IDPP: Imbalanced Datasets Pipelines in Pyrus

Amandeep Singh[1,2][0000-0002-6371-0435]
Olga Minguett[1,3][0000-0002-0224-7619]

[1] University of Limerick, Ireland
[2] Centre for Research Training in Artificial Intelligence (CRT-AI)
[3] Optum
amandeep.singh@ul.ie, olgaminguett@gmail.com

Abstract. We showcase and demonstrate IDPP, a Pyrus-based tool that offers a collection of pipelines for the analysis of imbalanced datasets. Like Pyrus, IDPP is a web-based, low-code/no-code graphical modelling environment for ML and data analytics applications. On a case study from the medical domain, we solve the challenge of re-using AI/ML models that do not address data with imbalanced class by implementing ML algorithms in Python that do the re-balancing. We then use these algorithms and the original ML models in the IDPP pipelines. With IDPP, our low-code development approach to balance datasets for AI/ML applications can be used by non-coders. It simplifies the data-preprocessing stage of any AI/ML project pipeline, which can potentially improve the performance of the models. The tool demo will showcase the low-code implementation and no-code reuse and repurposing of AI-based systems through end-to end Pyrus pipelines.

Keywords: Low-code · imbalanced medical datasets · data resampling techniques · Pyrus · AI/ML-systems · Responsible AI.

1 Introduction

The combination of Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) algorithms has uncovered enormous potential and unprecedented problems in the ever-changing environment of software engineering. Software engineering principles need to adapt to developing and evolving AI-based systems. Our work addresses the need of responsible AI engineering and by leveraging the strengths of the Pyrus tool. Pyrus [17] is a Python-based, web-based, graphical modelling environment for ML and data analytics applications.

A particular aspect of fairness and access to advanced AI is to increase its accessibility to domain experts that are non-coders. This is increasingly important in medicine, health and natural science context. Prior work successfully used low-code/no-code approaches to address workflow in bioinformatics [5] [4], computational science [1] and paired with computational thinking, in education [8]. Those approaches share similar abstraction, encapsulation and coordination mechanisms to ours, however, their underlying tools were desktop or server oriented, the system used Java for the low-code part, and modelling was single user. Choosing Pyrus, we now support Python as implementation language, the system is cloud-based, and it supports distributed collaborative modelling, three core characteristics for the ease of adoption in modern interdisciplinary and distributed teams of natural and medical scientists and practitioners.

Tools like Tines[4] and H2O.ai 's*Hydrogen Torch*[5] are specifically low-code/no-code or support ML applications. Tines is a web-based no-code tool that uses workflows in the domain of cybersecurity. The workflows can be automated using different no-code snippets of generic components called 'actions' within automated workflows called 'stories'. Although Tines has a robust and easy to use interface, it lacks the ability to code specific 'actions' required for any complex data analysis or ML application. H2O.ai Hydrogen Torch is a ML/DL-specific, web-based, low-code/no-code tool that can be used by non-coders for their big-data needs. This platform can also be used to deploy ML pipelines and models, and it has API functionality for remote use. In comparison, Pyrus supports the features offered by Hydrogen Torch, and it is open source, thus it can be used without subscription fees. For a business/organisation looking to develop its own in-house AI models using their own proprietary data, sharing data with a third party and model training costs are the biggest issues.

From an application point of view, the AI/ML models, workflows and pipelines need to be explainable and reusable to allow for ease of future development and collaboration. The low-code/no-code paradigm helps by presenting the end user transparent, explainable and reusable ways to implement the AI/ML models in a reliable and fair way. We demonstrate how IDPP is a good solution to these problems on a real-world use-case where we show how to deal with the data imbalance problem for a selection of popular ML models, applied to the medical domain. To resolve class imbalance, data resampling techniques are used and all the IDPP modelling pipelines are developed in Pyrus using the low-code/no-code paradigm. This research extends the M.Sc. thesis of Olga Minguett [12], who chose the datasets and resampling methods. The new contribution is the IDPP tool: it concerns the restructuring of the code for the data analytics and the new model driven approach with ML pipelines in Pyrus [6].

In this paper, Sect. 2 describes the IDPP framework used in this research, Sect. 3 demonstrates the IDPP framework using a case study of imbalanced datasets from the medical domain, Sect. 4 concludes the paper.

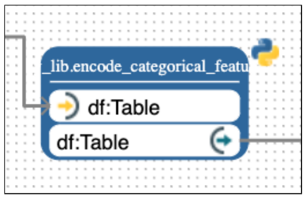
2 Framework Description

Pyrus is a Python-based, web-based, graphical modelling environment for ML and data analytics applications. Pyrus is also part of the larger CINCO family

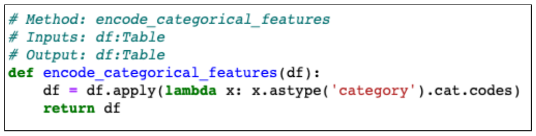
[4] https://www.tines.com/product

[5] https://h2o.ai/platform/ai-cloud/make/hydrogen-torch/

[6] All code, information on datasets used and the results are published on GitHub at: https://github.com/singhad/class_imbalance_pyrus



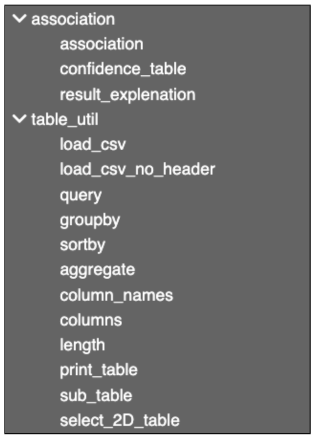
(a) The SIB in Pyrus



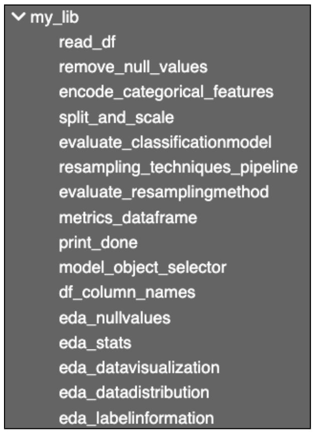
```
# Method: encode_categorical_features
# Inputs: df:Table
# Output: df:Table
def encode_categorical_features(df):
    df = df.apply(lambda x: x.astype('category').cat.codes)
    return df
```

(b) Python implementation and body annotation

Fig. 1: Pyrus: SIB representation of the encode_categorical_features() function from the pre-processing and transformation pipeline



(a) Default SIBs in Pyrus



(b) IDPP: New Custom SIBs

Fig. 2: IDPP and Pyrus: List of SIBs in the Ecore palette

of low-code/no-code development tools [13], and it is integral part of a Digital Thread solution [9] that enables inter-accessibility and reusability of the code-base for different applications and objectives. Pyrus models are data-flow models, they are the no-code graphical equivalent to programming workflows that orchestrate reusable Python functions. The Python functions are implemented in the renowned programming platform Jupyter[7] following the OTA (One Thing Approach) paradigm [10]. Special signature annotations added to these functions enable their identification by the Pyrus web-based orchestration tool. The code generated by Pyrus from the pipelines is also stored and executed in Jupyter. This separation of the low-code development of functionalities and no-code orchestration via modelling is based on MDE principles [11].

Fig. 3 shows a screenshot of the Pyrus web-based development environment. Each Python function implemented in Jupyter is represented in Pyrus as a collection of taxonomically grouped Service Independent Building blocks (SIBs) in

[7] https://jupyter.org

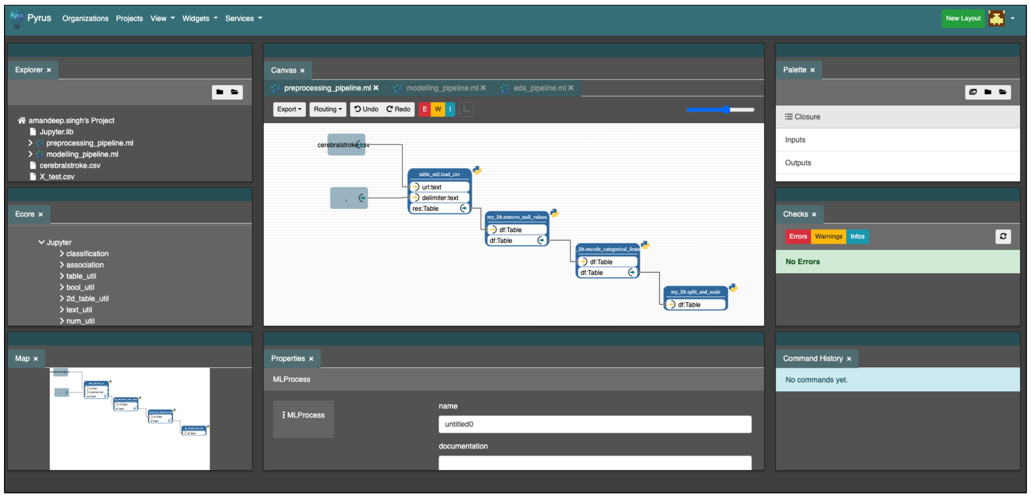


Fig. 3: Pyrus: View of the web-based development environment

its Ecore section, containing the SIBs palettes. Fig. 1 shows the annotated code in Jupyter in Fig. 1b and its representation as a SIB in Pyrus, see Fig. 1a. A summary of the SIBs available in the Ecore section of IDPP is shown in Fig. 2.

3 Use Case and Demonstration

ML classification algorithms assume that the classes are balanced, but this is rarely the case in any real-life data. Class imbalance happens when one or more of the classes/categories in a dataset are not well represented and hence are thought to be outliers, noise or anomalies by the ML algorithms during their training process. This is a challenge for the ML algorithms, as the underrepresented categories will be ignored or misclassified. The problem is exacerbated when these algorithms are used as applications in real-life settings and produce biased or wrong results. According to [16], most ML classification algorithms assume that the classes are balanced and that the cost of miscalculation is the same for any class. However, for diagnosing conditions, improving prognostics, accurate patient monitoring and in personalised medicine, the cost of misdiagnosing a patient is significantly higher. For example, for the classification of tumours as malignant or not-malignant [2], the cost of miscalculation is very different. To resolve class imbalance, resampling techniques are used, such as data-resampling techniques that modify the training dataset in order for the ML models to have equal representation of the minority class.

3.1 Python Pre-requisites

We chose Python as the programming language. The two central packages used for the programming tasks are scikit-learn [14], used for data cleaning, modelling and evaluation, and imbalanced-learn [6], used to apply the different data-driven resampling techniques on the datasets.

3.2 Datasets

We chose these three datasets for this study:

1. Cerebral Stroke Dataset [7]: This dataset was created to aid in detection of a stroke using classification algorithms. It has 12 features containing 43.4K observations out of which 783 observations are labelled to be stroke.
2. Diabetes Dataset [15]: This dataset was extracted from the Behavioral Risk Factor Surveillance System (BRFSS) 2014 dataset that was published by the CDC [8]. The BRFFS 2014 dataset contained survey collected responses from over 400,000 people on health-related risk behaviours, chronic health conditions, and the use of preventative services, conducted since 1984. The extracted Diabetes dataset has 22 features, 254.6K observations with the target variable having 2 classes - 0 for no diabetes, 1 for diabetes.
3. Sepsis Dataset [3]: This dataset was created in the Computing in Cardiology Challenge from Physionet 2019 with the goal of early detection of sepsis. The data was sourced from ICU patients in three separate hospital systems.

3.3 Data Pre-processing and Transformation

To remove missing values and encoding of categorical variables, different approaches are employed for the three different datasets.

1. Cerebral Stroke dataset: only two features have missing values - *BMI* and *Smoking Status*. For the *BMI* feature, the missing values are imputed with the modal value, and for the *Smoking Status* feature, the missing values are categorised into a new label named 'Unknown'. The categorical features in this dataset are encoded using the Pandas package.
2. Diabetes dataset: it has no missing values or categorical features. The dataset has 253680 rows and 22 columns with an imbalance of target label of 16.19%. Since the dataset is very large for this study, a subset of the dataset is taken by keeping the imbalance percentage constant. The subset dataset we use has 41075 rows and 22 columns.
3. Sepsis dataset: the features with more than 70% missing values are deleted. The remaining missing values in the features are imputed using the median values. There are no categorical features in this dataset.

The datasets are split into a ratio of 80:20 for training and testing respectively. The stratified splitting method is used. For outliers, the RobustScaler() transform function is used for feature scaling to remove the median values and perform scaling of data between the 1st and 3rd quartile.

[8] https://www.cdc.gov/brfss/annual_data/annual_2014.html

3.4 Experiments

Five classification algorithms were chosen: 1) Support Vector Machine (SVM), 2) Decision Tree (DT), 3) Gaussian Naïve Bayes (GNB), 4) K-Nearest Neighborhood (KNN), 5) Logistic Regression (LR).

Three types of data-driven resampling techniques were applied on the datasets: undersampling, oversampling, and hybrid techniques. The following specific data-driven resampling techniques were selected in the experiments:

1. Oversampling: RandomOverSampler, SMOTE, SMOTENC, BorderlineSMOTE, SVMSMOTE, KMeansSMOTE, ADASYN
2. Undersampling: RandomUnderSampler, ClusterCentroids, NearMiss, InstanceHardnessThreshold, TomekLinks, CondensedNearestNeighbour, AllKNN, EditedNearestNeighbours, RepeatedEditedNearestNeighbours, OneSidedSelection, NeighbourhoodCleaningRule
3. Combined/Hybrid: SMOTEENN, SMOTETomek

3.5 Pyrus Pipelines

The original code was transformed according to the OTA paradigm for modularization and reuse, and each SIB was annotated with special signature comments for the Pyrus orchestrator to recognise the functions in the pipelines. The code was then stored and implemented on Jupyter, and GUI-based pipelines were modelled in Pyrus. The Pyrus pipelines are depicted in Fig. 4, 5 and 6.

The performance of the algorithms on the datasets before and after resampling was evaluated using the metrics accuracy, precision, recall, f1 score, number of occurrences, predictions count, confusion matrix and area under the curve. The precision, recall and f1 score metrics were plotted, and the results were stored as a CSV file for each classification algorithm.

3.6 Results

Experiments were conducted on the three selected datasets. Fig. 4 shows the exploratory data analysis (EDA) pipeline to get the overview and basic statistics of the datasets. Fig. 5 shows the pre-processing and transformation pipeline used to clean the datasets and segment them into training/testing sets for ML models. Fig. 6 shows the modelling and evaluation pipeline used to apply and evaluate ML models on the selected datasets.

The highest and lowest scoring results for each of the datasets based on the f1 score metric are shown in table 1. The results are summarised as follows:

1. The Cerebral Stroke dataset has the most imbalanced class ratio. The best results were obtained by the KNN and DT models when undersampling and oversampling techniques were used. Although overall, oversampling techniques performed better. The worst results were obtained when a set of undersampling techniques were used.

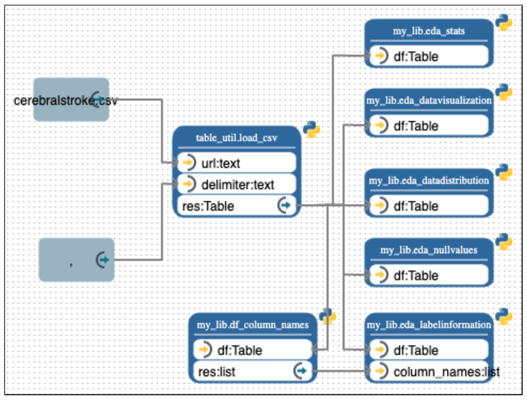


Fig. 4: Pyrus: Exploratory Data Analysis (EDA) Pipeline

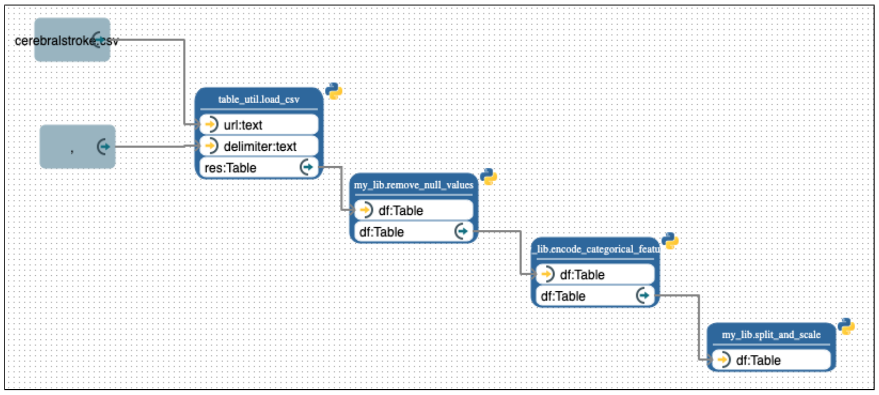


Fig. 5: Pyrus: Pre-processing and Transformation Pipeline

2. The Diabetes dataset had a mild class imbalance ratio and here too oversampling techniques performed better overall than undersampling techniques. For this dataset, the best algorithms were KNN and DT. The worst results were obtained when a set of undersampling techniques were used.
3. The Sepsis dataset had a moderate class imbalance ratio. Here the undersampling and oversampling techniques performed equally well. For this dataset, the best algorithm was DT. The worst results were obtained when a set of undersampling techniques were used.

Across all three datasets, the hybrid techniques had the best overall performance, with the least variance in f1 scores for different models. The best model for hybrid techniques was DT.

Across all datasets, specifically the TomekLinks/OneSidedSelection (undersampling), RandomOverSampler/KMeansSMOTE (oversampling) and SMOTE-Tomek (hybrid) methods performed the best.

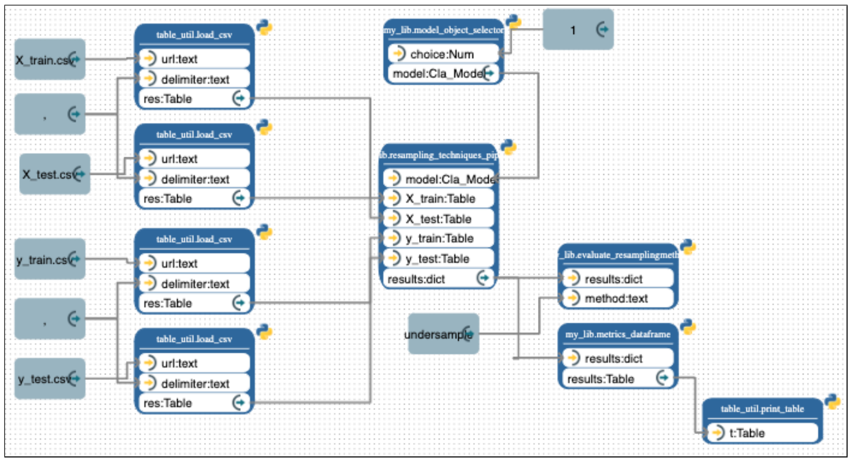


Fig. 6: Pyrus: Modelling and Evaluation Pipeline

Table 1: Results for all datasets - sorted by f1 Scores

Dataset	Technique	Model	Method	f1 Score	Accuracy	Precision	Recall
Cerebral Stroke Dataset	Undersampling	KNN	TomekLinks	**0.9906**	0.9812	0.9820	0.9992
		KNN	OneSidedSelection	**0.9906**	0.9813	0.9820	0.9992
		DT	NearMiss	*0.3702*	0.2376	0.9798	0.2282
	Oversampling	DT	RandomOverSampler	0.9831	0.9668	0.9821	0.9842
		DT	KMeansSMOTE	0.9760	0.9532	0.9832	0.9689
		GNB	ADASYN	0.8247	0.7057	0.9934	0.7049
	Hybrid	DT	SMOTETomek	0.9736	0.9486	0.9829	0.9645
		GNB	SMOTEENN	0.8264	0.7082	0.9937	0.7073
Diabetes Dataset	Undersampling	KNN	TomekLinks	**0.9111**	0.8422	0.8843	0.9596
		KNN	OneSidedSelection	**0.9111**	0.8421	0.8844	0.9393
		DT	ClusterCentroids	*0.3769*	0.3221	0.9020	0.2382
	Oversampling	SVC	KMeansSMOTE	0.8963	0.8220	0.8992	0.8934
		DT	RandomOverSampler	0.88	0.7945	0.8737	0.8737
		GNB	ADASYN	0.7596	0.6567	0.9558	0.6303
	Hybrid	DT	SMOTETomek	0.8768	0.7910	0.8900	0.8639
		KNN	SMOTEENN	0.7570	0.6526	0.9508	0.6289
Sepsis Dataset	Undersampling	DT	OneSidedSelection	**0.9736**	0.9510	0.9736	0.9736
		DT	TomekLinks	**0.9730**	0.9499	0.9734	0.9725
		SVC	ClusterCentroids	*0.4018*	0.2993	0.9634	0.2539
	Oversampling	DT	RandomOverSampler	**0.9736**	0.9510	0.9727	0.9745
		LR	ADASYN	0.8549	0.7601	0.9734	0.7620
	Hybrid	DT	SMOTETomek	0.9601	0.9270	0.9736	0.9470
		KNN	SMOTEENN	0.8399	0.7390	0.9738	0.7384

Overall, hybrid techniques perform the best with the least variance in f1 scores, oversampling techniques ranked second-best: with many higher f1 scores than hybrid techniques, but with more variance. Undersampling techniques ranked the lowest of the three types, with some high scores but a lot of variance in f1 scores. This result is in agreement with the established understanding that more data points are always better than fewer data points even when the resulting dataset is completely balanced. By design, undersampling techniques remove data points, which generally results in loss of information compared to oversampling/hybrid techniques that append more data points.

4 Conclusions

With IDPP we demonstrate that the low-code/no-code pipelines for imbalanced datasets in Pyrus serve as an embodiment of 'Responsible AI' concerning transparency, fairness, explainability, reliability and reusability of the AI/ML models and IDPP pipelines themselves. IDPP uses the web-based, low-code/no-code graphical modelling environment of Pyrus for AI/ML applications. We applied IDPP to imbalanced datasets, showing on 3 imbalanced medical datasets the performance of different data-driven resampling techniques in combination with a selection of ML classification algorithms.

The low-code Pyrus pipelines were easy to create and reuse. The development time of the pipelines was greatly reduced by using a web- and GUI-based tool. Pyrus was used to build the data-flow pipelines using SIBs generated from annotations in the Python code. With this low-code/no-code approach, future users can reuse the existing IDPP pipelines and SIBs by simply selecting them from the Ecore palette section in Pyrus, without the prerequisite of proficiency in programming. This ensures superior understandability of the logical steps in the pipeline w.r.t. the code based approach. IDPP's end-to-end Pyrus pipelines offer a variety of techniques, models and methods to rectify data imbalance in different scenarios without the need for redeveloping custom pipelines and AI/ML models from scratch for each use-case.
A challenge faced by IDPP and any low-code/no-code approach is the dependency of Python libraries on the Python kernel version. If the version of Python required by the libraries does not match the version of Python kernel used by Jupyter for the IDPP or Pyrus orchestration, the pipeline will not execute. It may help to re-deploy Pyrus framework using the most recent versions of Python and other supported packages. Ultimately, software obsolescence is inevitable, and it is essential to keep pace with newer versions, tools and technology.

Acknowledgments

This research was partially funded by Science Foundation Ireland (SFI) under Grant Number 18/CRT/6223 - SFI Centre of Research Training in AI.

References

1. Al-Areqi, S., Lamprecht, A.L., Margaria, T.: Constraints-driven automatic geospatial service composition: Workflows for the analysis of sea-level rise impacts. In:

Computational Science and Its Applications – ICCSA 2016. pp. 134–150. Springer International Publishing, Cham (2016)

2. Devarriya, D., Gulati, C., Mansharamani, V., Sakalle, A., Bhardwaj, A.: Unbalanced breast cancer data classification using novel fitness functions in genetic programming. Expert Systems with Applications **140**, 112866 (Feb 2020), bluehttps://www.sciencedirect.com/science/article/pii/S0957417419305767
3. Kuo, N., Finfer, S., Jorm, L., Barbieri, S.: Synthetic acute hypotension and sepsis datasets based on mimic-iii and published as part of the health gym project, bluehttps://physionet.org/content/synthetic-mimic-iii-health-gym/1.0.0/
4. Lamprecht, A.L., Margaria, T., Steffen, B.: Seven variations of an alignment workflow-an illustration of agile process design and management in bio-jeti. In: ISBRA 2008, Atlanta. pp. 445–456. Springer (2008)
5. Lamprecht, A.L., Margaria, T., Steffen, B., Sczyrba, A., Hartmeier, S., Giegerich, R.: Genefisher-p: variations of genefisher as processes in bio-jeti. BMC bioinformatics **9**(4), 1–15 (2008)
6. Lemaître, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. J. of Machine Learning Research **18**(17), 1–5 (2017), bluehttp://jmlr.org/papers/v18/16-365.html
7. Liu, T., Fan, W., Wu, C.: Data for: A hybrid machine learning approach to cerebral stroke prediction based on imbalanced medical-datasets **1** (Nov 2019), bluehttps://data.mendeley.com/datasets/x8ygrw87jw/1
8. Margaria, T.: From computational thinking to constructive design with simple models. In: Leveraging Applications of Formal Methods, Verification and Validation. Modeling: ISoLA 2018. pp. 261–278. Springer (2018)
9. Margaria, T., Schieweck, A.: The digital thread in industry 4.0. In: International Conference on Integrated Formal Methods, LNCS 11918. pp. 3–24. Springer (2019)
10. Margaria, T., Steffen, B.: Business process modeling in the jabc: the one-thing approach. In: Handbook of research on business process modeling, pp. 1–26. IGI Global (2009)
11. Margaria, T., Steffen, B.: Continuous model-driven engineering. Computer **42**(10), 106–109 (2009)
12. Minguett Pirela, O.M.: Evaluation of Machine Learning classification techniques for handling class imbalance in medical datasets. M.Sc. in Artificial Intelligence, University of Limerick (May 2022)
13. Naujokat, S., Lybecait, M., Kopetzki, D., Steffen, B.: 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)
14. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, e.a.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
15. Xie, Z.: Building risk prediction models for type 2 diabetes using machine learning techniques. Preventing Chronic Disease **16** (2019)
16. Xu, Z., Shen, D., Nie, T., Kou, Y.: A hybrid sampling algorithm combining m-smote and enn based on random forest for medical imbalanced data. Journal of Biomedical Informatics **107** (Jul 2020)
17. Zweihoff, P., Steffen, B.: Pyrus: An online modeling environment for no-code data-analytics service composition. p. 18–40. LNCS 13036 (2021)