Maching Learning Engineer Nanodegree
Capstone Project Report

Fedor Smirnov

November 26, 2017

1 Definition

1.1 Project Overview

This project addresses a problem from the education domain. Its goal is the creation of a framework that can be used by the scientific personnel of a university to evaluate the quality, i.e., the ability to convey scientific topics, of the exercises used throughout the university courses. Please refer to the capstone proposal document to find detailed information about the domain background, the related work and the data set that will be used throughout this project.

1.2 Problem Statement

The overall vision of this project is depicted in Fig. 3 of the capstone proposal document:

1. The tutor provides abstract, subjective descriptions of the problem that is treated in the evaluated exercise and of the students that this exercise is intended for.

2. In addition to this, the tutor describes the current structure of the exercise that will be used to teach the students to handle the problem at hand.

3. The trained prediction model (the Learning effectiveness prediction block in Fig. 3) takes these inputs and provides a prediction about the effectiveness of the exercise which classifies the structure of the exercise as effective or not. If the exercise is classified as ineffective, the tutor adjusts its structure and performs the next evaluation iteration.

As detailed in the capstone proposal, the problem that is addressed in this project can be divided into three subproblems.

Training the effectiveness prediction is obviously an important step during the framework creation. However, the data set that is used throughout this project describes the learning process of students that are using an online framework to learn algebra related problems. The situation in which the data set was gathered differs from the situation addressed in this project in two important aspects: **a)** the problems that the students were solving come from a different domain and **b)** the data set contains information that will not

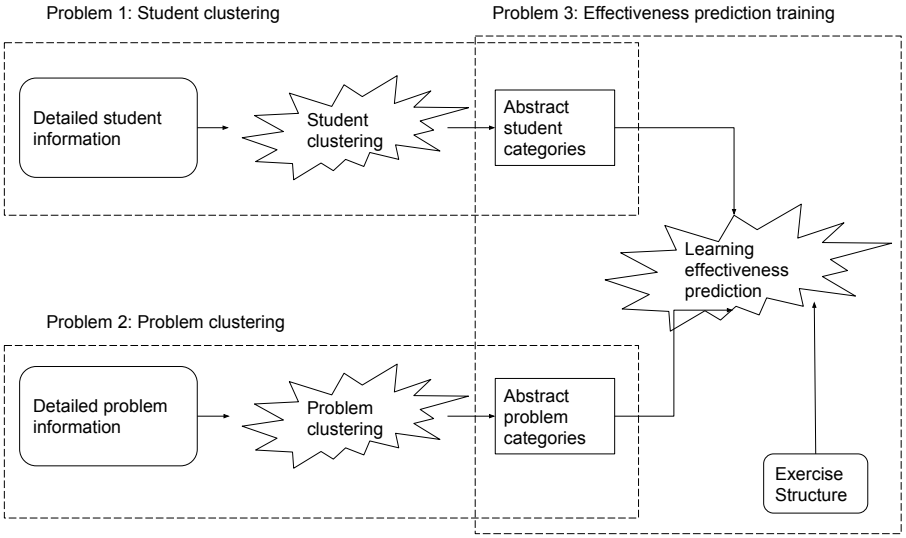


Figure 1: The overall problem can be divided into three subproblems

be accessible for the tutors of a university class, like, e.g., the exact time that was needed for solving the problem steps. Consequently, the data set can not be used for the training of the prediction model without a preprocessing step. During this preprocessing, the detailed data set entries describing the students and the problems have to be mapped onto subjective categories that can be intuitively used by the tutors. A schematic of the three subproblems is given in Fig. 1.

Following the recommendation of the reviewer of my capstone proposal, I would like to concentrate on one of the subproblems during the capstone project that I am doing in the scope of the Nanodegree and complete the other problems as a hobby project. The problem I would like to concentrate on is the clustering of the students on the abstract student categories (Problem 1 in Fig. 1).

Problem Description: The goal of the student clustering is a mapping, where the attributes from the KDD cup data set (consisting of measurements such as the time for the problem solutions) can be mapped onto more abstract student categories that a tutor would use to describe the skill level of students. In a way, the goal of the student clustering is the creation of a model that allows to transform the measurable attributes found in the KDD data set into the hidden features describing the skill levels of the students.

Problem Approach/Task Outline: To create a model capable of clustering the students onto categories describing the students skill level, the following tasks have to be carried out:

1. Data analysis / Preprocessing
2. Feature engineering
3. Clustering
4. Evaluation

Data Analysis The first step of the student clustering is an analysis of the data provided in the KDD data set. At the very start, I hereby intend to check the data for completeness to rule out the possibility that certain attributes are only present in a small subset of the data, making them useless for the clustering. In the next step, I will gather information describing the overall set (for example the number of individual students or the number of exercises processed by each student). In the final step of the data analysis, I then will check each feature in the KDD set and decide **a)** whether it is, in my opinion, likely to be relevant to describe the students skill level and **b)** whether this feature is accessible for a tutor trying to assess the skill level of his/her students or not (e.g., the exact time the students need for the exercises that were carried out in the past will not be known to the tutor designing the exercise. On the other hand, the information whether or not the students have seen a similar problem in the past is something that the tutor is likely to know). This distinction is important, as features that are not accessible for the tutor can not be part of the input provided for subproblem 3 and consequently, have to be transformed onto abstract categories while the accessible features may be directly used by the tutor as input for the learning effectiveness prediction.

Feature Engineering The goal of the feature engineering step is a transformation of the features of the KDD data set onto the hidden feature describing the skill level of the students. On the one hand, this step will reduce the number of features used as the input for the clustering algorithm, hereby hopefully preventing overfitting. On the other hand, it will create new features describing the students and give me a feeling for the overall qualities that determine the cluster the students are associated with. At this point, I intend to apply PCA and experiment with different numbers of components. The overall goal of the student clustering is to be able to work with categories that are understandable for the tutors. It is consequently very important that the new features created by the feature engineering step can be interpreted with respect to a certain student quality.

Clustering The goal of the clustering step is to take in the features provided by the feature transformation and use them to cluster the students. The clustering does not necessarily have to be done according to the skill level, but the result has to be interpretable so that the tutors have the possibility to determine the group their students can be assigned to. In this step, I intend to try different clustering algorithms and different sets of features in order and choose which combination delivers the best result. The results hereby will be evaluated based on the confidence with which the students are assigned to different groups and on how well the results can be interpreted and explained by an observable quality of the students.

Evaluation As the objective of the clustering is to assign students to a skill group, I think a good way to evaluate the results is to examine whether the membership of a student to a skill group correlates with their relative strength within the student group, i.e., their position in a list where the students are ordered based on their relative correctness.

1.3 Metrics

As described in the capstone proposal and in section 1.2, I will evaluate the results of the student clustering by comparing them to a list where the students are ordered based on the average correctness that they achieved in the problem steps they carried out.

I expect that the students that can be found in the same clusters will have a similar position on the correctness list. If the clustering results differ from the correctness list in a significant way, the reason for this difference has to be investigated.

Although I do expect a certain degree of correlation between the clustering results and the correctness list, I think that a certain degree of difference is also desirable. Otherwise, no clustering would be needed and we could simply use the average exercise correctness to classify the students.

2 Analysis

2.1 Data Exploration

Before approaching the actual clustering problem, the data set at hand (or rather the two data sets) have to be explored in order to answer several questions about the characteristics of the data set.

2.1.1 Split of the KDD data set

The KDD data set comes in the form of 2 different data sets containing data from different tutoring programs. The first set is called *bridge_to_algebra_2008_2009*, while the second set is called *algebra_2008_2009*. Both sets come within three files: the training file, the test file, and the submission file. The test and the submission files lack most of the data and are provided for the sake of the educational challenge submission. Within this project, I will only be working with the training parts of the two sets. For the sake of brevity, I will refer to the training set of the bridge_to_algebra_2008_2009 set as *Set A*, while the training set of the algebra_2008_2009 will be referred to as *Set B*.

2.1.2 Checking for missing attributes

Even a very important feature with great significance for the problem at hand can not be used for the solution of the problem if it is missing in most of the data entries. Consequently, the first step of the data exploration is to examine whether the features in the data set are missing in some of the entries.

I quantify this by calculating the so-called *feature fraction*. Hereby, a feature that is present in every entry of the entire data set has a feature fraction of 1.0, while a feature that is only present in every second entry has a feature fraction of 0.5. The feature fractions of the attributes of Set A and Set B illustrated by the Figs. 2a and 2b are calculated using the script ***find_feature_fraction.py***.

Discussion of the Feature Fractions: Having calculated the feature fractions, following observations can be made about the attributes in the two data sets:

- With more than 20 million entries, the data set A contains more than two times more entries than data set B with about 8 million entries.
- The organizational features marked cyan describing the name of the student, the name and the hierarchy of the problem, and the name of the step are present for every single entry of both data sets.
- The problem solving features marked yellow, namely the problem view, the corrects and incorrects, the number of hints, and the times for the first transaction, the correct first attempt and the step end times are given for every single entry of both data sets.
- The KC related features are marked red. Here, several important things have to be noted:
  - The KC features found in the sets are based on different KC models.

Figure 2: Feature fractions

(a) Set A (20 012 498 entries)

Attribute name	Feature Fraction
Anon Student Id	1.000
Error Step Duration (sec)	0.1383
Opportunity(SubSkills)	0.6198
Incorrects	1.000
Problem View	1.000
Corrects	1.000
First Transaction Time	1.000
Step Start Time	0.9995
Hints	1.000
KC (KTracedSkills)	0.5635
Problem Name	1.000
Problem Hierarchy	1.000
Opportunity (KTracedSkills)	0.5635
Step End Time	1.000
Correct First Attempt	1.000
Step Duration (sec)	0.9985
KC(SubSkills)	0.6198
Correct Step Duration (sec)	0.8602
Correct Transaction Time	0.9936
Step Name	1.000

(b) Set B (8 918 054 entries)

Attribute name	Feature Fraction
Anon Student Id	1.000
Error Step Duration (sec)	0.1343
Opportunity(SubSkills)	0.7223
Incorrects	1.000
Problem View	1.000
Opportunity(Rules)	0.9639
Corrects	1.000
First Transaction Time	1.000
Step Start Time	0.9702
Hints	1.000
KC (KTracedSkills)	0.4956
KC(Rules)	0.9639
Problem Name	1.000
Problem Hierarchy	1.000
Opportunity (KTracedSkills)	0.4956
Step End Time	1.000
Correct First Attempt	1.000
Step Duration (sec)	0.9503
KC(SubSkills)	0.7223
Correct Step Duration (sec)	0.8160
Correct Transaction Time	0.9733
Step Name	1.000

  - Hereby, the set B contains three (SubSkills, KTracedSkills, Rules), and the set A contains two (SubSkills, KTracedSkills) different KC models.
  - Only a fraction of the steps are mapped to the components of each of the KC models. The Rules model in the set B is the one which is present for the biggest part of the entries.

- The time features that are not present for every entry in the data set are marked green. While they are not present in every single data set, they all can be found in nearly or more than 94% of the data (the feature fractions of correct step duration and the error step duration features hereby have to be added, as these features are mutually exclusive).

Insights from the feature fraction analysis: Based on the results of the feature fraction analysis, I would like to use a subset of the set B as the data

set for the remainder of the project. This decision is motivated by multiple circumstances:

- The data set B is small enough so that I can process it on my computer without dividing it up into chunks. This also means that I do not have any restrictions for the machine learning algorithms I will be using, as I do not have to rely on their ability to train with subsets of the data set.
- Using data set B enables the consideration of the Rules model without worrying about the entries where the problem is not mapped on the KC components (the Rules - mapping is present for over 96% of the data).
- Within the chosen set all the features are present for a very big fraction of the entries so that removing entries with missing information does not come at the cost of high information loss.

For the remainder of this project, I would like to use the data set that I will refer to as the *filtered set*. The filtered set is created from the set B by removing all entries that have missing features for the KC class Rules or one of the time related features marked green in the above description of the feature fraction. The creation of this set is done by the script ***filter_set_B.py***. The feature fraction of the resulting set is illustrated in Tab.1.

Table 1: Feature fractions of the filtered set (8 035 374 entries). In this set, all attributes are present for all entries. In this set, we keep 90.10% of the data of Set B, so that the information loss is relatively small.

Attribute name	Feature Fraction
Opportunity(Rules)	1.0
Anon Student Id	1.0
Incorrects	1.0
Corrects	1.0
Problem View	1.0
Correct Transaction Time	1.0
Correct First Attempt	1.0
Step Duration (sec)	1.0
Correct Step Duration (sec)	0.87919193307
Error Step Duration (sec)	0.12080806693
Problem Name	1.0
KC(Rules)	1.0
Step Start Time	1.0
First Transaction Time	1.0
Problem Hierarchy	1.0
Hints	1.0
Step End Time	1.0
Step Name	1.0

2.1.3 Checking the number of problem steps processed by each student

Before starting to cluster the students, I examine whether each student in the data set has processed the same amount of problem steps. This is important to assure that a certain group of students does not have a disproportionately big influence on the result of the clustering. At the same time, this is the first step of a transformation of the filtered set, which is focused on the problem steps, into a data set with a focus on the students.

During the first step I create a set where each student is annotated with the number of problem steps which he/she solved. This is done by the script called ***get_problem_step_number.py***. The results can be found in the file ***student_step_ratio.txt***. The number of problem steps solved by each students can be described by the values illustrated in Tab. 2.

Investigating the number of problem steps solved by each student shows that this number varies greatly for different students. To prevent the situation where students who have processed a bigger number of problem steps gain an disproportionate influence on the clustering, some kind of normalization has to be performed.

2.1.4 Normalization of the data set

Normalizing the filtered set serves two purposes. On the one hand, it addresses the problem that different students have processed different amounts of problem steps. On the other hand, the normalization produces a set where each entry corresponds to a student, as opposed to the filtered set, where each entry corresponds to a problem step.

The filtered set is normalized using the script ***make_student_data_set.py***. The hereby created set is referred to as the *averaged set* and describes each student using the averaged values for the numerical features Incorrects, Problem View, Correct First Attempt, Step Duration (sec), Error Step Duration (sec), Correct Step Duration (sec) and Hints.

Note: The normalization that I am using can only be applied to numerical values. Consequently, the KC related features, which are represented as one-hot encoded booleans, can not be considered with this approach. While omitting these features comes with a certain information loss, it is **a)** necessary for the normalization **b)** probably better to omit these features for the sake of the overall problem, as the KC features offer a very specific description of

Table 2: Values describing the distribution of the number of problem steps solved by each student. As can be seen, this number is distributed very unequally.

Characteristic	Value
Overall number of students	3 269
Maximum number of solved problem steps	16 173
Minimum number of solved problem steps	1
Average number of solved problem steps	2 458.0526
Standard deviation	2 655.444

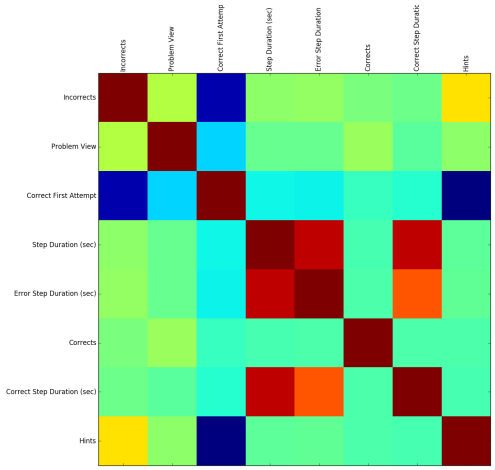


Figure 3: Correlation map of the features used in the averaged set.

the mathematical problems they describe and would probably hardly be usable for a tutor. On the other hand, the features that remain in the averaged set are all focused on the correctness of the solution and the time needed for the problem. These features are easily observable for the tutor, regardless of the subject he/she is teaching.

2.1.5 Correlation Check

Having determined the features that I want to use for the clustering, I have performed a correlation check to see whether some of the features have strong dependencies to each other. For this, I used the script ***preprocess_script.py***. The results of the correlation analysis are depicted in Fig.

Correlation Discussion: The provided image contains a heat map of the correlations between the different values. Cells with a 'warm' color signify a high positive correlation, while 'cold' colors signify negative correlation. Of course, the diagonal of the matrix has the strongest red color. Beside that, there are mainly two facts that can be seen here:

1. Correlation between the Incorrects, the Hints and the Correct First Attempt. Unsurprisingly, these three features correlate with each other. A problem step that is solved correctly on the first attempt does not contribute to the number of incorrects and requires no hints. Consequently, the feature correct first attempt has a strong negative correlation with the incorrects and the hints. On the other hand, hints are more likely to be given if the first attempts for a problem solution are incorrect. Consequently, there is a positive correlation between the hints and the number of incorrects.

2. Correlation between the step duration, the correct step duration and the error step duration. Unsurprisingly, the step duration strongly correlates

with the durations for both the correct and the incorrect steps. There is also a correlation between the time that students need for a correct and an incorrect step.

While there are some correlations in the data set, I do not think that this justifies dropping features. In both correlation situations, there are situations where dropping one of the correlating features would lead to an information loss.

3 Methodology and Results

In this section, I describe the steps that I have taken to reduce the number of features and to cluster the date generated in the previous section.

3.1 Scaling

To make sure that all features affect both the PCA and the subsequent clustering in the same way, I apply min-max scaling. This is done by the script ***rescale_before_clustering.py***. The produced set is referred to as the *rescaled set*.

3.2 Feature Reduction with PCA

3.2.1 PCA Motivation

The application of a feature reduction technique has various advantages for the problem at hand. It **a)** enables an easier (two-dimensional) visualization of the clustering results, **b)** creates hidden features that enable a better interpretation of the clustering results, and **c)** may improve the results of the clustering as the complexity is reduced with fewer features.

I have chosen PCA as the feature reduction technique. Processing the averaged set by the script ***apply_pca.py*** produces the *two principal components* set.

3.2.2 PCA Result Discussion

Number of Principal Components I have decided to use tow principal components. This is mainly motivated by the fact that I would like to visualize the clustering results and that this is much easier with two features. However, the two first principal components have a combined explained variance of 0.67, so that we lose some information through the application of PCA.

Feature Distribution - First Principal Component Figure 4 illustrates how the first principal component is calculated based on the features from the average set. As can be seen, this component is mainly affected by the features **Incorrects**, **Correct First Attempt**, and **Hints**. A student who had many incorrect attempts, rarely solved the problems with his/her first attempt, and required a lot of hints will have high values in this feature. Overall, I interpret this feature as a hidden measurement on the success that the student had while solving the problems. This feature is well suited for the usage by a tutor because

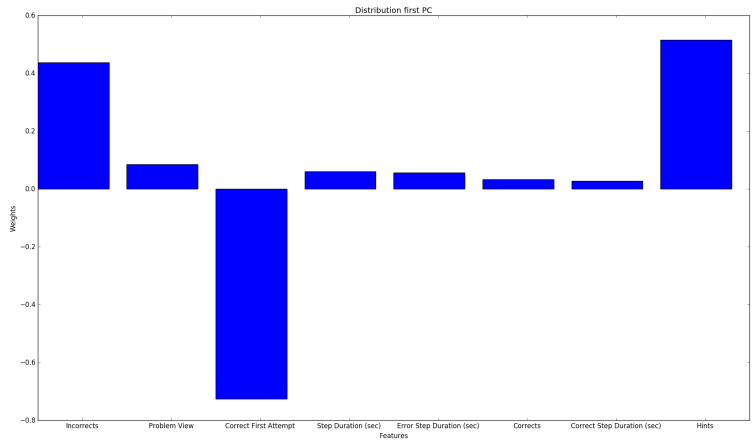


Figure 4: The first principal component is mainly affected by the features **Incorrects**, **Correct First Attempt**, and **Hints**. It, consequently, mainly describes whether the student solved the problems correctly and on his/her own.

it allows him to judge how often the students get the problems right. Students who rarely end up with the correct answer will have a high value in this feature.

Feature Distribution - Second Principal Component Figure 5 illustrates how the second principal component is calculated based on the features from the average set. As can be seen, this component is mainly affected by the features **Step Duration**, **Error Step Duration**, and **Correct Step Duration**. This hidden feature will, consequently be high for a student who needed a long time for the problems. As both the correct and the incorrect step duration are taken into account here, the main focus of this feature is on the time that was needed to provide the solution and not on the question whether the solution was correct. The second principal component is also well suited to be used by a tutor to describe the students, as it allows him/her to describe whether the students are rather fast or rather slow when solving the problem.

3.3 Clustering of the Data

There are three important decisions that have to be made before actually applying a clustering algorithm. On the one hand, the number of clusters has to be decided. For the problem at hand, the number of clusters describes the number of groups used to describe the students. On the other hand, the algorithm that will be used for the clustering has to be chosen. Finally, a benchmark has to be chosen to evaluate the clustering results.

3.3.1 Shape of the Data

The data obtained from the PCA step is illustrated in Fig. 6. Each point represents that data of one student. One noticeable quality of the data is the fact that most of the students can be found near the left lower corner of the

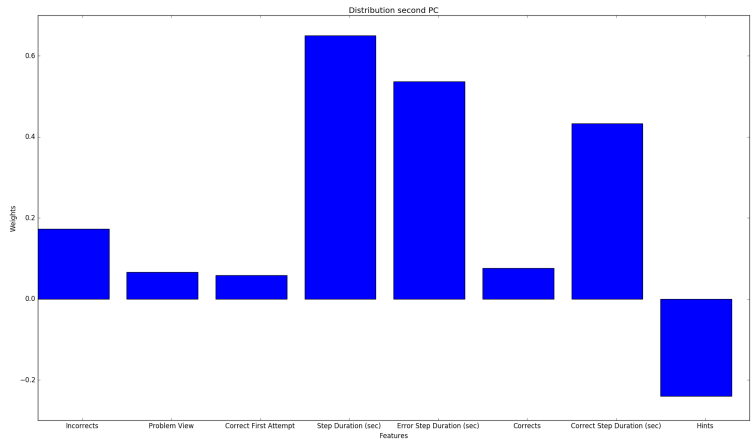


Figure 5: The second principal component is mainly affected by the features **Step Duration**, **Error Step Duration**, and **Correct Step Duration**. It, consequently, mainly describes how much time the student spent with the problem.

plot. This means that most students have rather low values in both principle components, meaning that their solutions were relatively correct and that they solved the problems in a relatively short amount of time.

3.3.2 Choosing the Number of Clusters.

All the figures introduced in this subsection were created using the script ***cluster.py***. I chose the silhouette score to evaluate the clustering when used with cluster numbers ranging from 2 to 5. I used the GMM clustering algorithm for this step. The resulting clusters are plotted in Fig. 7 while the Fig. 8 illustrates the the silhouette score for the different cluster numbers. The clustering algorithm never finds more than 3 different groups to distribute the students. In addition, the silhouette score is highest when using two clusters. Using only two

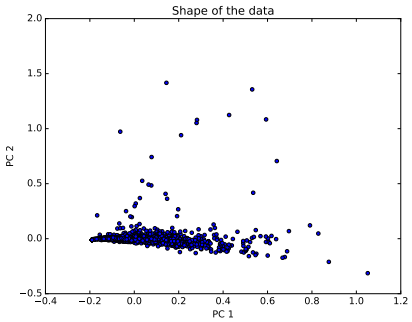
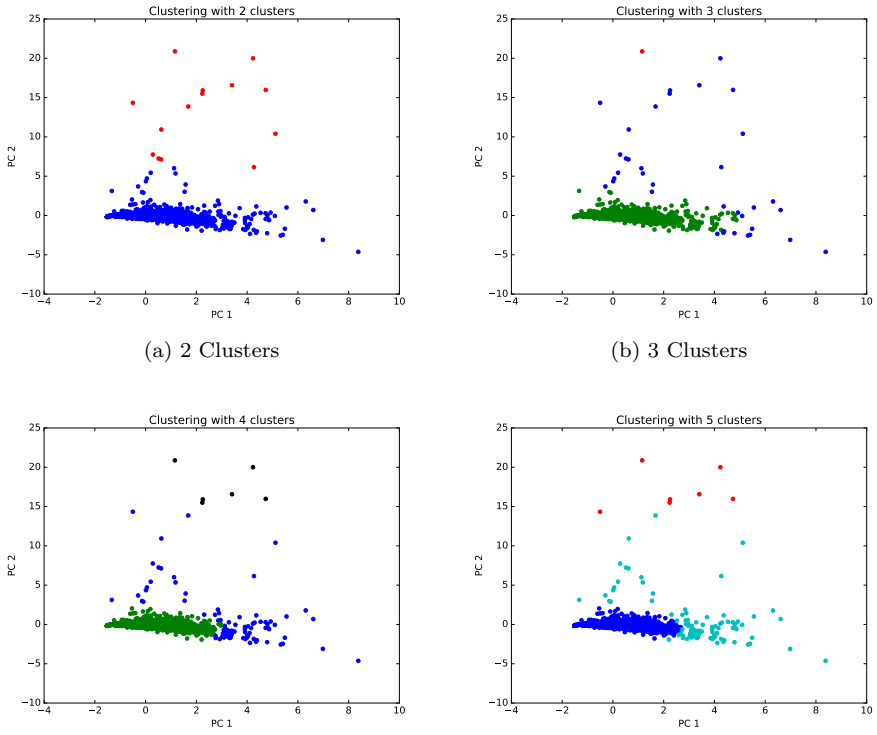


Figure 6: Shape of the data after the PCA step.

groups for the student classification also has a practical advantage, as it will be easier for the tutors to assign a student to one of two instead to one of three groups.



(a) 2 Clusters

(b) 3 Clusters

(c) 4 Clusters

(d) 5 Clusters

Figure 7: The results of the clustering for different numbers of clusters.

3.3.3 Choosing and Creating a Benchhmark.

As described in the capstone proposal, I want to evaluate the clustering by comparing the clustering result with a student clustering that is done solely based on the correctness score of the student. The bench mark is created using the script ***create_benchmark.py***. Fig. 9 illustrates the division in groups created as benchmark when using two and five groups. As I have decided to use two clusters, it is sensible to also use the benchmark with 2 groups.

3.3.4 Choosing the Clustering Algorithm.

Having decided that I want to cluster the students into 2 groups, the last step is to pick the clustering algorithm that is to be used. Fig. illustrates the results of the clustering when using different algorithms. These plots are created using the script ***comparison.py***.

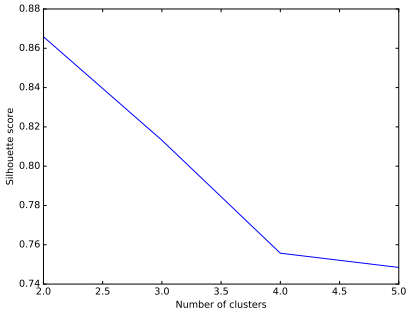
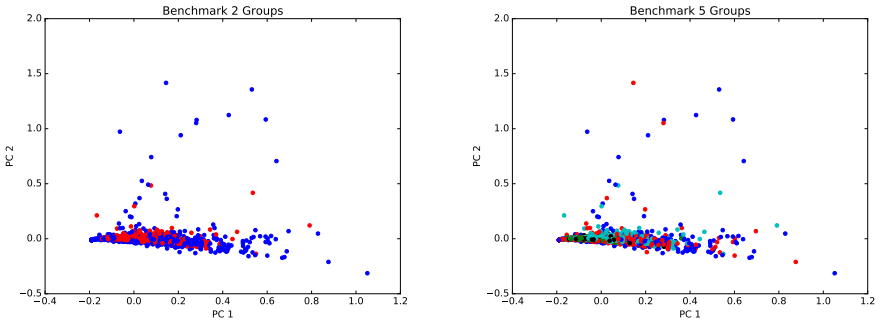


Figure 8: The silhouette score indicates that the most distinct clustering is obtained with 2 clusters.



(a) 2 Groups (split in the middle)

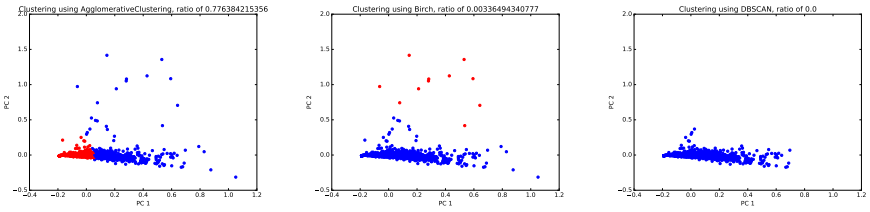
(b) 5 Groups

Figure 9: Distribution of the benchmark groups.

Clustering Algorithm Choice Discussion I think there are two criteria when choosing the clustering algorithm. On the one hand, it is important that the result of the clustering does look similar to the results of the created benchmark. One the other hand, there is also a practical aspect that must be considered. To be usable by a tutor, the grouping must divide the students in groups that at least approximately of a similar size, as it is not very useful to design an aspect of an exercise that will probably affect less than one per cent of the students. Considering these criteria, the three algorithms that are chosen for further investigation are Agglomerative Clustering, Kmeans, and Spectral Clustering. As can be observed in Fig. 10, these three algorithms provide results which resemble the benchmark. At the same time, they all divide the data into two groups, where one of the groups is approximately 4 times the size of the other. Relatively to the problem at hand, these algorithms, when trained, will try to estimate whether a student is in the top 20 % most able students.

3.3.5 Adjusting the Benchmark

As outlined in the previous section, the chosen clustering algorithms divide the data into two groups, where one of the groups is approximately 4 times larger



(a) Agglomerative Clustering (23 %) (b) Birch (0.3 %) (c) DBSCAN (0.0 %)

(d) GMM (4 %) (e) Kmeans (19 %) (f) Spectral Clustering (21 %)

Figure 10: Clustering results when using different algorithms.

than the other one. Consequently, it appears sensible to adjust the benchmark that divides the students into to groups of equal size. The benchmark used hereafter is created by assigning the top 20 % of the ordered correctness list to the first and the remaining 80 % to the second group.

3.3.6 Evaluating the Algorithms

To compare the chosen algorithms, I evaluate them against the benchmark and measure the rate of true positives, true negatives, false positives, and false negatives. The measurement is done by the script ***compare.py***. The results of this measurement are illustrated in Tab. 3. The evaluations indicate that the three algorithms perform similarly well and provide an accuracy of about 70%. Kmeans has a small disposition towards false positives, while Agglomerative Clustering rather classifies good students as bad. Spectral Clustering has nearly the same probability for false positives and for false negatives. As the performance of the three algorithms is very similar, I choose Kmeans as the algorithm for the clustering because it performs the clustering much faster than the other two.

3.3.7 Parameter Tuning

In the last step I used the script ***tune_parameters.py*** to experiment with different parameter settings of the Kmeans algorithm. However, the changing the parameters had no significant effect on the performance in relation to the benchmark. Changing the number of Kmeans runs and the maximum number of iterations in each run did not have any effect at all, so that it can be assumed that the algorithms displays a solid convergence. Changing the initialization technique had a small and mixed effect, improving some of the performance

characteristics while making other characteristics worse. All in all, tuning the parameters does not provide an improved result in comparison to the default settings.

3.4 Result Summary

As a result of the data preprocessing and a comparison of different clustering algorithms, I have created a pipeline that fits a Kmeans algorithm to the provided student data. The fitted algorithm predicts whether a student's performance will be within the top 20%. Compared with the presented benchmark, this predictor has an accuracy of 70%. The relatively low accuracy probably results from the fact that the used benchmark is not a golden model, but instead classifies based on a single feature. As the created predictor takes many more features, in particular the time for the solution of a problem, into account, it is to be expected that its results differ from the chosen benchmark.

4 Conclusion

Within this project, I have created a predictor that estimates whether a student is rather very or rather less able. During the preprocessing, I mapped the features from the original set onto two new hidden features, describing how correctly students solve the tasks as well as how much time they need to come up with a solution. Both these new features can be easily used by a tutor to describe the students in his/her class. The software can be used to create labels for the students in the data set and to use these labels to train a framework for the prediction of the effectiveness of an exercise.

Table 3: Performance comparison of the three algorithms considered for the clustering. The best value achieved in each category is highlighted with a bold font.

Performance Characteristic	Kmeans	Agglomerative	Spectral
True Positives	4.894%	**6.393%**	5.720%
True Negatives	**66.41%**	64.03%	65.00%
False Positives	15.11%	**13.61%**	14.29%
False Negatives	**13.58%**	15.97%	14.99%
Predicted as Good	18.48%	22.36%	20.71%