Course outcome prediction with transfer learning methods

Jarkko Lagus

Master's thesis
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, May 24, 2016

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Jarkko Lagus

Työn nimi — Arbetets titel — Title

Course outcome prediction with transfer learning methods

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	May 24, 2016	66

Tiivistelmä — Referat — Abstract

In computer science, introductory programming course is one of the very first courses taken. It sets the base for more advanced courses as programming ability is usually assumed there. Finding the students that are likely to fail the course allows early intervention and more focused help for them. This can potentially lower the risk of dropping out in later studies, because of the lack of fundamental skills. One measure for programming ability is the outcome of a course and the prediction of these outcomes is the focus also in this thesis.

In educational context, differences between courses set huge challenges for traditional machine learning methods as they assume identical distribution in all data. Data collected from different courses can have very different distributions as there are many factors that can change even between consecutive courses such as grading, contents, and platform. To address this challenge transfer learning methods can be used to as they make no such assumption about the distribution. In this thesis, one specific transfer learning algorithm, TrAdaBoost, is evaluated against selection of traditional machine learning algorithms. Methods are evaluated using real-life data from two different introductory programming courses, where contents, participants and grading differ. Main focus is to see how these methods perform in the first weeks of the course that are educationally the most critical moments.

Avainsanat — Nyckelord — Keywords
transfer learning, educational data mining, machine learning

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction

In recent years, we have started generating vast amounts of data and also collecting much of it. In many cases, there has not even been any direct use for it at that moment. We are going fast towards an era where everything is connected and generating data: online stores, social networks, manufacturing, mobile devices and so on. This opens up a grave need for efficient ways of processing and making use of such data. The need for efficient ways for processing makes the use of machine learning methods very appealing, as these methods allow us to extract knowledge and make sense out of this data.

Machine learning itself is a collection of methods from computer science, mathematics, and statistics, which are used in combination to extract knowledge from data. State-of-the-art methods and research of each field make machine learning methods computationally efficient and statistically robust.

Machine learning methods have found their way to many different data-intensive fields. It is used in fields such as biology, economy and physics and numerous applications have been developed and studied. These applications include, but are not limited to character recognition, genome sequencing, sales prediction systems, recommendation systems, intelligent tutoring systems, and intrusion detection.

Educational sciences have also adopted machine learning methods and these methods are applied in many ways: to automate tutoring [26], predict course outcomes [38], detect student characteristics [11], predict learning disabilities [64, 43], modeling learning process [48] and much more. Many of these can be done non-intrusively as they only use already existing data from learning environments to do the analysis and predictions. Machine learning methods can be very beneficial in the educational context as usually in the educational settings the number of personnel and resources are very limited. These methods allow faster and less cumbersome way to aid teaching, monitor progress and detect possible problems, when compared to the traditional methods used in education. All of these are major concerns from the point of view of the learning experience.

One of the major problems currently in educational sciences is that it is usually hard to collect data as most of the work done in schools is done with paper and pencil, but the needed technology is slowly finding its place also

in schools.

1.1 Problem statement

In the educational context, only small amounts of observations can usually be collected from a single course instance as courses have limited capacity for students. In some environments, we can collect huge amounts of data per user, but it probably is not very beneficial and might create only noise. Going through great quantities of data is also computationally inefficient. One can consider a case where we have ten users having a dataset of 1000 features, such as age, gender, correctness of exercise one, and then having 1000 users with a dataset containing ten carefully selected features per user. The latter case is always better, because of how the learning of rules is done.

There exist some special situations where we might have a great amount of data from many users, such as in massive open online courses (MOOCs). However, these kinds of environments can introduce other problems, like a sharp decline in course activity [9] that can have an effect on the usefulness of the data. MOOCs also suffer from the fact that many people originally have no intention to complete the course and, for example, do not do any assignments [32], which is rarely the case with traditional courses. At the same time, a considerable amount of data can also be collected in the more traditional settings if we allow contexts to be different but related. These contexts can be, for example, the same course from the previous year, a similar course with a more advanced topic or corresponding courses in other universities.

This kind of cross-context data set can be problematic from the point of view of traditional machine learning methods as they make the assumption that the distributions in the target task and source task are identical. Doing predictions based on this kind of assumption might decrease the prediction performance when these machine learning methods are applied [1]. Interventions can also introduce major problems in educational context from the point of view of impact evaluation because after an intervention, the following results are likely not comparable to most of the previous results gained from course data.

Transfer learning methods are designed for this kind of cross-context use and should be quite suitable as these methods do not make as strong assumptions about the data. Transfer learning is a subset of machine learning,

that deals with data coming from multiple sources that might have different distributions. Transfer learning methods consider the different distributions and adjust the learning from the data to minimize the negative effects on prediction performance. Transfer learning methods aim to help to learn new problems more quickly and efficiently by reusing the old data.

In theory, previously mentioned problem with interventions rendering previous data unusable should not be a problem when using transfer learning methods. As these interventions can be thought to be just a change in the context, transfer learning should allow the reuse of the previous data, even after the intervention.

1.2 Research questions

In this thesis my main research questions are:

- Is there significant help from transfer learning in educational context when compared to traditional machine learning methods?
- How much previous data do we need?
- How much labeled data is needed from the new task?
- Are the predictions accurate already on the first week or does it, for example, get better after a certain number of weeks?

With the research question one, the aim is to see if usage of transfer learning methods can be justified in the educational setting as transfer learning is not always applicable in every situation. In second and third questions we seek for the optimal amount of previous data that is needed for transfer to have an effect over the traditional methods.

In the last question the most important thing from the point of view of educational context is evaluated: can we have accurate results early enough, so that interventions can be done early enough.

For this thesis data from a programming environment that has been augmented to gather data from students programming process. In addition, background variables and course results are combined with data aggregated from the programming process.

Actual prediction task is to recognize students that pass and fail the course, based on the final grade of the course. Grading is based on work done during the course and on pen and paper exam.

The author has implemented the needed transfer learning methodologies based on the original articles.

1.3 Related work

At the time of writing, there has not been very much related research done with transfer learning in the educational context. There has been no directly related work done on transfer learning in traditional educational setting, but one bigger study considering MOOC was done by Boyer et al. [7]. In their work, Boyer et al. studied MOOC data from three instances of edX course called "Circuits and Electronics". Their aim was to predict stopout, an event where students stop engaging with the course material. Multiple transfer learning methods were used on features generated from MOOC data. These features contained mostly information on time spent on different tasks and number of submissions done. In contrast to this study, transfer learning methods were only compared to each other, which makes it impossible conclude if the transfer learning methods do better than traditional methods.

Boyer et al. [7] suggest that MOOC environment might be an ideal platform for transfer learning as there are multiple offerings of the same course, and much of the work is done on the platform. This though has not very much support from previous studies on MOOCs [9, 32], that suggest that MOOC can be a problematic platform.

Some other mentions about using data from different context have been made for example in an article by Ahadi et al. [1]. In that study, a notable decrease in predictive performance can be observed when applying same models to a different context. Similarly, the applicability of specific methods in different contexts has been studied by Petersen et al. [47]. In their study, a performance of a particular algorithm for evaluating programming performance, Error Quotient, was studied under different contexts. Likewise, a decrease in performance was recorded when context changed.

Fail or dropout predicting has been done in many studies with traditional statistics [14] and with machine learning [54, 38].

Transfer learning itself has been studied in many different situations other than educational [23, 37] and a few bigger surveys have been made [44, 37]. Some of the most promising applications have been in indoor localization and brain-computer interfaces. While not directly related to education, methods developed in other settings are usually transferable to other domains as well,

because no domain specific assumptions are usually made.

Variables affecting learning in the school environment has been studied for a long time in different settings. Probably the most thorough research about variables affecting learning in the traditional setting is a meta-analysis by Hattie [27]. In that study, more than 10,000 papers and over 100 variables are covered.

In the context of computer science, and more specifically in programming course setting, a lot of research has also been done. Most of the studies are focusing on an introductory course in programming [48, 1, 5], as it is one of the first courses in computer science studies and sets the base for more advanced topics.

1.4 Structure of the thesis

This thesis is organized as follows. Section 2 introduces some basic definitions for machine learning, some essential methods, applications and background information. It aims to give a solid foundation for the next section where we go more specific setting of transfer learning.

In section 3 we go through transfer learning thoroughly, although the main focus is to motivate the possible usefulness of transfer learning in educational context.

In section 4 the context will shift more towards educational sciences and starts by going briefly through some basic variables affecting traditional schooling. After this, a computer science specific variables are considered. The review of variables concerning computer science is more elaborate as those are in the main focus in the analysis part covered in sections 5 and 6.

Section 5 contains the context descriptions containing detailed information about the data and preparations done before the use in the analysis.

Section 6 contains the results from chosen machine learning and transfer learning methods, and the section 7 discusses the results and answers to research question set previously. In addition, some ideas for future work are given.

2 Machine learning

In this chapter, we go through the basic concepts of machine learning and its methods. We look for some examples how these methods are usually applied and evaluated. The focus is mainly on supervised machine learning methods, as those are the ones used later on in the actual research part. We also focus mostly on classification, but regression is also discussed briefly.

2.1 Overview

Merriam-Webster online dictionary [41] defines traditional learning as "the activity or process of gaining knowledge or skill by studying, practicing, being taught, or experiencing something: the activity of someone who learns." In machine learning, it is a computer that is effectively doing the learning and extracting the knowledge that can then be used in some meaningful way. The end-user can be the computer itself, other systems or some living person.

Machine learning methods are usually applied when it is too difficult to compose an exact algorithm for some particular task, exact meaning that the algorithm will in every situation give the correct result. A good example of this kind of algorithmic problem is image classification. In theory, it could be possible to handcraft very complex algorithm that identifies objects in images, but in real-world this algorithm would probably be too complex for humans to create, and even more difficult to prove that it will give correct classification for every possible image.

This is where machine learning significantly differs from the traditional algorithm design. In traditional algorithm design we usually define manually the process of how the algorithm does decisions and prove that given any input, the output will be the correct. In machine learning, we let, in some sense, the computer itself to decide how it functions by learning the input data. We give some input data to the learning algorithms and hopefully these algorithms can extract enough knowledge to produce good results. In machine learning, we do not require that the learned model is perfect, it only has to be good enough. It is good to notice that we still require the learning algorithms to be exact and correct.

Machine learning is applied in many fields and problems like medical diagnosis, weather forecasting, recommendation systems, sales predictions and sentiment analysis, to name a few. Every data-intensive field is probably

using methods of machine learning in some way.

There are also things like personalization where, for example, a new user gets personalized experience on some website based on the behavior of similar users on the site. If we would do this using traditional methods, someone would analyze user behavior and write the algorithm by hand. It gets infeasible almost instantly for humans, but can be done with machine learning without too much effort.

Algorithm-wise learning in machine learning is similar to a genetic algorithm or genetic programming where the algorithm adjusts or rewrites itself by optimizing some objective function. In machine learning, we usually optimize for prediction performance. One significant difference between genetic algorithms and machine learning is that in machine learning the given data defines the function, and genetic algorithm uses randomization.

2.2 Formal definition

Using notation from article [44] by Pan et al., to formally define machine learning problem, we give following definitions:

Definition 1. (Domain) $\mathcal{D} = (\mathcal{X}, P(X))$, where $X = \{x_1, x_2, ..., x_n\} \in \mathcal{X}$ is the vector of features and $P(X)$ is a marginal probability distribution. The domain defines the space of possible observations.

Definition 2. (Task) When give a specific domain, a task is defined as $\mathcal{T} = (\mathcal{Y}, f(\cdot))$, where $\mathcal{Y}$ is a label space and $f(\cdot)$ is a objective predictive function. Function $f(\cdot)$ is not previously known, but can be estimated from observations.

Definition 3. (Observation) Observations are pairs (x_i, y_i) where $x_i \in X$ is the vector of features and $y_i \in \mathcal{Y}$ is the label corresponding to vector x_i.

We also need additional definition of learning algorithm, the one that learns the objective predictive function

Definition 4. (Learning algorithm) $\mathcal{A}$ is an algorithm doing the learning of the function $f(\cdot)$. After learning the function $f(\cdot)$, it can be used to predict corresponding label $f(x)$ for instance x [63].

2.3 Categories of machine learning

Traditional machine learning is usually divided into two broad categories, supervised and unsupervised machine learning, based on the availability of labeled data.

In supervised machine learning, we have some data that has labels with it, and we train the model using that information. When doing predictions in supervised machine learning, there are usually two related problems that we are solving for, depending on the output we are seeking. We are either interested in classifying new data instances to some previously known classes like good or bad or trying to predict some continuous numerical value, for example, grade. The former is called classification, and the latter is called regression.

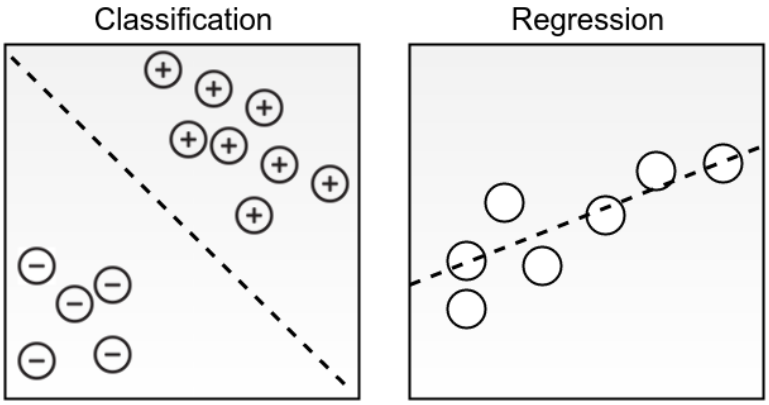

Figure 1: Difference between classification and regression

In unsupervised machine learning, we do not have any labeled data available. Even though there are no labels, the data can be very interesting and useful as there might still exist some patterns in it. In unsupervised machine learning, we are mostly interested in the similarity between data instances. The most used methods in unsupervised machine learning are clustering and association mining.

One specific method of unsupervised machine learning is clustering, where we try to group the instances of our data set to clusters of instances that are similar to each other. By clustering, we can identify similar groups of instances. One example is grouping persons by items they bought from an online store. The clustering gained from this can be useful in many cases

even though we do not have any specific labels on them. Of course, after we have done the clustering we can give some labels to each cluster and then create a suitable input for supervised machine learning and then do automatic labeling for new instances.

Another much-used process that is considered to be unsupervised machine learning is dimensionality reduction. In dimensionality reduction, we are trying to lower the number of dimensions and choose only the features giving the most information. This can be a very important application as we might have very high dimensional data, but because of lack of computing power, cannot analyze it efficiently, if we have to consider all the dimensions.

Dimensionality is not only used to reduce the number of features to get calculations to be feasible but can also give information about what features contain most of the information. When considering data mining, these features can tell much about the data itself. Dimensionality reduction is also a much-used method in transfer learning introduced in section 3.

2.4 Data and features

Data is the basis of everything in machine learning, and there are many steps from raw data to a useful model. It is important to know about the methods and properties of the data before cleaning and using it. Careful preparation of the data can have a huge effect on the final results, probably even bigger than a selection of some specific machine learning method.

Usually, most of the time in analysis done with machine learning methods is used on cleaning and getting the data. There are many things to consider like phenomena called the curse of the big data, interpretation of missing values and types and ranges of variables.

There are two extremes in pattern types that can be found from the data. There are ones that are effectively incomprehensible and ones that reveal the structure quite transparently. The latter ones are called structural patterns as they capture the decision structure in an explicit way.

A simple example of a structural pattern is, for example, a simple rule: "if points > 20 then grade $= 5$". The meaning of this rule can be easily interpreted, and the meaningfulness can be assessed. These explicit structures are almost as important as the predictive capabilities of some predictive model, as we are not just interested in the prediction itself, but also gaining knowledge about what lead to this prediction. We are not always even trying

to generate a model for prediction purposes, but just to see what kind of rules we can extract. These rules can quickly summarize huge datasets in a more comprehensible way as we remove some of the unnecessary details. These rules form the previously mentioned structural patterns.

The opposite situation to structural data can be for example when generating a neural network. In this kind of model, it is hard to tell what kind of predictions it will produce by just looking at the structure, and it is very hard to extract any meaningful information from the neural network itself. It can be thought to be the same thing that if you could see brains of someone, from where you could tell what kind of memories are saved there or what kind of thoughts are going through.

There have been some methods recently that have been used to detect what specific layers in image recognition neural networks do by feeding them some noise and seeing what kind of images they produce.

Usually, we would like to have a considerable amount of data to begin with, but a large amount of data can be very problematic in some cases. With enough data, one is almost guaranteed to find some relations and correlations by chance. This phenomenon is usually called, "the curse of big data". We might, for example, find almost perfect correlation between events "Math doctorates awarded" and "Uranium stored at US nuclear plants" [55]. One should always remember that correlation does not imply causation. That is why meaningfulness is also an important factor in discovered patterns. This usually requires some expert opinion on discovered patterns.

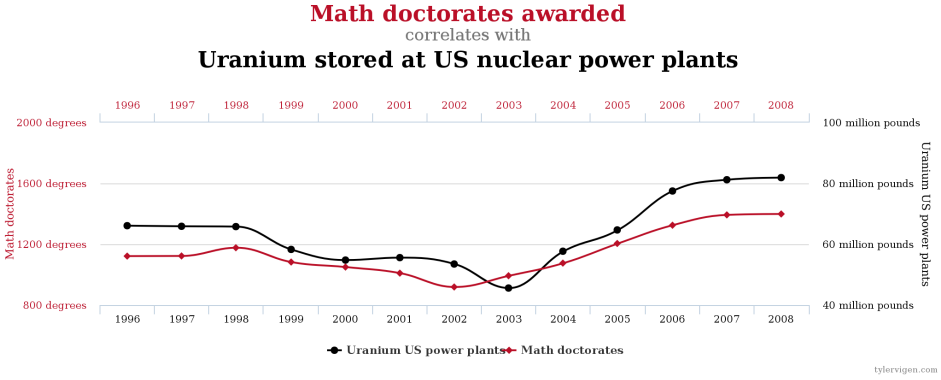

Figure 2: Two events that are unlikely to be related but have close to perfect correlation [55].

Another easily overlooked problem with the data is missing values. There

can be different reasons for missing values. It can be that measurement equipment was not able to measure some property or that some particular test was intentionally omitted. If some measurement was omitted, there might be some reason for that, which gives more information about the gathered data on meta-level. It might be that some other measurement or event lead to a decision to leave some other one out. For example, it could be that we measured that our sample of ice has already melted and now it is useless to do any observations about the shape.

There can be many interpretations for missing values. For example, in the case of missing assignment, there can be different reasons like the student did not return it or system did not correctly save it. It might also be that sometimes missing values are marked to some default value, for example, exam questions are usually given zero scores in both cases, either it was not correct, or it was completely missing. When creating models from data containing missing values, we usually have to map those to some value, simplest choices being zero for numerical values and "unknown" for categorical values.

With structural patterns overfitting can lead to very hard to read rule sets. If we create a very accurate model from a large amount of data, it will usually have some noise or outliers that will generate rules that apply only to those single data instances. This makes the number of rules much bigger and eventually leads to a less clear image of the structure and properties of the data.

2.5 Feature selection

Feature selection is an act of choosing features that are the most representative of the data used. This selection can be done with respect to the variable we are predicting or in a more generic way, by not caring about the target variable, but by just maximizing the overall information gain [24]. The reason for feature selection is that if we have many features, it adds unnecessary complexity to our model as usually some information is shared in many features. If we are given the actual joint feature-label distribution, the error decreases as we introduce more features, but with limited sample size, our error will increase as we add more and more features [28]. With real world data we are not able to find out the actual distribution, so we should always do some feature selection if we have a lot of features.

In many cases, we might have a lot more features than observations. This is especially true in a context where we generate features from data. In educational context, we could have a one or more features per exercise, for example, time used and correctness of the answer. We can have tens or hundreds of these exercises but only a few students, thus generating a lot of features per observation. If we have more features than observations, we easily end up overfitting as there are so many dimensions that we can have each observation to form its separate cluster. Educational context is not the only field that suffers from this kind of problem. Same kind of difficulties also exists in biology and text classification [24].

We usually want to remove some of the features that are highly correlated as they share a lot of information, and we only need one of them to catch most of the information [25]. We can also generate a new variable that explains the correlation between the two variables. These kinds of hidden variables that affect observed variables are called latent variables.

In addition to statistical methods, we can also use machine learning models to do the feature selection. The method used for feature selection doesn't have to be the same used on learning the actual predictive model. For example, in an article by Wu et al. [64], they had best results from feature selection done by support vector machine and actual classification done by the artificial neural network. We do not go into details how these work. Just to give some idea on how this is done a simple, but inefficient, example is that we train models with every possible subset of features and see which feature set gives the most information. Evaluating every possible subset of features requires us to evaluate n^2 subsets, where n is the number of features. Training and evaluating that many models is quite infeasible if we have many features and a great amount of data.

2.6 Learning the model

Learning of the model can be done in two ways. Either we create some abstract model from the data or do instance-based learning, sometimes called also memory-based learning, as we are required to keep the data instances in the memory [13].

In instance-based learning when we are doing the prediction, we compare every previous instance to the new instance and search for the most similar instance or instances. The instance-based learning is covered in more detail

when the nearest neighbor classifier is introduced in the section 2.9.1.

Abstract models usually lose the information about specific instances and aggregate the information to some predictive function. The mechanism behind the predictions is usually less clear, but some models can summarize the data more efficiently than instance-based classifiers. The best known model of this kind is decision tree that summarizes the data with simple rules and tree structure.

2.7 Evaluating models

When generating a model, there are an unlimited amount of ways to create it. There are different algorithms for model generation, and every one of these algorithms has parameters that can be changed to produce a different model. Even the selection of data used for the generation has an effect on this. For these reasons, evaluation is an important aspect of machine learning. Not every model is equally good, and there is no single method that works in every situation. We, of course, want to choose the best model for each situation, and this is where model evaluation plays a major role.

There are a few major themes in model evaluation: we want to avoid overfitting/underfitting the model, we want to use different datasets for training and evaluation and try different splits for these sets, and we want to choose a suitable learning algorithm and parameters for it.

When learning and evaluating a model we usually divide the data set into training and a test set. Using the same data set to do the training and evaluation is generally a bad way to do the assessment. For example, if we generate model so that we create a separate rule for every instance then evaluating this with the same data set gives a perfect score. Nevertheless, being a perfect predictor for the training data, our model will probably perform badly when we do predictions for the new instances, as we have most likely done serious overfitting of the model.

We should always have separate sets for training and testing when evaluating the performance of the model. When we have chosen the best model, we can always combine the training data and test data to create the final model. This way we can use as much data as possible, but do not evaluate the performance of model using false assumptions. This is of course not required if we have much data, but in smaller data sets it might be beneficial.

When creating a model it intuitively sounds like a good plan to optimize

the model for minimal error. Unfortunately, this is usually not the best way to go, as this introduces a problem called overfitting. In overfitting, we have learned the data too well and even though our error for training set is very minimal, we have a significant error when evaluating against the test set. This means that out model does not generalize well. Main reason for the problem in learning the data too well is that the data is often noisy, meaning it always has some error for example because our measurement equipment is not perfect. A good predictor for overfitting is a very small error with the training set and high error on the test set.

The opposite of overfitting is underfitting, where we have too general model. Too general in this case means that we are using too few features or not enough data and cannot for that reason capture the structure of the data well enough. Underfitting is usually manifested as a high error on both test and training sets. As a simple example of using too few features, we might try to model a quadratic function with a line (see the first picture in Figure 3).

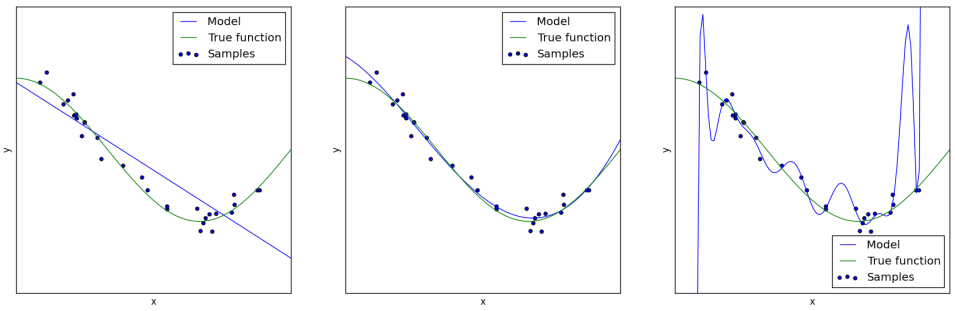

Figure 3: The first picture shows an example of underfitting. The middle one shows a relatively simple function with only small error. The last one fits the data perfectly, but might not generalize as we are overfitting.

2.7.1 Performance metrics

When evaluating the performance, we naturally need some metrics that can be used. Used metrics depend on what kind of problem, classification or regression, we are solving. Both of these have very different strategies to do the evaluation.

Evaluation is not always a straightforward process, and a few things have to be considered. One problem rises in performance evaluation if we have very

unevenly distributed dataset. For example in binary classification problem where we try to detect if an image contains a face, we might have 90 % of the dataset containing images without faces. Now if we create a classifier that classifies every instance to contain a face, we reach automatically accuracy of 90 %. This is of course not very good classifier even though the accuracy is good. That is why careful selection of the metrics used and knowledge about the data is critical.

One method to gain better results in such case is to a use procedure called stratification. In stratification, we select samples for datasets so that our sample will be representative [63]. The problem with ordinary random sampling is that we might not end up with a representative sample because there are some subgroups that might dominate the original dataset and thus also the sample. The idea in stratification is that we divide the data set into homogeneous groups and sample these groups independently. In the previous example, we would sample an equal number of items from both groups, ones containing and ones not containing faces.

Classification

In classification, because of the discrete nature, instead of considering all classes separately, we can rely on the fact that classification is either correct or incorrect, true or false.

One of the most used and simple metric, when we are evaluating classification results, is a confusion matrix. It is a table that has each class once on both vertical and horizontal axis. Because it lists all the outcomes, it is easy visually to detect if there is an imbalance in the dataset and see how the predictions are distributed among different classes.

The most used metric for classification evaluation is accuracy (ACC), shown in equation 1. We use abbreviations TP for the number of true positives, TN for the number of true negatives, FP for the number of false positives and FN for the number of false negatives.

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

From this definition, it can be seen that accuracy is quite vulnerable to imbalanced datasets. If there are for example 90 instances of class 1 and ten instances of the class -1, always predicting class 1 gives us 90 % accuracy.

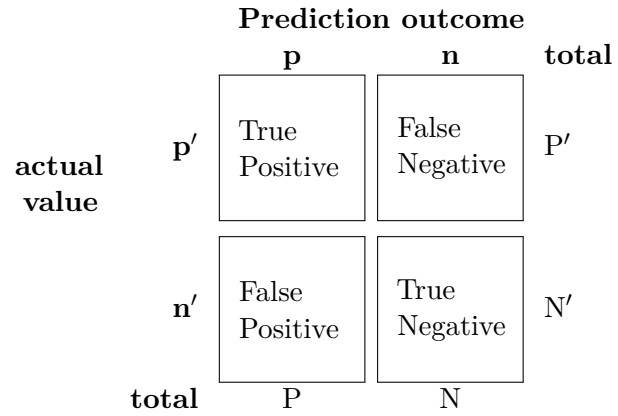

Figure 4: An example of confusion matrix showing a two class case. N, N', P and P' denote marginal totals

For this reason, accuracy should not be used, if it is known beforehand that the dataset is very imbalanced.

Other usual metrics for classification that are not so vulnerable to imbalanced datasets are precision (PREC), true positive rate (TPR), false positive rate (FPR) and F1 score. F1 is basically a weighted average of precision and true positive rate.

$$PREC = \frac{TP}{TP + FP}$$

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

$$F1 = \frac{2TP}{2TP + FP + FN}$$

One useful metric for prediction performance is Matthew's correlation coefficient (MCC) or phi coefficient. It ranges from 1 to -1, 1 means that every prediction is correct, 0 means random guessing (half correct, half incorrect) and -1 means that every prediction is incorrect.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Similar metric to MCC can be derived from a receiver operating char-

acteristics (ROC) graph by calculating the area under the curve (AUROC). An ROC graph is a graph used to visualize the performance of the classifier, where classifiers true positive rate is plotted against the false positive rate [19]. AUROC is a metric that measures the area that is covered under the ROC curve. It has the range from 0 to 1, where 1 means a perfect prediction, 0.5 is random guessing, and 0 means every true prediction is actually false.

With a probabilistic model that outputs probabilities for classes, ROC curves with different threshold values can be plotted. The threshold is the point that decides if the predicted class is true or false. For example, we can assign a threshold so that if the probability of being true is over 0.2 then we classify that instance to be classified as true and false otherwise.

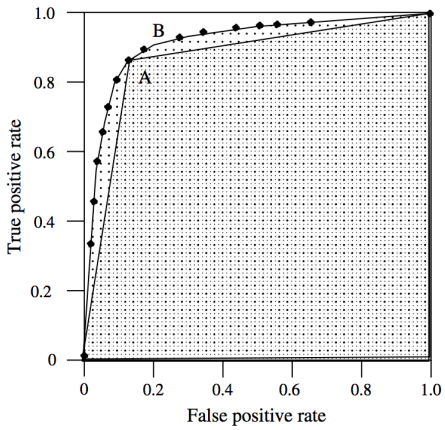

Figure 5: Example of AUROC with two classifiers, A and B, B being better as more area is covered [19].

Regression

In regression, because of the nature of continuous variables, there is no point of looking if the prediction is exactly correct. Instead, we can use performance metrics from statistics such as root mean squared error (RMSE), the sum of squared errors (SSE) and coefficient of determination (R^2).

In next equations we assume that we have n observed data instances $y_1, y_2, ...y_n$ having associated prediction values $\hat{y}_1, \hat{y}_2, ..., \hat{y}_n$.

Root mean squared error is calculated using equation

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

Sum of squared errors is calculated using equation

$$\text{SSE} = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

Coefficient of determination is calculated using equation

$$* R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{2}$$

where the sum of squares of residuals is

$$SS_{res} = \sum_{i}^{n}(y_i - \hat{y})$$

the total sum of squares is

$$SS_{tot} = \sum_{i}^{n}(y_i - \bar{y})$$

$\bar{y}$ is the mean of the observed data. R^2 value has a range of 0 to 1. It is 1 if every data instance perfectly fits on the regression curve and 0 if the curve does not fit the data.

2.7.2 Criteria for model selection

Instead of just looking at the performance of some models and comparing them straightforwardly, it might be useful to use some other metrics, like the complexity of the model, when evaluating which model is the best. Usually, more complex models give better results, but because of overfitting are not the best ones to use. For example, if comparing two decision trees, we could prefer the one with much fewer rules, even though the measured performance might be worse. There are many different methods that can be used, some of the most used being Akaike information criterion and cross-validation.

2.7.3 Cross-validation

When selecting the sets for training and testing, we might end up choosing sets that are not representative. This problem is quite straightforward to

solve by trying several different sets for training and testing. These sets can be selected for example randomly or swapping test and training sets with each other. One of the most used methods is so-called cross-validation where we choose some fixed number of partitions, folds, of the data. We then repeatedly select one of the partitions for testing, and the rest goes to training. For example, if we choose the number of folds to be three, we use one-third for testing and two-thirds for training. Then we do this another two times, every time selecting a different partition to be the test set. After calculating error rates for all of these, we take the average out of the to get the final result.

The overall best number for partitions based on multiple different research articles and datasets seems to be ten [63]. There is probably little difference if we would use nine or eleven folds, but 10-fold cross-validation seems to be most used. It is good to remember that increasing the number of folds increases also the computation time.

One quite straightforward modification of the previously described n-fold cross-validation is called leave-one-out cross-validation. It is basically just the same as the usual n-fold cross-validation, but where we choose n to be the size of the data set. This way we maximize the data used for training and test only with single instances of the data.

2.7.4 Other methods

There are also other ways of doing the evaluation. One much-used method is bootstrapping. It is based on the idea from traditional statistics of sampling with replacement. There we create a new dataset by randomly sampling the original dataset. In bootstrapping we allow the same instance to be chosen multiple times, so for example, choose a sample of size n data set of size n, we end up almost certainly having some data instance many times. The rest of the data set is then used for testing.

One particular variant of this is called 0.632 bootstrap [63]. In this method, we use sample size equal to the size of the data set. The number 0.632 comes from the fact that if we choose randomly elements from size n set, we have a probability of $\frac{1}{n}$ of choosing some particular element. The probability of not choosing some particular element is then $1 - \frac{1}{n}$. If we repeat selection for n times, we have probability of $(1 - \frac{1}{n})^n \approx e^{-}1 \approx 0.368$ for not choosing some specific element at all in n tries. So we expect to

choose approximately 63.2% of the unique elements from the data set some of them repeated.

2.8 Cost function

Closely related subject to performance evaluation is a cost function. Use of cost function allows different penalties for different cases. For example, in the case of predicting if a patient has HIV and if he should have a more detailed examination done, it is better to have more false positive patients diagnosed more carefully than have false negatives to miss early intervention. Also in many less dramatic events, these two cases usually have different costs that should be accounted for when evaluating the performance of the model. Cost assignment can be done arbitrarily, but usually, it should be done by an expert in the domain who is aware of the risks of misclassification. This might be, for example, a medical doctor if we are creating a model for medical diagnosis.

Most simple cost function is a 0-1 cost function. It gives zero cost for correct predictions and a cost of one for false prediction. Another popular cost function for evaluating probabilistic models is the quadratic cost function that is defined for a single data instance as

$$\sum_j (p_j - a_j)^2$$

where a_j is 1 if j was the predicted class and 0 otherwise, and p_j is the probability for the prediction that the class is j. Cost functions are good for evaluating probabilistic models because we can assign a cost for the uncertainty. Optimizing our predictive function can be seen as a problem of minimizing the cost function.

2.9 Classification methods

There has been developed many different machine learning methods that can have very different performance depending on the classification task. Even the simplest methods can on some occasions outperform a more complex models. Next, we will go some of the most used methods of classification.

2.9.1 Nearest neighbour classifiers

One of the simplest methods for solving the classification problem is the nearest neighbor classifier [13]. The idea is very simple and intuitive, just look at which labeled instance is nearest to the one we are classifying and use the same label as the class.

Nearest neighbor classifiers are so called instance-based classifiers that were already briefly discussed in previously. They rely only on a similarity measure that measures the distance between two instances. The similarity measure does not necessarily have to be a metric, but some optimizations can be done, if that assumption can be made [13]. The simplest choices for similarity measure are Manhattan and Euclidean distance that are special cases of Minkowski distance defined as

$$d(\mathbf{x}, \mathbf{y}) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n)$ and $\mathbf{y} = (y_1, y_2, ..., y_n)$. Now setting $p = 1$ gives the Manhattan distance and setting $p = 2$ gives the Euclidean distance.

There exist a few variations from the basic implementation, for example, k-nearest neighbor (kNN) classifier, that take a majority vote on the class using k-nearest instances. We can also extend kNN to consider also weights, by assigning weights on the instances. One use for weights is a setting where we have data from multiple contexts, and we are learning some specific target, that might have slightly different distribution. We can then assign probabilities for each context to have a similar distribution. In this way, we can have more weight on the instances that are from more similar distribution.

2.9.2 Support vector machines

Support vector machine (SVM) is a linear classifier that takes advantage of so-called margins to choose the best possible decision border [20]. The theory behind SVMs has been studied from early 1950, but the first description close to their current form was given by Boser et al. [6].

A linear classifier is a geometric model that uses a linear function as a decision border [20]. There usually exists an infinite number of possible linear classifiers for classification problems, some of them better than the others. The idea behind SVM is to choose the linear function so that the margin

between classes is maximized [20]. A margin is an empty space between our decision border and nearest instance.

Decision rule for the support vector machine is given by the equation

$$\mathbf{w} \cdot \mathbf{u} \geq c$$

where $\mathbf{w}$ is the normal vector to the hyperplane that is our decision border, $\mathbf{u}$ is some data instance we are classifying and c is some constant where the decision border lies on. In other words we are projecting our instance-to-be-classified to the vector normal of our decision border and looking if its "far" enough to have some specific class usually called positive sample. The question we need to find the answer to is then, what the values for vector $\mathbf{w}$ and the constant c are.

For the next steps we can reformat the equation without loss of generality (here $c = b$)

$$\mathbf{w} \cdot \mathbf{u} - b \geq 0$$

The data instances that coincide with the borders of margin are called support vectors and the decision boundary in SVM is defined using a linear combination of these vectors. These borders are hyperplanes that are defined by two equations

$$\mathbf{w} \cdot \mathbf{x} - b = 1$$

and

$$\mathbf{w} \cdot \mathbf{x} - b = -1$$

where $\mathbf{x}$ is the set of points closest to margin and b is the parameter telling the offset of the vector $\mathbf{w}$.

All the other data instances have to fulfill following equation

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \text{ for all } 1 \leq i \leq n$$

where $y_i = 1$ when x_i is a positive sample and $y_i = -1$ when x_i is negative sample. The width of the margin is then defined by

$$(\mathbf{x}_+ - \mathbf{x}_-) \cdot \frac{\mathbf{w}}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

Now finding $\mathbf{w}$ and b can be formulated as an optimization problem where we optimize the width of the margin, that is $max\frac{2}{||\mathbf{w}||} \implies min||\mathbf{w}||$ with constraint $y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1$, for all $1 \leq i \leq n$.

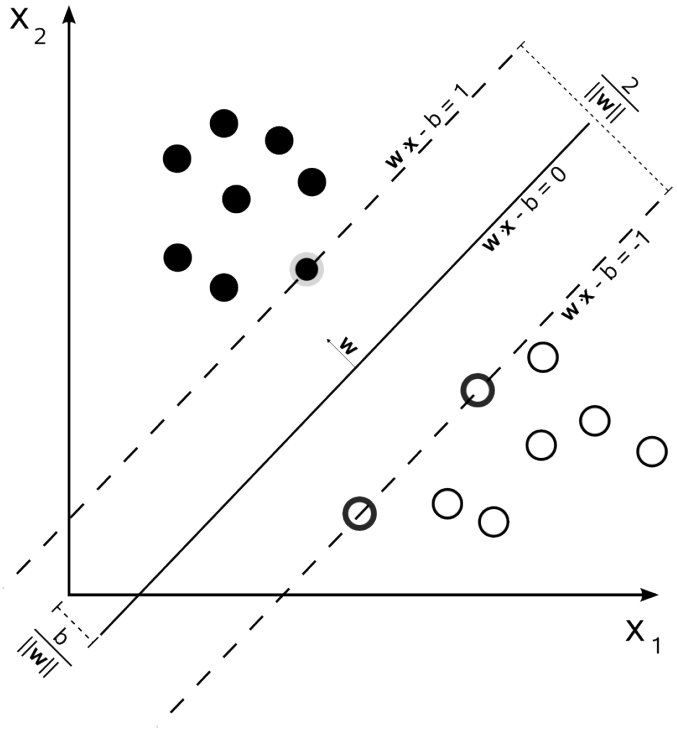

Figure 6: An example of maximal margin separating two linearly separable classes [12].

In the case of non-linearly separable data, kernel methods can be used. The idea behind these is to transform the current data points to a new space where these instances are linearly separable. Instead of actually transforming the space, we can use kernel function that has enough of the same properties as the transformation would have. In this case, the inner product is enough, as solving the previous optimization problem, for example, using Lagrangian multipliers, reveals that the optimal solution is dependent only inner products of data points [20].

So if ϕ is a transformation function from lower dimensional space to higher dimensional space we want to have kernel function $K(a, b) = \phi(a) \cdot \phi(b)$. Now if we can find this kernel function K, we can replace every dot product with it to speed up the calculations. After the transformation, we can use the ordinary linear classifier to do the classification.

One of most used of these kernel functions is called Radial basis function kernel. It is defined as

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\frac{||\mathbf{x} - \mathbf{y}||}{2\sigma^2})$$

2.9.3 Bayesian classifiers

Bayesian classifier is a so called probabilistic model. In probabilistic models we are interested in conditional probabilities, usually denoted by $P(Y|X)$ [20]. In short the previous notation means probability of Y if given X. An example could be probability of coin toss giving heads if we have unbiased coin, P(heads|unbiased coin), or if a person needs glasses if he/she is over 70 years old, P(need glasses|age > 70).

One advantage of probabilistic models is that we will get probabilities that are usually easy to interpret. If one class has the probability of 0.51 and other has a probability of 0.49, we will know that the prediction is not very confident. With other models, we would usually just get a result that the first class is the correct one.

Bayesian classifier is based on Bayes' rule that states that

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)}$$

Here $P(Y)$ is the prior probability and $P(X)$ is the probability of the data. It can be calculated using the law of total probability

$$P(X) = \sum_{Y} P(Y)P(X|Y)$$

If we are not interested in the actual probability, but just the magnitude, we can omit the division by $P(X)$ as this just normalizes the values to be on range 0 to 1. For example, in classification we are usually interested only in finding the greatest value and the class related to that, so the division by $P(X)$ does not give any additional information, but only complicates the calculations.

Using the previous assumptions we can get a class decision rule called maximum a posteriori [20]

$$y_{MAP} = \underset{Y}{\operatorname{argmax}} P(Y)P(X|Y) \tag{3}$$

If we add an assumption that Y has a uniform distribution ($P(Y)$ is equal for every Y) to the previous assumptions, we can get even simpler form for the classification called the maximum likelihood decision rule [20]

$$y_{ML} = \underset{Y}{\operatorname{argmax}} P(X|Y) \tag{4}$$

One specific implementation of Bayesian classifier is called Naive Bayes. It follows the same principles but makes the assumption that all the features are independent. From the perspective of probability theory, this means that to calculate the joint probability, we only need to multiply probabilities together. This makes the calculations computationally fast, but also introduces some error if the features are not truly independent. Depending on how dependent the variables are, this can have a huge effect on the error size.

Naive Bayes classification works by calculating conditional probability for each class with a condition that we have made some specific observations and chooses the best class for that observation. So if we have vector $\mathbf{x} = (x_1, x_2, ..., x_n)$ that has n features and we want to know the probability of class C_k, we are interested in following probability:

$$P(C_k|x_1, x_2, ..., x_n) = \frac{P(C_k)P(\mathbf{x}|C_k)}{P(\mathbf{x})}$$

Now because of the independence assumption we get

$$P(\mathbf{x}|C_k) = P(x_1|C_k)P(x_2|C_k)...P(x_n|C_k) = \prod_{i=1}^{n} P(x_i|C_k)$$

Now to choose best class to be our prediction $\hat{y}$, we want to select one with greatest probability. For this we can use the maximum a posteriori rule given in equation 3. The same equation shown below modified to this specific case:

$$\hat{y} = \underset{k}{\operatorname{argmax}} P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

With discrete variables, we can approximate prior probability by using counts. For example, if we have overall 100 observations and 24 of them

have label A, we have $P(class = A) = 24/100$. With continuous variables, like height or weight, we have to take a different approach. For continuous variables, we can use so called Gaussian Naive Bayes that assumes that variables follow a Gaussian distribution. Probability is then calculated by expression

$$P(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}$$

where σ_c^2 is the variance and μ_c is the mean associated with the class c.

2.9.4 Ensemble methods

Ensemble methods are essentially a collection of other machine learning methods. They group either similar or different methods together to for the final classifier. By grouping multiple classifiers into one final classifier, ensemble methods try to leverage the wisdom of the crowds.

Ensemble methods can be considered to be similar to a group of people trying to solve some problem. There might be one person in the group that is very skilled in problem-solving, but still there is an advantage to have others around. There might be an additional knowledge that some other person has that can then assist the whole group towards a correct solution. Ensemble methods allow us to combine the best worlds of each separate classifier into one final classifier.

The simplest way of doing ensemble model is to take a bunch of classifiers, usually called experts, and take a vote if we are classifying or average if we are making a numerical prediction [63]. We can also assign some weights on the classifiers, for example, we could consider the final model as a linear combination of the models we wish to use

$$C(x) = w_0 c_0(x) + w_1 c_1(x) + ... + w_n c_n(x)$$

where w_i is the weight for ith classifier and c_i is the ith classifier.

Bagging

Bagging, short for bootstrap aggregating, uses the method of sampling introduced when discussing bootstrapping. It takes the training data and creates multiple random samples with replacement [63]. These random

samples are then used to train the models that will end up in the final ensemble model. Bagging uses a collection of classifiers of the same type.

Actual prediction is done either by taking a vote in case of classification or averaging the results in a case of regression. Because of the random sampling, bagging should guarantee good results, as statistically we should end up more models giving correct predictions than false predictions.

One specific and much-used method related to bagging is a random forest. A random forest is essentially a collection of decision trees. The principal idea is to generate many randomized decision trees and use them to vote for the best class.

Random forests use bagging on decision trees to prevent some problems that they have. Decision tree induction is an unstable process: small changes in training data might result in very different trees as a change in top-most rules can alter the structure of the subtree [63]. Decision trees are also prone to overfitting. As shown by Breiman [8], random forests avoid the problem of overfitting and improve in accuracy when more trees are added.

Like previously mentioned, the actual prediction is done by taking a majority vote in the case of classification or averaging the values in the case of regression. Random forests are closely related to AdaBoost introduced in the section about boosting in the way they function [8] and also there is a connection to nearest neighbor methods [35].

Boosting

The idea that boosting builds on is that classifiers should complement each other and not just repeat already known knowledge. If one model has bad performance in some situation, the some other model compensates this lack of performance and vice versa [63]. Like bagging, it uses a collection of classifiers of the same type.

There are many variants of boosting, so we will next go through a one specific algorithm called AdaBoost [21] (short for Adaptive Boosting). AdaBoost is one of the most widely used boosting algorithm and many of the transfer learning algorithms described later on in section 3 use it as a basis or idea derived from it. The idea in AdaBoost is to assign iteratively weights on training data instances. By assigning weights on training instances, the algorithm can be forced to focus on a specific set of instances. Misclassified instances are given higher weight so that learning algorithm prefers those in

selection and weighing of classifiers.

AdaBoost is not a learning algorithm itself, but a method that uses collection of some learning algorithms. The only requirements for the learning algorithm AdaBoost is used with, is that it support weights on training instances, and it is so-called weak learner. A weak learning algorithm is an algorithm that does only slightly better than random guessing.

The algorithm starts by assigning equal weights to all instances. It then trains a classifier with this unweighted training data. This first iteration captures the "easiest" instances to label and uses this classifier to classify all the instances. New weights are then assigned based on the output of the classifier. Correctly classified instances get lower weights and misclassified instances get higher weights. This process is then applied multiple times.

What happens is that instances hard to classify are getting increasing attention from the learning algorithm as their weights increase on every round that they are misclassified [50].

3 Transfer learning

In this section we will go through the basics of transfer learning, mostly focusing on questions what separates it from traditional machine learning, when it is used and why it is used.

3.1 Overview

In traditional machine learning, we are usually making an assumption that the data we are using for learning and the data we are using for testing and predictions are drawn from the same distribution. The problem with this kind of assumption is that it is not true in all situations where machine learning is applied. This can lead to a situation where even small changes in the distribution or feature set can render the original model unusable for future uses or, at least, lower its performance significantly. We would then need labeled data from the new data and learn the model again.

Getting rid of this assumption is the central goal of transfer learning. The aim of transfer learning is to extract knowledge from other related tasks, called source tasks, to be used for improving the predictive model of the task we are trying to learn, a target task. Transfer learning itself is a very natural thing that humans do almost on a daily basis. We have, for example, at some point learned to read and speak our native language and later on in school used this knowledge as a basis when learning a new foreign language. If the grammar, alphabet and words are similar to our native language, we might learn the new language quite fast by taking advantage of the process of transfer learning.

Need for transfer learning methods is well justified, as in many real-world applications lack of data is a real problem. One of the main reasons for this is that gathering the data and processing it can be very laborious. In some particular fields, a great amount of expert knowledge and manual labor is needed and because of that, there is only a very limited amount of data available. This is the reason why making accurate predictions from only small amounts of data is a very compelling problem.

Transfer learning is applied for example in situations where there is not enough data from the target task available to create an accurate model or collecting a new data set is too expensive and time-consuming [44]. Usually, the gains from transfer learning decrease quite rapidly as the amount of

available data in target domain grows.

Studies show that it is almost always beneficial to use multiple sources instead of just a single source [17, 66]. This seems quite intuitive as the original idea of transfer learning is to allow usage of as much data as there is available, and exclude the non-representative instances from the learning process.

Transfer learning methods usually share some characteristics with ensemble learning, and many of the algorithms are based on AdaBoost, that is essentially an ensemble learning algorithm. In both, transfer learning and ensemble learning, we are trying to leverage the wisdom of crowds to improve the final result. The defining difference is that in ensemble learning we create a model that contains multiple models from the same data, but in transfer learning, we take the knowledge out of multiple models with different source data and transfer it to our final model.

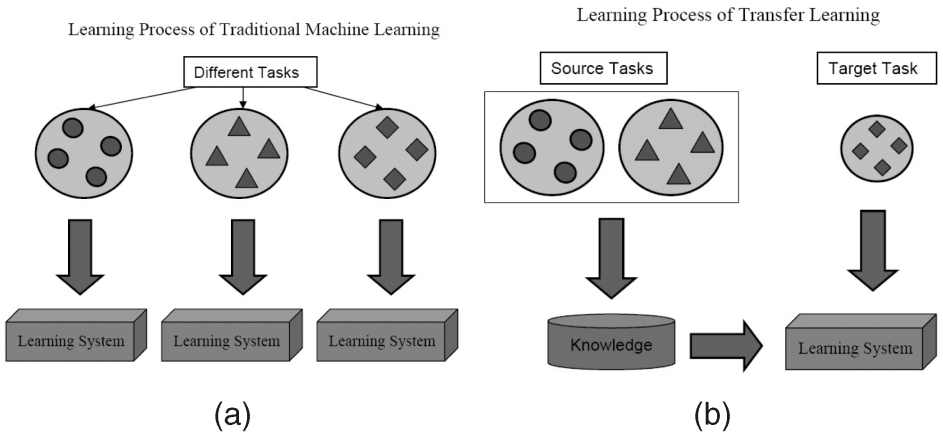

Figure 7: Visualization of difference between traditional machine learning and transfer learning [44]

There are many situations where transfer learning can be applied. Current applications include things like indoor localization [52] and brain-computer interface calibration [57]. In indoor localization, the constant need for calibration makes the transfer learning very suitable to speed-up the learning and in brain-computer interface calibration it is very expensive to acquire always the needed amount data to do the training from the scratch. It has also been successfully used in other fields like bioinformatics [65] and text mining [4].

Transfer learning has been adapted to many machine learning methods. Surveys by Pan et al. [44] and Lu et al. [37] lists transfer learning adaptions for boosting algorithms, support vector machines, Markov logic networks, neural networks, Bayesian classifiers, fuzzy systems, and genetic algorithms.

For the knowledge transfer to be successful, there has to be some common ground between the source and target task. Luckily this is true for many real-world applications as there are many cases when the data is updated only in small portions, and it should be reasonable to assume that at least some of the original data is useful. For example, in the educational context, the courses usually stay the same, but students and lecturer might change between course iterations. This leads to a situation where traditional machine learning methods might do worse because of the assumption that distribution is the same as in the original training data. Continuing with the analogy of learning a new language, it can be though of as a similar situation where in the new language there is an entirely different way of pronunciation of words, but we stubbornly try to use the same pronunciation that is used in our native language. We perform worse because we assumed that pronunciation is the same in both languages.

There are three relevant questions when applying transfer learning methods: what to transfer, how to transfer and when to transfer [44]. Current research mainly focuses on the what and how parts, but also the when part is gaining increasing interest [44].

An answer to "what to transfer" tells us what can and should be transferred. Not everything is useful and can actually make things worse. As an example, we want to avoid instances that are too far from out target distribution or select only the features that are common in both domains.

An answer to "how to transfer" describes the methods to be used in knowledge transfer. The answer for this includes the actual algorithms and the theory justifying that we are transferring knowledge in such way that it actually improves the model.

"When to transfer" is the least studied question and it tries to answer the question of in what situations we should transfer and more importantly, when not to transfer [44].

These all are important questions as if we just arbitrarily try to extract knowledge from some source domain; it can lead to a negative transfer. The negative transfer happens when instead of improving the target task

performance, we make it worse. The performance of transfer learning methods is highly dependent on the source task data [17].

An example of when a negative transfer could happen is a situation where we transfer knowledge of life expectancy prediction data from cancer patients to healthy persons while ignoring the cancer status as a variable. We would, for example, transfer data only from age and educational background that would probably be similar in both groups. This model then would most likely predict much lower life expectancy to healthy persons than it should.

3.2 Formal definition

Using previously defined notation, to define formally transfer learning problem, we can define source domains $\mathcal{D}_{S_i}$ and target domain $\mathcal{D}_T$. For simplicity, we assume that there is only one source domain $\mathcal{D}_S$, but it should be noted that in real-world applications limiting only to one source may lead to negative transfer [66].

Definition 5. (Transfer learning). Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_T$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. [44]

3.3 Settings

There are three different settings that may apply when we are doing transfer learning: inductive transfer learning, transductive transfer learning, and unsupervised transfer learning.

In the inductive transfer learning the target task differs from the source task ($\mathcal{T}_T \neq \mathcal{T}_S$), and we have data available on both source and target domain. In inductive transfer labeled data is required only in the target domain. In such case, we still use the unlabeled data from source tasks to boost the learning. In addition, in this situation, we do not care if the domains are the same or not.

In transductive transfer learning the target and source tasks are the same, we have data available only from the source domain and the target and source domains differ ($\mathcal{D}_T \neq \mathcal{D}_S$).

In unsupervised transfer learning, we have no data from either domain and target and source tasks are different but related. This is different from

previous two in the same manner as supervised machine learning is different from unsupervised learning. In this particular situation, we are usually interested in clustering and dimensionality reduction.

3.4 Approaches

For each setting, there are a few different approaches that one can take: instance-transfer, feature-representation-transfer, parameter-transfer, relational-knowledge-transfer, and classifier-transfer [44].

As the name says, in instance-transfer, we are using previously acquired data instances and do only reweighing or inclusion/exclusion of them. The idea is to choose data instances that are suitable to target domain in the sense that those instances could also be drawn from the distribution of target domain [17]. Dai et al. [15] used this method successfully in their TrAdaBoost algorithm to improve new model using old data. The basic idea in TrAdaBoost algorithm is to filter out every too odd data instances so that at least no negative transfer happens because of them. The filtering is done by reweighing the training samples. The idea behind in that approach is based on Probability Approximately Correct theory, that if greatly simplified, says that from many only slightly better guesses than purely random guessing, a more accurate guess can be constructed [21].

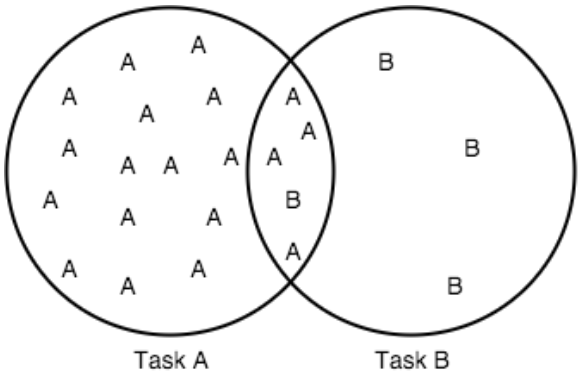

Figure 8: Instance transfer with with a source task A and the target B. In instance transfer idea is to give most weight on instances laying within the distribution of task B

TrAdaBoost originally uses only one source to do the boosting and is, therefore, more vulnerable to negative transfer [66]. This problem has been handled in a paper by Yao et al. [66] by adding support for multiple sources

in their MultiSourceTrAdaBoost algorithm.

Both, TrAdaBoost and MultiSourceTrAdaBoost, use all the data from source and target domains and therefore suffer from possibly irrelevant source data. TransferBoost algorithm by Eaton et al. in [17] tries to fix this problem by doing a selective transfer. TransferBoost is based on the same idea that TrAdaBoost, but tries to avoid choosing bad samples from the source data set.

In feature-representation-transfer we try to select the features most representative for the target domain [44]. This can be done by encoding features in source data to be used in target task [57]. Lotte et al. [36] used this approach to speed up the calibration of brain-computer interfaces for new users. Brain-computer interface systems allow a computer to human interaction through brain activity and are based on EEG measurements that are different for every user. Because EEG measurements of users differ, brain-computer interface system has to be usually calibrated from the scratch for every new user [36]. The algorithm Lotte et al. [36] developed used many source tasks, previous users, and chose a representative subset to be used in the calibration of the new user. The subject of transfer are the covariance matrices used by the learning algorithms [36]. Their results showed as much as 10 % increase in accuracy when there was an only small amount of training data available from the target [36].

In parameter-transfer, we assume that source and target tasks share some parameters or prior distributions and transfer this knowledge [44]. This approach was used by Yao et al. [66]. In their paper, they introduced a two-phase algorithm called TaskTrAdaBoost. The base algorithm is based on the same principles as the previously mentioned TrAdaBoost, but the transfer is separated into two phases. The first phase tries to extract the best parameters using traditional machine learning methods and the second phase does the parameter transfer [66].

In relational-knowledge-transfer, we assume that some relationship between the source and target domain data exists and transfer this information. Mihalkova et al. [42] used this approach with their Transfer via Automatic Mapping And Revision (TAMAR) algorithm. TAMAR is an algorithm that transfers knowledge of Markov logic networks between related domains [42]. Markov logic network is a model that combines first-order logic with probabilities, and the structure is defined by first-order logic clauses and weights.

TAMAR algorithm does the transfer by doing predicate mapping a theory refinement between domains [42]. The basic idea behind the algorithm is that there are relationships between different domains, as for example, a mapping between academic department and movie staff: directors to professors, actors to students and movies the research papers [42]. The predicate mapping part is the transfer of these connections, and the theory refinement is the reweighing of the connection weights. Results in their paper [42] show that the approach successfully reduced the learning time and accuracy of the final model.

In classifier-transfer, we create ensemble classifier from a collection of base classifiers that are learned from different domains and create one final classifier for target task [57]. Classifier-transfer also includes domain adaption of classifier [57]. This approach seems to be only studied on brain-computer interface research field.

Most of these methods are studied only for the case of inductive transfer, where we have data from both domains. Many of these also make the assumption that the feature sets are identical that limits the applicability on some tasks.

3.5 Related methods

There are also a few quite similar, but conceptually slightly different learning methods like multitask learning, knowledge transfer and meta-learning [44]. Sometimes these terms are also used interchangeably with transfer learning, so it is always good to check how the terms are actually interpreted.

In multitask learning we are trying to optimize several models simultaneously while in transfer learning we are interested in optimizing only a single target task, even if we use and train other models while doing it. So basically any multitask learning algorithm is a transfer learning algorithm where in the end we throw away the information learned about the tasks other than the target task.

Meta-learning is very close to transfer learning, but instead of relying only on data, it also uses relevant meta data like properties of learning algorithm or properties of the domain.

4 Variables affecting learning

In this section, we will go through some variables that are considered to be good indicators of academic success in traditional schooling system and in a more specific setting of programming courses.

4.1 Overview

Learning process in humans is very complex, and there are many different variables affecting the end results making it hard to measure. A study by Hattie [27], for example, lists 138 different variables all having different effects on learning. For machine learning methods to be successful, choosing the right variables is very important. Even though these methods allows us to use and test much more variables more easily, preliminary choice of variables should still be done in some manner. Methods of machine learning allow us also to do validation and evaluation on our selected variables, so we do not have to trust blindly that the variables are good.

Some variables might not be as good indicators as have been thought previously. For example, in a not directly related study done by Wu et al. [64], different methods for detecting learning disabilities were tested. They noticed that some of the features that are considered important predictors for learning disabilities did actually have very low confidence and contribution to the prediction performance. These kinds of false assumptions seem to be quite common in the educational context, probably due the fact that learning is hard to measure, and that even seemingly subtle changes can have quite a huge effect.

In traditional schooling system, variables affecting learning have been studied for many years, and tens of thousands of papers from the subject have been released up to the current date. Meta-analysis of meta-analyses by Hattie [27] summarizes results from over 800 meta-analyses that cover research data from over 10 000 research papers. The results in that huge meta-analysis suggest that almost everything works in teaching as nearly every method or variable studied there increased learning. Nevertheless, Hattie suggests that it is not a completely honest way to do the comparison by comparing some method to "not doing anything", as we are always using some method when teaching or learning. Setting the null hypothesis to be do-nothing and alternative hypothesis to be some new method, it is very

likely that there is always some effect.

The meta-analysis by Hattie [27] suggests that instead of comparing between not having and having some specific schooling innovation, it is best to compare different innovations between each other. This way we have some relative measure that is more useful because we cannot incorporate every possible innovation and method in the teaching. It is also possible that some methods are mutually exclusive, like giving more homework versus no homework at all. We have to do some cherry picking and choose only ones that are the most efficient and we have resources to perform.

In the educational context, there exist some additional difficulties, like if we want to predict how well students will do on a course, the predictions are not very useful if we need to six weeks of data on a seven-week course. This might get very accurate predictions on course outcomes, but this does not allow educators to do intervention early enough. In the educational context, the predictions are pretty useless if they are accurate only at the very end of a course. There can also be ethical problems similar to medical research on testing some methods only on some specific group and giving them likely better treatment than to the others.

4.2 Variables in traditional schooling system

In the traditional school, many factors are affecting learning. One might think that it is only about student's ability to learn and teacher's ability to teach, but it is a much more complex system of variables and interactions. These variables include things like learning environment, current curriculum, other students, parents, and so on [27]. Almost everything interacts with each other in some manner, so the actual learning process is a product of many factors and thus hard to model well.

Many studies have been conducted, and slowly there starts to form a clearer image on what works and what kind of things should be tracked. Quite unsurprisingly, one of the best predictors of academic success are prior achievements and success. For example, success in high-school has a positive correlation with success in college [2]. Prior achievements and success seem to predict well all kind of academic success and even success later in work life. These two are good predictors starting as early as from preschool and the first years of schooling [27].

More specific examination prior achievements and success shows that early

reading and mathematical skills are good at predicting subsequent academic success [16, 22]. These two specific skills are also somehow connected overall as they correlate quite strongly [22] and studies about mathematical learning disabilities show a connection with problems in verbal skills [53].

One very important factor is the quality of the teacher. Most important factors being microteaching, teacher clarity, and teacher-student relationship skills [27]. Surprisingly, teacher's subject matter knowledge has not very much evidence of being important [27]. It is important to remember that most of these studies are considering elementary and high-school level, where very deep subject matter knowledge is not probably required.

Feedback has a very strong effect on learning, but it can be double-edged sword: good feedback can improve performance, but bad feedback might lower it [33]. It is not only the feedback from teacher to students, but also students' feedback to the teacher is required.

One very interesting fact is that students are usually good at recognizing their skill level and school performance [34]. So instead of trying to detect the level of knowledge by some complicated forms and questionnaires, it should be possible to just ask student's opinion.

Against the common beliefs that specific gender is better at some tasks, gender seems to have no notable effect [22]. This is also supported by the fact that, measured in cognitive variables, boys and girls are more similar than dissimilar [29].

There are surprisingly many articles that have shown that traditional metrics used to classify students are not suitable for that use [2, 64]. This shows the importance of validation, something that methods of machine learning can provide more efficiently than was previously possible.

4.3 Variables in programming courses

Variables affecting computer science skills have been studied a long time. Finding good predictors for it has not been an easy job [18]. In the beginning, most of the studies were measuring cognitive abilities [18], but recently the focus has been shifted more towards data-driven variables.

Most of the recent studies considering success in computer science focus on an introductory course in programming [30]. This is likely because it is one of the first courses taken and learning to program is the main goal in many introductory computer science courses [40].

There seem to be a lot of mixed results in the study of variables affecting general computer science skills. On explanation for this might be given by Ihantola et al. [30]. In their review, they note that most of the studies considering programming are not founded on formal theories and are mostly limited to a single institution only. Limiting to one institution only is kind of an interesting situation as the data collected should be easy to transfer from one institution to another. There can be many reasons for limiting to one institution only like privacy, incompatible environments and so on. This likely reflects the state of educational data mining: there seem to be no standard ways of collecting data, no shared software or mutually agreed best practices.

Instruments used in data acquisition in most studies are questionnaires, specific tests for some personal properties like cognitive ability and self-efficacy, and data-driven methods. There are basically three different levels of collecting data from programming behavior. The most fine-grained level of data being keystroke data, a little higher level is code snapshots, and the highest level is submissions.

From the point of view of a student, it probably is better to collect the data in a non-intrusive way like they do in [58]. In their study data based tests outperform almost all the traditional test-based features.

This kind of testing is something that is pretty unique for computer-based subjects. With computers, it is relatively easy to accumulate vast amounts of behavioral data by just looking at the behavior within some program or on a computer. This kind of cyberpsychology has been recently studied for marketing purposes [23], but there seems to be no reason that this type of data could not also be gathered for educational purposes. Of course, it is possible to do similar analysis also in traditional teaching using the methods of traditional psychology. The teacher could analyze the behavior of students at the same time when teaching, but when there can easily be over 20 students in one group, this is not probably by any means a practical thing to do.

There are many ways to evaluate user progress in programming. Previously programming course performance has been mostly tested with academic, cognitive, psychological and demographic variables [58, 60]. Testing of these kinds of properties can, for example, be done by surveys [60], by looking at previous academic success [60], by looking at previous programming expe-

rience [5], by doing behavioral study in lectures and lab sessions [49] or by looking at the code snapshots at particular events [31, 59].

Like very often in educational sciences [39], there happens only very few replication of studies [30] in the programming education studies. It seems to be that novelty is the trend in educational sciences, and the replication of important results is overlooked [39]. Replication should be done more often as paper by Ihantola et al. [30] shows that changing even a single part of the research can have an effect on the results.

Based on recent studies by Ahadi et al. [1] and Watson et al. [58], it seems that data-driven methods are good at predicting course outcomes. Features being the best predictors in the study by Ahadi et al. [1], contains mostly features generated from the data. For example, the number of steps in some programming exercise or the correctness of an exercise.

4.3.1 Background variables

Background variables give information from the student's previous behavior and can, therefore, be very good estimators on upcoming success also in computer science.

Previous programming experience seems to give very mixed results. Intuitively one would probably assume that there are clear benefits from previous experience. Nevertheless, some studies seems to find significant differences between students with and without any experience [58, 62, 61] yet some studies seem to suggest that previous programming experience does not matter [1]. Because every course is different, there can be many things affecting this.

Mathematical background also seems to give very mixed results. In the articles by Wilson et al. [62] and Werth [60] they find a correlation between computer science skills and mathematical skills, but then again in articles [1] and [51] there seems to be no significant correlation. Of course the context in the articles by Wilson et al. and Werth is more general than the programming specific focus on the articles by Ahadi et al. and Stein et al. One explanation for this is that mathematical skill is also quite hard to measure, and usually the metric is mathematics courses taken before the CS course under investigation. For example, people with low mathematical skills might take more courses to compensate the lacking base skill [60]. So where one good student needs only one math course, it takes more courses for the

less talented student to get to the same level of mathematical competence.

Comfort level in class seems to have a significant correlation with success in computer science courses [62, 5]. Because of how the comfort level was measured, it is also possible that success in the course leads to comfort. In article they defined the comfort level to be "a continuous variable derived from seven questions on the questionnaire regarding asking and answering questions in class, in lab, and during office hours; anxiety level while working on computer assignments; perceived difficulty of course; perceived understanding of concepts in the course as compared to classmates; and perceived difficulty of completing the programming assignments". So for example, if the student knew everything at the beginning of the course, the comfort levels would probably automatically be very high. In the other hand, students with no previous knowledge would probably struggle with the concepts and feel less comfortable. The comfort level studied in [62] is probably related to self-efficacy that is also suggested to be significant factor [61] for performance in programming courses.

Most of the studies seem to suggest that there is no significant correlation between gender and programming skills [62, 1], but there are also some small studies [5] that suggest opposite results. If comparing to the results in the traditional schooling system, gender should have no effect, as the same variables that should have an effect on programming ability, are quite similar within both genders [29]. Of course, learning is the product of many different factors so even minor changes can potentially have a huge effect on the results.

In addition to these major topics, like previous success and mathematical ability, also more minor things have been studied. For example, an excess amount of playing video games seems to lower programming abilities [62].

4.3.2 Data-driven variables

The huge advantage of data-driven methods is that at least some of them can be calculated almost instantly, and they are non-obtrusive. This way they can give instant knowledge about the progress of a student. When comparing to the traditional way where for example some teaching assistant first checks the work of student and gives them some grade and possibly feedback, this, for example, enables interventions to be more timely. Another clear advantage when comparing to traditional methods is that these metrics

are usually directly related to programming ability of the student and not via some intermediate variable like mathematical ability.

Data-driven variables can be as simple as exercises done in the first week or correctness of exercise. These kinds of variables are usually very easy to generate and to interpret. More abstract level variables can also be engineered from data. A few different metrics have been developed based on compiler behavior of a user. Two of the most used data-driven metrics are Jadud's [31] Error quotient and Watwin [59] that are both based on the same idea of analyzing compilation behavior via source code snapshots. One of the most recent metric is Normalized programming state model [10].

Calculation of Jadud's Error quotient (EQ) as described in article [31] is quite simple having four steps. Events are divided into consecutive pairs in chronological order, pairs are given score by the EQ algorithm, normalized and then summed. Finally, the score is divided by the number of pairs giving the final score.

Watwin is a score similar to EQ but tries to fix the issues that make EQ perform badly by incorporating time into the algorithm [59]. It is not as simple algorithm as Jadud's as it requires some additional preparation of the data. The score is basically telling how much time is used on error states. More error states results in a higher score. Both of these scores range from 0 to 1, where 0 is the best score and means that student made no errors.

The basic idea in both is that if a student uses a lot of time with code that is not compiling, that should be a sign of some difficulties. Jadud's algorithm does not use actual time but instead counts the number of steps taken. Evaluation of these scores has shown that these are only a weak predictors of performance in some contexts, where the programming environment behavior is not similar to one in the original study [47].

Based on the research on currently most used data-driven metrics, they seem to be still quite immature and context-dependent [47]. However, the potential in these data-driven methods is great, so it is likely only a question of time before these will outperform the traditional methods and better metrics are developed. These allow more efficient use of time, when compared, for example, to observational methods used by Rodrigo et al. [49] that required two people full-time recording behavior.

5 Empirical research

In the research two datasets were used with additional background information for each student. Both datasets contained information from seven weeks long introductory programming course of the University of Helsinki called "Introduction to programming" with slightly different student distributions, exercises, and grading. Datasets were collected from the programming environment using Test My Code server (TMC) [56].

In the actual analysis machine learning environment called scikit-learn (sklearn) [46] for Python is used. All the traditional machine learning algorithms used are implemented in this library.

5.1 Implementation details

From the traditional machine learning methods k-Nearest Neighbors, Support Vector Machine, Random Forest, AdaBoost, and Gaussian Naive Bayes were chosen. For specific classes and parameters see table 1. As the TrAdaBoost was not available in the scikit-learn library, the algorithm was implemented by the author of this thesis in Python as described in the article by Dai et al. [15].

The analysis was run so that all the data were used. The source data was split into pieces of 25, 50 and 100 samples and then averaged over the results. Same was also done to the target data, but with sample sizes of 5, 10 and 25. This resulted in the nine different combinations of selections from source and target data. All the sample sizes from 5 to 200 with step size of 5 were originally tested, but it just generated much more data and did not contain any information that could not be seen from the much smaller set of sample sizes.

Metrics used for the analysis were also implemented already in sklearn. These metrics were receiver operating characteristic area under the curve (AUC in the table 5) and F1 score.

For feature selection, sklearn's SelectFromModel class was used. It takes a classifier as a parameter and selects features based on importance weights. It optimizes for the feature that is to be predicted. For all the classifiers feature selection was done with the same classifier given as a parameter for the SelectFromModel, except for TrAdaBoost the traditional AdaBoost was used as a parameter.

5.2 Context

Data was prepared by augmenting the background info to the two main datasets that were collected using the TMC environment. Missing values were mapped to zeroes and feature vectors were normalized by subtracting minimum and dividing by maximum to have a range between 0 and 1. This normalization method is usually called min-max normalization [45].

A few features from the background information were cleaned and quantified by hand as in the original data there were a few free form questionnaire questions. Features cleaned up by hand were previous programming experience, is currently working, has CS study right, a level of education, and gender. All of these were transformed to binary features.

The course related to the first dataset had grading from 0 to 5. Grades from 1 to 5 were considered to be passing grades and a new fail-pass binary feature was generated based on the grade. In the course related to the second dataset the grading was already a fail-pass grading, so no alterations were done to it.

The first dataset was from spring 2014 course containing data from 247 students. The majority of students were computer science majors. The second dataset was from autumn 2015 course containing data from 101 students containing only a few computer science majors. Both of the datasets were collected using TMC, background information came from questionnaires and course results came from the student register.

After combining all the background information to the exercise data, both of these datasets were then modified to contain same feature set. For features contained in the final datasets see table 5.2. As the datasets were already previously collected, some features that might have been interesting, could not be included, like previous achievement or mathematical background.

Predicted value was the result of the course. In the first dataset, 193 students passed the course and 54 failed. In the second dataset, 66 students

Classifier	**Implementation**	**Parameters**
k-Nearest Neighbors	KNeighborsClassifier	k=3
Support Vector Machine	SVC	kernel="linear", C=2.0
Random Forest	RandomForestClassifier	max_depth=5, n_estimators=10, max_features=1
AdaBoost	AdaBoostClassifier	None
Gaussian Naive Bayes	GaussianNB	None

Table 1: Used classes and parameters

Background features	has CS study right has previous Java experience programming experience: most lines programming experience: hours gender year of birth working hours per week is currently working level of education: bachelor's degree level of education: university of applied sciences graduate level of education: master's degree level of education: licentiate's or doctor's degree level of education: elementary school level of education: upper secondary school course result
Data features	total events on week 1 compiling states on week 1 completed exercises on week 1 total events on week 2 compiling states on week 2 completed exercises on week 2 total events on week 3 compiling states on week 3 completed exercises on week 3 total events on week 4 compiling states on week 4 completed exercises on week 4 total events on week 5 compiling states on week 5 completed exercises on week 5 total events on week 6 compiling states on week 6 completed exercises on week 6 total events on week 7 compiling states on week 7 completed exercises on week 7

Table 2: Final feature set for the coarse-grained data.

passed and 45 failed.

In addition to the heavily aggregated datasets, also more fine-grained data from only first week was tested. Results can be seen in table 4. In that dataset manual mapping was done so that exercises and their contents matched between the course instances. Exercises that had no corresponding exercise on the other course instance were removed. Data from 29 exercises was included in the dataset. The dataset contained the same background features as the coarse-grained, but data features contained events, numbers of compiling states and maximum correctness score for each exercise. This manual mapping was done only for the first week, because this kind of manual mapping is out of the scope of this thesis.

5.3 Results

In this section, most interesting results from the point of view of this thesis are summarized. Only first week's results are shown on the table 3 to keep the representation concise. Full analysis results can be found from appendix A. On table 4 the results from more fine-grained dataset can be seen.

The first week is chosen because that is the most interesting from the point of view of educational context. The best results per setting are on bold and best results overall are underlined. In addition to tables, line diagrams (figures 9 and 10) summarizing all the scores over the weeks are given so that results and the trend of scores overall are easy to see. All line diagrams have the range from 0.5 to 1.0 for clarity reasons, as no score is under the 0.5 threshold.

	T5 S25		T5 S50		T5 S100	
	F1	AUC	F1	AUC	F1	AUC
KNeighborsClassifier	0.801	0.597	0.774	0.583	0.800	0.605
SVC	0.764	0.547	0.789	0.561	0.793	0.565
RandomForestClassifier	0.795	0.615	0.803	0.620	0.815	0.614
AdaBoostClassifier	0.782	0.614	0.772	0.594	0.799	0.622
GaussianNB	0.773	0.622	0.801	0.619	0.800	0.615
TrAdaBoost	**0.806**	**0.650**	**0.817**	**0.647**	**0.827**	**0.673**
	T10 S25		T10 S50		T10 S100	
	F1	AUC	F1	AUC	F1	AUC
KNeighborsClassifier	0.801	0.589	0.774	0.587	0.797	0.610
SVC	0.768	0.548	0.787	0.562	0.786	0.562
RandomForestClassifier	0.797	0.620	0.797	0.611	0.806	0.607
AdaBoostClassifier	0.775	0.612	0.768	0.595	0.789	0.612
GaussianNB	0.719	0.608	0.797	0.619	0.798	0.616
TrAdaBoost	**0.807**	**0.640**	**0.815**	**0.646**	**0.823**	**0.668**
	T25 S25		T25 S50		T25 S100	
	F1	AUC	F1	AUC	F1	AUC
KNeighborsClassifier	0.802	0.624	0.782	0.600	0.793	0.599
SVC	0.784	0.556	0.796	0.574	0.796	0.563
RandomForestClassifier	0.795	0.611	0.803	0.617	0.812	0.612
AdaBoostClassifier	0.766	0.604	0.764	0.598	0.785	0.609
GaussianNB	0.794	0.634	0.798	0.624	0.806	0.621
TrAdaBoost	**0.813**	**0.639**	**0.815**	**0.646**	**0.826**	**0.676**

Table 3: Table showing first week's results for coarse-grained data. Best results from each metric on each setting is bolded and the best results where same amount of target data is used are underlined.

	T5 S25		T5 S50		T5 S100	
	F1	AUC	F1	AUC	F1	AUC
KNeighborsClassifier	0.781	0.595	**0.804**	0.642	0.799	0.617
SVC	**0.794**	**0.626**	0.791	**0.643**	0.777	0.622
RandomForestClassifier	0.786	0.602	0.785	0.594	0.807	0.603
AdaBoostClassifier	0.741	0.592	0.723	0.574	0.732	0.594
GaussianNB	0.590	0.584	0.663	0.599	0.661	0.630
TrAdaBoost	0.760	0.602	0.785	0.609	**0.830**	**0.642**
	T10 S25		T10 S50		T10 S100	
	F1	AUC	F1	AUC	F1	AUC
KNeighborsClassifier	0.787	0.585	0.793	0.622	0.800	0.624
SVC	**0.801**	**0.631**	**0.802**	**0.641**	0.785	0.624
RandomForestClassifier	0.786	0.577	0.799	0.594	0.809	0.598
AdaBoostClassifier	0.756	0.600	0.758	0.584	0.763	0.598
GaussianNB	0.661	0.578	0.711	0.606	0.712	**0.640**
TrAdaBoost	0.767	0.600	0.795	0.606	**0.823**	0.636
	T25 S25		T25 S50		T25 S100	
	F1	AUC	F1	AUC	F1	AUC
KNeighborsClassifier	0.781	0.583	0.801	**0.647**	0.792	0.619
SVC	**0.816**	**0.655**	0.802	**0.647**	0.798	0.635
RandomForestClassifier	0.801	0.601	0.803	0.600	0.814	0.602
AdaBoostClassifier	0.760	0.601	0.753	0.591	0.766	0.607
GaussianNB	0.680	0.591	0.730	0.639	0.730	0.650
TrAdaBoost	0.799	0.607	**0.807**	0.626	**0.829**	**0.664**

Table 4: Table showing first week's results for fine-grained data. Best results from each metric on each setting is bolded and the best results where same amount of target data is used are underlined.

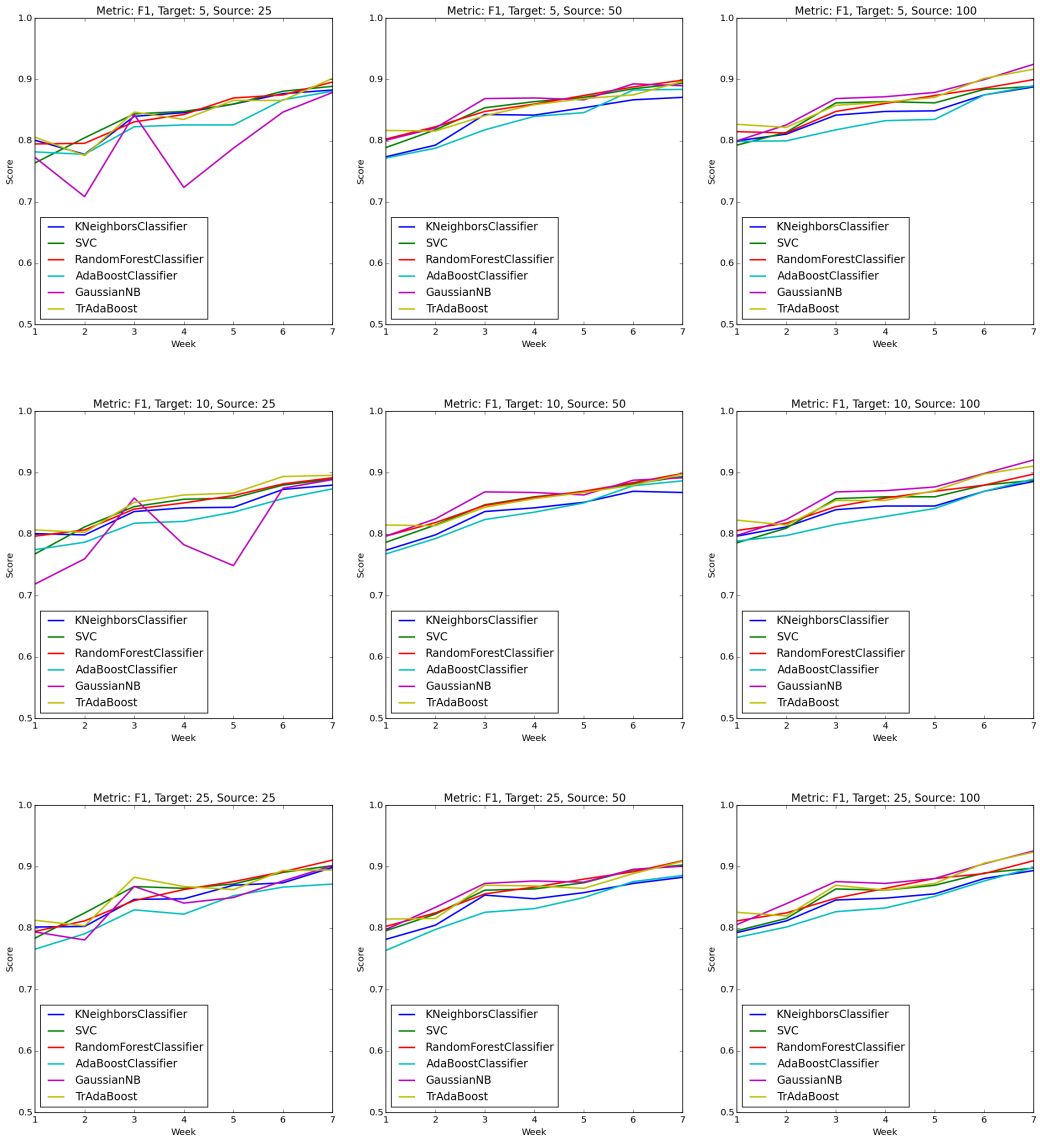

Figure 9: F1 scores of weekly data.

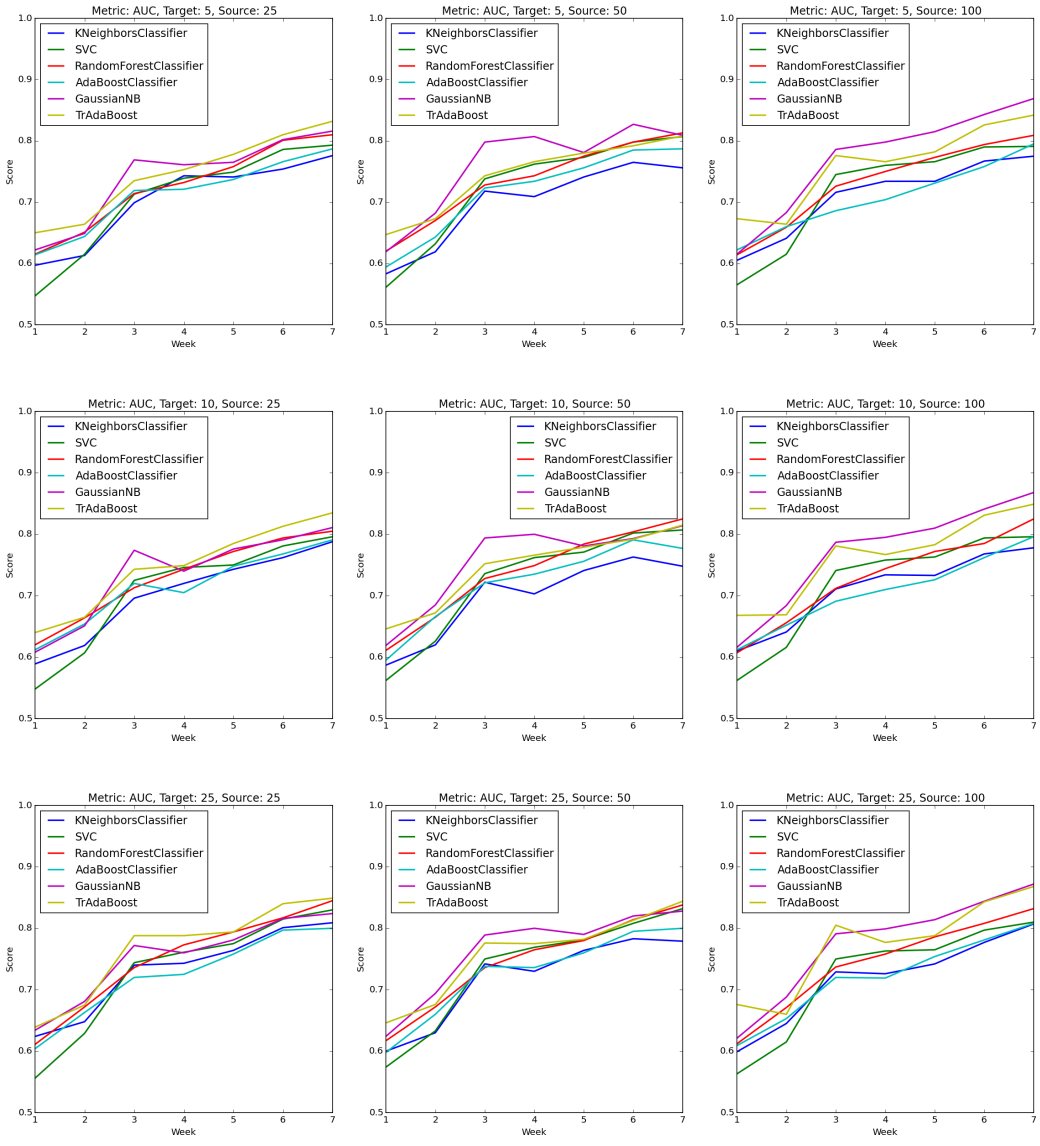

Figure 10: AUROC scores of weekly data.

6 Discussion

The discussion focuses mostly on the coarse-grained setting where weekly data is used, as that was the main focus of the thesis. The discussion considering fine-grained data results is left to the future work section.

The end results were quite close to what was expected in a sense that TrAdaBoost outperformed the others in almost every occasion. A little unexpectedly the difference between traditional and transfer learning methods was quite small, and every classifier seems to be almost equally good. The only classifier having somewhat inconsistent and surprisingly good results was Gaussian Naive Bayes. As it assumes every variable to have a Gaussian distribution, it is the most vulnerable to features selected, which likely explains its behavior.

From the educational point of view performance of TrAdaBoost is very positive as it outperforms every other classifier in almost every setting when looking at the performance in the first weeks. Because of this effect, its usage can be justified in the educational setting, even though the improvement is not very significant.

Overall predictions get mostly more accurate as the number of features grows. This is likely due how the feature engineering was done: every new week creates additional features and so the new features give potentially more accurate information. From the educational point of view this is problematic because we want to do interventions as early as possible and if we are accurate only in the last weeks of the course, early intervention is not possible. In the selection of best classifier in an educational context, some penalty should be considered for classifiers that require knowledge from the end of the course. This could be similar to the penalty given for complex models, for example, based on the number of features used, that tend to overfit data.

When looking at the AUROC scores, the results suggest that none of the classifiers are doing very well during the first weeks. All of the scores are much closer to 0.5 that is random guessing than to 1 that is perfect prediction performance. TrAdaBoost reaches the highest score, and even that is only 0.676. If considering that 0.5 is random and 1 is perfect then, at least a score over 0.75 should be expected, but those scores are reached only after week two.

In the prediction task, an imbalanced dataset was used. This imbalance

might explain the low scores of AUROC metric. With few additional steps, the task could have been made somewhat easier, for example, by dividing the dataset by median of grade to two groups of equal size. As this kind of imbalance is almost always present in this kind of data, it might not be well justified to do this kind of artificial division.

As all the rows from the original dataset were used, there might have been some rows creating unnecessary noise. For example, students that did all the exercises, but did not attend to exam or students that dropped out after two or three weeks. As these are quite clear outliers, it should be that most of the learning algorithms should be able to automatically bypass these as noise.

6.1 Research questions

Is there significant help from transfer learning in educational context when compared to traditional machine learning methods? Like expected the TrAdaBoost algorithm was able to use the data from the previous course more efficiently. When compared to traditional methods, the most significant help from transfer was in the first weeks. After the first weeks, the difference starts to decrease, and traditional methods start to do equally well.

When comparing TrAdaBoost to the closely related AdaBoost, the positive effect of transfer learning is the most concrete, TrAdaBoost beats the traditional version in every occasion.

It seems that if we have only a little data from the new context, we should always use transfer learning, but otherwise, the more mature traditional machine learning methods are an equally good choice.

These results do not support the assumption that traditional machine learning does not work well in different contexts as they are quite close to results of TrAdaBoost. Of course, it might be that the contexts are too similar to transfer learning to really make a significant difference.

How much previous data do we need? The more, the better seems to be the answer, even with the traditional machine learning methods. The TrAdaBoost algorithm seemed to gain most from the increase in previous data, but also traditional methods increased in performance. The increase in TrAdaBoost performance is expected as it is designed in such way that it should not at least get worse. It is quite surprising that also traditional

methods get better as more data from a different context is given, suggesting that even with these methods negative transfer is not happening.

How much labeled data is needed from the new task? Quite naturally all the methods perform better when more data from the target task is given. The gap between the TrAdaBoost and the traditional methods seems to stay almost the same when we grow the amount of the target data. It seems that the target instances are giving as much information as they can to aid the transfer, already in the smallest setting of five target instances. Like already stated in the discussion of the first question, it might be that the contexts are actually quite similar and target data does not give much more information.

Are the predictions accurate already on the first week or does it, for example, get better after a certain number of weeks? Overall the performance of all the methods is not very good when using only data from the first weeks. The best algorithm, TrAdaBoost reaches F1 score of 0.827, but only with AUROC score of 0.673. Even with great quantities of data from the previous course and the target course. The studied TrAdaBoost algorithm seems to have the best results and gives a minor boost to prediction performance when compared to other, even with a quite small amount of previous data.

On week three, all the scores take a significant leap. From educational point of view this might be too late to detect the students in need for assistance. It is also possible that the results get more accurate for the same reason the assistance is too late. The weaker students have already dropped the course and thus create less noise on the data.

6.2 Future work

The power of transfer learning methods in other settings, such as in indoor localization and brain-computer interfaces, is most likely because they have been modified and adapted to those situations. Similar context dependent modifications could likely be done also in educational context, to gain better performance. In addition to modifications to algorithms, also better feature engineering can likely be done as very fine-grained data is available for that.

In the study, only one previous course was used as a source task, but more data could be accumulated if more courses would be added. In that case, TrAdaBoost is no more sufficient as it handles data only from one

source dataset, so more advanced algorithms like TransferBoost [17] or MultiSourceTrAdaBoost [66] should be used. Also more exhaustive testing on traditional machine learning methods and transfer learning methods should be done, as this study tested only a few of the methods. It might be that some different methods in machine learning are more suitable for this kind of data.

Data used in the study was already collected before the study was done, so it created some limitations for choosing the features. Only weekly data on exercises could be transferred without manually mapping the exercises, as otherwise the feature sets would not match between the source and target sets. This seems to be this built-in problem within many supervised transfer learning methods that they assume equivalent feature sets. There exist some methods for doing automatic domain adaptation, for example, using meta-features. One of these methods is called heterogeneous domain adaptation. By using these to map the feature sets between the target domain and source domains, it could be possible to reach better results as more data is usable. One should try these domain adaptation and representation learning methods to maximize the data input and to gain greater advantage over the traditional machine learning methods. Representation learning has already been shown to have good performance on transfer learning problems [3]. Results with fine-grained data on table 4 also show that more features available means more accurate results. When comparing those results with the results from more coarse-grained dataset, the improvement is not quite significant though.

As this thesis was mostly done from the point of view of computer science, a different approach could also be taken. In an educational context, the main motivation should not be the prediction of outcomes, but to measure if a student has actually learned the concepts. This requires steps beyond just optimizing algorithms for outcome prediction. Because of the many factors in the learning process and complex interactions between them, more complex models should probably be used in the modeling. A good starting point could be probabilistic models, where we can incorporate the uncertainty to our model and model the interactions between variables. These models allow us to reason also about the causes in contrast to only predicting outcomes.

These probabilistic model incorporated with features generated from data, like programming and debugging behavior, might provide better insight on the actual skill level and programming ability.

References

[1] Ahadi, Alireza, Lister, Raymond, Haapala, Heikki, and Vihavainen, Arto: *Exploring machine learning methods to automatically identify students in need of assistance.* In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, pages 121–130, New York, NY, USA, 2015. ACM.

[2] Belfield, Clive and Crosta, Peter Michael: *Predicting success in college: The importance of placement tests and high school transcripts.* 2012.

[3] Bengio, Yoshua, Courville, Aaron, and Vincent, Pierre: *Representation learning: A review and new perspectives.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 35(8):1798–1828, 2013.

[4] Bennett, Paul N, Dumais, Susan T, and Horvitz, Eric: *Inductive transfer for text classification using generalized reliability indicators.* In *Proceedings of the ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data*, 2003.

[5] Bergin, Susan and Reilly, Ronan: *Programming: factors that influence success.* In *ACM SIGCSE Bulletin*, volume 37, pages 411–415. ACM, 2005.

[6] Boser, Bernhard E, Guyon, Isabelle M, and Vapnik, Vladimir N: *A training algorithm for optimal margin classifiers.* In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[7] Boyer, Sebastien and Veeramachaneni, Kalyan: *Transfer learning for predictive models in massive open online courses.* In *Artificial Intelligence in Education*, pages 54–63. Springer, 2015.

[8] Breiman, Leo: *Random forests.* Machine learning, 45(1):5–32, 2001.

[9] Brinton, Christopher G, Chiang, Mung, Jain, Sonal, Lam, HK, Liu, Zhenming, and Wong, Felix Ming Fai: *Learning about social learning in moocs: From statistical analysis to generative model.* Learning Technologies, IEEE Transactions on, 7(4):346–359, 2014.

[10] Carter, Adam S, Hundhausen, Christopher D, and Adesope, Olusola: *The normalized programming state model: Predicting student performance in computing courses based on programming behavior*. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, pages 141–150. ACM, 2015.

[11] Cetintas, Suleyman, Si, Luo, Xin, Yan Ping, and Hord, Casey: *Automatic detection of off-task behaviors in intelligent tutoring systems with machine learning techniques*. Learning Technologies, IEEE Transactions on, 3(3):228–236, 2010.

[12] Commons, Wikimedia: *Graphic showing the maximum separating hyperplane and the margin.*, 2008. `https://upload.wikimedia.org/wikipedia/commons/2/2a/Svm_max_sep_hyperplane_with_margin.png`.

[13] Cunningham, Padraig and Delany, Sarah Jane: *k-nearest neighbour classifiers*. Multiple Classifier Systems, pages 1–17, 2007.

[14] Curtis, Jonathan *et al.*: *Dropout prediction.* 1983.

[15] Dai, Wenyuan, Yang, Qiang, Xue, Gui Rong, and Yu, Yong: *Boosting for transfer learning*. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.

[16] Duncan, Greg J, Dowsett, Chantelle J, Claessens, Amy, Magnuson, Katherine, Huston, Aletha C, Klebanov, Pamela, Pagani, Linda S, Feinstein, Leon, Engel, Mimi, Brooks-Gunn, Jeanne, *et al.*: *School readiness and later achievement*. Developmental Psychology, 43(6):1428–1446, 2007.

[17] Eaton, Eric and desJardins, Marie: *Selective transfer between learning tasks using task-based boosting.* In *AAAI*, 2011.

[18] Evans, Gerald E and Simkin, Mark G: *What best predicts computer proficiency?* Communications of the ACM, 32(11):1322–1327, 1989.

[19] Fawcett, Tom: *An introduction to roc analysis*. Pattern recognition letters, 27(8):861–874, 2006.

[20] Flach, Peter: *Machine learning: the art and science of algorithms that make sense of data.* Cambridge University Press, 2012.

[21] Freund, Yoav and Schapire, Robert E: *A decision-theoretic generalization of on-line learning and an application to boosting.* Journal of computer and system sciences, 55(1):119–139, 1997.

[22] Goddard, Roger D: *Relational networks, social 'trust, and norms: A social capital perspective on students' chances of academic success.* Educational Evaluation and Policy Analysis, 25(1):P11–59, 2003.

[23] Guan, Zengda and Zhu, Tingshao: *An overview of transfer learning and computational cyberpsychology.* In *Pervasive Computing and the Networked World*, pages 209–215. Springer, 2012.

[24] Guyon, Isabelle and Elisseeff, André: *An introduction to variable and feature selection.* The Journal of Machine Learning Research, 3:1157–1182, 2003.

[25] Hall, Mark A: *Correlation-based feature selection for machine learning.* PhD thesis, The University of Waikato, 1999.

[26] Hämäläinen, Wilhelmiina and Vinni, Mikko: *Intelligent Tutoring Systems: 8th International Conference, ITS 2006, Jhongli, Taiwan, June 26-30, 2006. Proceedings*, chapter Comparison of Machine Learning Methods for Intelligent Tutoring Systems, pages 525–534. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, ISBN 978-3-540-35160-3.

[27] Hattie, John: *Visible learning: A synthesis of over 800 meta-analyses relating to achievement.* Routledge, 2013.

[28] Hua, Jianping, Xiong, Zixiang, Lowey, James, Suh, Edward, and Dougherty, Edward R: *Optimal number of features as a function of sample size for various classification rules.* Bioinformatics, 21(8):1509–1515, 2005.

[29] Hyde, Janet Shibley: *The gender similarities hypothesis.* American psychologist, 60(6):581, 2005.

[30] Ihantola, Petri, Vihavainen, Arto, Ahadi, Alireza, Butler, Matthew, Börstler, Jürgen, Edwards, Stephen H, Isohanni, Essi, Korhonen, Ari,

Petersen, Andrew, Rivers, Kelly, *et al.*: *Educational data mining and learning analytics in programming: Literature review and case studies.*

[31] Jadud, Matthew C.: *Methods and tools for exploring novice compilation behaviour.* In *Proceedings of the Second International Workshop on Computing Education Research*, ICER '06, pages 73–84, New York, NY, USA, 2006. ACM, ISBN 1-59593-494-4.

[32] Kizilcec, René F, Piech, Chris, and Schneider, Emily: *Deconstructing disengagement: analyzing learner subpopulations in massive open online courses.* In *Proceedings of the third international conference on learning analytics and knowledge*, pages 170–179. ACM, 2013.

[33] Kluger, Avraham N and DeNisi, Angelo: *The effects of feedback interventions on performance: a historical review, a meta-analysis, and a preliminary feedback intervention theory.* Psychological bulletin, 119(2):254, 1996.

[34] Kuncel, Nathan R, Credé, Marcus, and Thomas, Lisa L: *The validity of self-reported grade point averages, class ranks, and test scores: A meta-analysis and review of the literature.* Review of educational research, 75(1):63–82, 2005.

[35] Lin, Yi and Jeon, Yongho: *Random forests and adaptive nearest neighbors.* Journal of the American Statistical Association, 101(474):578–590, 2006.

[36] Lotte, Fabien and Guan, Cuntai: *Learning from other subjects helps reducing brain-computer interface calibration time.* In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 614–617. IEEE, 2010.

[37] Lu, Jie, Behbood, Vahid, Hao, Peng, Zuo, Hua, Xue, Shan, and Zhang, Guangquan: *Transfer learning using computational intelligence: A survey.* Knowledge-Based Systems, 80:14–23, 2015.

[38] Lykourentzou, Ioanna, Giannoukos, Ioannis, Nikolopoulos, Vassilis, Mpardis, George, and Loumos, Vassili: *Dropout prediction in e-learning courses through the combination of machine learning techniques.* Computers & Education, 53(3):950–965, 2009.

[39] Makel, Matthew C and Plucker, Jonathan A: *Facts are more important than novelty replication in the education sciences.* Educational Researcher, page 0013189X14545513, 2014.

[40] McCracken, Michael, Almstrum, Vicki, Diaz, Danny, Guzdial, Mark, Hagan, Dianne, Kolikant, Yifat Ben David, Laxer, Cary, Thomas, Lynda, Utting, Ian, and Wilusz, Tadeusz: *A multi-national, multi-institutional study of assessment of programming skills of first-year cs students.* ACM SIGCSE Bulletin, 33(4):125–180, 2001.

[41] Merriam-Webster: *"learning."*, January 2016. `http://www.merriam-webster.com/dictionary/learning`.

[42] Mihalkova, Lilyana, Huynh, Tuyen, and Mooney, Raymond J: *Mapping and revising markov logic networks for transfer learning.* In *AAAI*, volume 7, pages 608–614, 2007.

[43] Nanni, Loris and Lumini, Alessandra: *Ensemble generation and feature selection for the identification of students with learning disabilities.* Expert Systems with Applications, 36(2, Part 2):3896 – 3900, 2009, ISSN 0957-4174.

[44] Pan, Sinno Jialin and Yang, Qiang: *A survey on transfer learning.* Knowledge and Data Engineering, IEEE Transactions on, 22(10):1345–1359, 2010.

[45] Patro, S and Sahu, Kishore Kumar: *Normalization: A preprocessing stage.* arXiv preprint arXiv:1503.06462, 2015.

[46] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E.: *Scikit-learn: Machine learning in Python.* Journal of Machine Learning Research, 12:2825–2830, 2011.

[47] Petersen, Andrew, Spacco, Jaime, and Vihavainen, Arto: *An exploration of error quotient in multiple contexts.* In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, pages 77–86. ACM, 2015.

[48] Piech, Chris, Sahami, Mehran, Koller, Daphne, Cooper, Steve, and Blikstein, Paulo: *Modeling how students learn to program.* In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 153–160. ACM, 2012.

[49] Rodrigo, Ma Mercedes T, Baker, Ryan S, Jadud, Matthew C, Amarra, Anna Christine M, Dy, Thomas, Espejo-Lahoz, Maria Beatriz V, Lim, Sheryl Ann L, Pascua, Sheila AMS, Sugay, Jessica O, and Tabanao, Emily S: *Affective and behavioral predictors of novice programmer achievement.* In *ACM SIGCSE Bulletin*, volume 41, pages 156–160. ACM, 2009.

[50] Schapire, Robert E: *Explaining adaboost.* In *Empirical inference*, pages 37–52. Springer, 2013.

[51] Stein, Michael V: *Mathematical preparation as a basis for success in cs-ii.* Journal of Computing Sciences in Colleges, 17(4):28–38, 2002.

[52] Sun, Zhuo, Chen, Yiqiang, Qi, Juan, and Liu, Junfa: *Adaptive localization through transfer learning in indoor wi-fi environment.* In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*, pages 331–336. IEEE, 2008.

[53] Swanson, H Lee and Jerman, Olga: *Math disabilities: A selective meta-analysis of the literature.* Review of Educational Research, 76(2):249–274, 2006.

[54] Taylor, Colin, Veeramachaneni, Kalyan, and O'Reilly, Una May: *Likely to stop? predicting stopout in massive open online courses.* arXiv preprint arXiv:1408.3382, 2014.

[55] Vigen, Tyler: *Spurious correlations.* `http://tylervigen.com/spurious-correlations`.

[56] Vihavainen, Arto, Vikberg, Thomas, Luukkainen, Matti, and Pärtel, Martin: *Scaffolding students' learning using test my code.* In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 117–122. ACM, 2013.

[57] Wang, Peitao, Lu, Jun, Zhang, Bin, and Tang, Zeng: *A review on transfer learning for brain-computer interface classification.* In *Information*

Science and Technology (ICIST), 2015 5th International Conference on, pages 315–322, April 2015.

[58] Watson, Christopher, Li, Frederick W.B., and Godwin, Jamie L.: *No tests required: Comparing traditional and dynamic predictors of programming success.* In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 469–474, New York, NY, USA, 2014. ACM, ISBN 978-1-4503-2605-6.

[59] Watson, Craig, Li, Frederick WB, and Godwin, Jamie L: *Predicting performance in an introductory programming course by logging and analyzing student programming behavior.* In *Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on*, pages 319–323. IEEE, 2013.

[60] Werth, Laurie Honour: *Predicting student performance in a beginning computer science class*, volume 18. ACM, 1986.

[61] Wiedenbeck, Susan: *Factors affecting the success of non-majors in learning to program.* In *Proceedings of the First International Workshop on Computing Education Research*, ICER '05, pages 13–24, New York, NY, USA, 2005. ACM, ISBN 1-59593-043-4.

[62] Wilson, Brenda Cantwell and Shrock, Sharon: *Contributing to success in an introductory computer science course: A study of twelve factors.* SIGCSE Bull., 33(1):184–188, February 2001, ISSN 0097-8418.

[63] Witten, Ian H., Frank, Eibe, and Hall, Mark A.: *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011, ISBN 0123748569, 9780123748560.

[64] Wu, Tung Kuang, Huang, Shian Chang, and Meng, Ying Ru: *Evaluation of ANN and SVM Classifiers As Predictors to the Diagnosis of Students with Learning Disabilities.* Expert Syst. Appl., 34(3):1846–1856, April 2008, ISSN 0957-4174.

[65] Xu, Qian and Yang, Qiang: *A survey of transfer and multitask learning in bioinformatics.* Journal of Computing Science and Engineering, 5(3):257–268, 2011.

[66] Yao, Yi and Doretto, Gianfranco: *Boosting for transfer learning with multiple sources.* In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1855–1862. IEEE, 2010.

A Full analysis results

Table 5: Data from both target and source tasks

Target	Source	Classifier	Metric	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
5	25	KNeighborsClassifier	F1	0.801	0.778	0.840	0.846	0.860	0.877	0.883
5	25	SVC	F1	0.764	**0.805**	0.844	**0.848**	0.860	**0.881**	0.889
5	25	RandomForestClassifier	F1	0.795	0.796	0.831	0.843	**0.870**	0.875	0.896
5	25	AdaBoostClassifier	F1	0.782	0.778	0.823	0.826	0.826	0.867	0.881
5	25	GaussianNB	F1	0.773	0.709	0.843	0.724	0.788	0.847	0.879
5	25	TrAdaBoost	F1	**0.806**	0.776	**0.847**	0.835	0.866	0.866	**0.902**
5	50	KNeighborsClassifier	F1	0.774	0.793	0.843	0.842	0.854	0.867	0.871
5	50	SVC	F1	0.789	0.818	0.854	0.864	0.871	0.885	0.894
5	50	RandomForestClassifier	F1	0.803	**0.823**	0.848	0.860	**0.874**	0.888	**0.899**
5	50	AdaBoostClassifier	F1	0.772	0.788	0.818	0.840	0.846	0.883	0.884
5	50	GaussianNB	F1	0.801	0.821	**0.869**	**0.870**	0.867	**0.893**	0.890
5	50	TrAdaBoost	F1	**0.817**	0.816	0.841	0.859	0.869	0.875	0.897
5	100	KNeighborsClassifier	F1	0.800	0.811	0.842	0.848	0.849	0.875	0.888
5	100	SVC	F1	0.793	0.814	0.862	0.864	0.862	0.884	0.889
5	100	RandomForestClassifier	F1	0.815	0.813	0.848	0.861	0.874	0.886	0.900
5	100	AdaBoostClassifier	F1	0.799	0.800	0.818	0.833	0.835	0.875	0.890
5	100	GaussianNB	F1	0.800	**0.826**	**0.869**	**0.872**	**0.879**	0.900	**0.925**
5	100	TrAdaBoost	F1	**0.827**	0.822	0.858	0.863	0.871	**0.902**	0.917
10	25	KNeighborsClassifier	F1	0.801	0.799	0.837	0.843	0.844	0.873	0.880
10	25	SVC	F1	0.768	**0.812**	0.845	0.857	0.859	0.880	0.890
10	25	RandomForestClassifier	F1	0.797	0.807	0.841	0.851	0.863	0.882	0.892
10	25	AdaBoostClassifier	F1	0.775	0.787	0.818	0.821	0.836	0.858	0.874
10	25	GaussianNB	F1	0.719	0.760	**0.859**	0.783	0.749	0.875	0.889
10	25	TrAdaBoost	F1	**0.807**	0.803	0.852	**0.864**	**0.867**	**0.894**	**0.896**
10	50	KNeighborsClassifier	F1	0.774	0.799	0.837	0.843	0.852	0.870	0.868
10	50	SVC	F1	0.787	0.815	0.848	0.861	0.869	0.882	0.894
10	50	RandomForestClassifier	F1	0.797	0.819	0.847	0.860	**0.870**	0.884	**0.899**
10	50	AdaBoostClassifier	F1	0.768	0.793	0.824	0.836	0.851	0.879	0.887
10	50	GaussianNB	F1	0.797	**0.825**	**0.869**	**0.868**	0.864	**0.888**	0.892
10	50	TrAdaBoost	F1	**0.815**	0.814	0.844	0.858	0.868	0.880	0.898

Continued on next page

Table 5 – continued from previous page

Target	Source	Classifier	Metric	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
10	100	KNeighborsClassifier	F1	0.797	0.812	0.840	0.846	0.846	0.870	0.886
10	100	SVC	F1	0.786	0.810	0.858	0.861	0.861	0.880	0.888
10	100	RandomForestClassifier	F1	0.806	0.818	0.845	0.859	0.870	0.880	0.898
10	100	AdaBoostClassifier	F1	0.789	0.798	0.816	0.829	0.842	0.870	0.890
10	100	GaussianNB	F1	0.798	**0.824**	**0.869**	**0.871**	**0.877**	**0.899**	**0.921**
10	100	TrAdaBoost	F1	**0.823**	0.815	0.855	0.855	0.871	0.898	0.911
25	25	KNeighborsClassifier	F1	0.802	0.803	0.847	0.848	0.870	0.874	0.899
25	25	SVC	F1	0.784	**0.825**	0.868	0.865	0.872	0.891	0.902
25	25	RandomForestClassifier	F1	0.795	0.812	0.845	0.863	**0.876**	0.892	**0.911**
25	25	AdaBoostClassifier	F1	0.766	0.791	0.830	0.823	0.853	0.867	0.872
25	25	GaussianNB	F1	0.794	0.781	0.868	0.841	0.850	0.877	0.902
25	25	TrAdaBoost	F1	**0.813**	0.804	**0.883**	**0.868**	0.863	**0.894**	0.895
25	50	KNeighborsClassifier	F1	0.782	0.805	0.854	0.848	0.858	0.873	0.883
25	50	SVC	F1	0.796	0.823	0.862	0.864	0.874	0.895	0.903
25	50	RandomForestClassifier	F1	0.803	0.825	0.856	0.867	**0.880**	0.892	**0.910**
25	50	AdaBoostClassifier	F1	0.764	0.798	0.826	0.832	0.850	0.876	0.886
25	50	GaussianNB	F1	0.798	**0.834**	**0.873**	**0.877**	0.875	**0.896**	0.901
25	50	TrAdaBoost	F1	**0.815**	0.816	0.870	0.869	0.865	0.889	0.909
25	100	KNeighborsClassifier	F1	0.793	0.812	0.846	0.849	0.856	0.881	0.894
25	100	SVC	F1	0.796	0.816	0.864	0.862	0.870	0.890	0.898
25	100	RandomForestClassifier	F1	0.812	0.825	0.849	0.865	**0.881**	0.889	0.910
25	100	AdaBoostClassifier	F1	0.785	0.802	0.827	0.833	0.852	0.877	0.899
25	100	GaussianNB	F1	0.806	**0.840**	**0.876**	**0.873**	**0.881**	0.905	**0.926**
25	100	TrAdaBoost	F1	**0.826**	0.820	0.870	0.862	0.873	**0.906**	0.924
5	25	KNeighborsClassifier	AUC	0.597	0.613	0.699	0.743	0.741	0.754	0.776
5	25	SVC	AUC	0.547	0.615	0.713	0.739	0.749	0.786	0.793
5	25	RandomForestClassifier	AUC	0.615	0.651	0.714	0.732	0.758	0.801	0.810
5	25	AdaBoostClassifier	AUC	0.614	0.644	0.719	0.721	0.737	0.766	0.787
5	25	GaussianNB	AUC	0.622	0.649	**0.769**	**0.761**	0.765	0.802	0.816
5	25	TrAdaBoost	AUC	**0.650**	**0.664**	0.735	0.753	**0.778**	**0.810**	**0.832**

Continued on next page

Table 5 – continued from previous page

Target	Source	Classifier	Metric	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
5	50	KNeighborsClassifier	AUC	0.583	0.619	0.718	0.709	0.741	0.765	0.756
5	50	SVC	AUC	0.561	0.632	0.738	0.762	0.773	0.798	0.807
5	50	RandomForestClassifier	AUC	0.620	0.670	0.728	0.743	0.775	0.798	**0.813**
5	50	AdaBoostClassifier	AUC	0.594	0.643	0.723	0.734	0.756	0.785	0.787
5	50	GaussianNB	AUC	0.619	**0.682**	**0.798**	**0.807**	**0.781**	**0.827**	0.809
5	50	TrAdaBoost	AUC	**0.647**	0.673	0.743	0.766	0.780	0.792	0.808
5	100	KNeighborsClassifier	AUC	0.605	0.641	0.716	0.734	0.734	0.767	0.775
5	100	SVC	AUC	0.565	0.615	0.745	0.760	0.766	0.790	0.791
5	100	RandomForestClassifier	AUC	0.614	0.659	0.726	0.750	0.773	0.794	0.809
5	100	AdaBoostClassifier	AUC	0.622	0.660	0.686	0.704	0.731	0.758	0.795
5	100	GaussianNB	AUC	0.615	**0.683**	**0.786**	**0.798**	**0.815**	**0.843**	**0.869**
5	100	TrAdaBoost	AUC	**0.673**	0.664	0.776	0.766	0.782	0.826	0.842
10	25	KNeighborsClassifier	AUC	0.589	0.619	0.696	0.720	0.743	0.762	0.788
10	25	SVC	AUC	0.548	0.607	0.725	0.746	0.750	0.781	0.796
10	25	RandomForestClassifier	AUC	0.620	0.664	0.713	0.743	0.772	0.794	0.805
10	25	AdaBoostClassifier	AUC	0.612	0.654	0.720	0.705	0.748	0.768	0.791
10	25	GaussianNB	AUC	0.608	0.651	**0.774**	0.740	0.776	0.791	0.811
10	25	TrAdaBoost	AUC	**0.640**	**0.665**	0.743	**0.749**	**0.785**	**0.813**	**0.835**
10	50	KNeighborsClassifier	AUC	0.587	0.620	0.722	0.703	0.741	0.763	0.748
10	50	SVC	AUC	0.562	0.626	0.736	0.762	0.771	0.802	0.807
10	50	RandomForestClassifier	AUC	0.611	0.665	0.728	0.749	**0.784**	**0.804**	**0.825**
10	50	AdaBoostClassifier	AUC	0.595	0.666	0.721	0.735	0.756	0.791	0.777
10	50	GaussianNB	AUC	0.619	**0.685**	**0.794**	**0.800**	0.781	0.793	0.814
10	50	TrAdaBoost	AUC	**0.646**	0.672	0.752	0.766	0.779	0.792	0.815
10	100	KNeighborsClassifier	AUC	0.610	0.641	0.711	0.734	0.733	0.768	0.778
10	100	SVC	AUC	0.562	0.616	0.741	0.758	0.763	0.794	0.796
10	100	RandomForestClassifier	AUC	0.607	0.656	0.712	0.744	0.772	0.785	0.825
10	100	AdaBoostClassifier	AUC	0.612	0.652	0.691	0.710	0.726	0.762	0.796
10	100	GaussianNB	AUC	0.616	**0.684**	**0.787**	**0.795**	**0.810**	**0.841**	**0.868**
10	100	TrAdaBoost	AUC	**0.668**	0.669	0.781	0.767	0.783	0.831	0.849

Continued on next page

Table 5 – continued from previous page

Target	Source	Classifier	Metric	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
25	25	KNeighborsClassifier	AUC	0.624	0.648	0.740	0.743	0.764	0.801	0.809
25	25	SVC	AUC	0.556	0.629	0.744	0.761	0.775	0.815	0.830
25	25	RandomForestClassifier	AUC	0.611	0.672	0.736	0.773	**0.794**	0.817	0.845
25	25	AdaBoostClassifier	AUC	0.604	0.663	0.720	0.725	0.758	0.797	0.800
25	25	GaussianNB	AUC	0.634	**0.681**	0.772	0.760	0.781	0.816	0.824
25	25	TrAdaBoost	AUC	**0.639**	0.675	**0.788**	**0.788**	**0.794**	**0.840**	**0.849**
25	50	KNeighborsClassifier	AUC	0.600	0.630	0.742	0.730	0.764	0.783	0.779
25	50	SVC	AUC	0.574	0.633	0.750	0.769	0.781	0.808	0.832
25	50	RandomForestClassifier	AUC	0.617	0.672	0.736	0.765	0.780	0.814	0.838
25	50	AdaBoostClassifier	AUC	0.598	0.660	0.738	0.736	0.760	0.795	0.800
25	50	GaussianNB	AUC	0.624	**0.694**	**0.789**	**0.800**	**0.790**	**0.820**	0.828
25	50	TrAdaBoost	AUC	**0.646**	0.676	0.776	0.775	0.782	0.813	**0.844**
25	100	KNeighborsClassifier	AUC	0.599	0.645	0.729	0.726	0.742	0.777	0.807
25	100	SVC	AUC	0.563	0.615	0.750	0.763	0.765	0.797	0.810
25	100	RandomForestClassifier	AUC	0.612	0.671	0.737	0.758	0.786	0.808	0.832
25	100	AdaBoostClassifier	AUC	0.609	0.653	0.720	0.719	0.754	0.781	0.808
25	100	GaussianNB	AUC	0.621	**0.688**	0.791	**0.799**	**0.814**	**0.844**	**0.872**
25	100	TrAdaBoost	AUC	**0.676**	0.660	**0.805**	0.777	0.788	0.843	0.868