Electronic Communications of the EASST
Volume 82 (2022)

11th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation

-

Doctoral Symposium, 2022

Binary Decision Diagrams and Composite Classifiers for Analysis of Imbalanced Medical Datasets

Amandeep Singh, Olga Minguett, Tiziana Margaria

21 pages

Guest Editors: Sven Jörges, Salim Saay, Steven Smyth
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Binary Decision Diagrams and Composite Classifiers for Analysis of Imbalanced Medical Datasets

Amandeep Singh[1], **Olga Minguett**[2], **Tiziana Margaria**[3]

[1] Centre for Research Training in Artificial Intelligence (CRT AI)
University of Limerick
Amandeep.Singh@ul.ie
[2] Optum, and
University of Limerick
olgaminguett@gmail.com
[3] Centre for Research Training in Artificial Intelligence (CRT AI), and
Health Research Institute, and
University of Limerick
Tiziana.Margaria@ul.ie,

Abstract: Imbalanced datasets pose significant challenges in the development of accurate and robust classification models. In this research, we propose an approach that uses Binary Decision Diagrams (BDDs) to conduct pre-checks and suggest appropriate resampling techniques for imbalanced medical datasets as the application domain where we apply this technology is medical data collections. BDDs provide an efficient representation of the decision boundaries, enabling interpretability and providing valuable insights. In our experiments, we evaluate the proposed approach on various real-world imbalanced medical datasets, including Cerebral-stroke dataset, Diabetes dataset and Sepsis dataset. Overall, our research contributes to the field of imbalanced medical dataset analysis by presenting a novel approach that uses BDDs and composite classifiers in a low-code/no-code environment. The results highlight the potential for our method to assist healthcare professionals in making informed decisions and improving patient outcomes in imbalanced medical datasets.

Keywords: Low-code, imbalanced datasets, data balancing techniques, sampling techniques, classifiers, ADD-Lib, decision support systems, Binary Decision Diagrams.

1 Introduction

It is a well known problem in Machine Learning (ML) that most real-life datasets are imbalanced [DDC18, LNA17]. Class imbalance occurs when one or more of the classes/categories in a dataset are under-represented. For example, in a binary classification problem, there would be more elements from one class and than the other, whereas in multi-classification scenarios, the observations would be more clustered compared to the others. This is an issue while training ML models because it can create a performance bias in the model, making it perform poorly on real-life scenarios. This issue is highlighted while working with datasets in the medical

domain, especially in a binary classification scenario, where the number of affected patients are considerably lower than the unaffected patients. For example, in a dataset for diabetes, the number of patients with diabetes might be significantly lower than the number of patients without diabetes. In such a scenario the ML model may not make the best classification and may make false predictions. Making false predictions, especially when it directly affects the lives of people, is a big problem that arises from class imbalance. Moreover, the ML algorithms are designed to assume that each class has a similar number of specimens or instances [XSNK20], which seriously hampers the detection of rare events. To counter the issue of class imbalance, resampling techniques are implemented.

Resampling techniques can be used in three ways: (1) to modify the training dataset for the ML models so that it has equal representation of the minority class; (2) to modify the ML models by either associating a cost/weight variable with the minority class or designing new models for the specific dataset; or (3) a combination of techniques in (1) and (2). Based on these options, resampling techniques are broadly classified into three types: (1) data-driven resampling, (2) algorithm-driven resampling, and (3) hybrid resampling. Data-driven resampling techniques are further classified into three types based on how the majority-minority class distribution is augmented in the dataset: undersampling, oversampling, and hybrid. Undersampling removes observations from the majority class while keeping all observations from the minority class. Because this reduces the overall size of the dataset, it has the advantage of lowering memory consumption during model training. At the same time, removing observations from the majority class may result in a loss of information, which may result in poor model performance. Oversampling involves replicating specific or random observations from the minority class in order to increase the total number of observations. This balances the minority and majority classes and raises the overall number of observations. This may result in stronger class boundaries while not deleting any data, but it may also result in longer computation times and increased resource usage. For the hybrid data-driven resampling, both techniques, undersampling and oversampling, are used, with undersampling used to remove observations from the majority class and oversampling used to replicate observations in the minority class. For the purpose of this research, only data-driven resampling techniques are used.

The resampling techniques described can be implemented on the imbalanced datasets to potentially improve performance and make better predictions but it would require sufficient knowledge of statistics and programming. There are also a variety of data-driven resampling techniques that can be applied on a given dataset based on the degree of class imbalance. This means that for every imbalanced dataset, a custom logic and programming would have to be implemented to decide on what techniques are best suited for that scenario. This is another issue where domain experts like doctors, historians or biologists have to work with programming languages, which they might not be proficient in, just to decide a rudimentary workflow for a dataset to make it usable. In this regard, low-code/no-code applications can play a huge role in making knowledge more accessible to the domain experts while also addressing the class imbalance problem. From the point of view of the development of such workflows, this will also promote code reusability, saving precious development time and may even make the models perform better. So, the solution would be a system that suggests whether a particular technique is suitable for a specific dataset using the degree of class imbalance in that dataset, apart from being understandable, accessible and usable by domain experts.

One such representation that is designed for giving a yes/no decision for a choice, is also structured in a way that follows general logic without the requirement of extra skills to understand is a Binary Decision Diagram (BDD) [Ake78]. BDDs are the most popular form of Decision Diagrams (DDs) and is a representation that is based on the idea of going through many successive binary decisions till the objective is achieved. Since BDDs are a graph-based representation for Boolean functions, they were an optimum choice for modelling decision workflows in our study. One of the frameworks that works well with the BDD representation, is well established and provides a flexible and intuitive interface for our use-case is ADD-Lib [GMZS19], which will be the choice of framework in this research.

This research extends the M.Sc. thesis project of Olga Minguett [Min22], who researched the datasets and resampling methods in her thesis. The new contributions are as follows:

- Restructuring of Python code: The original code was written for the purpose of analysis in Jupyter notebooks. To use the code for data pipelines that can be reused, the code was restructured and rewritten in order to adhere to the encapsulation and the OTA (One Thing Approach) paradigms [MS09a].

- Data Analytics and ML pipelines: The new Python code was compiled in Pyrus [ZS21]. The data analytics and ML pipelines for three datasets, five ML models and various different resampling techniques were designed specifically with explainability and reusability in focus. The initial results using these pipelines are shown in Table 1 and discussed in Section 5.

- ADD-Lib Decision Diagrams: Using the new Python code and results from the Pyrus pipelines, the knowledge gained was used to create a decision workflow that was applicable across all selected datasets. BDDs and composite classifiers were then designed and applied in ADD-Lib. The BDDs and composite classifier were verified using a reference classifier as well.

This paper is organised as follows. Section 2 summarises the existing work in this domain and Section 3 lists all the tools and technologies employed in this study. Section 4 gives an overview of the methodology, followed by Section 5 that describes and discusses the results, and conclusions and future work are presented in Section 6.

2 Related work

The field of medicine and Artificial Intelligence, especially ML and Deep Learning (DL), have made many strides in the past few years, but the ethical and technical challenges still remain [RCBT22]. A lot of studies have used different ML/DL models to detect diseases, such as diabetes [GPC+16], skin cancer [EKN+17], and for chest radiographs [RIB+18]. The datasets used in such studies are referred to as medical datasets in this research.

Most of the progress in AI and medicine has been made in medical image analysis and randomized controlled trials. The authors opine that for non-image data and for unconventional

problems, there have to be more novel, collaborative efforts to take advantage of the opportunities. Tsopra et al. [TFL+21] agree that AI has the potential to transform medicine but has been limited in its implementation in the field due to lack of a robust validation framework. Tsopra et al. [TFL+21] present a framework with seven steps that is used to build a community competition ("*ITFoC Challenge*"), to critically evaluate and compare different predictive AI algorithms in medicine using real-world datasets. Mesquita et al. [MOSP20], on the other hand, reviewed the different algorithms used to detect misinformation/disinformation in medical science, especially during the COVID-19 pandemic.

Rapid breakthroughs in the medical domain using ML/DL have raised important concerns, safe role of ML in medicine, explainability of the ML/DL models being used and deployed, risk of misinformation due to use by non-experts, and other technical challenges that physicians are not aptly equipped to deal with. One such challenge for the ML/DL models is the imbalance in the medical data being used for such studies. Imbalance in data occurs when a category in the target value of the dataset is not equally represented, which makes the trained model biased and not suitable for predictions. This is an ongoing issue in ML, which also leads to the questions that are targeted in this research: What are the best machine learning algorithms for different degrees of class imbalance in medical datasets? How can a graphical representation be used to make the decisions around medical datasets more interpretable? How could developers and domain experts be benefited by using a more low-code/no-code approach for making the solution more explainable and reusable?

To address these questions of imbalance in medical datasets and to combat the challenge of explainability in AI in the medical domain, a low-code/no-code pipeline is proposed using the CINCO product [NLKS18] ADD-Lib [GMZS19] as part of a Digital Thread solution [MS19] for the health and medical domain. The other CINCO products consist of DIME [BFK+16], which is a low-code development environment for creating web applications, and Pyrus [ZS21], a web-based, low code tool for data analytics.

As explained in Section 1, there are a few types of resampling techniques that can be used in case the dataset is imbalanced. One of the ways is to customise the ML models being used for a specific domain to achieve the required results without modifying the dataset, which is called algorithm-driven resampling. The crucial part of this approach would be to make the ML models more explainable to enable more customisability. For example, Random Forests, which is a popular choice for an ML classification model, has a drawback that the larger the model size the better the results but the model training time increases linearly with the model size. With increase in the size of the ML model, it becomes difficult to understand the decisions and decision boundaries. One way to increase explainability and improve running time would be to aggregate the ML models into semantically equivalent Decision Diagrams (DDs) [GS20]. Gossen et al. (2020) [GMS20] have successfully applied this approach using Algebraic Decision Diagrams (ADDs) to improve running time and explainability of the Random Forests model applies on the popular *Iris* dataset [Fis88]. Gosen (2021) [Gos21] has also shown that domain specific program optimisation techniques using ADDs [GJMS19] can be successfully implemented in ADD-Lib, demonstrating the merits of the platform.

Algorithm-driven resampling, even though may produce better results than data-driven resampling since the ML model is customised to the problem, also has the drawback of being specific to the problem or the domain and lacks generalisability. Our approach in this study focuses on

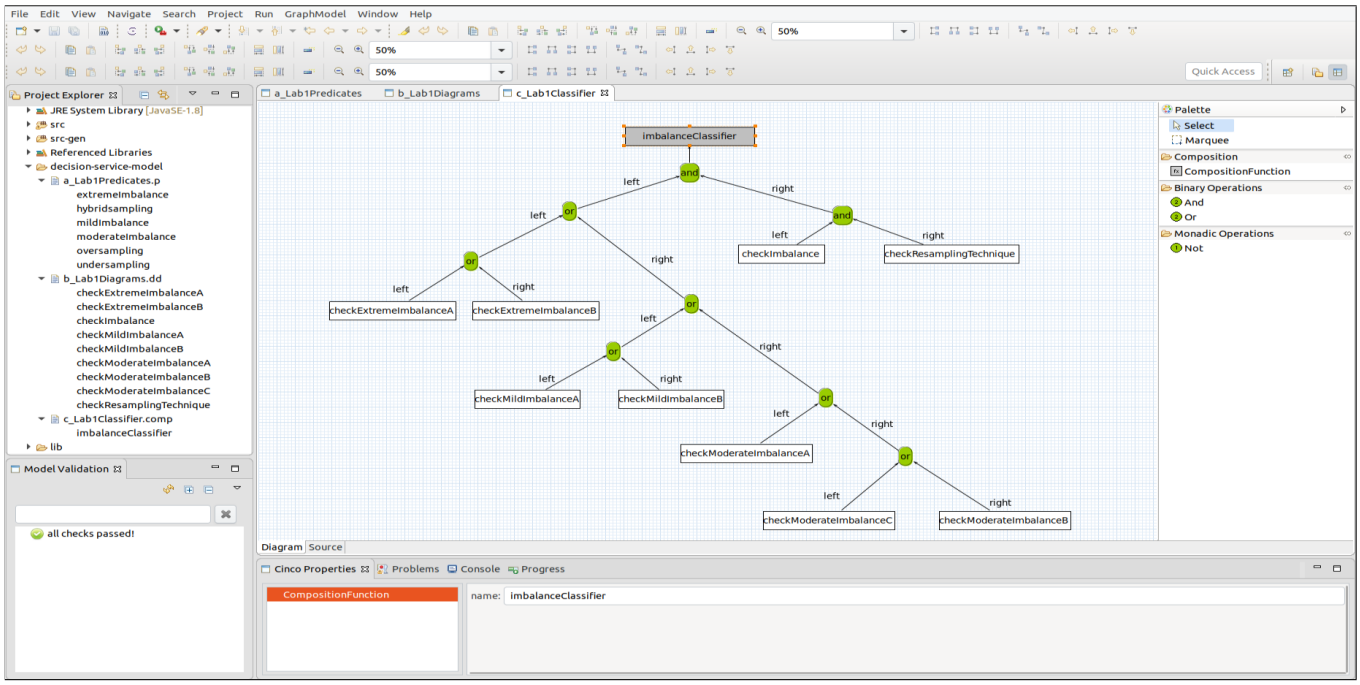

Figure 1: ADD-Lib: View of the development framework as modelling environment

data-driven resampling techniques and explores imbalanced medical datasets to generalise the solution as much as possible using BDDs.

3 Tools and Technologies

In this study, we used BDDs to perform symbolic analysis using Boolean functions as directed acyclic graphs. This approach allowed us to perform complex logical reasoning by encoding different parameters in Boolean variables [Ake78].

Since ADD-Lib works well with ADDs and BDDs, and has a flexible interface to model the objectives of this study, it was the choice of framework for this study. Choosing ADD-Lib also allows our solution to be reused and made generally available through a Digital Thread solution using CINCO family of low-code/no-code tools [MS19, NLKS18]. ADD-Lib is a framework that was created to overcome the limitations of CUDD [KGD23, Bry86], which was the existing library for the implementation of BDD. In CUDD, ADDs [BFG+97] were limited to real numbers and standard arithmetic operations. ADD-Lib overcomes these limitations by emphasizing on the interchangeability of the underlying algebraic structure. More so, the computationally intensive operations are delegated to the robust C implementation of CUDD in a service-oriented fashion. ADD-Lib can also be used to generate code in several languages such as C, C++, Java, and Javascript. The ADD-Lib tool also supports easy exchange of algebra that can be used to transform into a more granular recommender system.

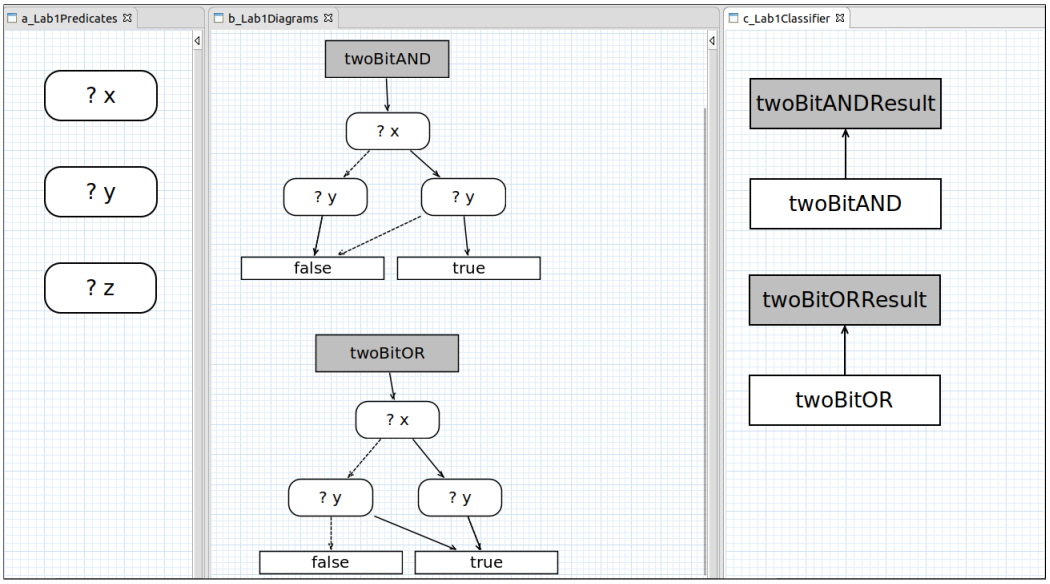

Figure 2: ADD-Lib: All three model files in view (example)

A view of the interface is shown in Figure 1. In Figure 1, the top left hand side houses the *Project Explorer* for showing all the project files, lower left hand side is the built-in *Model Checker* that displays errors if there are any incomplete or erroneous decision diagrams. The central section of interface is the area to drag and drop different Boolean functions to create decision diagrams. On the right-hand side is the *Palette* where blocks of Boolean functions (*AND*, *OR*, *NOT*) and terminal nodes (*true*, *false*) are located for easy dragging and dropping. The bottom section is where the properties of a selected Boolean function and the *Console* are shown.

ADD-Lib mainly has three models that users interact with:

- The first model is for the declaration of features and categories to be used in decision diagrams, as shown in the left region of Figure 2. These are called *predicates* and are generally an individual or a set of functions that characterize the input by a Boolean vector. The predicates are defined by their name and their characteristic function, and are computed only when needed.

- The second model is the decision diagram model for designing BDDs that determine the outcome based on predicates over the features, as shown in the central region of Figure 2.

- The third model is for creating a composite classifier using the individual decision diagrams, as shown in the right region of Figure 2.

The predicates, once declared in the first model, can be used in the BDDs in the second model. Similarly, the BDDs in the second model can be referenced in the third model using their function names. Generally, the BDDs can describe the decisions for individual conditions for a task and the composite classifier will be a composition of the decisions for all conditions using a

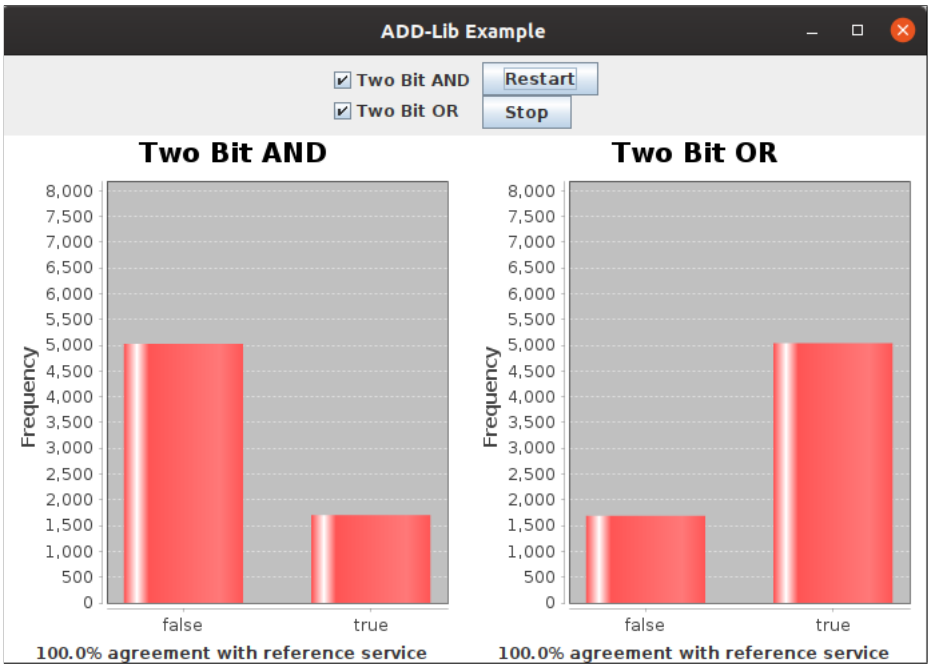

Figure 3: ADD-Lib: Example reference classifier output

combination of Boolean functions (*AND*, *OR*, *NOT*). For example, in Figure 2, the predicates x, y and z are created in the predicates model and then they were used to create two simple BDDs in the decision diagram model - namely *twoBitAND* and *twoBitOR*. A composite classifier is also created using these two BDDs.

Another feature in ADD-Lib is the ability to create a baseline/reference classifier that can compare the expected outcome to the actual outcome. This is a useful feature because it can help with tuning and adjusting the BDD and composite classifier models. For the above example with *twoBitAND* and *twoBitOR* BDDs, a reference classifier was created. After compiling it as a Java application from the source-code, the output is shown in Figure 3. The plot in this figure is generated using random values that are supplied to the reference classifier and the actual composite classifier. This figure shows that the expected outcome is 100% in agreement with the actual outcome of the BDDs.

4 Methodology

We describe here the datasets we chose, the kinds of resampling techniques and the effective specific techniques that were chosen for this study, and the five popular and very successful machine learning techniques that were chosen for comniation with resampling.

4.1 Datasets

Based on the established classification of datasets, a dataset can be classified in to three types based on the degree of imbalance in the target class: (1) mild (20-40%); (2) moderate (1-20%); or (3) extreme ($\leq$1%)[1]. Based on this classification, a variety of well-documented medical datasets were chosen.

The three datasets chosen were the Cerebral Stroke Dataset [LFW19], Diabetes Dataset [Xie19] and the Sepsis Dataset [KFJB].

1. Cerebral Stroke Dataset: A brain attack or cerebral stroke can occur due to a blockage in blood supply to the brain or from a burst blood vessel in the brain. This can present in patients in the form of various physiological symptoms but can avoid detection due to complex brain imaging procedures required for detection, time between symptoms and imaging, cost associated with imaging, and overlapping symptoms with other possible conditions [WF04]. The Cerebral Stroke dataset was created to aid in detection of a stroke using classification algorithms. This dataset has 12 features containing 43.4K observations out of which 783 observations are labelled to be stroke. The class-imbalance ratio for this dataset is 1.84%, so this dataset will fall under the extreme imbalance category ($\leq$1%).

2. Diabetes Dataset: Approximately 537M adults worldwide are suffering from diabetes[2]. Early detection, especially for type-2 diabetes which is the most prevalent type, can be an important factor for control and self-management [Cav16]. The diabetes dataset was extracted from the Behavioral Risk Factor Surveillance System (BRFSS) 2014 dataset that was published by the CDC[3]. The extracted diabetes dataset has 254.6K observations with the target variable having 2 classes - 0 for no diabetes, 1 for diabetes. The dataset required extensive pre-processing (removing empty values, outliers). The class-imbalance ratio for this dataset is 19.05%, so this dataset will fall under the mild imbalance category (20-40%).

3. Sepsis Dataset: Sepsis is the leading cause of death worldwide [LEG+14]. With the aim to aid in early detection of sepsis, this dataset was created for the Computing in Cardiology Challenge from Physionet 2019. The dataset has 40 features and 36.3K observations. The target feature, 'isSepsis', has 2 classes - 0 for no sepsis, 1 for sepsis. The class-imbalance ratio for this dataset is 7.87%, so this dataset will fall under the moderate imbalance category (1-20%).

All three datasets are freely available for research purposes.

4.2 Resampling Techniques

The choice of data-driven resampling techniques and machine learning (ML) models were dependent on the platform of choice for analysis of datasets and resources available for the analy-

[1] Imbalanced Data, Google Developers: developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data
[2] IDF Diabetes Atlas 2021: diabetesatlas.org/atlas/tenth-edition/
[3] https://www.cdc.gov/brfss/annual_data/annual_2014.html

sis. The platform of choice was *Jupyter* notebooks[4] using *Python* because of prior experience, vastness and modularity of libraries, and abundance of documentation for data analysis. The *imbalanced-learn* library [LNA17][5], which relies on the popular Python ML library *scikit-learn*[6], was used to implement the resampling techniques. The *imbalanced-learn* package was a natural choice because it was specifically created to contain all the major techniques used for resampling datasets. Moreover, the reliance on the *scikit-learn* package means that the compatibility with ML models would be seamless.

Since the aim for the experimentation was to be as robust and generalised as possible, the data-driven resampling techniques chosen were from all different sub-categories that were available in the *imbalanced-learn* package. The following data-driven resampling techniques were selected:

1. Oversampling: RandomOverSampler, SMOTE, SMOTENC, BorderlineSMOTE, SVMSMOTE, KMeansSMOTE, ADASYN

2. Undersampling: RandomUnderSampler, ClusterCentroids, NearMiss, InstanceHardnessThreshold, TomekLinks, CondensedNearestNeighbour, AllKNN, EditedNearestNeighbours, RepeatedEditedNearestNeighbours, OneSidedSelection, NeighbourhoodCleaningRule

3. Combination/Hybrid: SMOTEENN, SMOTETomek

For the scope of this research, we selected all the different techniques that were available under the *imbalanced-learn* package.

4.3 Machine Learning Models

The following ML models were chosen to represent the popular learning techniques generally used for most ML-based classification purposes:

1. Support Vector Machine (SVM)

2. Decision Tree (DT)

3. Gaussian Naïve Bayes (GNB)

4. K-Nearest Neighborhood (KNN)

5. Logistic Regression (LR)

These ML models were chosen because of their popularity an their varied use-cases. SVM, DT, GNB and LR are examples of supervised ML models whereas KNN is an unsupervised ML model. Furthermore, all these models use very different approaches to classify the target variables. For the purpose of this research, only five models are explored.

[4] The Jupyter Notebook: https://jupyter.org
[5] https://imbalanced-learn.org/stable/
[6] https://scikit-learn.org/stable/

5 Results

The aim of this study was to simplify the evaluation process for an imbalanced dataset and present users with a data-driven resampling technique that will best suit the scenario depending on the degree of imbalance of the target class. As stated in Section 4, the target class can have one of three degrees of imbalance - mild (20-40%), moderate (1-20%); or extreme ($\leq$1%) [**?**]. The degree of imbalance can then be used to select the best suited data resampling techniques before the dataset is used for any ML application. This can potentially improve results and also remove bias introduced through data imbalance. The best suited technique for the degree of imbalance can be recognised by running tests on different datasets from the medical domain. To demonstrate the problem and our solution, a variety of well-documented datasets were chosen (Section 4.1). The datasets were particularly from the medical domain because the problem of class imbalance is especially highlighted in this area. To showcase the performance of our solution with these specifically selected datasets would bring it one step closer to generalising on more prevalent, non-medical, imbalanced datasets. Furthermore, using low-code/no-code tools like ADD-Lib can speed the whole process of balancing the dataset by reusing the same data pipelines and decision diagrams.

5.1 Initial Results and Decision Workflow

Based on the results from the initial experiments using the techniques and models described in Table 1, a decision workflow diagram was constructed, as shown in Figure 4. The overall results generally indicated for oversampling (*KMeansSMOTE, RandomOverSampler*) and hybrid resampling (*SMOTETomek*) to be used for mild and extreme cases of data imbalance, whereas oversampling (*RandomOversampler*), undersampling (*TomekLinks*) and hybrid resampling (*SMOTETomek*) for moderate case of data imbalance.

5.2 ADD-Lib Decision Diagrams

The decision workflow diagram in Figure 4 represents the different steps in the decision process and the requirements our ADD-Lib solution would have to check for. The initial condition is to check whether the dataset is even imbalanced. The decision of this condition will determine whether the dataset would need to go through the ADD-Lib composite classifier or not. This was implemented using the predicates *mildImbalance*, *moderateImbalance* and *extremeImbalance* shown in Figure 5. The BDD using these predicates is shown in Figure 6 (left). The BDD *checkImbalance* shown will confirm that a dataset has at least one type of imbalance to reach the next stage.

The next condition in the workflow is to determine the degree of imbalance in the target class of the dataset - mild, moderate and extreme. This decision will determine the type of data-driven resampling technique to be applied. In the case of mild imbalance, the oversampling and hybrid techniques are to be used. This is implemented using the predicates *mildImbalance*, *oversampling* and *hybridsampling* predicates from Figure 5 are used in BDDs as shown in Figure 7. As can be seen from the BDDs *checkMildImbalanceA* and *checkMildImbalanceB* in the figure, if

Table 1: Initial Results using resampling techniques and ML models

Dataset	Technique	Model	Method	f1 Score
Cerebral Stroke	Undersampling	DT	NearMiss	*0.3702*
		SVC	NearMiss	*0.4101*
		KNN	TomekLinks	**0.9906**
		KNN	OneSidedSelection	**0.9906**
	Oversampling	GNB	ADASYN	*0.8247*
		GNB	SMOTENC	*0.8309*
		DT	KMeansSMOTE	**0.9760**
		DT	RandomOverSampler	**0.9831**
	Hybrid	GNB	SMOTEENN	*0.8264*
		DT	SMOTETomek	**0.9736**
Diabetes	Undersampling	DT	ClusterCentroids	*0.3769*
		DT	NearMiss	*0.5459*
		KNN	TomekLinks	**0.9111**
		KNN	OneSidedSelection	**0.9111**
	Oversampling	GNB	ADASYN	*0.7596*
		GNB	KMeansSMOTE	*0.7628*
		DT	RandomOverSampler	**0.8800**
		SVC	KMeansSMOTE	**0.8963**
	Hybrid	KNN	SMOTEENN	*0.7570*
		DT	SMOTETomek	**0.8768**
Sepsis	Undersampling	SVC	ClusterCentroids	*0.4018*
		GNB	NearMiss	*0.4941*
		DT	TomekLinks	**0.9730**
		DT	OneSidedSelection	**0.9736**
	Oversampling	LR	ADASYN	*0.8549*
		LR	BorderlineSMOTE	*0.8657*
		DT	SVMSMOTE	**0.9648**
		DT	RandomOverSampler	**0.9736**
	Hybrid	KNN	SMOTEENN	*0.8399*
		DT	SMOTETomek	**0.9601**

the degree of imbalance is mild, then either oversampling or hybrid resampling is applied, and no technique otherwise.

For moderate imbalance, *oversampling*, *undersampling* and *hybridsampling* predicates from Figure 5 are used in BDDs as shown in Figure 8. As can be seen from the BDDs *checkModerateImbalanceA*, *checkModerateImbalanceB* and *checkModerateImbalanceC* in the figure, if the degree of imbalance is moderate, then oversampling, undersampling or hybrid resampling is applied, and no technique otherwise.

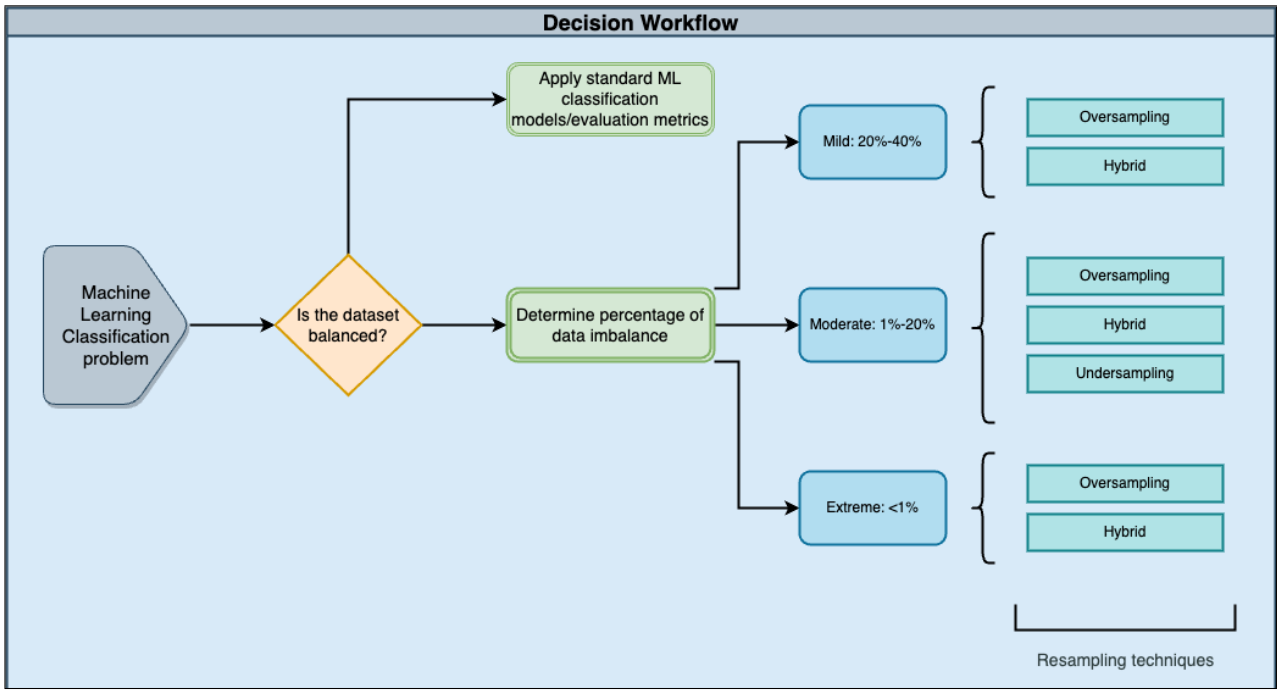

Figure 4: ADD-Lib: Decision Workflow constructed using initial results

For extreme imbalance, *oversampling* and *hybridsampling* predicates from Figure 5 are used in BDDs as shown in Figure 9. As can be seen from the BDDs *checkExtremeImbalanceA* and *checkExtremeImbalanceB* in the figure, if the degree of imbalance is extreme, then either oversampling or hybrid resampling is applied, and no technique otherwise.

There is also a possibility that somehow none of the conditions for at least one resampling technique are satisfied. To catch this edge case scenario, the *checkResamplingTechnique* BDD in Figure 6 was constructed using the *undersampling*, *oversampling* and *hybridsampling* predicates from Figure 5. This ensures that at least one resampling technique is applied.

All individual BDDs represent the decisions for individual conditions in the larger decision workflow described in Figure 4. To create a composite classifier, the individual BDDs would be combined using Boolean functions. By definition, the *OR* function returns *TRUE* if at least one of the conditions is correct and *AND* function returns *TRUE* if both conditions are correct.

5.3 ADD-Lib Composite Classifier

Figure 10 shows the composite classifier for this study. Each BDD described above is used here in combination with a Boolean function. The two *checkMildImbalance* (A and B) BDDs are combined using Boolean *OR* because at least one (but only one) of the techniques (oversampling or hybrid sampling) have to applied. For *checkMildImbalanceA* and *checkMildImbalanceB* BDDs, only one (but at least one) of them have to satisfy the condition for the combination of the two to be *TRUE*. Based on the same logic, the three BDDs for *checkModerateImbalance* (A, B and C) are combined with *OR* and the two BDDs for *checkExtremeImbalance* (A and B)

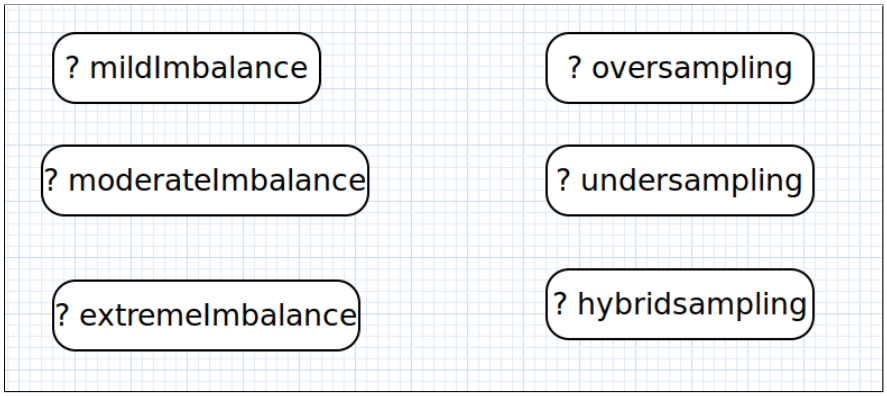

Figure 5: ADD-Lib: Predicates model

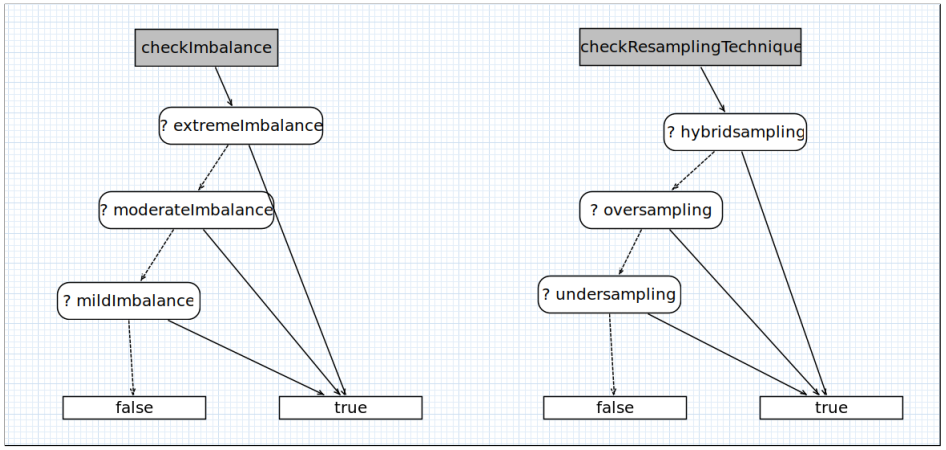

Figure 6: ADD-Lib: Binary Decision Diagram model (A) - (left) determining the degree of data imbalance, and (right) confirming that at least one of the data-driven resampling techniques is applied

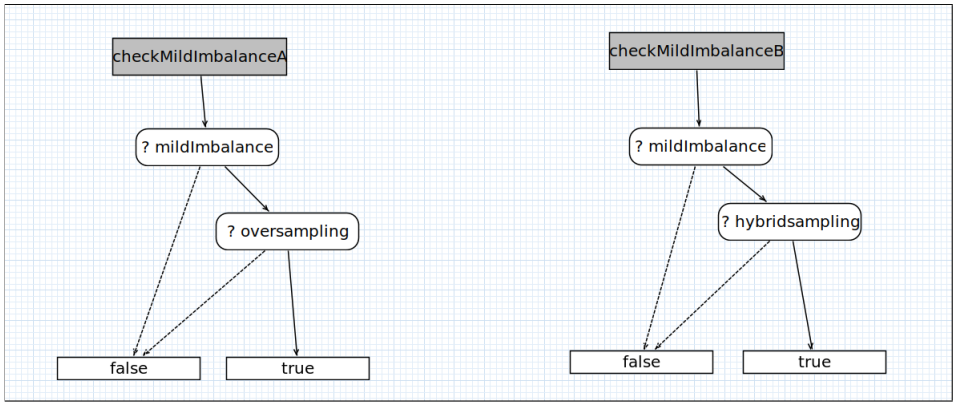

Figure 7: ADD-Lib: Binary Decision Diagram model (B) - for mild data imbalance oversampling and hybrid resampling are applied

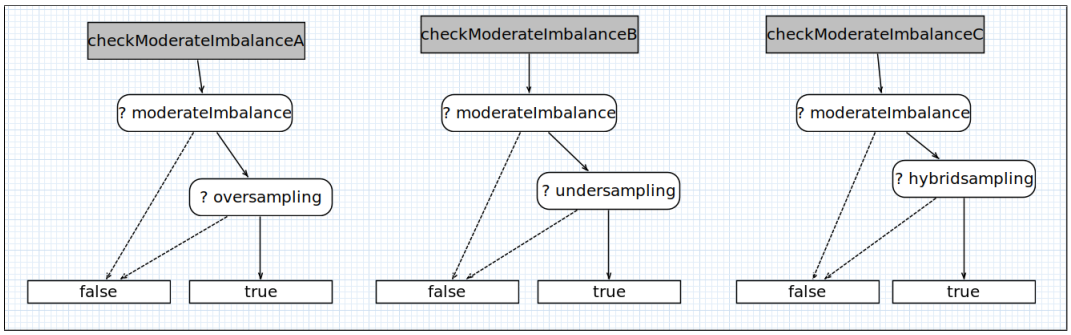

Figure 8: ADD-Lib: Binary Decision Diagram model (C) - for moderate data imbalance oversampling, undersampling or hybrid techniques are applied

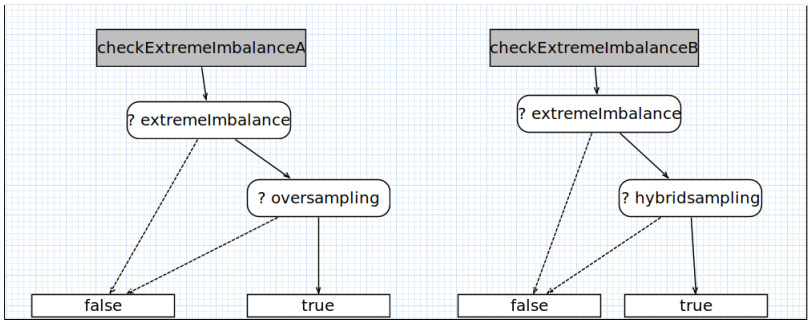

Figure 9: ADD-Lib: Binary Decision Diagram model (D) - for extreme data imbalance oversampling and hybrid resampling are applied

are also combined with *OR*. Now since a given dataset can only fall in one of the three imbalance categories (mild, moderate or extreme), only one (but at least one) of the three imbalance BDDs would be needed. So for this scenario, the resultant of all imbalance-condition BDDs is combined together with *OR*. Finally, the *checkImbalance*, *checkResamplingTechnique* and the resultant from the combination of resampling technique BDDs are combined through Boolean *AND* so that the composite classifier returns *TRUE* only when all conditions are *TRUE*, i.e., the dataset is imbalanced *AND* at least one resampling technique is applied.

The Java code behind the predicates and the BDDs in Figure 5 and Figures 6 to 9 respectively is shown in Figure 11. This code was generated automatically through the ADD-Lib code generator after initialising the predicates and BDDs through the graphical interface. This is an advantage of using a platform like ADD-Lib where complex logic and representations can be easily portrayed and the code is automatically generated.

5.4 Verifying Results

The final result is shown in Figure 12. This output was obtained from the comparison of results from the reference classifier and the actual composite classifier (composite classifier in

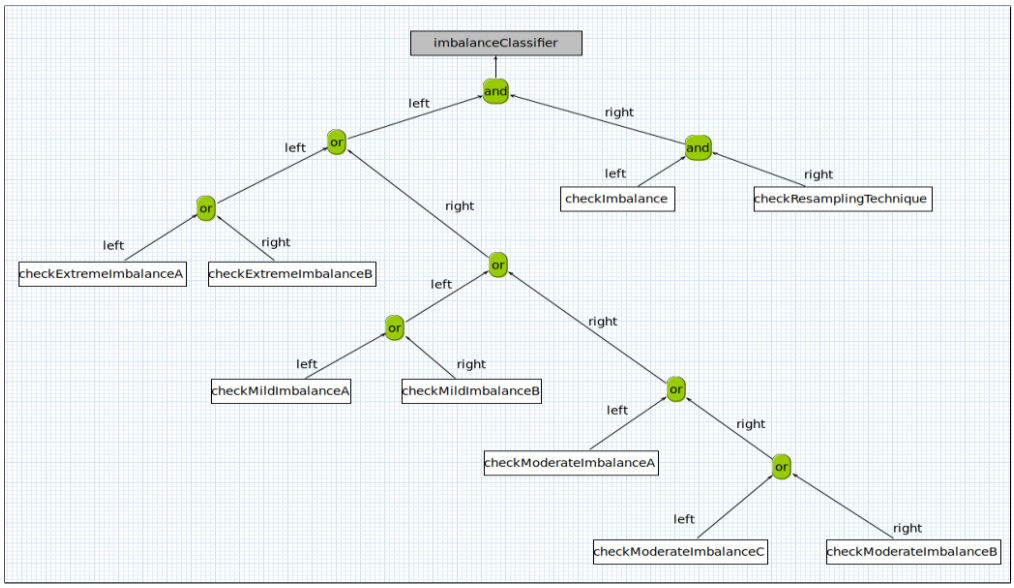

Figure 10: ADD-Lib: Composite Classifier model - a composition of all BDDs using a combination of Boolean functions

```java
/* This file was generated with the ADD-Lib
* http://add-lib.scce.info/ */
package info.classifiers;
public class BooleanExampleClassificationService {

    public boolean imbalanceClassifier(BooleanExamplePredicates predicates) {
        return eval140540725456960(predicates);
    }
    private boolean eval140540725456960(BooleanExamplePredicates predicates) {
        if (predicates.extremeImbalance())
            return eval140540725456928(predicates);
        else
            return eval140540725456864(predicates);
    }
    private boolean eval140540725456864(BooleanExamplePredicates predicates) {
        if (predicates.oversampling())
            return eval140540725456800(predicates);
        else
            return eval140540725456832(predicates);
    }
    private boolean eval140540725456832(BooleanExamplePredicates predicates) {
        if (predicates.hybridsampling())
            return eval140540725456800(predicates);
        else
            return eval140540725456704(predicates);
    }
    private boolean eval140540725456704(BooleanExamplePredicates predicates) {
        if (predicates.moderateImbalance())
            return eval140540725456672(predicates);
        else
            return eval140540725455968(predicates);
    }
    private boolean eval140540725456672(BooleanExamplePredicates predicates) {
        if (predicates.undersampling())
            return eval140540725455936(predicates);
        else
            return eval140540725455968(predicates);
    }

        if (predicates.undersampling())
            return eval140540725455936(predicates);
        else
            return eval140540725455968(predicates);
    }

    private boolean eval140540725456800(BooleanExamplePredicates predicates) {
        if (predicates.mildImbalance())
            return eval140540725455936(predicates);
        else
            return eval140540725456512(predicates);
    }
    private boolean eval140540725456512(BooleanExamplePredicates predicates) {
        if (predicates.moderateImbalance())
            return eval140540725455936(predicates);
        else
            return eval140540725455968(predicates);
    }
    private boolean eval140540725456928(BooleanExamplePredicates predicates) {
        if (predicates.oversampling())
            return eval140540725455936(predicates);
        else
            return eval140540725456896(predicates);
    }
    private boolean eval140540725456896(BooleanExamplePredicates predicates) {
        if (predicates.hybridsampling())
            return eval140540725455936(predicates);
        else
            return eval140540725456704(predicates);
    }
    private boolean eval140540725455968(BooleanExamplePredicates predicates) {
        return false;
    }
    private boolean eval140540725455936(BooleanExamplePredicates predicates) {
        return true;
    }
}

```

Figure 11: ADD-Lib: Auto-generated Java code for the predicates and BDDs

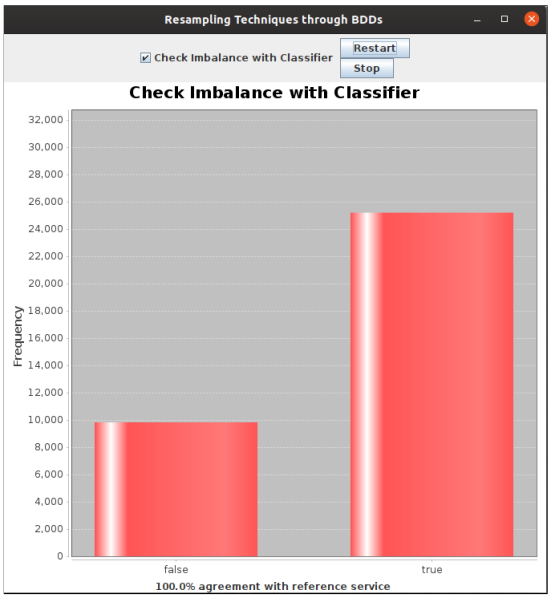

Figure 12: ADD-Lib: Final output from the comparison of results from the reference classifier and the actual composite classifier

Figure 10). This figure shows that the expected outcome is 100% in agreement with the actual outcome of the BDDs.

Since the reference classifier and the composite classifier are in 100% agreement, this ADD-Lib solution can be used to present medical domain experts with the most suitable choice of data-driven resampling techniques for the three different degrees of imbalance (mild, moderate and extreme). Since the structure of the BDDs is defined to be as general as possible, the findings of this research can be applied to a variety of domains and fields where the data being fed to the ML models is not balanced.

6 Conclusions and Next Steps

In this study, imbalanced datasets, specifically from the medical domain, were chosen to build and evaluate workflows to suggest the most appropriate data-driven resampling technique. Medical datasets were chosen because of the high occurrence of imbalanced datasets. Since the platform of choice for this study was ADD-Lib using the Binary Decision Diagram (BDD) representation, different workflows in the form of BDDs were created for different checks required for a dataset. The composite classifier made up of the individual BDDs was used to successfully evaluate the workflows on medical datasets. the results were confirmed by testing the workflows against a reference classifier. The expected outcome was 100% in agreement with the actual outcome of the BDDs. The BDDs, apart from being an efficient graphical representation of the decision boundaries, also were helpful in interpreting the logic behind the problem. This makes our solution inherently more interpretable, thus allowing the medical domain experts to make informed decisions and improve patient outcomes. Since the platform of choice supports low-

code/no-code methodology, it can be used, as shown in this study, by domain experts to build a custom BDD workflow for pre-checking their datasets for their machine learning (ML) models. The development of such a solution could be benefited by reusing the models developed for this research. The research can also be used in a more generalised, non-medical setting where the datasets being used need to satisfy a set of requirements before being used for ML models for training.

Future Work

This study can be extended in many ways for future iterations. The choice of platform as ADD-Lib was intentional, specifically to allow a wider development opportunity in terms of code reusability and code translation across the different CINCO products like DIME and Pyrus. The pipeline could be set-up such that the domain experts pre-check the dataset through the ADD-Lib platform, which will give them the exact techniques needed to balance their data, followed by more granular application and testing in Pyrus using the various Python libraries. The result could be displayed in a DIME web application to make the overall process more seamless for the domain experts, making it a 'one-stop-shop' for all their data needs. Also, since all these platforms are low-code/no-code, domain experts can themselves utilise their functionality to create a custom workflow with no/some programming expertise. Another direction could be to explore more techniques, such as algorithmic and hybrid resampling techniques, that are less dependent on the data and more on customising the ML models for a particular application.

Acknowledgements: Amandeep Singh received funding and research grants from Science Foundation Ireland (SFI) under Grant Number 18/CRT/6223 (SFI Centre of Research Training in AI). Olga Minguett is a full-time employee at Optum.

Bibliography

[Ake78] Akers. Binary decision diagrams. *IEEE Transactions on computers* 100:509–516, 1978.

[BFG+97] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, F. Somenzi. Algebric decision diagrams and their applications. *Formal methods in system design* 10:171–206, 1997.

[BFK+16] S. Boßelmann, M. Frohme, D. Kopetzki, M. Lybecait, S. Naujokat, J. Neubauer, D. Wirkner, P. Zweihoff, B. Steffen. DIME: a programming-less modeling environment for web applications. In *International Symposium on Leveraging Applications of Formal Methods*. Pp. 809–832. 2016.

[Bry86] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on* 100:677–691, 1986.

[Cav16] D. Cavan. Why screen for type 2 diabetes? *Diabetes research and clinical practice* 121:215–217, 2016.

[DDC18] S. Das, S. Datta, B. Chaudhuri. Handling Data Irregularities in Classification: Foundations, Trends, and Future Challenges. *Pattern Recognition* 81, 03 2018. doi:10.1016/j.patcog.2018.03.008

[EKN+17] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature* 542:115–118, 2017.

[Fis88] R. A. Fisher. Iris. *UCI Machine Learning Repository*, 1988. doi:10.24432/C56C76

[FXSS21] W. Fang, Q. Xue, L. Shen, V. S. Sheng. Survey on the application of deep learning in extreme weather prediction. *Atmosphere* 12:661, 2021.

[GCB+17] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, F. Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics* 36:1–12, Jul 2017. doi:10.1145/3072959.3073592

[GJMS19] F. Gossen, M. Jasper, A. Murtovi, B. Steffen. Aggressive aggregation: a new paradigm for program optimization. *arXiv preprint arXiv:1912.11281*, 2019.

[GMS20] F. Gossen, T. Margaria, B. Steffen. Towards Explainability in Machine Learning: The Formal Methods Way. *IT Professional* 22:8–12, 2020. doi:10.1109/MITP.2020.3005640

[GMZS19] F. Gossen, A. Murtovi, P. Zweihoff, B. Steffen. ADD-Lib: Decision Diagrams in Practice. Dec 2019. doi:10.48550/arXiv.1912.11308

[Gos21] F. Gossen. *Aggressive Aggregation (Domain-Specific) Program Optimisation with Algebraic Decision Diagrams*. 2021.

[GPC+16] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama* 316:2402–2410, 2016.

[GS20] F. Gossen, B. Steffen. Algebraic aggregation of random forests: towards explainability and rapid evaluation. *International Journal on Software Tools for Technology Transfer*, pp. 1–19, 2020.

[KFJB] N. Kuo, S. Finfer, L. Jorm, S. Barbieri. Synthetic Acute Hypotension and Sepsis Datasets Based on MIMIC-III and Published as Part of the Health Gym Project. doi:10.13026/P0TV-0R98

[KGD23] R. Krauss, M. Goli, R. Drechsler. Efficient Binary Decision Diagram Manipulation by Reducing the Number of Intermediate Nodes. In *2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*. Pp. 73–78. 2023.
doi:10.1109/DDECS57882.2023.10139373

[LEG+14] V. Liu, G. J. Escobar, J. D. Greene, J. Soule, A. Whippy, D. C. Angus, T. J. Iwashyna. Hospital Deaths in Patients With Sepsis From 2 Independent Cohorts. *JAMA* 312:90–92, 07 2014.
doi:10.1001/jama.2014.5804

[LFW19] T. Liu, W. Fan, C. Wu. Data for: A hybrid machine learning approach to cerebral stroke prediction based on imbalanced medical-datasets. 1, Nov 2019.
doi:10.17632/x8ygrw87jw.1

[LNA17] G. Lemaître, F. Nogueira, C. K. Aridas. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research* 18:1–5, 2017.

[Min22] O. Minguett. *Evaluation of Machine Learning classification techniques for handling class imbalance in medical datasets*. Thesis, M.Sc. in Artificial Intelligence, University Of Limerick, May 2022.

[MOSP20] C. T. Mesquita, A. Oliveira, F. L. Seixas, A. Paes. Infodemia, Fake News and Medicine: Science and The Quest for Truth. *International Journal of Cardiovascular Sciences* 33:203–205, Apr 2020.
doi:10.36660/ijcs.20200073

[MS09a] T. Margaria, B. Steffen. Business process modeling in the jABC: the one-thing approach. In *Handbook of research on business process modeling*. Pp. 1–26. IGI Global, 2009.

[MS09b] T. Margaria, B. Steffen. Continuous model-driven engineering. *Computer* 42:106–109, 2009.

[MS12] T. Margaria, B. Steffen. Service-orientation: conquering complexity with XMDD. In *Conquering complexity*. Pp. 217–236. Springer, 2012.

[MS19] T. Margaria, A. Schieweck. The digital thread in industry 4.0. In *International Conference on Integrated Formal Methods*. Pp. 3–24. 2019.

[MS20] T. Margaria, B. Steffen. Extreme model-driven development (xmdd) technologies as a hands-on approach to software development without coding. *Encyclopedia of Education and Information Technologies*, pp. 732–750, 2020.

[NCC+20] J. Ni, Y. Chen, Y. Chen, J. Zhu, D. Ali, W. Cao. A survey on theories and applications for self-driving cars based on deep learning methods. *Applied Sciences* 10:2749, 2020.

[NLKS18] S. Naujokat, M. Lybecait, D. Kopetzki, B. Steffen. CINCO: a simplicity-driven approach to full generation of domain-specific graphical modeling tools. *International Journal on Software Tools for Technology Transfer* 20:327–354, 2018.

[OLN22] Y. Otoum, D. Liu, A. Nayak. DL-IDS: a deep learning–based intrusion detection framework for securing IoT. *Transactions on Emerging Telecommunications Technologies* 33:e3803, 2022.

[RCBT22] P. Rajpurkar, E. Chen, O. Banerjee, E. J. Topol. AI in health and medicine. *Nature Medicine* 28:31–38, Jan 2022.
doi:10.1038/s41591-021-01614-0

[RIB+18] P. Rajpurkar, J. Irvin, R. L. Ball, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. P. Langlotz et al. Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists. *PLoS medicine* 15:e1002686, 2018.

[SAS+22] V. Sivalenka, S. Aluvala, Y. Sneha, K. Mannan, S. Farheen, E. Kumaraswamy. An empirical study of various face recognition and face liveness detection techniques and algorithms. In *AIP Conference Proceedings*. Volume 2418, p. 020056. 2022.

[TFL+21] R. Tsopra, X. Fernandez, C. Luchinat, L. Alberghina, H. Lehrach, M. Vanoni, F. Dreher, O. Sezerman, M. Cuggia, M. de Tayrac, E. Miklasevics, L. M. Itu, M. Geanta, L. Ogilvie, F. Godey, C. N. Boldisor, B. Campillo-Gimenez, C. Cioroboiu, C. F. Ciusdel, S. Coman, O. Hijano Cubelos, A. Itu, B. Lange, M. Le Gallo, A. Lespagnol, G. Mauri, H. Soykam, B. Rance, P. Turano, L. Tenori, A. Vignoli, C. Wierling, N. Benhabiles, A. Burgun. A framework for validating AI in precision medicine: considerations from the European ITFoC consortium. *BMC Medical Informatics and Decision Making* 21:274, Oct 2021.
doi:10.1186/s12911-021-01634-3

[WF04] J. Wardlaw, A. Farrall. Diagnosis of stroke on neuroimaging. 2004.

[WMZ+18] J. Wang, Y. Ma, L. Zhang, R. X. Gao, D. Wu. Deep learning for smart manufacturing: Methods and applications. *Journal of manufacturing systems* 48:144–156, 2018.

[Xie19] Z. Xie. Building Risk Prediction Models for Type 2 Diabetes Using Machine Learning Techniques. *Preventing Chronic Disease* 16, 2019.
doi:10.5888/pcd16.190109

[XSNK20] Z. Xu, D. Shen, T. Nie, Y. Kou. A hybrid sampling algorithm combining M-SMOTE and ENN based on Random forest for medical imbalanced data. *Journal of Biomedical Informatics* 107:103465, Jul 2020.
doi:10.1016/j.jbi.2020.103465

[ZS21] P. Zweihoff, B. Steffen. Pyrus: An Online Modeling Environment for No-Code Data-Analytics Service Composition. In Margaria and Steffen (eds.), *Leveraging Applications of Formal Methods, Verification and Validation*. Lecture Notes in Computer Science, p. 18–40. Springer International Publishing, Cham, 2021. doi:10.1007/978-3-030-89159-6$_2$