Milestone 4 Report

Machine Learning for Behavioral Data CS-421

María Isabel Ruiz Martínez 343154
Kai Cooper 342485
Nicolas d'Argenlieu 276507

Introduction

The growing accessibility of web technologies has helped sustain the development of online learning platforms such as Moodle and promote online teaching formats such as MooCs. They aim at providing more personalized and accessible education. One of these platforms is **Lernnavi** (lernnavi.ch), a high-school and technical middle-school online learning platform focused on teaching German and Mathematics. The project is backed by the Swiss canton of St. Gallen, with the cooperation of EPFL and the private high-school PH St.Gallen. The objective of the website is to prepare the students for the Matura (secondary education final diploma) using teachings and exercises. The platform's users can then track their progress through level checks assigning them mastery scores on a wide range of topics.

The extension of education to online frameworks has permitted the creation of a new and rich data source on learning behaviors. For this project, Lernnavi has partnered with EPFL to provide a dataset of user activity on the Lernnavi platform.

Using this dataset, our objective is to predict the student's improvement on level checks depending on their behavior on the platform. This way, we hope to gain insights on the type of behavior that leads to a more significant progress when using the platform.
To this end, we first preprocess the data and build new features out of it. Then we cluster our population into groups of students displaying similar behavior. Finally, for each of these clusters, we train a classifier using the level check results as labels. The end of this report discusses results and interpretations.

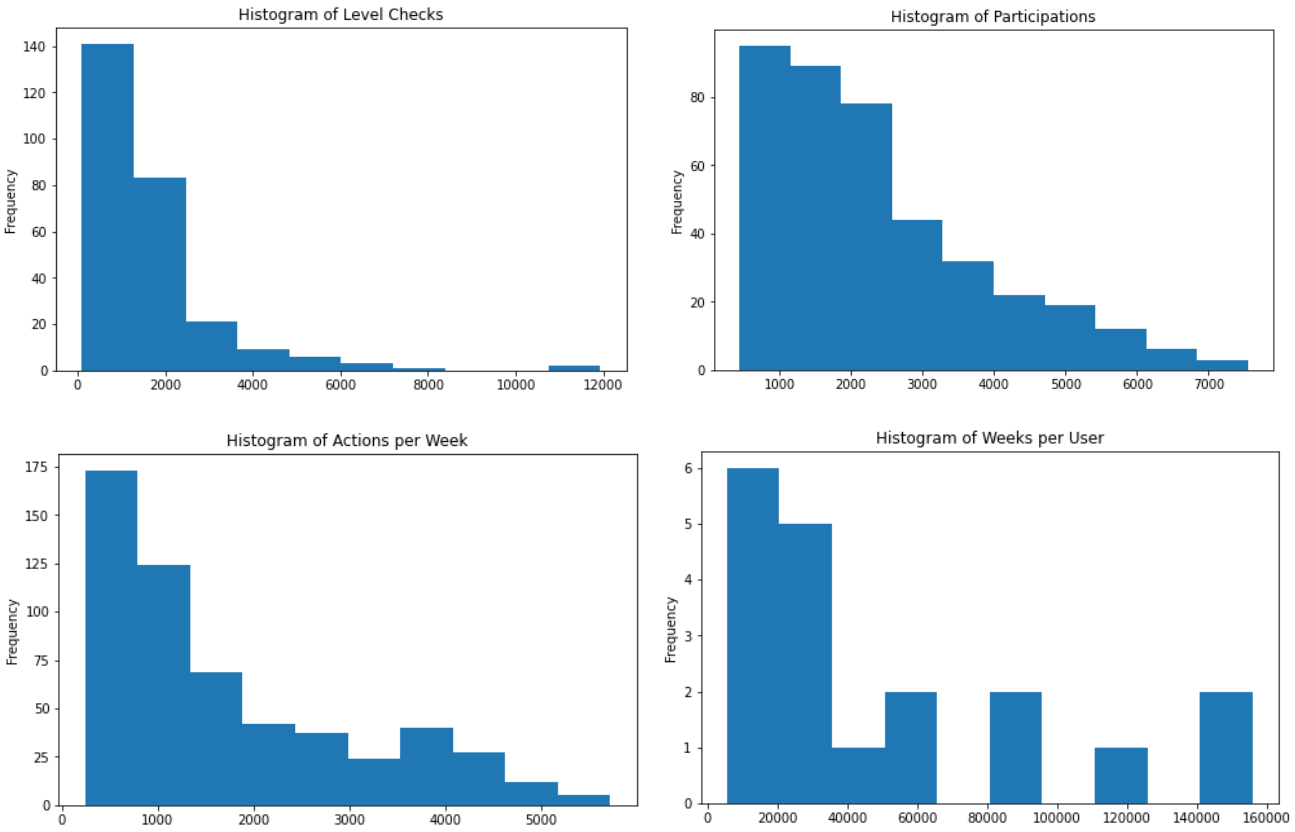
Data

Our dataset after the cleaning process consists of 790425 rows corresponding to actions of users on the platform. For each row, the following features will be detail:

- *'user_id'*: identifier of the user of the educational platform.
- *'timestamp'*: timestamp information of when the action was performed.
- *'week'*: derived from the feature 'timestamp', it corresponds to the week when the action was performed.
- *'category'*: classification of the action performed.
- *'action'*: type of action performed.
- *'start_time'*: start time of the action.

- *'commit_time'*: commit time of the action.
- *'num_checks'*: total number of level checks that the user with *'user_id'* has performed.
- *'num_participations':* total number of *'SUBMIT_ANSWER'* and *'GO_TO_THEORY'* actions performed by an user of the educational platform.
- *'num_actions_per_week'*: total number of significant actions performed by a particular user in a particular week. The definition of significant action can be found in the **Data Processing** section.
- *'num_weeks_per_user'*: total number of significant weeks performed by a particular user. The definition of significant week can be found in the **Data Processing** section.

The information corresponding to the level checks performed by the users of the educational platform is located in the *'learn_sessions_transactions'* table and can be linked to the *'transactions'* table via the *'transaction_id'*.



By plotting the histograms of some of the numerical features (*'num_checks'*, *'num_participations'*, *'num_actions_per_week'* and *'num_weeks_per_user'*), we can clearly see that none of them comes from a normal distribution. Indeed, they are all right skewed, which comes from the fact that only a few students are very active on the platform.

We also remark that there is a geographical element to the data, and this is important because it might introduce mixed effects between the groups of students by region. Consequently we investigated to some degree the cantonal representation within the dataset and how activity on the platform varied with location. At this stage it is quite preliminary is

essentially an aperçu into what additional features we may need to include in the model for more accurate results (find more details in the associated notebook.)

Research Question and Methodology

Using the previously described dataset, we aim at providing an answer to the following research question:

Is it possible to identify one or more studying behaviors leading to a significant improvement in level checks results?

Research into this question mainly provides insights to teachers and students using the platform; the results may indicate which learning habits and platform actions have the most impact on the level check results. However, for the developers, it may highlight how students are using their service on a weekly or hourly basis. We use level check score as a proxy for knowledge as Lernnavi also computes a “mastery score” from these level checks results.

Let us now outline our approach. First, we remove from the dataset the students which are identified as “inactive” (see **Data Processing** section) following criteria inspired by page 9 of Shirvani Bouroujeni, et al. We also remove features that are non-relevant for the clustering and/or classification.

Then, we compute online learning regularity metrics as defined in Shirvani Bouroujeni, et al. These metrics are shown to be efficient measures of the regularity in the online learning of a student. They have been defined using data from EPFL MooCs.

Following this data processing, we cluster our dataset on the different regularity features (see **Clustering and Classification** section). This method showed promising results in the original paper. By clustering, we hope to improve the accuracy of each one of our level check classifiers.

Finally, we train a classifier on each cluster (see **Clustering and Classification** section). The models classify each sample based on the predicted result of the level check, i.e. if the level check is predicted to succeed or not. As we will explain in the following sections, a level check outputs a *float value* as score. An increase of this score is considered to be a level check success whereas a decrease is taken as a failure.

To analyse the performance of the classifier, we will apply different metrics to qualify the performance of our classifications (see **Results** section). With this, we hope to provide confirmation that the data can be exploited in order to detect studying behaviors useful for the success of the students.

Data Processing

In the data processing phase, we first remove users with no transaction data (i.e. we keep users that are both in the events table and in the transactions table). To continue, inactive

users will also be removed from our dataset since we consider that they are not using the platform enough to extract any valuable conclusion from them.

Inactive students are students that are in one of the following situations:

- Students that have never performed a level check.
- Students that have not performed any *'GO_TO_THEORY'* action and neither any *'SUBMIT_ANSWER'* action.
- Students that have at least two *significant weeks*. A *significant week* is defined as a week with at least five *significant actions*. Moreover, the *significant actions* are the actions in the event table that are linked through the transaction token to the transactions table. As a consequence, we will exclude actions like *'LOGIN'*, *'LOGOUT'* or *'NAVIGATE_DASHBOARD'* from the list of *significant actions*.

Since our goal is to predict users' success in level checks based on their regularity in time, we are not interested in users who did not perform any level check. Moreover, we will remove untrackable students: those who never did any training question or theory reading event. The justification of the cleaning decision is simply that we are not interested in users that have not performed any training or preparation for the level checks because we cannot measure their regularity (since they have not used the platform enough).

Finally, we will remove users that have not been using the educational platform sufficiently during several weeks. Thus, we remove users with less than two significant weeks since we want to have data from different weeks to apply time series techniques. In addition, a significant week is defined like above since we consider that a student can reach the minimum of five significant actions per week in the platform very easily.

Additionally, features that are not useful for our study have been removed. The removed features are the following: *'transaction_id'*, *'transaction_token'*, *'document_id'*, *'document_version'*, *'user_agent'*, *'validation'*, *'solution'*, *'type'*, *'learn_session_id'*, *'topic_id'*, *'is_closed'*, *'type_id'*, *'is_accepted'*, *'event_id'*, *'session_id'*, *'tracking_data'*, *'event_type'*. Also, we are dropping the 'evaluation' and *'input'* columns because we are not interested in whether a question is correct or not but in the result obtained in the whole level check.

Moreover, the timestamp we get from the original raw data will be transformed to a datetime object via the function *to_datetime* of the pandas library taking into account the fact that the timestamp given is in milliseconds to further extract the week number. We need to compute the week number to extract features such as the *'num_actions_per_week'* or *'num_weeks_per_user'* because we want to analyse the behaviour of the students over time.

Time series will be used to check if regular students will perform better than those who do not have regular habits when using the platform. In other words, does regularity influence a student's performance in the level checks done on the platform?

Creation of regularity features.

Our goal is to try and characterise behaviour which leads to a greater learning potential in each student. A paper, Shirvani Bouroujeni, et al,, written by the ML for Education lab at EPFL focuses on the relationship between regularity of usage and eventual attainment. It does this by studying the student's 'activity' in various ways, for example by tracking on which days of the week students perform a certain 'action'. We will apply this approach for this milestone by defining an action as a student carrying out at least one action in the events table which is associated with a transaction token in the transactions table, namely: *'CLOSE', 'CLOSE_FEEDBACK', 'GO_TO_BUG_REPORT', 'GO_TO_COMMENTS', 'GO_TO_THEORY', 'NEXT', 'OPEN_FEEDBACK', 'REQUEST_HINT', 'REVIEW_TASK', 'SHARE', 'SKIP', 'SUBMIT_ANSWER', and 'VIEW_QUESTION'*. These actions are directly related to learning activities on the platform, although it may not be necessary to use them all. We calculate the students' regularity features (except DVL) based on their activity for all weeks available in the dataset. To do this we adapt the code in the flipped classroom repository on github.

Labels for classification.

In order to build the labels for the classification, we use the data in the *events* table. Indeed, for each Dashboard navigation event (i.e. DASHBOARD_NAVIGATION), Lernnavi stores information on the user in a *tracking_data* field. This field contains the mastery scores of the user in each topic for which a level check was done. From this mastery score, one can deduce the level check score. Thus in order to get the level check result, we track the changes in mastery score among the set of Dashboard navigation events for a single user. If the change is positive, we conclude that the level check was "passed", as the change marks an improvement. In the case of a negative change, we say that the level check was "failed" and the change denotes a decrease of the user's level.

Therefore, the labels are built using a *user_id*, a *week* identifier and a boolean *validated* value expressing the level check result.

These label vectors allow us to use the user features to build our training set. The training set is then used for the classification phase. It is composed of time series samples. Each time series starts from *week* 0 and goes up to the week preceding the level check. For example, if the level check was taken on *week=12*, then the corresponding time series will be the user's data up to *week=11* included. Therefore, the training set has as many time series from a single user as the user did level checks.

Finally, the level checks are linked to specific topics and the user may have done multiple level checks on various topics in the same week. Because in this analysis we do not distinguish the topics, we must have at most one level check label per week for each user. So we aggregate the level checks that were done in the same week by assigning the majority label to the week. For example, if on *week=9*, there were 3 level checks with *validated=[0:True, 1:False, 2:False]*, we assign label *False* to *week=9*. This operation is performed under the assumption that taking the majority label rightfully ignores the potential strong topics of a user and accounts for the result of its global studying done preceding this level check.

On the other hand, the features for the classification are created from the combination of the user's features computed for the whole time series and the labels. As mentioned previously, we build a user feature vector for each user's label up to the label's *week* identifier. The feature vector is composed of weekly metrics. For each week, we compute

- the total number of actions that happened during this week,
- the one-hot encoding of the count of each action in [REVIEW_TASK, SKIP, CLOSE, SUBMIT_ANSWER, GO_TO_THEORY, NEXT, VIEW_QUESTION, GO_TO_BUG_REPORT, OPEN_FEEDBACK, CLOSE_FEEDBACK, GO_TO_COMMENTS, SHARE, REQUEST_HINT],
- the average time spent on the week's actions.

The average time is computed by taking the mean of the differences between *commit_time* and *start_time* for all the actions in the list that happened during the week. Taking the mean allows to account for actions that took a really long time significantly as well as for very short actions.

Clustering and Prediction

Following the data processing, we perform a first phase of clustering followed by a phase of classification. Let us note that even if both methods use transformed data from the same source, their inputs are different.

Clustering

As previously mentioned, the properties of each student by which we cluster is inspired from Shirvani Bouroujeni, et al in their effort to identify behaviors in the use of MooCs by EPFL students.

For each student, we have a vector of dimension 8 summarizing their regularity behavior using all metrics presented in the paper except from DVL since it is not pertinent to our situation. We then use the **KMeans** algorithm in order to cluster the data. This iterative algorithm is the simplest and fastest one. Moreover, it performs well in most situations including the case of non-visually identifiable clusters. The optimal *k* is obtained using the **Silhouette score**. This score evaluates how well distinguishable and identified the clusters are by rating their centers' individuality and preciseness. It ranges from 1 (perfectly differentiable and identified clusters) to -1 (overlapping clusters with close centers).

With this optimal *k* and the clusters' labels for each sample, we define and train a classifier for each group.

Prediction

The classification's goal is to predict if a user's activity sample (represented as a time series) validates the level check at the end of its time series or not. Thus we classify with label *1* for a passing level check and label *0* for a failed level check. Because of the constraint of variable length time series, we make use of the tslearn Library implemented on top of *scikit-learn*. This library handles variable length time series for tasks such as clustering,

regression or classification. The library offers 3 variable length time series classification methods:

- *KNeighborsTimeSeriesClassifier*: a classifier implementing the K-nearest neighbor vote for time series. It uses Dynamic Time Warping (DTW) as the distance measure. DTW handles small and medium datasets well. Additionally, this metric handles variable length time series. With this classifier, the training time is very small as no computation is performed but the training data is rather stored in order to perform the K-nearest-neighbor algorithm. Because of the nature of the algorithm, the prediction time scales as a factor of the number of samples in the training set, the number of neighbors to query and the dimension of the data.
- *TimeSeriesSVC*: a support vector classifier adapted to handle time series. The support vector technique tries to separate the data linearly in each dimension using a hyperplane. The library's implementation is based on *sklearn*'s implementation of the Support Vector Machine (SVM) algorithm. *sklearn* indicates to us that "The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples."
- *LearningShapelets*: a technique based on a nonlinear transformation of the input followed by a logistic regression. However, this method uses the Tensorflow library. Therefore we chose to not include it in the experiments.

From the previous classifiers, we used both *KNeighborsTimeSeriesClassifier* and *TimeSeriesSVC* in order to perform predictions on a clustered dataset and on the whole dataset.

For each classification task (global and clustered), we divide our whole dataset into 3 groups:

- training set (72% of the whole dataset): samples used to train the classifier.
- validation set (8% of the whole dataset): samples used to tune the hyperparameters and design the optimal classifier.
- test set (20% of the whole dataset): samples used to perform a final evaluation of the classification.

In order to create these sets, we first perform an 80-20% split to build the test set. Then a 90-10% split is performed on 80% of the data in order to create the train and validation sets.

Following the train-validation-test split, we use the *resample* method from the *sklearn* library in order to correct class imbalance in the training set. Indeed, we perform a 50% interpolation between the size of the minority class and the size of the majority one. This interpolated class size in the target size of each class after resampling them. Thus we upsample the minority class and downsample the majority one to reach equilibrium. This is important in order to not bias our classifiers e.g. in the case of one class holding 90% of the samples, a classifier always predicting this class as output would achieve 90% as training accuracy.

Therefore, we perform a total of 4 final experiments as follows:

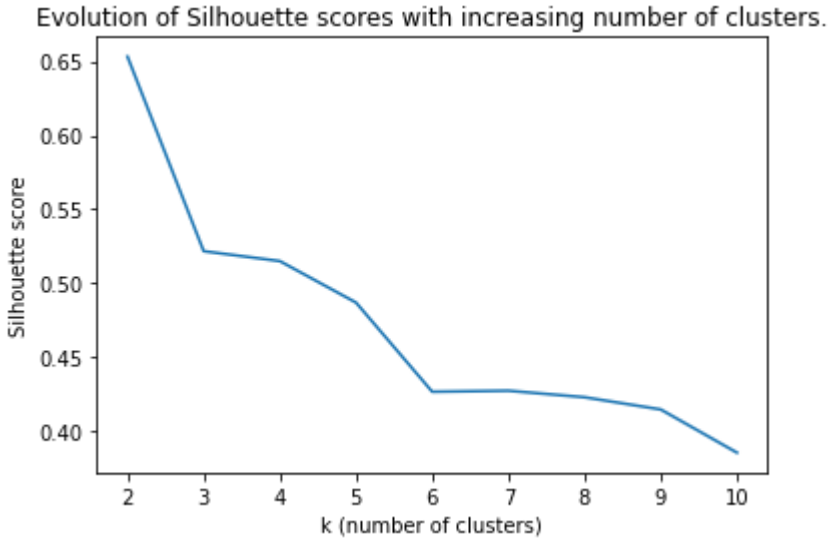
- *KNeighborsTimeSeriesClassifier*:
    - Global
    - Clustered (multiplied by the number of clusters)
- *TimeSeriesSVC*
    - Global
    - Clustered (multiplied by the number of clusters)

The next section details how these experiments were performed and what results they displayed.

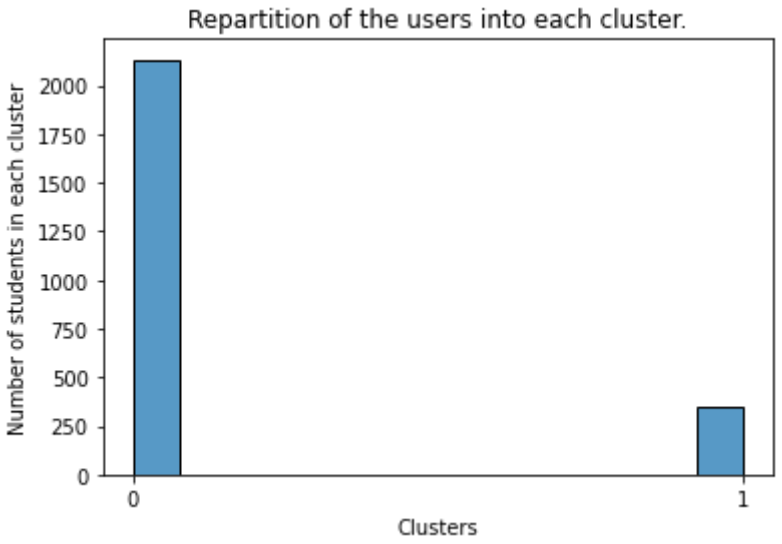
Results

As the project splits into clustering and prediction, we can analyze the results for both these parts separately. For the clustered part of the prediction, we used the *k* with the highest silhouette score as detailed in the **Clustering** subsection to create *k* clusters of our dataset.

Clustering



For the clustering we used the silhouette score as the measure of the cluster's explainability of the data. In the experiments, the whole dataset was clustered and the score was computed with each sample being attributed to a cluster. As depicted in the upper figure, we varied the number of clusters from *k=2* to *k=10*. Our experiments showed that the silhouette score was significantly decreasing from *k=2* to *k=3*. Then decreasing more slowly from *k=3* to *k=6* and then again more slowly from *k=6* to *k=10*.
However the tendency is clear and shows us that the less clusters the better the silhouette score. This result is discussed to a greater extent in the **Discussion** section. However, we choose to proceed with a version of our dataset clustered with *k=2* in order to observe if any significant improvement would come out of the clustering. One can observe the display of the clusters' balance among the data samples. We count *2'132* samples in cluster *0* and *344* samples in cluster 1.



Prediction

We can recall that the objective of the classification task is to predict if a sample (user's time series activity up to a certain week) passes the level check (end of the time series; *1* or *True*) or fails it (*0* or *False*).
In order to optimize the running time of both classifiers (*KNeighborsTimeSeriesClassifier* and *TimeSeriesSVC*), we have to find the right balance between the number of samples used to train (and to predict) and the scores of the classifier. To do this, we score the result of each classifier predicting on the validation set. The scoring metrics taken from *sklearn.metrics* as follows:

- **Accuracy**: ratio of correctly classified samples,
- **Balanced accuracy**: accuracy taking class imbalance into account,
- **Adjusted balanced accuracy**: balanced accuracy accounting for chance in classification.
- **F1 score**: harmonic mean of precision and recall (weighted average of their inverses with equal weights). Precision is the number of correctly classified true samples among predicted true samples. Recall is the number of correctly classified true samples among the total number of true samples.
- **Weighted F1 score**: F1 score weighted by each class's support,
- **Matthew's Correlation coefficient**: Matthew's binary classification correlation coefficient takes into account both true and false positives and negatives. It also handles class imbalance. It is also called the Phi coefficient. It ranges from *-1* to *1*.

Therefore, we vary the number of samples used in the training phase of the classifier, perform the binary classification using the trained classifier, and compute its scores using the metrics. We observe the following results.

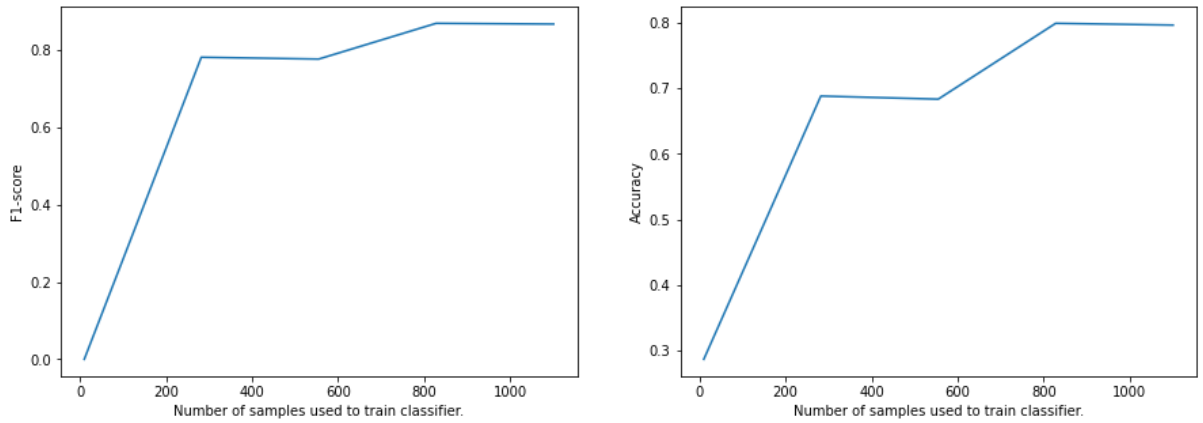


Figure 1: F-1 score and accuracy for global classification using *TimeSeriesSVC* (no clustering).

	accuracy	balanced_accuracy	adjusted_balanced_accuracy	f1	f1_weighted	matthews
10	0.287	0.500	0.000	0.000	0.128	0.000
282	0.688	0.621	0.242	0.781	0.689	0.241
555	0.683	0.620	0.240	0.776	0.685	0.237
828	0.799	0.702	0.403	0.868	0.784	0.469
1101	0.796	0.700	0.400	0.866	0.782	0.462

Table 1: All metrics' scores on the validation set for global classification using *TimeSeriesSVC* (no clustering) trained with an increasing number of samples.

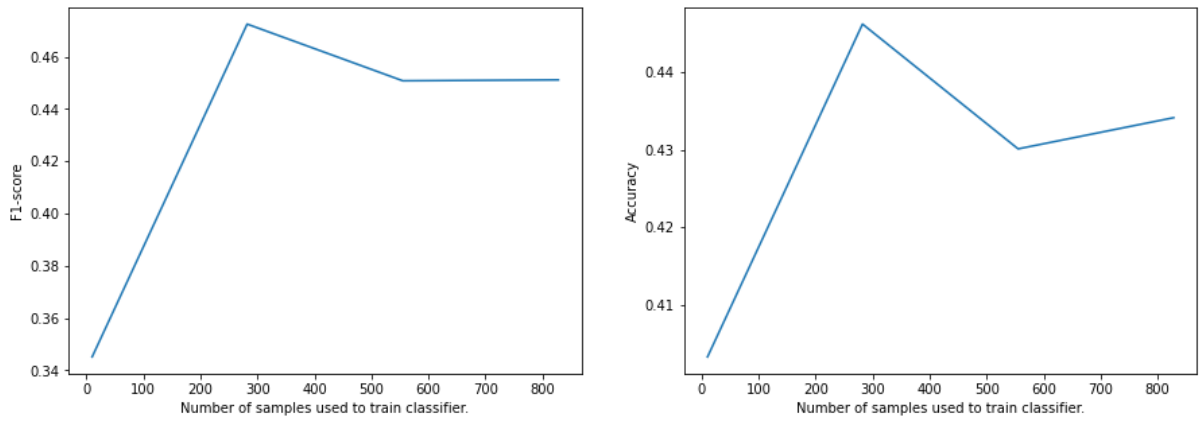


Figure 2: F-1 score and accuracy for global classification using *KNeighboursTimeSeriesClassifier* (no clustering).

	accuracy	balanced_accuracy	adjusted_balanced_accuracy	f1	f1_weighted	matthews
10	0.403	0.539	0.078	0.345	0.376	0.088
282	0.446	0.519	0.039	0.472	0.457	0.037
555	0.430	0.506	0.012	0.451	0.438	0.011
828	0.434	0.514	0.029	0.451	0.441	0.028

Table 2: All metrics' scores on the validation set for global classification using *KNeighboursTimeSeriesClassifier* (no clustering) trained with an increasing number of samples.

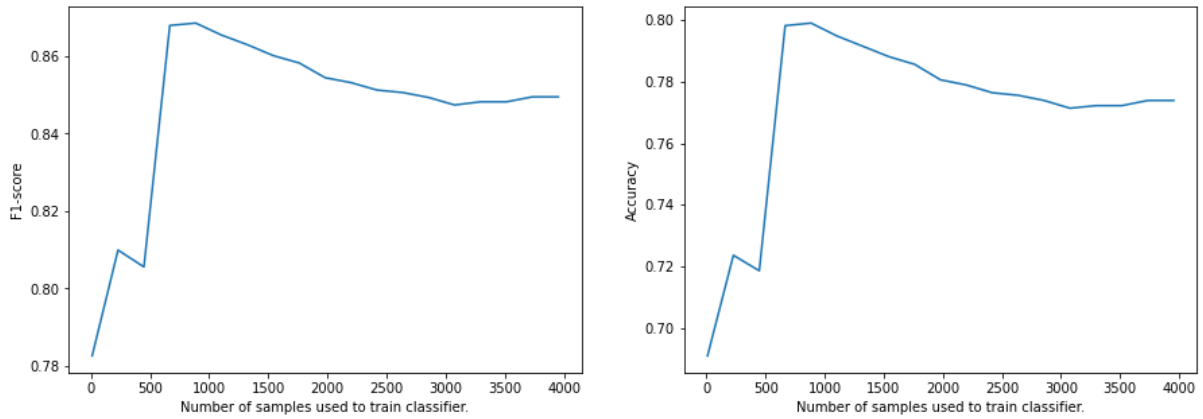


Figure 3: F1 score and accuracy for clustered classification using *TimeSeriesSVC* (Cluster 0).

	accuracy	balanced_accuracy	adjusted_balanced_accuracy	f1	f1_weighted	matthews
10	0.691	0.627	0.254	0.783	0.694	0.249
229	0.724	0.649	0.299	0.810	0.721	0.304
448	0.719	0.647	0.294	0.806	0.717	0.297
667	0.798	0.701	0.403	0.868	0.785	0.460
886	0.799	0.702	0.404	0.869	0.786	0.462
1105	0.795	0.699	0.398	0.865	0.782	0.452
1324	0.791	0.697	0.393	0.863	0.779	0.444
1543	0.788	0.696	0.392	0.860	0.777	0.437
1762	0.786	0.694	0.389	0.858	0.775	0.431
1981	0.781	0.691	0.382	0.854	0.770	0.420
2200	0.779	0.690	0.379	0.853	0.769	0.416
2419	0.776	0.688	0.376	0.851	0.767	0.410
2638	0.776	0.687	0.375	0.851	0.766	0.409
2857	0.774	0.686	0.372	0.849	0.765	0.405
3076	0.771	0.684	0.369	0.847	0.762	0.399
3295	0.772	0.684	0.368	0.848	0.763	0.400
3514	0.772	0.684	0.368	0.848	0.763	0.400
3734	0.774	0.685	0.371	0.849	0.764	0.404
3953	0.774	0.685	0.371	0.849	0.764	0.404

Table 3: All metrics' scores on the validation set for clustered classification using *TimeSeriesSVC* (Cluster 0) trained with an increasing number of samples.

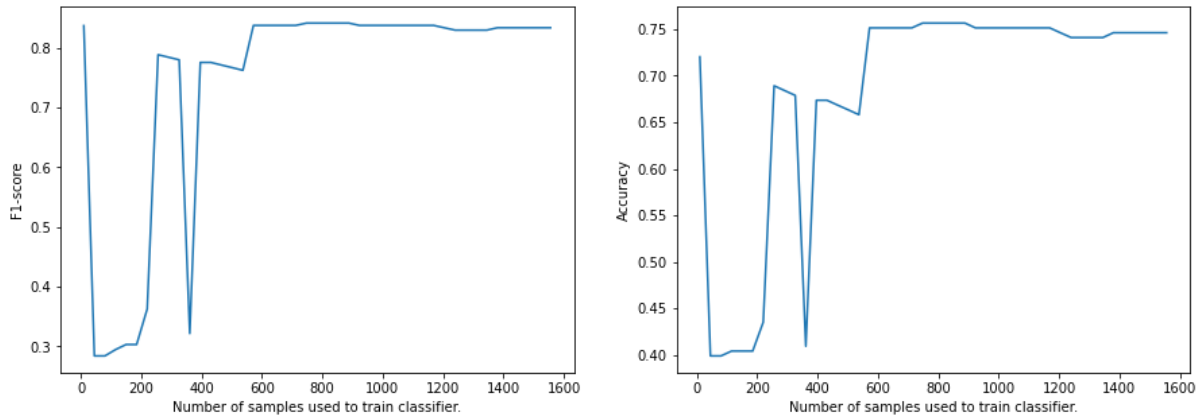


Figure 4: F1 score and accuracy for clustered classification using *TimeSeriesSVC* (Cluster 1).

	accuracy	balanced_accuracy	adjusted_balanced_accuracy	f1	f1_weighted	matthews
10	0.720	0.500	0.000	0.837	0.603	0.000
45	0.399	0.583	0.165	0.284	0.339	0.229
80	0.399	0.583	0.165	0.284	0.339	0.229
115	0.404	0.586	0.173	0.294	0.348	0.235
150	0.404	0.581	0.161	0.303	0.352	0.212
185	0.404	0.581	0.161	0.303	0.352	0.212
220	0.435	0.602	0.205	0.363	0.399	0.247
256	0.689	0.597	0.195	0.789	0.683	0.202
291	0.684	0.594	0.187	0.784	0.679	0.193
326	0.679	0.590	0.180	0.780	0.675	0.185
361	0.409	0.579	0.157	0.321	0.365	0.197
396	0.674	0.587	0.173	0.776	0.671	0.176
431	0.674	0.587	0.173	0.776	0.671	0.176
466	0.668	0.583	0.166	0.771	0.666	0.168
502	0.663	0.579	0.159	0.767	0.662	0.160
537	0.658	0.576	0.151	0.763	0.658	0.151
572	0.751	0.640	0.281	0.838	0.734	0.324
607	0.751	0.640	0.281	0.838	0.734	0.324
642	0.751	0.640	0.281	0.838	0.734	0.324
677	0.751	0.640	0.281	0.838	0.734	0.324
712	0.751	0.640	0.281	0.838	0.734	0.324
748	0.756	0.644	0.288	0.842	0.738	0.336
783	0.756	0.644	0.288	0.842	0.738	0.336
818	0.756	0.644	0.288	0.842	0.738	0.336
853	0.756	0.644	0.288	0.842	0.738	0.336
888	0.756	0.644	0.288	0.842	0.738	0.336
923	0.751	0.640	0.281	0.838	0.734	0.324
958	0.751	0.640	0.281	0.838	0.734	0.324
994	0.751	0.640	0.281	0.838	0.734	0.324
1029	0.751	0.640	0.281	0.838	0.734	0.324
1064	0.751	0.640	0.281	0.838	0.734	0.324
1099	0.751	0.640	0.281	0.838	0.734	0.324
1134	0.751	0.640	0.281	0.838	0.734	0.324
1169	0.751	0.640	0.281	0.838	0.734	0.324
1204	0.746	0.637	0.274	0.834	0.730	0.312
1240	0.741	0.633	0.267	0.830	0.725	0.301
1275	0.741	0.633	0.267	0.830	0.725	0.301
1310	0.741	0.633	0.267	0.830	0.725	0.301
1345	0.741	0.633	0.267	0.830	0.725	0.301
1380	0.746	0.637	0.274	0.834	0.730	0.312
1415	0.746	0.637	0.274	0.834	0.730	0.312
1450	0.746	0.637	0.274	0.834	0.730	0.312
1486	0.746	0.637	0.274	0.834	0.730	0.312
1521	0.746	0.637	0.274	0.834	0.730	0.312
1556	0.746	0.637	0.274	0.834	0.730	0.312

Table 4: All metrics' scores on the validation set for clustered classification using *TimeSeriesSVC* (Cluster 1) trained with an increasing number of samples.

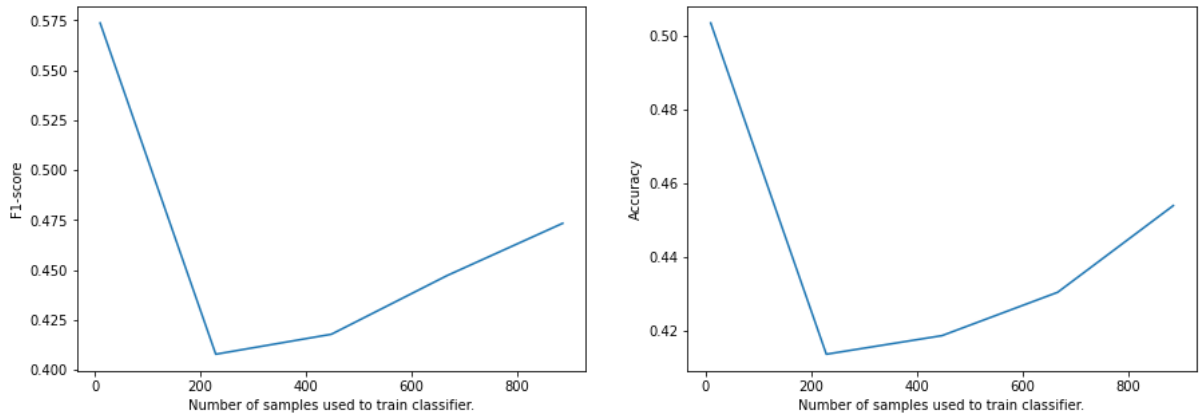


Figure 5: F1 score and accuracy for clustered classification using *KNeighboursTimeSeriesClassifier* (Cluster 0).

	accuracy	balanced_accuracy	adjusted_balanced_accuracy	f1	f1_weighted	matthews
10	0.503	0.534	0.067	0.574	0.526	0.061
229	0.414	0.518	0.036	0.408	0.411	0.036
448	0.419	0.520	0.039	0.418	0.418	0.039
667	0.430	0.517	0.034	0.447	0.438	0.033
886	0.454	0.542	0.084	0.473	0.462	0.081

Table 5: All metrics' scores on the validation set for clustered classification using *KNeighboursTimeSeriesClassifier* (Cluster 0) trained with an increasing number of samples.

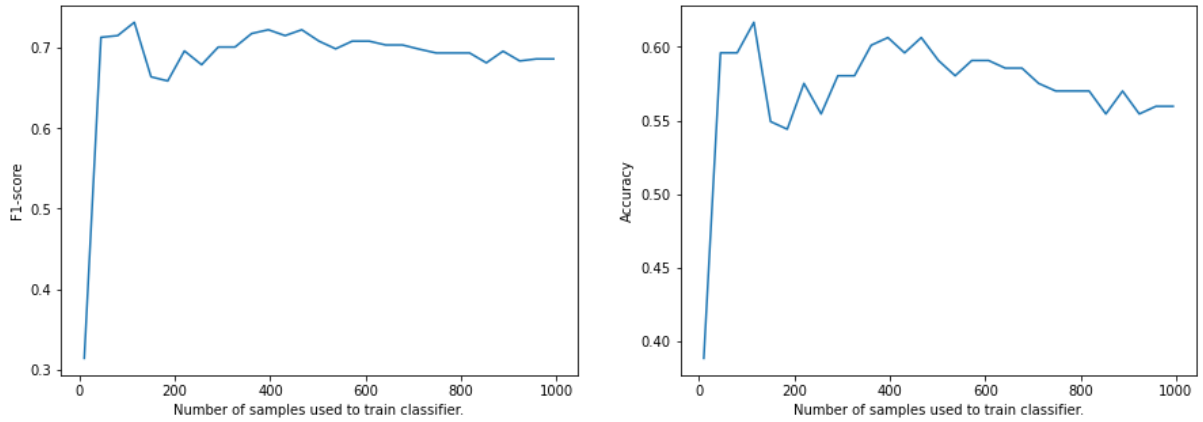


Figure 6: F1 score and accuracy for clustered classification using *KNeighboursTimeSeriesClassifier* (Cluster 1).

	accuracy	balanced_accuracy	adjusted_balanced_accuracy	f1	f1_weighted	matthews
10	0.389	0.542	0.083	0.314	0.352	0.099
45	0.596	0.516	0.031	0.713	0.602	0.030
80	0.596	0.510	0.020	0.715	0.600	0.019
115	0.617	0.530	0.060	0.732	0.619	0.059
150	0.549	0.495	-0.011	0.664	0.566	-0.010
185	0.544	0.491	-0.018	0.659	0.562	-0.017
220	0.575	0.496	-0.009	0.696	0.583	-0.009
256	0.554	0.475	-0.049	0.679	0.565	-0.047
291	0.580	0.499	-0.002	0.701	0.588	-0.002
326	0.580	0.499	-0.002	0.701	0.588	-0.002
361	0.601	0.519	0.038	0.718	0.606	0.037
396	0.606	0.523	0.046	0.723	0.610	0.045
431	0.596	0.510	0.020	0.715	0.600	0.019
466	0.606	0.523	0.046	0.723	0.610	0.045
502	0.591	0.512	0.024	0.708	0.598	0.023
537	0.580	0.505	0.010	0.699	0.589	0.009
572	0.591	0.512	0.024	0.708	0.598	0.023
607	0.591	0.512	0.024	0.708	0.598	0.023
642	0.585	0.508	0.017	0.704	0.594	0.016
677	0.585	0.508	0.017	0.704	0.594	0.016
712	0.575	0.490	-0.020	0.699	0.582	-0.020
748	0.570	0.486	-0.027	0.694	0.577	-0.026
783	0.570	0.486	-0.027	0.694	0.577	-0.026
818	0.570	0.486	-0.027	0.694	0.577	-0.026
853	0.554	0.470	-0.060	0.681	0.563	-0.058
888	0.570	0.481	-0.039	0.696	0.576	-0.038
923	0.554	0.464	-0.072	0.684	0.561	-0.070
958	0.560	0.473	-0.053	0.686	0.567	-0.051
994	0.560	0.473	-0.053	0.686	0.567	-0.051

Table 6: All metrics' scores on the validation set for clustered classification using *KNeighboursTimeSeriesClassifier* (Cluster 1) trained with an increasing number of samples.

Let us first note that due to both classifiers' complexities, the number of samples has a very significant impact on the training and evaluation time. This was the main limit to these experiments and has constrained us in the tuning of the classifiers' parameters:

- *KNeighboursTimeSeriesClassifier* :
    - *k* : number of neighbors for the vote.
    - *metric* : metric used to compute distance between samples (2 variations of DTW were relevant).
- *TimeSeriesSVC* :
    - *tol* : tolerance for stopping criterion.

Therefore, these parameters were fixed to the following for all experiments : *k=3*, *metric='dtw'* and *tol=1e-3* (default value). However, taking into consideration the observed results, we estimate that the experiments still bring interesting insights.

Then, let us also note that we choose to focus more on the accuracy and the F1-score by plotting these metrics in addition to displaying the results. Indeed, these metrics are the most commonly used due to their understandability and simplicity.

A first approach to analyzing the results is to compare the classification methods in the global (no clustering) setup. To do this, we focus on figures 1 & 2 and tables 1 & 2.
For *TimeSeriesSVC*, we can see that all scores increase significantly up to 828 samples. The best scores for all metrics are always obtained either for 828 samples or 1101 samples. For the accuracy and F1-score, we reach a maximum of **79.9% accuracy** for 828 samples (79.6% for 1101 samples) and **86.8% F1-score** for 1101 samples (86.6% for 828 samples). Finally, we see a maximum correlation of +0.469 (Matthew's coefficient) at 828 samples.
For *KNeighboursTimeSeriesClassifier*, we observe a less promising evolution. The accuracy, F1-score and weighted F1-score reach their highest scores when trained on 282 samples (**accuracy: 44.6%; F1-score: 47.2%**). However, the training set size goes up to 828 samples. The accuracy seems to oscillate between 43% and 44.6%, and F1-score oscillates between 45.1% and 47.2%. Both scores are below 50% and no clear trend can be observed in any of the metrics. Finally the maximum correlation coefficient is +0.088 which is obtained with 10 samples.

Then, by looking at the clustered experiments, we can further confirm some insights. This concerns figures 3 to 6 and tables 3 to 6.
Let us note here that in some cases (tables 3 and 4), the number of samples used to train the classifier is larger than the combined sizes of the clusters' labels. This is because the clusters are created based on the set of unique users whereas the training set contains all level checks (of which there can be more than one per user).
For *TimeSeriesSVC*, regardless of the clusters we observe that all scores increase sharply until 667 samples training-set-size (cluster 0) and 572 samples (cluster 1). Following this point, the scores seem to decrease slightly for cluster 0 and stay steady for cluster 1. This variation in cluster 0 could be interpreted as overfitting (discussed in **Discussion** section). However the highest accuracy and F1-score are confidently reached for 886 samples (**accuracy: 79.9%; F1-score: 86.8%**) for cluster 0 and 853 samples (**accuracy: 75.6%; F1-score: 84.2%**) for cluster 1. Confidence comes from the fact that cluster 0 evaluates classifiers trained up to 3'953 samples and cluster 1 up to 1'556 samples, both of which are significantly higher than the number of samples for which we observe the best scores.
For *KNeighboursTimeSeriesClassifier*, the results are very different between cluster 0 and cluster 1. Let us note that cluster 0 evaluates results only 4 times up to 886 samples, whereas cluster 1 has 29 evaluations up to 994 samples. However the trends are still significantly different.
Cluster 0 seems to feature a sharp drop of all scores (excepted balanced adjusted accuracy and correlation coefficient, which drop but recover later) when going from 10 to 229 samples. Moreover the best accuracy and F1-score are reached for 10 samples with **50.3% accuracy** and **57.4% F1-score**. Second best scores are reached for 886 samples with 45.4% accuracy and 47.3% F1-score. The only metrics that beat their initial scores (10 samples - adjusted accuracy: 6.7%; correlation: +0.061) are adjusted balanced accuracy and correlation coefficient, with 8.4% adjusted accuracy and +0.081.
Cluster 1 has a more consistent behavior, with all metrics (except adjusted accuracy and correlation) showing a sharp increase from 10 to 45 samples, and then a steadiness up to 994 samples. Best **accuracy** is reached at 115 samples with **61.7%** (second best at 396 and 466 samples with 60.6% accuracy). Best **F1-score** is also reached at 115 samples with **73.2%** (second best at 396 and 466 samples with 72.3% F1-score). Moreover, adjusted accuracy and correlation oscillate around 0 with best scores of 8.3% and 0.099 at 10 samples (worst scores of -6% at 853 samples and -0.07 at 923 samples). There is no trend

for these two metrics as negative values are observed as early as 200 samples, before going back to positive values at around 400 samples and then negative again around 750 samples.

These results lead to interpretations that we discuss in the following section.

Discussion

Our discussion focuses on the two tasks that produced results, namely the clustering and classification parts.

First, the clustering did not look very promising according to the silhouette score results. The trend was clearly showing that the less clusters, the higher the silhouette score. However clustering did not seem to significantly degrade the classification results. It even improved the best results of the *KNeighboursTimeSeriesClassifier*. Therefore, we could imagine trying out other more specific cluster evaluation methods (the Elbow technique was judged unadapted to our time series format, and visual confirmation was impossible due to the dimensionality of the data). One could think of adapting some distance metric with a threshold. Another interpretation is that the regularity metrics did not describe our data well enough for the classification task. However, the clustering based on these metrics did not hurt the classification, suggesting that the regularity metrics hold some information about our dataset, probably less relevant in the case of our task.
Then, the high imbalance between the cluster's sizes (2'132 - 344; factor ~6 difference) denotes that the majority of users have a specific regularity profile whereas a small minority has a different one. We can hypothesize that a small minority is more regular than the majority. However, the little impact of our classification results seems to indicate that regularity does not impact much a student's knowledge improvements. Another way of handling this result would be to treat the regular students as outsiders (if a reasonable interpretation is found).

Secondly, the classification results lead to some clear conclusions. Indeed, we can easily observe that the *TimeSeriesSVC* classifier seems to perform in all setups significantly better than the *KNeighboursTimeSeriesClassifier*. The former reaches up to ~80% accuracy compared to the maximum ~62% accuracy of the latter (former: ~87% F1-score; latter: ~73% F1-score). Moreover, we observe a more consistent behavior among all metrics using *TimeSeriesSVC*. At around 800 samples in the training set, it stabilizes around its optimal performances for all setups. The scores observed for 361 samples in Figure 4 and table 4 are very different from the rest of the dataset and could be part of a computation error as the rest of the scores follow a clear trend. The decreases observed in Figure 3 can be explained by an overfitting on the training data i.e. the classifier specializes so much on the training data that it performs worse on the evaluation data. Moreover this setup is the only one for which the classifier was trained on a number of samples significantly higher than 1'000. Therefore, we can suspect that this behavior will be reflected to some extent in the other setups in case of greater training set sizes. This analysis makes sense with the description of the classifier in *sklearn* (see **Clustering and Prediction**).
Finally we got further proof that the *KNeighboursTimeSeriesClassifier* is not adapted to the task. Indeed, it features very sharp variations and shows no clear trends in the global setup

and cluster 0. This must however be considered with caution as the number of evaluations are few (~4) for these setups. But even in the case of a considerable amount of evaluation points and a clearer trend as in Figure 6 and Table 6, the steadiness and overall worse performance (compared to the former classifier) indicate that this technique will not match the former's results. Even if one thinks about increasing the number of samples or varying the *k* parameter, it must be kept in mind that the running times become intractable starting from ~3500 samples (for *k*=*3*, probably worse with a greater *k* considering the algorithm's complexity). Finally, this setup features the most consistently low scores for the adjusted balanced accuracy and Matthew's correlation coefficient. The former metric's particularity is that it accounts for chance. Therefore, its score being ~0% indicates that the classification is very poorly performed. The latter metric being ~0 also tells us that the prediction is an average random one.

We conclude this discussion by acknowledging the fact that even if some further tests can be made with the models, the *TimeSeriesSVC* trained on a dataset of around 800 samples reaches a consistent classification accuracy of 80% and an F1-score of 87%. This strongly supports the hypothesis that a behavior is identifiable in the processed features that this experiment uses. However this behavior does not seem to be related to the regularity of the student. Further and more in-depth analysis is needed in order to further precise the answer to our research question.