Fire Technology

Manufactured in The United States
https://doi.org/10.1007/s10694-024-01655-0

Semantic Enrichment of a BIM Model Using Revit: Automatic Annotation of Doors in High-Rise Residential Building Models Using Machine Learning

Soheila Bigdeli, Research Group Combustion, Department of Structural Engineering and Building Materials, Ghent University, Ghent, Belgium*
Pieter Pauwels, Department of the Built Environment, TU Eindhoven, Eindhoven, The Netherlands
Steven Verstockt, IDLab, Ghent University - IMEC, Ghent, Belgium
Nico Van de Weghe, Department of Geography, Ghent University, Ghent, Belgium
Bart Merci, Research Group Combustion, Department of Structural Engineering and Building Materials, Ghent University, Ghent, Belgium

Received: 14 February 2024/**Accepted:** 20 September 2024

Abstract. This study explores the potential of automated fire safety conformity checks using a BIM tool. The focus is on travel distance regulations, one of the major concerns in building design. Checking travel distances requires information about the location of exits. Preferably, the Building Information Model (BIM) of the building should contain such information, and if not, user input can be requested. However, a faster, yet still reliable and accurate, methodology is strived for. Therefore, this study presents an automated solution that uses machine learning to add the required semantics to the building model. Three algorithms (Bagged *K*NN, SVM, and XGBoost) are evaluated at a low Level of Detail (LOD) BIM models. With precision, recall, and F1 scores ranging from 0.87 to 0.90, the model exhibits reliable performance in the classification of doors. In a validation process with two separate sample buildings, the models accurately identified all 'Exits' in the first building with 94 samples, with only 5 to 6 minor misclassifications. In the second building, all models- with the exception of the SVM - correctly classified every door. Despite their theoretical promise, oversampling techniques do not enhance the results, indicating their inherent limitations.

Keywords: Building information model, Semantic enrichment, Automated code compliance checker, Machine learning, Fire safety codes

* Correspondence should be addressed to: Soheila Bigdeli, E-mail: soheila.bigdeli@ugent.be
These authors contributed equally to this work.

Published online: 08 October 2024

1. Introduction

Building codes and regulations specify various rules that focus primarily on the health and safety of occupants [1]. Verifying that a building meets all of these rules is a tedious, error-prone and time consuming task that requires considerable human effort [2]. A collaborative research effort between Business and Economic Research Limited (BERL), the Building Research Association of New Zealand (BRANZ), and Price Waterhouse Coopers (PWC) found that a 10% improvement in efficiency within the construction sector could lead to a notable 1% impact on the overall Gross Domestic Product (GDP) [3]. The results of a survey conducted in the U.S. show that on a typical project, approximately 200 hours are spent by architects, engineers, contractors and owners on building code compliance reviews [4]. Although automation of building code compliance verification long predates the advent of Building Information Modeling (BIM) technologies [5], the ability to automate the code compliance check process has become more prominent, with the opportunities provided by BIM. In an effort to reduce the amount of work required while maintaining (or even improving) the quality of checks, many researchers [6–9] have sought to leverage the power of BIM to automate and computerize this process.

BIM models often contain some but not all of the specific data required by local regulations [10]. Following any potential design alteration, the user must allocate time during each iteration of the design review process to re-enter missing data or rectify erroneous data. Most commercially available Automated Code compliance Checkers (ACCs), such as Solibri Model Checker (SMC) [11] or SmartReview [12], require the user to manually supplement the data needed to check a specific rule [13, 14]. Automating this process helps to reduce repetitive work and human error[15]. Semantic Enrichment (SE) is a promising solution for preparing model data to provide or correct the information required by regulations [16]. SE is a (semi-)automatic process of adding domain- or application-specific information to the building model and has received much attention in the last decade [17–20]. Simeone et al., [21] enriched the heritage BIM models with non-geometric data using Semantic Web technologies. FORNAX as an external library platform, developed and maintained by novaCITYNETS Pte. Ltd. Singapore, contains building objects enriched with high-level semantics required by code compliance checkers, but the provided attributes are limited [14]. In [22], Châteauvieux et al. used domain-specific rule sets to add semantics needed for the noise and vibration analysis of timber models, such as the type of element in the junction. Recent research, [23], with an extensive study on the number of scientific papers on semantic enrichment of BIM models in the AEC industry, shows that the number of researches is still not very high and that the topic still lacks a “formal definition of the process”, “possible applications”, “contributions” and “computational approaches”. This study aims to address these gaps by proposing a simplified computational approach using basic machine learning techniques. A potential application of the enriched model is demonstrated in automatic evacuation route rule checks.

While information is at the heart of BIM, many data remain implicit within the models. Many studies have focused on Rule Inferencing (RI) for semantic enrichment of BIM models (e.g., [17, 18, 24], and [25]). However, especially with the recent innovative and complex building designs, not everything can be translated into explicit rules. With the capability to handle uncertainties and exceptions [26], Artificial Intelligence (AI) and Machine Learning (ML) techniques seem to be well-positioned to uncover the hidden insights within BIM models. Even though AI and ML are expanding across many sectors, their adoption in the construction industry has been slow [27]. However, recent successful studies underscore a promising research trajectory, particularly in the semantic enrichment of BIM models.

In their research, Jin et al. [28] introduced an unsupervised learning method to analyze IFC-based BIM data, focusing on the relationships between building spaces. By extracting BIM data properties and using an adapted affinity propagation algorithm, their approach effectively identified standard space functions using a dataset of a 20-story building with 595 spaces. Calzado et al. [29] delved into the automatic labeling of rooms in housing units based on geometry using ML algorithms, a traditionally manual task in BIM. Based on two Autodesk Revit models, each with more than 200 units, they used three methods: decision trees, logistic regression, and neural networks. To perform a sensitivity study, they extracted data both directly and via the Revit application programming interface (API). With this enriched dataset, both logistic regression and neural networks achieved an accuracy of 80%-90%. Koo et.al. [30] used 3D geometric deep neural networks, specifically MVCNN24 and PointNet25, to classify Building Information Modeling (BIM) elements, with a focus on door and wall subtypes. Using a 7:3 training-to-testing data ratio, MVCNN outperformed with accuracies of 92% for doors and 95% for walls, whereas PointNet faced challenges due to resolution issues from selective point cloud use. Additionally, when classifying road infrastructure BIM elements, MVCNN again proved superior with a 98% accuracy compared to PointNet's 83%. Using a Graph Convolutional Network (GCN), Collins et al.[31] achieved up to 85% accuracy when categorizing BIM objects from a dataset of 22 IFC files,containing a variety of structural elements, equipment, and interior furnishings. Kim et al. [32] employ a multi-view method, integrated into an Autodesk Revit plugin, to train a Convolutional Neural Network (CNN) on 2D images of objects from 32 angles, encompassing 820 objects across ten classes. They explore a two-step classification, initially sorting objects into broader categories before sub-classifying certain groups. In a very recent study, Lou et al. [33] addressed the interoperability challenges within Building Information Modeling (BIM) by proposing a two-branch geometric-relational deep learning framework. This approach integrated both geometric and relational data, enhancing BIM object classification capabilities. Such a method not only augmented the semantic richness of BIM objects but also promised to mitigate the long-standing challenge of manual model checks, thereby elevating the practical utility of enriched BIM models.

The implementation of ML approaches for semantic enrichment of BIM models offers the advantage of a low level of detail (LOD) required for automated design

checks. This is attributed to the capability of ML to infer and extract relevant data from the existing model, alleviating the need for extensive effort spent on model development. Given that early stages of design phases often operate with a relatively low LOD in models, a method capable of efficiently identifying essential architectural features presents significant value. This underpins the rationale of the present study, which seeks to assess the viability of an approach geared towards automatic annotation of elements within high-rise building models.

Most of the aforementioned studies use complex deep learning models that learn mainly from the shape of BIM objects, some of which entail extensive pre-processing and considerable computational overhead, making them unsuitable for many real-world applications [34]. In contrast, we use ML techniques, such as *K*NN and XGBoost, which are significantly faster and more efficient. By prioritizing low-level geometric attributes and deliberately excluding shape features, our methodology remains resilient even when the geometric representation in BIM may be imprecise or incomplete with low LOD.

The core this study is part of a larger effort that focuses on the automation of rules related to evacuation routes. Compliance to these rules play an important role in getting building permit and ensuring occupant safety. A crucial piece of information for this automation was the identification of exits on each floor, which was missing. Therefore, this study investigated the automatic addition of this information to the BIM model. To achieve this the current study presents a novel workflow that integrates simple geometrical features to train a machine learning model to do this. While the individual components of our approach are not new, their innovative combination within this workflow represents a significant advancement. This methodology not only demonstrates high performance and computational efficiency but also facilitates the enrichment of BIM models with low LOD. To realize the primary goal of current paper-a proof of concept for employing ML techniques in the automated annotation of BIM models with low LOD, several objectives are addressed. First of all, a comparative analysis of different ML techniques is undertaken to automatically annotate 'Exits' on BIM models of buildings. 'Exits' refer to the doors that allow leaving the floor towards emergency staircases. Given an imbalanced dataset, different oversampling techniques are tested and evaluated. The importance of proper implementation when combining cross validation with oversampling is subsequently underscored. Finally, the outcomes of the automated approach are subsequently bench-marked against manual human-based annotations.

The remainder of the paper is structured as follows. Section 2 first gives an overview of the general methodology used to develop the automated code compliance checker. Then an overview of the dataset and its preparation is given. Later, the selected ML models and their hyperparameters are described in detail, and finally the oversampling techniques we used are explained. Subsequently, Section 3 delves into a detailed discussion of the obtained results. Following that, Section 4 extends the discussion by exploring considerations beyond our methodology implementation, shedding light on unaddressed aspects that could impact the results. Finally, Section 5 presents the conclusions.

2. Methodology

This research is part of a study focused on leveraging Eastman's four-step process [2] to automate rules related to the evacuation route in Belgium's national fire safety codes. The focus of the current work is on the 'normalization' or 'Preparation of the Building Model Data' stage, the second step from the Eastman's proposed process, Fig. 3, however, it is also important to briefly explain the other steps as they have a significant impact on what needs to be enriched. Therefore, a brief overview of the main tasks of each step is given first in Section 2.1. As building construction is a step-by-step process, more information is added to the model as the project progresses. Typically, designers and engineers model the building collaboratively, using different software to produce models with multiple aspects or disciplines. These models are then merged into a coordination model, often using a Common Data Environment (CDE). before and during this process, it is crucial to set exchange requirements and to specify the information needed to be included and exchanged according to the ISO 19650 standard. However, it is often the case that there are no exchange requirements or other agreements made, which means that the required information is not modeled and therefore not available. As fire safety engineers are not normally involved in the BIM process, this becomes even more problematic for the information they require. For this reason, current BIM models usually lack the necessary data (spatial, topological, geometric, graphical, and nongraphical) for evacuation route rules to check compliance with building regulations. Although user input could be queried to add the data, this study aims to propose an automated technique to enrich models with this information, more specifically the 'Exits'.

Fig. 1 illustrates the general proposed process of model preparation for rule execution. As shown in the figure, module 1 accesses the object-oriented BIM model (which is the architectural model here) to extract information and identify structures and relationships between building objects based on regulation concepts, such as exits or boundaries of a fire compartment, and retrieves the properties, such as geometry data or fire rating, to prepare a dataset for rule execution. If the required data are not found in the model, the data can be provided by engineers or prompting a query for user input. Although these remain the preferred approaches, alternatives such as deriving data from available data, as formalized by [2], or automated SE techniques (module 2) can be used to add them to the model. As asking for user input means extra cost and effort, the current study chooses the second option (module 2), in this case, ML techniques, to find the missing information in an automated manner. After preparing the data set for the execution of the rules, the Transformer module (module 3) is responsible for developing the enriched BIM model by reconnecting to the BIM model and adding new classification attributes to the original BIM model.

The Autodesk Revit tool was chosen for this study because it is one of the most widely used tools among designers and architects. The information associated with each building element is retrieved via the Revit API. This is done in module 1 (see Fig.1). Using the capabilities of the Revit API, a plug-in is developed in the C# language in the .Net Framework (version 4.7.2). C# is an object-oriented pro-

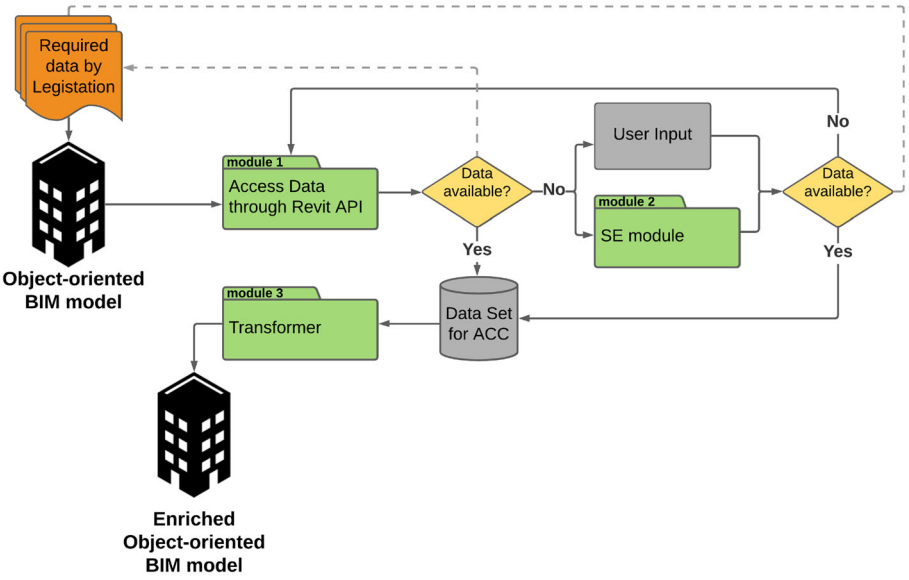

Figure 1. General overview of the data enrichment process.

gramming language created by Microsoft that operates on the .Net Framework. The .Net Framework is a comprehensive development platform for building and executing applications on Windows. Being a strong typed language, C# is a logical choice for developers, particularly when creating plugins for Revit.

For the rules in Table 1, the apartments and exits need to be defined. While exit information is usually missing, apartment units can be identified using the 'Area' drop-down menu (Area Plan) on the 'Architecture' tab, 'Room & Area'. Fig. 2 shows some examples of emergency staircases and exits in a floor plan of existing high-rise buildings in Belgium. The pattern in most high-rise buildings is quite similar (exits are the two doors (depicted with dotted purple lines in Fig. 2) connecting an air-lock space to the stairwell), so machines could be trained to produce human-like recognition and responses.

A supervised ML approach is employed, necessitating the use of training and validation samples. Five distinct features are defined for each door within the dataset. These features include the width of the door, the Euclidean distance to the adjacent door, the Euclidean distance to the nearest staircase, the travel distance to the adjacent door, and the travel distance to the nearest staircase. The training and validating data is extracted from [35], which includes data from five high-rise buildings containing total of 2521 doors. The following section provides an overview of the dataset preparation process and details the selection, configuration, and evaluation of the chosen machine learning models. Given the selection of Revit as the BIM tool utilized in this study, the following section provides a comprehensive explanation of the data preparation using the Revit API. However,

Table 1
Selected Belgian Fire Safety Code Clauses for Evacuation Route Analysis

Clause exact text	Simplified text	Element	Parameter
Annex 4/ 4.4.4.1: No point of a compartment shall be at a distance greater than: For rooms or groups of rooms used at night time:	No point of the fire compartment in a residential building shall be further than:	- Area -Door (exit or None) - Stair - Wall - Room	- Level - Geometrical data
_ 30 m from access to the nearest stairs or exit.	_ 30 m between the furthest point of the furthest apartment to the nearest staircase or the nearest exit.		
_ 60 m from access to the second staircase or the second exit.	_ 60 m between the furthest point of the furthest apartment to the second staircase or the second exit.		

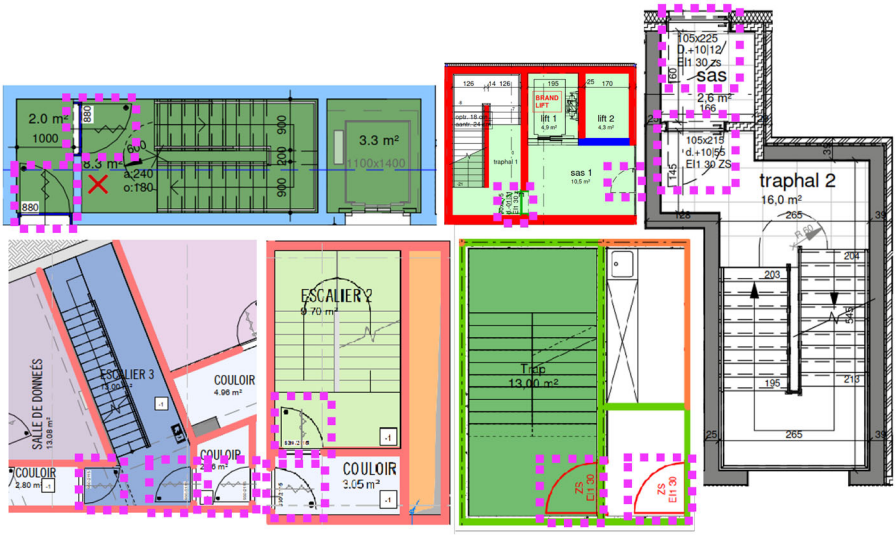

Figure 2. Examples of exits and stairs in high-rise building plans in Belgium.

the features described in Section 2.2 can be extracted from any other object-oriented BIM tool capable of providing geometric information or IFC files.

2.1. Eastman Four Step Process

The main components of the rule checking process were introduced by Eastman [2] in four general phases, shown in Fig. 3. It begins by focusing on the content of the building code and how to convert this into formats that are understandable for the computer, i.e., 'Translation of the Rules to a Machine-Readable Language'. The next step is to investigate whether the model can provide the required information, and if not, what techniques need to be implemented to add data to or retrieve data from the building model data, i.e., 'Preparation of the Building Model Data'. The third step is rule execution, where the rules are applied using the prepared building model data, i.e., 'Execution of Rules'. The last phase is reporting the outcome of the design check, i.e. 'Reporting of the Checking Results'.

Step 1: Translation of the Rules into a Machine-Readable Language The Belgian Fire Safety Code for high rise buildings has been thoroughly analyzed in terms of evacuation route requirements and the relevant codifiable clauses investigated in this study and are parameterized to identify the important elements from the BIM model and the required corresponding properties. Building code parametrization, also known as code-BIM matching [36], is a challenging task with existing semi-automatic approaches [37] being difficult to generalize and not scalable. Previous attempts using ML algorithms achieved about 76% accuracy , [36], but failed to extract the necessary element properties. To achieve near-perfect matching, manual parameterization is utilized in the current study.

Table 1 summarizes the selected rules for automation in this study. The table reflects the numbering and wording of the regulations, with a simplified text specifically tailored for residential buildings, together with the corresponding required building elements and properties obtained by manual parameterization.

Solihin and Eastman [38] addressed the complexities inherent in rules by classifying rules into four main categories based on the complexity of the data processing required to evaluate them. (1) Rules that require a single or small number of explicit data (which involves direct comparisons or validations checking explicit attributes or entity references within the BIM dataset), (2) Rules that require simple derived attribute values, (which involves some basic processing to derive new values from exiting data), (3) Rules that require extended data structure (which refer to rules that require complex geometric or spatial analysis, requiring the construction of additional relationships or networks to accurately evaluate compliance.) and (4) Rules that require a 'proof of solution' (which are the most complex category, require a performance-based analysis, often involving simulations or advanced calculations to demonstrate that the design meets the specified criteria.) According to this classification, the rules coded in the current study, see Table 1, can be categorised into class 2. The required data are not explicit in the BIM models but can be derived through additional coding; i.e., identifying the compartment's furthest points and calculating travel distances towards the 'Exits'.

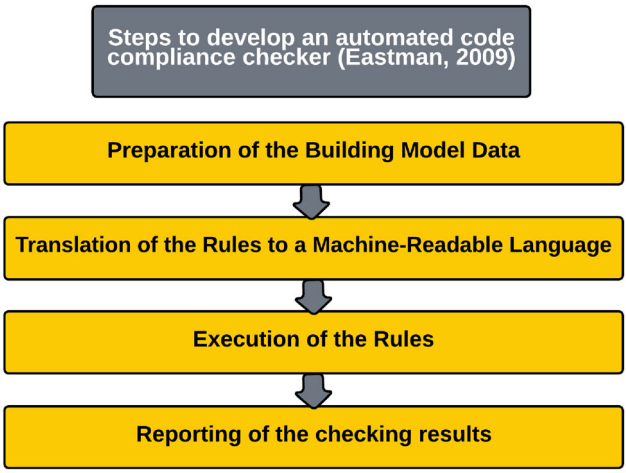

Figure 3. Eastman's four steps for developing an automated code compliance checker [2].

Given the geometric layout of most buildings, it is logical to consider corner points as the possible candidates for furthest points within a compartment, as corners are the points where walls intersect, creating the geometry boundaries. Calculating the travel distances from these points to the exits, though complex, is feasible using tools available within the BIM software, such as Revit. Additionally, it is necessary to identify which doors function as 'Exits'. Although accessing this information might seem straightforward through user input, this study, aiming to automate as much as possible and reduce user work, proposes a novel solution to automatically add this data to the BIM model.

Step 2: Preparation of the Building Model Data

In this step, which forms the core of the present work, the building model is prepared for the execution of the rules. Regardless of the classification of the rule - whether it belongs to Class 1, Class 2, Class 3 or Class 4- all necessary information must be available and accessible during the execution of the rule check in order to ensure an accurate evaluation. While corner points and travel distances, including their lengths, can be derived from the BIM model dataset through coding, automating the extraction of 'Exits' information solely through coding, without the need for user input, remains particularly challenging, if not impossible. This study proposes to use a machine learning application to enrich the BIM model with this missing data.

Step 3: Execution of Rules

The third step in the development of ACCs is the implementation of rules. The rules are converted into If-then-Else logic and coded using the C# programming language. The pseudo-code for the rules in Table 1 are shown in Section Appendix A. To find the non-complaint parts of the building, a C# code was developed to use data from Revit model to first find all possible corners of apart-

ments in the model, and then check travel distances from all of these points towards exits to find the ones exceeding the requirements.

Step 4: Rule Reporting

This part is not a focus point in our research and article. In our case, we use a simple highlighting mechanism that shows in the model which elements are non-compliant. Reporting is also implemented in the form of an exported CSV file.

2.2. Preparation of Dataset for ML

Due to the absence of pre-existing datasets meeting the criteria required for the research objectives outlined, a dataset as introduced in [35], was used which comprises 2521 door samples collected from five high-rise residential building models. To safeguard against potential data leakage during model evaluation, a delineation was established: three of these buildings-comprising 27, 11, and 11 floors-were designated exclusively for training. Notably, while high-rise designs typically involve repetitive floor layouts, meticulous alterations were integrated into these configurations. Modifications included the strategic repositioning of doors, stairs, certain walls, etc., which maintained the architectural integrity of the original designs, but also ensured a dataset devoid of redundancy. For the purpose of model validation, a separate subset of 134 samples was aggregated from two individual floors across the remaining two buildings.

In high-rise buildings, exits are often situated near each other and close to the emergency staircase. In addition, these exit doors tend to be wider, not always, than other doors. Based on these observations, an initiative was undertaken to define new features from the least data available in BIM models (i.e., geometrical data from doors and stairs) for the development of a robust dataset for machine learning. These features are: 'Euclidean distance to the nearest door' (Feature 1), 'Euclidean distance to the nearest staircase' (Feature 2), 'travel distance to the nearest door' (Feature 3), 'travel distance to the nearest staircase' (Feature 4), and the door's 'Width' (Feature 5). The following describes how these features are extracted for each door using the Revit API. Although the details of the extraction are specific to Revit, it is important to note that the core methodology and features are not specific to any particular software. Users need to customise the data extraction process to the entities and data structures relevant to IFC files or other BIM tools they use.

To find these features, doors and stairs must be found on each floor of the building using the Revit API. The first step is to filter the levels of the building and then find the doors and stairs on each level. The building levels are first found by filtering the ViewPlans by FloorPlan view type and then reading the built-in 'level' parameter for each FloorPlan view. Doors and stairs are filtered based on the associated built-in category and the ElementLevelFilter condition. For the distance features, it was essential to ascertain the precise locations of both doors and staircases within the model. It should be noted that when working with architectural models, where architects apply their own style to the design of the building model, one of the biggest challenges can be retrieving data from the Revit model, i.e., it is possible that a door family has been designed so that the

'location point' is in a different place than the door itself. In the case of the door families in the current study models, the 'location points' were located in the middle of the door. For in-place families that do not have a single insertion point, the 'Transform' class was called from the 'Geometry' namespace and the 'Origin' property of the transform was used as the door point. Stairs are sometimes imported as a lumped family. As a result, they are not categorized as a staircase object to filter with the Revit API. Many other modeling practices can lead to poor data, such as importing geometry as plain geometric 'generic models'. This is an important practical issue that needs to be considered at the source, as any data export retains poor data modeling quality. In our study, all stairs were imported or created in such a way that they could be filtered with the built-in category 'OST-Stairs'. To achieve a relative consistency across staircases from different models with different shapes and orientations, the location of each staircase is pinpointed at the midpoint between its 'Bounding Box' minimum and maximum extents.

To calculate the Euclidean distances, direct horizontal distances between elements were calculated. On the other hand, the travel distances were determined using the 'PathOfTravel' feature of the Revit API. This feature provides the capability to designate which elements on the Path Of Travel are treated as obstacles. In this study, aside from doors, all other categories such as furniture, ducts, walls, and the like were considered. Design considerations often determine how a door's width is parameterized in architectural modeling, either as an instance or as a type. With this in mind, our research explored both configurations. For the purposes of this research, both parameters were investigated. The instance parameter was extracted directly from the door element using 'BuiltInParameter. DOOR_WIDTH', while the type parameter was sourced using BuiltInParameter. FAMILY_WIDTH_PARAM via the door's 'ElementType'.

To build the labeled dataset, a project parameter named "Door Type" which is a 'YesNo' parameter was defined for each door element. The parameter was then manually set for Exit doors within the Revit model. This information, along with other features, was extracted using a plug-in developed in the C# language in the . Net Framework (version 4.7.2).

Given that the number of 'Exits' in the building models is limited compared to other door types, we are presented with an imbalanced dataset.

Before applying the ML algorithm, the training and test samples must be preprocessed. Feature scaling is an important step in this process. Different features with different ranges can cause large-scale features to dominate small ones, leading to results that are biased toward the features with the higher magnitude. When data are scaled to an approximately equal range, the computational cost is reduced due to the fast convergence of gradient descent [39, 40]. A preliminary evaluation showed that standardization scaling, performed on the data of each building individually, results in better performance compared to normalization scaling.

2.3. Choice of ML Techniques

The current study demonstrates to what extent the use of some of the most well-known and widely used supervised learning models, namely Bagged *K*-Nearest Neighbors (*K*NN), Support Vector Machine (SVM) and eXtreme Gradient Boosting (XGBoost) can automatically locate exits in high-rise building models, by classifying building doors as exits or non-exits. The *K*NN technique was chosen for its simplicity and effectiveness in classification problems. Moreover, it seems to favor our unbalanced data because it does not remove low-frequency events to simplify the model [41] and it was improved by the bagging technique. Bagging as an ensemble learning method is used to improve the stability and accuracy of ML techniques. It generates many datasets from the original training dataset by bootstrap sampling [42], where a base learner, i.e. *K*NN in our study, is developed in parallel on each of these samples. If the results of the classifier are labels, as in our case, plurality voting is performed at the end to predict a class. If the trained models differ, then the variance[1] of the estimate improves by a factor of the number of models generated, otherwise it behaves like the base learner. Thus, it has no negative effect, i.e., it either improves the model or does not differ from using the base model [43].

SVM was chosen for its flexibility, robustness, strong generalization capabilities, and ability to consistently yield globally optimal solutions while preventing overfitting [44]. Especially considering the non-linear nature of our dataset, SVM is particularly well suited, as it implicitly maps the input data into a higher-dimensional feature space, offering the potential for enhanced predictive accuracy [45].

In handling our tabular dataset for binary classification, we chose XGBoost. Known for its adaptability and proven performance, XGBoost leverages the power of decision trees via gradient boosting [46]. Although neural networks are typically dominant in processing unstructured data such as text or images, for structured or tabular data sets-especially on a small to medium scale-algorithms such as XGBoost stand out as the preferred choice [47].

2.4. Hyper-Parameter Tuning

In ML, the development of a model includes more than just the selection of an appropriate algorithm. The true potential of an algorithm is unveiled through a meticulous process known as hyperparameter tuning. Hyperparameters, distinct from the model parameters learned during training, are predefined settings that govern the training process itself. They dictate crucial aspects such as the learning rate, the depth of trees in an ensemble, or the number of hidden layers in a neural network. Determining the optimal hyperparameters is paramount, as their values directly influence the model's performance on unseen data.

Traditional methods of manually selecting hyperparameters often rely on intuition or domain knowledge, but such approaches are time-consuming and may not guarantee the best performance. Automated techniques, such as grid search and

[1] *The variance is an error from sensitivity to small fluctuations in the training set. High variance may result from an algorithm modeling the random noise in the training data (overfitting).*

random search, offer systematic ways to explore hyperparameter space. When the data samples are divided into k folds (groups) of approximately the same size, it is called k-fold CV. Each fold is treated as a validation dataset and the remaining data are used as the training dataset. The model is fit to the training dataset each time and evaluated against the validation samples. The choice of k is important to ensure that both the training and validation samples are large enough to be statistically representative of the broader dataset. A special version of k-fold is stratified k-fold, where each fold contains the same proportion of each target class as the main dataset. Therefore, the individual folds are well representative of the main dataset. In our study, we use 10 folds, because this value has been shown to give the best results for different techniques [48–50]. Due to the stochastic nature of the ML technique, the repeated stratified k-fold CV was used. It performs the CV multiple times (in our case 10 times) over all folds, as shown in Fig. 4, and reports the average performance over all folds from all runs. The dataset used for CV must be different from that used for model validation, otherwise there will be data leakage where the model gets the information it is supposed to predict. This will lead to overoptimistic, incorrect, and unreliable performance.

In the current study, we trained on three different architectural models using a dataset of 2383 samples. This dataset included 190 'Exit' doors and the rest were labeled as 'Normal Door', indicating standard doors. For validation, we selected two additional high-rise buildings and tested the model on one floor of each. The first building had 94 doors with 4 'Exit'. The second building contained 44 doors with 4 'Exit'. As the model trained at this stage learned from a smaller dataset used in the CV, it must be re-trained with the entire training set before the validation test. Then the validation set is used as unseen data to check how the model performs when deployed.

Table 2 presents the range of hyperparameters for the three machine learning models used, along with the best estimators (optimal hyperparameters) identified through repeated stratified k-fold CV. The models discussed throughout the rest of the paper are constructed using these optimal hyperparameters.

2.5. Oversampling Technique

Given that our dataset was imbalanced, a condition that could lead to biased, poor generalization or incorrect results, wide range of oversampling techniques were investigated. The "smote-variants" Python library package, developed by [51], which provides the Python implementations of the 85 improvements made to SMOTE, is used to examine some of the methods proposed in the literature to tackle imbalanced and oversampling issues in our dataset. Over 30 different oversampling techniques were explored. Many of these techniques, as presented in Table 3, are modifications of SMOTE [52], which is among the most widely applied methods to address class imbalance in datasets. However, for clarity and brevity, only "Supervised SMOTE" is presented, as the results of alternative techniques did not demonstrate significant distinctions or contribute substantially to the overall findings.

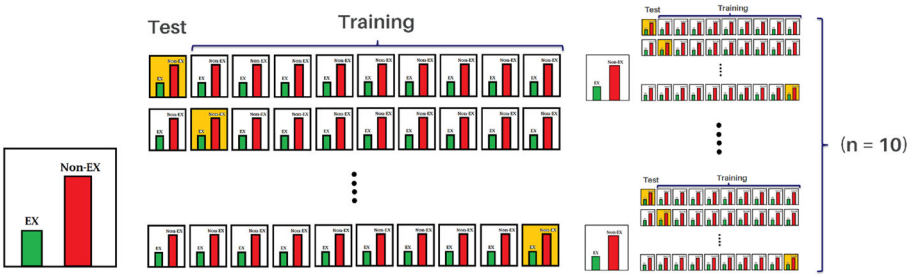

Figure 4. Illustration of repeated(10-times) k-fold cross validation.

Table 2
Hyperparameter Ranges and Best Parameters

ML Model	Hyperparameter	Range	Best estimator
Bagged *K*NN	#n_neighbors	3 to 19 (step 2)	5
	p	[1, 2]	1
	weight	['uniform', 'distant']	'distant'
	max_samples	[0.5, 1.0]	0.5
	max_features	[0.5, 1.0]	1.0
	n_estimators	[50, 100]	50
XGBoost	learning_rate	[0.01, 0.05, 0.1]	0.1
	n_estimators	[50, 100, 200]	100
	max_depth	[3, 4, 5]	5
	subsample	[0.8, 0.9, 1]	0.9
	colsample_bytree	[0.8, 0.9, 1]	0.8
	gamma	[0, 0.1, 0.2]	0
	min_child_weight	[1, 2, 3]	2
SVM	C	[0.1, 1, 10]	10
	kernel	['linear', 'rbf', 'poly', 'sigmoid']	'rbf'
	degree	[2, 3, 4]	2
	gamma	['scale', 'auto']	'scale'

Numerous studies have been conducted employing oversampling techniques before partitioning data into two separate and mutually exclusive sets dedicated to training and validation purposes [82–87]. A critical consideration when using oversampling techniques is to apply them accurately and exclusively to the training set, as failure to do so may lead to inflated, biased, and overoptimistic results [88, 89]. Even though some of the available Python packages, such as Scikit learn, allow easy implementation of CV, they do not allow implementation of oversampling during CV and only on the training set. Therefore, a custom Python code, developed by [35], was used to apply oversampling only on the training set in each round of CV. The importance of the above approach is illustrated by contrasting it with the prevailing practice of applying oversampling to the entire dataset,which

Table 3
Applied Oversampling Techniques in this Study

Technique	Reference	Technique	Reference
SMOTE	[53]	Borderline_SMOTE2	[54]
ADASYN	[55]	ProWSyn	[56]
BorderLine_SMOTE1	[54]	RandomOverSampler	[57]
NRSBoundary_SMOTE	[58]	Safe_Level_SMOTE	[59]
Polynom_fit_SMOTE	[60]	SMOTE_ENN	[61]
OUPS	[62]	SMOTE_RSB	[63]
SPY	[64]	SMOTE_TomekLinks	[65]
SSO	[66]	SN_SMOTE	[67]
SUNDO	[68]	SMOTE_IPF	[69]
Supervised_SMOTE	[70]	LN_SMOTE	[71]
CCR	[72]	SMOTE_PSOBAT	[73]
Cluster_SMOTE	[74]	SYMPROD	[75]
Distance_SMOTE	[76]	TRIM_SMOTE	[77]
Edge_Det_SMOTE	[78]	VIS_RST	[79]
MDO	[80]	MWMOTE	[81]

is often adopted without the necessary depth of knowledge or understanding of Python library packages.

2.6. Evaluation Metrics

When evaluating ML models, several metrics can be used to assess their performance. While accuracy, defined as the ratio of correct predictions to all predictions, is commonly used in balanced datasets, it may not be appropriate for unbalanced problems like ours, where one class significantly outweighs the other. In such cases, alternative measures such as the confusion matrix, precision, recall, F1 score and cross-entropy loss (known as log loss) become more meaningful and informative for evaluation purposes [90].

To better show the performance of our predictive model and its generalization, and to prevent overfitting [91], as in the hyperparameter tuning section 2, the repeated stratified k-fold CV was applied.

3. Results

In this section, we thoroughly assess the performance of our methodology. This encompasses practical applications, the consequences of improper integration of CV and oversampling, the efficacy of oversampling techniques, and a direct comparison between manual and automatic annotation of BIM models.

3.1. Illustration of Correct and False Identification of Exits

Fig. 5 shows a correctly predicted exit on one of the floors, using the methodology proposed in Section 2 with the Bagged *K*NN model applied, and illustrates

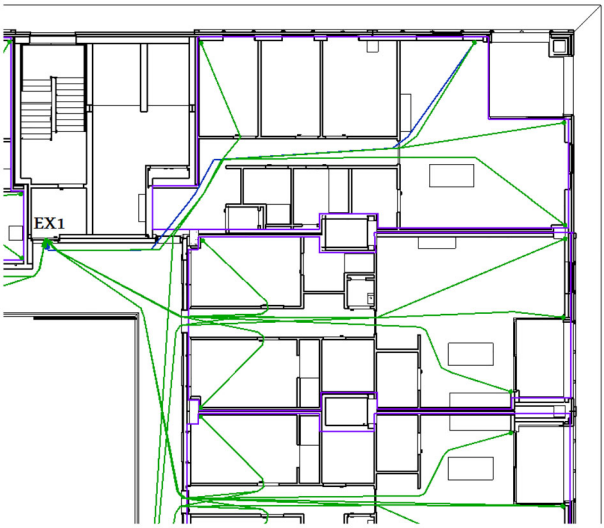

Figure 5. Representation of the travel distances between apartment corners and exits.

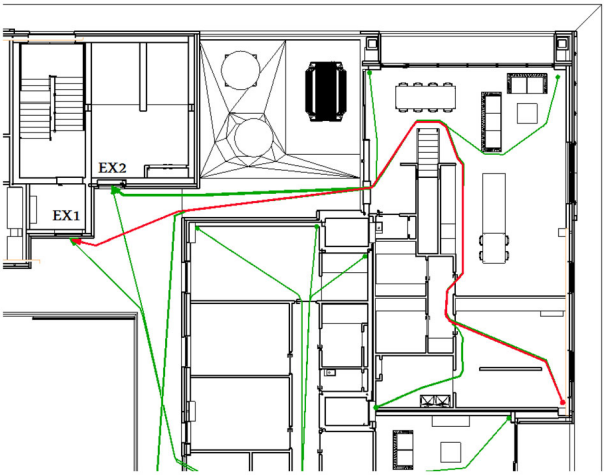

Figure 6. Illustration of highlighted, non-complaint distances on plans; example of a misidentified exit.

how the distances from the corner points of the apartments to the exits are measured to check the rules. Green lines indicate that the distances are within allowable limits, while red lines, as in Fig. 6, mark distances that exceed the limits. Fig. 6 also shows an incorrectly predicted exit (EX2). This could be avoided to some extent by adding additional constrains such as the presence of two exits per floor. However, this is not sufficient and user intervention is still required. The

final user check remains necessary to ensure accurate labeling of exits. This process is expedited, as the tool conveniently highlights identified exits in green, allowing users to quickly identify them when reviewing building plans. In the event that doors are mislabeled, users can adjust the 'Door Type' parameter parameter to rectify the labeling.

The following sections explain how the ML can be effectively applied in our case to automatically identify the exits on building models.

3.2. Emphasizing the Proper Fusion of Cross Validation and Oversampling

When using CV, it is important to correctly apply oversampling to the dataset. As described in [89], some researchers have incorrectly implemented the joint use of CV and oversampling: by applying oversampling to the entire dataset and then performing CV. This flawed approach introduces the risk that the model learns patterns from the test set due to the presence of oversampled data distributed across both training and test sets. Consequently, the models exhibit an exaggerated level of optimism. In contrast, the recommended approach involves oversampling within each CV iteration.

In our investigation, we conducted two tests to assess the impact: in the first test (Method 1), we oversampled the entire dataset using the SMOTE algorithm and performed CV, while in the second test (Method 2), we oversampled only the training dataset within each CV round and evaluated the model on unseen data. As pointed out by [89], our findings support the presence of overoptimism in Method 1. This observation is substantiated by Table 4, which demonstrates a significantly higher model performance for Method 1 compared to the validation results. Although Method 1 achieved a perfect prediction of exits, its validation performance was comparable to the lower-performing Method 2. This discrepancy can be attributed to the fact that Method 1 was exposed to portions of the data on which it was subsequently tested, resulting in inflated performance metrics. The purpose of this particular section is not to show the improvement in classification performance, but to shed light on a common pitfall in implementing the joint use of CV and oversampling; accordingly, we've chosen to use only the basic SMOTE algorithm in conjunction with the Bagged *K*NN model.

It is crucial to clarify the concept of overoptimism in our study, as it differs from the definition presented in [89]. The aforementioned study characterizes overoptimism when the classification performance of the training and validation sets show similarity. This suggests that oversampling is applied to the entire dataset before it is divided into training and validation sets. However, in our study, overoptimism is defined as a scenario where the trained model demonstrates high performance while the validation test yields poor results. This occurs when the validation set is correctly separated but the oversampling is applied before CV on the whole training set. We argue that the most appropriate implementation of oversampling is not just before the separation of the training and validation sets, but also during each iteration of the k-fold CV. Therefore, a portion of the data should first be set aside for validation. Subsequently, within each iteration of k-fold CV, oversampling should be exclusively applied to the training set. This

Table 4
Comparative Analysis of Trained Model Performance for Method 1 (Oversampling Before Applying CV), and Method 2 (Oversampling During CV) and Validation Results in Confusion Matrix for Both Methods

Method 1	Precision	Recall	F1 score	Log loss
Model Performance	0.97	1	0.98	0.15
Validation Performance	0.72	1	0.84	0.27
Method 2	Precision	Recall	F1 score	Log Loss
Model Performance	0.75	0.98	0.85	0.25
Validation Performance	0.71	1	0.83	0.41

methodology ensures the prevention of data leakage and mitigates the potential for overoptimistic results. An online repository provides access to a Python-based code designed to apply oversampling exclusively to training datasets within cross-validation processes [35].

3.3. Application of ML Techniques to Annotate 'Exits'

Three distinct ML models were constructed for this study: Bagged *K*NN, SVM, and XGBoost. The hyperparameters previously described were employed to ensure optimal configuration of the models. The training data comprised observations from three separate high-rise buildings, providing a diverse and robust dataset for the learning process. Post-construction, the models were subsequently validated for their performance capabilities on two independent floors taken from two distinct buildings. This validation step allowed an assessment of the models' generalization capabilities and efficacy in predicting real-world scenarios. The ensuing results offer a comprehensive view of each model's performance metrics and their applicability in building annotations.

Table 5 shows that the precision, recall, and F1 scores of the Bagged *K*NN, SVM, and XGBoost models are notably high, predominantly ranging between 0.87-0.89, 0.88-0.90, and 0.88-0.89, respectively. The relatively high and consistent performance of the Bagged *K*NN, SVM, and XGBoost models can be attributed to several factors. First, the systematic hyperparameter optimization performed for each model, ensured that they operated under optimal configurations tailored to the dataset. Second, the selected feature set appears to effectively capture the subtleties of the classification task, allowing different models to discern similar performance.

The confusion matrices in Table 5 further emphasize this proficiency. Of the 477 doors in the test case, only 9 were misclassified by all models. In the first vali-

Table 5
Evaluation of ML Models: Bagged *K*NN, SVM, and XGBoost for 'Exits' Annotation

		Metrics				Confusion matrix		
		Precision	Recall	F1	Log loss		ND	EX
Bagged *K*NN	Model	0.87	0.9	0.89	0.058	ND	432	6
		±0.02	±0.03	±0.02	±0.002	EX	3	36
	Validation 1	0.44	1	0.61	0.149	ND	85	5
		–	–	–	–	EX	0	4
	Validation 2	1	1	1	0.029	ND	40	0
		–	–	–	–	EX	0	4
SVM	Model	0.87	0.89	0.88	0.056	ND	432	5
		±0.03	±0.03	±0.02	±0.005	EX	4	35
	Validation 1	0.44	1	0.61	0.080	ND	85	5
		–	–	–	_	EX	0	4
	Validation 2	1	0.75	0.86	0.165	ND	40	0
		–	–	–	–	EX	1	3
XGBoost	Model	0.89	0.88	0.89	0.046	ND	433	4
		±0.01	±0.03	±0.02	±0.002	EX	4	35
	Validation 1	0.40	1	0.57	0.203	ND	84	6
		–	–	–	_	EX	0	4
	Validation 2	1	1	1	0.021	ND	40	0
		–	–	–	–	EX	0	4

dation set, consisting of 94 samples, every 'Exit' was accurately identified, but 5 or 6 other samples were misclassified as 'Exits'. For the second validation set, all models except for the SVM, correctly identified all doors. The SVM model made one misclassification. Most of the log loss values are below 0.1, indicating the confidence of the models in their predictions, with occasional values slightly exceeding this threshold.

3.4. Exploring Oversampling Techniques for Data Imbalance

The problem of imbalanced data that exists in many real-world ML problems [92] has attracted great research interest and led to remarkable advances in the field [51]. In our study, we focused on data-level techniques and specifically used oversampling. We found that undersampling led to worse results, possibly due to the loss of important data instances. Therefore, we opted for oversampling to augment the minority class and achieve a more balanced representation.

Upon examining the results of the ML models with and without oversampling, as presented in Table 6, a few key differences emerge. For the Bagged *K*NN, the application of oversampling reflects a marginally diminished precision, recall, and F1 scores for the model, with a slight increase in Log Loss. The SVM model shows comparable performance between the two conditions, although the oversampled data shows slightly lower values in precision and F1 score, and a slightly higher Log Loss. For the XGBoost model, the oversampled data display a minute

Table 6
Evaluation of ML Models with Oversampled Data: Bagged *K*NN, SVM, and XGBoost for 'Exits' Annotation

		Metrics				Confusion matrix	
		Precision	Recall	F1	Log Loss	ND	EX
Bagged *K*NN	Model	0.78 ±0.06	0.98 ±0.02	0.86 ±0.03	0.19 ±0.11	427 1	11 38
	Validation 1	0.44 –	1 –	0.62 –	0.86 –	85 0	5 4
	Validation 2	0.62 –	1 –	0.8 –	0.073 –	38 0	2 4
SVM	Model	0.76 ±0.06	0.97 ±0.03	0.84 ±0.04	0.071 ±0.022	426 1	12 38
	Validation 1	0.44 –	1 –	0.61 –	0.18 –	85 0	5 4
	Validation 2	0.75 –	0.75 –	0.75 –	0.409 –	39 1	1 3
XGBoost	Model	0.87 ±0.04	0.92 ±0.03	0.89 ±0.05	0.056 ±0.07	432 3	6 36
	Validation 1	0.44 –	1 –	0.62 –	0.27 –	85 0	5 4
	Validation 2	1 –	1 –	1 –	0.0006 –	40 0	0 4

decrease in recall, but other metrics remain largely consistent. It is imperative to note that these results are among the most promising outcomes across the diverse range of oversampling techniques that we employed in our study. In fact, most of the oversampling methods we experimented with did not provide significant improvement. Assessing the complexity measures in these models is instrumental, as it provides a nuanced understanding of the underlying changes introduced by the oversampling technique, thus offering a more comprehensive and in-depth view of the effects of data manipulation.

3.5. Automated vs. Manual Annotations: Comparison

To evaluate the practicality and efficiency of the proposed method, a comparative analysis was performed between the manual inspection of evacuation routes and the automated method introduced in this study. Manual evaluation involved two primary tasks: annotating the BIM model and verifying the compliance of the building with the evacuation route requirements specified in Table 1. The test was carried out on two floors of a simple high-rise building design, featuring eight exit doors, 132 normal doors, and 19 apartment units. A professional with Revit experience, familiar with all necessary steps, performed the task to ensure that no time was wasted learning or troubleshooting the process.

As a necessary initial step for the manual check and because neither the current nor previous versions of Revit provide a parameter to label doors as 'Exit' or 'Normal Door', a new project parameter called 'Door Type' has to be added to the BIM model. The annotation process involved sequentially opening each floor plan view, selecting only the doors identified as exits, and filling the checkbox next to the Door Type parameter. Once all exit doors were annotated, this phase was complete.

To ensure compliance with evacuation route regulations, the Path of Travel tool in Revit was employed. Available since Revit 2020, in the 'Analysis' tab under the 'Route Analysis' section, it enables users to measure travel distances between points while avoiding specified obstacles. To check the rules listed in Table 1, the travel distances from the apartment corners to the exits were assessed. The person minimized the number of corners by visually identifying those that appeared to be the most distant and likely to exceed the requirements. Each identified corner was then used as a starting point, and a Path of Travel line was drawn towards the exits. The default visual graphics in Revit were overridden by applying a filter rule for the length of the travel lines, facilitating the visual identification of corners that exceeded the requirements. For the selected building, this manual process took approximately 31 minutes to complete for two floors. In more complex building designs or with less experienced designers, additional corners would need to be selected and more path-of-travel lines would need to be drawn, increasing the time required for each step.

The same person then performed the automated design check using the tool developed in this study. This process also consisted of two steps: automatic labeling of exits and automatic rule checking. The developed plugin adds a new ribbon, named 'EvacRouteChecker', as shown in Fig. 7, which contains two buttons. The 'Annotate Doors' button automatically extracts data for the machine learning model, applies the model to classify doors as either 'Exit' or 'Normal Door' and highlights identified exits in green. User can immediately see if any doors were labeled incorrectly and manually correct them if necessary. In this test case, the predictions were correct, which did not require user intervention, resulting in a labeling process that was 90% faster.

Given the high accuracy of the machine learning model, the probability of needing manual intervention is minimal. In the event that manual correction is necessary, it would be limited in scope and would not be applicable to every door. Following the annotation of the doors, the 'Check Evac Rules' button was utilized to verify compliance with the evacuation rules. The entire process, which included both annotation and verification, was completed in 10 minutes with minimal user intervention, required only post-annotation. This stands in stark contrast to the manual process, which required constant user participation, thereby underscoring the significant value of the tool, particularly for iterative design processes and for large and complex buildings.

Based on this test case, the tool facilitated the evacuation route rule-check process, making it more than three times faster. Manual control was required for less than one minute, as accurate exit predictions required only visual inspection. Con-

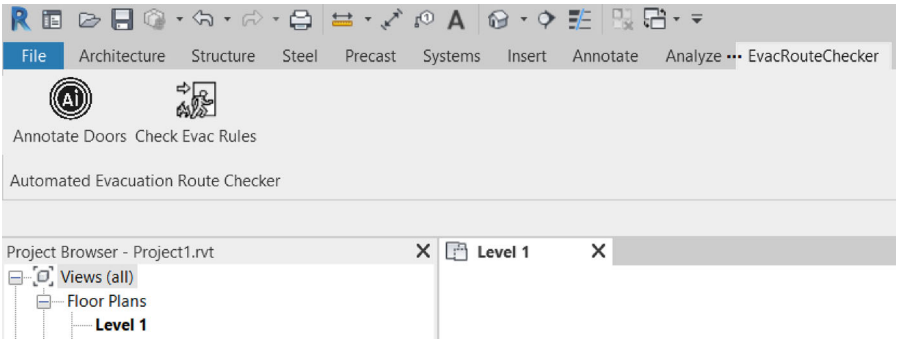

Figure 7. Screenshot of the 'EvacRouteChecker' plugin interface in Revit. (Note: The 'EvacRouteChecker' ribbon in Revit appears after the last default Revit ribbon. However, due to space limitations, the figure has been manually cropped. The dots indicate the existence of additional ribbons between the 'Analyse' and 'EvacRouteChecker' ribbons in Revit).

sequently, the tool reduced manual work from 31 minutes to approximately one to two minutes.

4. Discussion

This section explores aspects that were not directly addressed in our study in order to provide additional insights into the implications of our findings. In addition, the complexity measures associated with oversampling techniques are explored in depth and discussed to identify the reasons for the unexpected inefficiency of oversampling in our study.

4.1. Analyzing Dataset Complexity

In oversampling assessment, data complexity measures are used to evaluate the difficulty of a classification task based on the characteristics of the dataset. Table 7 presents a comparison of complexity measures computed for the original dataset and the oversampled dataset generated through the application of Supervised-SMOTE, as implemented in the problexity Python library [93]. Interestingly, the provided table clarifies the absence of overlap, as indicated by a volume of overlapping region (F2) [94] equal to zero in both the original and oversampled datasets. Although initially suggested successful class imbalance mitigation through oversampling, evident in the entropy of class proportions (C1) [94] and imbalance ratio (C2) [94] reducing to zero, a closer examination unveils nuanced intricacies. The lower Maximum Fisher's discriminant ratio (F1), [94], for the oversampled dataset suggests a reduced capability for separating class features. Furthermore, inefficiency in individual feature separation (F3) [94] is more pronounced in the data set oversampled. Regarding neighborhood characteristics, the oversampled dataset demonstrates higher values in measures such as the fraction

Table 7
Comparison of Complexity Measures Between the Oversampled Dataset (using Supervised-SMOTE) and the Original Dataset

Complexity Measure	c1	c2	f1	f2	f3	n1	n2
Oversampled Dataset	0	0	0.251	0	0.594	0.008	0.355
Original Dataset	0.566	0.806	0.650	0	0.265	0.013	0.819

of borderline points (N1) [94] and the ratio of intra-class to extra-class nearest neighbor distance (N2) [94], indicative of a more intricate neighborhood structure.

In summary, the results suggest that the oversampled dataset using Supervised SMOTE has certain characteristics that make it more complex in terms of class separation, neighborhood structure compared to the original dataset. The original dataset seems to have a simpler structure in terms of linearity, neighborhood, and class distribution.

4.2. Considerations Beyond Methodology Implementation

Revit was chosen primarily due to its widespread popularity and adoption in the AEC industries. This ensures that the end tool is immediately useful and accessible to a large group who are already familiar with Revit's environment. It is essential to understand while the general methodology of the project, including the machine learning model, its hyperparameters, and the proposed features, remain unchanged, to apply the methodology using IFC files or other BIM tools customization is required to the data extraction codes to align with the data structure and formats of the chosen BIM tool.

With our limited features and data set, the result is very promising. We believe that our methodology could be employed for fast annotation and would be well suited for use in automated code compliance checks by retrieving data from BIM models without requiring additional user input. Despite achieving success with five key features, the study recognizes the need for additional discriminative features. Future research will focus on advanced feature engineering to improve model performance and address the complexity of real-world buildings more comprehensively. The dataset, derived from five high-rise residential buildings, raises questions about the generalization of the methodology. While the study acknowledges potential limitations, future research endeavors to diversify datasets, encompassing more building types and construction standards to enhance generalization. Acknowledging the efficacy of simpler models (Bagged *K*NN, SVM, XGBoost), our exploration may extend to more complex models, such as deep learning architectures. This consideration aims to strike a balance between capturing nuanced patterns and managing computational efficiency, offering potential avenues for improved model performance.

While the comparison between automated and manual annotations was based on time efficiency, future evaluations will incorporate a qualitative analysis. This

involves engaging real users in the daily application of the tool, allowing for a hands-on examination of its performance. In addition, surveys will be conducted meticulously to collect insights into user experiences, ensuring a comprehensive understanding of how well automated annotations align with industry standards.

All issues relating to non-compliance with fire safety regulations should ideally be resolved before the construction phase begins. Otherwise, any major change to the design model would lead to changes to other models or calculations, resulting in a significant loss of time and increased costs. This means that architects are important users at an early stage of the process. As a plugin for Revit, the tool enables iterative design review for architects at earlier stages, where resolving conflicts and non-conformities is less costly and easier. Others can also benefit; for example, authorities can speed up the conformity assessment of buildings. The current study is a proof-of-concept that lacks detailed practical implementation and integration. Future work will explore strategies to seamlessly integrate the proposed solution into existing industry workflows and software systems, spanning the entire process from design to building approval, ensuring its practical applicability.

It should be noted that there may be slightly different results when the algorithm is used by other users. There are several reasons for possible minor discrepancies, namely the use of different training datasets, the stochastic nature of the applied ML technique (Bagged *K*NN and XGBoost), different platforms or development environments used to run the tool [95].

The task of automatically checking evacuation routes in BIM models was chosen because of its critical importance in ensuring safety and compliance with regulations that are fundamental to building safety. While it could be argued that the manual annotation of exit doors as part of this automation is not an overwhelmingly complex task, its significance is particularly evident in the iterative design process. Additionally, having this tool enables fire safety engineers, who are often not familiar with BIM and its tools, to directly engage with original BIM models. Typically, these engineers receive 2D plans which may not include the latest updates or crucial design details. Using BIM tools, fire safety engineers gain access to real-time project information, enabling seamless collaboration with stakeholders and ensuring accurate understanding of the design details crucial for safety compliance.

Moreover, it is important to recognize that this research serves as a proof of concept and that the methodology is general and adaptable. The developed technique has significant potential for application to other complex BIM annotation tasks, such as identifying compartment walls, mapping evacuation routes, or detecting building exteriors. These applications are just a few examples of the broad applicability and need for semantic enrichment to automate and streamline compliance checks, reducing manual effort for designers, and ensuring the accuracy in complex structures such as high-rise buildings.

Although this study focuses on the Belgian case, the general ideas of using machine learning and extracting data and features from the BIM model are flexible and adaptable. The machine learning models can be retrained with localized data to ensure compliance with specific national or regional regulations, accom-

modate different design standards and ensure broader applicability of the methodology.

5. Conclusion

The current study focused on automating the design check against the rules for evacuation routes. In order to apply the rules, the exits must be known. However, this information, which should be included in the building model, is usually missing in BIM models. This can be added to the model either by engineers or by prompting the user for input. However, in this paper we presented a new approach to automatically obtain this information automatically from other available data by using ML techniques, using Bagged *K*NN, SVM and XGBoost classifiers. The proof-of-concept was demonstrated by consistently obtaining precision, recall, and F1 scores ranging from 0.87 to 0.90 for all models.

By adding extra semantics to the doors, whether a door is an exit or not, and using the Path of Travel tool in Revit, verification of the given constraints on the paths was achieved. Future work will involve feature selection as an important component in building predictive models, and it is expected that better results will be obtained if more features are added. Future endeavors will also explore the extension of the application to annotate other elements within BIM models.

Additionally, the study emphasized the correct implementation of oversampling techniques when applying CV to measure the performance of the model. Inadequate implementation will result in data leakage and overly optimistic results.

Our study draws attention to the limitations of commonly employed oversampling techniques to address imbalanced classification problems. Our extensive evaluation of numerous oversampling techniques suggests that despite their widespread use and past success, these techniques may not always deliver the desired improvements in classifier performance. It is highlighted that class imbalance alone does not solely dictate classification performance. The inner structural intricacies of the dataset, including the presence of noisy or rare samples, borderline points, neighbourhood of instances, overlap of feature values, etc., can significantly influence the efficacy of oversampling techniques. These complexities emphasize the need for exploring alternative methods to address the challenges posed by intricate dataset characteristics in imbalanced scenarios.

Funding

This work was supported by the Flanders innovation & entrepreneurship (VLAIO) [grant number HBC.2019.2623]; and Jensen Hughes Company.

Appendix

Pseudo Codes for Evacuation Route Rules

Pseudo code for rules in Table 1: "No point of a compartment shall be at a distance greater than 30 m from access to the nearest stairs or exit and 60 m from access to the second staircase or the second exit."

```
# Run Machine Learning code to find Emergency Exits
KNNalgorithm.py

# Run Transformer code to label Exit doors in BIM model
Transformer.sln

# Filter Apartments in Revit tool
apartments = FilteredElementCollector(RevitDocument).Category(BuiltInCategory.
    OST_Areas)

# Filter All Doors in Revit tool
all_doors = FilteredElementCollector(RevitDocument).Category(BuiltInCategory.
    OST_Doors)
levels = apartments.Level;
exits = [];

# For each level in the building
for level in levels:
    # Filter Exits in Revit tool
    for door in all_doors:
        if door.LookupParameter(``Exit"):
            exits.add(door);

    # Create new view plan with floor plan type in Revit tool
    new_view_plan = ViewPlan.Create(RevitDocument, ViewFamilyType(Floor Plan).Id,
        level.Id)
    for apartment in apartments:
        # Find apartment corner points
        start_points = AppendApartmentNearCornerPoints(apartment);
        # Generate PathOfTravel between all Emergency Exits and all apartment
            corner points
        path_of_travels =
            CreatePathsOfTravelInAllRoomsAllDoorsMultiplePointsManyToMany(
            UIDocument, NewViewPlan, exits, start_points)
        path_length = path_of_travels.get_Parameter(BuiltInParameter.
            CURVE_ELEM_LENGTH)
        # Check the legislation requirement
        if path_length.Min() > 30 or path_length.Max() > 60:
            path_id = path_length.Id
            Highlight-NonCompliances.OverrideElements(RevitDocument, path_id)
```

Pseudo Code for Transformer Module

```
# Create and open definition file
CreateExternalSharedParamFile(‘file address’)
SetAndOpenExternalSharedParamFile(Revit Application, ‘file address’)

# Create an instance definition in definition group
My-DefinitionGroup = myDefinitionfile.Groups;

# Create an instance definition in new created definition group
myOption = ExternalDefinitionOptions(‘‘Exit", ParameterType.YesNo);

# Create a category set and insert category of door to it
myinstCategory = Category.GetCategory(Revit Document, BuiltInCategory.OST_Doors)

# Create an instance of Instance Binding
instanceBinding = UserInterfaceApplication.create.NewInstanceBinding(myinstCategory
    )

# Get the Binding Map of current open Revit Document
bindingMap = RevitDocument.ParameterBindings;

# Bind Definition to the document
bindingMap.Insert(instanceBinding, BuiltInParameterGroup.PG-Text)
```

References

1. Arthur EC, John RH, Pam P, Casey CG, Robert ES (2008) Codes and standards for the built environment. In: Arthur, E.C., Casey, C.G. (eds.) Fire protection handbook, vol. 1, 20th edn., pp. 1–51. National Fire Protection Association, Quincy, Mass
2. Eastman C, Lee J-m, Jeong Y-s, Lee J-k (2009) Automatic rulebased checking of building designs. Autom Constr 18:1011–1033. 10.1016/j.autcon.2009.07.002
3. Dimyadi J, Amor R (2013) Automated building code compliance checking - where is it at? 19th International CIB World Building Congress, https://doi.org/10.13140/2.1.4920.4161
4. Norbert W. Young Jr, Stephen A. Jones HMB (2007) Interoperability in the construction industry, smartmarket report. Technical report, McGraw-Hill Construction Research and Analytics. www.construction.com
5. Fenves JS (1966) Tabular decision logic for structural design. J Struct Div 92:473–490. 10.1061/JSDEAG.0001567
6. Khemlani L (2021) CORENET e-PlanCheck: Singapore’s Automated Code Checking system. Last Accessed 19, 02, 2021. http://www.aecbytes.com/feature/2005/CORENETePlanCheck.html
7. Lee Jin K (2011) Building environment rule and analysis (BERA) language and its application for evaluating building circulation and spatial program. Universiteit Antwerpen, Belgium, Phd
8. Martins EP, Monterio A (2013) A BIM based automated code-checking application for water distribution systems. Autom Construct 29:12–23. 10.1016/j.autcon.2012.08.008
9. Hjelseth E, Nisbet N (2010) Exploring semantic based model checking. In: Proceedings of the CIB W78 2010: 27th International Conference, pp. 341–351
10. Malsane S, Matthews J, Lockley S, Greenwood D (2015) Development of an object model for automated compliance checking. Autom Constr 11:51–58. 10.1016/j.autcon.2014.10.004

11. Solibri Model Checkeing(SMC). https://www.solibri.com/. Last Accessed: 30, 12, 2022 (2022)
12. SmartReview Automated Plan Review. https://smartreview.biz/apr_learn_more. Last Accessed: 30, 12, 2022 (2022)
13. Bloch T, Sacks R (2020) Clustering information types for semantic enrichment of building information models to support automated code compliance checking. J Comput Civil Eng 34(6):04020040. 10.1061/(ASCE)CP.1943-5487.0000922
14. Solihin W, Shaikh N, Rong X, Lam K (2004) Beyond interoperatibility of building model: a case for code compliance checking. In: BP-CAD Workshop, Carnegie Melon University, pp. 1, 13
15. Bloch T, Katz M, Yosef R, Sacks R (2019) Automated model checking for topologically complex code requirements – security room case study. 2019 European Conference on Computing in Construction, https://doi.org/10.35490/EC3.2019.157
16. Xiong X, Adan A, Akinci B, Huber D (2013) Automatic creation of semantically rich 3D building models from laser scanner data. Autom Construct 31:325–337. 10.1016/j.autcon.2012.10.006
17. Sacks R, Ma L, Yosef R, Borrmann A, Daum S, Kattel U (2017) Semantic enrichment for building information modeling: Procedure for compiling inference rules and operators for complex geometry. J Comput Civil Eng . 10.1061/(ASCE)CP.1943-5487.000070
18. Simeonea D, Cursia S, Acierno M (2019) BIM semantic-enrichment for built heritage representation. Autom Constr 97:122–137. 10.1016/j.autcon.2018.11.004
19. Hong T, Wang Z, Luo X, Zhang W (2020) State-of-the-art on research and applications of machine learning in the building life cycle. Energy Build 212:109831. 10.1016/j.enbuild.2020.109831
20. Bomba M (2020) Level Of Development (LOD) specification part I & commentary for building information models and data. BIM Forum, 15–19
21. Simeone D, Cursi S, Acierno M (2019) Bim semantic-enrichment for built heritage representation. Autom Constr 97:122–137. 10.1016/j.autcon.2018.11.004
22. Châteauvieux-Hellwig C, Abualdenien J, Borrmann A (2020) Towards semantic enrichment of early-design timber models for noise and vibration analysis. ECPPM 2020:1–7
23. Bloch T (2022) Connecting research on semantic enrichment of bim-review of approaches, methods and possible applications. J Inform Technol Constr 27:416–440
24. Bloch T, Sacks R (2018) Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models. Autom Constr 91:256–272. 10.1016/j.autcon.2018.03.018
25. Belsky M, Sacks R, Brilakis I (2016) Semantic enrichment for building information modeling. Comput Aided Civil Infra Eng 31:261–274. 10.1111/mice.12128
26. Flach PA, Kakas AC (2000) Abduction and induction: Essays on their relation and integration. In: Flach PA, Kakas AC (eds) Abductive and inductive reasoning: background and issues Springer, Dordrecht, pp 1–27
27. Strug B, Slusarczyk G (2023) Machine learning methods in bim-based applications - a review. Vietnam J Comput Sci . 10.1142/S2196888823300028
28. Jin C, Xu M, Lin L, Zhou X (2018) Exploring BIM Data by Graph-based Unsupervised Learning. In: Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods - ICPRAM, pp. 582–589. SciTePress. https://doi.org/10.5220/0006715305820589 . INSTICC
29. Núnez-Calzado PE, Alarcón-López IJ, Martínez-Gómez DC (2018) Machine learning in bim. In: EUBIM 2018: Proceedings of the International BIM Conference, pp. 99–109
30. Koo B, Jung R, Yu Y (2021) Automatic classification of wall and door bim element subtypes using 3d geometric deep neural networks. Adv Eng Inform 47:101200

31. Collins FC, Braun A, Ringsquandl M, Hall DM, Borrmann A (2021) Assessing ifc classes with means of geometric deep learning on different graph encodings. In: Proc. of the 2021 European Conference on Computing in Construction, pp. 332–341
32. Kim J, Song J, Lee J (2019) Recognizing and classifying unknown object in bim using 2d cnn. In: Lee, J.-H. (ed.) Computer-Aided Architectural Design. "Hello, Culture"-18th International Conference, CAAD Futures 2019, Selected Papers. Communications in Computer and Information Science, pp. 47–57. Springer, Germany. https://doi.org/10.1007/978-981-13-8410-3_4
33. Luo H, Gao G, Huang H, Ke Z, Peng C, Gu M (2023) Automatic classification of wall and door bim element subtypes using 3d geometric deep neural networks. In: Karlinsky L, Michaeli T, Nishino K (eds) Computer Vision - ECCV 2022 Workshops Springer, Cham, pp 349–365
34. Koo B, Jung R, Yu Y, Kim I (2021) A geometric deep learning approach for checking element-to-entity mappings in infrastructure building information models. J Comput Design Eng 8(1):239–250
35. Bigdeli S, Pauwels P, Verstockt S, Weghe N, Merci B (2023) ML-based Exit identification. CodeOcean. Accessed 02(08):2023
36. Zhang R, El-Gohary N (2020) A machine-learning approach for semantic matching of building codes and building information models (BIMs) for supporting automated code checking. In: Rodrigues H, Morcous G, Shehata M (eds) Recent Res Sustain Struct Springer, Cham, pp 64–73
37. Zhang J, El-Gohary NM (2016) Extending building information models semiautomatically using semantic natural language processing techniques. J Comput Civil Eng . 10.1061/(asce)cp.1943-5487.0000536
38. Solihin W, Eastman C (2015) Classification of rules for automated bim rule checking development. Autom Const 53:69–82. 10.1016/j.autcon.2015.03.003
39. Mohd Nawi N, Hussein A, Samsudin N, Hamid N, Mohd Yunus MA, Aziz MF (2017) The effect of pre processing techniques and optimal parameters selection on back propagation neural networks. Int J Adv Sci Eng Inform Technol 7:770
40. Ak D, Venugopalan SRD (2017) The effect of normalization on intrusion detection classifiers (naïve bayes and j48). Int J Future Revol Comput Sci Commun Eng 3:60–64
41. Hoste V (2005) Optimization issues in machine learning of coreference resolution. PhD thesis, Universiteit Antwerpen, Faculteit Letteren en Wijsbegeerte
42. Zoubir A, Iskander D (2007) Bootstrap methods and applications. Signal Process Mag IEEE 24:10–19. 10.1109/MSP.2007.4286560
43. Ghojogh B, Crowley M (2019) The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial. arXiv. https://doi.org/10.48550/ARXIV.1905.12787.https://arxiv.org/abs/1905.12787
44. Awad M, Khanna R (2015) Support vector machines for classification. Apress, Berkeley, CA, pp. 39–66
45. Fawzy H, Rady EHA, Abdel Fattah AM (2020) Comparison between support vector machines and k-nearest neighbor for time series forecasting. J Math Comput Sci 10 (6):2342–2359
46. Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. In: the 22nd ACM SIGKDD International Conference, pp. 785–794 . https://doi.org/10.1145/2939672.2939785
47. Subhi Malallah H, Bahjat Abdulrazzaq M (2023) Web-based agricultural management products for marketing system: Survey. Academic J Nawroz Univ 12(2):49–62

48. Witten IH, Frank E, Hall MA (2011) Data Mining: Practical Machine Learning Tools and Techniques. Third edition edn. The Morgan Kaufmann Series in Data Management Systems, Boston, pp. 587–605
49. Caon DRS, Amehraye A, Razik J, Chollet G, Andreão RV, Mokbel C (2010) Experiments on acoustic model supervised adaptation and evaluation by K-Fold cross validation technique. In: 2010 5th International Symposium On I/V Communications and Mobile Network, pp. 1–4. https://doi.org/10.1109/ISVC.2010.5656264
50. Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. Int Joint Conf Artif Intell Organ 14:1137–1143
51. Kovács G (2019) Smote-variants: a python implementation of 85 minority oversampling techniques. Neurocomputing 366:352–354. 10.1016/j.neucom.2019.06.100
52. Elreedy D, Atiya A (2019) A comprehensive analysis of Synthetic Minority Oversampling TEchnique (SMOTE) for handling class imbalance. Inform Sci . 10.1016/j.ins.2019.07.070
53. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16(1):321–357
54. Han H, Wang W-Y, Mao B-H (2005) Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang D-S, Zhang X-P, Huang G-B (eds) Adv Intell Comput Springer, Berlin, Heidelberg, pp 878–887
55. He H, Bai Y, Garcia EA, Li S (2008) ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN), pp. 1322–1328
56. Baru (2013) Prowsyn: Proximity weighted synthetic oversampling technique for imbalanced data set learning. In: Advances in Knowledge Discovery and Data Mining, pp. 317–328
57. Menardi G, Torelli N (2012) Training and assessing classification rules with unbalanced data. Data Mining Knowl Discov . 10.1007/s10618-012-0295-5
58. Feng H, Hang L (2013) A novel boundary oversampling algorithm based on neighborhood rough set model: NRSBoundary-SMOTE. Mathematical Problems in Engineering, 10
59. Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C (2009) Safe-Level-SMOTE: Safe-Level-Synthetic minority over-sampling technique for handling the class imbalanced problem. In: Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, pp. 475–482
60. Gazzah S, Amara NEB (2008) New oversampling approaches based on polynomial fitting for imbalanced data sets. In: The Eighth IAPR International Workshop on Document Analysis Systems, pp. 677–684
61. Batista G, Prati R, Monard M-C (2004) A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor 6:20–29. 10.1145/1007730.1007735
62. Rivera WA, Xanthopoulos P (2016) A priori synthetic over-sampling methods for increasing classification sensitivity in imbalanced data sets. Expert Sys Appl 66:124–135
63. Ramentol E, Caballero Y, Bello R, Herrera F (2012) SMOTE-RSB*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. Knowl Inform Sys 33(2):245–265. 10.1007/s10115-011-0465-6
64. Dang XT, Tran DH, Hirose O, Satou K (2015) SPY: A novel resampling method for improving classification performance in imbalanced data. In: 2015 Seventh International Conference on Knowledge and Systems Engineering (KSE), pp. 280–285

65. Batista GEAPA, Prati RC, Monard MC (2004) A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor Newslett 6:20–29
66. Rong T, Gong H, Ng W (2014) Stochastic sensitivity oversampling technique for imbalanced data. Commun Comput Inform Sci 481:161–171. 10.1007/978-3-662-45652-1_18
67. García V, Sánchez J, Martín Félez R, Mollineda R (2012) Surrounding neighborhood-based SMOTE for learning from imbalanced data sets. Progress Artif Intell 1:347–362. 10.1007/s13748-012-0027-5
68. Cateni S, Colla V, Vannucci M (2011) Novel resampling method for the classification of imbalanced datasets for industrial and other real-world problems. In: 2011 11th International Conference on Intelligent Systems Design and Applications, pp. 402–407
69. Sáez JA, Luengo J, Stefanowski J, Herrera F (2015) SMOTE-IPF: addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. Inform Sci 10:184–203
70. Hu J, He X, Yu D-J, Yang X-B, Yang J-Y, Shen H-B (2014) A new supervised over-sampling algorithm with application to protein-nucleotide binding residue prediction. PLos ONE 10:1–10
71. Maciejewski T, Stefanowski J (2011) Local neighbourhood extension of SMOTE for mining imbalanced data. In: 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 104–111
72. Koziarski M, Wozniak M (2017) Ccr: a combined cleaning and resampling algorithm for imbalanced data classification. Int J Appl Math Comput Sci 27:727–736
73. Li J, Fong S, Zhuang Y (2015) Optimizing SMOTE by metaheuristics with neural network and decision tree. In: 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI), pp. 26–32
74. Cieslak DA, Chawla NV, Striegel A (2006) Combating imbalance in network intrusion datasets. In: 2006 IEEE International Conference on Granular Computing, pp. 732–737
75. Kunakorntum I, Hinthong W, Phunchongharn P (2020) A synthetic minority based on probabilistic distribution (SyMProD) oversampling for imbalanced datasets. IEEE Access, 114692–114704
76. Calleja J, Fuentes O (2007) A distance-based over-sampling method for learning from imbalanced data sets. In: Proceedings of the Twentieth International Florida Artificial Intelligence, pp. 634–635
77. Puntumapon K, Waiyamai K (2012) A pruning-based approach for searching precise and generalized region for synthetic minority over-sampling. In: Tan P-N, Chawla S, Ho CK, Bailey J (eds) Adv Knowl Discov Data Min Springer, Berlin, Heidelberg, pp 371–382
78. Urban JL, Song J, Santamaria S, Fernandez-Pello C (2019) Ignition of a spot smolder in a moist fuel bed by a firebrand. Fire safety J 108:102833
79. Borowska K, Stepaniuk J (2016) Imbalanced data classification: a novel re-sampling approach combining versatile improved SMOTE and rough sets. In: Saeed K, Homenda W (eds) Computer Information Systems and Industrial Management Springer, Cham, pp 31–42
80. Abdi L, Hashemi S (2016) To combat multi-class imbalanced problems by means of over-sampling techniques. IEEE Transactions on Knowledge and Data Engineering, 238–251
81. Barua S, Islam MM, Yao X, Murase K (2014) Mwmote-majority weighted minority oversampling technique for imbalanced data set learning. IEEE Trans Knowl Data Eng 26:405–425

82. Fergus P, Cheung P, Hussain A, Al-Jumeily D, Dobbins C, Iram S (2013) Prediction of preterm deliveries from ehg signals using machine learning. PLOS ONE 8(10):1–16. 10.1371/journal.pone.0077154
83. Peng J, Hao D, Yang L, Du M, Song X, Jiang H, Zhang Y, Zheng D (2020) Evaluation of electrohysterogram measured from different gestational weeks for recognizing preterm delivery: a preliminary study using random forest. Biocybern Biomed Eng 40 (1):352–362
84. Ren P, Yao S, Li J, Valdes-Sosa PA, Kendrick KM (2015) Improved prediction of preterm delivery using empirical mode decomposition analysis of uterine electromyography signals. PLOS ONE 10(7):1–16. 10.1371/journal.pone.0132116
85. Naeem SM, Ali AF, Eldosoky MA (2013) Kl. comparison between using linear and non-linear features to classify uterine electromyography signals of term and preterm deliveries. In: 2013 30th National Radio Science Conference (NRSC), pp. 492–502. https://doi.org/10.1109/NRSC.2013.6587953
86. Idowu IO, Fergus P, Hussain A, Dobbins C, Khalaf M, Casana Eslava RV, Keight R (2015) Artificial intelligence for detecting preterm uterine activity in gynecology and obstetric care. In: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, pp. 215–220. https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.31
87. Hoseinzadeh S, Amirani MC (2018) Use of electro hysterogram (ehg) signal to diagnose preterm birth. In: Electrical Engineering (ICEE), Iranian Conference On, pp. 1477–1481. https://doi.org/10.1109/ICEE.2018.8472416
88. Vandewiele G, Dehaene I, Kovács G, Sterckx L, Janssens O, Ongenae F, De Backere F, De Turck F, Roelens K, Decruyenaere J, Van Hoecke S, Demeester T (2021) Overly optimistic prediction results on imbalanced data: a case study of flaws and benefits when applying over-sampling. Artif Intell Med 111:101987. 10.1016/j.artmed.2020.101987
89. Santos M, Soares J, Henriques Abreu P, Araujo H, Santos J (2018) Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches. IEEE Comput Intell Mag 13:59–76. 10.1109/MCI.2018.2866730
90. Thai-Nghe N, Gantner Z, Schmidt-Thieme L (2011) A new evaluation measure for learning from imbalanced data. In: The 2011 International Joint Conference on Neural Networks, pp. 537–542. https://doi.org/10.1109/IJCNN.2011.6033267
91. Berrar D (2019) Cross-validation. In: Ranganathan S, Gribskov M, Nakai K, Schönbach C (eds) Encyclopedia of Bioinformatics and Computational Biology Academic Press, Oxford, pp 542–545
92. Huang Z, Sang Y, Sun Y, Lv J (2022) A neural network learning algorithm for highly imbalanced data classification. Inform Sci 612:496–513. 10.1016/j.ins.2022.08.074
93. Komorniczak J, Ksieniewicz P (2023) problexity-an open-source python library for supervised learning problem complexity assessment. Neurocomputing 521:126–136
94. Komorniczak J, Ksieniewicz P (2023) problexity - an open-source Python library for supervised learning problem complexity assessment. Neurocomputing 521:126–136. 10.1016/j.neucom.2022.11.056
95. Brownlee J (2022) Why Do I Get Different Results Each Time in Machine Learning? https://machinelearningmastery.com/different-results-each-time-in-machine-learning/