Predicting and preventing student failure – using the *k*-nearest neighbour method to predict student performance in an online course environment

Tuomas Tanner[1] and Hannu Toivonen[2]

[1] Typing Master Finland Oy, Eerikinkatu 4 A 16, FI-00100 Helsinki, Finland.
tuomas.tanner@typingmaster.com

Tuomas Tanner (MSc) has a BA in English Philology and MSc in Computer Science from the University of Helsinki. His Master's thesis is based on the same research and results this paper describes. Tuomas Tanner is a software engineer at TypingMaster Finland Oy, which develops e-learning courseware to teach touch-typing. The results of this research will be incorporated into a future version of the program as an aid for teachers.

[2] Department of Computer Science and HIIT, PO Box 68, FI-00014 University of Helsinki, Finland.
hannu.toivonen@cs.helsinki.fi

Hannu Toivonen (PhD) is a Professor of Computer Science at the Department of Computer Science in the University of Helsinki. His research interests include knowledge discovery, pattern and link discovery, data mining, analysis of scientific data, search and exploration of information, with applications in various fields.

Abstract

We study the problem of predicting student performance in an online course. Our specific goal is to identify at an early stage of the course those students who have a high risk of failing. We employ the *k*-nearest neighbour method (KNN) and its many variants on this problem. We present extensive experimental results from a 12-lesson course on touch-typing, with a database of close to 15000 students. The results indicate that KNN can predict student performance accurately, and already after the very first lessons. We conclude that early tests on skills can be strong predictors for final scores also in other skill-based courses. Selected methods described in this paper will be implemented as an early warning feature for teachers of the touch-typing course, so they can quickly focus their attention to the students who need help the most.

Keywords: k-nearest neighbour; KNN; student prediction; performance assessment; e-learning; online learning; electronic learning; predictive methods; teacher aids; machine learning; data mining.

1 Introduction

Online course environments allow automatic collection of large amounts of student activity data, which has great potential value to course organizers when analysed with data mining techniques. Possible uses range from detailed student performance metrics to automatic online adaptation of the course. We study a task that falls between these extremes: helping teachers quickly find and focus their attention to the students who need help the most.

The specific target environment of our study is a tutored online course for touch-typing, consisting of 12 lessons and a final test. The goal is to predict the success or failure of students automatically based on their performances already during the first lessons; a tool could then alert teachers at an early stage to students who are at risk of failing the course or dropping out from it.

The use of computers for effective learning has been studied extensively for quite some time (see e.g. Alessi and Trollip, 1991). More recently, web-based adaptive learning systems have been developed to provide individualized learning and tutoring over the web (e.g. Brusilovsky, 1999). However, computer-based training (CBT) systems are inherently narrow and limited. Even when adaptive to the student, they lack an understanding of the student as an individual and are unsuited especially for under-performing students (Figlio, Rush, and Yin, 2010). In this paper, we seek to help teachers of online courses discover these under-performing students in need of assistance as early as possible. To this end, we propose and extensively evaluate a *k*-nearest neighbour (KNN) method to predict student performance at early stages of study.

We develop and experimentally evaluate a wide range of different settings and variants of the KNN method. We report experimental results on thousands of students of a commercial online touch-typing course. As a real and large-scale experiment, it is probably interesting also to other developers of online learning environments. While course contents and the difficulty of student performance prediction certainly vary a lot between courses, the KNN technique (Cover and Hart, 1967) applied here generalizes well. We review the method in some detail to make the treatment here self-contained.

The rest of the paper is organized as follows. In Section 2 we describe our target online learning environment and student performance data available for mining, and we define the student failure prediction problem. In Section 3 we discuss related work done for the student prediction problem. Section 4 defines the *k*-nearest neighbour method, its current uses and more advanced variants of the method that are used in this study. Section 5 is devoted to results and Section 6 for discussion. We conclude in Section 7.

2 The student failure prediction problem

The problem that we want to solve is the following: can we predict, with reasonable accuracy, the performance of students at the end of the course given their study results at early phases of the course? More specifically: can we predict which students will drop out without completing the final test, and also predict the students who will complete the final test, but fail the set specified pass requirements?

2.1 The touch typing course

The target application is an online course environment for teaching touch-typing[1]. The course environment is available for educational institutions and businesses that wish to train students and staff to use the touch-typing technique. When a school or company purchases a license for the service, an account is created for them where students enrol and teachers can manage and view students' study progress.

Students study the touch-typing technique for 12 lessons. New keys are introduced at each lesson and the student studies them through key drills, games and typing exercises. Each lesson ends with a graded typing test that evaluates the student's current typing ability. Table 1 details the features collected at each lesson. At the end of the course, the student has learned all lowercase and uppercase keys of the keyboard and the number keys. The course ends with a final test that measures the overall (net) typing speed of the student and determines the score for the whole course.

The goal is to use the lesson features in Table 1 to make reasonably accurate predictions about students' final test performance already after 25%–30% of the course has been completed (the first four lessons of a total of 12 in our target environment). This is required in order to provide teachers with effective early indication of students that need tutoring.

Student status features (for each lesson)	
Lesson status	Status of the lesson: not attempted, incomplete, passed, failed (if teacher set requirements not met)
Student performance features (for each lesson)	
Gross speed	Typing speed of the student (in keystrokes per minute; before any errors are deducted)
Accuracy	Accuracy percentage: One mistyped word equals 5 error keystrokes by default.
Net speed	The typing speed (in keystrokes per minute) where 5 keystrokes for each mistyped word is removed from the gross speed
Lesson time	Time student spent studying during this lesson.
Exam attempts	The number of times the lesson exam has been taken.
Lesson requirement features	
Required net speed	A minimum net-speed requirement that can be set by the teacher. This must be met before the lesson is marked passed. If this is not set, the lesson is marked passed as soon as the lesson exam is completed.
Required accuracy	A minimum accuracy percentage that can be set by the teacher.

[1]TypingMaster Online, http://www.typingmaster.com/education/online/

Customer demographic features	
Customer type	Company vs. school
Account size	The number of students in an account
License type	Different license types a school or company can purchase. Possible values are: user, transferable user, concurrent user, workstation, site.
License size	Number of licenses purchased

Table 1: Features available for each of the 12 lessons.

2.2 Student data

Students. The course database currently contains 428 licensed accounts (i.e., customer organizations) and 14,875 students that can be used as training data. Out of the students, 7604 have successfully completed the course and 7271 have failed.

In order to avoid misclassifying students who are in the middle of their studies as failed, only students who have not been active during the last three months are used for training data. Additionally, the failed students are only chosen from groups that also have successful classmates in order to filter out groups where studying has never actually begun or has been aborted.

Features. As students advance through the course, their status is recorded for each of the 12 lessons. Each lesson ends with a lesson exam that tests students' speed and accuracy for the new keys learned during that lesson. The performance is described in total by eight features, i.e., attributes (Table 1). Of these, one gives the overall status of a student (*lesson status*) and five describe actual student performance (*gross speed, accuracy, net speed, lesson time, exam attempts*), two are pass limits set by the teacher (*required net speed, required accuracy*). The performance features are mutually redundant: net speed is a function of gross speed and accuracy (see Table 1 for details). The final test has similar features to the 12 lessons with the most important ones being status (pass/fail) and the net speed.

The final results of a student are defined as follows. An admissible final test result is defined as having a duration longer than 50 seconds and net speed greater than 20 keystrokes per minute. Results that do not meet these criteria are deemed as simply browsed or cancelled tests, and a student without an admissible final test result is considered failed. For students considered passed, the main measure of success is their net speed.

The TypingMaster Online database consists of many different kinds of accounts such as elementary schools, universities, small businesses, large corporations and libraries. These different organizations have very different students age- and skills-wise and differentiating these different demographic groups could affect prediction accuracy. However, the service does not request detailed demographic information for students or classes. Because of the limitations, we are only able to divide the accounts into two groups: educational institutions and businesses or libraries. In our study, we mainly use the student status and student performance features to predict student failure.

3 Related work

Several studies have analysed student performance in educational institutions. These studies have mainly focused on classifying students into two categories – either pass or fail for a given course. In a representative study, Kotsiantis, Pierrakeas, and Pintelas (2003) use multiple machine learning techniques to classify university students into dropouts and non-dropouts. The study shows that it is possible to classify the dropout-prone students by using only students' demographic data. The precision of the classification was adequate initially and increased when the students acquired curriculum data during the school year. They tested different algorithms and found that the Naive Bayes algorithm achieved the best results, but differences between methods were small. The F_1 measures for all classifiers, including a KNN classifier using K=3 neighbours, were roughly 60% using only demographic data about the students. With a half year's worth of study results, the F_1 measures reached 80% for all classifiers.

Minaei-Bidgoli et al. (2003) conducted a similar study to ours where they tried to predict the final test grades for students enrolled in a web-based course. In this study, three different classifications for the students' results were used: dividing results into two classes (pass and fail), three classes (high, middle and low), or into 9 classes, according to their grade. Several learning algorithms were compared: decision trees, neural networks, naïve Bayes, logistic regression, support vector machines, and KNN with feature weights adjusted by a genetic algorithm. The study concluded that each of the machine learning algorithms performed satisfactorily with all performing roughly equally.

As in these previous studies, our study will include the classification of students into those who will "drop out" and those who will complete the final test. Additionally, we use regression to predict the actual scores of the students' final test. To our knowledge, classification and regression functions have not been combined before in the educational context.

A slightly different educational prediction problem is described in a study by Shih and Lee (2001). This study uses KNN to predict which level of study material is shown to students depending on their abilities and performance. The study proposes creating and categorizing many versions of each lesson for multiple levels of expertise. The nearest neighbour algorithm is then used to choose the most suitable level of lesson for each student to match his or her current performance.

Although the k-nearest neighbour method is simplistic and does not rely on any assumptions about the prior probabilities of the training data, the performance of this method in real-world classification tasks has been shown to be satisfactory (Manning, Raghavan, and Schütze, 2008). Its performance when compared to other classifiers is quite good considering the simplicity of the method (Daelemans and Van den Bosch, 1992; Dumais et al., 1998; Li and Yang, 2003; Islam et al., 2007).

In machine learning and student performance prediction, the accuracy of KNN has been found competitive with more complex methods such as support vector machines, support patterns and kernel methods (Han and Lam, 2006; Zhanga, and Zhou, 2007). Especially when extended with genetic learning of features and distance weights, its performance is close to and in some cases surpasses competing methods (Ishii, et al., 2009; Minaei-Bidgoli et al., 2003). KNN is especially well suited in situations where predictions need to be made from noisy and incomplete data (Zou, An, and Huang, 2005).

The simplicity of KNN and its robustness to noisy training data make it a good candidate for classification tasks in many situations.

4 K-nearest neighbour method

The *k*-nearest neighbour method (KNN) is a classical, widely used prediction method in the fields of machine learning and data mining. This is mostly due to its simplicity and versatility in predicting many different types of data. It was first introduced in 1951 by E. Fix and J. L. Hodges in their unpublished report for the US Air Force School of Aviation Medicine. In 1967 Cover and Hart (1967) formalized the original idea and discovered the main properties of this method. KNN is an instance-based, or lazy, method since it does not require building a model to represent the underlying statistics and distributions of the original training data. Instead, it works directly on the actual instances of the training data.

This section provides an overview of current uses of the KNN method and how it compares to other machine learning methods. The KNN method is defined and its extensions are detailed. Furthermore, our evaluation methodology of the results is outlined.

4.1 The KNN method and its extensions

Since the baseline KNN method is well known, its explanation here is brief. Cover and Hart (1967) describe the KNN method more formally and also find the upper error bound of the method to be twice that of Bayes' error probability. The following sections introduce the baseline KNN method and discuss in detail extensions to KNN, which are used in our study to improve classification performance.

The baseline KNN makes a prediction on the performance of a given student as follows. The algorithm first calculates the test subject's (student being predicted) similarity to all instances in the training set and finds the k most similar ones. Similarity is calculated with a simple Euclidean distance between the features of the test subject and corresponding features of each instant in the training set.

A simple example illustrating KNN with two features (typing accuracy and gross speed) is presented in Figure 1. In this example a test subject (square) with the first lesson completed is compared to students in the training set (triangle, diamond, cross). The two nearest students (diamond being closest followed by triangle) have both passed the final test. Therefore the test subject is likely to also pass the course.

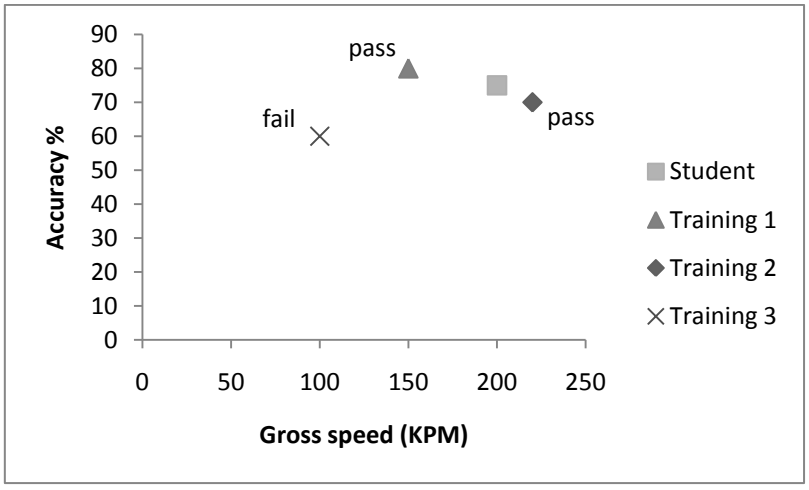

Figure 1: KNN prediction example

Once the *k* training instances with the shortest distance are found (i.e. the most similar students), these are used to decide the predicted class of the test subject – in our case failure or success in the final test. This is decided with a majority vote. In Figure 1, the vote with three neighbours ($k = 3$) is "pass" with a 2 to 1 majority. In prediction, the required majority threshold is tuned for best results. As the threshold increases, fewer students are classified as failing. This produces fewer false alarms, but also misses more failed students.

KNN is also used for regression analysis to predict the test subject's final score instead of classifying success or failure. In the baseline regression analysis the *k* nearest neighbours' admissible test results are averaged for the predicted score. In our application, we discard any neighbours that do not have results for the current lesson being tested (i.e. students who quit before getting as far as the student we are predicting for).

To reduce the chance of over-fitting due to similar pass limits and the same teacher's influence we take the following additional measure. When computing the nearest neighbours of a particular test student, we discard all neighbours that belong to the same account and group as the test subject.

Feature standardization

The most common extension to KNN is the use of standardized features for distance calculation. Since the features have greatly varying value ranges, using them directly in distance metrics would effectively give more weight to features with larger values. For instance, even a dozen of *lesson attempts* would be insignificant compared to *gross speed* with values reaching up to hundreds. Z-score standardization is commonly used in data analysis to avoid such problems and to give all features roughly equal weight. The z-score (or standard score) z_i is calculated by

$$z_i = \frac{x_i - \mu}{\sigma},$$

where x_i is the original value of the feature for student *i*, μ is the mean value of the feature, and σ is the standard deviation of the feature. This transformation scales the values so that the mean is 0 and variance is 1.

The categorical *lesson status* feature was converted to the following values: failed: -0.5, not attempted: 0.0, incomplete: 0.5, passed: 1.5. The chosen values are comparable to z-score values and are arranged so that they reflect the "distances" between the different categories.

The results of the final test are left in their original form as these are not used for neighbour distance calculation. Instead, the original exam results of the found neighbours are used to calculate the category (pass or fail) and regression score.

Distance-weighted KNN

Distance-weighted KNN extends the baseline by taking advantage of distance information of the found neighbours. Dudani (1976) first proposed a weight that increases when the distance between the test sample and the found neighbour decreases, i.e. more weight is given to nearer neighbours. He defines the weight w_j of the jth nearest neighbour as

$$w_j = \begin{cases} \frac{d_k - d_j}{d_k - d_1}, d_k \neq d_1, \\ 1, d_k = d_1 \end{cases},$$

where d_j is the distance between the test sample *x* and its jth nearest neighbour $x(j), j = 1, \dots, k$.

Our implementation of the weighted decision method works by first calculating a weight for each of the neighbours found using Dudani's (1976) method. If a neighbour has failed to complete the final test, the weight is positive (i.e. that neighbour votes for failure). If the neighbour completed the final test successfully, the weight is set as negative (i.e. the vote is against failure). Finally, all votes from

the neighbours are added together and scaled according to the neighbour count. If the vote is 0%, then the weighted vote is a tie. When the weighted vote is positive, it signifies that the weighted majority of neighbours consider the student to be a failure.

For KNN regression, in turn, Altman (1992) introduced the following distance weighted method. First find the *k* neighbours nearest to a sample. Then calculate the estimated value $\hat{y}$ by weighting the nearest *k* values by their inverse squared Euclidean distance to the sample:

$$\hat{y} = \frac{\sum_{i=1}^{k} (1/d_i^2) y_i}{\sum_{i=1}^{k} (1/d_i^2)},$$

where y_i is the value of the ith nearest neighbour, and d_i is the distance from the sample to the ith nearest neighbour. According to Altman, this weighting reduces bias in the estimator without increasing its variance much. We will consider this weighting for KNN regression.

Feature weighting

In contrast to distance weighting, which favours certain neighbours based on their distance, feature weighting gives preference to more important features to improve distance calculation. The extreme case of feature weighting is just a binary choice of inclusion or exclusion of a feature.

We use the results of a related study by Minaei-Bidgoli et al. (2003) as a starting point for feature weighting. In their study, the lesson score was clearly found to be the most important feature. The second most important feature was the number of attempts, while other features provided less information to the prediction. We will use domain knowledge to locate the best performing sets of weighted features. We will also evaluate the importance of each feature by removing them one at a time and retesting the classifier to gauge its effect.

In our implementation, each used feature is assigned a weight that is used to multiply the feature component of the Euclidean distance between two students. If the feature weight is set to 2, the distance component of that feature (and also its effect on the whole distance calculation) is doubled. If the feature weight is set to 0.5, the feature's distance component is halved accordingly. To find the best combined weighted classifier, we opted to first test each feature weight in isolation and then find the combinations that yield the best results.

4.2 Evaluation methodology

We now describe how student prediction results are evaluated for our KNN classifier. The evaluation is based on calculating predictions for existing training data, i.e. students whose status and test results are already known. After a prediction the result can then be contrasted to the student's actual results.

All our evaluations are carried out using leave-one-out cross-validation. This is an extreme form of *k*-fold cross-validation, where *k* is the size of the training set itself. In other words, each student *i* from the training set is temporarily used for prediction at lesson *j*. Prediction performance is assessed by comparing the predicted final test results to actual results of student *i*.

The primary goal is to identify students that have a high risk of failing the course. Recall and precision are a widely used pair of measures for success in such tasks:

$$\text{recall} = \frac{\text{number of correct positive predictions}}{\text{number of positive examples}},$$

$$\text{precision} = \frac{\text{number of correct positive predictions}}{\text{number of positive predictions}},$$

where "positive" means a student who failed and should be recognized by the prediction method. In other words, recall tells how many of those students that really failed were among those the teacher

was alerted to, while precision tells how many of the alerts were correct. The F_1 measure then combines recall and precision into a single overall effectiveness score (van Rijsbergen, 1979):

$$F_1 = \frac{2*\text{recall}*\text{precision}}{(\text{recall}+\text{precision})}.$$

The secondary goal is to predict the net speed of students who pass the course. The error rates of such numeric regression tasks are commonly measured with the root mean squared error (RMSE). The RMSE uses the same distance metric as the actual estimator making results easy to judge. The RMSE is defined as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}},$$

where $\hat{y}_i$ is the estimated value of an observation, y_i is its actual value and n is the number of observations. The RMSE is calculated only for those students who have a valid final test score.

5 Results

We next evaluate the performance of the KNN method in predicting student failure or the final test scores.

5.1 Goals and setup for the experiments

The prediction task consists of two different but connected questions: "Will this particular student fail the course and the final test" and "If this student completes the final test, what will his or her test result be". The aim of our experiments is to study if these prediction tasks can be completed with reasonable accuracy, and how different variants and parameters affect the prediction performance.

The variants, parameters and their values are given in Table 2. The default settings are indicated in boldface. The default parameters of the KNN method were selected during the test process as robust alternatives for this application and data (see next subsection).

Parameter/variant	**Values (default values in boldface)**
Parameters of the problem	
Prediction task	Failure vs. success OR
	Final test score (for successful students)
Number of lessons studied	1 – 12 (4)
Parameters of KNN	
k, number of neighbours	5 – 300 (30, **70**, 150)
Failed neighbour threshold	10% – 90 % (20%, **35%**)
Distance weighting	Not used/used
Parameters on features (see Table 1)	

Set of features used	**"All features"** (except demographic features) OR "5 features": lesson status, gross speed, accuracy, net speed, lesson time OR "3 features": lesson status, net speed, lesson time, OR Net speed
Feature weights	Constant for all, OR "5 weighted features": net speed (3.2), lesson status (1.4), exam attempts (0.9), lesson time (0.8), customer type (0.4), all other features (0)
Demographic features	Not used/used

Table 2: Parameters and their values

5.2 Failure prediction

5.2.1 The baseline classifier

First, initial tests were conducted to find suitable values for the two KNN parameters: failed neighbour percentage and neighbour count. These tests were done for the whole ranges of values for both parameters (Table 2). A robust failed neighbour threshold was found at 35% with roughly the optimum F_1 value (Figure 2). We will use 20% as an additional experimental value; it can pick more failed students (since it has much higher recall), but it will also pick false positives (it has a lower precision).

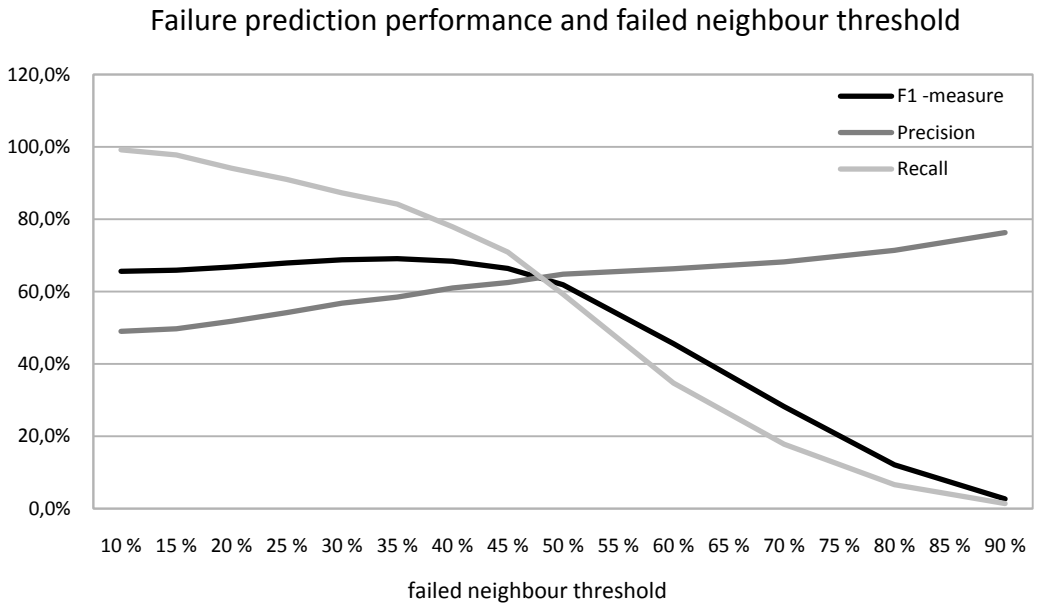

Figure 2: Classification performance and failed neighbour percentage

Figure 2 shows the F_1, precision and recall measures at different neighbour failure percentages. Expectedly, a higher failed neighbour threshold increases precision but causes fewer students to be

categorized as failed, decreasing the recall measure. Even at 90% failed neighbour threshold, the precision measure only reaches 76%. This means that even with the strictest conditions, it is impossible to avoid a significant number of false positives. With a lower required percentage of failed neighbours, more students are categorized as drop-outs and the recall percentage is higher. However, this also causes the number of false positives to increase lowering the precision measure.

Finding a suitable best neighbour count *k* proved more difficult, however. Figure 3 shows how F_1 measures markedly improve when the number of neighbours increases up to 30. From 30 neighbours onwards, the increase in F_1 measures is much slower, and it eventually seems to converge to 69%.

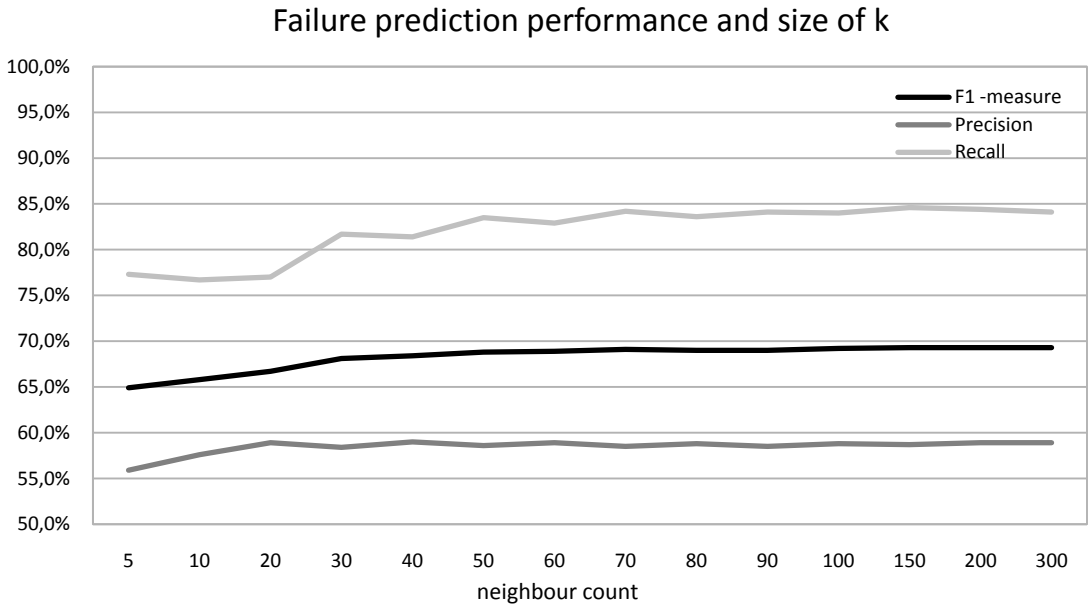

Figure 3: Classification performance and neighbour count

While precision seems to converge at about 20 neighbours, recall has no such clear point. We chose 70 neighbours as a good compromise. At 70 neighbours and 35% failed neighbours, the F_1 measure is 69% with 59% precision and 84% recall. We additionally chose 30 and 150 neighbours as test cases for lower and higher neighbour counts, respectively.

5.2.2 Extensions to the classifier

We now consider a number of modifications or extensions to the basic KNN method. We start by testing the effect of using **different sets of features** for prediction. At first, no feature weighting is done, features are simply included or excluded. We considered a decreasing series of feature sets (see Table 2): a 5-feature set, a 3-feature set, and net speed alone. In all these experiments, the F_1 measure is essentially the same as when using all features (Table 3). Surprisingly, net speed alone performed well, and marginally even outperformed larger sets of features.

Classifier	Features	Neighbours	Best F_1 measure	Precision	Recall
Different feature sets					
Standard	All features	70	69.1%	58.5%	84.2%
Standard	5 features	70	69.0%	58.7%	83.7%
Standard	3 features	70	68.9%	58.0%	84.8%
Standard	Net speed	70	69.3%	58.4%	85.1%
Standard	All features	30	68.4%	57.1%	81.7%
Standard	3 features	30	68.5%	58.0%	83.8%
Standard	Net speed	30	68.6%	57.8%	84.3%
Standard	All features	150	69.3%	58.7%	84.6%
Standard	Net speed	150	69.4%	58.7%	84.9%
Addition of demographic information					
Standard	Net speed + demographic	70	69.1%	58.1%	85.2%
Standard	3 features + demographic	30	68.9%	58.3%	84.3%
Distance-weighted KNN					
Distance weighted	Net speed	150	69.3%	57.3%	87.8%
Distance weighted	3 features	150	69.4%	56.5%	88.3%
Weighted features					
Standard	5 weighted features	70	69.3%	58.5%	84.8%
Standard	5 weighted features	30	69.0%	57.5%	86.3%

Table 3: Summary of best F_1 measures at lesson four and 35% failed neighbours, and precision and recall at the best F_1 measure.

Next, we consider adding **demographic information** to the classifier. The baseline KNN classifier treated students from all accounts in the online course environment as equal. The expectation was that study in an educational environment is more structured than in a business or library. The results with demographic information were again essentially identical to the previous results. Additional tests on the 30-neighbour case using the school/company feature together with the 3-feature set resulted in an F_1 measure (69%) that was slightly improved over results with 30 neighbours but without demographic features. This result indicates that the additional demographic feature could have some impact with lower neighbour counts. However, currently the limited demographic data available also limits its effectiveness for prediction.

The next extension, use of **distance-weighted neighbours**, did not improve results when compared to the single feature unweighted classification. Actually, using all features with distance weighting produces inferior results to unweighted prediction. A test with the 3-feature set with distance weighting did improve prediction results over unweighted results in some settings, but never improved on the results reached with unweighted single net-speed feature prediction.

The final extension to the baseline KNN method is the addition of **feature weights** to the distance calculation. The best performing set of features and weights we found was: school/company 0.4, lesson status 1.4, net speed 3.2, lesson time 0.8, attempts 0.9. While the results were nearly identical

to just using the single net-speed feature for neighbour distance calculation, feature weighting proved more successful at early lessons 1–3. These results will be discussed below.

5.2.3 Course progress and classifier accuracy

Let us next see how prediction performance changes during different stages of the course. It would be natural to expect that the accuracy should increase as we get more and more data from the students. The results are quite different, however (Figure 4).

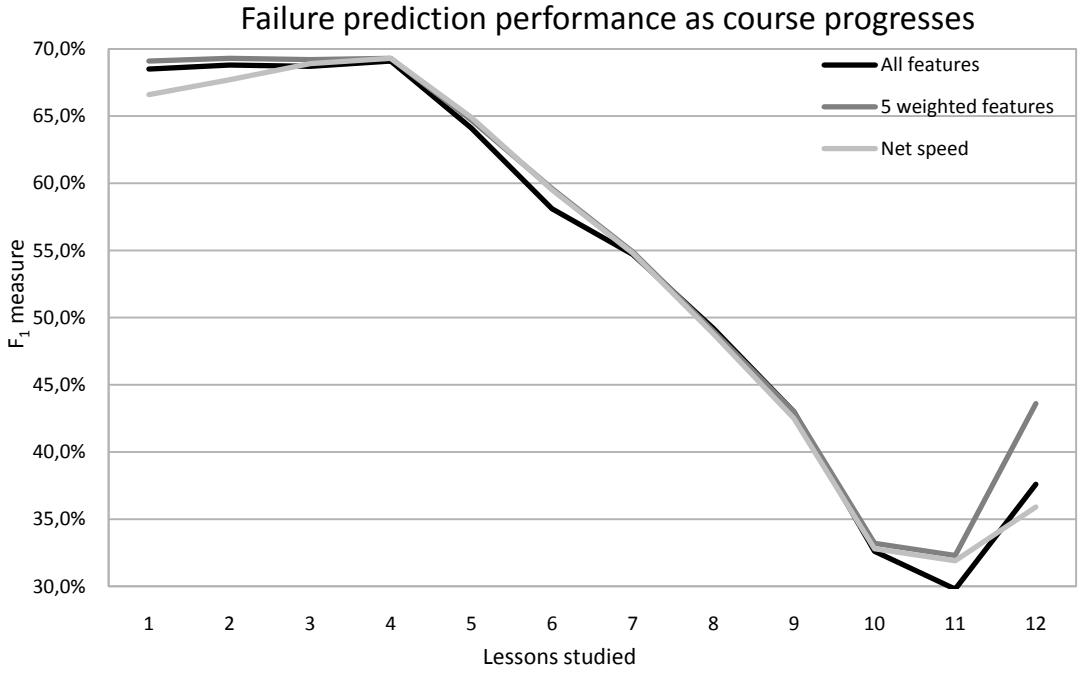

Figure 4: Prediction performance by lessons studied

It seems that the prediction performance slightly improves from lessons one to four. However, after lesson 4, the prediction performance steadily decreases towards the end of the course. The reason for this is that a large number of students drop out after lesson 4 (over 1600 students). As the course advances, the number of failed students decreases steadily. Since the relative number of students who fail during the remaining course becomes lower, the performance of the KNN method naturally suffers – there are simply not enough neighbours of the correct type to produce successful predictions. This also explains why the best observed failed neighbour threshold is 20% instead of 35% from lesson 7 onwards (results not shown).

Out of the 3 different feature sets tested here (all features, five weighted features, net speed; Figure 4), the weighted feature set performed the best. At early and late lessons, it was clearly the best performing feature set and also matched the baseline single net-speed feature at lessons 4-10.

5.3 Regression analysis of the final test score

We now move on to the regression problem of predicting final scores for those students who pass the course.

5.3.1 Prediction after the fourth lecture

Again, we start by looking for suitable key parameter values: neighbour count and feature set. We experimentally considered the following three feature sets: all features, 3 features and the single net-speed feature.

Let us first, however, examine the difference between the distance-weighted and unweighted variants of KNN. Figure 5 shows the difference as a function of the number *k* of neighbours; values below zero indicate that the distance-weighted variant performed better. It is interesting to note how distance weighting works best when the number of features and neighbours is large. When using all features for prediction, distance weighting starts outperforming the unweighted variant already at 10 neighbours. On the other hand, when only the net-speed feature is used, distance weighting is always inferior to the unweighted method.

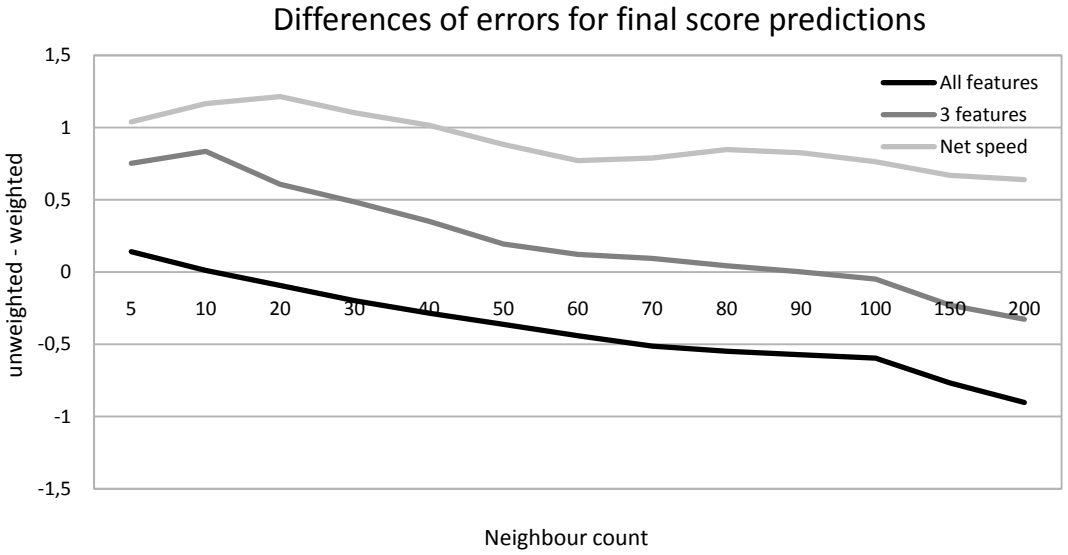

Figure 5: RMSE difference between standard and distance-weighted methods

At low neighbour counts, distance weighting loses because it does not use all the available information effectively. In turn, as the neighbour count increases, the noise from far-away neighbours degrades the performance of the unweighted method. This effect is further amplified by the noise from additional features.

Using distance-weighted neighbours and some additional features does ultimately produce slightly better results than using just the single net-speed feature (Figure 6). However, the improvement is negligible in our application. The standard, unweighted version with the 3-feature set is a robust alternative, with essentially equally good performance between 40 and 100 neighbours, and the best performance for lower numbers of neighbours. Use of demographic features was not useful (Table 4). We choose to use the 3-feature set and 70 neighbours as default values with the unweighted variant, or 200 neighbours with the distance-weighted variant.

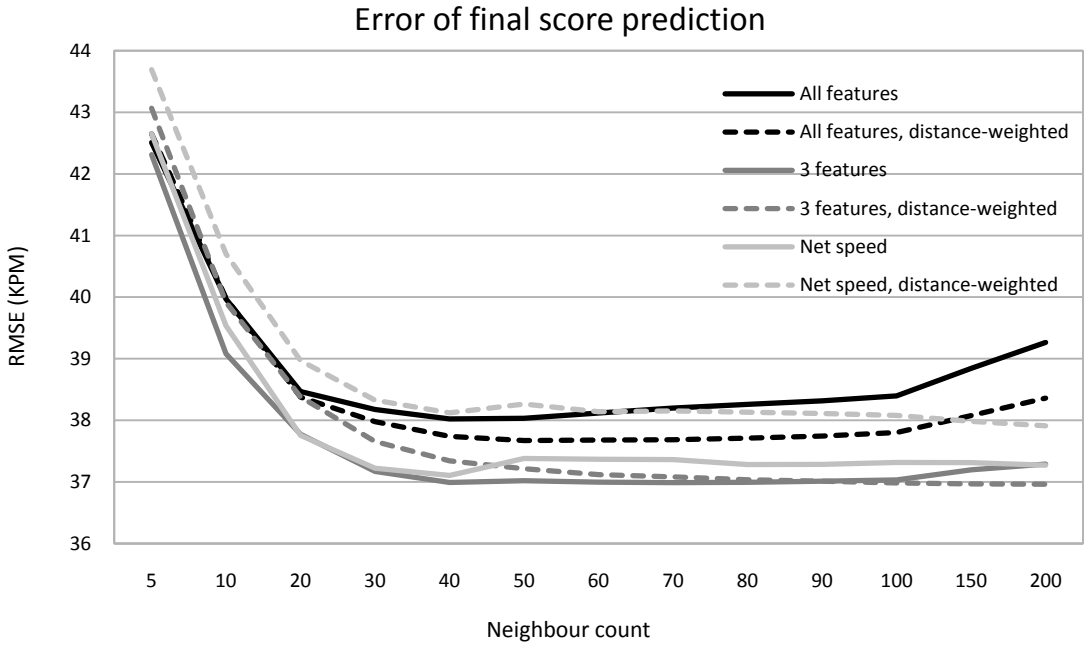

Figure 6: Regression errors at lesson 4

Regression type	**Features**	**Neighbours**	**RMSE**
Standard	All features	40	38.02
Distance weighted	All features	50	37.67
Standard	Net speed	40	37.11
Distance weighted	Net speed	200	37.91
Standard	3 features	70	36.99
Distance weighted	3 features	200	36.96
Standard	3 features + demographic	70	37.63
Distance weighted	3 features + demographic	200	37.00

Table 4: Summary of best achieved RMSEs by regression type at lesson four

5.3.2 Regression performance during course progress

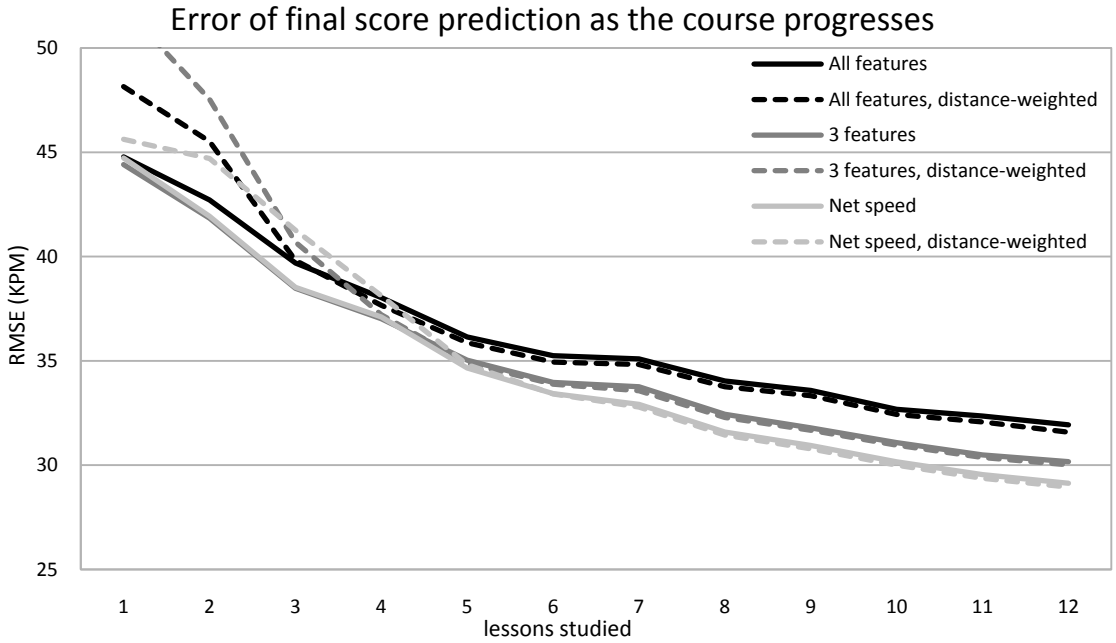

Figure 7: Regression performance RMSE and course progress

Looking at the regression performance after the very first lesson (Figure 7), all tested feature sets perform roughly equally without distance weighting. However, from lesson 2 onwards regression performance with all features is poorer. During the later lessons the single feature case performs the best with a clear difference to the three-feature case. This in turn outperforms predictions with all features. For the single-feature case, the RMSE steadily decreases from 35 at lesson 5 to 29 at the last lesson 12. Naturally, prediction of final test scores is more useful – and more difficult – at an early stage.

At lessons one to four, distance weighting causes the RMSE to be 4–6 keystrokes per minute higher than the unweighted results. From lesson 5 on, the performance of weighted versus unweighted variants was practically equal, and the choice of features has a much larger effect on the RMSE than the use of weighting. This observation largely coincides with results from the previous subsection: relative performance of the distance-weighted variant improves as more data (lessons) become available.

In summary, the single net-speed feature performs the best overall. Although the three-feature case is equally accurate in early lessons, its performance degrades as the course progresses. We believe this result is due to the noise introduced by features that are not so important. The total number of features is a multiple of the number of lessons. As the lesson count grows, the less important features start adding noise to the prediction.

5.4 An analysis of predicted failures

We next analyse false positives, i.e., cases where neighbour calculation has predicted failure but the student has completed the final test. Predictions for final scores turn out to be correlated with course failure in a useful way. Within students that are predicted to fail, the distributions of predicted final scores are somewhat different for the groups of students who eventually succeeded or failed the course (both classification and regression predicted after lesson four using the net-speed feature

only). In our standard setting using 35% failed neighbour percentage (Figure 8, solid lines), there are roughly twice as many failed students among those that had low predicted final scores (appr. 20-80 KPM). On the other hand, the distributions are – surprisingly – quite similar for students with high predicted final scores (120 KPM or more).

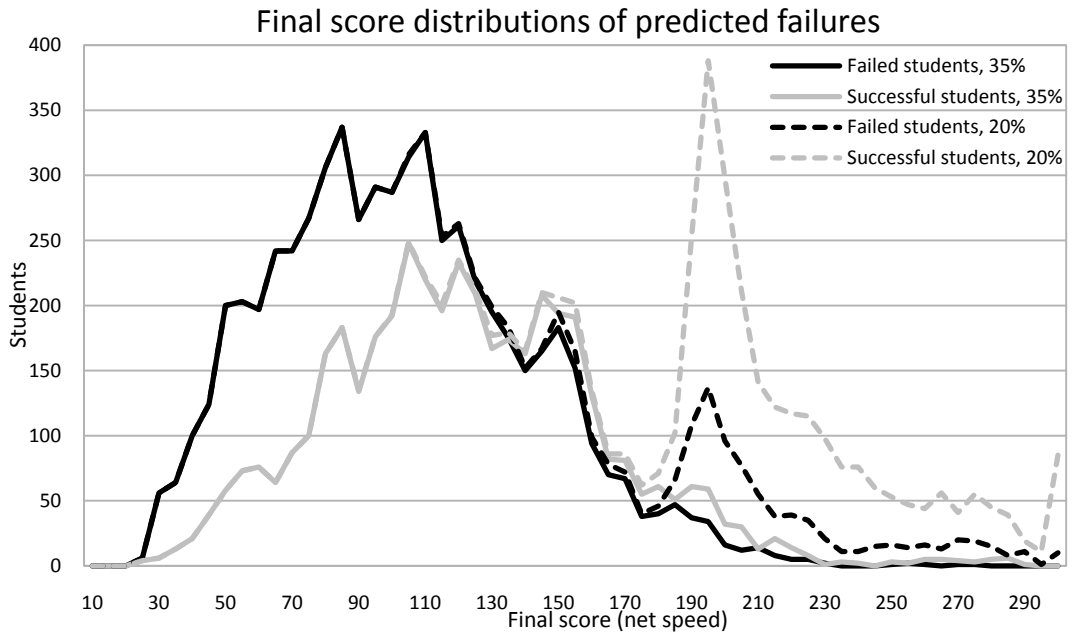

Figure 8: Regression distributions for the baseline classifier at both 35% and 20% neighbour failure

Use of the lower failed neighbour threshold of 20% (dashed lines) increases the number of predicted failures. Several observations can be made here. First, the distribution of predicted final scores now has a second mode around 200 KPM. Second, practically all of the additional students that are now predicted to fail have high predicted final scores of 180 KPM or more. Third, there are two to three times as many successful students among these as there are failed ones (i.e., lower precision). Finally, a good number of additional failed students is actually captured (i.e., higher recall) despite the high predicted final scores.

The first peak at 35–165 KPM seems to contain students who have not yet properly mastered the touch-typing technique. One third eventually does learn it (grey line) and will pass the course. The second group with predicted scores of 170–300 KPM in turn, has essentially learned the touch-typing technique. The failed students in this group (solid and dashed black lines) may have decided to drop out because of reasons not related to the course, or even because they felt they already learned to touch-type and thus reached their learning objective. This would explain the reason for classification results not equalling those of Kotsiantis Pierrakeas and Pintelas (2003).

Corresponding experiments for the distance-weighted regression and classification using the 200-neighbour and the three-feature case resulted in similar distributions to the unweighted case (not shown). Any marginal gains in using the weighted method do not justify the added complexity here.

In conclusion, it seems that the KNN method is able to more accurately find the students that are unable to learn the touch-typing technique and therefore quit. The observations can be utilized in predictions in the following two ways.

5.4.1 Combining regression and failure predictions

The predicted final score can be used to split students into separate subgroups with different KNNs for predicting failure. For instance, the 20% failed neighbour count performed slightly better than the 35% threshold when the predicted net speed was 0-174 KPM. Taking advantage of this fact, we can create a combined classifier that used the 20% neighbour failure requirement when the predicted net speed is between 0 and 174 KPM. The 35% neighbour failure requirement is used from 175 to 229 KPM. At 230 KPM and higher predicted net speeds, no students are predicted to fail. This combined classifier produces a slightly improved classification result with an F_1 measure of 70% with 59% precision and 86% recall (Table 5). A similar strategy of using regression to choose a neighbour failure requirement provides slight improvements also with lower neighbour counts *k*.

Classifier	Features	Neighbours	Best F_1 measure	Precision	Recall
Standard	Net speed	70	69.3%	58.4%	85.1%
Standard, combined	Net speed	70	69.6%	58.6%	86.0%
Standard	5 weighted features	70	69.3%	58.5%	84.8%
Standard, combined	5 weighted features	70	69.6%	58.6%	85.9%
Standard	Net speed	30	68.6%	57.8%	84.3%
Standard, combined	Net speed	30	69.5%	58.1%	86.3%
Standard	Net speed	150	69.4%	58.7%	84.9%
Standard, combined	Net speed	150	69.6%	58.8%	85.3%

Table 5: Summary of best F_1 measures at lesson four and 35% failed neighbours, and precision and recall at the best F_1 measure.

5.4.2 Estimating reliabilities of failure predictions

Given the different distributions of failed and successful students and their predicted final scores (Figure 8), we can also derive estimates of reliabilities of failure predictions. This does not improve the actual precision, recall or F_1 measures; instead, it gives an additional measure that tells the likelihood of a prediction being correct.

To calculate the likelihood the following procedure can be used. When a student is being categorized as a potential failure by the classification method, the predicted final score is also calculated for the student. The ratio of actually failed students to successful students at the predicted net speed (see Figure 8) gives the odds for this prediction to hold true.

For example, assume that we predict a student to fail, and that the predicted final test score is 70 KPM. The likelihood of actual failure is 67% = 242/(242+87), since at the 70–74 KPM predicted speed there are 242 actual failures and 87 successes. This probability information can then be incorporated into the warning indicator in the actual program. For instance, we could categorize the likelihood of failure as weak (less than 40%), moderate (40%–59%) and strong (60% or higher). One possibility is to vary the intensity or opacity of the warning graphic based on prediction certainty.

5.5 Execution time and memory footprint

Beyond the actual prediction performance, it is necessary to assess whether the KNN method is efficient enough to be used in a production server. Furthermore, the memory footprint of the training set cannot be too large, since it needs to be always resident in RAM when the service is active and thus directly reduces the memory available to other parts of the system. The current data set of 14,870 students requires approximately 14 MB of RAM, which is a non-issue in the current production server that has 2 GB of RAM available.

At 30 neighbours, the time taken for a single student's KNN prediction is consistently 5-10 milliseconds. With 70 neighbours, the KNN prediction takes 16–20 milliseconds. 150 students require 30–35 ms and 200 neighbours 35–40 ms. Naturally, a lower processing time is favourable. In the production environment, this calculation needs to be done only when a student completes and exits a lesson, which happens only once per 20–45 minutes for any given actively studying student. The load on the servers is therefore minimal with the above-mentioned running times.

6 Discussion

Usability of KNN for the student performance prediction problem. We experimentally tested a large number of variants of the KNN method for the problem of predicting student performance. Overall, the KNN method can predict student failure quite well. Although our results are not as impressive as those of Kotsiantis Pierrakeas and Pintelas (2003) or Minaei-Bidgoli et al. (2003), they certainly are useful in identification of students at the risk of failure. With a recall of about 85%, most of the students who will fail (or would fail without intervention) are identified. At the same time, with precision of about 58%, there are almost as many false positives. It is not clear how false these are, however, since these students may genuinely belong to the risk group, but have succeeded despite the risk.

The KNN regression method produced surprisingly accurate predictions for the final score. A student who has successfully learned the touch-typing technique usually reaches a speed of 200–300 keystrokes per minute (KPM), so an RMSE of 37 KPM (see Table 2) equals a prediction error of 12%–20%. This kind of variation is common even in a single student's successive test attempts, depending on the difficulty of the test and the physical and mental preparedness of the test taker. This also implies that it could be hard or even impossible to obtain more accurate predictions in this application.

Optimal KNN parameters for classification and regression. Immediate observations about how to apply KNN to the student performance prediction tasks can be summarized as follows (see also Tables 3 and 4). First, the selection of suitable values for the KNN parameters has a significant effect on the results. The number k of neighbours needs to be sufficiently large and the failed neighbour threshold directly affects the precision/recall trade-off in classification. Second, neighbour distance weighting turned out to produce good results only with large numbers of neighbours, if at all, and is not useful in our application. Third, feature selection and weighting had only a minor effect on classification results. The net-speed feature alone performed as well for classification as any set of features at lesson four. However, a weighted set of five features was more accurate during the progress of the course, especially at the first and last lessons (Figure 4). The final score prediction yielded opposite results: The three-feature set seems preferable at lesson four as it is less sensitive to the number of neighbours. The single net-speed feature, in contrast, performs more consistently throughout the course (Figure 6).

Many of the improvements over the simple baseline KNN produced improvements that have to be categorized as incremental at best. This is not surprising, however, since also the results of improvements in original research mirror these findings. In both Dudani's (1967) distance-weighted classification and Altman's (1992) weighted regression, improvements were modest and required large neighbour counts. Furthermore, combining all of the improvements in our study did not result in a combined improvement overall. Instead, it seems that these individual improvements cancel each other out.

A possibility for improving the prediction results is to gather much more detailed demographic data to complement the study data already present. In its current state, demographic data is very limited. By giving account holders the possibility to define their organization type in more detail and by adding age information to groups or individuals, it will be possible to match similar organizations and students of similar ages and skill levels. This might lead to improved prediction performance.

Implications for skill-based courses. An interesting result is that the net-speed feature alone performed so well. This implies that the KNN method could be just as effective in other more generic learning management systems (LMSs) such as Moodle, where only a single lesson score is available for student assessment. Especially other skill-based courses could be a good fit for the KNN method.

Another result of interest to skill-based courses is the good prediction accuracies achieved already after the first lesson. In this course at least, the performance (skills) of a student after the first lesson strongly predicts the student's final test score as well. The KNN method can thus be used as a very early warning indicator.

Completing the course vs. learning. Based on the prediction performance numbers, the failure prediction problem seems to leave room for possible improvement. However, trying to identify people who decide to quit the course can be inherently difficult. The results indicate that failed students missed by the KNN classifier would have most likely performed very well on the final test (cf. the difference between the dashed and solid black lines in Figure 8). The reasons for quitting were therefore extracurricular and hard to predict.

In the extreme case, students quit simply because they feel that they know the touch-typing technique well enough. This holds especially for students who quit at the later lessons. It seems that those students who quit due to the difficulty they experience with the course material do so already during the first 4 lessons of the course.

This observation raises the question if the original goal of predicting the pass or failure of a student is the right one. Alternatively, would it be more appropriate to predict if a student learns to touch type, regardless of whether he or she follows the course to the end and takes the final test? After all, this is the goal of the students, and the primary task of the teacher is to help students learn, not to pass the course.

Following this idea, we can define students that have a predicted final test score of 140 KPM or greater as having successfully learned the touch-typing technique. The results now look completely different. When we exclude these students from being predicted as failures, the prediction results become significantly better with an F_1 measure of 77% with 63% precision and 99.5% recall vs. 70% F_1 measure, 59% precision, and 86% recall at best otherwise (Table 5).

7 Conclusions

We have studied the problem of predicting student performance in an online touch-typing course. The motivation for the work is to provide teachers improved ways to use computer-based training

systems on their courses. With reliable predictions of the performances of the students, the teachers can focus their efforts on those students and issues that need attention the most.

We have applied the KNN method for predicting whether students will complete the final test and, if they do, what their test score will be. The results show that KNN can produce such predictions accurately, especially for the final scores, and already after the very first lessons. We experimentally evaluated a large amount of extensions and feature sets, and found out that differences in predictor performances are small, but some parameter values are more robust than others across different settings.

An interesting experimental result, in particular to other skill-based courses, is that early tests on skills (net typing speed in our case) can be strong predictors for final scores. On the other hand, we observed a relatively high dropout rate among students with high scores, i.e., ones that most likely would have performed well in the final test. Predicting who will quit for extracurricular reasons, or because of already having learned touch-typing sufficiently well, is difficult. However, good results in predicting final scores indicate that students with learning problems can be found reliably.

The results of this research will be used directly to implement an early warning feature for teachers that will alert them of students who are likely to fail the final test. This will allow teachers to identify and focus on the students who are at risk, thus helping them already before difficulties become overwhelming.

References

Alessi, S.M., & Trollip, S.R.(1991) Computer-based instruction: Methods and development (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.

Altman, N. (1992) 'Introduction to kernel and nearest-neighbour nonparametric regression', *The American Statistician*, vol. 46 no. 3, pp. 175–184.

Brusilovsky, P. (1999) 'Adaptive and Intelligent Technologies for Web-based Education', *Special Issue on Intelligent Systems and Teleteaching, Künstliche Intelligenz*, issue 4, pp 19-25.

Cover, T. and Hart, P. (1967) 'Nearest neighbour pattern classification', *IEEE Transactions on Information Theory*, vol. 13, issue 1, pp 21–27.

Daelemans, Walter and Van den Bosch, Antal (1992) 'Generalisation performance of backpropagation learning on a syllabication task', In: *Proceedings of the Twente Workshop on Language Technology 3: Connectionism and Natural Language Processing*, pp. 27–37

Dumais, S., Platt, J., Heckerman, D. and Sahami, M. (1998) 'Inductive learning algorithms and representations for text categorization', In: *Proceedings of the International Conference on Information and Knowledge Management*, pp. 148–155.

Dudani, S. A. (1976) 'The distance-weighted k-nearest neighbour rule', *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, no 4, pp. 325–327.

Figlio, D. N., Rush, M., and Yin, L. (2010). *Is it live or is it internet? Experimental estimates of the effects of online instruction on student learning* (NBER Working Paper No. 16089). Cambridge, MA: National Bureau of Economic Research.

Ishii, N., Hoki, Y., Okada, Y. and Bao, Y. (2009) 'Nearest neighbor classification by relearning'. In: *Proceedings of the 10th international conference on Intelligent data engineering and automated learning (IDEAL'09)*, pp., 42-49.

Islam, M. J., Wu, Q. M. J., Ahmadi, M. and Sid-Ahmed, M. A. (2007) 'Investigating the Performance of Naive-Bayes Classifiers and K-Nearest Neighbor Classifiers', *International Conference on Convergence Information Technology*, pp.1541–1546.

Han, Y. and Lam, W. (2006) 'Exploring Query Matrix for Support Pattern Based Classification Learning', *Advances in Machine Learning and Cybernetics, Lecture Notes in Computer Science*, Vol. 3930/2006, pp. 209-218.

Kotsiantis, S., Pierrakeas, C. and Pintelas, P. (2003) 'Preventing student dropout in distance learning systems using machine learning techniques', In: *Proceedings of Seventh International Conference on Knowledge-Based Intelligent Information & Engineering Systems, Lecture Notes in Artificial Intelligence*, Vol. 2774, Springer-Verlag, pp. 267–274.

Li, F. and Yang, Y. (2003) 'A loss function analysis for classification methods in text categorization', In: *Proceedings of the International Conference on Machine Learning*, pp. 472–479.

Manning, C., Raghavan, P. and Schütze, H. (2008) *Introduction to Information Retrieval*, Cambridge University Press.

Minaei-Bidgoli, B., Kashy, D.A., Kortmeyer, G. and Punch, W.F. (2003) 'Predicting student performance: an application of data mining methods with an educational Web-based system', *33rd Annual Frontiers in Education*, vol. 1, pp.T2A–18

van Rijsbergen,C. (1979) *Information retrieval*, Butterworths, London.

Shih, B., and Lee, W. (2001) 'The application of nearest neighbour algorithm on creating an adaptive on-line learning system' In: *31st Annual Frontiers in Education Conference,* vol.1, pp.T3F–10–13.

Zhanga, M. and Zhou, Z. (2007) 'ML-KNN: A lazy learning approach to multi-label learning', *Pattern Recognition*, Vol. 40, Issue 7, July 2007, pp 2038-2048.

Zou, Y., An, A., Huang, X. (2005) 'Evaluation and automatic selection of methods for handling missing data', *IEEE International Conference on Granular Computing*, vol.2, pp. 728- 733.