Position Papers of the 18th Conference on Computer Science and Intelligence Systems pp. 103–110
DOI: 10.15439/2023F1990
ISSN 2300-5963 ACSIS, Vol. 36

Comparative Analysis of Low-Code Computation Systems

Anna Rosłan
Warsaw University of Technology
Warsaw, Poland

Michał Śmiałek
Warsaw University of Technology
Warsaw, Poland
0000-0001-6170-443X

***Abstract*—The paper aims to systematically compare computation platforms where the development of custom computation applications is done visually. By this, we mean platforms equipped with a visual language to define the flow of actions or data, thus allowing us to treat them as low-code systems. The chosen platforms include two mature systems: Orange and Azure Machine Learning Studio, and also a newcomer – BalticLSC. For the purpose of the study, two sample computing tasks were created and executed on the three platforms. Based on this, the platforms were compared with each other taking into account the following characteristics: versatility, scalability, user entry barrier, cost of use, availability of documentation, maintainability and extensibility, availability, security, user interface friendliness, and variety of interfaces for input data.**

***Index Terms*—BalticLSC, cloud computing, Orange, Azure, Data Mining, low-code**

I. Introduction

THE NEED to use low-code platforms comes from the need for companies and enterprises to develop digital tools easily, cheaply and quickly [13]. Currently, digital tools are developed by teams of developers. Team members know programming, programming languages and algorithmics. The development team creates the tool based on the information provided by people who know the field of business requirements and the specifics of the domain in which the tool will be used. Growing demand for the creation and development of digital solutions is creating the need for a large number of programming specialists. Still, there needs to be more of these specialists to meet the growing demand fully. To counteract these shortages and meet the increasing demand for IT solutions, emerging low-code systems aim to develop software in a simple way that allows less skilled workers (in terms of programming skills) to participate in software development tasks related to software development. The paper indicates that the number of publications on low-code systems has increased in recent years, proving the interest in low-code platforms in the commercial market and among the scientific and academic community. The scientific articles surveyed demonstrate that the field of low-code is still being developed and researched and will continue to gain importance.

One such solution is the BalticLSC platform [16], which is a result of a research project to apply the low-code approach to large-scale computations. The claim of BalticLSC is that it should be simple to use and easy to understand by scientists, researchers and other people not proficient in programming. In this paper, we aim at verifying this claim by comparing BalticLSC with two mature platforms that use similar means for defining computation apps – the Orange platform and the Azure Machine Learning Studio. These platforms provide the functionality to create customized computing, data flow and data analysis applications. The platforms also have in common the provision of graphical editors. Additionally, access to these platforms for private use was easy, unlimited and free (except for Azure).

For this purpose, we use a software quality model to compare the features of these three platforms.

II. Related work

The topic of software quality research has been studied many times [17]. There are many software quality models like ISO 9126, McCall's model, Boehm's model, etc. Most models consider the internal perspective taking into account the software development process. There are also approaches which emphasize the external perspective, i.e. user satisfaction [4], [7], [17]. The proposed quality models consider functional and non-functional software requirements and focus on the user perspective and expectations. Examples of software quality metrics from the user perspective include functionality, usability, reliability, performance, security or support.

Research papers have used different approaches to comparative analysis of applications. Most of them focus on the technical parameters of the systems and the features provided [5], [6], [12]. The analyzed features include the service delivery model (IaaS, PaaS, SaaS), architecture, type of input supported, network configuration, and security-related issues. Another set of benchmarks can include various specific functions provided by the compared tools [8], [14]. This might include, for example, the types of machine learning models available, Decision Trees, neural networks, the ability to load input data in different formats, and types of data visualization (histograms, graphs). Still, such approaches to application analysis provide information about the functions and features of the systems, but provide no data allowing for explicit comparison between tools.

Another comparative application analysis approach is based on comparing features related to user experience and ease of management with each other. Maiya et al. [11] have selected system features such as the learning time required to complete

Thematic track: Practical Aspects of and Solutions for Software Engineering

a task, the number of steps taken, ease of use, and the availability of documentation. Each feature was assigned a qualitative measure (e.g., easy, complex) or quantitative measure (10 steps of execution, 10 min). These measures were normalized to a scale of 1-5. This approach to benchmarking allows for a comparative evaluation of systems. Another approach [10] focuses on comparing the computing power of performance platforms and the price of executing an assumed computational problem on selected platforms.

The comparative analysis can also involve solving a selected problem (like heart attack recognition) [9], [15] using the same kind (e.g. machine learning) model on different platforms. Measures of the quality of the services offered are then determined. In the case of using machine learning models, this is, for example, precision and sensitivity. This type of analysis provides information regarding which tool to choose to solve a particular problem (e.g. medical image classification).

III. BalticLSC platform characteristics

BalticLSC [16] is a platform for developing or using large-scale computing applications. The system is designed to provide access to large-scale computing resources to small businesses and institutions that often do not have the resources to buy and maintain the desired infrastructure, as well as the ability to make unused computing resources available to companies and institutions in exchange for financial benefits. The idea is that the system should be easy to use, affordable, and efficient. The user does not need to have specialized knowledge, as by using a graphical language, the users, through the user interface, can design their own applications. The system offers a platform for using ready-made algorithms and applications and creating and sharing one's own designs.

Currently, the system is made available to users in a demo version. This means that some of the functionality has yet to be made available to users. In this paper, the platform will be described in its full version taking into account the functions not available to users to explore the system's full potential. Also, computing centres have not been made available in the demo version. There is one centre located at the Warsaw University of Technology, so the research does not take into account performance tests and speed of execution of test computing programs. The platform was developed as part of a research project co-financed by the European Regional Development Fund. One of the authors of the paper, Michał Śmiałek, is one of the developers of BalticLSC platform, but the study was conducted by someone unfamiliar with the system.

IV. Azure ML Studio characteristics

Azure Machine Learning Studio [2] is a tool that enables one to create, train and deploy machine learning models in the Microsoft Azure cloud. It is a fully managed service that enables machine learning across multiple platforms, including R and Python. Azure Machine Learning Studio allows users to create and deploy machine learning models without programming or machine learning knowledge. Users can import data, perform data mining, produce models, and share results through the user interface. Azure Machine Learning Studio offers many ready-made machine-learning algorithms and allows users to create their own models. Users can also use existing models and customize them to suit their needs.

V. Orange characteristics

Orange [3] is an open-source data mining and business analytics platform that allows users to create and visualise machine learning, neural network, regression, and classification models. It is a drag-and-drop tool, which means users can easily create and modify machine learning models by dragging and dropping feature blocks. The system is provided as a desktop application. The application uses the computing power of the device on which the application is used. Orange includes many built-in machine-learning algorithms and tools for data visualization and presentations of descriptive statistics. The platform also offers many add-ons and plug-ins that allow users to customize the tool to fit their needs.

VI. Methodology for comparative analysis

In order to perform a comparative analysis, two problems were formulated and then solved by the author of this paper on the selected platforms. Following this, the solutions to the problems were compared, taking into account the following features of the systems:

- **Universality**: the openness of the platform to define custom flows and applications.
- **Scalability**: the ability to perform selected tasks on augmented data.
- **User entry barrier**: the number of technologies and tools a user needs to know to use a given platform.
- **Cost of use**: the potential costs associated with performing tasks on the platform.
- **Documentation availability**: the amount and quality of available documentation and information on how to use a given platform.
- **Ease of maintenance and extensibility**: the features provided by the system that enable and facilitate changes, bug fixes, extensions, and enhancements to the application.
- **Availability**: trouble-free operation of the service,
- **Security**: the quantity and quality of the data protection mechanisms used.
- **User interface friendliness**: the ease and clarity of the user interface. The number of steps a user must take to perform a given task.
- **Variety of interfaces for input data**: the ability to provide input data of different formats.

The selection of features for benchmarking was modeled on the software quality models presented in Section II The features selected for comparative analysis reflect an external perspective, i.e. user satisfaction. Another important aspect was to select quality features appropriate to the chosen domain of systems, i.e. low-code platforms.

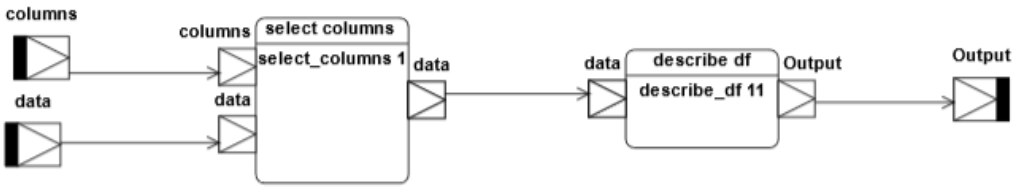

Figure 1. Solution of the first problem in the BalticLSC system

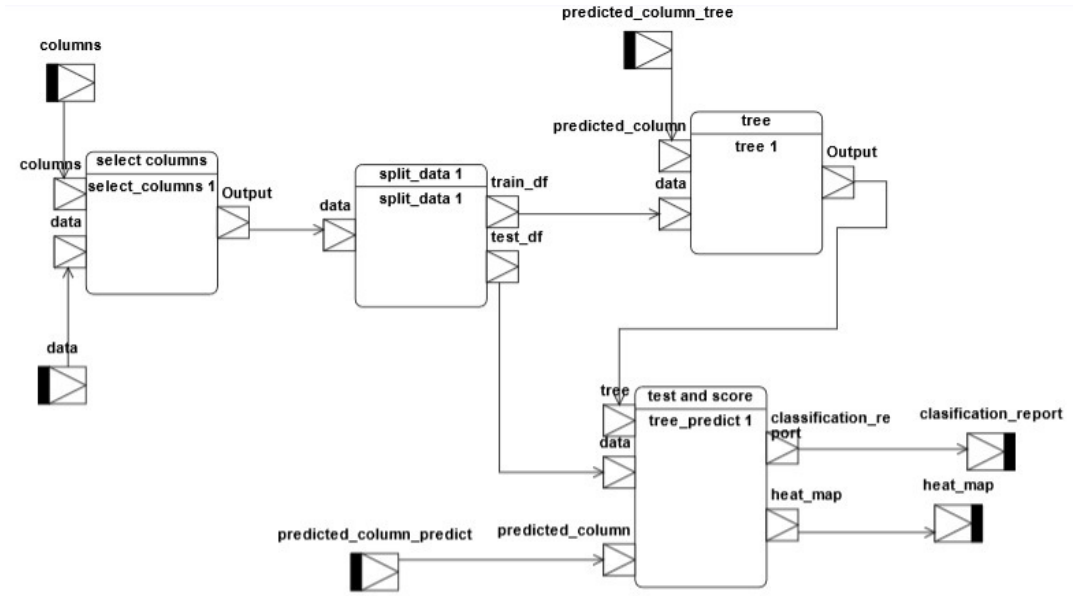

Figure 2. Solution of the second problem in the BalticLSC system (part one)

The first problem to be solved is static analysis of the provided data. We need to create a simple script in Python or another available language that analyzes the provided data. The result of the program is a CSV file that contains the descriptive statistics of the provided data. The second task is to create a decision tree on input data. We divide the data into a test set and a learning set, then create a decision tree and test the model's performance on the test data. The results of the program are files showing the effect of the model, e.g. tree graphics, and analysis of the model prediction, e.g. confusion matrix, and measures of prediction accuracy.

The first task is to check the selected platforms whether they enable the execution of user-defined actions, which checks platform flexibility and versatility. The creation of descriptive statistics action is used as an example of an action that the user must prepare himself. The execution of the task is designed to test the platform's ability to define custom flows freely

Task two is designed to test the execution of an example workflow related to machine learning. Task two contains steps that are typical of an area related to machine learning, i.e. data processing, data partitioning, training the model, and testing the model. Evaluation of the system consists of a comparative analysis of selected platforms for the presented set of features. As a result of the comparative analysis, systems are ranked by awarding first, second and third place.

VII. Implementation of low-code applications

A. BalticLSC

The BalticLSC system offers integration with technologies that store data, but it does not have data storage facilities. It was decided to prepare the input data as a CSV file available via an FTP server. The computation results would also be stored on this server. In addition, BalticLSC has no functionality for configuring computation modules through a GUI. Instead, configuration parameters such as what columns to choose for analysis have to be provided as additional input. This allows to select different parameter sets simply by changing the address of the file with the parameters.

Figure 1 shows the solution to the first problem. On the left are graphical representations of the input data, called the data pins. The user can configure input data on the platform by specifying the address (e.g. a URL and a file name) through which the platform obtains the data. In our case, data was provided through an FTP server. The data pin called "data"

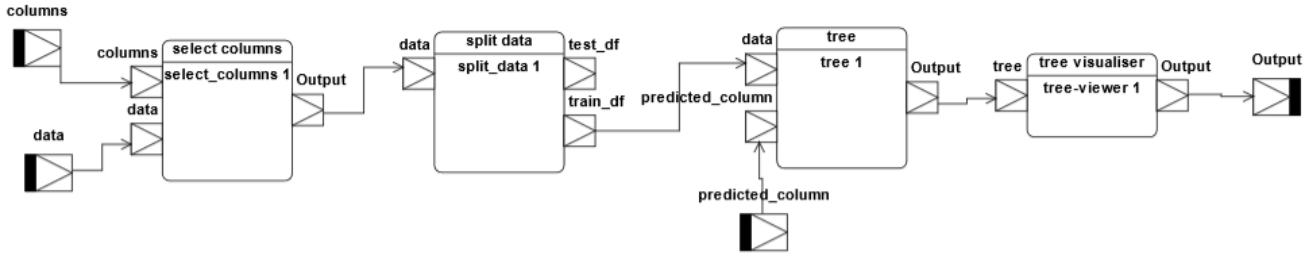

Figure 3. Solution of the second problem in the BalticLSC system (part two)

represents a CSV file with the data to be analysed. The data pin called “columns” represents the list of columns to be analyzed. These two data elements are input to the first module, called “select_columns”. It transforms the input dataset using the selected columns and returns it as the module’s output. The output effect of the module is passed to the next computation module, named ”describe_df”. This module creates descriptive statistics for the transformed data organised into columns. The effect of these computations is passed to the output data pin (“Output”). As for the input data pins, this pin is configured so that the data can be saved to a specific address, which in this case is on an FTP server.

While designing the solution to the second problem, we have noticed that it was not possible to use data coming out of a component iteratively, so two applications had to be created. The first one (Figure 2) accepts the same kinds of data as for the first problem (“data” and “columns”). They are used to create the calculation model. The application has two additional input data pins - “predicted_column_tree” and “predicted_column_predict”. They describe what column from the main input data set (“data”) will be predicted in the machine model. We need two input data pins because a data pin cannot be used more than once. Execution of the application starts with selecting columns for further processing using the “select_columns” module. This results in selecting columns from the main data set to be used for further processing. The result is passed to the “split_data” module, which randomly splits the data into a training and a test set which are output to the “train_df” and “test_df” pins. The training set is passed along with the information about column predictions to the “tree” module. This module creates a decision tree, where the result is a plotted model. This model is passed to the final “test and score” module. Apart from the decision tree, this module receives two inputs – the test dataset “test_df” and information about the predicted column “predicted_column_predict”. The module tests the created model and returns a report (“classification_report”) and a confusion matrix as a heat map (“heat_map”). These two outputs from this application are then stored in the same way as in the first application.

The second application is shown in Figure 3 and is prepared similarly to that from Figure 2. The main difference is the last module – “tree visualizer”. It receives a trained decision tree model as input and creates its graphical representation. The resulting visualisation (diagram) is passed to the output and is stored on an FTP server, as in the other applications.

B. Orange

As in BalticLSC, the execution of tasks in Orange is done within the environment. However, computations are done locally, so there is no need for external storage. In our case, CSV files containing input data were imported directly into the system. The resulting data was accessible directly through a graphical interface.

The application solving the first problem in the Orange notation is shown in Figure 4. Its execution begins with loading the CSV file with data into the “CSV File Import” module. Then, the data is passed to the “Select Columns” module, which selects data contained in selected columns. Unlike for BalticLSC, the columns are not input as a separate file but are defined as direct parameters of the “Select Columns” module. This is done through a GUI and needs to be changed for different computations. After selecting columns, the application runs the “Python script” module to execute a dedicated Python script that creates descriptive statistics for the provided data. The script is defined within the module details. The result of these computations is a data set that is saved to the local machine using the “Save Data” module. The location of the file is given in the details of the module. In addition, it is possible to display data directly in the application using the module “Data Table”. It can be done by showing the module details.

To solve the second problem (see Figure 5), we use the “CSV File Import” and “Selected Columns” modules described above. Data from selected columns is passed to the “Data Sampler” module, which randomly separates the data into a test set and a training set. The training set is passed to the “Tree” module, which creates a decision tree. The trained model is then passed to the “Tree Viewer” module, which creates a graphical representation of the tree. The “Test and Score” module uses the test data to test the tree and returns test information. This information is used by the “Confusion Matrix” module, which creates a confusion matrix as its name suggests. Data passed and produced by the “Tree Viewer”,

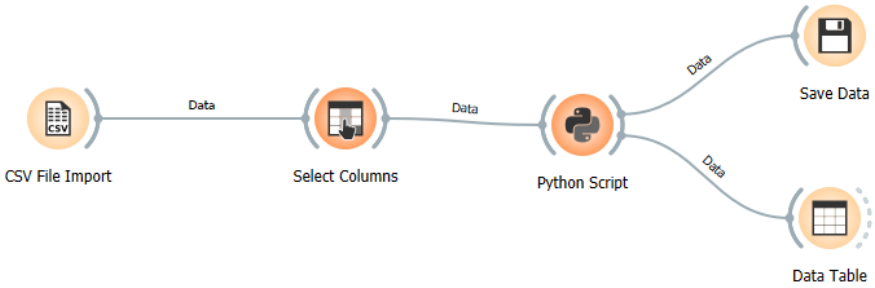

Figure 4. Solution of the first problem in the Orange system

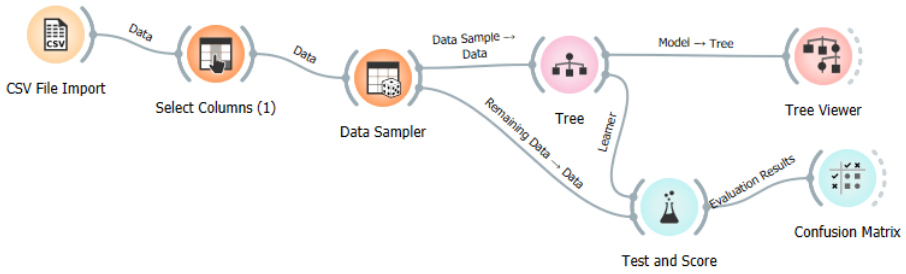

Figure 5. Solution of the second problem in the Orange system

"Confusion Matrix", and "Test and Score" modules can be viewed in the module details. There, one can also specify additional parameters, for example, the predicted column in the "Tree" module.

C. Azure ML Studio

The Azure ML Studio platform operates in the cloud, similar to BalticLSC. We need to create and configure an appropriate service and then design and execute a dedicated solution application. Input data was imported through a GUI into the Azure Blob Storage service, integrated with the platform. As previously, data was contained in CSV files. The output data was visible through the platform's GUI. It was also saved in the Azure data storage services.

The solution to the first problem (Figure 6) contains the input data "household_production" module connected with the "Select columns in Dataset" module. As in the previous cases, this module creates a dataset for selected columns. As in Orange, the list of columns is defined in the details of the module. The resulting data is passed to the "Python Script" module, which executes a script that creates descriptive statistics for the provided data. The script is defined using a graphical interface for editing the module details. The created statistics are passed to the "Export Data" module, which saves the data in a database integrated with the service.

The solution to the second task (Figure 7) starts with the same two modules as in Figure 6. The result is passed to the "Split Data" module, which randomly splits the data into training and test sets. The "Two-Class Boosted Decision Tree" module contains an untrained model, which, together with the training data, is passed to the "Train Model" module. This module trains the input machine learning model on the provided data. The "Score Model" module uses the trained model and the test data to test the tree's predictions. The "Evaluate Model" module calculates various metrics, e.g. precision sensitivity and confusion matrix for the tested model. Finally, these results are stored in a database integrated into the platform.

VIII. Comparative Evaluation of the Systems

Universality. In Azure ML Studio, users only can use the platform's components. The Orange system allows to use the platform's components or create own component, and allows to install additional packages provided by the developers. Both of these systems provide machine learning and artificial intelligence functions. The BalticLSC system can create custom components and applications from any domain and allows to publish the created components. In addition, extensions can be created in any programming language. For this reason, the BalticLSC platform was identified as the most versatile among the respondents. The Azure ML Studio platform fared the worst.

Scalability. The Orange system uses the machine's computing power on which the program is run. The machine's computing power should match the computing power needed to perform the task. Executing a computationally complex task may require adjusting the physical infrastructure so that the available computing power matches the required power needed

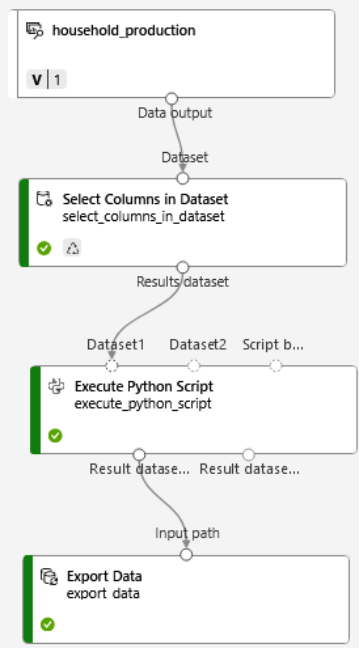

Figure 6. Solution of task one on the Azure ML Studio

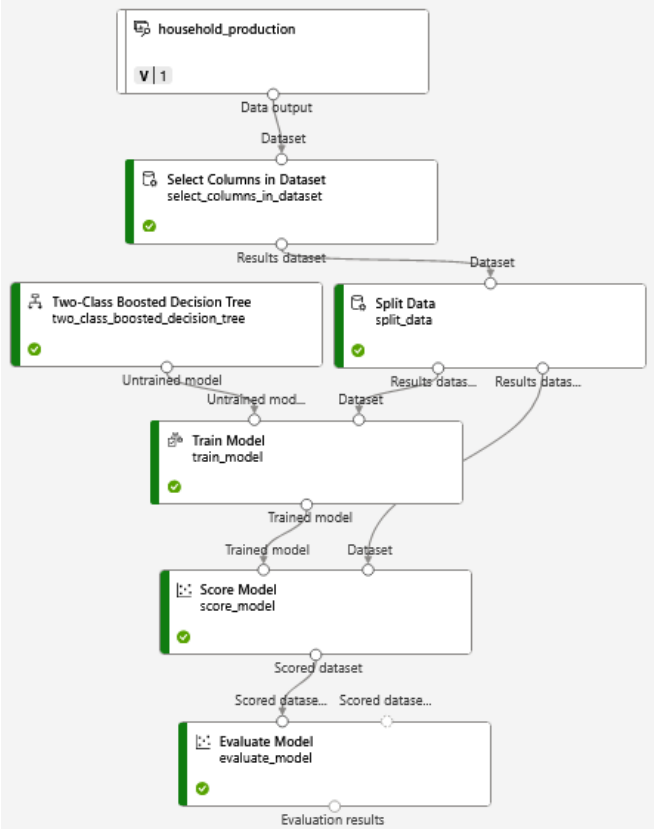

Figure 7. Solution of task two on the Azure ML Studio

to perform the task. The BalticLSC system and Azure ML Studio offer virtual use of computing power through cloud computing. The cloud is easier and generates less cost, so it is tidier. In BalticLSC, choosing the cluster on which to perform the commissioned task is possible. It is also possible to define a range of parameters related to the use of resources, such as GPU, CPU and memory. Since the BalticLSC system is in the demo version, only one cluster is available, and testing the above-described functionalities is impossible. Moreover, BalticLSC offers functions related to the parallelization of computing operations. It is possible to create applications to perform some tasks in parallel.

Azure ML studio also provides a choice of a cluster for processing the task. The computing power is automatically adjusted to the task and can be increased if necessary. The user defines the necessary parameters, such as the maximum/minimum number of clusters and location size and the scaling process itself is performed automatically. The user does not influence the process of task scaling and job scaling, and he only defines specific parameters and limits. Azure ML Studio offers the most features related to the ability to perform compute-intensive tasks. The Orange platform performed the worst.

User entry barrier. To perform tasks on selected platforms, users should have a basic knowledge of machine learning models. In addition, the user should know the Python programming language to create a custom script. To perform tasks in the Orange system, the user must be familiar with the system's components. All interaction with the components, their use and parameterization are done through a graphical interface. In order to create a script in Python, one must become familiar with the "Orange Data Mining Library". In the system, it is possible to import data from a database; thus, basic knowledge of database systems is needed. The amount of time spent learning required to complete the task is the least among the selected applications, so this platform was chosen as the best.

Using the BalticLSC Platform requires knowledge of the components offered by the system. To run the applications, users can configure a computing cluster requiring basic physical infrastructure knowledge. To import data into the application, it is necessary to create a dataset and define the type of access to it. Depending on the chosen technology, basic knowledge of database systems and FTP servers, Azure-DataLake platform, Amazon Simple Storage Service platform is needed. To create a component for BalticLSC (Python script), you need to make it using the Docker tool. For this, users need knowledge of this tool and basic knowledge of containerization. In addition, knowledge of the BalticLSC library is needed to create the code. The variety of technologies and tools for using the system is the greatest. Most of the time was spent learning about the platform and additional tools and technologies, so this platform was rated the lowest.

In order to use the Azure ML Studio platform, one should know about the services offered by the Azure platform. Azure ML Studio requires creating, configuring, and maintaining several services available on the platform. Each of these services

is non-trivial, requiring many parameters to be defined. Azure ML Studio offers a lot of features and capabilities. In order to use this tool, users should have spent the most time learning about the platform, but there was no need to be familiar with other tools and technologies. The learning time for the new tools was estimated based on the time difference between the time it took to complete the entire task and the time it took to create and run the diagram.

Cost of use. Orange ML Studio is a free platform. The BalticLSC system in the demo version does not charge fees, but the full version will require them. There is a fee for using the Azure platform. You pay for using computing power, i.e., running the designed applications. The exact costs depend on the configuration of the computing cluster. Considering this, the Orange platform is the best in terms of cost of use, and the worst is Azure ML Studio.

Documentation availability. The BalticLSC platform has the least extensive documentation. The user can obtain information from the website https://www.balticlsc.eu. The documentation is in pdf form and is 19 pages long. On the YouTube platform, three videos discuss the functions of the system and a video with an example of how to use the platform. The documentation describes the system's architecture, available features and the interface for developing applications. In terms of availability of documentation and community, this platform is rated the worst.

The Orange platform offers an overview of the system through a website at https://orangedatamining.com/. In addition, there are 63 videos provided on the YouTube platform describing the components or showing an example use of the platform. On the website, in addition to a description of features, there are examples of how the system is used.

The Azure platform provides information about Azure ML studio through the website https://learn.microsoft.com/en-us/azure/machine-learning/. The documentation includes a description of the platform's features as well as instructions for creating your environment. It is possible to download extensive documentation in pdf form (2962 pages). In addition, there are many videos about the system on the YouTube platform created by the creators as well as others. Access to the documentation has been rated as the best for Azure ML Studio.

Ease of maintenance and extensibility. All of the platforms studied provide application expansion through a graphical interface. Adding more components is done by selecting a component and dragging it with the mouse to the application development view. The Orange system has no features to support making changes, extending and enhancing applications, so this system was rated the worst. In the BalticLSC system, components and applications are versioned. Operating on versions makes the process of maintaining applications easier. The Azure ML studio platform provides features that double down on sharing components and pipelines with other users, scheduling and automatically running pipelines. In addition, Azure offers features related to model monitoring and automatic model training. This system was rated the best in terms of ease of project maintenance.

Availability. The Azure platform, which includes the Azure ML studio under study, provides 99.99% availability to Virtual Machines. Azure implements many practices including inter-center data redundancy, backup. This platform is rated the best in terms of service availability.

The BalticLSC platform provides the possibility to select a computing cluster, which makes it possible to operate on the remaining clusters in case of failure of one cluster. The system does not store user data on its own. Data is stored in external services, which means there is no risk of data loss in case of system failure.

The Orange platform is entirely dependent on the efficiency and availability of the physical infrastructure on which it is run, which is why it is rated as the worst in terms of availability.

Security. The Orange system has no security-related mechanisms implemented. The user should secure the physical infrastructure on which the program is run. Therefore, this system is rated as the worst in terms of security.

The BalticLSC system is designed in such a way that it does not store user-supplied data or output data from the operation of the program. In addition, the use of the Docker tool allows full separation of the executed program from other elements of the system. The system uses an encrypted https protocol. The system does not provide data on how the computing clusters are protected from unauthorized access or attacks.

ML Studio provides the ability to define and manage user permissions, which increases security. Azure uses an encrypted https protocol. Application developers follow many practices [1] to ensure the security of the physical infrastructure against unauthorized access or attacks. The Azure system has been rated as the best in terms of security.

User interface friendliness. The Orange platform provides the most straightforward user interface. The user can perform a task on a single view of the application entirely through the GUI. The least number of steps are taken to perform the designed tasks.

A more complex system is the BalticLSC system. To perform a task, the user has to use four separate views to define input data, select (or design) applications, launch applications, and track the applications' progress.

The most complex system in terms of the user interface is the Azure ML Studio platform. Setting up the platform requires creating several services and configuring platform parameters. Performing the task requires using four views of the application, providing input data, creating an experiment, designing the application, running the application. The complicated user interface is due to the large number of features offered by the platform as well as the ability to customize the platform.

Variety of interfaces for input data. Orange's system allows for data transfer in three ways:

- by uploading data from the system or network address,
- downloading data from a relational database,
- self-creating data via the application interface.

Table I
PLATFORM COMPARISON SUMMARY

	Azure ML	Orange	BalticLSC
Universality	III	II	I
Scalability	I	III	II
User entry barrier	II	I	III
Cost of use	III	I	II
Documentation availability	I	II	III
Ease of maintenance	I	III	II
Availability	I	III	II
Security	I	III	II
User friendly interface	III	I	II
variety of interfaces for input data	II	III	I

BalticLSC allows data transfer in five different ways:

- by cloud services (Amazon S3, Azure Data Lake),
- FTP server,
- uploading data from the system,
- downloading data from relational databases,
- downloading data from NoSQL databases.

The Azure ML Studio platform allows data transfer in four ways:

- uploading data from the system directly into the application via the GUI or command line.
- downloading data from relational databases,
- via cloud services (Amazon S3, Azure Data Lake).

BalticLSC offers the most variety of data delivery options, and Orange the least.

IX. SUMMARY

Table I summarises the results of our analysis. We have compared two mature computation platforms with a new system stemming from a research project. It should also be noted that Azure ML Studio and Orange are dedicated to specific application domains. Their computation capabilities can be extended with simple Python scripts. When comparing them to BalticLSC, we have, in fact, reduced the capabilities of BalticLSC to handle only such simple Python procedures. However, this system allows executing containers with code of any size and complexity and written in any language.

The comparative analysis allowed us to identify each application's unique features, advantages and disadvantages. Orange stood out for its intuitive interface and ease of use. Azure ML Studio offers advanced customization and a wide range of features to help organize work, which attracts users with more advanced needs. BalticLSC excels in the universality of use and the ability to create and share custom components.

Evaluating software quality is a challenging and complex process influenced by subjective factors like user experience and expectations, personal preferences, etc. Some wants are mainly based on subjective feelings like user interface friendliness, and others are less like the cost of use. Therefore, depending on the sample group, the results of the same analysis may vary.

Having in mind these differences, it can be noted that it is not possible to isolate a platform that would be unequivocally the best. Each of the studied systems has its strengths and weaknesses that contribute to their quality which is a multidimensional concept. The studied platforms have features that are attractive to different groups of users. Azure ML Studio outperforms the other systems in documentation, ease of maintenance, availability and security. However, Orange and BalticLSC dominate in such criteria as cost-effectiveness, learnability and universality.

REFERENCES

[1] Azure facilities, premises, and physical security. https://learn.microsoft.com/en-us/azure/security/fundamentals/physical-security. Accessed: 2023-03-10.

[2] Azure Machine Learning documentation. https://learn.microsoft.com/en-us/azure/machine-learning. Accessed: 2023-03-10.

[3] Orange data mining documentation. https://orangedatamining.com/docs. Accessed: 2023-03-10.

[4] Anas Bassam Al-Badareen, Mohd Hasan Selamat, Marzanah A Jabar, Jamilah Din, Sherzod Turaev, and S Malaysia. Users' perspective of software quality. In *The 10th WSEAS international conference on software engineering, parallel and distributed systems (SEPADS 2011)*, pages 84–89. World Scientific and Engineering Academy and Society (WSEAS) Cambridge, 2011.

[5] Meenakshi Bist, Manoj Wariya, and Amit Agarwal. Comparing delta, open stack and xen cloud platforms: A survey on open source iaas. In *2013 3rd IEEE International Advance Computing Conference (IACC)*, pages 96–100, 2013.

[6] C. Höfer and G. Karagiannis. Cloud computing services: Taxonomy and comparison. *Journal of Internet Services and Applications*, 2:81–94, 01 2010.

[7] Amna Ikram, Isma Masood, Tahira Sarfraz, and Tehmina Amjad. A review on models for software quality enhancement from user's perspective.

[8] A. Jovic, K. Brkic, and N. Bogunovic. An overview of free software tools for general data mining. In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1112–1117, 2014.

[9] Sarangam Kodati and R Vivekanandam. Analysis of heart disease using in data mining tools orange and weka. *Global journal of computer science and technology*, Feb 2018.

[10] Charlotte Kotas, Thomas Naughton, and Neena Imam. A comparison of amazon web services and microsoft azure cloud platforms for high performance computing. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–4, 2018.

[11] Madhavi Maiya, Sai Dasari, Ravi Yadav, Sandhya Shivaprasad, and Dejan Milojicic. Quantifying manageability of cloud platforms. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 993–995, 2012.

[12] Junjie Peng, Xuejun Zhang, Zhou Lei, Bofeng Zhang, Wu Zhang, and Qing Li. Comparison of several cloud computing platforms. In *2009 Second International Symposium on Information Science and Engineering*, pages 23–27, 2009.

[13] Niculin Prinz, Christopher Rentrop, and Melanie Huber. Low-code development platforms-a literature review. In *AMCIS*, 2021.

[14] Venkateswarlu Pynam, R Spanadna, and Kolli Srikanth. An extensive study of data analysis tools (Rapid Miner, Weka, R Tool, Knime, Orange). *International Journal of Computer Science and Engineering*, 5:4–11, 09 2018.

[15] Ritu Ratra and Preeti Gulia. Experimental evaluation of open source data mining tools (weka and orange). *International Journal of Engineering Trends and Technology*, 68(8):30–35, 2020.

[16] Radosław Roszczyk, Marek Wdowiak, Michał Śmiałek, Kamil Rybiński, and Krzysztof Marek. Balticlsc: A low-code hpc platform for small and medium research teams. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–4, 2021.

[17] Jagannath Singh and Nigussu Bitew Kassie. User's perspective of software quality. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1958–1963, 2018.