Maching Learning Engineer Nanodegree
Capstone Project Report

Fedor Smirnov

December 1, 2017

1 Definition

1.1 Project Overview

My name is Fedor Smirnov and I am currently pursuing a Phd degree in Computer Science at the University of Erlangen-Nuremberg in Germany. Beside their research work, all phd students who work in our department also participate in teaching activities by giving exercise courses where the students work on problems that help them to understand and learn computer science topics necessary for their studies. Although preparing these courses takes quite some time and does not directly help me with my research work, I enjoy the teaching activities very much.

One thing that bothers me about the teaching organization is that all the people responsible for the teaching (the professors and the phd students) do not have any pedagogic background or education. Nevertheless, the phd students have to come up with exercises that can be used to effectively convey the topics addressed in the courses. I myself find it very hard to estimate how well an exercise that is organized in a certain way is suited to teach concepts to the students.

As my capstone project in the machine learning nanodegree, I would like to create a framework that can be used to evaluate the effectiveness of an exercise and (by altering the input) provide hints how altering the structure of the exercise (for example the number of steps, the number of problems that has to be practiced or the number of hints that the tutor should provide for specific steps) can improve its effectiveness. When I am talking about the effectiveness of an exercise, I mean that an exercise is effective if at the end of it, the students have learned to handle the underlying problem and would be able to solve the problem correctly without requiring any help if they encounter the problem again.

Following the recommendation of my reviewer of the capstone proposal, I will be concentrating on a subproblem of the framework, namely the clustering of the students onto different performance groups.

1.2 Related Work

Searching for ways to provide more effective learning experiences for students who learn in classroom environments has been a popular research topic, espe-

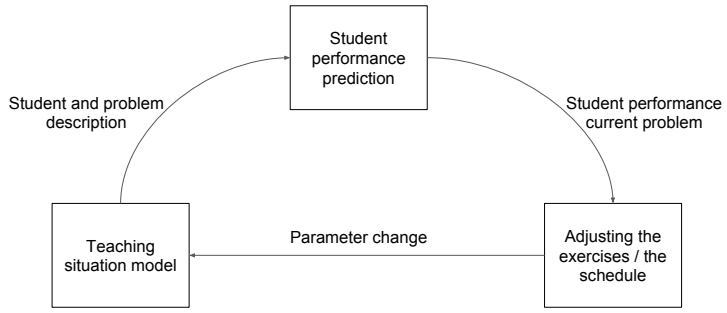


Figure 1: A model for the prediction of the student performance can be used for an iterative optimization of the teaching situation.

cially after the publication of [1]. There, the authors present evidence that, depending on the learning environment, students of the same performance level can differ in their mastery of the taught subject by as much as two Sigma, i.e., two times the standard deviation.

Subjects where the classes can be taught via virtual tutoring programs are especially interesting for this area of research, as they offer an opportunity for a relatively easy acquisition and detailed analysis of huge amounts of data. The authors of [2] and [4] show how the data sets of these problems can be used to make the learning more effective (meaning that a skill can be mastered with a smaller number of exercises) and to optimize the schedule in order to prevent under- and over-practicing.

An experimental evaluation of each possible adjustment of the teaching process would be extremely time consuming. Consequently, a model for the prediction of the student performance is a key component of approaches for the optimization of teaching systems. This model takes a description of the students and the exercise and provides an estimation of the student performance, hereby enabling the implementation of the optimization loop illustrated in Fig. 1.

Here, the effect of each change of the teaching situation, like using a different exercise structure or presenting the exercises in a different order, on the investigated objective function, like the general performance of the students or their learning rate, is estimated with the help of the performance prediction. The teaching situation can then be iteratively changed into the direction of an improving objective function.

As a basis for the student performance prediction, most state-of-the-art approaches rely on the so-called Learning Factors Analysis (LFA) model. There, the prediction of the student performance is done based on the following equation [3]:

$$\log(\frac{P_{ijt}}{1-P_{ijt}}) = \sum(\theta_i X_i) + \sum(\beta_j Y_j) + \sum(\gamma_j Y_j T_{jt}) \tag{1}$$

In this equation, the sum of the overall smarts (i.e., strength in the different

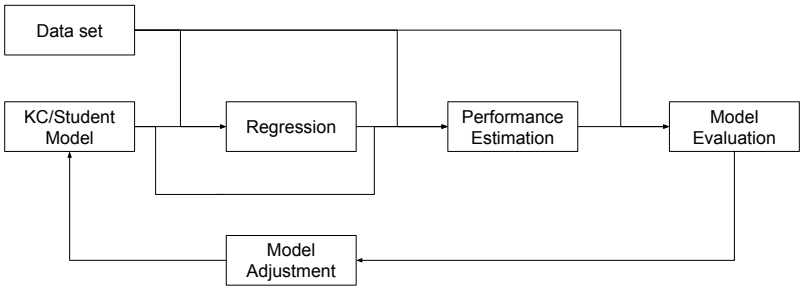


Figure 2: Iterative optimization. In the first step, a regression model is used to fit the current KC model to the data by adjusting the weighting factors for the students smarts and the KCs easiness. This fitted model is then used for the estimation of the investigated objective like, e.g., the learning rate. The difference to the objective calculated using the actual data set (the error rate of the current KC/student model) is then iteratively minimized by adjusting the KC/student model.

skill fields) of the student (θ_i), the easiness of a knowledge component (KC) and the amount of experience gained (γ_j) for each practice opportunity is used to calculate the probability that the i-th student will get a problem based on the j-th KC right when presented with the t-th opportunity to practice the KC. When applying this model, a key assumption is that the solution of each problem can be seen as an opportunity for the application of one or multiple KCs. A KC is hereby a generalization of all pieces of information that can be used for the accomplishment of a task, like concepts or skills. Identifying the KCs that the given problems are based on then becomes the main challenge for the creation of the student performance model.

Figure 2 illustrates the state-of-the-art approach that is used for the creation of a student performance model.

A KC/student that provides a small error relatively to the real data set enables a precise estimation of the student performance and can, consequently, be used for the estimation of the effect that certain changes of the teaching situation will have on the student performance.

1.3 Problem Statement

The framework that I want to create will take a description of the exercise that is being evaluated (number of steps, skill level of the students in the class, number and/or kind of knowledge components necessary to solve the problem) and classify the exercise as effective or not effective. If an exercise is effective, the students who have finished it (in the way specified by the input, for example through hints from the tutor) will be able to solve this problem correctly and without help when they encounter it the next time. Otherwise the exercise is

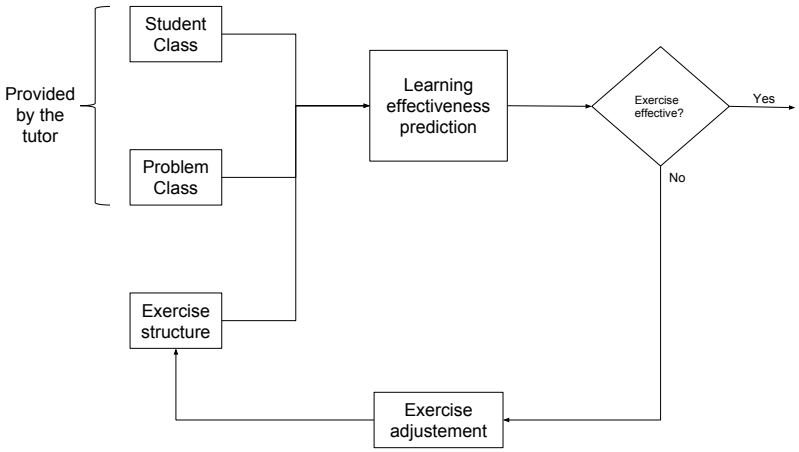


Figure 3: The tutor provides his assessment of the difficulty of the problem and the skill level of the students in the course along with the current structure of the exercise. The framework then predicts whether the exercise is effective in the current form or whether the tutor has to adjust its structure.

considered to be ineffective.

The framework that I want to create has two key differences to the state-of-the-art approach presented in the last section. On the one hand, I intend to apply a model trained with certain data (solving algebra problems) for the performance evaluation of problems from an entirely different field (computer science). On the other hand, I want this framework to be usable for tutors not having knowledge about the KC- or student models. The framework consequently has to work with inputs that can be intuitively provided by the tutors, rather than the features found in the data sets used for the training of the frameworks predictions models.

The overall vision of the framework is depicted in Fig. 3:

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

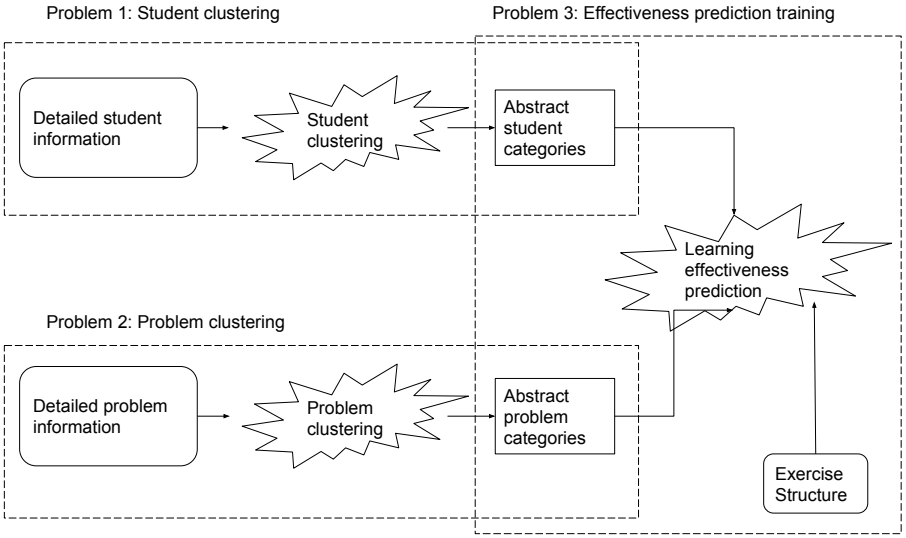


Figure 4: The overall problem can be divided into three subproblems

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 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. 4.

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. 4).

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.

1.4 Metrics

As described in section 1.3, 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. Basically, both the trained clustering algorithm and the correctness list will work similarly to a classifier, in that it will assign a label to a student indicating the performance group that the student is in. For the case of 2 performance groups, the correctness list and the clustering processing a student s can be written as $L(s)$ and $C(s)$, respectively. Hereby, the result of the correctness list will be 1 if the student has had a high correctness ($L(s) = 1$) and be 0 otherwise ($L(s) = 0$). Similarly, the result of the clustering operation will indicate whether the student is predicted to be part of the very skilled cluster ($C(s) = 1$) or not ($C(s) = 0$).

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.

Hereby, the correlation between the clustering result and the correctness list can be measured by metrics similar to those used to evaluate the result of classification algorithms. I intend to measure the rate of true positives T_p, true negatives T_n, false positives F_p, and false negatives F_n. For a student set S, these metrics are described by the following equations:

$$T_p = \frac{|L(S) = 1 \wedge C(S) = 1|}{|S|} \tag{2}$$

$$T_n = \frac{|L(S) = 0 \wedge C(S) = 0|}{|S|} \tag{3}$$

$$A = T_p + T_n \tag{4}$$

$$F_p = \frac{|L(S) = 0 \wedge C(S) = 1|}{|S|} \tag{5}$$

$$F_n = \frac{|L(S) = 1 \wedge C(S) = 0|}{|S|} \tag{6}$$

T_p, hence, describes the fraction of the students that are predicted to be very skilled and at the same time have a high average correctness. T_n is the fraction of students that are predicted to be less able and at the same time have a low average correctness. A describes something similar to the accuracy score, i.e., the fraction of students where the clustering prediction matches the students' position in the correctness list. F_p describes the probability for a student with a low average correctness to be predicted to be skilled, while F_n describes the opposite case where a student with a high average correctness is predicted to be less able.

For a promising clustering, I would expect a high (higher than 50%), yet not a perfect A-value.

2 Analysis

In this section, I first motivate the algorithms and techniques that I use for the problem addressed in this project. After this, I present an exploration of the data set used in the project.

2.1 Algorithms and Techniques Discussion

To create software for the classification of students, I want to first reduce the number of features by applying the principal components analysis technique. After this, I will use clustering algorithms to group the students into different groups. In the following, I will briefly explain the general idea of these techniques and motivate their usage in this project.

2.1.1 Principal Components Analysis (PCA)

General Idea. Principal components analysis is a technique that is frequently used for feature engineering as a preprocessing step for the application of machine learning algorithms. Hereby, the goal is to reduce the number of used features while minimizing the amount of information lost during this reduction process. PCA is performed by creating new features, the so called *principal components* (PCs) that are calculated by a weighted sum of the original features. The weighting factor of each feature hereby depends on the variance of this feature across the data set. Features with a bigger variance are likely to be weighted higher. The underlying idea is that features that are nearly the same across the entire data set (and therefore have a low variance) are unlikely to be useful during a prediction, as they can not be used to distinguish different groups within the data set.

Motivation for the Usage in this project. There are several reasons why I would like to use PCA before applying clustering algorithms:

- **Reducing the number of features**. When the amount of the available data is limited (which is the case in this project), reducing the number of the features oftentimes improves the performance of machine learning algorithms.

- **Enabling a visualization**. An important advantage of working with 3 or less features is the possibility of a visualization with three or less dimensions that is easily understandable for humans. In this project, we do not have a golden model benchmark (as stated in the analysis section, I would actually expect some deviation from the correctness list). Consequently, it is important to be able to visualize and interpret the results of the clustering.

- **Finding hidden features**. The PCs found during the PCA oftentimes represent hidden features, that is features which may by rather abstract and not directly measurable, but which at the same time represent characteristics with a high problem relevance (for example the family friendliness of a neighborhood can be a hidden feature when predicting the house prices in the neighborhood). For the problem at hand, these hidden features may be very useful for a usage by the tutors. For example, it is much easier to say whether a student is generally fast when solving problems rather than to specify the exact times he/she needed for the problems.

2.1.2 Clustering Algorithms

General Idea. Clustering describes a group of algorithms from the domain of unsupervised learning. The goal of a clustering algorithm is to identify groups within the provided data objects. To insert problem specific knowledge, clustering algorithms are typically used with a so-called *distance function*. This function takes two data objects and calculates a so-called distance between these. A high distance hereby indicates that the two data objects are not very similar. The goal of clustering algorithms is then to find groups of similar data objects, e.g., of objects with a small distance between them.

Motivation for the Usage in this project. Unsupervised learning techniques are typically used in situations where it is difficult to label the data. As in our case, the overall problem of the teaching effectiveness is rather complex, I think that it is not enough to consider just one feature of the students, like, e.g., the average correctness, during the student classification. On the other hand, I am not sure how a certain combination of the features present in the data set affects the learning abilities of the students. Clustering, consequently, seems to be a good choice, as it is likely to divide the students into groups based on measurable features. I also hope to gain additional insight into the problem by investigating the clustering results and their correlation with the features of the students.

Clustering algorithms I consider several different clustering algorithms to perform the clustering part of this project. At this point, I present a brief description for each of them and outline the differences in their training:

- **Kmeans:** Probably the best known clustering algorithm. Here, the training is initiated by placing a certain number of so-called *centroids* on random positions in the feature hyperspace. After that, the training takes place iteratively. Each iteration consists of two phases. In the first phase, each data entry is assigned to the centroid that it is most similar to. The

similarity is hereby measured by the absolute value of the used distanced function between the data entry in question and the centroid. In the second phase of an iteration, the centroids are then moved to a position where the distance to all data entries associated with this centroid is minimized (the mean of these entries - hence the name of the algorithm). After that, the next iteration starts. The algorithm is stopped after a certain number of iterations or when it converges, i.e., when the centroids are not moved any more. At this point, stable clusters are found and each centroid represents a typical entry of the cluster that it describes. After the training phase is finished, an unknown point is assigned to one of the found clusters by simply measuring its distance to the centroids. The point is then put into the cluster of the centroid to which it has the minimal distance.

- **Gaussian Mixture Models (GMM):** The GMM algorithm is somewhat similar to Kmeans in that it also works with centroids and considers the distance between the data entries and the centroids when assigning the entries to a cluster. However, GMM is built upon a different underlying assumption. In Kmeans each centroid represents the typical (in the sense of average) entry of the respective cluster and the trained algorithm simply assumes that an unknown entry is to be associated with the cluster of the centroid that it is most similar to. In GMM, however, the assumption is that all entries are stochastically generated following a certain distribution. Each centroid then represents the expected value of one of the distributions that generated the data. While the assumptions of Kmeans and GMM are quite different, the training procedure is very similar. The big difference here is that in GMM, the data entries are not fully assigned to one of the centroids. Instead, there is a certain probability that a certain entry was generated by each of the centroids. The same concept is also used when a trained GMM clusterer processes unknown entries. Instead of fully assigning them to one of the clusters, GMM provides probabilities for the association of the new point with each of the known clusters.

- **Agglomerative Clustering:** Agglomerative clustering is a member of the family of the hierarchical clustering algorithms. These algorithms operate on a tree, where the nodes represent different clusters. The root of the tree is hereby a cluster that contains all the data entries, while each leaf is a cluster consisting of a single entry. The clustering process starts at the bottom of the tree and then processes some part of the tree in an iterative bottom-up fashion. During each iteration, the two clusters with the closest resemblance to each other are merged into one cluster. This merging takes place until some stopping condition is reached, for example when the number of clusters reaches a specified value. Different implementations of this clustering approach differ in the ways to measure the distance between two clusters (e.g. the distance between the mean of the entries in the cluster or the maximum distance between two entries in the different clusters) and in the ways that the pair that is to be merged next is chosen.

- **Birch Clustering:** The Birch Algorithm has some similarities to hierarchical clustering methods in that it also works on a tree, where the position of nodes in the tree indicate their cluster. The main difference here is that,

instead of working on all the data entries, the Birch algorithm performs a lossy compression of the data by merging the data entries into so-called *subclusters*, hereby reducing the instance number.

- **Spectral Clustering:** Spectral clustering is an algorithm that uses a so-called similarity matrix. This matrix stores the similarity values for each pair of data entries. These similarities can also be represented as a graph, where the nodes are the data entries. Each node pair is then connected by an edge weighted by the similarity of the two nodes. A clustering step is then performed by cutting this graph into two parts. An optimal cut hereby minimizes the weight of the edges severed by the cut.

- **Density-Based Spatial Clustering (DBSCAN):** The DBSCAN algorithm operates by finding areas with a high density of data entries and seeing them as clusters. For this, it defines the so-called core samples. A core sample is a data entry that has at least a certain number of other data entries within a radius smaller than a defined threshold value. The algorithm finds a cluster by finding a core sample and then iteratively expanding it. In each iteration, it finds all neighbors of the current core sample that are core samples themselves and adds them to the cluster. In the next iteration, it will then repeat the same operation for each of the newly found core samples. An advantage of DBSAN when comparing it to Kmeans is that the clusters found hereby can be of any shape and do not have to be spherical.

2.2 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.

2.2.1 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*.

2.2.2 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. 5a and 5b are calculated using the script ***find_feature_fraction.py***.

Figure 5: 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

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.
  - 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.

2.2.3 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

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

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

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

disproportionate influence on the clustering, some kind of normalization has to be performed.

2.2.4 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 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.

2.2.5 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.

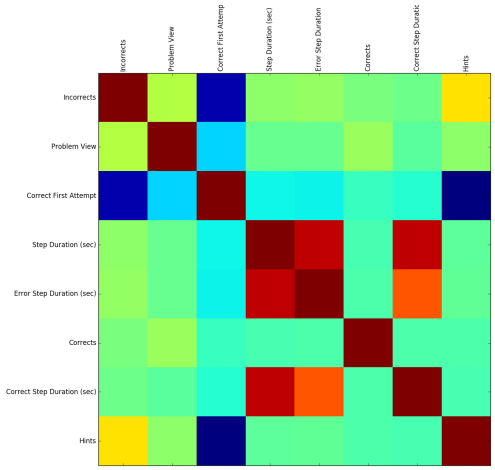


Figure 6: Correlation map of the features used in the averaged set.

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

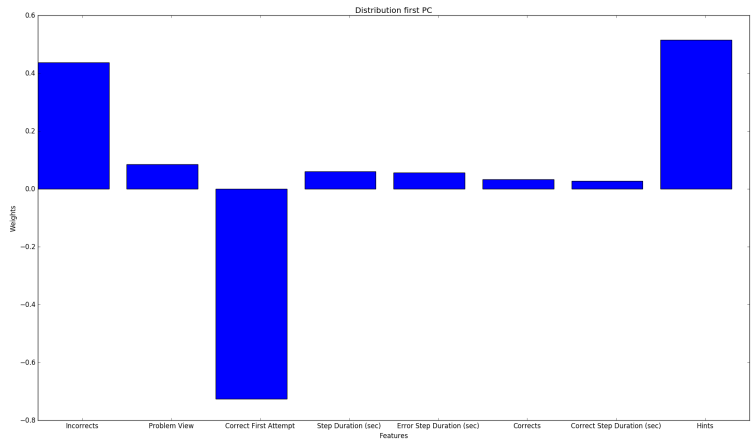


Figure 7: The first principal component is mainly affected by the features **Incorrects**, **Correct First Attempt**, and **Hints**. It, consequently, mainly describes whether the student solved the problems correctly and on his/her own.

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 7 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 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 8 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

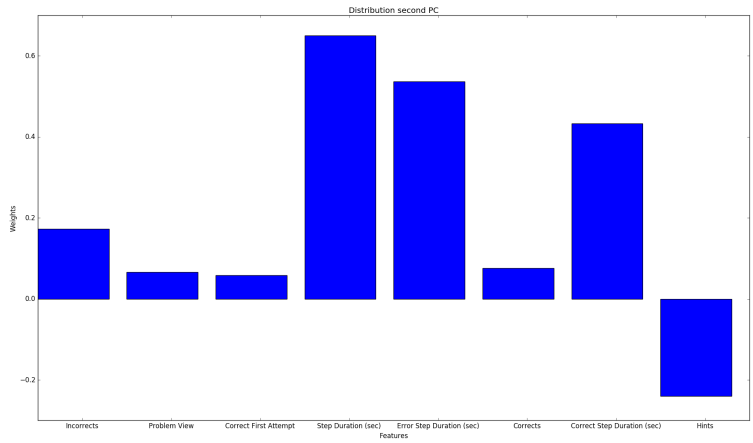


Figure 8: The second principal component is mainly affected by the features **Step Duration**, **Error Step Duration**, and **Correct Step Duration**. It, consequently, mainly describes how much time the student spent with the problem.

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. 9. 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 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.

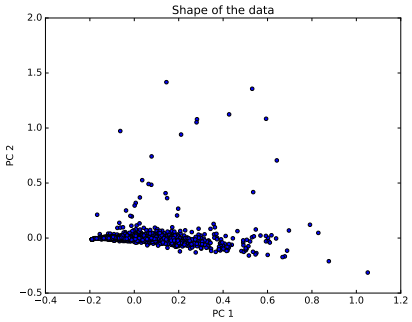


Figure 9: Shape of the data after the PCA step.

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. 10 while the Fig. 11 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 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.

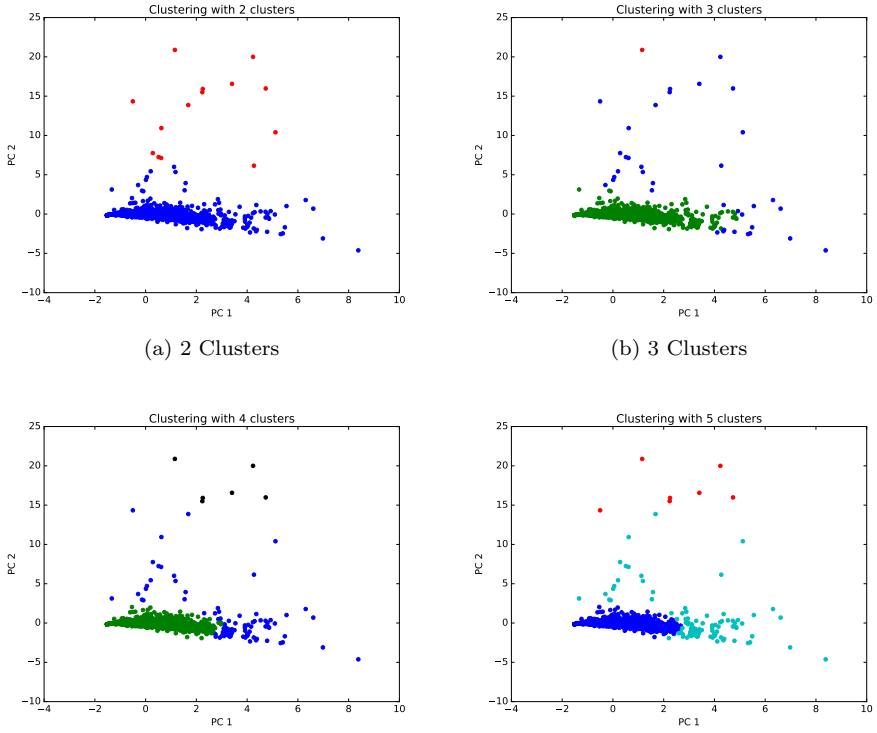
3.3.3 Choosing and Creating a Benchhmark.

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. 12 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***.

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



(a) 2 Clusters

(b) 3 Clusters

(c) 4 Clusters

(d) 5 Clusters

Figure 10: The results of the clustering for different numbers of clusters.

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. 13, 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 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 neg-

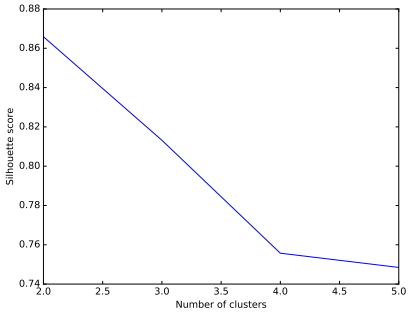
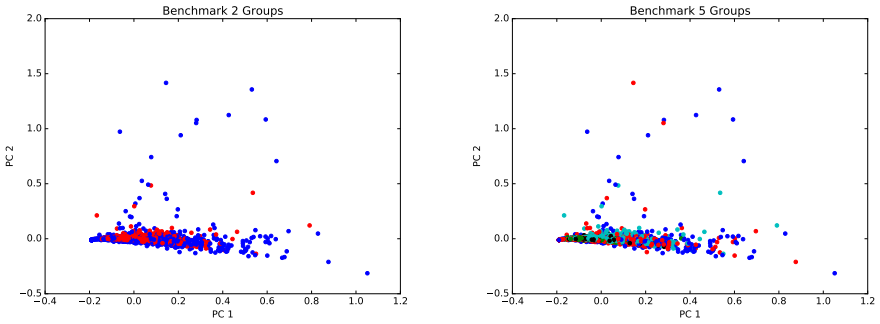


Figure 11: The silhouette score indicates that the most distinct clustering is obtained with 2 clusters.



(a) 2 Groups (split in the middle)

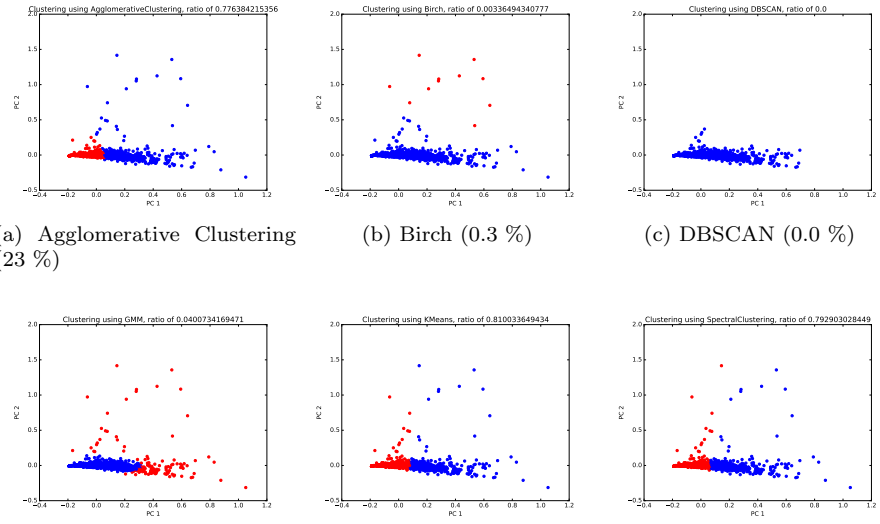
(b) 5 Groups

Figure 12: Distribution of the benchmark groups.

atives. 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



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

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

Figure 13: Clustering results when using different algorithms.

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

3.3.8 Robustness Test

To test the robustness of the created machine learning pipeline, I implemented a test, similar to those performed for the supervised classification. For this, I created a training and a test set consisting of the PC data of the students from the two principal components set. The training set is hereby created by randomly selecting 320 (roughly 10% of the original set) student entries, while the training set contains all remaining entries. For convenience, I also create the test bench set, which contains the correction based classification for the 320 students selected for the test. The creation of these data sets is done using the script ***create_data_set.py***.

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%

The robustness is then tested by training the Kmeans clusterer with the training data and then comparing the predictions the trained clusterer makes for the test data (that was not used during training) with the correction based classification of the test data. This is done by the script ***test_robustness.py***. To evaluate the robustness, I use the same metrics as for the evaluation of the algorithm itself.

The results of the robustness test can be found in Tab. 4. The test values are very similar to the values obtained when using the full training set. The robustness test, hence, seems to indicate that the model created in this project is robust when confronted with unknown data and can be used as a part of the framework for exercise design (at least if the data set that this project is working with is representative for the students and problems from the computer science area).

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% and seems to be robust in the presence of unknown data. 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.

3.5 Challenges during the Implementation

During the implementation of the preprocessing and machine pipeline described in this section, I have encountered one difficulty.

The difficulty occurred during the calculation of the accuracy values that I used to measure the correlation between the clustering and a classification based on the average correctness. While the correctness function always assigned a 1 to the students in the upper part of the list, the clustering algorithms just separated the students into two groups. The labels for the individual groups, however, were not reproducible (sometimes, the 'better' students were assigned a 1 and at other times, they were assigned a 0). This made it difficult to automatically calculate the accuracy values. After making sure that the actual clustering always provides the same results regardless of the chosen labels and

Table 4: Results of the robustness test.

Performance Characteristic	Absolute Value	Delta to Training Value
True Positives	4.375%	−0.519%
True Negatives	68.12%	+1.71%
False Positives	14.38%	+0.8%
False Negatives	13.12%	−5.36%

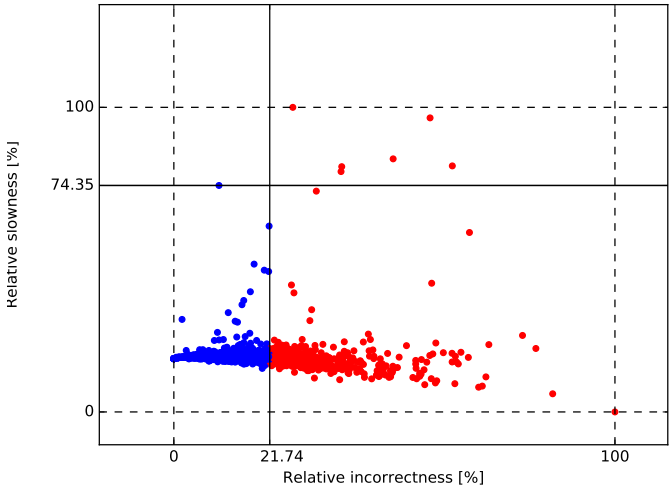


Figure 14: Visualization of the model found throughout this project. For the classification of a student as gifted or not, the relative incorrectness (the first principal component) seems to be more important.

that it always predicts roughly 20% students as gifted, I implemented a dynamic labeling, where the cluster with the bigger size (the less able students) was labeled with a 0, while the cluster with the smaller size (the gifted students) was labeled with a 1. This solved the problem.

4 Conclusion

4.1 Explanatory Visualization

For the explanatory visualization, I have decided to visualize the way that the model found throughout this project clusters the students. This visualization is created by the script ***explanatory_viz.py*** and illustrated in Fig. 14. Here, the dotted black lines highlight the maximum and the minimum values of the two principal components (which represent the incorrectness and the slowness that the students exhibited while working on the problems). The two axes are min-max scaled. The solid black lines dividing the two clusters are drawn through the data points with the maximal values for the two principal components which are still classified as gifted students. From this visualization, the first principal component seems to be more important for the separation of the two student groups. In fact, when performing a split at 21.74 % of the relative incorrectness, the second principal component could be ignored entirely without changing the result. As a summary, one can say that the found model classifies a student as less able if his/her relative correctness is worse than 22 % of the worst known incorrectness of if he/she has a slowness higher than 75 % of the maximal slowness seen in the training set.

4.2 Recap of the Learning Experience

Beside the practice that I got using the sklearn and the pandas frameworks, there were several practical insights I got through this project, where I had to apply machine learning techniques to the solution of a real problem, working with real data:

- **Evaluation:** In the other projects of the nanodegree, the evaluation metric was mostly clearly defined and unambiguous, in that a model with an optimal result in this metric was actually the best one to solve the problem. However, in the capstone project, there was no given evaluation metric and the metric that I found myself, in the best case, had a mere correlation with the quality of the model. This makes the design of the predictor more challenging, as the effect on the model's usefulness for the problem cannot be directly read from the evaluation function and each decision, consequently has to be carefully considered by, e.g., looking at the resulting visualizations. However, I do think that this is a challenge that occurs frequently, especially during the application of unsupervised learning techniques. Working on this project definitely gave me valuable experience for these situations.
- **Importance of Visualization:** When choosing the number of principal components, I ended up using just two, although they explained only 70% of the variance. However, using two PCs enabled a very comprehensive evaluation, where I could immediately see the effect of certain design decisions. Before working on this project, I would not have expected a case where a decision with worse performance values (like the explained variance) could be preferable. During the project, I definitely learned the importance of visualization.
- **Importance of Practical Aspects:** Similarly to the point with the visualization, there were several other points in the project where I accepted information loss or a decision with a worse performance metric because it was more practical for other components of the pipeline or for the actual usage in the overall framework. For example, it may have been interesting to examine how the clustering algorithms would perform without a prior PCA step. However, even in the case where the predictors trained with more features would be more accurate, they would be impractical for the tutors, as the tutors would have to provide a very big number of specific characteristics about a student to obtain a classification. With this project, I learned that, for certain decisions, practicality with respect to the underlying problem may outweigh a better performance score when designing a machine learning pipeline for a real, practical problem.

4.3 Possible Improvements

Although I think that the pipeline created during this project could already be used within a framework for exercise design, there are several opportunities for further improvement:

- **Using the entire Framework as Benchmark:** Overall, the student classification is intended to be a part of the framework for the exercise de-

sign. Consequently, each design decision should be evaluated by measuring its effect on the the classification of the teaching effect of the exercises. After the development of the other parts of the framework, the design decisions made for the student classifier should be reviewed with respect to their effect on the exercise classification.

- **Using the Problem View Feature:** I did not come up with a good way to average over the problem view feature that indicated how often a student has seen a problem. I do, however, definitely think that this feature is important for the student classification, as solving a problem correctly becomes easier when the problem was encountered before. Finding a way to average over this feature and considering the feature during the clustering may provide better results during the student clustering.

- **Using the KC features/the problem classification:** I did not use the KC features as I suspected them to be specific to the area of math problems. However, using them may provide a stronger distinction between the different problems and their difficulty. An alternative way to incorporate the difficulty of the problem would be to use the results of the problem clustering as an additional feature for the student clustering.

4.4 Project Summary

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.

References

[1] Bloom, B. S. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher 13*, 6 (1984), 4–16.

[2] Cen, H., Koedinger, K. R., and Junker, B. Is over practice necessary?-improving learning efficiency with the cognitive tutor through educational data mining. *Frontiers in Artificial Intelligence and Applications 158* (2007), 511.

[3] Draney, K. L., Pirolli, P., and Wilson, M. A measurement model for a complex cognitive skill. *Cognitively diagnostic assessment* (1995), 103–125.

[4] Stamper, J., and Koedinger, K. Human-machine student model discovery and improvement using datashop. In *Artificial intelligence in education* (2011), Springer, pp. 353–360.