KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Predicting Student Performance in Programming Courses Using Test Unit Snapshot Data

SANHERIB ELIA

Stockholm, Sweden, 2024

Predicting Student Performance in Programming Courses Using Test Unit Snapshot Data

SANHERIB ELIA

Degree Programme in Computer Science and Engineering
Date: February 4, 2024

Supervisor: Olga Viberg
Examiner: Cristian Bogdan
School of Electrical Engineering and Computer Science
Swedish title: Förutsägelse av Studentprestationer i Programmeringskurser med hjälp av Snapshot-data för Testenheter

Abstract

Predicting student performance is an important topic in academia, especially so in programming context, where identification of struggling students allows teachers to offer early and continuous assistance to help them improve their performance. It is thus essential to analyze student programming behavior to detect those at-risk students. This thesis uses data generated from 220 students in a master's level programming course at a large European university. The students run unit tests in order to test their code when solving assignments, with a snapshot being taken of each test as it is executed. Unit testing is a method of testing software where individual units of source code are tested for correctness. A data set with simple features is derived from a database of snapshots and labeled with students' grades. Then, the machine learning models support vector machine (SVM), naive Bayes (NB), random forest, and neural networks with one, two and three hidden layers each are trained, evaluated and performance is compared. The results show that SVM and neural networks models are likely the best performing all-rounders, with a possible naive Bayes selection depending on what goal one has. The thesis contributes by training machine learning models on students' programming behavior. By arming teacher with models such as these, more students that need assistance can get in-time support and thus improve their performance. Future work can improve the models by using or combining other types of student data as features or use a larger data set.

Keywords

Student performance prediction, Programming education, Unit test snapshot, Machine learning, Educational data mining

Sammanfattning

Att förutsäga studenters prestationer är ett viktigt ämne inom akademin, särskilt i programmeringssammanhang, där identifiering av studenter som kämpar med sina studier gör det möjligt för lärare att erbjuda tidig och kontinuerlig hjälp för att hjälpa dem att förbättra sina prestationer. Det är därför viktigt att analysera studenternas programmeringsbeteende för att upptäcka dessa studenter som är vid risk. Denna uppsats använder data från 220 studenter i en programmeringskurs på masternivå vid ett stort europeiskt universitet. Studenterna kör enhetstester för att testa sin kod när de löser uppgifter, och en snapshot tas av varje test när det körs. Enhetstestning är en metod för att testa programvara där enskilda enheter av källkoden testas för korrekthet. En datamängd med enkla features härleds från en databas med snapshots och märks med studenternas betyg. Därefter tränas och utvärderas maskininlärningsmodellerna support vector machine (SVM), naive Bayes (NB), random forest och neurala nätverk med ett, två och tre dolda lager vardera och deras prestanda jämförs. Resultaten visar att SVM och neurala nätverk sannolikt är de bäst presterande allroundmodellerna, med ett möjligt naivt Bayes-val beroende på vilket mål man har. Uppsatsen bidrar genom att träna maskininlärningsmodeller på studenters programmeringsbeteende. Genom att utrusta lärare med modeller som dessa kan fler studenter som behöver hjälp få stöd i tid och därmed förbättra sina prestationer. Framtida arbete kan förbättra modellerna genom att använda eller kombinera andra typer av studentdata som features eller använda en större datamängd.

Nyckelord

Förutsägelse av studentprestationer, programmeringsutbildning, snapshot av enhetstest, maskininlärning, Educational data mining

Acknowledgments

I would first and foremost like to thank my supervisor Olga Viberg for providing valuable assistance and input during the whole thesis. I would also like to thank my examiner Cristian Bogdan for providing feedback and challenging me to improve my work. Lastly I would like to thank my dear family and friends for supporting me through all my years at KTH.

Stockholm, February 2024
Sanherib Elia

Contents

Chapter 1

Introduction

Predicting the performance of students in university courses is of great interest to teachers because it can help identify struggling students and adapt their teaching methodology accordingly [1]. By identifying at-risk students early, student performance prediction (SPP) can help prevent academic failure, reduce drop out rates and improve students overall performance [2]. For instructors, SPP helps to adjust learning materials and course design based on students' abilities, such as identifying parts where students commonly struggle. In the context of programming education, offering continuous assistance to struggling students is key to helping them improve their academic performance. For example, Watson and Li [3] found an almost identical mean worldwide pass rate of 67.7% for introductory programming courses. While not alarmingly low, improvements could be made to increase pass rates. Exploring the design and the use of warning systems to identify at-risk students is an ongoing research field [4]. Most of these works save data from e-learning systems and use machine learning for making predictions (e.g. [2, 5]).

Many of these works focus on predicting students' performance and dropouts. Commonly used data types include socio-demographic characteristics, students' activity logs and platform interaction, grades on previous courses, attendance and retention rate. In the context of programming courses, several techniques have been proposed to predict the performance of students [6], for example analyzing typing patterns [7]. These studies in general analyze student behaviors, extract information and apply them as features in training machine learning models with the goal of student performance prediction (SPP).

Features that have been used in programming SPP vary greatly. Longi et al. [8] analyzed students' latency between keystroke in order to distinguish

beginner programmers from more skilled ones. Jadud [9] explored student compilation behavior, the programming behavior that occurs when a student edits and recompiles their program repeatedly. Carter et al. [10] used several different features that are collected from students' integrated programming environments (IDEs), including time between the first edit and assignment deadline, the difference in number of lines between two consecutive attempts and the number of variable in the source code. Other works include total number of attempts [11] and the total time spent by students solving problems [12].

This thesis uses data generated from a programming course offered to master students. The data is collected from snapshots from unit tests that students test their code on. Every time the tests are run by a student, a snapshot is taken and various forms of data are saved. Details are described in section 3. By using this type of data generated by students, this thesis aims to train and explore machine learning models for predicting student performance in the setting of programming higher education. The performance of the models is then to be evaluated and compared to find the best predictor. This work stands out by using data analogous to previous research, e.g. total number of attempts but slightly different and in higher education. The unit tests for students are not meant as a means of turning in assignments but rather testing if they are correct, making their behavior different from number of attempts of turning in work. The assignments are then presented orally to teacher assistants.

1.1 Problem

In higher education, it is important to be able to identify at-risk students to be able to provide timely interventions to help them succeed in their studies [1]. It is of interest of educators to be able to detect students which exhibit behavioral patterns that may lead them to fail a course. In a programming education context it is especially important to provide assistance and there are many different student behavioral aspects that may be analyzed for performance prediction [6]. This thesis aims to contribute to solving this problem by analyzing data from students in programming and predict their performance. With the problem as described, the research question is stated in section 1.2.

1.2 Research question

The main research question that this thesis aims to answer is:

- To what extent can machine learning models, trained on data from students in a higher education programming course, accurately identify struggling students by predicting their grades in the course?

1.3 Objective

The objective of this thesis is to train and evaluate machine learning models on data from students programming behavior when solving programming assignments in higher education. Specifically, this thesis contributes by proposal of features derived from unit test snapshot data and evaluation and comparison of machine learning models trained on these features. Hopefully, this allows teachers to understand what their students struggle on when performing programming tasks. It also allows researchers to build upon the models by introducing and combining more features with snapshot data.

1.4 Delimitations

This thesis aims to improve upon the research of SPP in higher education programming courses. The data collected and used is from a programming course held for master's students and expects general programming knowledge beforehand. The data is only from a single course offering with data from 220 students. The course is not specifically an online course, but lectures and help sessions are held online.

Student features such as past academic records and demographics are not used for model training and prediction. Features that are used are only generated from testing of lab exercises and final grades of students', detailed in section 3. The performance prediction is predicting the final grade of a student, given the features of their lab exercise testing.

This thesis performs student performance prediction by way of binary classification. This means that even though the course grades are multiclass, classification is done by predicting if a student passes or not. This is similar to student drop out prediction but with some differences. A student who stays a whole course but does not pass would not be considered a drop out, but in this thesis it would be predicted as a fail. The reasoning for binary classification instead of multi class is multifaceted. First, the goal is to ultimately help struggling students and improve learning outcomes. By performing binary classification, focus can be put on those that need the most help, i.e. fail their schoolwork, and completion rates of courses can be improved. Second, this

work is the first that is done on the course data and is using a type of data (unit test results) that is not currently commonly researched. With that in mind, it allows for simple beginnings where future work can build upon the work here such as expanding to multiclass classification if needed. Furthermore, the work can be built upon to function as an early warning system, where the goal for teachers is to detect students at-risk of dropping out mid-course. In that case, most of the work required would be to train the models on limited and early data, instead of all the data. Lastly, author experience steers the thesis towards binary classification as multiclass classification may use different models, parameters and metrics that are not suitable or applicable for binary classification.

Chapter 2

Background

This chapter introduces machine learning as a concept, types of machine learning classifiers, aspects of machine learning such as features and evaluation metrics and a section on educational data mining and related works.

2.1 Machine learning

Machine Learning is the technique of building learning algorithms that create models from data [13]. By providing input to the learning algorithm in the form of data, we create models that can make predictions on new and unseen data. Data, more accurately *data sets*, are collections of *records* or *instances*. A record or instance describes the properties of an object. These properties are usually referred to as *features*. For example, if a model was trained to predict if a fruit was ripe or not, the features could consist of weight, volume and color describing each fruit in the data set. Using algorithms to build models is called *training*. For a effective model to be trained, it must know outcome information [13]. In the fruit example, the training data set would include if the fruit is ripe or not. The outcome of a record is called a *label*. If the outcome is discrete, it is a *classification* problem. If there is only two outcomes, i.e. ripe or unripe, it is a *binary* classification problem. If the outcome is continuous, it is a *regression* problem. A data set can be divided into two subsets; a training set and *testing* set. As the training set is used for training the model, the testing set is used for making predictions and comparing to the value to measure performance. Depending on the training data being labeled or not, learning can be divided into *supervised learning* or *unsupervised learning*.

There exists a variety of applied ML models for student performance prediction. Sections 2.1.1 and 2.1.2 introduces and explains shortly how they

function.

2.1.1 Supervised Machine Learning Classifiers

Support vector machines (SVM) are a set of supervised machine learning models used for classification and regression. The goal of the SVM is to find a hyperplane* that best separates different classes in a feature space. The hyperplane is chosen such as to maximize the margin between the the different classes. A hyperplane can be expressed by the following function:

$$\mathbf{w}^\top \mathbf{x} + b = 0$$

where $\mathbf{w}$ is the normal vector controlling the direction of the hyperplane and b controls the distance between the hyperplane and the origin. The margin is defined as the distance between the hyperplane and the closest data points from each class. In SVM models, data points are represented as vectors in the feature space, and the two closest points are the ”support vectors”. Once the model has been trained and a hyperplane found, new data points can be classified based on which side of the hyperplane they are found [14]. SVM's can be of a linear or non-linear type. A linear SVM is used for linearly separable data points, e.g. a line in $\mathbb{R}^2$. Non-linear SVM's are thus used when data points are only separable by non-linear hyperplanes, e.g. a curve in $\mathbb{R}^2$ [15]. Non-linear SVM's work by using a kernel function to transform the data to a higher dimensional feature space where the data points are more likely to be linearly separable [16]. Common kernels are the linear kernel, given by

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top x_j$$

polynomial kernel given by

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top x_j)^d$$

and Gaussian kernel, also known as Radial basis function kernel:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2})$$

Figures 2.1, 2.2 and 2.3 show SVM's trained with different kernels.

Naive Bayes (NB) classifiers are simple supervised machine learning algorithms used for classifying data points. Naive Bayes is based on Bayes'

*A hyperplane is an N-1 subspace in an N-dimensional space.

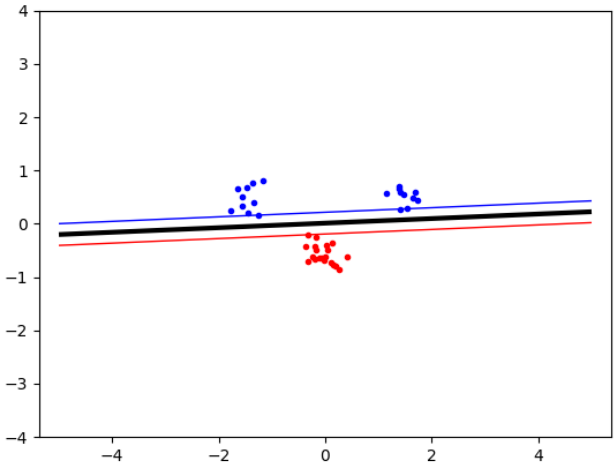


Figure 2.1: A linear SVM kernel.

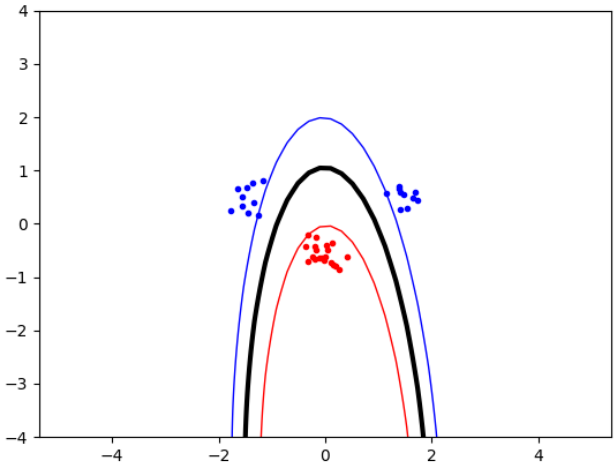


Figure 2.2: A polynomial SVM kernel.

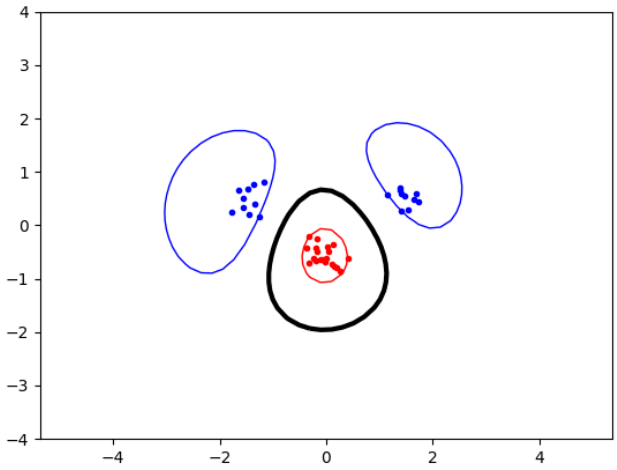


Figure 2.3: A radial SVM kernel.

theorem with the assumption that each pair of features of the data points are conditionally independent of each other given the class label. Bayes' theorem states the following probability given a class variable y and dependant features $x_1 \dots x_n$:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n \mid y)}{P(x_1, \dots, x_n)}$$

Then, with the assumption of conditional independence:

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y),$$

which can be simplified for every i to

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

With $P(x_1, \dots, x_n)$ being constant for a given input, the expression can be simplified

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

and estimate $P(y)$ and $P(x_i \mid y)$. Despite the noticeable simplification by naive assumption, Naive Bayes classifiers have proven to be effective in dealing with with real world classification problems, such as spam filtering and document classification. It typically require a smaller amount of training data to estimate the parameters needed. Naive Bayes models can be considerably faster than other more complicated classifiers. Types of Naive Bayes models are Gaussian, Multinomial and Bernoulli. The assumptions in estimating $P(x_i \mid y)$ is the main difference between the different models. [17]

Decision trees are supervised machine learning approaches that make decisions based of rules learned from training data. The decision structure is built as a tree with nodes consisting of features, branches indicating decision outcome and end nodes called *leaves*. The leaves consist of the class labels and are the final prediction of a record. A prediction is made by starting at the root node, then traversing the tree and following decisions made at each node until a leaf is reached. The tree structures are constructed by determining the best features for splitting the data at each node. For classification, criteria such as *entropy* is used for determining where to split. Decisions are made so as to reduce the entropy at each step. *Random Forest* is an *ensemble* * machine learning which combines multiple decision trees in order to make more accurate predictions. Random forest creates multiple decision trees during training then aggregates predictions to reduce overfitting and increase accuracy

2.1.2 Neural Networks

Neural networks, also referred to as artificial neural networks, are machine learning models that are inspired by the neural architecture of animal brains. They originated in the 1940's and have become highly used tools today [13]. The most simple element is the neuron. Biological neurons send out signals to connected neurons when they become "excited". If the electric potential of the signal exceeds a *threshold*, the neuron is activated, i.e. excited. The first neural model, the McCulloch-Pitts model (M-P model), modeled this behavior where each neuron receives input signals from *n* neurons through weighted connections. The weighted sum of the signals is compared to the threshold and the output is produced by a *activation function*. An activation function

*Ensemble methods is a set of techniques that combine predictions of different classifiers in order to improve the overall performance. The idea is that combining predictions results in better performance than an individual model.

can be a step function:

$$\text{sgn}(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0th \end{cases}$$

or a continous *squashing function*:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x=}}$$

A squashing function squashes the input values of large intervals into the interval (0,1). It is often used because the step function has unwanted properties such as being discontinuous and non-smooth [13]. The network is thus built by connecting the neurons into multiple layers, consisting of *input layers*, a *hidden layer* and *output layers*. Figure 2.4 shows a simple connected neural network.

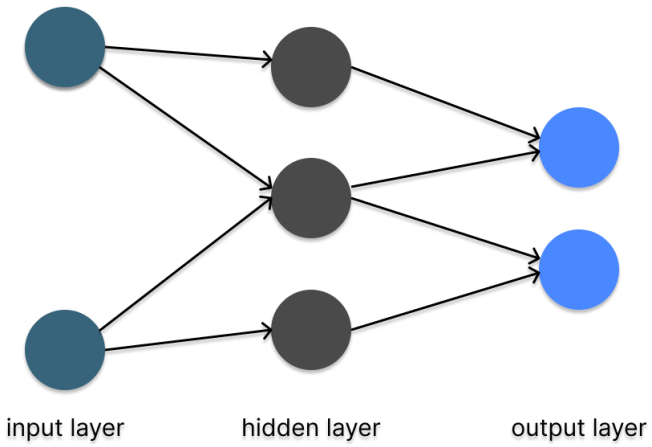


Figure 2.4: A simple neural network.

2.1.3 Deep Learning

Deep learning is a subset of machine learning that involves using neural networks with multiple hidden layers. Deep learning models are designed to automatically learn features and patterns from raw data in order to exclude the need for manual feature engineering. Stacking multiple layers allows a network to learn more complex and abstract patterns in the input data [18]. Common deep learning methods include convolutional neural networks (CNN) and recurrent neural networks (RNN). Figure 2.5 shows a connected

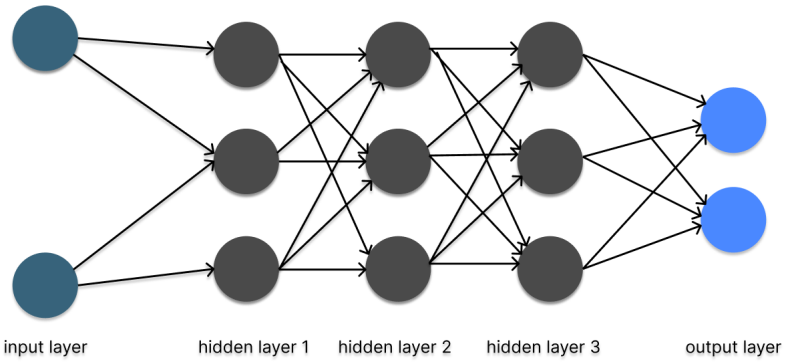


Figure 2.5: A deep neural network with three hidden layers.

deep neural network.

2.2 Feature selection

For training ML models, features from the data are analyzed. For some problems, not all features are equally useful. Features that are useful for learning the problem are *relevant features* and useless ones are *irrelevant features*. The process of selecting the most relevant features from a feature set is called *feature selection*, a form of *dimensionality reduction*. In addition, when ML algorithms are applied on high-dimensional data, the issue known as *curse of dimensionality* arises. This issue refers to the phenomenon that data becomes sparser in a higher dimensional space, affecting the performance of algorithms designed for lower dimensional space [19]. Furthermore, learning models tend to overfit when trained on data with a large amount of features, leading to decreased performance when predicting on new data [20]. Memory storage requirements and computational costs can also substantially increase with high dimensional data. Dimensionality reduction is one of the most useful tools to reduce the impact of high-dimensional data issues. Feature selection is an important step of data preprocessing in order to improve model performance [19]. There are different types of feature selection methods [21]:

- *Filter methods.* Filter methods are feature selection methods that are applied before learning and are model independent, meaning it they can be applied for any ML model. Filter methods work by ranking feature importance using relevance statistics and selecting a subset of the highest ranked features. Common statistical relevancy measures include Pearson correlation coefficient and Chi-squared test.

- *Wrapper methods.* Wrapper methods work by selecting a subset of features based on the performance of the machine learning model. Wrapper methods hence use the performance of the model directly as a criterion for selecting a subset of features, unlike filter methods. Wrapper methods train models several times and evaluates each one, making them more computationally costly than filter methods. However, this generally results in better feature selection as the feature subset is tailored for the learner model and problem domain [13]. Two common wrapper methods are Forward selection and Backward elimination.

- *Embedded methods.* Embedded methods combine the feature selection process and model training process into one process, meaning that the features are automatically selected during training. Usually, a penalty term is added during model training, to encourage selecting only the most useful features. The term can be a *regularization* term such as L_1 or L_2 regularization which constrain the weights of the features and help with overfitting [13]. Features with higher weights are considered more important.

In addition to using algorithms for feature selection, it may also be done manually by experts [22].

2.2.1 Feature extraction

Feature extraction is a dimensionality reduction technique based on deriving new information from existing features and map it to a new lower dimensional feature subspace. The main goal is data compression while trying to maintain as much of the relevant information as possible in order to reduce model complexity, reduce overfitting and improve computational costs [21]. Drawbacks are that newly generated features may still lose information about the original feature space [23]. Common feature extraction methods are Principal component analysis (PCA) and Linear discriminant analysis (LDA).

2.3 Evaluation metrics

The evaluation of different models is a crucial aspect of determining the most effective prediction methods. Consequently, selecting appropriate metrics for assessment is essential. There exists several well-known evaluation metrics for research in statistics, machine learning and deep learning. Prenkaj et al.

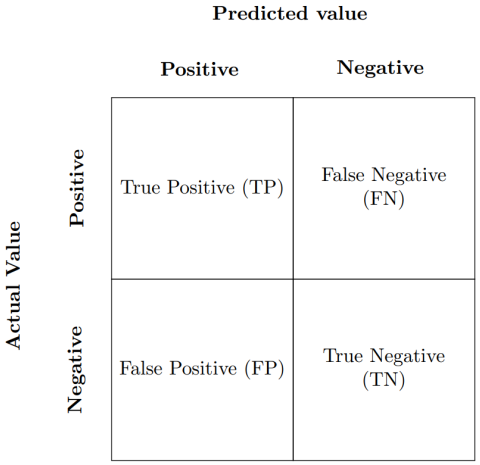
[24] identified and reported measurements with a survey conducted on ML approaches for student dropout prediction (SDP) in online courses:

1. *Accuracy.* The earliest works use only accuracy as a measurement for assessing model performance. It is calculated by the number of correct predictions divided by total predictions. Accuracy by itself can be a misleading metric. Particularly for online courses, classes often have unbalanced data, meaning that the amount of students completing a course differs from the amount of drop outs. Considering this imbalance, a model used for SDP can predict the value of the majority for each prediction and thus receive a high accuracy. Therefore, it is required to apply additional metrics to properly evaluate the classifier.

2. *Confusion Matrix and F-measure.* Addressing the shortcomings of accuracy by itself in identifying strong classifiers, some papers [25, 26] utilize rates calculated from the generated confusion matrix. For example, Precision, Recall and their derivatives, e.g. F-scores/F-measures. Figure 2.6 shows a confusion matrix and how their derivatives are calculated. A significant amount of the studies that rely on precision and rate-based metrics use the F-measure. However, the F-score lacks an inherent ability to provide intuition of the performance. Combining precision, recall and F-score give a better intuition of the performance of a classifier.

3. *AUCROC and AUCPR.* AUCROC is ”area under curve” (AUC) of the receiver operating characteristic (ROC) curve which is built upon the recall measurement and false positive rate. The ROC curve plots the true positive rate against the false positive rate at different thresholds. AUCROC provides a measure of a models ability to differentiate between classes. The higher the AUCROC, the better the model is at predicting. AUCROC thus uses the true positive and false negative rates to give a more accurate view of a model's performance. However, AUC Precision-Recall is more suitable for the problem domain since it is less susceptible to the effects of imbalanced class data [27].

4. *Mean Squared Error (MSE).* MSE measures the average of the squares of the difference between the predicted and actual value. It is defined as such by $\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - y_i)^2$ where $\tilde{y}_i$ is the predicted value and y_i is the actual value. An MSE score is always positive and a score closer to zero is considered better, as the prediction difference is smaller. MSE is mainly used for regression tasks.



(a) Confusion matrix.

$$\text{fp rate} = \frac{FP}{N} \qquad \text{tp rate} = \frac{TP}{P}$$

$$\text{precision} = \frac{TP}{TP+FP} \qquad \text{recall} = \frac{TP}{P}$$

$$\text{accuracy} = \frac{TP+TN}{P+N}$$

$$\text{F-measure} = \frac{2}{1/\text{precision}+1/\text{recall}}$$

(b) Confusion matrix derivatives.

Figure 2.6: A confusion matrix and how its derivatives are calculated.

5. *Mean Absolute Error (MAE).* Similar to MSE, but averages the absolute error between predictions and true values: $\frac{1}{n}\sum_{i=1}^{n}|\tilde{y}_i - y_i|$, with $\tilde{y}_i$ and y_i having the same value. Overestimates and underestimates are treated identically. As with MSE, a lower score is better and is used for regression.

2.4 Related works

Educational data mining (EDM) is a research area that deals with the application of machine learning, data mining and statistics to data sets generated from educational settings. The goal of is to find and develop methods to find patterns and analyze such data in order to understand student learning processes and in order to improve learning outcomes. The field is similar to learning analytics which shares certain aspects.

Predictive analytics deals with the methods of analyzing data in order to make inference about future events. In educational context, it may be that one wants to make predictions about student dropout or success, impact of teaching styles or other academic factors. Predictive analytics is a well-established area of research [28]. It is important to be aware of the difference between predictive modelling and explanatory modelling in education. Explanatory modelling seeks to use all available information in order to explain how each variable (e.g. age, gender) contributes to a given outcome (e.g. student grade,

drop out). Predictive modelling is about building models on a set of known data in order to predict values on new data based on the features.

Predictive analytics has been used in higher education to predict student performance. For example, early warning systems have been developed to predict online students' learning performance by analyzing learning activities [29, 30]. These analytics have been extended to include programming courses ([1]). In practice, predictive systems have been used by teachers successfully. By giving teachers access to these systems, Herodotou and colleagues [31] helped teachers that used these systems to flag students at-risk of failing weekly assignments. Students flagged and intervened were found to perform significantly better than previous years students.

This rest of this section provides an overview of related works in the different aspects of predictive modelling.

2.4.1 Data collection

Data collection is an essential first step in data mining. In EDM, there are several ways that data is collected for analysis. Learning platforms can capture various forms of data, such as trace logs, session activity, navigational patterns, discussion forum use and exercise submissions, which can be analyzed by researchers for insights [32, 33, 34]. This captured behavior is referred to as learning behavior, engagement data or log data. This pertains to the behavior exhibited by students during their engagement with an educational platform when performing different learning activities. The granularity of log data can vary, with very fine-grained data being extracted such as video or multimedia engagement events. For example, these events may consist of pausing, stopping and replaying videos. Learning behavior features are the most frequently utilized [23].

Some data sets used do not contain any platform telemetry but use teachers' observation reports for performance prediction [35]. Research might also use pre-course data, such as students' past academic performance as an information source [36] but also demographic background. Demographic information of learners may include characteristics such as name, age, gender, mother language, origin, occupation, socioeconomic status and hobbies.

The results of a survey by Xiao et al. [22] show that 78% of surveyed studies use data collected from educational institutions using digital systems such as student information systems (SIS) or learning management systems. 12% use public data sets and 10% use questionnare or survey type data. Although a large amount of data sources are SIS or LMS, studies may use

vastly different types of features which makes models dependant on the type of data and complex to compare.

2.4.2 Data size

The size of data used for studies varies greatly between works. For instance, Mengash et al.[37] gathered 2039 computer science students' record from a Saudi university while Yousafzai et al. [38] collected 80,000 students' data from the school board in Islamabad. Xiao et al. [22] show that 63.5% of selected studies use data smaller than 1000 records. Training on small datasets makes models more prone to overfitting and susceptible to noise in data. Xiao et al. claim this is a "common and significant problem in previous researches [sic]". 16.5% of selected studies use data consisting of 5000 records or more. A larger amount of training data can improve generalization ability of prediction models.

2.4.3 Prediction models

Surveys conducted on the area of SPP show that almost all studies use supervised machine learning algorithms to train prediction models [22]. Researchers also often use more than one classifier in order to study the performance of different classifiers. Amrieh et al. [39] use a set of classifiers consisting of artifical neural networks, Naive Bayes and Decision Trees. In addition, they use ensemble methods to improve the performance of the classifiers. The ensemble methods they use are Bagging, Boosting and Random Forest. The results show a clear relationship between student's behaviors and academic performance. The model they propose that includes behavioral features shows 22.1% improvement compared to without, and achieves more than 80% accuracy on testing new students.

Castro-Wunch et al. [40] focus their work on using neural networks in order to detect students in need of assistance as early in a programming course as possible. The study builds upon earlier work by Ahadi et al.[41] that shows that ML is effective at predicting student performance with accuracy in the range of 80-90%. They extend the study by Ahadi et al. through reproducing and introducing a different context and extend the work by using a neural network. The results show that shallow neural networks (one layered) performed as well as more complex networks and that neural networks performed as well as the best Bayesian classifiers.

Kloft et al. [42] train an SVM classifier on weekly data on a MOOC.

They state that ”historical” data that come from previous weeks are useful until 12 weeks in. The work focuses on using clickstream data including features such as number of active days, number of page views, number of start-stop during video plays and number of wiki views that are gathered from a course on Coursera. They conclude that prediction accuracy is the highest at the end of the course and could be improved in the future by including forum data.

Wang et al. [43] propose a new model for dropout prediction in MOOCs. The model is a deep neural network that is a combination of CNN and RNN. It is motivated by the fact that at the time of writing, current methods of SDP rely on features extracted by manual feature engineering which is costly, time consuming and sensitive to changing contexts such as different platforms or courses. They produce a new model that can automatically extract features from raw course data. Testing on public datasets show comparable results to feature engineering based methods.

2.4.4 Evaluation metrics

Certain studies [44, 45, 46] employ supplementary metrics in addition to accuracy in order to gain a more realistic view of the model’s performance. Wolff and colleagues [47] use precision, recall and F-measure to evaluate their ML model that analyze clicking behaviour in a virtual learning environment. Robinson et al. [48] use machine learning models for natural language processing in order to predict student achievement. They evaluate text answers of open response questions to see if it results in better prediction that data retrieved from fixed-response items. They state that their model can learn from text to predict whether a student will complete an online course. Because of the imbalanced class data of online courses, they use area-under-curve metric of the ROC curve for evaluation.

Chapter 3

Method

This chapter provides an overview of the method used in this thesis. Section 3.1 describes the data source, the course which the data is from and its syllabus. Section 3.2 details the data preprocessing and feature engineering. Section 3.3 focuses on which classifier models were used for student performance prediction. Section 3.4 lists which evaluation metrics were used for evaluating and comparing machine learning models.

The problem domain was formulated as supervised binary classification where the class label to predict are whether the student received a passing grade or not.

3.1 Course and data collection

The course from which the data was gathered from is a master's level programming course given at a university. The course consists of three programming assignments and a group project. The first programming assignment is an individual assignment and the others in groups of two. The assignments are constructed as tutorials to introduce the students to ideas and concepts in developing interactive applications, such as Model-View-Controller. The group project is done in teams of four. The course grade is solely determined by the group project grade, although the assignments are required to pass the course.

While working on the assignments, students are given unit tests that they can test their code on. The unit tests consist in total of 45 test suites. The unit tests are connected to a backend which saves a snapshot each time the tests are run. Information such as timestamp, semester, test pass/failure, user, test suite name and error message are saved in the database. Students can opt to appear

as anonymous users or not save test information at all. All data was from one course offering in autumn 2022. Data from anonymous submissions was not collected since they were unified under one user ID.

The data that was collected for each student was the total number of runs for each test suite, with each suite run count as a feature. For group assignments and tests, the group run count was added to the individual students run counts since they were expected to work together. The records were extracted with 45 features each.

3.2 Data processing and labeling

Once the records were saved, cleaning and labeling of the data occurred. Records belonging to teachers and teacher assistants were removed. As some records features contained null values, meaning the student had never run that test suite, null values were set to 0. The grades that students achieved in the course were joined to the corresponding record to label the data and the data set was anonymized. As the course grade is a multi-grade one, passing grades were mapped to the same value, and failed or missing grades were mapped similarly, due to the binary classification modeling.

The data set is quite imbalanced, with 220 total records consisting of 183 students who passed the course and 37 who did not. The implications of this is discussed in 5.4.2 and efforts to mitigate the possible issues are detailed in 3.3.

3.3 ML models

The data set was used to train multiple predictive models: SVM, naive Bayes and random forest. In addition, multilayer perceptrons (MLP), a class of neural networks, were trained with one, two and three hidden layers each. All models were trained using the scikit-learn library for Python. The motivation for this choice of models stem mainly from them being common in student performance prediction for classification (e.g. [49, 23]) and previous experience.

The SVM, NB and random forest classifiers has similar procedures. Hyperparameters were tuned by using a grid search method and a 5-fold stratified cross validation for evaluating the best performing parameters. Since the data is imbalanced, the AUCPR metric was exploited for evaluating parameters. For SVM and NB, feature selection using a filter method was

performed. Random forest feature selection was done through hyperparameter tuning. Features were standardized before training the SVM and NB models using a scikit-learn scaler, which removes the mean and unit variance, to avoid bias towards features with large values and variance. Random forest is an ensemble method consisting multiple decision trees, which do not require scaling and no scaling was performed as such.

For the neural networks, a ”vanilla” neural network with one hidden layer was trained and two deep neural networks with two and three hidden layers, respectively. The procedure was similar to the classic ML classifiers with feature scaling, hyperparameter tuning for each of the three models and 5-fold cross validation. All models were evaluated by the mean cross-validated AUCPR score over the 5-folds.

3.4 Evaluation metrics

For evaluation and comparing the models, six metrics where chosen and calculated. AUCPR, AUCROC, precision, recall, F1 and accuracy. Since the data set is imbalanced, the choice of AUCPR is evident. The other metrics are the most commonly used in student prediction [22] and thus included.

Chapter 4

Results

This chapter presents the results gathered from the procedure outlined in chapter 3. Table 4.1 lists all the models trained and their performance metrics, with the best model in each metric bolded. The metrics are calculated as the mean over the cross validation folds and rounded to two decimals. For the neural networks, the amount of hidden layers is in parenthesis.

Considering the AUC metrics, the neural networks performed the best for both metrics with the single hidden layer having the best AUCPR and the NN with two hidden layers performing slightly better by 0.02 percentage points (rounded off). The SVM performed very similarly to the best neural networks, coming in slightly short of 0.1 percentage points in both AUC metrics, but had the highest precision, accuracy and F1 score making it the best all-around performer. Naive Bayes scored very highly on recall compared to the others, but was the weakest performer in precision, F1 and accuracy, and tied for AUCROC. Random forest did not perform the best in any metric whatsoever and the worst AUCPR. Worth noting is that the feature selection did not improve performance in any model.

Table 4.1: Classification performance. Top performers for each metric bolded.

Model	AUCPR	AUCROC	Precision	Recall	F1	Accuracy
SVM	0.71	0.89	**0.78**	0.56	**0.64**	**0.90**
NB	0.56	0.76	0.23	**0.95**	0.36	0.44
RF	0.54	0.76	0.70	0.30	0.41	0.87
NN (1)	**0.72**	0.90	0.64	0.57	0.59	0.87
NN (2)	0.71	**0.90**	0.69	0.52	0.57	0.88
NN (3)	0.68	0.88	0.63	0.54	0.58	0.86

Chapter 5

Discussion

This chapter discusses the findings of this thesis in the context of earlier research results. Limitations of the study are also discussed and potential improvements that future work can implement.

5.1 Results

The results show an interesting spread of best performers, depending on which evaluation metric is considered. However when looking at all the metrics for a given model, it becomes clear that the best model depends on what the goal is. If the goal is to find as many students as possible that are at risk of failing, one might want to choose naive Bayes as a predictor. Yet, this would include many false negatives since the precision is very low and might waste teachers' time as they would have to manually confirm if the student needs assistance. Since the goal is to identify as many failing students as possible, it might be an acceptable compromise. Similarly, Amrieh and colleagues [39] obtain results that show NB being a weaker classifier when compared to a neural network and decision tree. Although their NB model has a lower recall of 0.80 compared to this thesis' 0.95, their model's accuracy, precision and F-1 score are significantly higher at 0.80, 0.85 and 0.79 respectively. Their model also did not perform much worse than their neural network, in contrast to this thesis. AUCPR and AUCROC was not included by them as metrics.

SVM is the best performer in three of the metrics and second best in AUCPR. This makes it seem like the best all-around model in this context, which is shown especially in its AUCPR score. SVM would likely be an appropriate model for courses with many students where a balance between true positives and false negatives is needed. The neural network with one layer

performed the best considering AUCPR, also making it a likely candidate. Adding more hidden layers did not seem to make a definite improvement for the neural networks. Castro-Wunsch et al. [40] demonstrate similar results where a neural network with a single layer performed the best.

Random forest seems to be the worst model in the context of this study. Comparing its results to the other models, it would be hard to justify using RF as a predictor model for future students in this course. Interestingly, when predicting final grades of students of programming courses, Araya et al. [6] claim they obtain good score using RF models and is comparable to other works as they only use data that covers the first third of a course. The weak performance of RF in this study is likely due to the feature engineering or low amount of samples.

Sivasakthi and Padmanabhan [50] train a decision tree and naive Bayes model for students programming performance prediction. Using a data set of 490 records, the NB model achieves a recall of 0.90, precision of 92.21 and accuracy of 91.02, while the decision trees obtains 84.89 recall, 86.57 precision and 85.10 accuracy. Comparatively, their NB achieves vastly higher precision and accuracy, but slightly lower recall. However their features differs greatly, as they use data from 16 questions generated from a questionnaire that the students answered.

5.2 Application

A key application of educational data mining is student modelling [51]. Student modelling represent students' state or characteristics, allowing for systems such as predictive models to learn about how students work and learn. Modelling in programming EDM can be done in several ways, which are discussed in section 2.4. For the case at hand, this thesis experiments by modelling students' programming and testing characteristics in order for prediction models to predict their course grade outcome. For the teacher of the course, it proposes an idea on how to directly represent students' work that is relevant to the course, i.e. programming habits for a programming course.

For choosing a prediction model, ideally the model with best score would be chosen. In this thesis the main score was tied between SVM and neural network. Due to the slower training times and higher complexity of neural network, the SVM model is likely the best to use. SVM generally are simpler models than neural networks and easier to interpret for non-experts.

To make the results of this thesis useful for teachers, the model ideally should be put into real use, meaning that it should be tested for future students

in the course. In the course that the models were trained on, the unit tests are used for roughly half the course. That means that for this course, it could be used as a early warning system, although it was not designed as such. An even earlier warning system could be developed, by only using data from earlier assignments, considering there are 3 for the course. It would be interesting to see the performance on limited data.

For the model to be considered good, it should increase the amount of pass grades compared to earlier years. How much larger percentage increase of passing students justifies the increased workload would be up to the teacher. Herodotou et. al [52] developed a predictive student model and put it to use. They managed to increase students who pass the course by a statistically significant amount by contacting them, compared to a control group. This means that using a predictive model such as the ones trained in this thesis is capable of teacher support and increasing student learning outcomes. By quickly identifying struggling students, support for them is just an email or phone call away.

The models right now are separate from the course, the source code and data set are not attached to any system such as a learning platform. For it to be most useful, teacher would benefit to have a model be integrated into a learning management system or data base that the course is using. This would require quite additional work and would be a good way forward for future work.

5.3 Contribution

This work contributes to the area of student programming performance prediction in higher education. By transforming raw, real data generated from students in an advanced level programming course, to training prediction models, the results of this thesis can assist teachers in the future to improve students' learning performance in the setting of programming education. In regards to the research question, it can be determined that data from programming students in higher education can be analyzed to predict their course grades. Engaging teachers with these analytics can positively affects students performance by assisting in identifying when students struggle on assignments[53]. By hopefully arming the teachers with, either the results gathered or models trained, they can train their own machine learning models or use them directly when the course is active to be able to provide in-time support to students, ultimately leading to their improved acquisition of programming skills. For researchers, the contribution is the experimentation with training machine learning models on unit test snapshot data. This type

of data is of interest due to the insight that it gives on students working on programming tasks. Future work may build upon this, considering the improvements discussed in 5.5.

5.4 Limitations

5.4.1 Data set size

The data size that the models were trained on is relatively small, with 220 records. What constitutes a small data set depends on many factors such as the model, features and type of data being predicted. A suggestion for optimal feature size is $\sqrt{N}$ for N size data sets and highly correlated features [54]. Since the amount of features in this data set is 45 and are likely correlated, the optimal amount of records would be about two thousand. There are several obstacles that arise when training on a small data set. When trained on a small data set, a model may overfit and does very well on the training data but performs poorly on unseen data, such as predicting grades on new students. With few training samples, models adjust excessively to the training data and learns patterns that do not exist with new data. Techniques such as regularization and cross validation exist to mitigate overfitting on small data sets, which would be recommended to consider for future research. In addition, simple or pre-trained models can help. Cross validation was utilized in this case. Furthermore, simply gathering more samples would likely be the most effective way to overcome to this problem. The course which the data was from was only from one semester. Collecting data from multiple semesters would be a good way forward, given that the course does not change drastically from semester to semester.

5.4.2 Imbalanced data set

As section 3.2 mentions, the data set is imbalanced towards passing grades. While this is positive from a teacher perspective, it can cause significant issues when training classifiers on them. Models trained on imbalanced data sets may become biased towards the majority class. In this case, it would predict students as passing more frequently, causing students at-risk to be overlooked. In addition, it can lead to poor generalization and overfitting. Models might struggle to generalize on new, unseen data. There are several methods available to deal with data imbalance. For example, resampling techniques such as oversampling and undersampling. Castro-Wunsch et al. [40] use

resampling techniques to balance the training data so that the two categories have the same amount of samples. This thesis did not use any resampling due to the already limited data set. Instead, to use all available data, a stratified 5-fold cross validation method was utilized. The high accuracy combined with lower scores on the other metrics is an indication that the models were likely biased. The combination of relatively small and imbalanced data set complicated making well-trained and accurate models.

Depending on several factors such as course subject and being online or face-to-face, it is usually the fact that majority of students will pass a course. This makes data from education imbalanced due to its nature, meaning that some counter actions against data set imbalance such as collecting more samples will not naturally balance the data. However, this means that other methods such as over and undersampling will be more important to apply to models in cases such as this for them be most effective. Imbalanced data sets in education is a well known problem and has received much attention with papers suggesting methods such as resampling and cost-sensitive methods [55]. Instead of only predicting passing students, it may be of interest to predict all grades, i.e. A-F, which is likely to be more balanced, although with its own difficulties.

5.4.3 Feature engineering

The feature engineering process that was performed on the raw data was quite simple. By only counting total number of runs of each test suite for every student, a large amount of information was not included as features for training. Performing more complicated feature engineering was not in the goals for this thesis, but future researchers with more expertise could produce more intricate features. There are many different features that could and have been used for student performance prediction. For example, Dong et al.[33] used student trace logs to identify struggling moments during programming assignments. They develop algorithms to define struggling moments and determine struggling students with 77% accuracy. Features similar to this incorporating error logs and timestamps could be extracted from the data used in this thesis. More general features about students could be added to the samples quite easily. Khan et al.[56] use features such as gender, cumulative GPA, major, degree, year, attendance percentage and grades for student performance prediction.

Prenkaj et al. [24] contribute to the field by formally defining student modelling of the dropout problem. They also formally define the dropout

problem both in a time and non-time dependant manner. These model formulations could be used when building upon this thesis in the future

5.4.4 Binary classification

The problem was formulated as a binary classification, meaning that any grade above failure (A-E) was reduced into a single class. Due to the large amount of students passing the course, it means that finer grained classification could have been performed. It would be interesting to see the same data to be used for training multiclass classifiers. As the first thesis that used the data at hand, it seems reasonable to start simpler and smaller with binary classification. It is also easier to train for binary classification, in a general ML sense and multiclass classification might need researchers with more expertise. It would be simple for future work to train on the data as multiclass, although it complicate comparing to the models here. In that case however, it would be most interesting to try to predict every grade that students can get (A-F). Due to the high amount of passing students, it might be more useful for the teacher to predict low grades and help these students receive a higher grade. Very few courses will have 100% pass rate and it might be more sensible to try to raise the average grade that students achieve instead. Partial grade ranges could also be predicted such as A-C or D-E. Whatever classes that are chosen to be predicted, it would certainly increase difficulty in achieving high performance as more classes are introduced meaning higher risk of predicting incorrectly. Nonetheless, it would be up to the teachers to decide what is most useful and to proceed from there.

5.4.5 Feature analysis

A method that could have been applied is some sort of feature importance analysis. Currently, no such technique has been done which means it is not clear which features, i.e. unit tests, contribute most to a prediction. All features seem equally important which is unlikely to be the case. It may be the case that early unit tests are more important in determining whether a student keeps progressing through the course. Students with higher test counts likely have higher chance to pass the course, as it is a indicator of the student being motivated and working often with assignments, compared to someone that runs the tests a few times and gives up, or do not run any tests at all. This means that the features are likely positively correlated with grades. Nevertheless, a few records in the data sets show normal to high test counts but did not pass

the course, indicating other issues likely occurring later in the course.

5.5 Future work

The results of this thesis could be improved upon in the future. Most importantly would be to address the limitations as chapter 5.4 discusses. The main limiting factor of this study is arguably the small data set, which manifests into many different complications. Work that builds upon this thesis should specifically try to gather a larger data set in order to train more accurate models.

Exploring and experimenting with different sets of features is a relevant aspect that future work can focus on. Since this thesis employs simple feature engineering, new research should undertake different ways. Other features could be ones that section 5.4.3 discuss or examine trivial features that current research is lacking on.

Although a balanced data set would be ideal to train and test on, in the context of this course and education in general, it would be hard to achieve. A balanced data set in this case would mean a failure rate of 50% for the course which would mean a less students passing than before. Instead, a larger data set should be gathered and combined with the mitigation techniques that section 5.4.2 discusses.

5.6 Conclusion

This thesis experiments with different machine learning models in educational data mining for improved teacher support and students improved learning outcomes. Focusing on programming in higher education, real world data from a university course is used to generate a data set and train the models. The results and model might hopefully aid the teacher in predicting student performance to identify at-risk students. The results suggest that the best model depends on the goal of the teacher, but an SVM or neural network model is appropriate for an all-around model. A naive Bayes model might be suitable if the goal is to not miss any poor performers. The next step would be to arm teachers with this tool or to build upon the thesis by addressing the limitations.

References

[1] D. Moonsamy, N. Naicker, T. T. Adeliyi, and R. E. Ogunsakin, "A meta-analysis of educational data mining for predicting students performance in programming," *International journal of advanced computer science & applications*, vol. 12, no. 2, pp. 97–104, 2021. [Pages 1, 2, and 15.]

[2] B. Albreiki, N. Zaki, and H. Alashwal, "A systematic literature review of student' performance prediction using machine learning techniques," *Education Sciences*, vol. 11, no. 9, 2021. doi: 10.3390/educsci11090552. [Online]. Available: https://www.mdpi.com/2227-7102/11/9/552 [Page 1.]

[3] C. Watson and F. W. Li, "Failure rates in introductory programming revisited," in *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ser. ITiCSE '14. New York, NY, USA: Association for Computing Machinery, 2014. doi: 10.1145/2591708.2591749. ISBN 9781450328333 p. 39–44. [Online]. Available: https://doi-org.focus.lib.kth.se/10.1145/2591708.2591749 [Page 1.]

[4] J. L. Rastrollo-Guerrero, J. A. Gómez-Pulido, and A. Durán-Domínguez, "Analyzing and predicting students' performance by means of machine learning: A review," *Applied Sciences*, vol. 10, no. 3, 2020. doi: 10.3390/app10031042. [Online]. Available: https://www.mdpi.com/2076-3417/10/3/1042 [Page 1.]

[5] D. Buenaño-Fernández, D. Gil, and S. Luján-Mora, "Application of machine learning in predicting performance for computer engineering students: A case study," *Sustainability*, vol. 11, no. 10, 2019. doi: 10.3390/su11102833. [Online]. Available: https://www.mdpi.com/2071-1050/11/10/2833 [Page 1.]

[6] I. Araya, V. Beas, K. Stamulis, and H. Allende-Cid, "Predicting student performance in computing courses based on programming behavior," *Computer Applications in Engineering Education*, vol. 30, no. 4, pp. 1264–1276, Jul. 2022. doi: 10.1002/cae.22519 Publisher Copyright: © 2022 Wiley Periodicals LLC. [Pages 1, 2, and 26.]

[7] J. Leinonen, K. Longi, A. Klami, and A. Vihavainen, "Automatic inference of programming performance and experience from typing patterns," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, ser. SIGCSE '16. New York, NY, USA: Association for Computing Machinery, 2016. doi: 10.1145/2839509.2844612. ISBN 9781450336857 p. 132–137. [Online]. Available: https://doi.org/10.1145/2839509.2844612 [Page 1.]

[8] K. Longi, J. Leinonen, H. Nygren, J. Salmi, A. Klami, and A. Vihavainen, "Identification of programmers from typing patterns," in *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, ser. Koli Calling '15. New York, NY, USA: Association for Computing Machinery, 2015. doi: 10.1145/2828959.2828960. ISBN 9781450340205 p. 60–67. [Online]. Available: https://doi.org/10.1145/2828959.2828960 [Page 1.]

[9] M. C. Jadud, "Methods and tools for exploring novice compilation behaviour," in *Proceedings of the Second International Workshop on Computing Education Research*, ser. ICER '06. New York, NY, USA: Association for Computing Machinery, 2006. doi: 10.1145/1151588.1151600. ISBN 1595934944 p. 73–84. [Online]. Available: https://doi.org/10.1145/1151588.1151600 [Page 2.]

[10] A. Carter, C. Hundhausen, and D. Olivares, *Leveraging the Integrated Development Environment for Learning Analytics*, ser. Cambridge Handbooks in Psychology. Cambridge University Press, 2019, p. 679–706. [Page 2.]

[11] A. Ahadi, R. Lister, and A. Vihavainen, "On the number of attempts students made on some online programming exercises during semester and their subsequent performance on final exam questions," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '16. New York, NY, USA: Association for Computing Machinery, 2016.

doi: 10.1145/2899415.2899452. ISBN 9781450342315 p. 218–223. [Online]. Available: https://doi.org/10.1145/2899415.2899452 [Page 2.]

[12] F. Pereira, S. Fonseca, E. Teixeira de Oliveira, D. de Oliveira, A. Cristea, and L. Carvalho, "Deep learning for early performance prediction of introductory programming students: a comparative and explanatory study," *Revista Brasileira de Informática na Educação*, vol. 28, pp. 723–749, 10 2020. doi: 10.5753/RBIE.2020.28.0.723 [Page 2.]

[13] Z.-H. Zhou, *Machine learning*. Gateway East, Singapore: Springer, 2021. ISBN 981-15-1967-6 [Pages 5, 9, 10, and 12.]

[14] C. M. Bishop, *Pattern recognition and machine learning*, ser. Information science and statistics. New York: Springer, 2006. ISBN 0-387-31073-8 [Page 6.]

[15] Javaatpoint, "Support vector machine algorithm," accessed: 2023-03-21. [Online]. Available: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm [Page 6.]

[16] M. Ahmed, "Non-linear support vector machines explained," sep 2021, accessed: 2023-03-21. [Online]. Available: https://linguisticmaz.medium.com/support-vector-machines-explained-ii-f2688fbf02ae [Page 6.]

[17] scikit learn, "Naive bayes," accessed: 2023-03-21. [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html [Page 9.]

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org. [Page 10.]

[19] T. Hastie, *The Elements of Statistical Learning Data Mining, Inference, and Prediction*, ser. Springer Series in Statistics. New York, NY: Springer New York, 2001. ISBN 0-387-21606-5 [Page 11.]

[20] J. Li, K. Cheng, S. Wang, F. Morstatter, R. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM computing surveys*, vol. 50, no. 6, pp. 1–45, 2018. [Page 11.]

[21] A. Kumar, "Feature selection vs feature extraction: Machine learning," Mar 2023. [Online]. Available: https://vitalflux.com/machine-learning-feature-selection-feature-extraction/ [Pages 11 and 12.]

[22] W. Xiao, P. Ji, and J. Hu, "A survey on educational data mining methods used for predicting students' performance," *Engineering reports (Hoboken, N.J.)*, vol. 4, no. 5, 2022. [Pages 12, 15, 16, and 21.]

[23] A. Alhothali, M. Albsisi, H. Assalahi, and T. Aldosemani, "Predicting student outcomes in online courses using machine learning techniques: A review," *Sustainability*, vol. 14, no. 10, 2022. doi: 10.3390/su14106199. [Online]. Available: https://www.mdpi.com/2071-1050/14/10/6199 [Pages 12, 15, and 20.]

[24] B. Prenkaj, P. Velardi, G. Stilo, D. Distante, and S. Faralli, "A survey of machine learning approaches for student dropout prediction in online courses," *ACM Comput. Surv.*, vol. 53, no. 3, may 2020. doi: 10.1145/3388792. [Online]. Available: https://doi.org/10.1145/3388792 [Pages 13 and 29.]

[25] C. C. Gray and D. Perkins, "Utilizing early engagement and machine learning to predict student outcomes," *Computers and education*, vol. 131, pp. 22–32, 2019. [Page 13.]

[26] W. Li, M. Gao, H. Li, Q. Xiong, J. Wen, and Z. Wu, "Dropout prediction in moocs using behavior features and multi-view semi-supervised learning," in *2016 International Joint Conference on Neural Networks (IJCNN)*, vol. 2016-. IEEE, 2016. ISBN 1509006192. ISSN 2161-4407 pp. 3130–3137. [Page 13.]

[27] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: Association for Computing Machinery, 2006. doi: 10.1145/1143844.1143874. ISBN 1595933832 p. 233–240. [Online]. Available: https://doi-org.focus.lib.kth.se/10.1145/1143844.1143874 [Page 13.]

[28] C. Brooks and C. Thompson, "Predictive Modelling in Teaching and Learning," in *The Handbook of Learning Analytics*, 1st ed., C. Lang, G. Siemens, A. F. Wise, and D. Gaševic, Eds. Alberta, Canada: Society for Learning Analytics Research (SoLAR), 2017, pp. 61–68. ISBN 978-0-9952408-0-3 [Page 14.]

[29] Y.-H. Hu, C.-L. Lo, and S.-P. Shih, "Developing early warning systems to predict students' online learning performance," *Comput. Hum. Behav.*, vol. 36, no. C, p. 469–478, jul 2014. doi: 10.1016/j.chb.2014.04.002.

[Online]. Available: https://doi.org/10.1016/j.chb.2014.04.002 [Page 15.]

[30] G. Akçapınar, M. N. Hasnine, R. Majumdar, B. Flanagan, and H. Ogata, "Developing an early-warning system for spotting at-risk students by using ebook interaction logs," *Smart Learning Environments*, vol. 6, no. 1, p. 4, May 2019. doi: 10.1186/s40561-019-0083-4. [Online]. Available: https://doi.org/10.1186/s40561-019-0083-4 [Page 15.]

[31] C. Herodotou, M. Hlosta, A. Boroowa, B. Rienties, Z. Zdrahal, and C. Mangafa, "Empowering online teachers through predictive learning analytics," *British Journal of Educational Technology*, vol. 50, no. 6, pp. 3064–3079, 2019. doi: https://doi.org/10.1111/bjet.12853. [Online]. Available: https://bera-journals.onlinelibrary.wiley.com/doi/abs/10.1111/bjet.12853 [Page 15.]

[32] Loginova, Ekaterina and Benoit, Dries, "Embedding navigation patterns for student performance prediction," in *Proceedings of The 14th International Conference on Educational Data Mining (EDM 2021)*, Hsiao, I-Han and Sahebi, Shaghayegh and Bouchet, Francois and Vie, Jill-Jenn, Ed. Educational Data Mining Society, 2021, pp. 391–399. [Online]. Available: https://educationaldatamining.org/EDM2021/virtual/static/pdf/EDM21_paper_17.pdf [Page 15.]

[33] Y. Dong, S. Marwan, P. Shabrina, T. Price, and T. Barnes, *Using Student Trace Logs to Determine Meaningful Progress and Struggle during Programming Problem Solving*. International Educational Data Mining Society, 2021. [Pages 15 and 29.]

[34] D. Tzimas and S. Demetriadis, "The impact of learning analytics on student performance and satisfaction in a higher education course," in *Proceedings of The 14th International Conference on Educational Data Mining (EDM 2021)*, 2021, pp. 654–659. [Page 15.]

[35] M. Fateen and T. Mine, "Predicting student performance using teacher observation reports." *International Educational Data Mining Society*, 2021. [Page 15.]

[36] C. Romero and S. Ventura, "Educational data mining and learning analytics: An updated survey," *WIREs Data Mining and Knowledge Discovery*, vol. 10, no. 3, p. e1355, 2020. doi: https://doi-org.focus.lib.kth.se/10.1002/widm.1355. [Online]. Available:

https://wires-onlinelibrary-wiley-com.focus.lib.kth.se/doi/abs/10.1002/widm.1355 [Page 15.]

[37] H. A. Mengash, "Using data mining techniques to predict student performance to support decision making in university admission systems," *IEEE Access*, vol. 8, pp. 55 462–55 470, 2020. doi: 10.1109/ACCESS.2020.2981905 [Page 16.]

[38] B. K. Yousafzai, M. Hayat, and S. Afzal, "Application of machine learning and data mining in predicting the performance of intermediate and secondary education level student," *Education and Information Technologies*, vol. 25, no. 6, pp. 4677–4697, Nov 2020. doi: 10.1007/s10639-020-10189-1. [Online]. Available: https://doi.org/10.1007/s10639-020-10189-1 [Page 16.]

[39] E. Amrieh, T. Hamtini, and I. Aljarah, "Mining educational data to predict student's academic performance using ensemble methods," *International Journal of Database Theory and Application*, vol. 9, pp. 119–136, 09 2016. doi: 10.14257/ijdta.2016.9.8.13 [Pages 16 and 25.]

[40] K. Castro-Wunsch, A. Ahadi, and A. Petersen, "Evaluating neural networks as a method for identifying students in need of assistance," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '17. New York, NY, USA: Association for Computing Machinery, 2017. doi: 10.1145/3017680.3017792. ISBN 9781450346986 p. 111–116. [Online]. Available: https://doi.org/10.1145/3017680.3017792 [Pages 16, 26, and 28.]

[41] A. Ahadi, R. Lister, H. Haapala, and A. Vihavainen, "Exploring machine learning methods to automatically identify students in need of assistance," in *ICER '15*. United States: ACM New York, Aug. 2015. doi: 10.1145/2787622.2787717. ISBN 978-1-4503-3630-7 pp. 121–130, jufo-id:70610 ; ACM International Computing Education Research ; Conference date: 09-08-2015 Through 13-08-2015. [Page 16.]

[42] M. Kloft, F. Stiehler, Z. Zheng, and N. Pinkwart, "Predicting MOOC dropout over weeks using machine learning methods," in *Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014. doi: 10.3115/v1/W14-4111 pp. 60–65. [Online]. Available: https://aclanthology.org/W14-4111 [Page 16.]

[43] W. Wang, H. Yu, and C. Miao, "Deep model for dropout prediction in moocs," in *Proceedings of the 2nd International Conference on Crowd Science and Engineering*, ser. ICCSE'17. New York, NY, USA: Association for Computing Machinery, 2017. doi: 10.1145/3126973.3126990. ISBN 9781450353755 p. 26–32. [Online]. Available: https://doi.org/10.1145/3126973.3126990 [Page 17.]

[44] S. Ameri, M. Fard, R. Chinnam, and C. Reddy, "Survival analysis based framework for early prediction of student dropouts," in *International Conference on Information and Knowledge Management, Proceedings*, ser. CIKM '16, vol. 24-28-. ACM, 2016. ISBN 9781450340731 pp. 903–912. [Page 17.]

[45] J. Berens, K. Schneider, S. Gortz, S. Oster, and J. Burghoff, "Early detection of students at risk – predicting student dropouts using administrative student data from german universities and machine learning methods," *Journal of educational data mining*, vol. 11, no. 3, pp. 1–, 2019. [Page 17.]

[46] L. Haiyang, Z. Wang, P. Benachour, and P. Tubman, "A time series classification method for behaviour-based dropout prediction," in *2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT)*. IEEE, 2018. ISBN 9781538660492. ISSN 2161-377X pp. 191–195. [Page 17.]

[47] A. Wolff, Z. Zdrahal, A. Nikolov, and M. Pantucek, "Improving retention: Predicting at-risk students by analysing clicking behaviour in a virtual learning environment," in *Proceedings of the Third International Conference on Learning Analytics and Knowledge*, ser. LAK '13. New York, NY, USA: Association for Computing Machinery, 2013. doi: 10.1145/2460296.2460324. ISBN 9781450317856 p. 145–149. [Online]. Available: https://doi.org/10.1145/2460296.2460324 [Page 17.]

[48] C. Robinson, M. Yeomans, J. Reich, C. Hulleman, and H. Gehlbach, "Forecasting student achievement in moocs with natural language processing," in *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, ser. LAK '16. New York, NY, USA: Association for Computing Machinery, 2016. doi: 10.1145/2883851.2883932. ISBN 9781450341905 p. 383–

387. [Online]. Available: https://doi.org/10.1145/2883851.2883932 [Page 17.]

[49] Y. Zhang, Y. Yun, R. An, J. Cui, H. Dai, and X. Shang, "Educational data mining techniques for student performance prediction: Method review and comparison analysis," *Frontiers in psychology*, vol. 12, pp. 698 490–698 490, 2021. [Page 20.]

[50] M. Sivasakthi and K. R. A. Padmanabhan, "Prediction of students programming performance using naïve bayesian and decision tree," in *Soft Computing for Security Applications*, G. Ranganathan, X. Fernando, and S. Piramuthu, Eds. Singapore: Springer Nature Singapore, 2023. ISBN 978-981-19-3590-9 pp. 97–106. [Page 26.]

[51] R. S. Baker, K. Yacef *et al.*, "The state of educational data mining in 2009: A review and future visions," *Journal of educational data mining*, vol. 1, no. 1, pp. 3–17, 2009. [Page 26.]

[52] C. Herodotou, G. Naydenova, A. Boroowa, A. Gilmour, and B. Rienties, "How can predictive learning analytics and motivational interventions increase student retention and enhance administrative support in distance education?" *Journal of Learning Analytics*, vol. 7, no. 2, pp. 72–83, 2020. [Page 27.]

[53] C. Herodotou, B. Rienties, A. Boroowa, Z. Zdrahal, and M. Hlosta, "A large-scale implementation of predictive learning analytics in higher education: the teachers' role and perspective," *Educational Technology Research and Development*, vol. 67, no. 5, pp. 1273–1306, Oct 2019. doi: 10.1007/s11423-019-09685-0. [Online]. Available: https://doi.org/10.1007/s11423-019-09685-0 [Page 27.]

[54] R. L. Figueroa, Q. Zeng-Treitler, S. Kandula, and L. H. Ngo, "Predicting sample size required for classification performance," *BMC Medical Informatics and Decision Making*, vol. 12, no. 1, p. 8, Feb 2012. doi: 10.1186/1472-6947-12-8. [Online]. Available: https://doi.org/10.1186/1472-6947-12-8 [Page 28.]

[55] C. Márquez-Vera, A. Cano, C. Romero, A. Y. M. Noaman, H. Mousa Fardoun, and S. Ventura, "Early dropout prediction using data mining: a case study with high school students," *Expert Systems*, vol. 33, no. 1, pp. 107–124, 2016. [Page 29.]

[56] I. Khan, A. R. Ahmad, N. Jabeur, and M. N. Mahdi, “Machine learning prediction and recommendation framework to support introductory programming course,” *International journal of emerging technologies in learning*, vol. 16, no. 17, pp. 42–59, 2021. [Page 29.]

TRITA-EECS-EX-2023:962