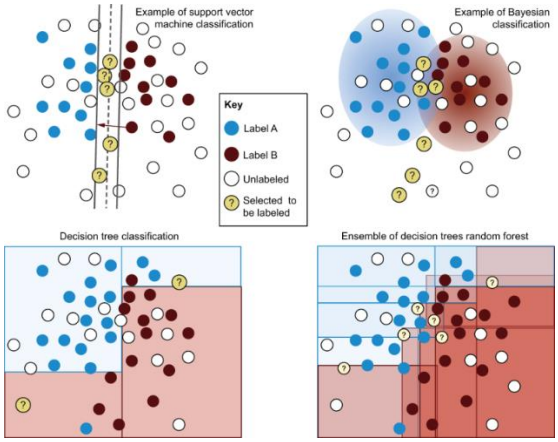

Figure 11 "Uncertainty sampling from different ML algorithms" Source: [6]

2.3.2. Diversity Sampling

In simple words, uncertainty sampling is identified where the model is uncertain: what the model "knows it does not know." By contrast, diversity sampling is to identify what is missing from the model: what the model "does not know that it does not know." This problem is more complex than the one of uncertainty sampling because what the model needs to know is a moving target in a constantly changing world.

Diversity is essential in medical data because it can also be used to ensure demographic diversity. It is not the topic of this work, but we want to mention that almost all the datasets are biased toward a specific gender, race, and socioeconomic background. Generally, this biased is toward the most privileged demographics: diseases from the wealthiest nations, patients from the wealthiest economies, and other biases that result from power imbalances. If the models are built only on randomly sampled raw data, or in our case, the synthetic data is generated randomly; these biases will amplify. We believe that applying diversity sampling with SMOTE in medical datasets will likely increase the diversity of the people who can benefit from models built from that data.

The hardness of the problem that diversity sampling aims to solve makes the solutions more algorithmically diverse than those for uncertainty sampling. In this work, we only explore cluster-based sampling. However, we are going to explain the following four approaches to diversity sampling:

- Cluster-based sampling: Using cluster methods independent of the model, we can identify natural trends in the data to ensure that we will not miss rare but meaningful trends.
- Model-based outlier sampling: Determine which items are unknown to the model in the current state or, in other words, items that the model has not encountered before.
- Representative sampling: Finding a sample of unlabeled items that looks most like the target domain, compared with the training data. For example, creating a machine learning algorithm that will treat persons from 18 to 50 years. We will be interested in sampling from that age range.
- Sampling for real-world diversity: Perhaps the hardest to accomplish is to ensure that a diverse range of real-world entities is in the training data to reduce real-world bias. In the medical example, this could include targeting patients with as many ages, places, and ethnicities as possible.

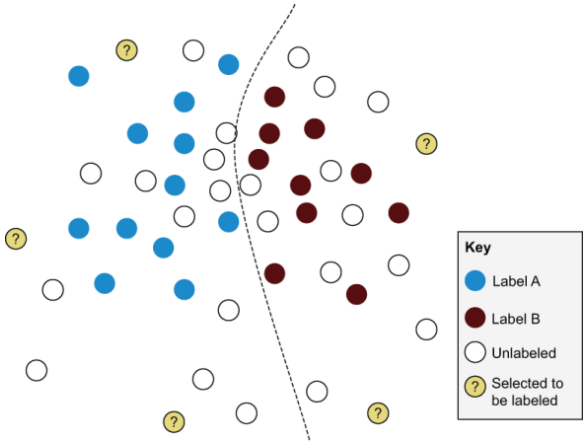

Figure 12 "Example of Diversity Sampling" Source: [6]

In Figure 12, we can see that diversity sampling selects maximally different samples from the existing training items and one another. By contrast, with uncertainty sampling, we want to see only what is near the current decision boundary; this is a relatively small and well-defined feature space.

2.3.3. Combining Uncertainty and Diversity Sampling

One of the easiest and most common ways that uncertainty sampling and diversity sampling are used in industry is to take a large sample from one method and then reduce that sample with another. For example, sampling 50% of the most uncertain items with an uncertainty sampling measure and then applied cluster-based sampling to sample 10% of those items. An example of this is shown in Figure 13, where the centroids from each cluster are sampled. Alternatively, random members of the cluster could be chosen.

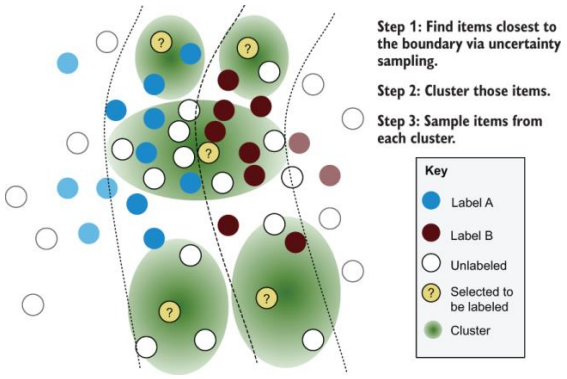

Figure 13 "Least-Confidence combine with clustering-based sampling" Source: [6]

It is essential to mention that the simple method from Figure 13 has vanquished many other complex ones regarding performance. However, the simple methods of combining strategies rarely make it into academic papers; academia favors papers that combine methods into a single algorithm rather than chaining multiple simple algorithms. However, in real-world applications, simple methods win because it is easier to maintain them in the long run.

Chapter 3 Experimentation

3.1. Active Learning with SMOTE, algorithm description

As stated in the introduction, this work aims to show that it is possible to have good model performance without generating too many synthetic samples. We want to achieve this by changing the first step of SMOTE with Active Learning techniques. In other words, instead of choosing a point at random from the training set to use as the pivot point to generate the synthetic samples, we will choose the points intelligently with uncertainty and diversity sampling. Concretely, this algorithm is:

ALGORITHM : SMOTE WITH ACTIVE LEARNING

function SMOTE_AL(X_min, X_maj, epochs, perc_uncertain, perc_diversity, uncertain_measure, n_kmeans)
Input: X_min; epochs; perc_uncertain; perc_sample; uncertain_measure; n_kmeans
X_min: Minority class samples.
X_maj: Majority class samples.
epochs: Number of iterations.
perc_uncertain: % of most uncertain samples from T to retain (uncertainty sampling).
perc_diversity: % of samples from uncertainty sampling that are going to pass to diversity sampling.
uncertain_measure: Uncertainty measure to use to calculate confidence (margin and least confidence)
n_kmeans: Number of clusters to be generated with kmeans in diversity sampling (diversity sampling).

Output: model (model train with an iterative active learning strategy, thus with many X_smote samples)

1. **for** i ← 1 to epochs:

2. # First do Uncertainty Sampling
   if uncertain_measure = "margin confidence"
   X_uncertain = Compute the model's confidence with every sample of X_min with margin confidence.
   X_uncertain = X_uncertain sort by descending order.
   columns_to_retain = X_uncertain * perc_uncertain.
   X_uncertain = Retain the first columns_to_retain from X_uncertain.
   else if uncertain_measure = "least confidence"
   X_uncertain = Compute the model's confidence with every sample of T with least confidence.
   X_uncertain = X_uncertain sort by descending order.
   columns_to_retain = X_uncertain * perc_uncertain.
   X_uncertain = Retain the first columns_to_retain from X_uncertain.

3. # Second do Diversity Sampling
   X_diversity = Cluster X_uncertain with Kmeans obtaining n_kmeans clusters.
   X_diversity = Do cluster sampling of X_diversity until size (X_diversity * perc_diversity) is reach.

4. # Third generate one synthetic sample from every diversity sample with the original SMOTE.
   X_smote = SMOTE(X_diversity, N=100, k=5)

5. # Fourth retrain the ML model with (X_min + X_smote + X_maj)
   model = retrain the model with the training set composed of (X_min + X_smote + X_maj)
   X_min = X_min + X_smote

return model

Algorithm 2 "SMOTE with Active Learning Algorithm"

3.2. Experiment description

3.2.1. Machine Learning models evaluated

In order to evaluate the performance of SMOTE with Active Learning, we train the following ML algorithms:

1. Logistic Regression
2. Support Vector Classifier
3. Gradient Boost

4. Ada Boost
5. Random Forest

The training procedure was to do a random search with ten folds of cross-validation from the hyperparameter space described in Figure 14. The model that scored best in the average of f1-score-weighted k-validation sets was chosen.

```
# Logistic Regression
{'C': np.linspace(0.001,2,500)}
# Support Vector Classifier
{'kernel': ['rbf'],
'gamma': [ 0.001, 0.01, 0.1, 1],
'C': [1, 10, 50, 100, 200,300, 1000]}
# Gradient Boosting
{'n_estimators' : [10, 20, 50],
'learning_rate': [0.01, 0.075, 0.005],
'max_depth': [1, 2, 3, 4, 5],
'min_samples_leaf': [1, 5, 10,20, 50, 70],
'min_samples_split': [20, 30, 50, 100, 150],
'max_features': [ 0.3, 0.2, 0.1] }
# Ada Boost
{ "n_estimators": np.arange(10, 5000, 10),
"learning_rate": np.linspace(0.001, 1, 300),
"base_estimator":[dtc, log_reg],
"algorithm": ["SAMME", "SAMME.R"]}
# Random Forest
{"max_depth":[5,10,20],
"n_estimators":[i for i in range(10,100,10)],
"min_samples_leaf":[i for i in range(1,10)],
"criterion":["gini","entropy"],
"max_features":["auto","sqrt","log2"]}
```

Figure 14 "Hyperparameter space for the models"

3.2.2. Sampling techniques evaluated

The training data used was generated from the following two methods:

Classical Methods

The classical methods are SMOTE variants, simple random undersampling, and oversampling. All these techniques were used to make a balanced dataset, except with UNBAL. The abbreviations are the following:

1. UNDER: Randomly undersample the majority class.
2. OVER: Random oversampling of the minority class.
3. SMOTE: Create synthetic samples of the minority class with the original SMOTE.
4. SVM: Generate synthetic samples of the minority class with SVM SMOTE.
5. BORDER: Create synthetic samples of the minority class with Borderline SMOTE.
6. ADASYN: Generate synthetic samples of the minority class with Adasyn SMOTE.
7. UNBAL: Do not make any changes to the original training dataset. That is, keeping the classes unbalanced.

New methods

Random (simple SMOTE) is used to evaluate if randomly choosing the pivot points has the same or better results as Active Learning with SMOTE.

1. Random (simple SMOTE): In each iteration, randomly generate N synthetic samples with the original SMOTE.
2. Diversity & Uncertainty – margin: In each iteration, generate N synthetic samples with the technique described in Algorithm 2. In uncertainty sampling, margin confidence is used as an uncertainty measure.
3. Diversity & Uncertainty – least: In each iteration, generate N synthetic samples with the technique described in Algorithm 2. In uncertainty sampling, least confidence is used as an uncertainty measure.
4. Uncertainty – margin: In each iteration, generate N synthetic samples with the technique described in Algorithm 2, only doing uncertainty sampling with margin confidence.

3.3. PIMA Indians Diabetes Dataset

3.3.1. EDA PIMA

3.3.1.1. Dataset Description

The target is to predict whether or not a patient will have diabetes based on specific diagnostic measurements included in the dataset. The features and target of the dataset are the following:

- **Pregnancies:** indicates the number of pregnancies.
- **Glucose:** indicates the plasma glucose concentration
- **Blood Pressure:** indicates diastolic blood pressure in mm/Hg
- **Skin Thickness**: indicates triceps skinfold thickness in mm
- **Insulin:** indicates insulin in U/mL
- **BMI:** indicates the body mass index in kg/m2
- **Diabetes Pedigree Function:** indicates the function which scores the likelihood of diabetes based on family history
- **Age:** indicates the age of the person
- **Outcome:** indicates if the patient had diabetes or not (1 = yes, 0 = no)
-

3.3.1.2. Class Imbalance

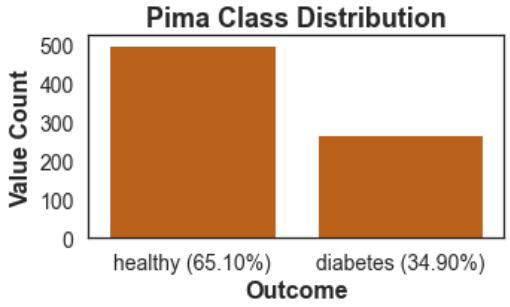

Figure 15 "Data Distribution of PIMA dataset"

As shown in Figure 15, the data from the PIMA dataset is imbalanced because there are more healthy examples in the dataset. Thus, the minority class is the persons who have diabetes.

3.3.1.3. Pair plot

Figure 16 "Pairplot PIMA"

From Figure 16, we can state the following:

- Some features have values that make no sense, for example, the 0's values in BMI, Glucose, Insulin, and Skin Thickness.
- There is a positive correlation between Skin Thickness and BMI.
- Skin Thickness has two groups because this feature shows two normal distributions.

- We can see that the variable Glucose can separate linearly those who have diabetes and those who do not.

3.3.1.4. Feature's Correlation

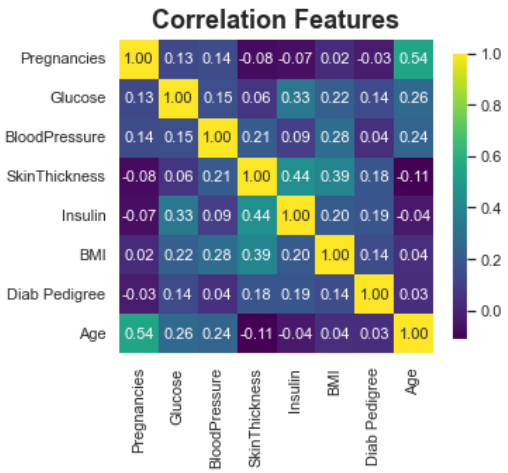

Figure 17 "Correlation between Features PIMA"

From Figure 17, we can see no high correlation between the features. That is why we can not reduce the dimensionality of the dataset and avoid the curse of dimensionality.

3.3.1.5. NA Values

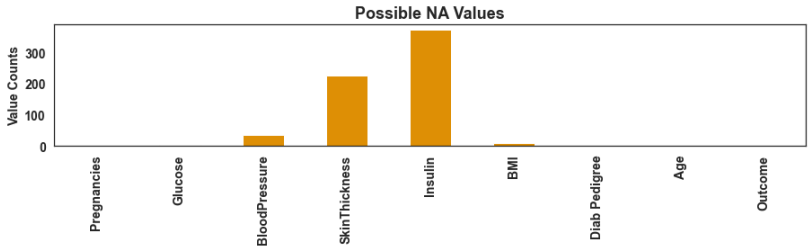

Figure 18 "Possible NA Values PIMA"

In Figure 18, we said that those are possible NA values because, in the PIMA dataset, these features have a value of 0. However, it does not make sense that these values have a value of 0. As an illustration, if someone has a blood pressure of 0, it is death.

Many academic papers take those values as 0, even though those are unrealistic data points.

3.3.1.6. Features Distribution

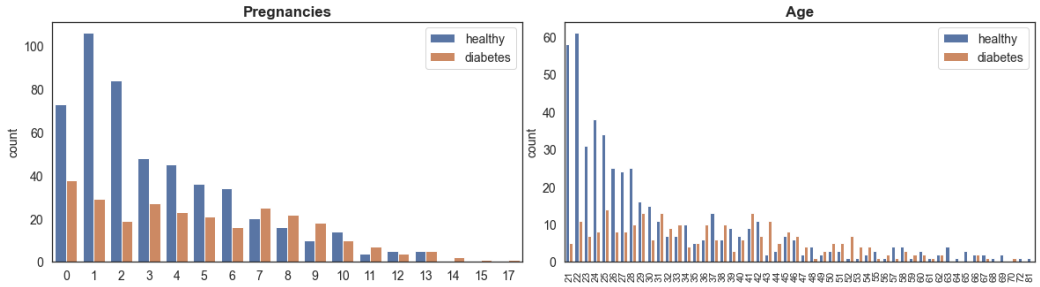

Figure 19 "Distribution of Diabetes between Age and Pregnancies"

From Figure 19, we can state that there is a different proportion of diabetes in the age and pregnancy groups.

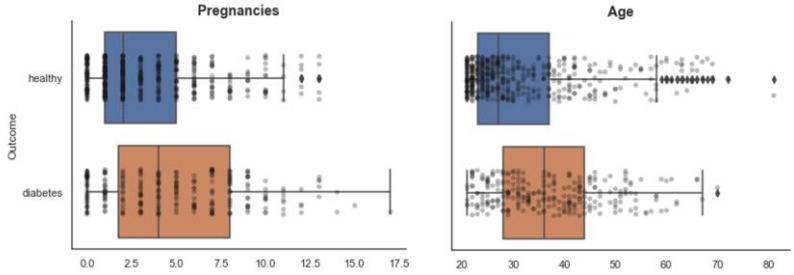

Figure 20 "Distribution of Age and Pregnancies"

From Figure 20, we can see that diabetes is widespread in all ages and healthy is found in younger individuals. Persons who have diabetes tend to have more pregnancies.

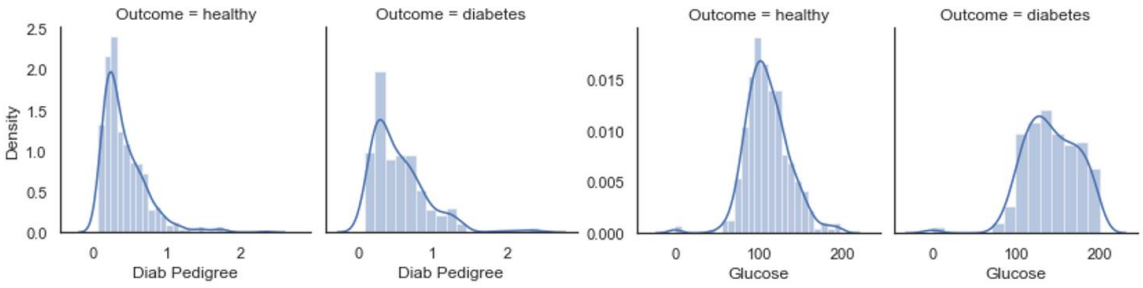

Figure 21 "Density Plot Glucose and Diab Pedigree"

In Figure 21, we can see that patients with diabetes have higher glucose than healthy patients.

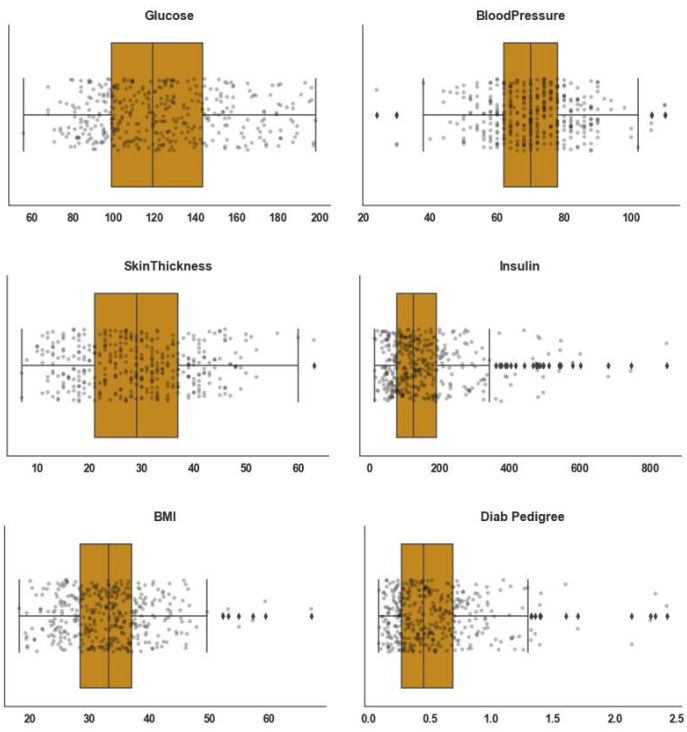

Figure 22 "Boxplots Pima features"

We can see from Figure 22 that our data tends to be in a normal distribution. There are some outliers; however, we spoke to specialists in diabetes, and those values are valid for a patient. For this reason, we will keep those values in our dataset.

3.3.2. Preprocessing and Feature Engineering

We decided to make an educated guess in the NA values from Figure 18 and set them as the median of that feature. The code of this can be seen in Figure 23.

```
df['Glucose'].fillna(df['Glucose'].median(),inplace=True)
df['BloodPressure'].fillna(df['BloodPressure'].median(),inplace=True)
df['SkinThickness'].fillna(df['SkinThickness'].median(),inplace=True)
df['Insulin'].fillna(df['Insulin'].median(),inplace=True)
df['BMI'].fillna(df['BMI'].median(),inplace=True)
```

Figure 23 "Educated Guess PIMA"

Also, we scale our values with standard scaler of sklearn so we can speed up learning, calculate well distances with Kmeans and Knn, and apply regularization in training. The code of this can be seen in Figure 24.

```
# Initiate Standard Scalaer
scaler = StandardScaler()

# Separate target and variables
X = df.iloc[:, 0:8]

# Scale variables
X_scaled = scaler.fit_transform(X)
```

Figure 24 "Standard Scaler PIMA dataset"

Also, stratified sampling was done to split into training and test set.

3.3.3. Training results PIMA Active Learning with SMOTE and other SMOTE techniques

Note: All the experiments below generate ten new synthetic samples in every epoch in every method. In epoch 18th, the dataset is balanced.

3.3.3.1. Logistic Regression – PIMA

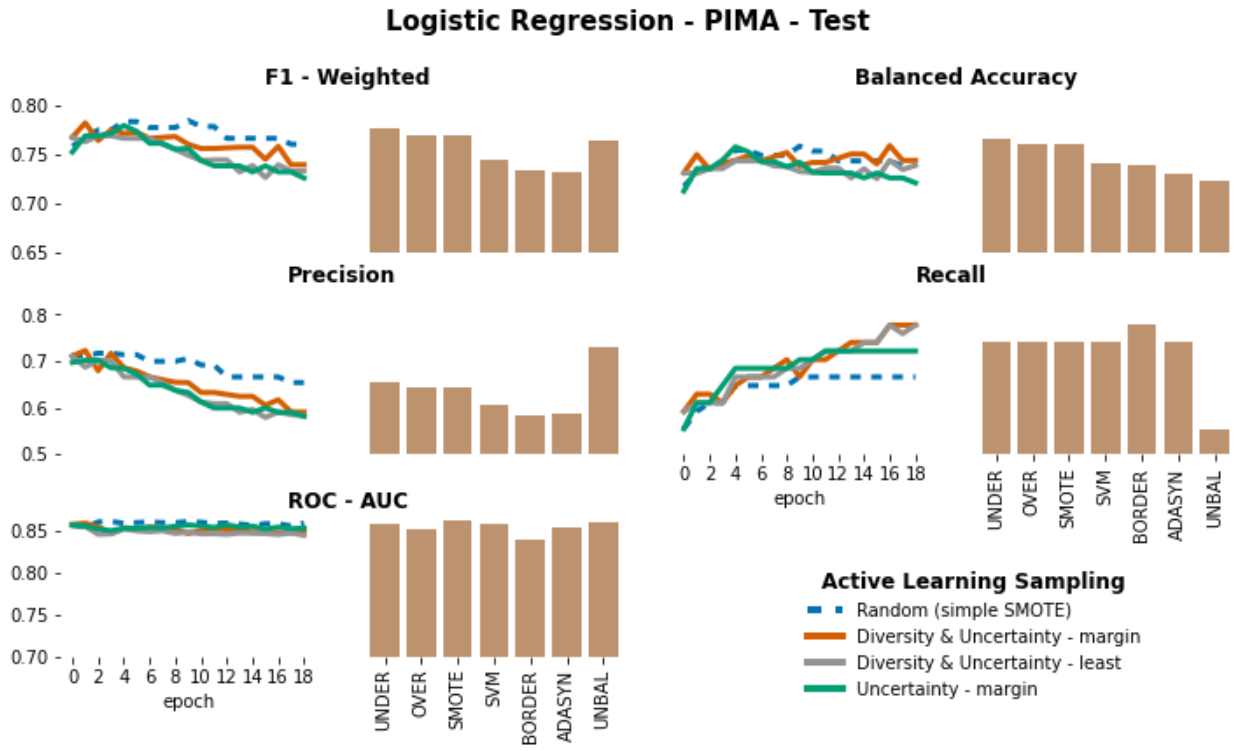

Figure 25 "Active Learning SMOTE vs. Classical Methods with LogReg in PIMA"

Remarks Figure 25:

1. The methods of AL with SMOTE achieve more outstanding performance in recall than the classical ones and are better than simple iterative SMOTE.
2. The better performance of the minority class comes at the expense of the majority class. That is why there is a slight decrease in Precision, F1-Weighted and Balanced Accuracy.
3. AL with SMOTE achieved a similar performance in recall as the classical methods at around iteration 12. That means that with only 120 synthetic samples, the AL with SMOTE achieved equivalent performance to the classical methods with 186 synthetic samples.

3.3.3.2. Support Vector Classifier - PIMA

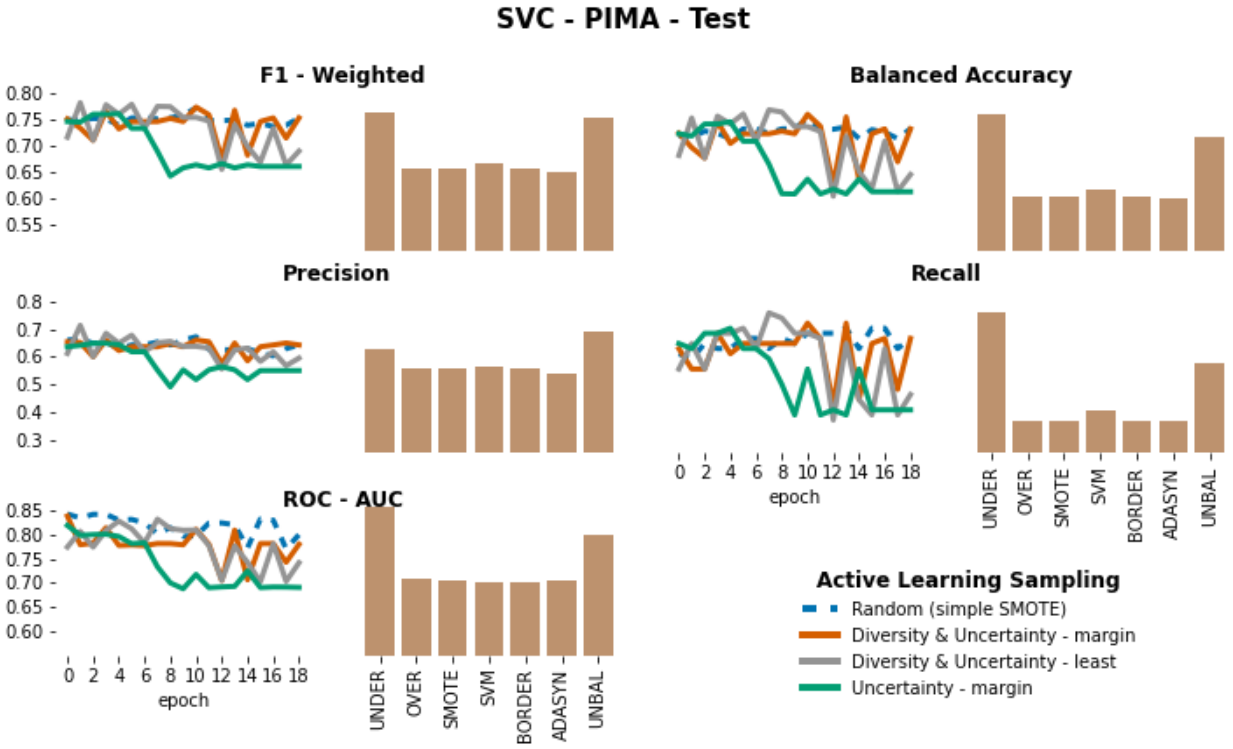

Figure 26 "Active Learning SMOTE vs. Classical Methods with SVC in PIMA"

Remarks Figure 26:

1. There is considerable variability with all the methods when using SVC.
2. There is a decrease in recall as epochs pass. Our intuition is that we need to augment the diversity sampling cluster size to make a more diverse sampling because the model generates the synthetic samples in a single region. However,

we can not say that the new methods are ineffective, as the Classical Methods are also not performing well.

3. The undersampling technique achieves the best results in all the metrics. Our intuition is that the hyperspace where the new synthetic samples are generated is wrong.
4. In this example, we can see one of the advantages of Active Learning with SMOTE. If we see that the model is doing a lousy performance as epochs pass, we can ask an expert to verify if our synthetic samples are ok.

3.3.3.3. Gradient Boost – PIMA

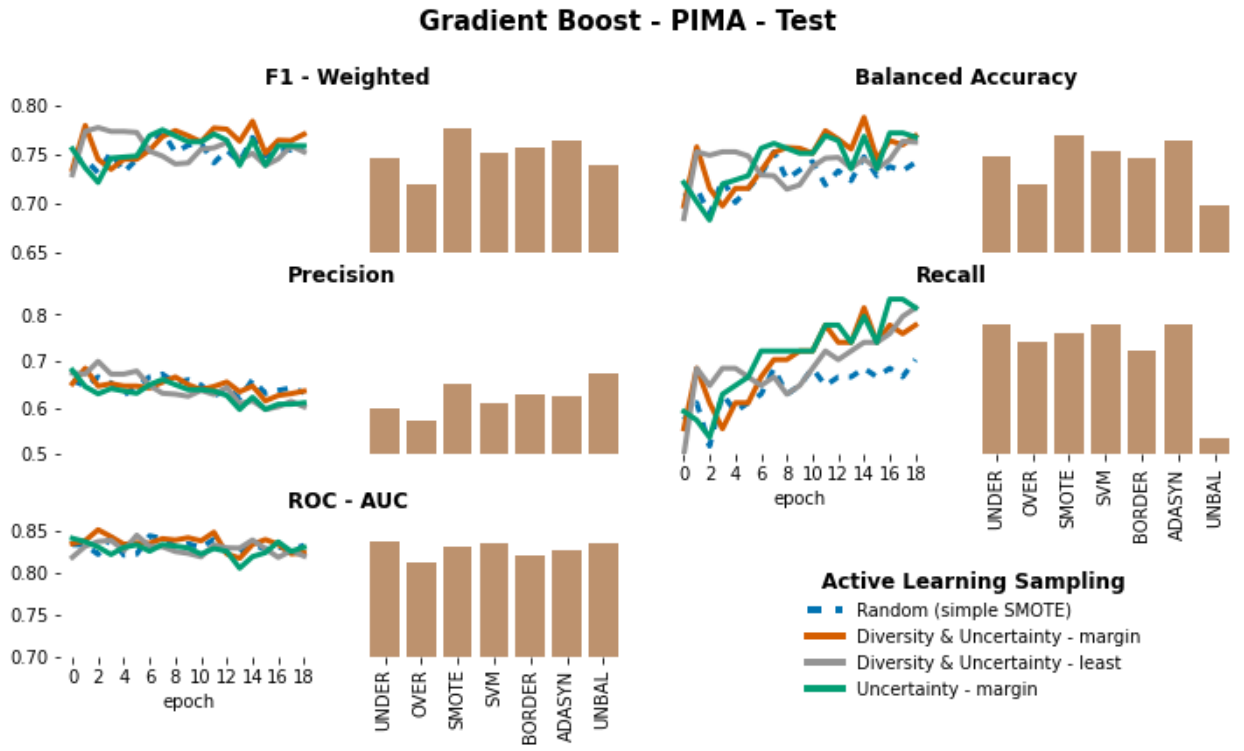

Figure 27 "Active Learning SMOTE vs. Classical Methods with Gradient Boost in PIMA"

Remarks Figure 27:

1. The behavior of these graphs is very similar to Logistic Regression.
2. There is a considerable increase in recall and not a big decrement in precision, which is good.
3. As with Logistic Regression, AL with SMOTE can achieve similar results as Classical Methods at around epoch 12.

4. ROC – AUC does not change much as epochs pass.

3.3.3.4. AdaBoost – PIMA

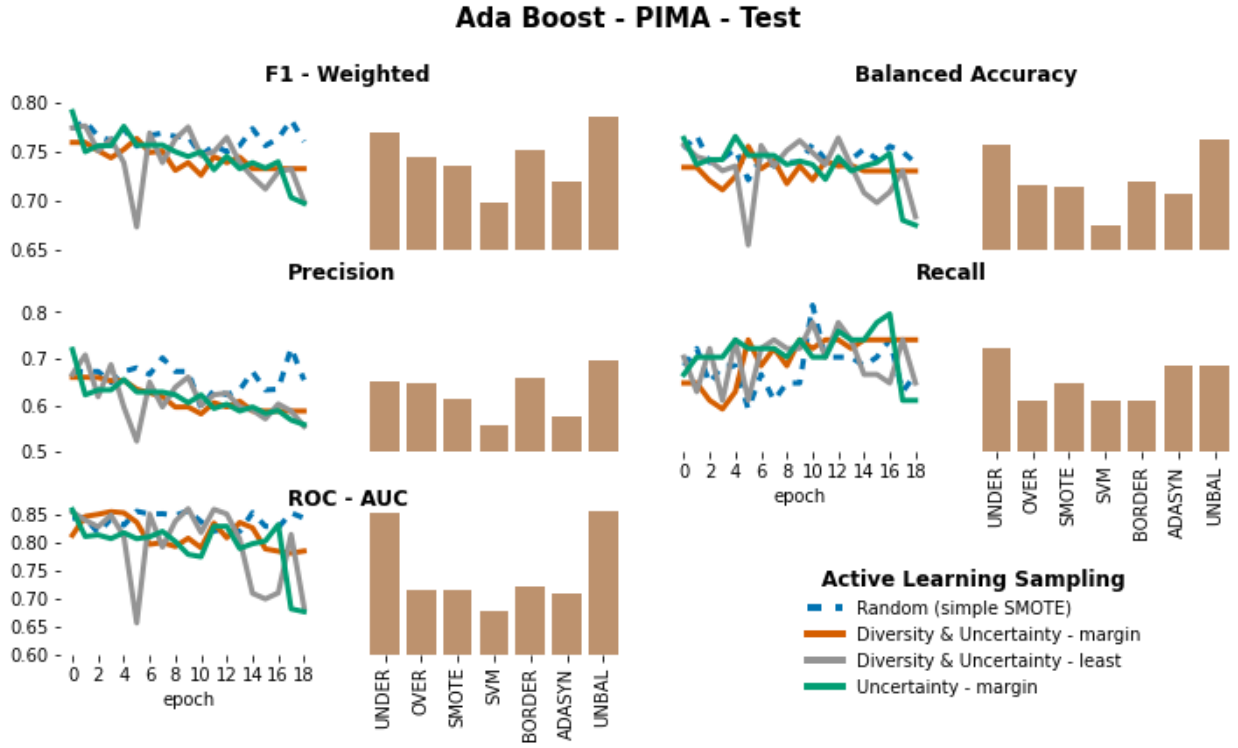

Figure 28 "Active Learning SMOTE vs. Classical Methods with AdaBoost in PIMA"

Remarks Figure 28:

1. There is a high variability with all the methods.
2. It cannot be said that the model is getting better performance as epochs pass.
3. The unbalanced dataset has equal or better performance than all the other sampling techniques, which means that Ada Boost is not getting better with more synthetic samples for this case.
4. It can not be stated that AL with SMOTE performs worse with AdaBoost in this case because all the other methods performed the same way.

3.3.3.5. Random Forest – PIMA

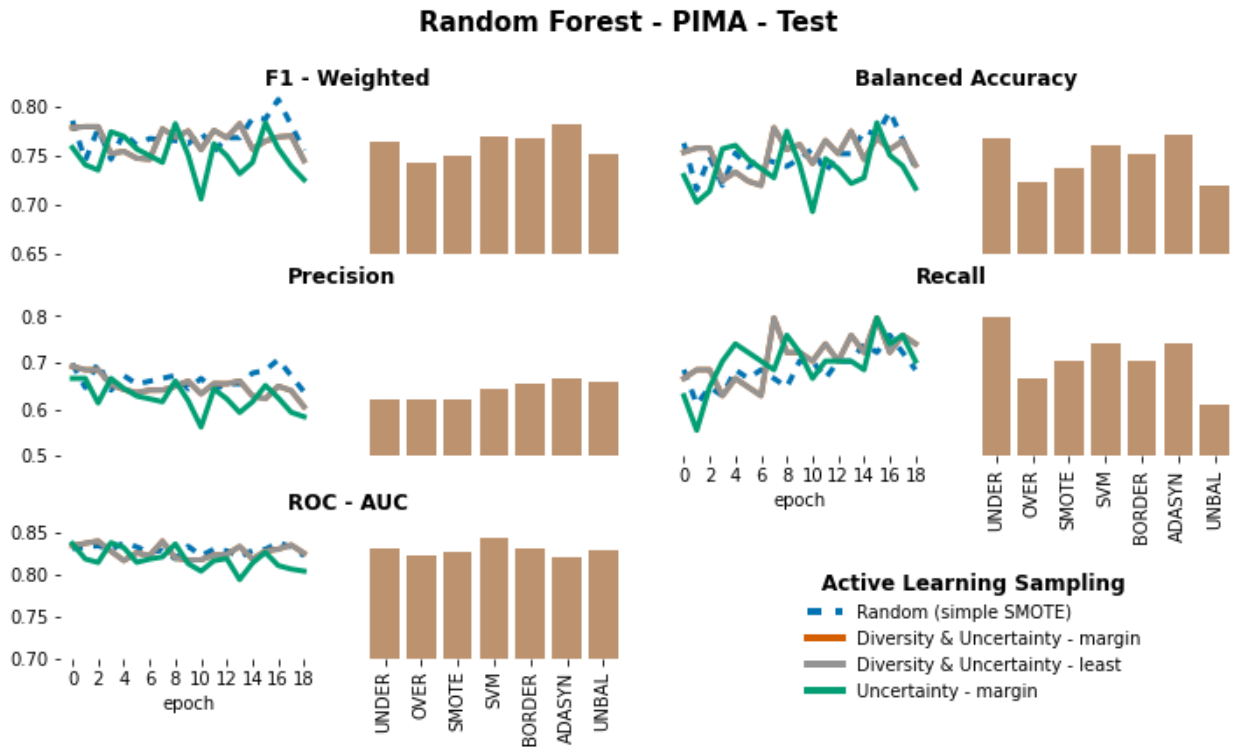

Figure 29 "Active Learning SMOTE vs. Classical Methods with Random Forest in PIMA"

Remarks Figure 29:

1. Undersampling has equal or similar metrics as all other sampling techniques.
2. There is a similar performance of Active Learning and all other techniques. However, the evolution of the recall metric as epochs pass reinforces the idea that we do not need to generate synthetic samples to have a complete class balance.
3. Precision is not heavily penalized as recall increases

3.4. Breast Cancer Wisconsin Dataset

3.4.1. EDA Breast Cancer

3.4.1.1. Dataset Description

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe the characteristics of the cell nuclei present in the image.

The separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method that uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features.

The linear program used to obtain the separating plane in the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets," Optimization Methods and Software 1, 1992, 23-34].

Attribute Information:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field13 is Radius SE, field 23 is Worst Radius. All feature values are recoded with four significant digits.

3.4.1.2. Class Imbalance

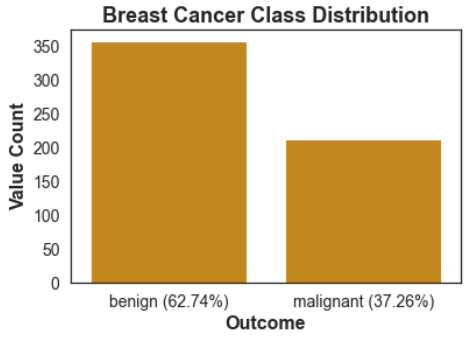

Figure 30 "Class Imbalance from Breast Cancer Dataset"

- This dataset has no missing values.

No further exploratory analysis was done with this dataset because all the features come from an experiment, and no outliers or weird values have been detected in the literature.

3.4.2. Preprocessing and Feature Engineering Breast Cancer

The only preprocessing done for training was to scale the features with Standard Scaler of sklearn, as shown in Figure 24. Also, stratified sampling was done to split into training and test set.

3.4.3. Training results from Breast Cancer Active Learning with SMOTE and other SMOTE techniques

One general comment on the following results is that we believe that the dataset Breast Cancer is not very difficult for an ML model to learn. That is why all the models and techniques have a good performance. Also, other papers showed similar outcomes. We thought these results could be caused by data leakage from the test set; however, we searched exhaustively, and evidence of such leakage was not found.

We chose to leave the results in this dataset to show that sometimes it is not necessary to balance the classes to obtain good model performance. Also, to demonstrate that our new method is not counter-productive.

Also, six samples are generated for every epoch in the Active Learning iterations. Thus in 20 epochs, the classes are balanced. We made more epochs to see what happens when more epochs are made after the dataset is balanced.

3.4.3.1. Logistic Regression – Breast Cancer

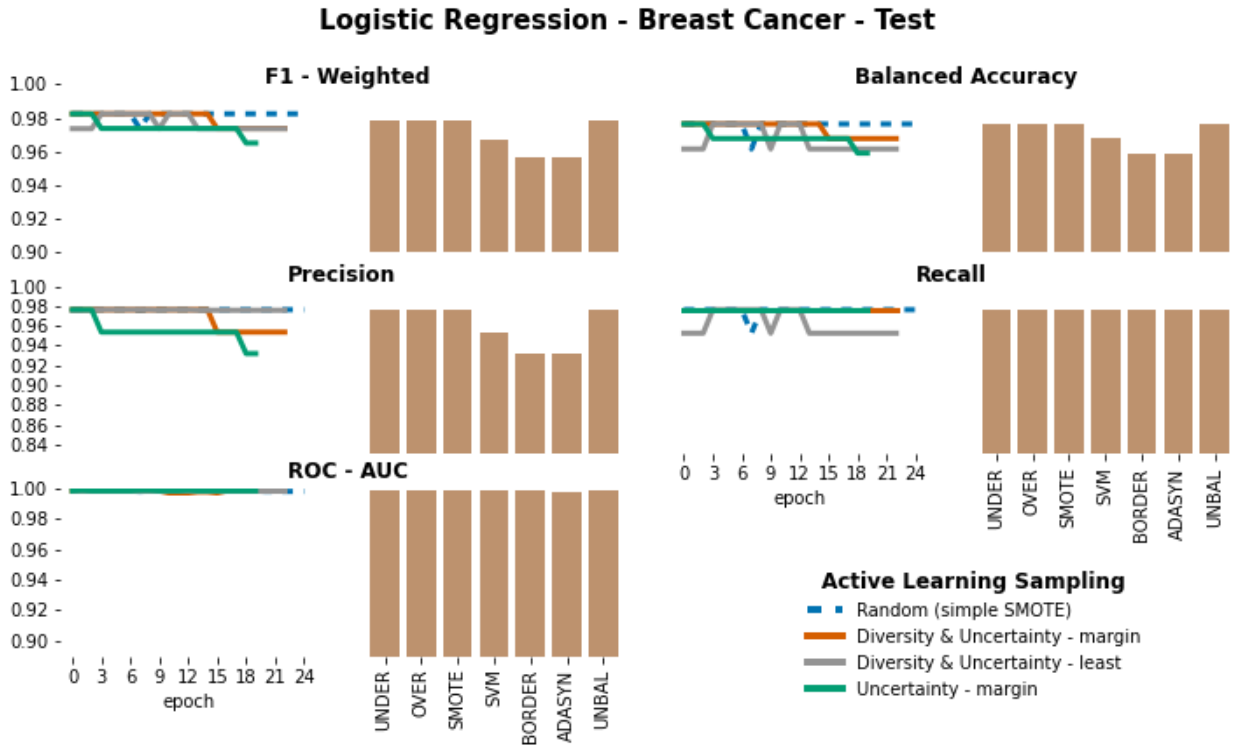

Figure 31 "Active Learning SMOTE vs. Classical Methods with LogReg Breast Cancer"

Remarks Figure 31:

- When the confidence measure margin is used, there is a decrease in precision with the active learning strategy.
- There is a slight variation in the metrics as we generate new synthetic samples, which shows that a large margin already separates the classes.

3.4.3.2. Support Vector Classifier – Breast Cancer

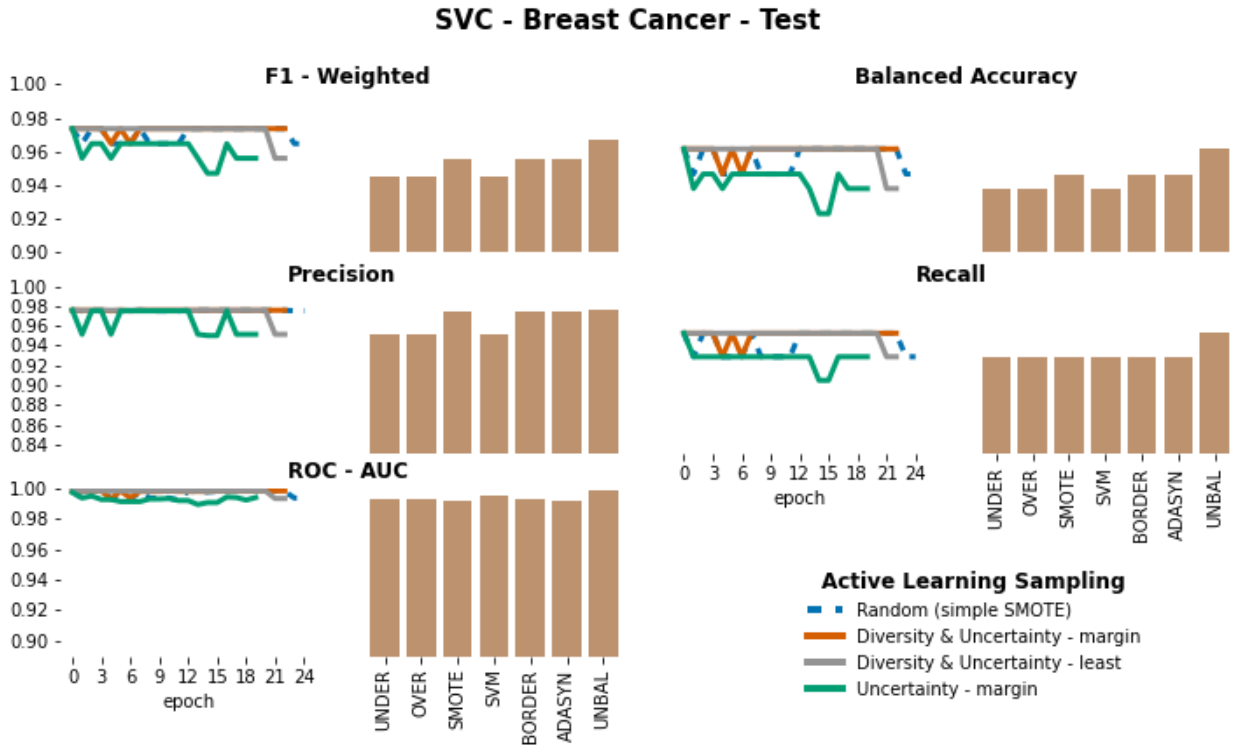

Figure 32 "Active Learning SMOTE vs. Classical Methods with SVM Breast Cancer"

Remarks Figure 32:

- The worst performance metric with Active Learning is done with margin of confidence as an uncertainty measure.
- The proposed new methods perform better than classical methods in recall.
- It is interesting that the dataset with unbalance classes has the best results.

3.4.3.3. Gradient Boost – Breast Cancer

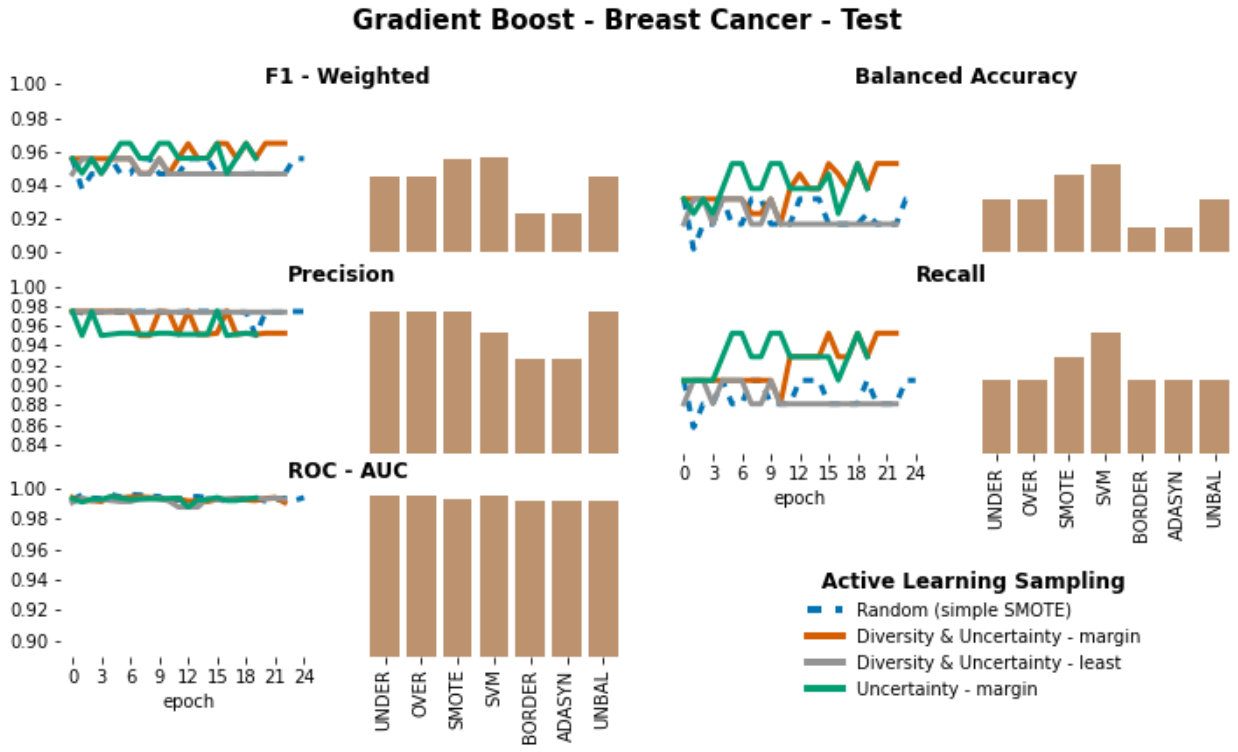

Figure 33 "AL SMOTE vs. Classical Methods Gradient Boost Breast Cancer"

Remarks Figure 33:

- The previous algorithms did not show much variance because they are simple than ensemble techniques. That is why we will see a more significant variance in the results in Gradient Boost, Ada Boost, and Random Forest. In other words, this is a case of the bias and variance trade-off.
- There is a substantial increase in recall from iterations 9 to 10. Those newly generated samples were of great utility for the model.
- Precision does not decrease when recall increases.

3.4.3.4. Ada Boost – Breast Cancer

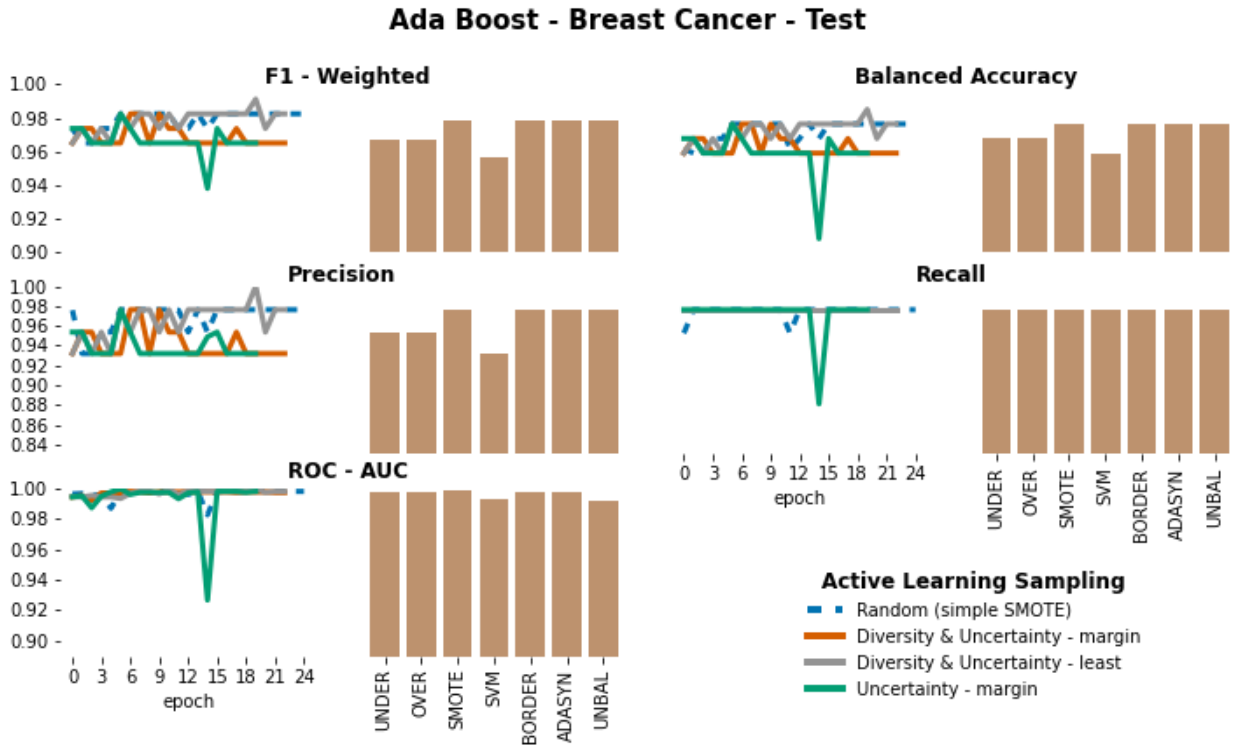

Figure 34 "AL SMOTE vs. Classical Methods Ada Boost Breast Cancer"

Remarks Figure 34:

- High variance in all metrics with active learning sampling.
- There is a considerable decrease with the active learning sampling with margin of confidence at epoch 14. However, it recovers itself very fast.

3.4.3.5. Random Forest – Breast Cancer

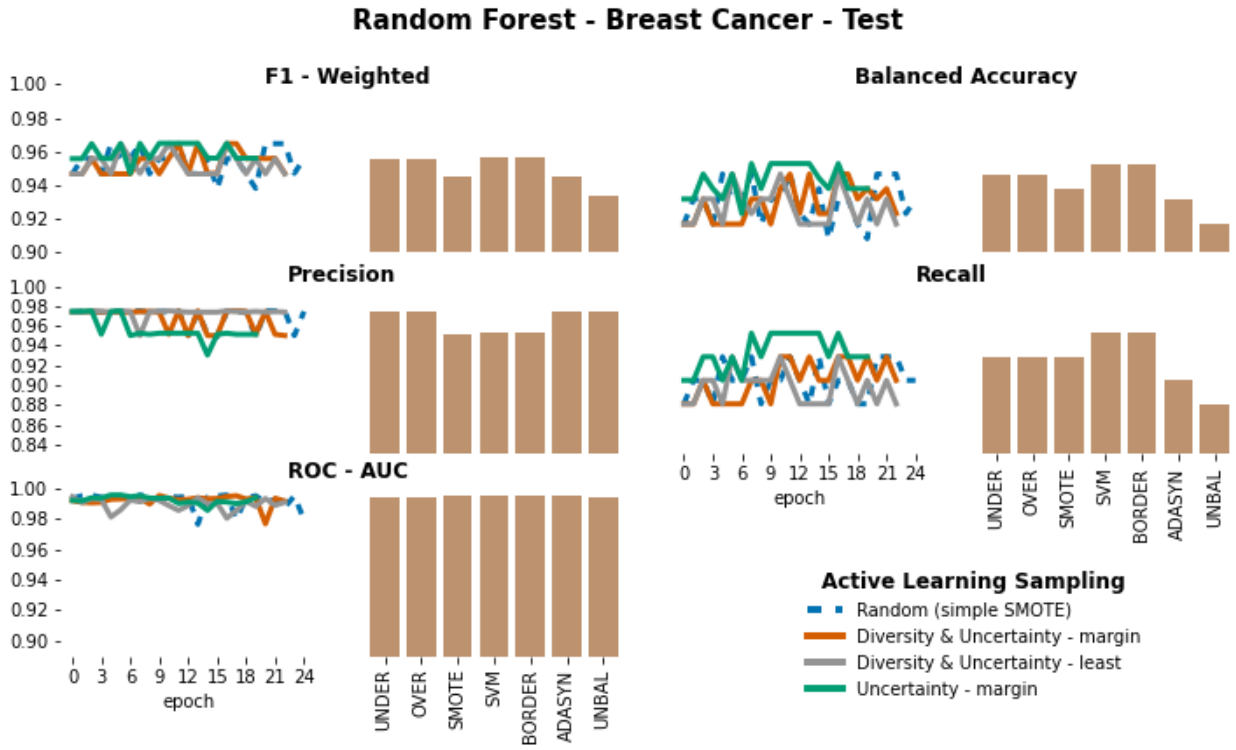

Figure 35 "AL SMOTE vs. Classical Methods Random Forest Breast Cancer"

Remarks Figure 35:

- There is high variance with the metrics, and it is not clear that sampling with an active learning strategy is helping in model performance.

Chapter 4 Conclusions

1. We can constate from the experiments that using active learning with SMOTE does not decrement the model performance.
2. We can not say that using active learning with SMOTE is better than using only random (simple SMOTE). Nevertheless, some experiments showed an improvement.
3. It is not essential to balance the dataset with synthetic or random sampling when handling class imbalance. We affirm this because, in almost all the experiments, an unbalanced dataset achieved good or better performance than a balanced dataset.

In general, we can say that the practice of balancing the dataset might not be as necessary as some might think and that there is evidence that generating fewer synthetic samples could be more beneficial.

Chapter 5 Recommendations and Future Research

As stated in section 2.3.2, many datasets are biased toward a specific gender, race, and socioeconomic background. That bias generally occurs in favor of the most privileged demographic: persons from the wealthiest nations, problems from the wealthiest economies, and other biases resulting from a power imbalance. We believe it is vital to research later if doing diversity sampling with SMOTE shows evidence of increasing the diversity of the people who can benefit from models built from data.

One of the limitations of our proposed algorithm is that it takes a considerable amount of time to execute. Another exciting research direction could be to see if it is necessary to do a random search with an ample hyperparameter space when training the model at each iteration.

Active Learning with SMOTE could also be used to know where to do medical studies to maximize model performance. For example, knowing that the model generates synthetic samples in a particular age group could enforce the decision to study that age group in real life.

All the code done in this project can be found in: [7]

References

[1] A. Fernandez, S. Garcia, M. Galar , R. C. Prati , B. Krawczyk and F. Herrera , Learning from Imbalanced Data Sets, Cham, Switzerland: Springer, 2018.

[2] M. T. V. B. V. M. Y. L. S. W. B. Tuong Le, "A Hybrid Approach Using Oversampling Technique and Cost-Sensitive Learning for Bankruptcy Prediction," in *Complexity, vol. 2019*, 2019.

[3] S. Gonzalez, "GitHub," 6 June 2017. [Online]. Available: https://github.com/sergiogvz/imbalanced_synthetic_data_plots.

[4] S. TANG and S.-p. CHEN, "The Generation Mechanism of Synthetic Minority Class Examples," in *5th International Conference on Information Technology and Applicaiton in Biomedicine*, Shenzhen, China, 2008.

[5] G. Menardi and N. Torelli, Training and assessing classification rules with imbalanced data, Data Min. Knowl., 2014.

[6] R. Munro , Human-in-the-Loop Machine Learning, New York: Manning, 2021.

[7] R. Sena Rojas, "GitHub," 31 August 2022. [Online]. Available: https://github.com/rsena-felipe/medical-class-imbalance.

[8] H. Han, W. Wang and B. Mao, "Borderline–SMOTE: a new over–sampling method in imbalanced data sets learning," in *International Conference of Intelligent Computing*, Hefei, 2005.

[9] H. He, Y. Garcia, E.A. and S. Li, "ADASYN: adaptive synthetic sampling approach for imbalanced learning," in *International Joint Conference Neural Networks*, Hong Kong, 2008.

[10] UCI Machine Learning, "PIMA Indian Diabetes Dataset," 2014. [Online]. Available: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.

[11] UCI Machine Learning, "Breast Cancer Dataset," [Online]. Available: https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic).