Web Application with GUI for Data Analysis Automation*

M.S. Manakhova[I], V.A. Dudarev[II]

[I] National Research University Higher School of Economics, Moscow, Russia
[II] A.A. Baikov Institute of Metallurgy and Materials Science of RAS, Moscow, Russia

Abstract. In the current digital age, the world has a huge amount of data. Therefore, people are more and more confronted with the use of such methods as data analysis and machine learning. Moreover, many people are considering using machine learning algorithms for their own purposes. However, data analysis is a complex process that can hardly be carried out by people who do not have sufficient knowledge both in this field and in programming. This paper presents an approach to give non-expert users the ability to apply machine learning algorithms to their datasets using an application with a graphical interface. There are a lot of challenges involved in creating ML-solutions, even if we take advantage of existing ML-algorithms: feature engineering, outliers' detection, filling the missing values, ML-method's hyperparameters optimization and so on. The main point of the research is to find a balance in solving these complex tasks and to provide a Web-based user interface for unexperienced people to enable them to utilize the power of ML-methods in automatic or semi-automatic way. The practical outcome is an information system development, that consists of three interrelated parts: a web application, an API and several microservices that implement ML-algorithms from Scikit-learn library.

Keywords: *web Application, Graphical User Interface, data analysis, ML automation.*
DOI: 10.14357/20790279230107

Introduction

Over the last decade, the level of application of machine learning has grown considerably [1]. Nowadays, it is widely used in the areas such as medicine, marketing, finance, retail, logistics, robotics and so on [2, 3]. Today, almost all companies that collect and store large amounts of data have entire departments involved in data analysis and machine learning. In addition, small teams or even individuals are also interested in using machine learning, because they find data analysis techniques useful for developing their projects and research. For example, scientists who could use the existing data to predict possible values of certain variables or coefficients for conducting scientific experiments.

Simple data analysis such as graph plotting, creating charts and calculating some statistical coefficients can be done using the standard tools such as Microsoft Excel. But often more complex methods involving the use of machine learning models are required. In this case, not only a comprehensive knowledge of linear algebra, mathematical analysis and statistics is required, but also understanding of machine learning concepts and coding experience, particularly in languages such as Python or R, as well as special frameworks and libraries are needed.

Recently, there is a growing number of automated machine learning solutions that allow training and using models for making predictions without writing code in a programming language but using visualization or other methods of user interaction. Nevertheless, some of these solutions, such as Google AutoML have been developed for users with significant experience and are mostly intended to simplify and speed up the work of machine learning engineers and data scientists rather than to lower the entry threshold into this area for non-expert users. Most of the other solutions that can be successfully used by people with basic domain knowledge are often designed to solve only some specific classes of problems, such as computer vision or text analysis.

This paper presents an approach that allows users with basic machine learning knowledge to train and

* This work was performed under partial financial support from the Russian Foundation for Basic Research, projects 20-01-00609 and 21-51-53019, and according to the State Order no. 075-00715-22-00.

use different models for obtaining predictions on the tabular datasets using a graphical interface. The problems encountered in creating systems for automated machine learning and approaches to solve them are also discussed. This work aims to develop a client-server web application with a graphical user interface for data analysis, especially for the supervised learning problems such as regression and classification.

The rest of this paper is organized as follows. Section 2 discusses related work and common challenges included in machine learning process. Then, Section 3 describes the algorithms and methods used in the implementation. Section 4 contains details about the implementation of the web application. Finally, Section 5 provides the conclusion of the work.

1. Literature review

As previously stated, the popularity of solutions for automated machine learning is growing. The paper [4] discusses the possible challenges involved in creating these kinds of solutions.

One of the most challenging problems is the process of extracting features from raw data, called feature engineering. This process often involves handling missing values, encoding of categorical variables, feature scaling and selection. It is notable that feature engineering is not always a simple task even for experts in machine learning and it is usually performed manually by empirical methods.

Missing values are one of the most common problems when it comes to preparing data for use with machine learning models. Human errors, privacy issues, and other factors can lead to the absence of values in the dataset. However, missing values require a correct handling, since most of the models in the existing machine learning libraries are not capable to work with the data that contain gaps.

In fact, there is no clear answer regarding how to handle missing values in datasets. Although, quite several papers discussing the problem of filling gaps in a tabular data have been published, most of the proposed solutions were developed for specific tasks and may not be as useful in other cases. The standard way to solve this problem is to remove features that contain many missing values or to fill in the missing values with some estimate based on other values of the same feature [5].

Most implementations of machine learning algorithms also require all values in the dataset to be represented in a numerical form. Because of this, categorical data must be converted to numerical values before being used for model training or prediction. The simplest way to encode categorical features is label encoding (ordinal encoding). The main idea behind this method is to associate each category with some integer number. This method of encoding is rarely used in practice because of its big disadvantage: it can add additional dependencies that did not exist in the original data, which is critical for linear models, and in general can lead to wrong interpretations of feature values.

One-hot encoding or dummy encoding is the modification that solves the problem of the previously discussed approach. In this algorithm, a new variable, sometimes called a dummy variable, is created for each category of a feature, where a binary value (0 or 1) denotes that a particular feature belongs to a certain category. The main problem with this method and its existing modifications is that a new attribute is created for each unique value of a category variable. Thus, the number of attributes grows quadratically, so naive encoding is only applicable when categorical variables contain a small number of unique values. Another problem of naive encoding is that it produces large number of binary features in the dataset, which can significantly reduce the quality of models when using tree-based algorithms (e.g., decision trees or random forest). In this regard, this algorithm is not suitable for use in systems for automated machine learning, because the number of unique values in the categorical variables in datasets can be quite large.

Another approach to encode categorical variables is target encoding. The idea of target encoding is to use the statistics of a target variable to encode a categorical value. According to the theoretical justification for this approach given in the paper discussing categorical feature preprocessing scheme [6], the key transformation used in this method is a transformation that maps each value of a categorical feature to an estimate of the probability of occurrence of the target variable.

When categorical features are encoded in the training sample, the numerical representation of the categorical feature corresponds to the posterior probability of occurrence of the target feature, provided that the categorical variable takes particular value. In the case of categorical feature encoding in the prediction sample, the numerical representation corresponds to the expected value of the categorical variable. Essentially, this means that for each category, the average value of the target variable is calculated, with which the category is subsequently encoded. This method works for both binary classification and regression. For multiclass classification a similar technique is used, where a categorical variable is encoded with $m - 1$ new variables, where m is the number of classes. It should be noted, that although the author of the article claims that the statistics of the target variable are used to encode the categorical variable, in fact only its mean value is used. Although the mean value is a sufficient statistic for binary classification, it is not suitable for regression because it disregards the intra-category variance of the target variable. In this regard, this algorithm in its pure form is prone to overfitting, so some modifications are often added to software implementations to reduce the probability of overfitting.

In most cases, not all the variables in the raw datasets are useful when building machine learning models. Using many redundant features may reduce the accuracy of the predictions and the generalization ability of the model, as well as dramatically increase its overall complexity.

The paper [7] contains a detailed review of the existing methods for feature selection. As stated, there are three main types of feature selection techniques: filtering, wrapper, and embedded models. Filtering methods are a general set of methods that do not involve the use of a specific machine learning algorithm. They are based on probability theory and statistical approaches and include visual analysis (e.g., construction of a correlation matrix to identify the features that have a weak correlation with the target variable), evaluation of features using some statistical criterion (variance, correlation, χ^2, etc.), and feature ranking by significance. In filtering methods, each feature is considered separately, so it is not possible to identify more complex dependencies in the data, and the resulting subset of features that are most correlated with the target will not always be the subset on which the prediction accuracy will be the highest.

In addition, the existing implementations of these methods often require a choice of a certain threshold value to filter out the redundant features, which is quite difficult to determine automatically, while ensuring equally good quality for datasets that differ in structure (as in the case of automated machine learning systems). In general, these methods are more suitable for a machine learning process fully controlled by user. The main advantage of this class of methods over other feature selection algorithms is a low computational complexity that linearly depends on the total number of features and, consequently, high computation speed. Moreover, filters can be used when the dimensionality of the feature space is larger than the number of observations in the sample, which is not always possible with other methods.

In wrapper methods, the process of feature selection is based on applying some classifier to different subsets of features in the training sample. After selecting the optimal subset, the algorithm is tested on the dataset that was not involved into selection process. This class of methods is divided into two main approaches: forward and backward selection. In the first case, the algorithm starts with an empty subset of features to which, at each iteration, the feature that has the greatest influence on the quality of the model is added. In the second case, the initial subset contains all the attributes of the training sample, from which the least significant attributes are removed at each iteration. In both cases, the process continues until a statistically significant improvement in the quality of the model is obtained (the stopping criterion is reached). Wrapper methods use a greedy search approach to evaluate all possible combinations of features using some evaluation criterion (e.g., p-value and determination coefficient (R^2) for regression; accuracy, precision, recall or F-score for classification), thus having a rather high computational complexity. Another problem with this approach is that the backward selection method cannot be used when the number of features exceeds the number of observations in the training sample.

Embedded methods combine the advantages of filters and wrappers, integrating feature selection into the learning process. The most common embedded methods are based on tree-based algorithms. At each recursion step, some feature is selected, and the sample is divided into smaller subsets. The more child nodes in a subset belong to the same class, the more informative the feature is considered. In classification problems, the partitioning is usually performed either according to the Gini coefficient (index) or using the information gain, which is based on the concepts of entropy and the volume of information. In regression problems, the partitioning is performed by a dispersion value. In addition to tree-based algorithms, regularization approaches are also common. The idea of regularization approaches to construct an algorithm that minimizes not only the model error but also the number of variables used. In such cases, both L1-regularization or L2-regularization and their combinations are used. These regularization methods reduce some model coefficients to zero, which allows removing such features from the model. Embedded methods allow to identify more complex dependencies in datasets and are less prone to overfitting and computationally complex than wrapper methods. Even though embedded methods are still more computationally complex than filtering methods, this class of methods is best suited for automating feature selection.

Another challenge is related to the hyperparameter optimization. Machine learning models often include hyperparameters whose values are very important for achieving high quality models [4]. The hyperparameter optimization algorithms work with the model as with a black box: only the value of the model loss function obtained by training with the considered set of hyperparameters is important, not the algorithm itself. In formalized form the problem of hyperparameter optimization can be written in the following way: let A be the model of the algorithm characterized by hyperparameters $\lambda = \{\lambda_1, \dots, \lambda_n\}, \lambda_1 \in \Lambda_1, \lambda_n \in \Lambda_n$. Then, the space of hyperparameters associated with it is $\Lambda = \Lambda_1 \cdot \Lambda_2 \cdot \dots \cdot \Lambda_n$. The goal is to find such set of hyperparameters $\lambda^* \in A$ with which the given model of algorithm A is the most efficient.

Several methods of automatic hyperparameter selection have been proposed by researchers in the field of computer science. As stated in one of the papers discussing the use of automated machine learning [8], the simplest ways to optimize hyperparameters are grid and random search. Grid and random search are uninformed methods, which means that they do not learn any information from previous iterations.

Grid search is a brute-force algorithm in which model is trained and evaluated for a complete set of hyperparameter combinations. Because of this, increase in the size of the hyperparameter search space leads to an exponential rise in computational complexity. Therefore, this algorithm is often an unsuitable choice as it could be inefficient in terms of performance.

In random search a complete set of hyperparameter optimization is replaced by a subset of a randomly chosen length. Since length of a hyperparameter set is less than in grid search, this algorithm requires less computational time, but here comes a risk that the best combination of hyperparameters would not be included in the tested set.

Recently, such method as Bayesian optimization is increasingly used for the hyperparameter optimization. Its major difference from the previously presented approaches is that it is an informed method, so the tuning algorithm optimizes the choice of parameters at each step according to the evaluation of the previous step. In summary, this method creates a probabilistic model which maps hyperparameters to their corresponding estimation probability. Instead of trying complete set or subset of hyperparameters, the Bayesian optimization method can converge to the optimal hyperparameters. Thus, the best hyperparameters can be obtained without examining the entire sample space. However, additional time is required to determine the next hyperparameters to estimate based on the results of previous iterations, so this method could be slower than random search.

2. Chosen algorithms implementations

The algorithm for preparing a dataset for further use in model training consists of four steps:

1. Imputation of missing values.
2. Encoding of categorical features.
3. Feature selection.
4. Data scaling.

Non-categorical features containing more than 50 percent of missing values are deleted. Categorical variables with more than half of the missing values are filled with a special mark. For imputing continuous variables, the k nearest neighbors (kNN) method based implementation from Scikit-learn library [9] called *KNNImputer* is used. Each sample's missing values are imputed using the mean value from 5 nearest neighbors found in the training set. The gaps in the remaining features are filled with the most popular value using *SimpleImputer* from Scikit-learn library.

The rows of the dataset are shuffled randomly before encoding categorical features, as some datasets may be sorted according to the value of the target variable, which can lead to problems when using target encoding algorithm. After random shuffling the text processing algorithm to determine whether the values of the feature are textual representations of integer numbers is applied (e.g., "seven" is converted to 7). Features containing only one unique value are removed since they have low effect on the target variable. If a feature consists of only two different values, then ordinal encoding is applied. In other cases, the target encoding algorithm from CatBoost library called *CatBoostEncoder* is applied.

The features are encoded according to the following formula:

$$\frac{targetSum + prior}{featureCount + 1} \tag{1}$$

where $targetSum$ is a sum of the target value for that particular categorical feature (before the current one), $prior$ is the constant value defined as the ratio of the sum of all values of the target variable in the dataset to the total number of observations, $featureCount$ is the total number of categorical features observed before the current one and having the same value as the current one. With this approach, the first few observations in the dataset always have the statistics of the target feature with much higher variance than the subsequent ones. To reduce this effect, many random permutations of the same data are used to calculate the statistics of the target variable, and the final encoding is calculated by averaging across these permutations.

As previously discussed, the best approach for selecting the most significant features in machine learning systems is the embedded methods, so the Scikit-learn implementation called *SelectFromModel* with *ElasticNet* estimator for regression problems and *DecisionTreeClassifier* estimator for classification problems was chosen. At the first step of the feature selection algorithm, a model based on a training sample is constructed. Then an approach based on feature importance calculation is used. Features are considered unimportant if the corresponding feature importance values are below a given threshold parameter. The threshold is calculated programmatically using the median value of importance of all features multiplied by a constant as a heuristic. At the last step of the algorithm, the features that the algorithm has marked as unsignificant are removed.

Feature scaling is based on Scikit-learn *StandardScaler* which standardize features by removing the mean and scaling to unit variance. The standard score of a sample *x* is calculated as:

$$z = \frac{x - u}{s} \tag{2}$$

where *u* is the mean of the training samples and *s* is the standard deviation of the training samples.

For hyperparameter optimization an implementation of the Bayesian optimization method from Scikit-optimize library called *BayesSearchCV* is used. The choice of Bayesian optimization method was made for reasons of reducing training time and increasing the models' quality. As mentioned earlier, grid search is not a suitable choice for automated machine learning systems because of its high computational complexity, as it can lead to an excessive load on the system when the system is used by a sufficient number of users at the same time. Random search, as stated before, may not find the best hyperparameter combination in a given number of iterations.

To prove the above statements, a couple of experiments with different number of hyperparameter combinations for random search classifier was conducted on the breast cancer dataset (https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data). The results of hyperparameter optimization on 2160 hyperparameter combinations is presented in the Table 1. Default algorithms parameters were not changed. As can be seen, the grid search is about three times slower than the Bayesian optimization and 62 times slower than the random search. However, the grid search algorithm gives the best model score, while the random search gives the worst. Therefore, to achieve a balance between computation time and model quality, *BayesSearchCV* is a suitable choice.

Table 1

Comparison of hyperparameter optimization algorithms from Scikit-learn library.

Algorithm	**F1 score**	**Elapsed time (seconds / s)**
GridSearchCV	0.971874	176.38
RandomizedSearchCV	0.959786	2.83
BayesianSearchCV	0.968036	63.71

For model evaluation the Scikit-learn implementation of Leave-One-Out cross-validation method is used. The advantage of this method is that each object of the sample participates in the control sample exactly once, and the length of the training subsamples is only one less than the length of the full sample. The main disadvantage of this method is high resource intensity, since the learning process is performed as many times as number of objects in the sample. Nevertheless, this method is the most accurate among all cross-validation methods because in most other cases (Hold-Out Validation, *k*-fold cross-validation) the training sample is divided into fewer parts.

The algorithm for preprocessing prediction data is similar to the dataset preparation before model training:

1. Imputation of missing values.
2. Removal of the features that are not presented in the training sample after preprocessing.
3. Encoding of categorical features.
4. Data scaling.

3. Web application implementation details

A web service (source code: https://github.com/sirenescx/fastml-web-application) is a system of five interrelated parts (Fig. 1):

1. Web application with a graphical user interface, which is responsible for processing user actions, data input and output, and forming and sending requests to the API.
2. A microservice (API) which processes incoming requests from a web application and distributes data according to the algorithms selected by user.
3. A microservice for data preprocessing.
4. Set of microservices with regression and classification algorithms for training and prediction.
5. A database used for storing user data and algorithms and microservices' settings.

The microservice architecture was chosen for the following reasons:

1. For implementation of a microservice for data processing and microservices for machine learning, the Python language was chosen because of significant number of tools and libraries for data analysis and machine learning. However, when it comes to creating of web services, Python is not the best choice due to its low performance compared to most other programming languages, as well as the difficulty of testing. The microservice architecture provides the ability to use different technology stacks for different tasks and allows to easily connect services written in different programming languages into a common system.

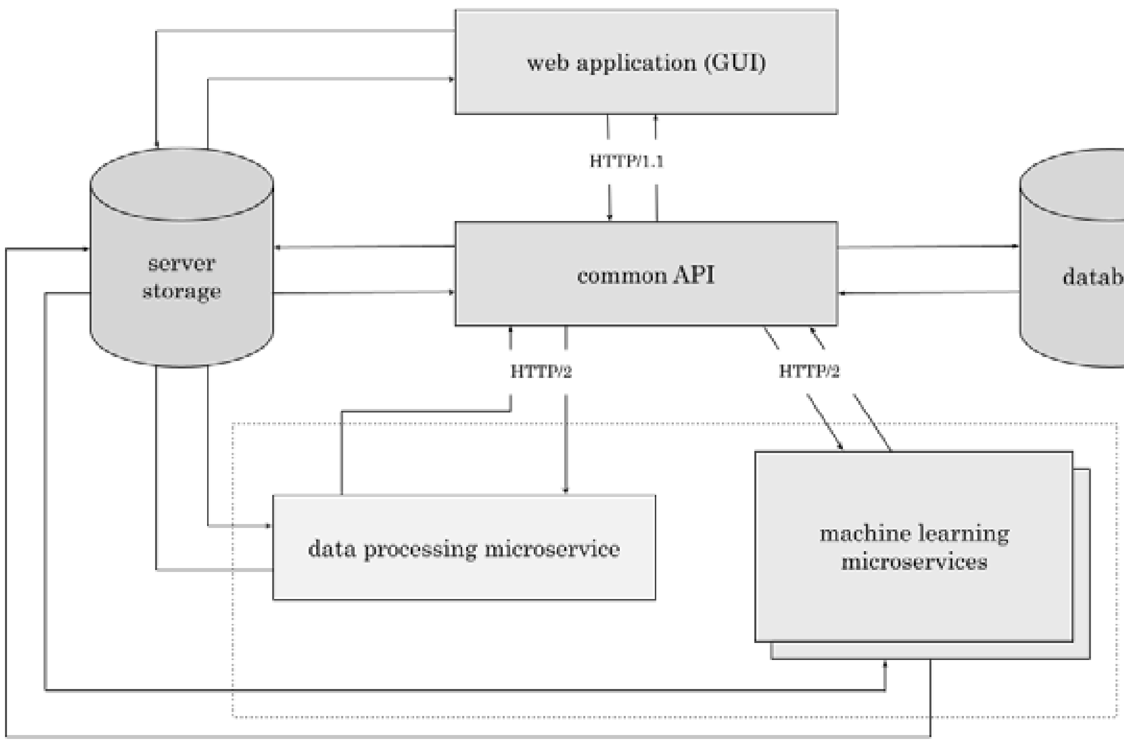

Fig. 1. Web service architecture scheme

2. The microservice architecture allows to extend the system functionality without rewriting the existing source code which makes future development easier and faster.
3. On certain datasets, usage of some machine learning algorithm implementations from libraries may lead to an infinite loop. Placing the algorithms in separate microservices prevents the entire application from crashing by using method execution timeouts. In the case when the specified time for method execution is exceeded, the task is terminated.

One of the most common tabular data formats for use in machine learning is CSV. In addition, Microsoft Excel is often used to create and edit tables, where files have the extension .xlsx. Therefore, the both XLSX and CSV file formats are supported.

The first line of the uploaded dataset should contain the names of the columns, the second and the next lines should contain the feature description of the objects (one object per line). Also, by default, it is considered that the first column of the dataset contains the object name. At the same time, the user can specify that the first column of the sample loaded also contains the feature description of the objects. The output files are in CSV format, with a comma as the separator. As with user-loaded datasets, the first line of the output file contains the column (feature) names. The column containing the result of the target variable prediction is marked as “target”.

The graphical user interface is simple and consists of a set of HTML pages that requires minimum user interaction to create, configure, and use machine learning models for prediction. The training or prediction process involves a step-by-step navigation through the several web pages of the application.

First page of training process requires dataset upload and choice of delimiter character if the data is presented as a .csv file (screenshot: https://github.com/sirenescx/DAMDID-data/blob/master/gui/upload_train.png). Next page provides an ability to choose problem type, target variable and needed machine learning algorithms (screenshot: https://github.com/sirenescx/DAMDID-data/tree/master/gui/problem_settings). On this page user is also able to set custom model parameters for one or more selected models. After training process is set up, user is redirected to a web page on which he or she can track the progress of learning process (screenshot: https://github.com/sirenescx/DAMDID-data/blob/master/gui/log.png). Once training process is complete, the user will see a table containing the values of quality metrics for each of the selected algorithms which could be sorted by algorithms names or metrics values (screenshot: https://github.com/sirenescx/DAMDID-data/blob/master/gui/choice.png). At this stage, the user is prompted to select the best models for saving and further use.

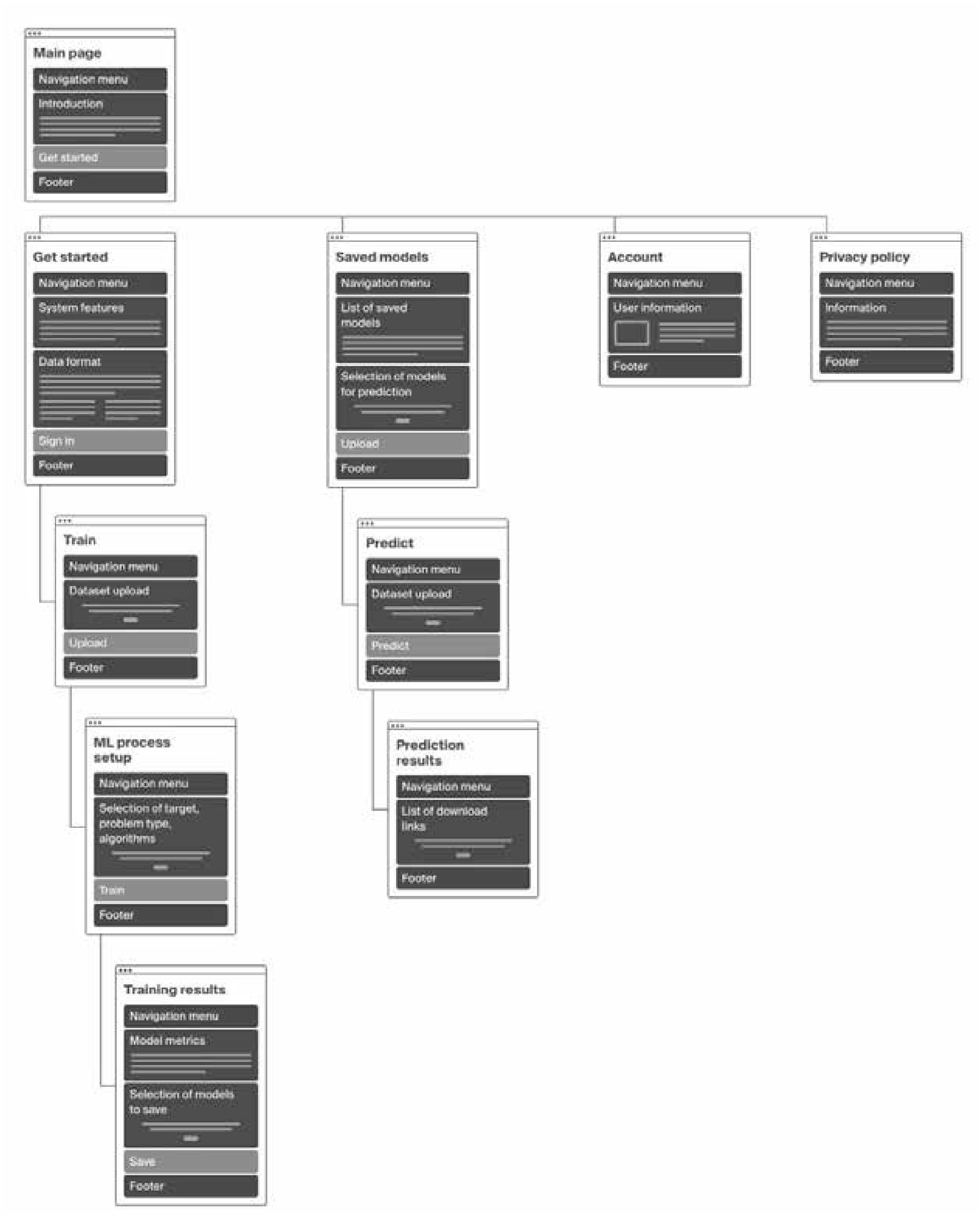

Fig. 2. Web pages navigation scheme

The interface of the prediction process is even simpler, the user just needs to select one of the saved pre-trained models (screenshot: https://github.com/sirenescx/DAMDID-data/blob/master/gui/results.png), upload a dataset (screenshot: https://github.com/sirenescx/DAMDID-data/blob/master/gui/predict.png), and wait for the prediction process to complete. Then, user can download prediction results to personal com-

Table 2

Model evaluation results

Algorithm	R^2	MSE	MAE
Ridge	0.987098625135458	0.0008175182548120166	0.01915270571705796
Lasso	0.987187519807067	0.0008118852879725164	0.021722537383973752
ElasticNet	0.986884345202036	0.0008310964787650111	0.02216231766619828

puter (screenshot: https://github.com/sirenescx/DAMDID-data/blob/master/gui/prediction_results.png). The page navigation scheme is presented below (Fig. 2).

To describe an example usage of the web application, a regression dataset containing 29 chemical objects – chalcospinels with $ABCX_4$ composition – with 108 continuous features was used for training (raw training dataset: https://github.com/sirenescx/DAMDID-data/blob/master/training_set.csv). This dataset contains data about chalcospinel compounds and their properties. The value of the target variable (crystal lattice parameter, *a,* ranges from 7.419 to 8.635 Å).

After data preprocessing one feature (E2-67) was dropped as non-informative because of a constant value of 1.8 (preprocessed training dataset: https://github.com/sirenescx/DAMDID-data/blob/master/training_set_processed.csv). Chosen algorithms set incuded three regularization methods (L2, L1 and L1/L2 regularization) and its implementations in Scikit-learn: *Ridge*, *Lasso* and *ElasticNet* models.

To evaluate the quality of the obtained models standard metrics for regression problems were used: the coefficient of determination (R^2), mean squared error (MSE) and mean absolute error (MAE). As can be noticed from the results (application output: https://github.com/sirenescx/DAMDID-data/blob/master/metrics.csv) given in Table 2, after applying hyperparameter optimization and cross-validation, all of the trained models had sufficient quality because R^2 score is close to 1. However, best algorithm is Lasso according to MSE value, and Ridge according to MAE value.

Conclusion

The popularity of machine learning is growing every year, so programs and web services for automated machine learning seem to be quite a promising area, as they make machine learning accessible not only to experts, but also to users with a basic understanding of the field. In addition, these systems can simplify and speed up development while analyzing data.

This paper presents one of the possible approaches to automate and simplify training, evaluation and obtaining predictions from machine learning algorithms for tabular datasets. The proposed approach is based on the development of the web application with graphical user interface.

Existing approaches for filling in missing data, encoding categorical variables, feature selection, and hyperparameter optimization were analyzed in this work. Chosen algorithms, methods, and implementations were also provided.

The current result of this work is a web application with GUI for the tabular data analysis which allows users to upload raw tabular dataset in one of the supported formats (.xlsx or .csv) and use Scikit-learn implementations of classification and regression machine learning algorithms to train the models or use them for making predictions on structurally identical data. Data preprocessing, hyperparameter optimization and model evaluation are done automatically by the web service. However, it is also possible for user to set custom model parameters if required.

As a next step, it is planned to add more complex solutions of value imputation, support of more regression and classification algorithms for tabular data from Scikit-learn library as well as to add such models from Keras library. Since the system can be easily extended due to the microservices architecture, it is also planned to provide users an ability to add handwritten models at a runtime.

In addition, at the time of writing this paper, the web application is being tested by real users, which allow us to collect a feedback and use it to improve the user interface and overall system performance.

References

1. *Sarker, I.H.* 2021. Machine Learning: Algorithms, real-world applications and research directions. SN Computer Science 2. Available at: https://doi.org/10.1007/s42979-021-00592-x (accessed November 15, 2021).
2. *Kumar Y., Kaur K., Singh G.* 2020. Machine learning aspects and its applications towards different research areas. 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM) Proceedings. Dubai. 150-156.
3. *Angra S., Ahuja S.* 2017. Machine learning and its applications: A review. 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC) Proceedings. Chirala. 57-60.
4. *Santu S. K. K., Hassan M. M., Smith M. J., Xu L., Zhai C., Veeramachaneni K.* 2022. AutoML to Date

and Beyond: Challenges and Opportunities. ACM Computing Surveys (CSUR) 54(8). Available at: https://doi.org/10.1145/3470918 (accessed November 17, 2022).

5. *Harrison M., eds.* 2019. Machine Learning Pocket Reference: Working with Structured Data in Python. 1st ed. Sebastopol, CA, USA: O'Reilly Media. 320 p.
6. *Micci-Barreca D.* 2001. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. SIGKDD Explorations 3(1). Available at: https://doi.org/10.1145/507533.507538 (accessed March 31, 2022).
7. *Jović A., Brkić K., Bogunović N.* 2015. A review of feature selection methods with applications. 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Opatija. 1200-1205.
8. *Waring J., Lindvall C., Umeton R.* 2020. Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. Artificial Intelligence in Medicine 104. Available at: https://doi.org/10.1016/j.artmed.2020.101822 (accessed November 21, 2021).
9. *Pedregosa F. et al.* 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12. Available at: https://doi.org/10.48550/arXiv.1201.0490 (accessed May 12, 2022).

M.S. Manakhova. National Research University Higher School of Economics, 11 Pokrovsky boul., Moscow, 109028, Russia, e-mail: mmanakhova@hse.ru

V.A. Dudarev. PhD, A.A. Baikov Institute of Metallurgy and Materials Science of RAS, 49, Leninsky pr., Moscow, 119334, Russia, e-mail: vic@imet.ac.ru (correspondent author)