DataScience4NP - A Data Science Service for Non-Programmers

Bruno Leonel Lopes, Artur Pedroso, Jaime Correia, Filipe Araujo, Jorge Cardoso, and Rui Pedro Paiva

CISUC, Dept. of Informatics Engineering, University of Coimbra, Portugal
{bllopes, apedroso}@student.dei.uc.pt
{jaimec, filipius, jcardoso, ruipedro}@dei.uc.pt

Abstract. With the emergence of Big Data, the scarcity of data scientists to analyse all the data being produced in different domains became evident. Moreover, the processing of such amounts of data also is challenging due to current technologies in use. With this in mind, the DataScience4NP aims to explore the use of visual programming paradigms to enable non-programmers to be part of the data science workforce at a faster pace and at the same time to provide a scalable data science service. By observing the common process employed by data scientists in the extraction of knowledge from data, which includes data insertion, preprocessing, transformation, data mining and interpretation/evaluation of results, we envisioned a system to perform all these steps without requiring users to program. Thus, our solution aims to provide an intuitive user interface where users can build personalized sequential data science workflows that are consequently processed by a back-end service. The back-end service translates the received workflows to a lower-level representation, enabling the execution of the translated tasks by separate scalable and distributed data science services in parallel. The entire system is composed of different services containerized with Docker and orchestrated with Kubernetes, allowing it to be easily deployed in different clusters. To evaluate our tool, and particularly to verify if the concept we envisioned for the creation and execution of data science tasks was intuitive, we conducted preliminary usability tests with two different groups of people, where we observed a high level of user satisfaction. Concluding, from the feedback obtained, it was clear that this concept of sequential workflows would bring added value to both novice and advanced data scientists.

Keywords: Data Science · Distributed systems · Cloud computing.

1 Introduction

Nowadays, large amounts of data are being produced from multiple sources. However, not all these data can be analysed and some value might be consequently lost. One particular challenge to performing data analysis is the lack of data scientists, a resource in high demand these days [1–3]. Data scientists are

a fundamental human resource for the extraction of knowledge from data due to their data analysis and model creation skills. To overcome this issue more data scientists need to be trained, which will take time due to the diversity of knowledge areas that must be taught, where computer science is included [4]. Thus, by reducing computer science topics from the data scientists curriculum, and providing means to create models without requiring users to use programming languages, we can reduce the overall time required to train the new data scientists. As such, we envisioned a software-as-a-service (SAAS) for data scientists where it is possible to perform data mining experiments without requiring programming skills from the users.

By visualizing the knowledge discovery process reported in [5], which is used by many data scientists, we created a system that enables the application of this process by allowing the construction of data science workflows composed by sequential tasks that go from data insertion to interpretation/evaluation of results. We enforce good practices of data mining, such as evaluation of models using cross validation[17], nested cross validation, hold-out and train-validation-test methods. We also enable the creation of multiple parallel models using different parameters and features to select in the end the model that provides the best results. All this functionalities are available from a browser, without requiring users to install new software. The system follows a microservices architecture and was already deployed on a kubernetes cluster to be tested.

To evaluate the acceptance of the concept provided in this software, we conducted usability tests with a group of users familiar with data mining frameworks, obtaining results that confirm our assumptions in relation to the envisioned concept. We performed also a usability test with a group of students without experience with such software tools but with background in statistics, whom can also benefit with our software. We observed again an overall positive user satisfaction in the last case.

The remaining document is organized as follows. In Section 2, we analyse other related software tools. In Section 3, we describe the major requirements of our software, its architecture and user interface. In Section 4, we present the setup used to conduct the usability tests, and in Section 5 we present and discuss the results acquired from the usability tests. Finally, in section 6 we draw the main conclusions of this work and point out possible future research directions.

2 Related Work

The data mining process is composed of several steps. It starts with the insertion of a dataset that is processed iteratively until a desired result is obtained and in some cases the final result is a model created using a machine learning algorithm.

A classifier is one type of model that is produced by supervised machine learning algorithms. It receives typically a vector of discrete and/or continuous feature values and outputs a single discrete value, the class [6].

To assess how the classifier will behave in the presence of new data, the user must evaluate its performance, that is, its capacity to predict correct outputs

in the presence of new data. This assessment is properly performed by using data that was not employed in the training phase, otherwise the evaluation might be overly-optimistic [7]. Nonetheless it is not uncommon to see, even in published articles, evaluations done with data already seen in the training process, especially when data pre-processing/transformation precedes the use of the final machine learning algorithm. Other common situation where overly-optimistic results are verified occurs when parameter optimization is done. The issues might be overcome by employing evaluation mechanisms such as nested cross-validation [8].

Some applications that offer users the possibility to build data mining processes without programming also lack in enforcing good data mining practices to evaluate the produced models. Other applications provide correct evaluation procedures by introducing some complexity to the user while building the data science workflow. Next, we cite some of these applications.

AzureML [9], H2O.ai [10], Orange [11], Weka [12], and RapidMiner [13] are systems / applications in production that provide visual programming paradigms to help users building their models. Among these applications, AzureML is the only one publicly deployed that can be accessed from a browser. H2O.ai is not publicly deployed but can be installed in a cluster or locally and then used from a browser. Weka and Orange are standalone solutions that must be installed locally. RapidMiner is the only application among the previous ones that provides nested cross validation for parameter optimization, however it also needs to be installed locally.

ClowdFlows [14], DAMIS [16], and Zorrilla, M. and García Saiz, D. [18] are research projects. In Clowdflows and DAMIS were created cloud systems that allow users to define data science workflows in a browser using visual programming paradigms. Clowdflows assumes some previous experience with tools like Weka, Orange or Scikit-Learn [15]. In Zorrilla, M. and García Saiz, D. it was created a system following a SOA architecture that allows users to extract knowledge from data by using predefined templates. Instead of allowing users to create new models and evaluate them, the system just applies operations defined in predefined templates to a user's dataset. None of the last three projects provide nested cross validation.

All the cited applications that provide the creation of models require the user to build more complex workflows to create experiments where multiple features and parameters are tested to produce the model with best performance. In our system we will enforce the execution of this process with less complexity to the user and using good data mining practices.

3 Implementation

Having in mind the limitations identified in related applications, presented in the previous section, we focused in creating a prototype to overcome some of these issues. In this section we proceed with a more detailed description of our system.

3.1 Requirements

The identification of issues in related applications gave us the following list of main requirements to address in our solution:

- Provide an application with high usability standards for non-programmers to execute data science tasks.
- Provide different data pre-processing/transformation methods, feature selection and machine learning algorithms.
- Allow the creation of models using different features and parameters to select the best configuration of features and parameters automatically in the end. Here, good data mining practices are enforced, e.g., using nested cross validation.
- Provide access to the application without requiring users to install it in their machines.
- Parallelize data science tasks when possible to get faster results.
- Provide a scalable system to support large numbers of users.

3.2 Architecture

To satisfy the previous requirements we envisioned a cloud application available through the Internet that follows a microservices architecture and is depicted in Fig. 1.

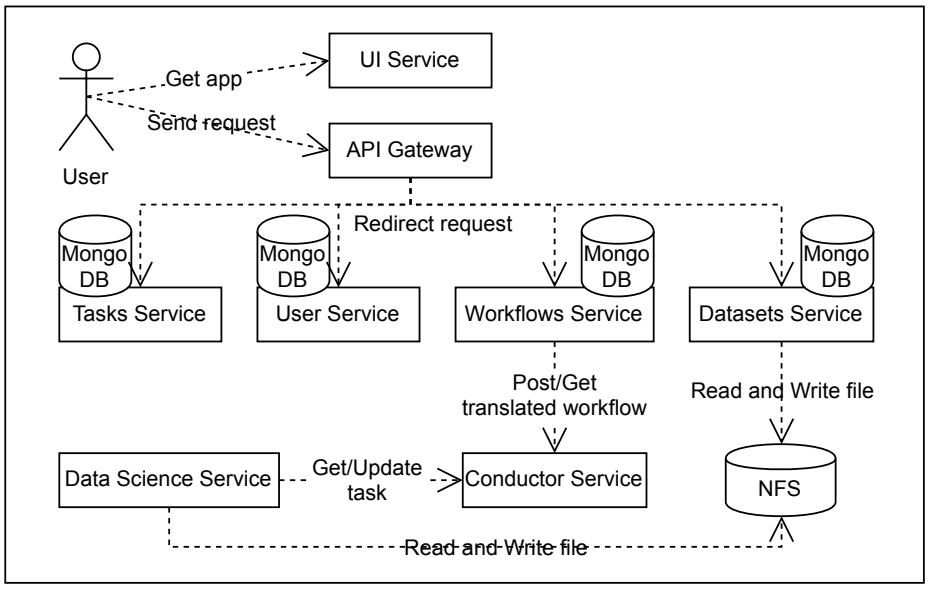

Fig. 1. System architecture.

When a user contacts our system, the first access is directed to the UI Service that provides a web application written in ReactJS, from which further requests are done to our API Gateway that redirects the requests to different services accordingly.

The Tasks Service returns the data science tasks that can be used by the user to compose a sequential data science workflow. The User Service enables users to login in the system with a username and a password and holds information related to users. The Datasets Service stores uploaded datasets in a distributed file system (an NFS server) and also returns data from the NFS according to users requests. Then, we have the Workflows Service that translates sequential workflows sent by users, which are composed of simple data science tasks, into a representation that is understandable by Netflix Conductor [19]. The new workflow representation is sent to the Conductor Service and becomes available to be processed by different Data Science services. The Workflows Service is also contacted to return the status of workflows sent by users. Finally, the Data Science Service is in reality composed of multiple fine grained services that work on specific data science tasks present in the Conductor Service. These Data Science Services share files (e.g., datasets, models) between them by writing and reading to/from the NFS.

The communications between all the services presented in the architecture are performed using the HTTP protocol, mainly through REST APIs.

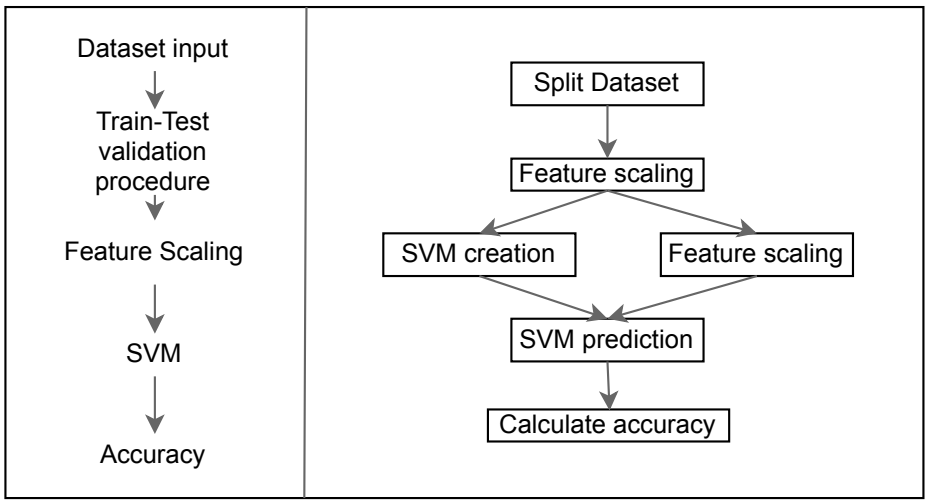

Fig. 2. Example of a data science workflow translation.

To better understand how individual data science tasks are processed in the system, in Fig. 2 we present an example of a translation from a simple sequential workflow sent by the user (on the left), to its representation in Netflix Conductor (on the right).

In the sequential workflow, the user inserts the location of the dataset to use. Then he specifies that the procedure might be evaluated using the hold out / train-test method. He also specifies that he wants to apply a feature scaling operation, followed by the creation of a model using the SVM algorithm. In the end, the user wants to check the accuracy of the model.

Upon receiving the workflow, the Workflows Service translates it to a representation that is understandable by Netflix Conductor and sends such representation to the Conductor Service. It contains a Split Dataset task (split original data in train and test sets) that is followed by a Feature scaling task (applied to train set). The Feature scaling task precedes a fork that allows the execution of an SVM creation task (applied to train set) and a Feature scaling task (applied to test set) in parallel. The SVM prediction task (applied to test set using the model created before) will only be able to execute after the previous two tasks become completed. Finally, there exists a task to compute the accuracy of the model.

By using the Netflix Conductor technology we can parallelize the tasks and orchestrate Data Science services to work in the different tasks in parallel and independently, following a competing consumers pattern [20]. The Data science services will be able to scale independently according to the type of tasks that require more workers.

The translation described before is simple, though when a user sends for example a workflow containing a cross validation task to evaluate the model or sends a workflow with different numbers of features or parameters to produce an automatic selection of the model with best parameters and features, these can be translated into more complex workflows, however with several tasks running in parallel.

With the Data Science Services working independently and in parallel in these tasks, it is expected that the required time to process the entire workflow will be reduced compared to running all the tasks sequentially.

The remaining services that compose the architecture can also be scaled out independently. The architecture also enables users to let data science workflows running, even after closing their browsers, and come back later to visualize the final results.

3.3 User Interface

The user interface was designed with the objective of creating a minimalistic and simple application where the user can produce the most value with the lowest amount of effort. The interface is divided to 2 key areas. In Fig. 3 we can visualize the interface used in the usability tests; on the left, with a dark background, we have a sidebar where the user has his saved workflows and uploaded datasets; he can also run or stop the workflow. When a user clicks the start button he is sending the workflow to the workflows-service to execute and wait for its completion. The user can also stop it at any time during its execution. The area on the right is where the user can add tasks to the workflow, which will then later be executed.

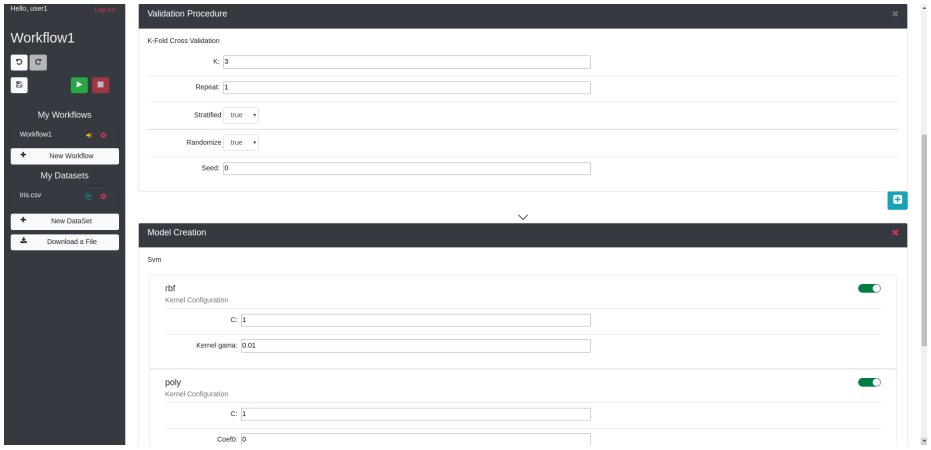

Fig. 3. Interface of the prototype used in the usability tests

When adding a task, the user is shown the types of tasks that can be added. This means that when the user clicks the plus button to add a task, depending on the current state of the workflow, he can only see the tasks that can follow and is not cluttered with all tasks at once. By doing this, the tasks are chained together, guiding the user during the construction process.

There are 6 types of tasks that the user can perform:

- **Dataset input:** a task where the user specifies the dataset to use.
- **Validation procedure:** each task specifies the method used for the validation of all other subsequent tasks. Only makes sense if the user adds a model creation type of task.
- **Preprocessing:** tasks that apply transformations to attribute values, such as feature scaling.
- **Feature selection:** tasks where the user can filter attributes based on the input parameters, such as the relieff algorithm.
- **Model creation:** is the type of tasks that creates machine learning models.
- **Model Evaluation:** specifies the metrics for model evaluation, such as accuracy or f-measure.

4 Experimental Setup

The usability tests provided a crucial role in evaluating the prototype and validating the paradigm of visual programming using sequential tasks. The tests consisted in having the users execute a few exercises using the interface and getting their feedback. This feedback was then used to evaluate the users' experience, the usability of the interface and value that was provided to them, hence validating this concept of visual programming applied to data science.

We divided the users to 2 types:

- **Type A:** Users with no experience at all and no knowledge in data mining, composed by a group of researchers, four with a masters degree in ecology and three with a doctoral degree in biology (7 users).
- **Type B:** Users that knew about data mining but were not programmers. This group was composed by students who were enrolled in a master's degree in biochemistry and were undertaking a course in data mining (11 users).

The process was separated to different steps: The first step started with a quick overview of the platform and its functionalities, which took less than 3 minutes. After this introduction and answering any questions the users might have, we gave them a paper with a problem and a list of exercises for them to perform in order to solve that problem. The exercises fundamentally consisted in using the data science tasks mentioned in section 3.3. If the users successfully finished the exercises they would have solved the problem. This challenge was estimated to take about 20 minutes. The last step was a questionnaire that the users had to fill about their experience, and their thoughts on the relevance of this platform. The questions were written in Portuguese but were translated to English for this paper.

4.1 The iris flower dataset problem

In order to keep the tests brief and not overly complicated we decided to introduce one of the common problems new data scientists learn during their training: the iris flower dataset. This data was collected by Edgar Anderson to quantify the morphological variation in iris flowers of three related species [21]. It contains a total of three species of iris: Iris Setosa, Iris Versicolour, Iris Virginica; and consists in the measurements of the species of iris and the dimensions of its petals and sepals (centimeters).

Based on the measurements, the users would then create a model that could predict the species of iris. The test was separated to 5 exercises:

1. The first exercise consisted in scaling the attributes of the dataset between 0 and 1.
2. Exercise two required the user to split the dataset to training and testsets (60/40%). The one for training would later be used to train the SVM model and the one for testing to see the accuracy and f-measure metrics.
3. Exercise three combined the first and second one. This was set to show the user that tasks can be added and removed from the workflow and applying feature scaling to an already created workflow was at a distance of a few clicks.
4. In exercise four the user was asked to add the Relieff algorithm to the workflow in order to see what attributes would have the most predictive capabilities.
5. Exercise number five used the best two attributes discovered in the previous exercise and added the validation procedure called K-fold cross validation, hence completing the assignment and creating a model.

The exercises were simple and intertwined making the user have a feeling of progress during their execution.

5 Results

5.1 Questionnaire

The questionnaire allowed us to know how much the users liked the interface, their experience using the tool and if they found it useful. Each statement could be answered as: totally disagree, disagree, indecisive, agree and totally agree. In order to analyze the average response and the standard deviation we converted the answers to numbers, where number 1 translates to "totally disagree" and 5 to "totally agree".

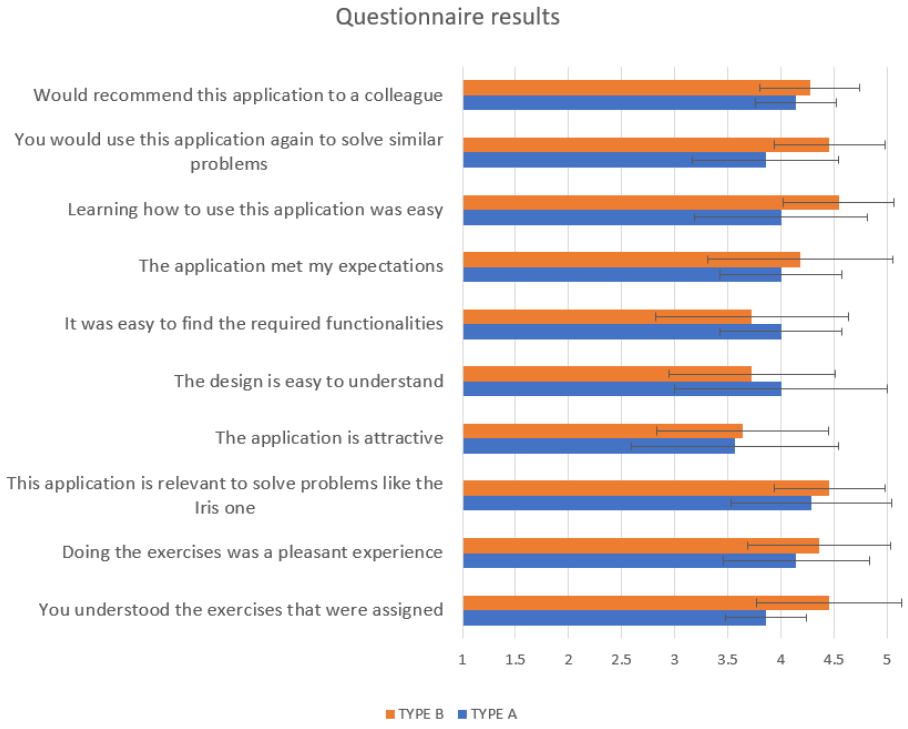

Fig. 4. Average and standard deviation of the users' responses

As seen in Fig. 4 the values are all above average. The most satisfactory results where that they found the interface easy to use, they would recommend it to colleagues and that they would use it again to solve related problems. The attractiveness of the interface, even though it was very positive, scored lower

than the other metrics; this was expected since this is a prototype and that part was not a priority. The results acquired from the type A users are lower than the ones from type B. This showed that the users with no experience (type A) had an higher difficulty using the interface, but surprisingly they found easier to find the required functionalities and the design simpler to understand. To assess whether the differences in the answers among the two populations were statistically significant, we performed unpaired statistically significant tests. Before that, both distributions were tested for Gaussianity using the Kolmogorov-Smirnov test. In case both distributions were Gaussian, an unpaired T-test was conducted; otherwise, a Wilcoxon rank sum test was performed. Hence, for each of the ten questions, only the question "You understood the exercises that were assigned" showed statistical significant differences among the two populations (at $p < 0.05$). We hypothesise that it was easier for the type A subjects to understand the exercises because they had experience in data mining and knew about the Iris dataset since it is a very common problem to teach new data scientists.

5.2 Feedback

Besides answering the questionnaire the users also had a place to write suggestions, critiques and things they liked better in the application. Bellow we have a list with the compilation of comments the users wrote for each of these topics.

Key **suggestions**:

- It should be possible to see all the tasks that were added to the workflow at all times. At the moment the user needs to scroll up and down to see and edit the tasks.
- In the dataset input, the option of selecting attributes to remove from the dataset should be replaced with attributes to select.

Key **critiques**:

- Sometimes the users did not know that a task belonged to a certain type, e.g feature scaling is a task that is of preprocessing type but some users when asked to use it did not intuitively know that it was of that type. This is a problem because the users first select the type of task they want to choose from and then the task itself. This problem can be solved by finding an alternative for the way users add tasks to the workflow or having a place where the users can search for all available tasks and read more information about them. Since this is a prototype and at the moment we do not support that many different tasks, the users found that task in a few seconds. However but if there were more tasks it would be harder.
- To use a dataset in any workflow the users must copy the dataset's uri that is shown in the sidebar and paste it to the dataset input task. Some users did not found that intuitive and another alternative should be taken to consideration.

Things they **liked the most**:

- Simplicity, accessibility and design.
- Low learning curve and easiness to use.
- How fast it was to run an experiment and get the results.
- Intuitiveness.
- It does not require any installation and it can be used anywhere with internet access.
- The tasks were chained together guiding the process of constructing the workflow.
- Grid search.
- The outputs are direct and very informative.

This feedback reinforced what was discovered during the questionnaire and was very satisfactory, none of the critiques were about the concept we aim to prove and the things they liked the most were inline with the objectives we tried to achieve when building the application.

6 Conclusion

In this work we presented a service for non-programmers to build Data Science experiments employing good data mining practices. We prototyped a cloud application that follows a microservices architecture. The interface built tried to achieve a high degree of simplicity and usability. To test it, experiments were made with experienced and non-experienced users to evaluate the prototype and validate the paradigm of visual programming using sequential tasks. The results were satisfactory with a positive feedback and without critiques related to what we are trying to achieve. In the future we plan to add predefined data science workflow templates that might be searched, changed and shared by the users, as well as make comparative benchmarks with other platforms. Regarding the usability tests we plan to improve the application by making changes to the user interface according to the feedback received.

Acknowledgments

This work was carried out under the project PTDC/EEI-ESS/1189/2014 Data Science for Non-Programmers, supported by COMPETE 2020, Portugal 2020-POCI, UE-FEDER and FCT.

References

1. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Hung Byers, A. : Big data: The next frontier for innovation, competition, and productivity. McKinsey & Company, (2011).
2. Henke, N., Bughin, J., Chui, M., Manyika, J., Saleh, T., Wiesman, B., Sethupathy, G.: The age of analytics: Competing in a data-driven world. McKinsey & Company, 2016.

3. Miller, S., Hughes, D.: The Quant Crunch: How the Demand For Data Science Skills is Disrupting the Job Market. Burning Glass Technologies, 2017.
4. Becoming a Data Scientist Curriculum via Metromap, http://nirvacana.com/thoughts/2013/07/08/becoming-a-data-scientist/. Last accessed 14 Jun 2018
5. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: The KDD process for extracting useful knowledge from volumes of data. Communications of the ACM, **39**(11), 2734, (1996).
6. Domingos, P.: A few useful things to know about machine learning. Communications of the ACM, **55**(10) 77-87, (2012)
7. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. 2nd edn. Springer, (2001)
8. Cawley, G. C., Talbot, N. L.: On over-fitting in model selection and subsequent selection bias in performance evaluation. Journal of Machine Learning Research, **11**(07), 2079-2107, 2010
9. AzureML, https://studio.azureml.net/. Last accessed 14 Jun 2018
10. H2O.ai, https://www.h2o.ai/. Last accessed 14 Jun 2018
11. Orange, https://orange.biolab.si/. Last accessed 14 Jun 2018
12. Weka, https://www.cs.waikato.ac.nz/ml/weka/. Last accessed 14 Jun 2018
13. RapidMiner, https://rapidminer.com/. Last accessed 14 Jun 2018
14. Kranjc, J., Orač, R., Podpečan, V., Lavrač, N., Robnik-Šikonja, M.: ClowdFlows : Online workflows for distributed big data mining. Future Generation Computer Systems, **68** 3858 (2017)
15. Scikit-Learn, http://scikit-learn.org/stable/index.html. Last accessed 14 Jun 2018
16. Medvedev, V., Kurasova, O., Bernatavišienė, J., Treigys, P., Marcinkevišius, V., Dzemyda, G.: A new web-based solution for modelling data mining processes. Simulation Modelling Practice and Theory, **76** 3446 (2017).
17. Krstajic, Damjan: Cross-validation pitfalls when selecting and assessing regression and classification models. Journal of Cheminformatics,
18. Zorrilla, M. García-Saiz, D.: A service oriented architecture to provide data mining services for non-expert data miners. Decision Support Systems, **55**(1) 399411 (2013)
19. Netflix Conductor, https://netflix.github.io/conductor/. Last accessed 14 Jun 2018
20. Competing consumers pattern, https://docs.microsoft.com/en-us/azure/architecture/patterns/competing-consumers. Last accessed 14 Jun 2018
21. Edgar Anderson: The Species Problem in Iris. Annals of the Missouri Botanical Garden, **23**(3) 457-509 (1936)