Big Data Analytics APIs Architecture for Formative Assessors

Wassim Mahfouz
Faculty of Computer Science and Automation
Ilmenau University of Technology, Germany
wassim.mahfouz@tu-ilmenau.de

Heinz-Dietrich Wuttke
Faculty of Computer Science and Automation
Ilmenau University of Technology, Germany
dieter.wuttke@tu-ilmenau.de

Abstract*— This Research to Practice Full Paper is driven by the question: Within limited time resources available to trainers in projects for Big Data Analytics (BDA) problems, how can they define project requirements for Formative Assessment (FA) actions? The paper suggests BDA APIs architecture as helping tool for formative assessors. It helps them effectively produce and adapt visual diagnostic reports for FA-actions in agile based requirements (i.e. features) definition. The paper presents two core architectures: Architecture for a parametrized feature-descriptor-system to define/refine a BDA API feature and its visual diagnostic reports, and an initial resources architecture for BDA API to initialize an analytics algorithm with its input big data sets. Clarifying visually the trainee's challenges (i.e. incremental features in a BDA API) is our main FA action. The FA action is designed based on Csikszentmihalyi's flow model to support a trainee in matching balance between his/her challenges and his/her skills. To test the architecture's functions, the paper has test setups for two formal projects (each has 1 to 6 trainees) and two informal projects (each has 1 to 3 trainees). The projects are to attack BDA problems in learning analytics and in image automatic classification. The test results show that the visual diagnostic reports produced by the trainers are very effective in clarifying visually incremental BDA API features not only for simple classifiers (i.e. classical data mining algorithms) but also for complex classifiers (i.e. deep learning algorithms). The results show also how visual diagnostic reports are easily produced for comparing the algorithm performances using different input big data sets, whereas other reports are produced for comparing performances between different algorithms, using one input data set. Related works are also discussed to show the architecture's differences and advantages. Its main advantages are: 1) it enables the trainers to use deep learning algorithms beside classical data mining algorithms in its BDA API parameterizable feature descriptors for visual diagnostic reports. 2) The descriptors can be extended, reused, shared, and scaled out to help trainers in other universities providing flow model based FA actions. 3) Finally, it has extensions to integrate other theoretical frameworks like Buckingham Shum and Deakin Crick's framework for dispositional learning analytics instead of the used flow model.

Keywords— assessment in engineering education, planning for formative assessment, Big Data Analytics.

I. INTRODUCTION

Assessment is an essential component of learning and teaching, as it allows the quality of both teaching and learning to be judged and improved [1]. It often determines the priorities of education [2], it always influences practices on learning [3]. Changes in curricula and learning objectives are ineffective if assessment practices remain the same [4], as learning and teaching tend to be modelled against the assessment purposes and goals. Assessment is usually understood to have two purposes: summative and formative. Summative assessment is used to judge students' achievements at the end of an instructional unit, whereas formative assessment aims to gather evidence about students' learning progress to influence teaching methods and priorities [2]. Although researchers have long advocated the potentially positive influences of formative assessment on learning, its implementation has been challenging. One challenge is the alignment of formative assessment activities/practices during the teaching or training offers. This challenge will be very difficult for a teacher of a big class because of his/her limited time resources and the big number of students. In [5], we have presented an automatic classifiers framework and its data integration tool to support teachers of big classes in planning for formative assessment during teaching offers (i.e. courses). In this paper, we are interested in the implementation challenges of the formative assessment not during teaching offers but during training offers. As an example, in third semester of a bachelor program, we offer for 6 students a training project for industrial problems. If the problems are for BDA, how can we define project requirements for formative assessment actions? This question drives our contributions in this paper. Our main contributions are:

- For agility in BDA training projects, we have built a BDA APIs architecture to enable trainers produce and adapt visual reports for clarifying the trainee's incremental challenges. Its innovative design provides two effective helping tools.
    1. Tool for visual diagnostic reports. It enables trainers produce and adapt algorithm performance diagnostic reports to clarify visually incremental BDA API features (i.e. trainee's challenges) for automatic classifiers. (Sections 3, 4, 6)
    2. Tool for automatic producing of API visual documentations. It saves the trainer's time resources required for requirements clarification and documentation.

In Section 7 the related works and in Section 8 the conclusions and future work will be presented. In next section, we present our method for agility in training.

II. OUR METHOD: FLOW-MODEL-BASED FORMATIVE ASSESSMENT ACTIONS FOR AGILITY IN TRAINING

Agility in training is the ability to respond to changing trainee needs in different training projects. To facilitate effective responding, we integrate FA-actions in BDA training

projects by using the iterative process illustrated in Fig.1. Fig 1 shows how the process's iteration 0 is used to estimate the trainee needs to initialize training goal oriented features. Beside iteration 0, the process has a group of repeated iterations (1 ...n) for rapid development. Test Driven Development (TDD) is recommended to trainees because it is the best known method to practice incremental design, but also a trainee can select what fits his/her style in programming and testing. Each iteration is started by a meeting to plan a list of BDA API features. A feature (i.e. a high level requirement) then is selected, to be divided into incremental sub-features (i.e. trainee incremental challenges). Clarifying trainee incremental challenges is our main FA action. The FA action is designed based on Csikszentmihalyi's Flow Model [6] (see Fig 2) to support a trainee in matching balance between his/her challenges (e.g. incremental challenges /features for BDA API performance improvement) and his/her skills for developing analytics algorithms. Fig.2 shows eight mental states in terms of a trainee challenge level and his/her skill level. Fig.1 has three trainer tasks to highlight how our main FA action is integrated in an iteration. The tasks are:

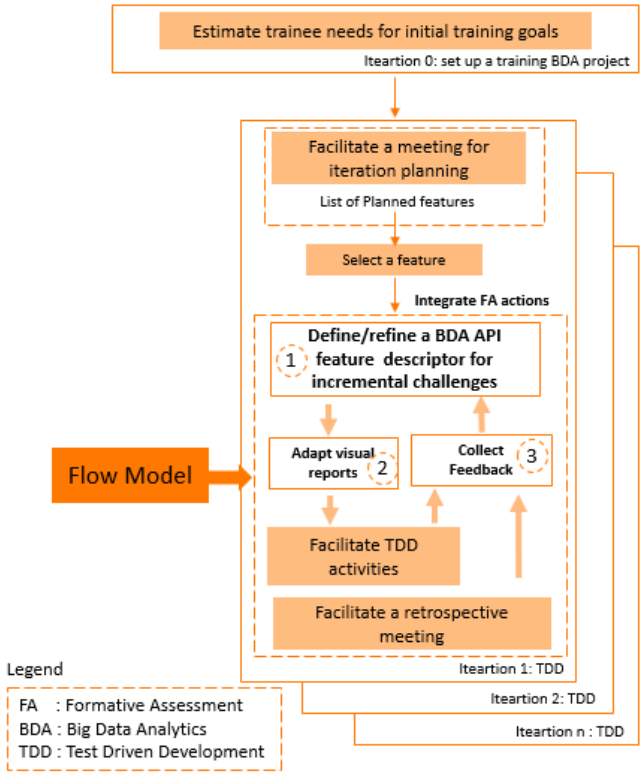

Fig. 1. Iterative process: trainer tasks to integrate FA actions in an agile BDA project

1. Define/refine a feature descriptor for incremental challenges. To put it differently, a feature descriptor consists of reusable description templates for incremental sub-features and their visual diagnostic reports. For concrete example a feature to improve BDA API performance can be divided into two incremental sub-features namely "improve individual algorithm performance in the API", and "performance comparison between many algorithms to optimize the API for best algorithm". For each sub-feature, visual performance diagnostic reports have to be described using different performance metrics (table 4 in sections 5 and section 6 have more details for this concrete example).

2. Adapt visual reports for clarifying incremental trainee challenges (e.g. a feature and its sub-features for BDA API performance improvement) to help trainees develop analytics algorithms in a BDA API.

3. Collect trainee feedback. The feedback is collected in context of two feedback activities namely:

- During an individual training offer. Our offers have many training formats like brain storming activity, code review activity or pair programing for TDD activity.
- We facilitate retrospective meeting for a trainee or group of trainees in each iteration end (see Fig.1).

During these activities, we ask questions about the 8 states to collect trainee feedback to refine the visual reports in the feature descriptor for next incremental challenge (see Fig.1).

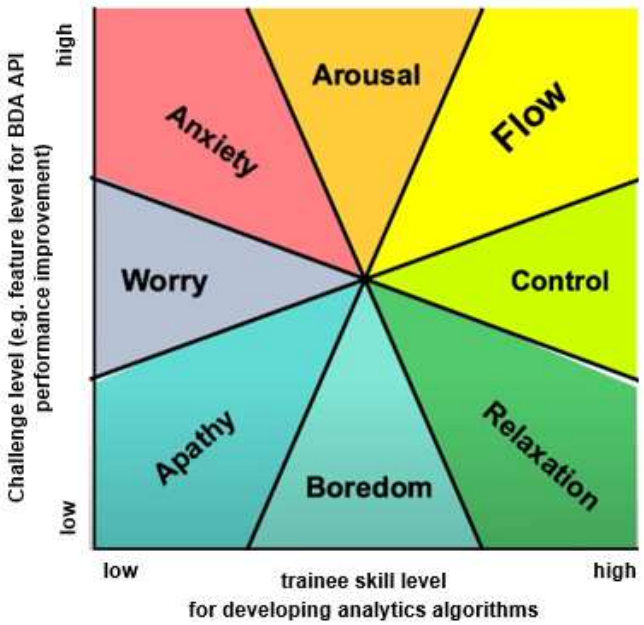

Fig. 2. Flow Model [6]: The eight mental states in terms of challenge level (e.g. feature level for BDA API performance improvement) and trainee skill for developing analytics algorithms

The iterative trainer tasks (see Fig.1) are extremely time consuming. Therefore, we automate some of the repeated tasks to make agility in training feasible for multidisciplinary training projects. Because our projects include intensive trainee needs or requirements for internal and external designs, we suggest BDA APIs architecture as helping tool for trainers. We select BDA API as first class property, because API driven architecture will give the trainers suitable descriptive power. Based on that, trainers have a tool for effective responding to the trainee's needs in internal and external designs during a BDA API training project. We facilitate API designs via parametrized BDA API feature descriptors. For our parametrized BDA API feature-descriptor-system, we present its architecture in next section.

III. Architecture for a Parametrized BDA API Feature-Descriptor-System

The architecture includes two diagrams; system context diagram, and standalone system diagram.

A. System Context Diagram

Fig.3 shows the system context diagram.

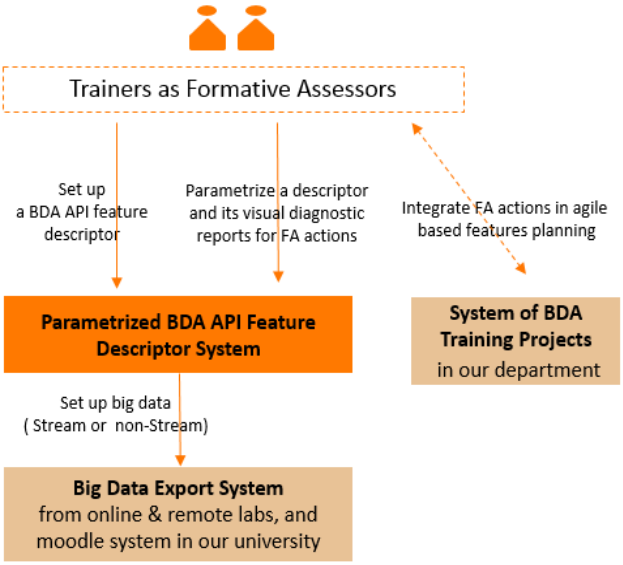

Fig. 3. System context diagram

The diagram summarizes how a trainer can set up a BDA API feature descriptor, parametrize it with its visual diagnostic reports for FA actions, and integrate the FA actions in agile based features planning (i.e. defining/refining). The parametrized BDA API feature-descriptor-system is designed to operate not only in context of the two surrounding systems in our university, as seen in Fig.3, but also to operate in context of surrounding systems in other universities. To put it differently, it is designed to operate or run standalone, as we will explain in next section

B. Standalone System Diagram

Fig.4 shows the standalone parametrized BDA API feature descriptor system for FA-actions in agile based features definition. The system consists of client and server for feature description beside an embedded NoSQL store for sampled big data sets. This store enables the trainers to run the system as standalone. The store's sampled big data sets are published based on the data privacy rules in EU. If the published big data sets do not fit a trainer, he/she can activate the system's maintenance mode to use its maintenance UIs to import his/her big data sets into the embedded NoSQL store. Because the target of this paper is not to explain our system's maintenance mode, the maintenance UIs in Fig.4 are deactivated. In contrast to the deactivated maintenance UIs, the parametrization UIs for feature descriptor are designed to provide three helping tools/functions for 1) producing visual diagnostic reports, 2) automatic producing of BDA API visual documentations and 3) fast simulations for BDA API clients and servers. The client and its UI components communicate with the server via a transfer object called parameterizable feature descriptor as illustrated in Fig. 4. The object is a transfer representation of the server's main component called a BDA API feature descriptor. This descriptor is mapped to one requirement component that briefly describes a requirement in a list of BDA project's requirements. In BDA project component (see Fig.4), there are attributes for training format and BDA problem definition. Fig. 4 shows only "feature clear goal and incremental sub-features as the main attributes of the feature descriptor. Besides its main attributes, it has also two extension components for more detailed descriptions. The components are:

- Model Extension: This enables a trainer to use a suitable model to facilitate the BDA API feature description process in formative assessment. The flow model is highlighted in Fig.4 because the paper's goal is to present our method: flow-model-based FA-actions for agility in training (as described in section 2). Our design for extensions, as seen in fig. 4, enables also a trainer to use other theoretical frameworks like Buckingham Shum and Deakin Crick's framework [7] for dispositional learning analytics instead of the used flow model.
- Resource Metadata Object: Fig.4 shows a metadata object to describe three resources, namely resource for analytics algorithms, resource for big data sets, and finally resource for diagnostic reports. For these three resources and other resources in a BDA API, we present an initial architecture in next section.

IV. Initial Resources Architecture for BDA API

Fig.5 shows the architecture's single BDA API metadata object associated with five resources: namely analytics algorithm, big data set, diagnostic report, simulation and documentation. The single metadata object is to simplify the parametrization process for each resource and its count (e.g. if its count is 3, then its association is 1...3). Via its centralized

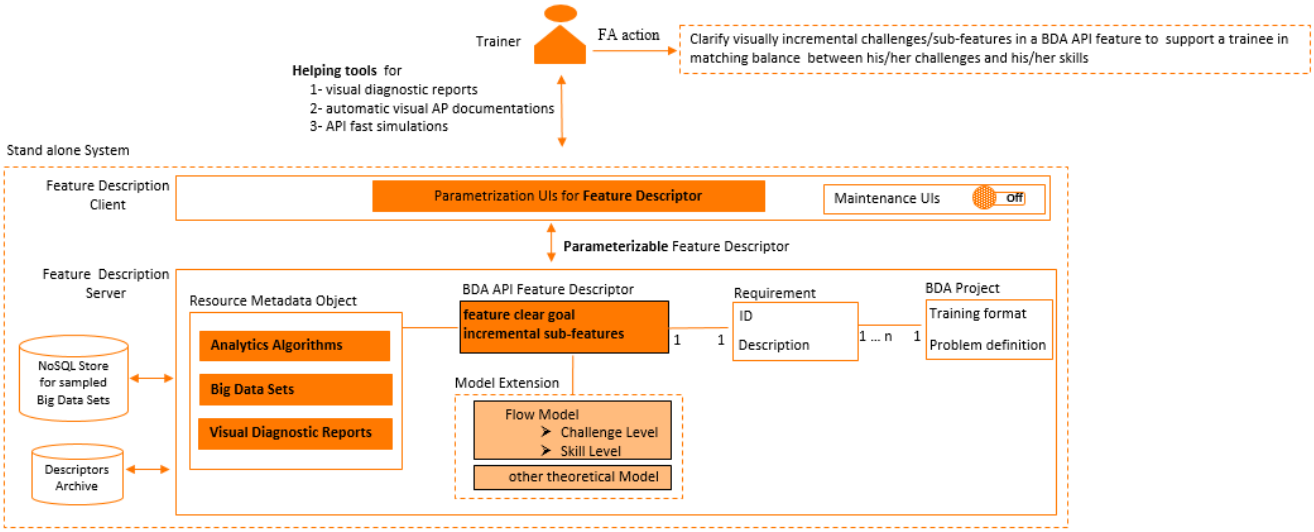

Fig.4 Standalone Parametrized BDA API Feature Descriptor System for FA-actions in agile based features definition

configurable options the metadata object saves time and offers flexibility not only in initial resource parametrization but also later in refinement resource parametrization, done as part of trainer's FA-action, to facilitate the matching balance between trainee's tasks/challenges and his /her skills. To simplify resource parametrization Fig.5 shows how we classify the configurable options for input and output resources.

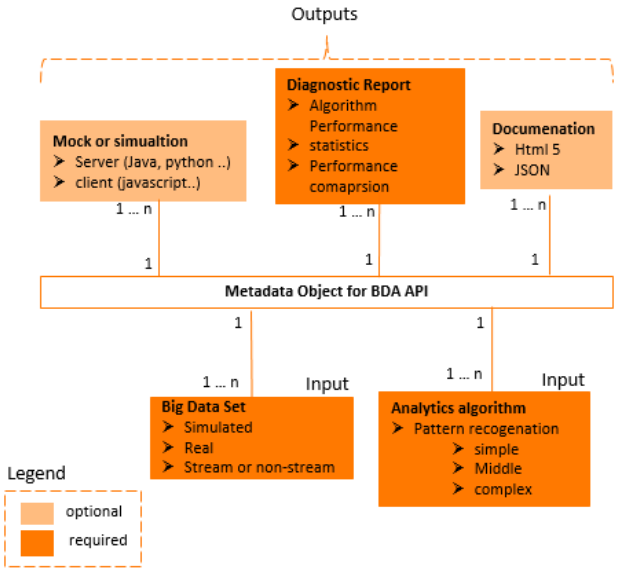

Fig. 5. Initial Resources Architecture for BDA API

It shows also if a resource is required or not (i.e. optional). For brevity, it shows only two or three configurable options for each resource. To give a simple example, the resource "diagnostic report" has three configurable options: algorithm performance, statistics, and performance comparison. In algorithm performance option, the trainer can get a report about the algorithm performance, whereas in statistics option, she can get a report about the big data set's characteristics. In contrast to algorithm performance option that provides a performance report of one algorithm, the performance comparison option provides reports to compare performance between different algorithms. Via these three configurable options, a trainer can effectively produce and adapt the visual diagnostic reports for clarifying visually the incremental challenges (i.e. a feature and its incremental sub-features) in a BDA API for simple or complex classifiers. Beside diagnostic reports, visual documentation reports can be automatic produced by trainers. In next section, we present test cases to show how producing and adapting visual reports (i.e. diagnostic and documentation reports) can be tested not only for simple classifiers (i.e. classical data mining algorithms) but also for complex classifiers (i.e. deep learning algorithms).

V. Test Cases for Producing and Adapting Visual Reports

Producing and adapting visual reports (diagnostic and documentation reports) to define/refine incremental BDA API features for automatic classifiers is tested in context of four BDA training projects described briefly in table 1 that has two columns; trainee info, and project info. Moreover, each project setup includes also big data sets and their characteristics. To give a concrete example, table 2 summarizes characteristics for the four big data sets shared between the four projects. The pre-defined classes: {Iris-setosa, Iris-versicolor, Iris-virginica} in Iris flower dataset or {success, fail} in CO dataset are examples for simple pattern recognition problems, beside simple pattern, the analytics algorithm resource (see Fig.5) has also middle and complex pattern recognition types to enable incremental BDA challenge formulations.

Table 1. Four BDA training projects

id	Project Info	Trainees Info
1	Goal: learning analytics visualization APIs Context: bachelor program (third semester). It lasts 24 weeks for 8 iterations Format: formal	Count: 6 Trainees, Culture background: West Europa
2	Goal: prediction algorithms review for learning analytics. Context: international engineering master program (first semester) Format: formal	Count: 1 Trainee, culture background: South Asia
3	Goal: image automatic classifiers APIs Context: internship program for 1 year Format: informal	Count: 3 Trainees, culture background: East Europa
4	Goal: image automatic classifiers APIs Context: bachelor program (fifth semester) Format: informal	Count: 1 Trainee, culture background: Middle East

Table 3 has two prerequisite test cases for setting up input datasets. The first prerequisite test case is used in table 4 that has a test case for a BDA API feature: "improve performance in a BDA API for simple classifiers". The test case in table 4 is to test how trainers can produce and adapt visual reports (diagnostic and documentation reports) to define/refine two incremental sub-features, 1 and 2 for simple classifiers.

Table 2. Four Big Data Sets shared between the Projects

id	Name	Description
1	CO dataset [5]	It is about exam scores for 129 students. The scores collected during the Computer Organization (CO) course in winter semester (2019). It has 129 instances. Each instance is a 9-dimensional attribute vector. The vector's 9 attributes contain student eaxm scores in nine time points. Each point is for 2 weeks It has a pre-defined set of classes: C = {c_1, c_2... c_m}, for example C = {success, fail}, or C = {Excellent, good, middle, fail}.
2	Iris flower dataset [8]	It has 150 instances. Each instance is a 4-dimensional attribute vector. The 4 attributes are: sepallength, sepalwidth, petallength and petalwidth. For each class there is 50 instances. It has 3 pre-defined classes: {Iris-setosa, Iris-versicolor, Iris-virginica}
3	MNIST dataset [9]	MNIST (Modified National Institute of Standards and Technology database) dataset consists of handwritten sets of numerical digits ranging from 0 to 9.
4	Fashion-MNIST [10]	Fashion-MNIST, a product images dataset, which comprises of 28x28 grayscale images. It has 10 fashion item classes such as T-shirt/Top, Trouser, Pullover, Dress, Coat, Sandals, Shirt, Sneaker, Bag, Ankle boots.

Table 3. The Prerequisite Test Cases

id	Prerequisite test case for setting up input datasets
1	Parametrize or adjust BDA API feature descriptor for dataset 1, and 2 in table 2
2	Parametrize or adjust BDA API feature descriptor for dataset 3 and 4 in table 2

Table 4 summarizes the test steps (namely six steps for sub-feature 1 and two steps for sub-feature 2). The steps validate how the architecture's functions (i.e. helping tools in Fig.4) enable the trainers produce visual reports to clarify visually trainee challenges (i.e. the two sub-features) for BDA API performance improvement. The steps also validate how the trainers can adapt the visual reports based on trainee feedback collected using the flow model, in context of the four

training projects. The most repeated and representative trainee feedbacks, in the four projects, are presented in table 4. Table 4 shows four trainee feedback (namely A, B, C and D) embedded in the six steps of sub-feature 1. As demonstrated in table 4, a test case for complex classifiers (i.e. deep learning algorithms) in a BDA API can be also presented. Presenting a test case for complex classifiers is removed for brevity. In next section we present the test results and discuss how they support a trainee in matching balance between her challenges and her skills

TABLE 4. TEST CASE: IMPROVE PERFORMANCE IN A BDA API FEATURE FOR SIMPLE CLASSIFIERS

Prerequisites: a) Test cases 1 and 2 as defined in Table 3. b) In the four training projects, (see table 1) we announced to the trainees how we offer in context of TDD activities and retrospective meetings short interviews to collect feedback using the flow model.		
Feature: Improve performance in a BDA API for simple classifiers. **sub-feature 1**: improve individual algorithm performance in an API **sub-feature 2**: performance comparison between many algorithms to optimize the BDA API for best algorithm		
id	Steps for producing and adapting visual reports (diagnostic and documentation reports)	
	Test Step Details	test results
1.1	Define a BDA API feature descriptor to parametrize simple classifiers (i.e. classical data mining algorithms) and its input datasets	See Fig 6
1.2	produce automatically a visual API documentation for feature descriptor defined in step 1.1	See Fig 7
trainee feedback A	balance classes in a training data set is not clear for some of trainees and estimated mental state for some of them are ” apathy and boredom “	
1.3	Produce visual statistics reports to discuss the topic “balance classes in a training data set”	See Fig 8
trainee feedback B	Performance accuracy is not clear to some of trainees: Estimated mental state for some of trainees is ” worry “	
1.4	Produce visual reports to clarify the term accuracy in performance improvement goals. Produce two reports to help the trainees reduce their worry. Report 1 for clarifying test mode options. Report 2 for clarifying algorithm input parameter values	See Fig.9 and 10
trainee feedback C	detailed accuracy is not clear for the trainees: Estimated mental state ” control “	
1.5	Adapt visual report for detailed accuracy by class (i.e. precision, recall, f-measure) for an algorithm.	See Fig 11
trainee feedback E	Estimated mental state ” control “	
1.6	Adapt report for clarifying how algorithm accuracy can be compared using 2 input big datasets.	See Fig 12
2.1	Adapt report to clarify visually how to compare the performance between the four algorithms using the CO Dataset	See Fig 13
2.2	For other datasets, repeat step 2.1	

VI. TEST RESULTS AND DISCUSSION

For discussion, we present first the test results to improve performance in a BDA API for simple classifiers as reported in table (4), and second the test results to improve performance in a BDA API for complex classifiers (i.e. deep learning algorithms). For step 1.1 in table 4, Fig.6 shows the BDA API feature descriptor, parametrized for simple classifiers. It has easy to read, text formatted parameters, to set up classical data mining algorithms and its input big datasets. The descriptor is used as input for automatic producing of visual HTML5 documentation report shown in Fig 7 (i.e. test results for step1.2 in table 4). This automatic producing saves the trainer’s time resources required for requirements/features clarification and documentation. Fig.7 shows visually the flexible options selected to set up four algorithms namely: OneR [11], naive bayes [12], k-nearest neighbors [13], and C4.5 [14] for simple classifiers and two input big datasets namely: CO dataset, and iris flower dataset. The descriptor and its flexible options in Fig 7 can be easily reused and extended by trainers in other universities because we have designed the BDA API feature descriptor for reusability and easy extension using the Open API specification swagger [15]. In Fig.8 a visual statistics report for datasets (CO and iris flower) is produced, as agile respond to the trainee feedback A in table 4. The report helps the trainees, who have low skills in training a machine learning algorithm, get rid of their apathy and boredom mental states, because it clarifies visually the topic “balance classes in a training data set. It shows that “Iris flower dataset” has three balanced classes whereas the CO data set hast two unbalanced classes. Moreover, the trainee feedback B, in table 4, shows that the term “performance accuracy“ is not clear to some of the trainees, and the trainees are worry , in terms of flow model, because they have limited skill in developing analytics algorithms for API performance improvement.

```yaml
swagger: "2.0"
info:
  description: "BDA API feature descriptor for simple classifiers.
  version: "1.0.0"
  title: "BDA API Feature-Descriptor"
host: "localhost.feature-descriptor-system.com"
basePath: "/v1"
tags:
  - name: "Feature-Descriptor"
    description: ""
schemes:
  - "https"
  - "http"
paths:
  /v1/prametrize:
    post:
      tags:
      - "simple classifers: classical data mining algorithms"
      summary: "algorithms and input big datsets "
      description: ""
      parameters:
        - in: body
          description: "simple classifers"
          required: true
          name: name
          schema:
            type: array
            items:
              type: string
            example: ["OneR algorithm", "naive bayes algorithm",
            "k-nearest neighbors algorithm","C4.5 algorithm"]
      responses:
        default:
          description: OK
definitions:
  big-datsets:
    type: "object"
    properties:
      dataset-1:
        type: "string"
        example: "CO dataset"
      dataset-2:
        type: "string"
        example: "Iris flower dataset"
```

Fig. 6 Feature descriptor for simple classifiers. Swagger format [15]

As agile respond to support trainees reduce or get rid of their worry, in Fig 9 a report is produced to highlight the accuracy metric. Fig 9 highlights also how options for test mode, can improve the accuracy for the C.4.5 algorithm. The report shows that the 10-fold cross validation technique is better than the test mode “spilt 66% for training and 34% for testing”. Not only the test mode options can help the trainer to clarify the performance improvement goals of an individual algorithm, but also options for the algorithm’s parameters can be visualized using reports like report in Fig.10. The report shows how the accuracy for C 4.5 algorithm is decreased by

increasing the value of the algorithm input parameter "max number of instances in leaf". Both visual reports in Fig 9 and 10 support a trainee in matching balance between her/his challenge/ sub-feature 1 (i.e. improve individual algorithm performance in a BDA API) and her/her skills for developing a classical data mining algorithm like C.4.5 algorithm. The trainee Feedback C, in table 4, shows that the estimated mental state for some of the trainees are ""control" that means the trainees have the required skills as developers to handle the next performance improvement goals, therefore a report in Fig 11 is produced to clarify visually the detailed accuracy by class for C 4.5 algorithm. For the classes (success and fail), the report highlights three performance metrics; namely precision, recall and F-measure.

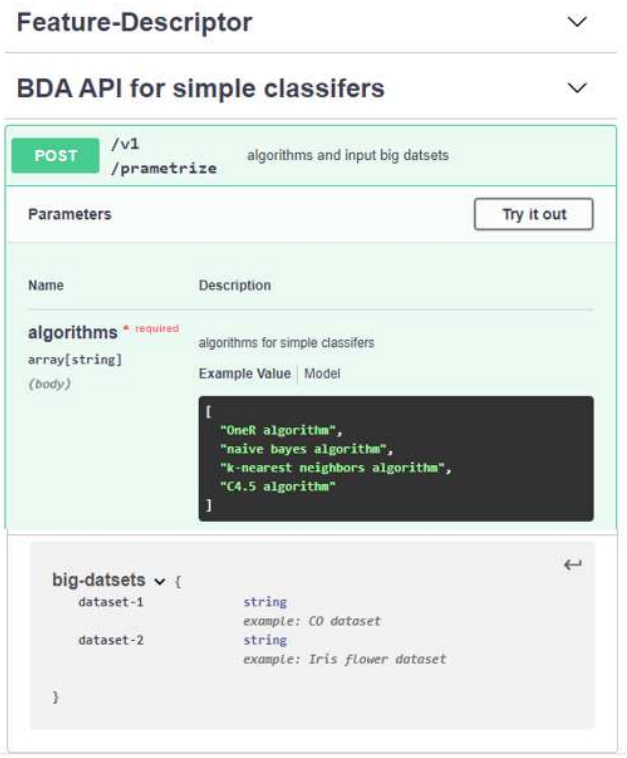

Fig. 7 Automatic produced visual HTML5 documentation report

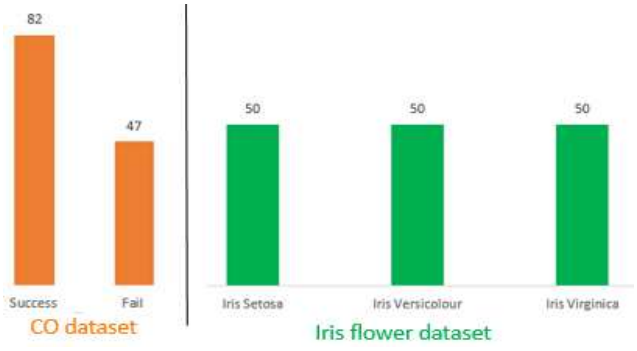

Fig. 8. Statistics report to discuss the topic "balance classes in a training data set"

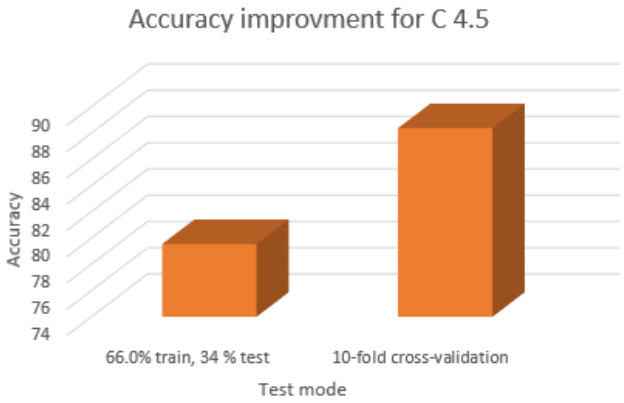

Fig 9. Accuracy improvement based on test mode options

For trainees who are in control (based on trainee feedback E in table 4) of developing algorithm performance improvement features in BDA API, the trainer can adapt visual reports to clarify more advanced performance goals via comparing algorithm accuracy using 2 input big datasets as seen in Fig.12. Fig.12 (i.e. test results for step 1.6 in table 4) shows an example report for k-nearest neighbors' algorithm using two input datasets.

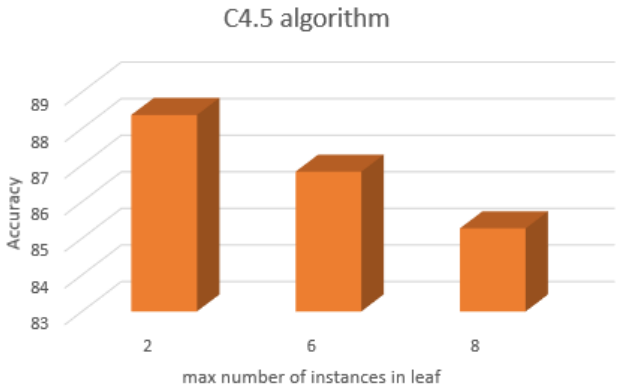

Fig.10 Report to clarify the performance of an individual algorithm based on values of its input parameter: max number of instances in leaf

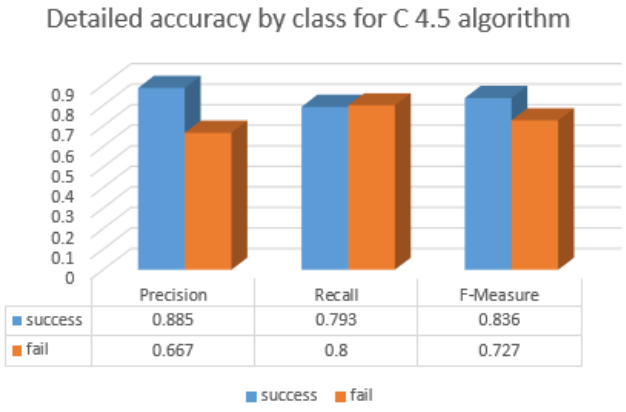

Fig 11. Report for detailed accuracy (precision, recall and F-measure), test mode: split 66.0% train, remainder test, dataset: CO Dataset

To clarify visually the more advanced performance improvement goals for sub-feature 2 (i.e. test results for step 2.1 in table 4), the trainer can adapt the visual report in the BDA API feature descriptor to produce a report as seen in Fig.13. Fig.13 shows a report to compare the performance accuracy for the four algorithms (OneR, Naive bayes, k-nearest neighbors, and C4.5) using the CO dataset. In term of

algorithm accuracy, the report helps the trainers discuss the topic “optimizing the BDA API for best algorithm”. The report shows that OneR outperforms C4.5, whereas Naïve bayes and k-nearest neighbors outperform OneR. Naïve bayes and K-nearest neighbors have same accuracy in Fig 13, therefore one of them can fit to optimize the BDA API for best algorithm. For brevity, we will not present visual reports for step 2.2 in table 4. Because it is about repeating step 2.1 with different input datasets. Enabling the trainers to adapt the visual reports in BDA API feature descriptor to provide reports like in Figs 11, 12 and 13 is very effective in responding to trainee’s needs. Via this effective responding the trainer can support a trainee in matching balance between his/her incremental challenges (for example more advanced performance improvement goals in a BDA API for simple classifiers) and his/her skills for developing analytics algorithms. To sum up, the discussion for simple classifiers (i.e. test case in table 4) as well as the visual reports in Figs 6, 7,…13, help the trainer clarify visually the trainee’s challenges (i.e. incremental features for performance improvement in a BDA API). It also enable him/her to clarify the simplicity and complexity of the selected algorithms in the BDA API under development to support a trainee in matching balance between her incremental challenges and her skills. To put it differently in concrete example, in Fig 7, the OneR algorithm is recommended to trainees who have low skills (i.e. beginners) in developing analytics algorithms, because it has a simple strategy. Its strategy assumes that there is one of the input data set’s attributes doing the entire prediction task. That is a simple strategy. Another simple strategy, is the opposite, to assume all of the input data set’s attributes equally and independently contribute to the prediction task. This is called the "Naive Bayes" algorithm and it is recommended to trainees who have basic skills in developing analytics algorithms because they have to think about and criticize the "Naive Bayes" algorithm’s two assumptions namely: the attributes are equally important and they are statistically independent. This independence assumption is never actually correct, but the algorithm based on it often works well for many datasets reported in practice. In contrast to simple strategy of OneR and Naive Bayes, the “C 4.5” and “K-nearest neighbors” algorithms have strategies that are more complex, therefore they are recommended to trainees who have more advanced or middle skills in developing analytics algorithms.

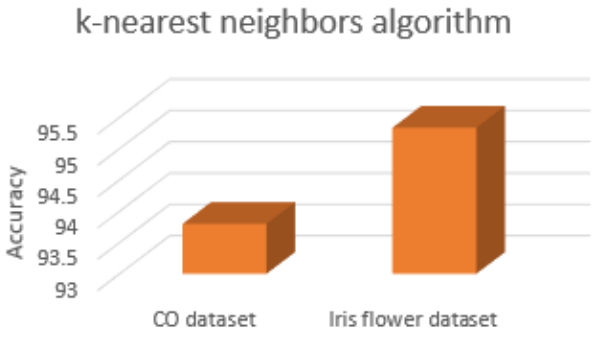

Fig.12. Accuracy report for one algorithm, many datasets,

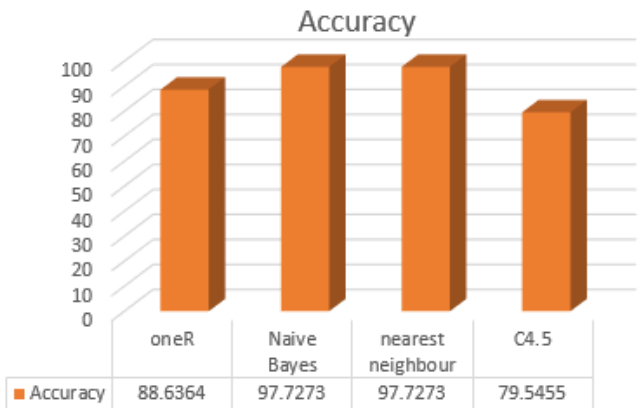

Fig. 13. Accuracy comparison report for four algorithms, Test mode: split 66.0% train, remainder test, dataset: CO Dataset

To clarify the BDA API performance improvement features for algorithms that have more complex strategies, the trainers can easily produce visual reports not only for classical data mining algorithms but also for more complex automatic classifiers that use deep learning algorithms. Fig.14 has a report for accuracy comparison for Convolutional Neural Network (CNN) algorithm [16]. The comparison between two CNN architectures, the first has 3 layers and the second has 5 layers. Layers details are described in table 5. The report in Fig 14 is produced using the MNIST dataset (see table 2), which contain ten classes (i.e. images for handwritten numbers 0…9). Using the same settings, Fig 15 has report for detailed accuracy by class for CNN 3-layers. The report shows the precision, recall, and f-measure for each class. To trainees who have high skills of developing deep learning algorithms or who are in control (in terms of the flow model) we recommended improving CNN algorithm performances (as seen in Fig 14 and 15) for automatic classifiers.

TABLE 5. DETAILS OF CNN LAYERS

	Layer details
CNN 3-layers	The four layers are : 1) convolutional, 2) subsampling , and 3) output
CNN 5-layers	The five layers are 1) convolutional, 2) subsampling , 3) convolutional , 4) subsampling and 3) output

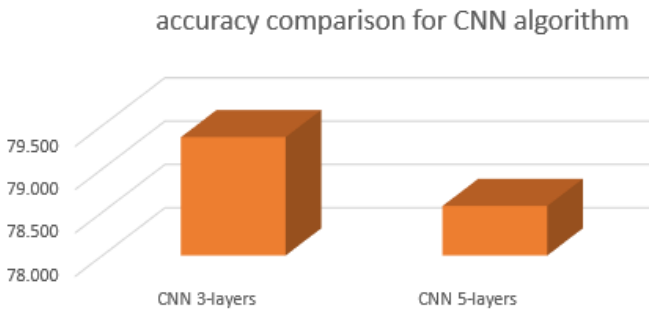

Fig.14. Accuracy report: MNIST dataset, 126 instances, split 70.0% train, remainder test

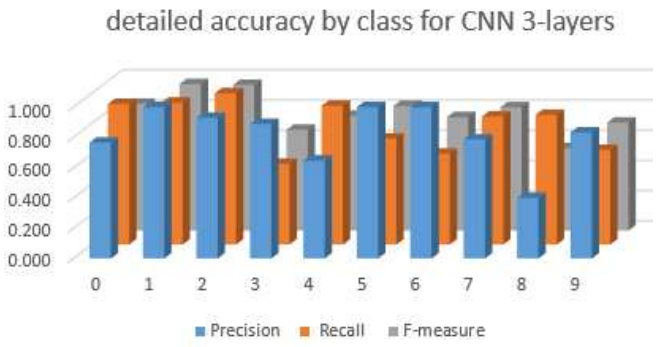

Fig.15. Detailed accuracy by class report: MNIST dataset, 126 instances, split 70.0% train, remainder test

Besides using CNNs for incremental development of automatic image classifiers as BDA API features, our BDA training projects, enhanced by the BDA API architecture's helping tools, enable also the trainees to experiment recurrent neural networks to incremental formulate and attack online learning analytics problems**.** To give concrete example**,** the student online temporal behaviors in our university Moodle LMS and our online Labs are exposed as time series data sets to be used as inputs for early predicting of student final exam score. Test results for defining BDA API incremental features for recurrent neural networks **(**namely long short-term memory (LSTM) networks and Feedback Neural Network (FNN**))** are not reported in this paper because future paper is planned for this goal. Last but not least, deep learing4java [16] and the weka tool [17] are recommended by us, but the trainees can also use Tensoflow [18], to implement the classifiers in the BDA API. For all classifiers in this paper the weka tool [17] and deep learing4java [16] are used for implementation. To summarize, Figs 6, 7, …15, show that the visual reports produced by the first two helping tools in Fig.4 are very effective for trainer's agile responding. In other words, they are very effective in definition/refining the incremental BDA API features with their diagnostic visual reports not only for classical data mining algorithms but also for deep learning algorithms. In contrast to, the first two helping tools, the third tool for API fast simulations (see Fig.4) was not effective for the trainers. That is because most of the students, in the four training projects, reported that they found difficulties in understanding the automatic produced codes for API servers/clients. Moreover, they reported that the codes are not useful for them. Finally, the two helping tools for visual reports were very effective because they help in clarifying visually the trainee's challenges (i.e. incremental features in a BDA API) to support him/her in matching balance between her challenges and her skills using the flow model. Last but not least, the BDA API feature descriptors and their diagnostic reports in our architecture are designed for reusability and extensions (see fig 6, and 7 for concrete example). That means they can be extended, reused, and shared, to support trainers in other universities. Demo about how other trainers can extend the BDA API feature descriptors for their agile training scenarios will be planned in our conference presentation.

VII. RELATED WORK

For predicting student performances, several works have been done by applying automatic classifiers techniques. In his PhD thesis [19] Thai-Nghe shows how the student performances can be predicted in an intelligent tutoring system. Whereas Thai-Nghe's work is limited to predict student performances inside an intelligent tutoring system, our work is different via supporting the trainers by effective tools to integrate FA actions for incremental development of automatic classifiers as BDA API features. The tools enable the cooperation between trainers and trainees, via flow model based FA actions to export/import big data sets, for predicting tasks not only inside the technology enhanced tutoring systems but also outside of these systems. In their inspiring Dirk et al [20] use a leaner regression algorithm to present a predictive model/automatic classifier and to study its stability and sensitivity with a large input data set. Our effective helping tools for incremental development of automatic classifiers differs the work of Dirk et al [20] in providing not only regression algorithms but also many classical data mining algorithms beside advanced deep learning algorithms (like Convolutional Neural Networks). In their study, F. Chen et al [21] have applied a deep learning approach –long short-term memory (LSTM) networks to analyze student online temporal behaviors using their LMS data for early prediction of course performance. For early prediction of course performance, our tools offer not only (LSTM) networks but also Feedback Neural Network (FNN**).**

VIII. CONCLUSIONS AND FUTURE WORK

The BDA APIs architecture was designed to enable trainers produce and adapt visual reports to clarify visually the trainee's challenges (i.e. incremental BDA API features). Its innovative design provides two effective helping tools for trainers: 1) Tool for visual diagnostic reports. It enables trainers produce and adapt algorithm performance diagnostic reports to define and clarify visually incremental BDA API features for automatic classifiers, and 2) Tool for automatic producing of API visual documentations. It saves the trainer's time resources required for requirements clarification and documentation. The tools were put under test and test results show that they are very effective for trainer's agile responding. That means in definition/refining the incremental BDA API features with their diagnostic visual reports not only for data mining algorithms but also for deep learning algorithms like Convolutional Neural Networks CNNs). In future work we plan to extend the deep learning algorithms in the BDA APIs architecture to include the recurrent neural networks **(**namely long short-term memory (LSTM) networks and Feedback Neural Network (FNN**)).** Moreover, we plan to extend the big datasets to include time series big data sets extracted from our university Moodle LMS and online Labs.

REFERENCES

[1] Ferrari, A., Cachia, R., Punie, Y.: Innovation and Creativity in Education and Training in the EU Member States: Fostering Creative Learning and Supporting Innovative Teaching. In: JRC-IPTS (2009).

[2] NACCCE: All Our Futures: Creativity, Culture and Education (1999).

[3] Ellis, S., Barrs, M.: The Assessment of Creative Learning. In: Sefton-Green, J. (ed.) Creative Learning, pp. 73–89. Creative Partnerships, London (2008).

[4] Cachia, R., Ferrari, A., Ala-Mutka, K., Punie, Y.: Creative Learning and Innovative Teaching: Final Report on the Study on Creativity and Innovation in Education in EU Member States. In: JRC-IPTS (2010).

[5] W. Mahfouz and H. Wuttke, "Automatic Classifiers for Formative Assessment," 2019 IEEE Frontiers in Education Conference (FIE), 2019, pp. 1-9, doi: 10.1109/FIE43999.2019.9028713.

[6] Mihaly Csikszentmihalyi : Good Business: Leadership, Flow, and the Making of Meaning PENGUING BOOKks, 2003.

[7] Buckingham Shum, Simon and Deakin Crick, Ruth (2012). Learning dispositions and transferable competencies: pedagogy, modelling and

learning analytics. In: 2nd International Conference on Learning Analytics & Knowledge, 29 Apr - 02 May 2012, Vancouver, British Columbia, Canada.

[8] https://archive.ics.uci.edu/ml/datasets/iris, last retrieved 10.05.2021

[9] http://yann.lecun.com/exdb/mnist/, last retrieved 10.05.2021

[10] https://github.com/zalandoresearch/fashion-mnist, last retrieved 10.05.2021

[11] https://www.dbs.ifi.lmu.de/~zimek/diplomathesis/implementations/EHNDs/doc/weka/classifiers/rules/OneR.html, last retrieved 10.05.2021

[12] https://www.cs.tufts.edu/~ablumer/weka/doc/weka.classifiers.NaiveBayes.html, last retrieved 10.05.2021

[13] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm, last retrieved 10.05.2021

[14] https://en.wikipedia.org/wiki/C4.5_algorithm, last retrieved 10.05.2021

[15] https://swagger.io/

[16] https://deeplearning4j.org/ java implementation for CNNs . last retrieved 10.05.2021

[17] http://www.cs.waikato.ac.nz/ml/weka/, last retrieved 10.05.2021

[18] https://www.tensorflow.org/, last retrieved 10.05.2021

[19] Thai-Nghe, Nguyen, “Predicting Student Performance in an Intelligent Tutoring System”, PhD thesis, Department of Computer Science Information Systems and Machine Learning Lab (ISMLL) University of Hildesheim, Germany,2011

[20] Dirk T. Tempelaar, Bart Rienties and Bas Giesbers, “Stability and sensitivity of learning Analytics based prediction Models”. 7th CSEDU 2015: Lisbon, Portugal.

[21] F. Chen, Y Cui: Utilizing Student Time Series Behavior in Learing Managment Systems for Early Predction of Course Performance. Journal of Learning Analytics Volume 7(2) (published 19.09.2020).