Magazine of Civil Engineering

journal homepage: http://engstroy.spbstu.ru/

ISSN 2712-8172

Research article

UDC 69.05.04

DOI: 10.34910/MCE.126.2

Machine learning model for the BIM classification in IFC format

M.V. Petrochenko , ***P.N. Nedviga***, ***A.A. Kukina*** , ***K.I. Strelets***, ***V.V. Sherstyuk*** ✉

Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russian Fedration

✉ *sherstyuk2.vv@yandex.ru*

Keywords: classification in construction, artificial intelligence, classifier, classification model, machine learning, neural networks, BIM technologies, BIM, IFC, Revit, civil engineering, CAD

Abstract. In the rapid development of information technology in the field of Building Information Modeling (BIM) there is a growing need for efficient classification of construction information. One of the key steps to move towards digital construction involves creating reliable systems for classifying BIM elements, providing the foundation for various use cases, from facilitating model navigation to obtaining practical outcomes such as cost estimates and materials quantities. However, the BIM classification process in practice is labor-intensive and time-consuming and leads to an increase in the cost. This study explores the application of an innovative method, based on artificial intelligence algorithms. This method automates the assignment of codes to information model components. The research investigates classification systems, machine learning models and selects the most accurate one for the classification task. It is based on metrics such as accuracy and F1-score in order to achieve an optimal balance between the efficiency and accuracy according to predefined parameters. The article presents software for automatic prediction and assignment of codes in accordance with the selected classifier, developed on selected algorithms.

Citation: Petrochenko, M.V., Nedviga, P.N., Kukina, A.A., Strelets, K.I., Sherstyuk, V.V. Machine learning model for the BIM classification in IFC format. Magazine of Civil Engineering. 2024. Article no. 12602. DOI: 10.34910/MCE.126.2

1. *Introduction*

In modern times, BIM (Building Information Modeling) technologies are widely used in various fields, revolutionizing the traditional processes associated with the design, construction and operation of buildings. A BIM model contains information about geometry, materials, resource consumption, work schedules, costs, and other parameters. This digital model is developed throughout the entire project lifecycle, from the conception to the operation and maintenance.

One of the key aspects that makes BIM highly significant in the construction industry is the ability to facilitate collaboration and data exchange among all the project stakeholders. Architects, engineers, builders, estimators can efficiently work together, minimizing errors and collision, while reducing time and resource wastage. BIM technologies allow for the creation of digital twins of structures, which can be used for a variety of scenarios: efficient property management, minimizing environmental impact, cost calculations for construction, operation and maintenance. Classification systems are used to solve these tasks. In order to organize the huge amount of data involved in BIM-model.

The classification of information models allows to structure and organize the huge data volumes, simplify access to information and makes it more understandable and manageable. It also enables to extract the necessary project data quickly, reducing the probability of errors during information searches and enhancing communication among project participants. Structured and classified data integrates easily into various software and systems for automating planning, management, and construction control processes, facilitating decision-making and improving efficiency.

There are various classification systems such as UniFormat, MasterFormat 2016, OmniClass, UniClass 2015, CoClass, as well as CCS and Talo classification systems. All of them are widely used in the international construction practices.

Uniformat is a classification system commonly used for organizing information about construction projects and infrastructure. This classifier divides project data into four levels of hierarchy from a general description of the object to the detailed technical characteristics [1]. The standard is based on functional elements or components, without considering the materials and methods used for their execution. The system can be used to ensure consistency in the economic evaluation of construction projects.

MasterFormat 2016 is a classification system for organizing construction information and documentation in North America [2]. This system was developed by the Construction Specifications Institute (CSI) and Construction Specifications Canada (CSC). It provides a structure for cataloging and describing various elements and processes related to construction and renovation. MasterFormat is organized as a hierarchical structure with divisions into different sections, groups, and subgroups. Each element is assigned a unique number, making the system easy to manage and access. MasterFormat is widely used in the construction industry and serves as a standard for organizing construction documentation, including specifications, contracts and budgets.

OmniClass is a classification and categorization system for construction information designed to more efficiently organize data in the construction and engineering industry [3]. It is an integrated system, that combines both MasterFormat and Uniformat. It allows it to cover various aspects of projects, ranging from technical specifications to financial reporting.

Uniclass 2015 is a classification system for construction information developed by the Royal Institute of British Architects (RIBA) in the UK [4]. This system is designed to simplify and standardize the description of construction projects and information within the construction industry. Uniclass 2015 is the standard classification system in the UK and widely used in the development, documentation, and management of construction projects. The system covers various disciplines, including architecture, engineering, construction, and more, making it valuable for different participants in the construction process.

CoClass is a Swedish digital classification system developed in accordance with ISO 12006-2:2015/IEC CD 81346 standards. It was originally created to reduce construction expenses related to communication issues among participants in the construction process at all stages of the object's lifecycle, from early design phases to technical maintenance, operation and demolition [5].

CCS (Common Classification System) is a Danish classification system used in Denmark for categorizing and describing construction and engineering projects [6, 7]. It is designed to simplify information exchange and improve data consistency in the construction industry. CCS is widely used in Denmark across various sectors including construction, design and project management. It serves as a standard for organizing information about construction projects.

TALO 2000 classification system is a Finnish classification system developed for organizing and structuring information in the construction and engineering industry in Finland [8, 9]. The name "TALO" means "house" in Finnish, implying that the system is used for classifying construction objects and related information. Here are some key characteristics of the Finnish classification system TALO 2000.

CIC (Construction information classifier) is the Russian national classification system for construction information, based on international standards ISO 12006 and IEC 81346. It is organized through a systematic approach, creating a structured set of disparate objects [10, 11]. All classes of construction information and their corresponding classification tables according to ISO 12006-2:2015 and fall into one of four basic categories of construction information: resource, process, result, and characteristic. Codes are assigned by combining numbers and letters.

Each of these systems provides encoding for elements of building information model based on their purpose, location, materials and other attributes. The more complex the classifier the broader its application in projects. But it may require more time for specialists to encode model elements.

Currently, information models can be classified in editable formats. This introduces certain complexities in terms of creation, coding method and subsequent handling of coded attributes for conversion into other formats without data loss. Since the information model dynamically changes during the design process, element classification typically occurs at the final modeling stage when exporting to the universal and easily processed IFC format. It is necessary for successful integration with other software, subsequent checks, filtering, use in ERP systems for futher facilitating effective communication among all construction participants.

The Industry Foundation Classes (IFC) format is an open and cross-industry standard for exchanging information about construction and infrastructure objects in a computer graphics format. IFC is designed as an open standard to ensure that the building information model is available for use by all interested parties without licensing restrictions [13]. This promotes compatibility and collaboration between different programs and systems dealing with construction data.

IFC is designed to cover a wide range of industries related to design, construction, and facility management, including architecture, engineering systems, construction, asset management and others. This allows to create comprehensive models and data exchange between various specialized systems.

The format used an object-oriented approach to represent information. Each element is presented as an object with a set of properties and attributes, including both semantic descriptions (purpose, characteristics, etc.) and their geometric representations, providing the capability for visualization and analysis of construction objects in a three-dimensional environment.

The IFC format is not tied to a specific platform or program. It makes it usable in various applications, such as design software (e.g., BIM applications), construction management systems and others.

Classifying the model in the IFC format offers the possibility to encode both user-defined attributes and as part of the IFC structure, specifically within IfcClassification. This structure carries several advantages. Classification information embedded in the hierarchical IFC structure can be provided in two ways: through an external reference to the classification, describing IfcClassification, containing the classification name, edition, and resource location; or through IfcClassification with IfcClassificationReference as classification notations, facilitating the integration of the classification system structure into exchange systems.

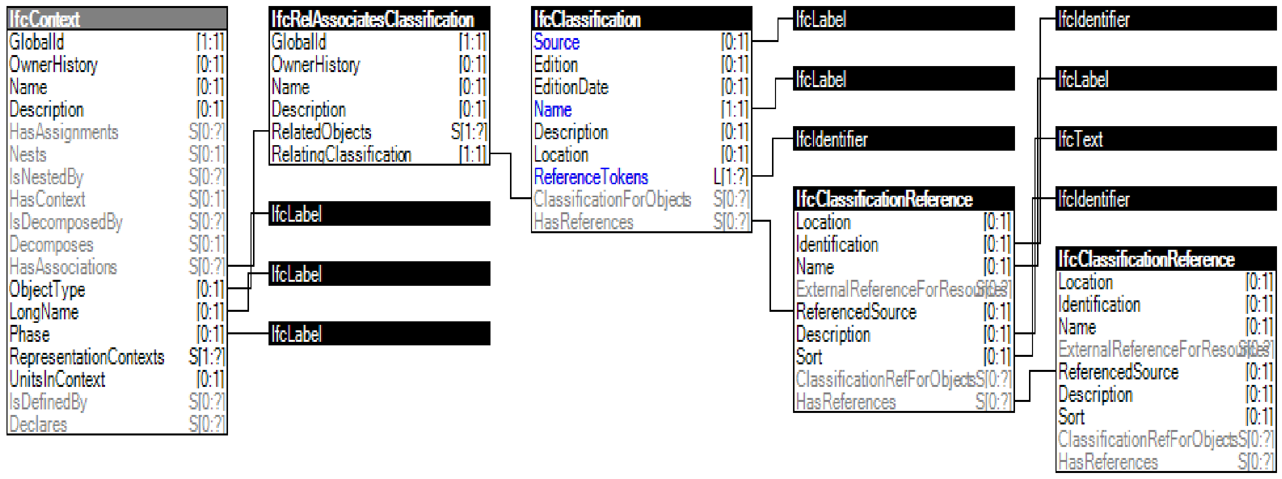


Figure 1. The structure of IfcClassification within BuildingSMART [14].

The interconnection of each Industry Foundation Classes (IFC) element allows the reliable and structured information transfer about construction objects, their components and classification among various software programs and systems. This promotes more efficient design, construction, operation and management of real estate. Therefore, the open format is the most suitable for the encoding process for subsequent data processing.

The purpose of this study was to find an effective machine learning model for automated classification of elements in the open cross-industry standard, Industry Foundation Classes (IFC).

2. *Methods*

The classification process in the formation of a building information model today can be organized manually, semi-automatically and automatically [15]. An automatic method, based on machine learning techniques was considered to solve the classification and coding tasks. Machine learning as a subfield of artificial intelligence (AI) focused on developing algorithms and models capable of learning and making predictions based on data. This technology enables to analyze the large volumes of data and to discover the hidden patterns by grouping data based on similarity without predefined categories. It also involves training models on labeled data, where both input and corresponding output data are known.

There was made decision to use neural networks, which belong to deep learning, to find a high-quality machine learning model. Neural networks utilize interconnected nodes or neurons in a layered structure, resembling the human brain.

In this research, convolutional neural networks (CNN) and recurrent neural networks (RNN) were analyzed as the primary options in the search for a suitable machine learning model for classifying elements in information models [16, 17].

Recurrent Neural Networks (RNN) are a class of deep neural networks widely used in processing sequential data such as texts and time series. There are several variants of RNNs, including standard RNNs, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). For the classification task the most effective recurrent neural network was LSTM (Long Short-Term Memory), an advanced type of RNN, known for its powerful capabilities in analyzing sequential data [18].

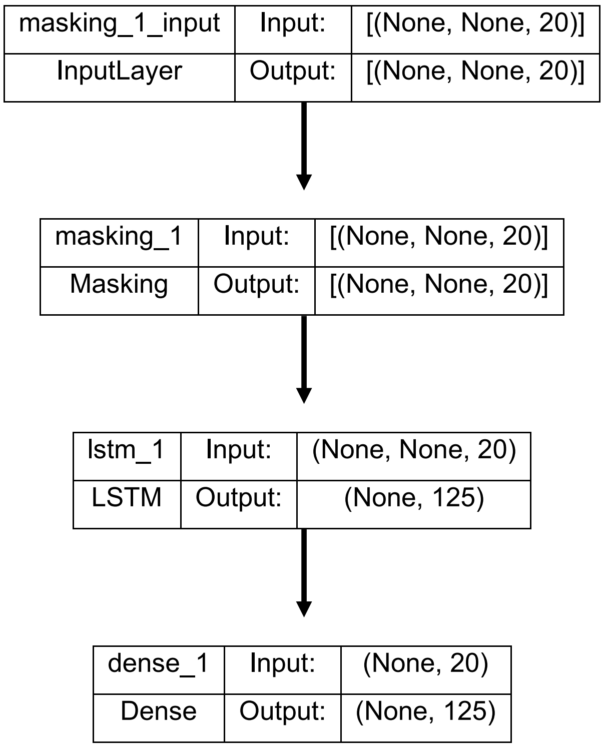


Figure 2. LSTM Architecture.

During the testing this neural network the prediction accuracy did not reach 30 % (Fig. 2). It has taken 2.5 hours to train the specified configuration. For this reason, it was hypothesized that approaches based solely on LSTM are not suitable for the considered case or, at the very least, not for the considered method of extracting information from the text. As a result, the study of recurrent networks was finished without achieving acceptable results.

The next research stage involved the consideration the various variations of Convolutional Neural Networks (CNNs). It is a class of deep neural networks, designed for processing and analyzing data related to images and videos and are also effectively used in text classification: CNN_1 and CNN_2.

CNN_1 was presented as local features with the summation of responses from different parts of the input (Fig. 3).

The implementation used keras-2.10.0 (tensorflow-2.10.0). The activation function on the final layer was 'softmax'. Activation functions were not applied on the other layers, they were linear. L1 regularization was applied on the first of the fully connected layers (default coefficient = 0.01). Class weights were not specified, as specifying them resulted in reduced accuracy due to a preference for less common classes in disputed situations. The number of epochs was set to 5.

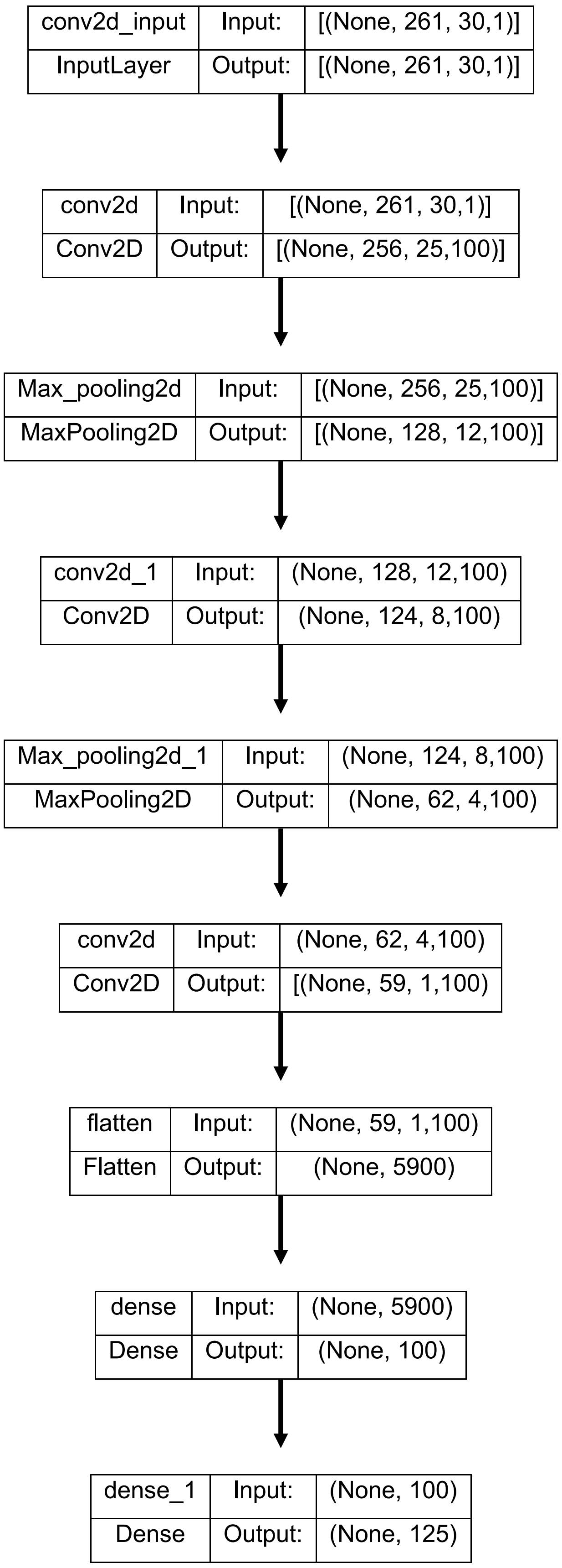

conv2d_input | Input: | [(None, 261, 30,1)]
InputLayer | Output: | [(None, 261, 30,1)]
conv2d | Input: | [(None, 261, 30,1)]
Conv2D | Output: | [(None, 256, 25,100)]
Max_pooling2d | Input: | [(None, 256, 25,100)]
MaxPooling2D | Output: | [(None, 128, 12,100)]
conv2d_1 | Input: | (None, 128, 12,100)
Conv2D | Output: | (None, 124, 8,100)
Max_pooling2d_1 | Input: | (None, 124, 8,100)
MaxPooling2D | Output: | (None, 62, 4,100)
conv2d | Input: | (None, 62, 4,100)
Conv2D | Output: | [(None, 59, 1,100)
flatten | Input: | (None, 59, 1,100)
Flatten | Output: | (None, 5900)
dense | Input: | (None, 5900)
Dense | Output: | (None, 100)
dense_1 | Input: | (None, 100)
Dense | Output: | (None, 125)

Figure 3. CNN_1 Architecture.

CNN_2 was characterized by the extraction of local features without considering responses from other positions (Fig. 4).

The implementation used keras-2.10.0 (tensorflow-2.10.0). The activation function on the final layer was 'softmax', and on the other layers, it was linear. Class weights were not specified, as specifying them resulted in reduced accuracy due to a preference for less common classes in disputed situations. The number of epochs was set to 5.

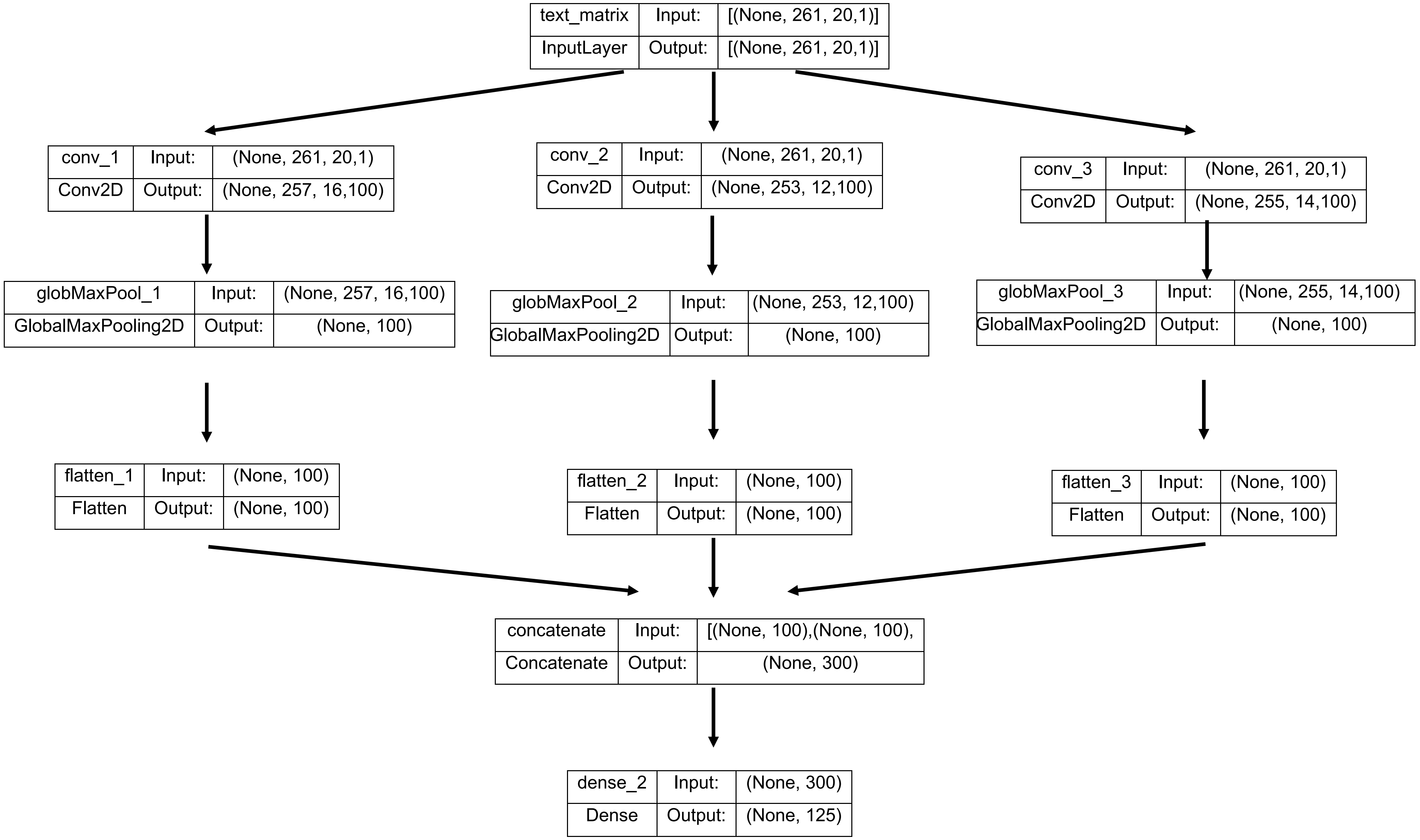

text_matrix | Input: | [(None, 261, 20,1)]
InputLayer | Output: | [(None, 261, 20,1)]
conv_1 | Input: | (None, 261, 20,1)
Conv2D | Output: | (None, 257, 16,100)
conv_2 | Input: | (None, 261, 20,1)
Conv2D | Output: | (None, 253, 12,100)
conv_3 | Input: | (None, 261, 20,1)
Conv2D | Output: | (None, 255, 14,100)
globMaxPool_1 | Input: | (None, 257, 16,100)
GlobalMaxPooling2D | Output: | (None, 100)
globMaxPool_2 | Input: | (None, 253, 12,100)
GlobalMaxPooling2D | Output: | (None, 100)
globMaxPool_3 | Input: | (None, 255, 14,100)
GlobalMaxPooling2D | Output: | (None, 100)
flatten_1 | Input: | (None, 100)
Flatten | Output: | (None, 100)
flatten_2 | Input: | (None, 100)
Flatten | Output: | (None, 100)
flatten_3 | Input: | (None, 100)
Flatten | Output: | (None, 100)
concatenate | Input: | [(None, 100),(None, 100),
Concatenate | Output: | (None, 300)
dense_2 | Input: | (None, 300)
Dense | Output: | (None, 125)

Figure 4. CNN_2 Architecture.

Both CNN configurations were trained in 10 minutes on Google Colab. The average prediction time was 1 ms per record (taking into account preprocessing of the received record).

In the next stages, machine learning models were considered, that use record correction. One of them is the Naive Bayes classifier – a statistical machine learning algorithm used for classification tasks. It is based on Bayes' theorem and makes a "naive" assumption of independence among features, which may not hold true in real data but makes the algorithm computationally efficient and easy to implement. The main idea of the algorithm is to use Bayes' theorem to estimate the probability of an object belonging to a particular class based on its features [19]:

$$\mathrm{P}\left(\mathrm{Class}\middle|\mathrm{Features}\right)=\frac{\mathrm{P}\left(\mathrm{Features}\middle|\mathrm{Class}\right)*\mathrm{P}\left(\mathrm{Class}\right)}{\mathrm{P}\left(\mathrm{Features}\right)},$$

where $\mathrm{P}\left(\mathrm{Class}\middle|\mathrm{Features}\right)$ is the conditional probability that an object belongs to a class given the available features; $\mathrm{P}\left(\mathrm{Features}\middle|\mathrm{Class}\right)$ is the conditional probability of having those features if the object indeed belongs to the given class; $\mathrm{P}\left(\mathrm{Class}\right)$ is the prior probability of the class, i.e., the probability of an object belonging to that class without considering any features; $\mathrm{P}\left(\mathrm{Features}\right)$ is the marginal probability of the features occurring, regardless of the class.

The algorithm made the naive assumption of feature independence. It means, that the probability of all features occurring in a class was calculated as the product of the probabilities of each feature occurring in that class. After training, the Naive Bayes classifier used probability estimates to determine the class of a new object, selecting the class with the highest probability.

During the testing, it was assumed, that the same words in the elements descriptions of different classes occurred with different probabilities. As a result, conditional probabilities were used for class prediction. An implementation of the Naive Bayes algorithm from scikit-learn v1.1.2 was used.

The next considered machine learning algorithm was the Random Forest, which builds multiple decision trees during training. Each tree was trained on a random subsample of the data (with replacement) and a subset of features, making each tree in the ensemble different [20]. For classification, each tree in the ensemble voted for the predicted class and the class with the most votes became the final prediction of the algorithm.

An implementation from scikit-learn v1.1.2 was used with 500 trees, maximum depth of 100, minimum samples per leaf set to 2, minimum samples for node splitting set to 4, bootstrap samples were used, class weights were computed inversely proportional to the class frequencies in the subsample for each tree separately (‘balanced_subsample’) and other parameters remained at their defaults.

Additionally, simple argmax was applied to the "vector of class probabilities." This approach was based on the idea that each word could be associated with a set of classes in which it appeared during training. Initially, each word was associated with a vector of zeros, the length of which was determined by the number of classes. Each class corresponded to a position in the vector. If a word appeared in the descriptions of elements in a particular class, a 1 was placed in the corresponding cell. Finally, vector normalization was performed using l_1 normalization. In this case, words corresponding to a smaller set of classes had higher significance.

When working with the recordings, all word vectors were summed and l_1 normalization was performed, resulting in a "vector of class probabilities" for each recording.

This approach should not be perceived as a standalone classification method due to the low likelihood of achieving good results. However, it is important to consider that, in general, this method can be used as a way to extract information from the text.

The method had an obvious drawback: two words that appeared in the same set of classes but with different probabilities were indistinguishable. Using the values in vectors corresponding to words as the proportion of class elements in which the word appeared led to a decrease in the accuracy of models built upon this approach. Less frequent classes with lower diversity tended to attract elements from more frequent classes with higher diversity.

On the next stage a random forest was applied to the "vector of class probabilities" – implemented with scikit-learn v1.1.2. The number of trees was 400, the maximum depth was 100, the minimum number of elements in a leaf was 2, the minimum number of elements for node splitting was 4, bootstrap samples were used and class weights were calculated inversely proportional to the class frequency in the subsample for each tree separately ('balanced_subsample'). Other parameters were kept at their defaults.

The CatBoost model was applied to the "vector of class probabilities" using catboost-1.1.1 with the following parameters:

- loss_function = 'MultiClass', learning_rate = 0.05, n_estimators = 2000, max_depth=7,
- auto_class_weights='Balanced', while other parameters remained at their defaults.

The use of boosting for other models was not applied, as it would lead to overfitting for strong classifiers. For example, the vectors of probabilities obtained from random forests did not generally allow distinguishing incorrectly predicted objects from those correctly predicted because they were too similar.

The training time for the considered models varied from 3 to 20 minutes, with the Bayesian algorithm being the quickest to train (due to data preprocessing) and CatBoost being the slowest. The average prediction time ranged from 1 ms for the Bayesian algorithm to 1.6 ms for CatBoost (considering data preprocessing).

3. Results and Discussion

As a result of testing the convolutional neural networks, CNN_1 showed an average F1-score of 0.9474 for class prediction, where the "FSC" code was not recognized. As a result, elements belonging to this class were not subsequently encoded.

Next, the Convolutional Neural Network CNN_2 was tested, CNN_2 showed a lower average F1-score compared to CNN_1, with an F1-score (macro) of 0.9037, resulting in an increased number of unrecognized classes including “BUA”, “KNA” codes.

The results of testing Convolutional Neural Networks (CNN) were summarized in Table 1, including accuracy and F1-score (macro) values.

Table 1. Testing convolutional neural networks.

	CNN_1	*CNN_2*
accuracy	0.9904	0.9885
F1-score (macro)	0.9474	0.9037

During testing, the Naive Bayes classifier (complementNB**)** using record correction showed an F1-score (macro) of 0.8627, with a larger number of classes not being recognized compared to neural networks. Elements belonging to classes of “FSC”, “NAB”, “NED”, “WMC” were not encoded.

After the Complement Naive Bayes the Random Forest was tested. The average F1-score (macro) during testing of the Random Forest was 0.9484, which resulted in higher prediction accuracy compared to the Naive Bayesian classifier. Only code RUA was not recognized.

When using a straightforward argmax approach on the "vector of class probabilities," the average F1-score was slightly lower than Random Forest, at 0.9249, but some classes “FSC”, “NAB” went unrecognized.

Random Forest using the "probability class vector" achieved 0.9717, providing nearly perfect accuracy in predicting codes 0.9947.

CatBoost with "class probability vector" exhibited the highest F1-score among other models, reaching 0.9749 with a maximum accuracy of 0.9956.

The results of models using record correction are summarized in Table 2.

Table 2. Results of models using record correction.

	Bayesian classifier	Random Forest	Simple argmax extraction on the "vector of class probabilities"	Random Forest on the "vector of class probabilities"	CatBoost on the "vector of class probabilities"
accuracy	0.9645	0.9875	0.9723	0.9947	0.9956
F1-score (macro)	0.8627	0.9484	0.9249	0.9717	0.9749

As a result of the research, among the models that do not use record correction, the Convolutional Neural Network CNN_1 was chosen due to its higher accuracy of 0.99 and F-score of 0.95. Among the models that use record correction, CatBoost was selected with the following characteristics: accuracy of 0.9956 and F1-score of 0.9749.

The results of the research coincided with previous studies, the Convolutional Neural Networks demonstrate high accuracy in natural language processing [21], CatBoost surpasses other publicly available implementations of quality improvement in terms of quality on various datasets among the models that use record correction [22].

For practical application, software named "Impulse" was developed, allowing the encoding of elements in formats such as rvt, tekla, and ifc based on the user's chosen classifier.

To effectively use the "Impulse" platform for automatically assigning codes to model elements based on a specific classifier, the following stages are anticipated (Figure 5):

1. Classifier selection for encoding;
2. Selection of attributes for code recording;
3. Identification of elements subject to further classification;
4. Automatic encoding;
5. Validation and correction of assigned codes;
6. Writing codes to the attributes of information model elements.

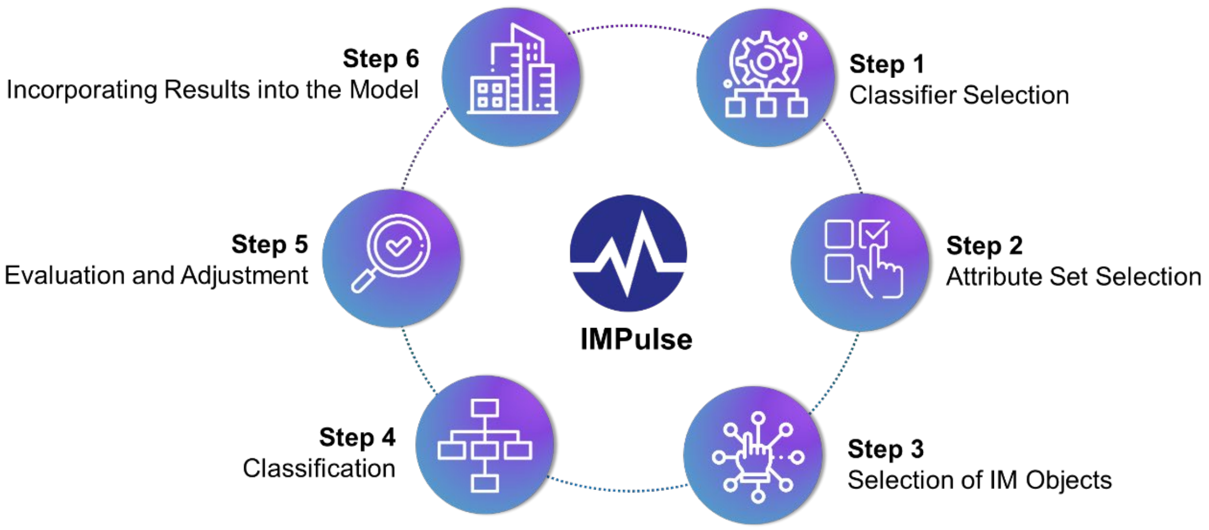


Figure 5. Workflow of the "Impulse" Application.

The algorithm of the system with a machine learning model consists of stages, depicted in Figure 6. When unclassified data appears, a specialist further trains the model by manually inputting the code into the attribute through the application or by selecting a code suggested by the system, resulting in labeled elements becoming training data for the machine learning algorithm.

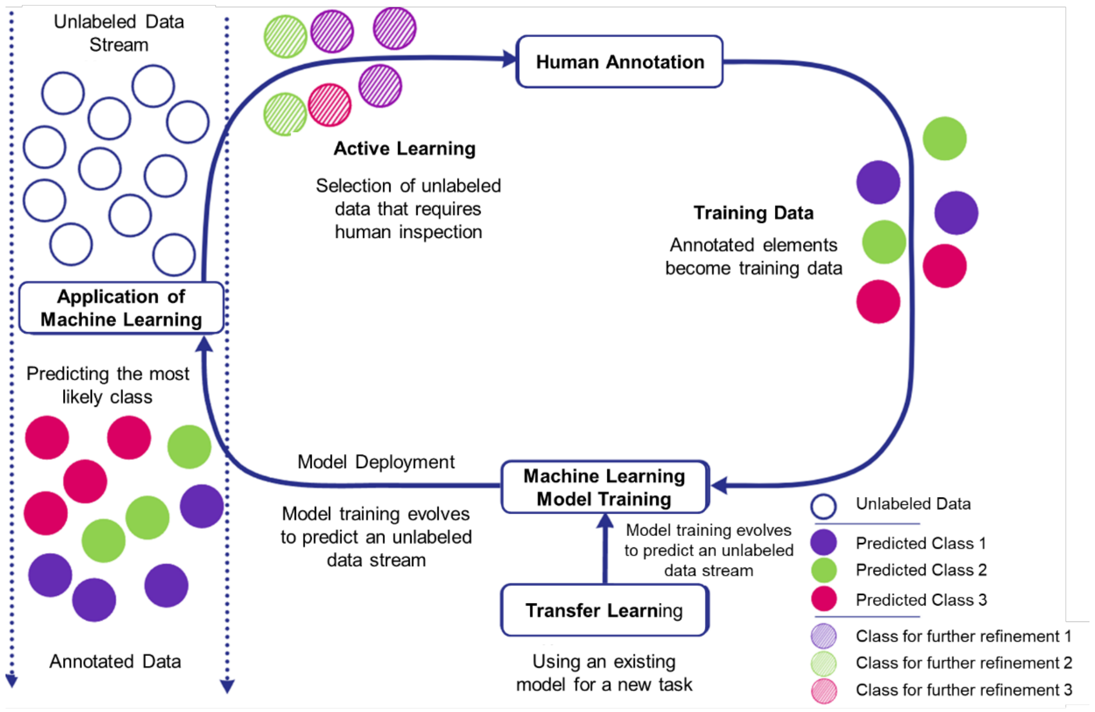
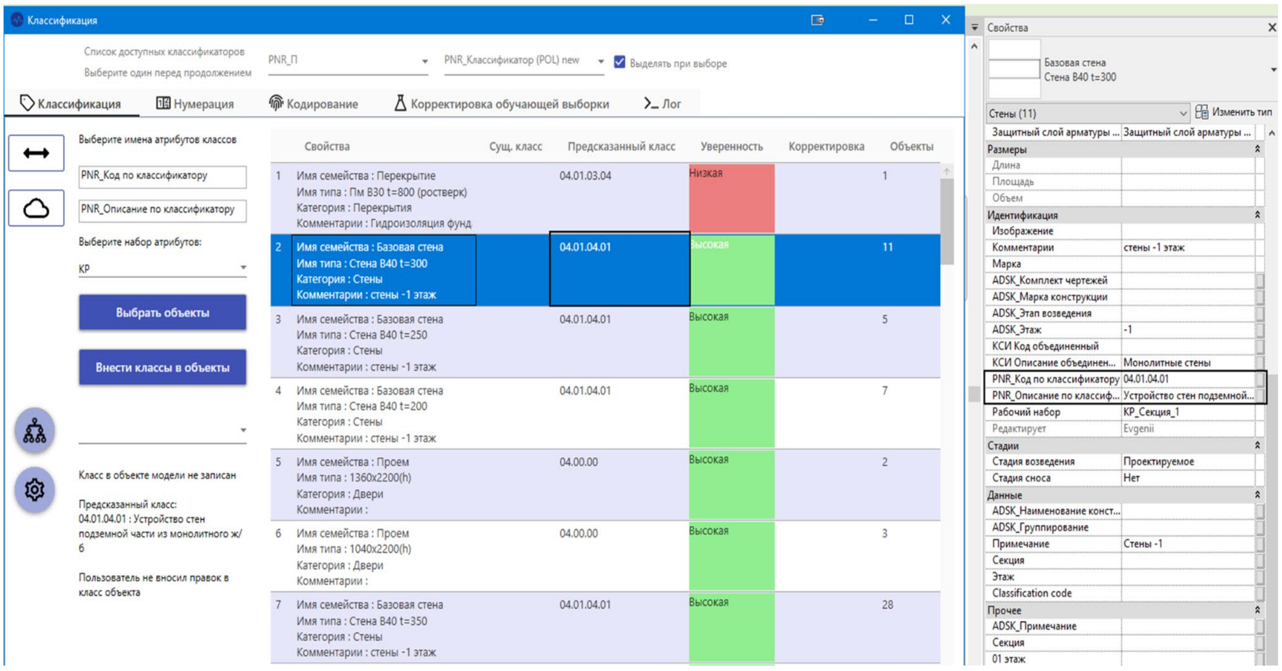

Unlabeled Data Stream
Human Annotation
Active Learning
Selection of unlabeled data that requires human inspection
Training Data
Annotated elements become training data
Application of Machine Learning
Predicting the most likely class
Model Deployment
Model training evolves to predict an unlabeled data stream
Machine Learning Model Training
Model training evolves to predict an unlabeled data stream
Transfer Learning
Using an existing model for a new task
Annotated Data
Unlabeled Data
Predicted Class 1
Predicted Class 2
Predicted Class 3
Class for further refinement 1
Class for further refinement 2
Class for further refinement 3

Figure 6. Architecture of the "Impulse" Software.

During the validation and adjustment of the selected code for a clear representation of the numeric characteristic, "Impulse" displays a confidence level – a metric that reflects the frequency of encoding an element with similar attributes using the predicted code (Figure 7). The confidence level allows specialists to identify and recognize incorrectly selected codes and replace them with the correct ones by retraining the model.



Выберите имена атрибутов классов

PNR_Код по классификатору

PNR_Описание по классификатору

Выберите набор атрибутов:

KP

Класс в объекте модели не записан

Предсказанный класс:
04.01.04.01 : Устройство стен подземной части из монолитного ж/ 6

Пользователь не вносил правок в класс объекта

	Свойства	Сущ. класс	Предсказанный класс	Уверенность	Корректировка	Объекты
1	Имя семейства : Перекрытие Имя типа : Пм B30 t=800 (ростверк) Категория : Перекрытия Комментарии : Гидроизоляция фунд		04.01.03.04	🟥 Низкая		1
2	Имя семейства : Базовая стена Имя типа : Стена B40 t=300 Категория : Стены Комментарии : стены -1 этаж		04.01.04.01	🟩 Высокая		11
3	Имя семейства : Базовая стена Имя типа : Стена B40 t=250 Категория : Стены Комментарии : стены -1 этаж		04.01.04.01	🟩 Высокая		5
4	Имя семейства : Базовая стена Имя типа : Стена B40 t=200 Категория : Стены Комментарии : стены -1 этаж		04.01.04.01	🟩 Высокая		7
5	Имя семейства : Проем Имя типа : 1360x2200(h) Категория : Двери Комментарии :		04.00.00	🟩 Высокая		2
6	Имя семейства : Проем Имя типа : 1040x2200(h) Категория : Двери Комментарии :		04.00.00	🟩 Высокая		3
7	Имя семейства : Базовая стена Имя типа : Стена B40 t=350 Категория : Стены Комментарии : стены -1 этаж		04.01.04.01	🟩 Высокая		28

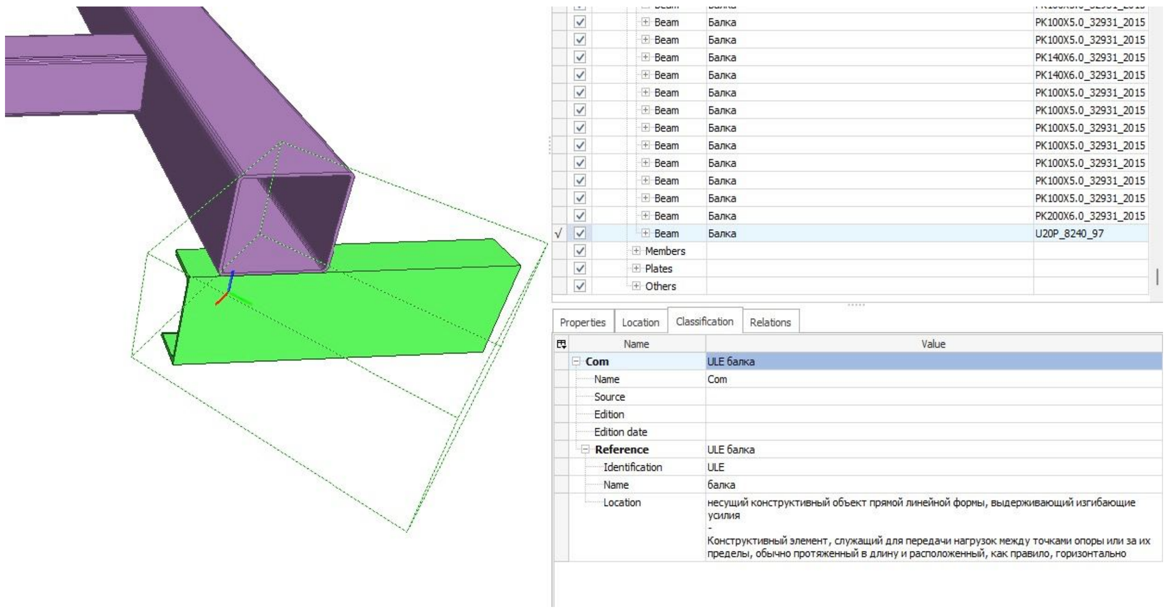
Базовая стена
Стена B40 t=300

Стены (11)

Защитный слой арматуры ...	Защитный слой арматуры ...
Размеры	
Длина	
Площадь	
Объем	
Идентификация	
Изображение	
Комментарии	стены -1 этаж
Марка	
ADSK_Комплект чертежей	
ADSK_Марка конструкции	
ADSK_Этап возведения	
ADSK_Этаж	-1
КСИ Код объединенный	
КСИ Описание объединен...	Монолитные стены
PNR_Код по классификатору	04.01.04.01
PNR_Описание по классиф...	Устройство стен подземной...
Рабочий набор	KP_Секция_1
Редактирует	Evgenii
Стадии	
Стадия возведения	Проектируемое
Стадия сноса	Нет
Данные	
ADSK_Наименование конст...	
ADSK_Группирование	
Примечание	Стены -1
Секция	
Этаж	
Classification code	
Прочее	
ADSK_Примечание	
Секция	
01 этаж	

Figure 7. User Interface of the "Impulse" Software.

Code recording can be done in both user-defined attributes and in the root IFC classification class, enabling navigation from code to all elements with the same code and vice versa (Figure 8).



	✓	Beam	Балка	PK100X5.0_32931_2015
	✓	Beam	Балка	PK100X5.0_32931_2015
	✓	Beam	Балка	PK140X6.0_32931_2015
	✓	Beam	Балка	PK140X6.0_32931_2015
	✓	Beam	Балка	PK100X5.0_32931_2015
	✓	Beam	Балка	PK100X5.0_32931_2015
	✓	Beam	Балка	PK100X5.0_32931_2015
	✓	Beam	Балка	PK100X5.0_32931_2015
	✓	Beam	Балка	PK100X5.0_32931_2015
	✓	Beam	Балка	PK100X5.0_32931_2015
	✓	Beam	Балка	PK100X5.0_32931_2015
	✓	Beam	Балка	PK200X6.0_32931_2015
√	✓	Beam	Балка	U20P_8240_97
	✓	Members		
	✓	Plates		
	✓	Others		

Name	Value
Com	ULE балка
Name	Com
Source	
Edition	
Edition date	
Reference	ULE балка
Identification	ULE
Name	балка
Location	несущий конструктивный объект прямой линейной формы, выдерживающий изгибающие усилия - Конструктивный элемент, служащий для передачи нагрузок между точками опоры или за их пределы, обычно протяженный в длину и расположенный, как правило, горизонтально

Figure 8. Code Recording Using the "Impulse" Software.

There was conducted a building information models (BIM) classification of civil and industrial objects with a specific number of elements in each model to test the "Impulse" software, based on machine learning models. The steps were following:

– training the machine learning model on a sample group of elements;

– classification the selected group of elements, validation and correction the codes obtained;

– encoding the remaining elements or the next group of the information model.

There was made an element-by-element classification to compare the time spent on encoding BIM elements using the "Impulse" tool and the manual method. The results were summarized in Table 3.

Analyzing the obtained results, the automated classification process showed a reduction in time compared to manual coding by an average factor of 2.54.

Table 3. Comparison of "Impulse" Automated Classification and Manual Classification.

№ of Information Model	Work in "Impulse" Software						Time Spent on Manual Classification, min
	Stage	Total Number of Groups, pcs.	Number of Incorrectly Predicted Elements, pcs.	Accuracy, %	Time Spent on Stage, min.	Total Time Spent on Automatic Classification, min.	
1	Stage 1 (Training)	255	-	-	17	66	136
	Stage 2 (Validation and Retraining)	255	66	74.1	15		
	Stage 3 (Validation and Retraining)	255	157	38.4	14		
	Stage 4 (Validation and Retraining)	255	96	62.4	12		
	Stage 5 (Validation)	255	22	91.4	8		
2	Stage 1 (Training)	884	-	-	45	111	175
	Stage 2 (Validation and Retraining)	884	26	97.1	36		
	Stage 3 (Validation and Retraining)	884	27	96.9	19		
	Stage 4 (Validation)	884	11	98.8	11		
3	Stage 1 (Training)	86	-	-	4	9	24
	Stage 2 (Validation and Retraining)	86	8	90.7	3		
	Stage 3 (Validation)	86	0	100.0	2		
4	Stage 1 (Training)	155	-		9	24	56
	Stage 2 (Validation and Retraining)	155	5	96.8	5		
	Stage 3 (Validation and Retraining)	155	6	96.1	4		
	Stage 4 (Validation and Retraining)	155	18	88.4	4		
	Stage 5 (Validation)	155	9	94.2	2		
5	Stage 1 (Training)	443	-	-	23	63	136
	Stage 2 (Validation and Retraining)	443	53	88.0	20		
	Stage 3 (Validation and Retraining)	443	49	88.9	13		
	Stage 4 (Validation)	443	19	95.7	7		

№ of Information Model	Work in "Impulse" Software						Time Spent on Manual Classification, min
	Stage	Total Number of Groups, pcs.	Number of Incorrectly Predicted Elements, pcs.	Accuracy, %	Time Spent on Stage, min.	Total Time Spent on Automatic Classification, min.	
6	Stage 1 (Training)	68	-		4	8	25
	Stage 2 (Validation)	68	0	100.0	2		
7	Stage 1 (Automatic Classification and Validation)	201	112	44.3	14	26	74
	Stage 2 (Validation)	201	2	99.0	7		
	Stage 3 (Model Update after Refinement and Validation)	186	2	98.9	5		
8	Stage (Automatic Classification and Validation)	372	277	25.5	10	10	34
9	Stage (Automatic Classification and Validation	248	150	39.5	5	5	17

4. Conclusions

The conducted research resulted in a numerical comparative analysis of classification models: models without data correction - convolutional neural networks and models with data correction – Naive Bayes classifier, Random Forest, simple argmax extraction on the "class probability vector," random forest on the "class probability vector", and CatBoost on the "class probability vector".

As a result, the following machine learning models were chosen as effective for automated classification of elements in Information Models within the open cross-industry standard Industry Foundation Classes (IFC): CNN_1, which exhibited the highest accuracy and F-score, equal to 0.99 and 0.95, and CatBoost, with an accuracy of 0.9956 and an F1-score of 0.9749. The obtained research results exceed the accuracy of the forecasts in comparison with the results of other authors on the classification of elements of BIM models.

Software "Impulse" was developed for the practical implementation of these models. It is based on the selected algorithms and tested on civil and industrial purpose Information Models. The developed application has significantly reduced the classification time by an average of 2.54 times, improving the coding accuracy and reducing labor costs. It is recommended to use "Impulse" as a tool for automatic classification of Building Information Models for the substantial optimization the classification timelines and overall efficiency.

References

1. CSI, CSC, UniFormatTM: A Uniform Classification of Construction Systems and Assemblies, 2010.
2. CSI, CSC, MasterFormat® 2018 Edition: Master List of Numbers and Titles for the Construction Industry, 2018.
3. CSI, OmniClassTM: A Strategy for Classifying the Built Environment. Introduction and User’s Guide, first ed., 2006
4. Crawford, M., Cann, J., O’Leary, R. Uniclass: Unified Classification for the Construction Industry, first ed. RIBA Publications Ltd, London, 1997.
5. Svensk Byggtjanst, A.B., ¨ CoClass – Nya Generationen BSAB Klassifikation Och tillampning, 2016. (in Swedish)
6. Pupeikis, D., Navickas, A.A., Klumbyte, E., Seduikyte, L. Comparative study of construction information classification systems: CCI versus Uniclass 2015. Buildings. 2022. 12(5). 656. DOI: 10.3390/buildings12050656
7. ICIS, Comparison of OmniClass, Uniclass, Cuneco and CoClass with Reference to ISO 12006-2 and ISO 81346-12, 2018. Zurich.
8. Knopp-Trendafilova, A., Suomi, J., Tauriainen, M. Link between a structural model of buildings and classification systems in construction, in: K. Belloni, Kojima Jun, I. Pinto-Sepp¨ a (Eds.). Proceedings of the 1st International Conference on Improving Construction and Use through Integrated Design Solutions. CIB IDS 2009, 2009. 259. Pp. 285–301.
9. Rakennustieto Oy, The Finnish Construction 2000 Classification System, 2016.
10. Volkodav, V.A., Volkodav, I.A. Development of the structure and composition of a building information classifier towards the application of BIM technologies. Vestnik MGSU. 2020. 15(6). Pp. 867–906. DOI: 10.22227/1997-0935.2020.6.867-906
11. Timchenko, V.S., Volkodav, V.A., Volkodav, I.A., Timchenko, O.V., Osipov, N.A. Development of classification tables “Process management”, “Design processes” and “Information” of the classifier of construction information for creating and maintaining information models of capital construction objects. Vestnik MGSU. 2021. 16(7). Pp. 926–954. DOI: 10.22227/1997-0935.2021.7.926-954
12. Qiu, Q., Zhou, X., Zhao, J., Yang, Y., Tian, Sh., Wang, J., Liu, J., Liu, H. From sketch BIM to design BIM: An element identification approach using Industry Foundation Classes and object recognition. Building and Environment. 2021. 188. 107423. DOI: 10.1016/j.buildenv.2020.107423
13. Park, S.I., Lee, S.-H., Almasi, A., Song, J.-H. Extended IFC-based strong form meshfree collocation analysis of a bridge structure. Automation in Construction. 2020. 119. 103364. DOI: 10.1016/j.autcon.2020.103364
14. BuildingSMART International, 2020. https://standards.buildingsmart.org/IFC/RELEASE/IFC4/FINAL/HTML/link/project-classification-information.htm (Accessed 14 September 2023).
15. Petrochenko, M.V., Nedviga, P.N., Kukina, A.A., Sherstyuk, V.V. Classification of information models in BIM using artificial intelligence algorithms. Vestnik MGSU. 2022. 17(11). DOI: 10.22227/1997-0935.2022.11
16. Huang, Sh.-Yu., An, W.-J., Zhang, D.-Sh, Zhou, N.R. Image classification and adversarial robustness analysis based on hybrid quantum–classical convolutional neural network. Optics Communications. 2023. 533. 129287. DOI: 10.1016/j.optcom.2023.129287
17. Babalhavaeji, A., Radmanesh, M., Jalili, M., Gonzalez, S.A. Photovoltaic generation forecasting using convolutional and recurrent neural networks. Energy Reports. 2023. 9. Suppl. 12. Pp. 119–123. DOI: 10.1016/j.egyr.2023.09.149
18. Ahmad, M.W., Mourshed, M., Rezgui, Ya. Tree-based ensemble methods for predicting PV power generation and their comparison with support vector regression. Energy. 2018. 164. Pp. 465–474. DOI: 10.1016/j.energy.2018.08.207
19. Diab, D.M., El Hindi, K.M. Using differential evolution for fine tuning naïve Bayesian classifiers and its application for text classification. Applied Soft Computing. 2017. 54. Pp. 183–199. DOI: 10.1016/j.asoc.2016.12.043
20. Sun, Zh., Wang, G., Li, P., Wang, H., Zhang, M., Liang, X. An improved random forest based on the classification accuracy and correlation measurement of decision trees. Expert Systems with Applications. 2024. 237. Part B. 121549. DOI: 10.1016/j.eswa.2023.121549
21. Kane, B., Rossi, F., Guinaudeau, O., Chiesa, V., Quénel, I., Chau, S., Joint intent detection and slot filling via CNN-LSTM-CRF. 2020 6th IEEE Congress on Information Science and Technology. CiSt, 2020. Pp. 342–347. DOI: 10.1109/CiSt49399.2021.9357183
22. Song, Ch.M. Data construction methodology for convolution neural network based daily runoff prediction and assessment of its applicability. Journal of Hydrology. 2022. Vol. 605. 127324. DOI: 10.1016/j.jhydrol.2021.127324

Information about authors:

Marina Petrochenko, *PhD in Technical Sciences*
ORCID: https://orcid.org/0000-0002-4865-5319
E-mail: mpetroch@mail.ru

Pavel Nedviga,
E-mail: pavel.nedviga@gmail.com

Anna Kukina,
ORCID: https://orcid.org/0000-0003-4271-7408
E-mail: kukina_aa@spbstu.ru

Kseniya Strelets, *PhD in Technical Sciences*
E-mail: kstrelets@mail.ru

Valeriya Sherstyuk,
ORCID: https://orcid.org/0000-0002-5644-5629
E-mail: sherstyuk2.vv@yandex.ru

Received 24.10.2023. Approved after reviewing 29.12.2023. Accepted 29.12.2023.