TEACHING AND LEARNING DATA-DRIVEN MACHINE LEARNING WITH EDUCATIONALLY DESIGNED JUPYTER NOTEBOOKS

YANNIK FLEISCHER
Paderborn University
yanflei@math.uni-paderborn.de

ROLF BIEHLER
Paderborn University
biehler@math.uni-paderborn.de

CARSTEN SCHULTE
Paderborn University
carsten.schulte@uni-paderborn.de

ABSTRACT

This study examined upper secondary students' modeling with machine learning algorithms. In the context of a yearlong data science course, the study explored how students applied machine learning using Jupyter Notebook and how they documented the modeling process as a computational essay taking into account the different steps of the CRISP-DM cycle. The students' work was based on a teaching module about automatically created decision trees as a machine learning method. Worked examples were used to support students' modeling processes. The study investigated the students' performance in technically carrying out the machine learning methods and their reasoning about different data preparation steps, bias in the data, the application context, and the resulting decision model.

Keywords: *Statistics education research; Machine learning; Decision trees; Jupyter Notebook*

1. INTRODUCTION

"All models are wrong, but some are useful" (Box & Draper, 1987, p. 424). This aphorism reminds us that even elaborated data-driven decision models are useful but limited. This is a crucial insight into a digitalized world. Models are widely used for automated decision-making in medicine, criminalistics, or internet platforms. The relevance of these data-driven decision models for individuals and society demands more data science education at the school level (Biehler & Schulte, 2018; Engel, 2017; Engel et al., 2019; Ridgway, 2016; Ridgway et al., 2018).

Many of these decision models are built with a large amount of meaningful data and powerful machine learning methods for predictive modeling. Literacy concepts like statistical literacy, data literacy, or data awareness focus on different facets of data science while also having significant overlaps. However, although predictive modeling is an essential facet of data science education (Ridgway et al., 2018), most literacy concepts mainly focus on the awareness, manual handling, and interpretation of data. All these concepts foster critical skills and competencies, but they focus to a lesser extent on predictive modeling (or machine learning). For instance, a framework in statistics education (Philip et al., 2013) focuses on big data as a subject for education, where the predictive modeling side does not play an explicit role. Both parts are essential to enable students to be "critical consumers" when confronted with data-driven systems or media reports (Engel, 2017). It is a crucial competence to critically assess predictive models, know something about possible errors, and have a sense about which tasks can be solved appropriately with them. This is addressed by some recent approaches aiming at machine learning literacy (Register & Ko, 2020) or AI literacy (Long & Magerko, 2020). The focus in these approaches is the critical evaluation of machine learning models with their opportunities and risks, which includes reasoning about possible flaws in the data and the modeling process. Recent curricular frameworks also use machine learning as an advanced part of predictive modeling. In the *Pre-K–12 Guidelines for Assessment and Instruction in Statistics Education II*

(Bargagliotti et al., 2020), machine learning is an outlook for a level that outstanding students might reach. However, the International Data Science in Schools Project (IDSSP) Curriculum (IDSSP Curriculum eam, 2019) explicitly describes machine learning methods like automatically created data based decision trees as an essential part for all.

Referring to the aphorism from the beginning, our aim of teaching machine learning in school includes enabling students to recognize and reason about what might be wrong with a data-driven decision model and why it may (or may not) be useful. This objective is not easy to reach. Sulmont et al., (2019a, p. 953) found, for non-major students at the university level, that "it is possible to teach ML to those with little to no math/CS background" but that they are "having difficulty appreciating the human decision-making aspects of ML." Sulmont et al. (2019b) note that barriers for students are not related to understanding machine learning algorithms but that they get "stuck on the development and evaluation of models" (p. 14) Regarding teacher education, Zieffler et al. (2021) found that decision tree algorithms as such can be well taught in in-service teacher education, but the evaluation of decision trees poses a more significant challenge. We designed a course for upper secondary students to enable them to use machine learning for predictive modeling, e.g., in the form of automatically created decision trees, for modeling and to enable them to reason about the outcomes. This paper will analyze the work of students who documented their modeling and reasoning processes.

2. CONTEXT OF THE STUDY: THE COURSE ON DATA SCIENCE

The yearlong data science course for the upper secondary level in which our research took place was developed as part of the ProDaBi project (www.prodabi.de), an interdisciplinary joint project of didactics of mathematics and didactics of computer science at Paderborn University. We developed our material for a so-called project course in Grade 12, which was organized and taught in cooperation with two schools in Paderborn for the third time in the 2020/21 school year. Interested students could select such project courses. The final grading becomes part of the final secondary school examination (Abitur). We had 10 to 20 participants, but only three male students participated in the course this year. Students were able to choose the course if they previously attended computer science (including programming) as a subject, which is not obligatory in Germany, different from mathematics. The course aimed to teach students how to use data to solve a real-world problem with the aid of predictive modeling. An authentic and meaningful data project where students do all the work, from starting with raw data to the final product, underpinned the whole course. The final product from this year's class consisted of a web app (including a decision model based on machine learning), a computational essay, and a presentation explaining and documenting the data and the methods. Students worked with parking occupancy data dating back up to 12 years. The project's main objective was to design, implement, and document data-based predictive models that predict different parking spaces' occupancy in the next coming hours in real-time. The purpose of creating these models was to help people find a free place quicker to improve traffic flow in Paderborn and reduce emissions.

The didactic concept of the course was that students learn aspects of data exploration and machine learning processes during various well-prepared teaching modules and use their just gained knowledge and skills to work on their data project in more self-regulated project phases after each of these modules. The course structure is shown in Table 1, where the project phases are highlighted in green, and the teaching modules are highlighted in red. During the course, we used two digital tools. The first one is the web-based interactive data science platform CODAP (codap.concord.org), where students can explore data intuitively within a menu-based environment. The second one is Jupyter Notebook (JNB) (Toomey, 2017) with Python that students use as a programming environment and tool for doing, documenting, and commenting on data analyses. We, as teachers, use CODAP and JNB as pedagogical tools to prepare learning environments for students.

The directed teaching modules included data activities with CODAP, a brief introduction to programming with Python, and the use of different so-called tool JNBs. These tool JNBs are learning environments for students to learn data exploration and machine learning. For these JNBs, other data were used than the project data (e.g., multivariate data on students' media use (Behrens & Rathgeb, 2017)). Some of the tool notebooks explicitly use Python code, and others are menu-based with hidden Python code to ease access to the methods for students. Based on these materials, teaching sequences

were planned in which teachers gave inputs, interacted with students, and discussed results with the whole group.

During the project phases, the students' working process was oriented at the CRISP-DM cycle (Shearer, 2000) (see Figure 1), which comprised business understanding, data preparation, modeling, evaluation, and deployment. Some aspects of the professional process were reduced for school teaching. For example, the phase of modeling in the professional process included comparing and selecting models from different model classes to identify the best-performing model. In our course, the students successively learned about only two different model classes (decision trees, artificial neural networks) because our approach was to teach the methods used in-depth. In our opinion, two different methods sufficiently exemplify the process of machine learning. Using more models would have increased the complexity of the course and decreased the number of lessons for each technique. Even understanding the creation and evaluation of only one model type is challenging for students.

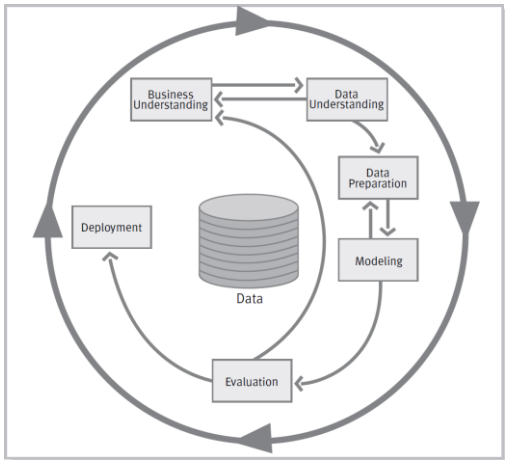


Figure 1. CRISP-DM cycle for data project (Shearer, 2000)

In the process, the students developed two milestone products (JNBs) as indicated in Table 1: The *first milestone* is a report about data explorations to identify variables that may influence parking space occupancy. The reports included the business understanding, the data understanding, and the first steps of the data preparation. The *second milestone* product is a report about a first predictive model using machine learning that includes further stages of data preparation, creating a decision tree model, and evaluating the model. The reports the students developed as milestones were documented as interactive data analysis reports in the sense of reproducible research. These reports are called computational essays. The concept of the computational essay is elaborated in Section 3.1.

Table 1. Structure of the project course

	Phase Type	Phase Content	Aim/Task	Resources	CRISP-DM
1	Project phase	Start	• understand the project context of predicting parking space occupancy • develop a vision of the final product		Business Under-standing
2	Teaching module	Data exploration with CODAP	• carry out simple data explorations and data moves to gain first data competences	CODAP	
3	Teaching module	Programming with Python	• learn basics of Python	Online Course	
4	Teaching module	Data exploration with JNB	• learn data exploration with Python • understand, clean, and prepare the project data	5 tool JNBs	

5	Project phase - Milestone 1	Data exploration for project data	• clean, prepare, and analyze the project data for getting insights into which variables influence occupancy of parking spaces • create a computational essay about business and data understanding		Data Under-standing, Data Preparation
			Milestone 1 – Data exploration		
6	Teaching module	Decision Trees	• understand the machine learning method of decision trees	CODAP, 3 tool JNBs	
7	Transition phase	Computational essay for the ML models	• explore worked example for computational essay for decision trees	Worked Example JNB	
8	Project phase -Milestone 2	ML modeling for project data	• continue working with data from milestone 1 • create a computational essay documenting a machine learning process with decision trees for the project data	Milestone 1 JNB, Worked Example JNB	Data Preparation, Modeling, Evaluation
			Milestone 2 – First predictive model		
9	Teaching module	Artificial neural networks (ANN)	understand the machine learning method of ANN	5 JNBs	
10	Project phase	Deployment of ML model	• create ANN to compare with the previously created decision tree • implement the best machine learning model into a web app	Milestone 2 JNB	Modeling, Evaluation, Deployment

This paper focuses on the products the students produced for milestone 2. At this stage of the course (after Phase 8), the students only knew decision trees as machine learning models and not yet neural networks. Thus, the products contained the students' first decision tree models enriched by comments and argumentations. To support and scaffold the self-regulated project work of milestone 2, the researchers prepared the worked example JNB beforehand. It contains the necessary programming commands to perform a machine learning process with decision trees, and additionally it provides explanations and a structure for documenting the process. Students were expected to use the prepared worked example JNB to adopt, adapt and enhance the programming commands they needed to work with their project data in a new notebook, and finally also to add comments and explanations to document and communicate their results in the form of a computational essay.

As a first step, all students created different decision trees, presented them to each other and decided which of the trees was the most suitable. Later in the course, they learned about artificial neural networks as the second model type. The project goal was to create predictive models of both types and embed the best model in a web app as the final deployment stage. During the course, we observed the following obstacle for the students. The students produced their first milestone JNB with a report on data understanding and business understanding. We found that the students were able to technically carry out a data exploration based on the tool JNBs from the teaching module. After that, they were also able to explain many of their thoughts and findings orally. However, they had not put these insights in the written report (computational essay)different from what we asked them to do. (Rule et al., 2018) reported a similar finding. They found that documentation in computational notebooks was not naturally given, nor was it easy to "clean" and streamline working notebooks and find a suitable form of explanation, particularly if the audience is diverse. They recommended more guidance and support for such documentation. We reacted to these findings during the course by adding a transitional phase (highlighted in yellow in Tabel 1) for which we created a worked example that illustated how a computational essay in a JNB might look. Before describing this transitional phase and its worked example in detail, we will give some theoretical background about the concept of a computational essay (see Section 3.1), worked examples (see Section 3.2), and the potential of JNBs (see Section 3.3).

3. SUPPORTING MATERIAL AND THEORETICAL BACKGROUND

3.1. THE CONCEPT OF A *COMPUTATIONAL ESSAY*

The term computational essay was initially introduced by diSessa (2000, p. 185), meaning an essay contains small programs producing content for the reader to play with. Additionally, it contains integrated text, hypertext, diagrams, and pictures for explanation and reasoning. diSessa proposed this format for presenting scientific insights (especially for students) or instructing how to solve a task that requires programming and visualizations to find a solution. McNamara (2019) builds on this idea when arguing for a technology supporting narrative, publishing, and reproducibility of data analyses. Reproducibility is a crucial benefit of computational essays in *Juptyter Notebook* (or with the mark-down language in R) compared to data analysis with other tools. In this context, reproducibility can be understood in two ways: first as repeatability and second as comprehensibility. In a JNB, all steps of creating the final product are documented in the correct order (sequence of ordered code cells) so that it can be repeated with the same (or similar) data to reproduce the same (or analogous) results technically. Explanatory elements can be added in cells by written comments and interpreted visualizations. Ideally, these comments form a coherent narrative, describing steps, explaining statistical arguments, and assessing results considering the actual context. The reader can comprehend the decision-making and reasoning undertaken throughout the process.

The use of computational essays in a class can offer different pedagogical advantages. Students can use pre-written code from prepared computational essays to adopt, adapt, and enhance it for their own data analyses. Furthermore, when the students create their computational essays to document their results, they need to reflect on their written code and the results of their data analysis more deeply and have to document their reasoning. As indicated above, we use computational essays in two functions. We prepare worked examples with computational essays for the students to learn from. Students could then use these worked examples as models for creating their own computational essays for the project data.

3.2. THE EDUCATIONAL USE OF WORKED EXAMPLES

We root our approach to worked examples in the research literature on using worked examples in teaching. The effectiveness of worked examples, especially for teaching problem-solving tasks, is not in dispute (Atkinson et al., 2000). Worked examples are particularly suitable for far transfer tasks that require contributions from students beyond simple adaptions(van Gog et al., 2006). They typically specify the problem to solve, formulate criteria of a solution, offer a concrete solution, and show the steps to find this solution. A distinction is made between process-oriented and product-oriented worked examples (van Gog et al., 2008). While product-oriented worked examples consist of the components just mentioned, process-oriented worked examples contain further information on how the solution steps are carried out and why they work. Process-oriented worked examples are especially appropriate for novice learners working on the task for the first time. In contrast, product-oriented worked examples are more suitable for students with at least some prior knowledge because redundant information that the students already know can hamper the efficiency of learning and solving the problem (van Gog et al., 2008).

We created a worked example in the form of a computational essay to guide the students in producing their own computational essay. The worked example is based on data different from the project data so that the students have to transfer the machine learning modeling to the project context (Milestone 2). It was comprised of components of every step of the CRISP-DM cycle for decision tree modeling except the deployment of the prediction model in a webapp that follows in later sections of the course. Data and business understanding are only mentioned briefly. The worked example can be seen in two ways. It is a process-oriented worked example for carrying out machine learning-based modeling, and a product-oriented worked example for creating a computational essay. A crucial aspect of its design was the degree of guidance that it provided. Our objective was to provide orientation and not narrowly prescribe every necessary step.

3.3. THE *PRODABI DECISION TREE JUPYTER NOTEBOOKS* AND THE PYTHON PACKAGE *PYTREE* AS TOOLS FOR LEARNING AND APPLYING DECISION TREES

Jupyter Notebook (JNB) (Toomey, 2017) is a cell-based programming environment with the possibility to write and execute code, visualize the output of the code right below, and includes hyperlinks, notes, and pictures interspersed. *Python* code can be executed in this environment. Different Python packages (pandas, plotly, numpy, etc.) can be imported to access a powerful data analysis tool with high-level programming commands.

We distinguish several notebook types as teaching material for learning and applying decision trees: a) didactically designed "tool notebooks," these are JNB that contain working environments for data exploration and machine learning modeling, b) the "worked example notebook" that contained a fully elaborated example of applying a set of tools to a data set including narrative explanations (see Section 5.2), and c) two milestone computational essay notebooks, which were to be developed by the students (see Section 2).

The tool notebooks are a source for creating the other notebooks. The introductory tool notebook for semi-automatically creating and optimizing decision trees is menu-based to ease access for students. We call it the *Basic ProDaBi Decision Tree JNB*. It can be used to develop and change trees semi-automatically or use the fully automated tree algorithm. We use a Python package (PyTree) that we specially designed for teaching decision trees in these notebooks. Our decision tree package has several educational benefits compared to standard Python packages for decision trees (e.g., *sci-kit learn*). To name the most important, our package PyTree is compatible with the *pandas* package that we use for data exploration and PyTree can handle both categorical and numerical variables. The created decision trees are visualized in a simple but meaningful way and allow students to create, edit, and prune trees manually. The programming commands we designed for these activities are relatively simple. The decision tree algorithm we implemented in PyTree is related to the algorithms ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), and CART (Breiman et al., 1998).

For solving a classification problem, training data is used to examine and compare different predictor variables to find the ones best suited to predict a target variable. A predictor variable is evaluated by the misclassification rate (optionally entropy can be used) it generates when used for a decision rule. To assess the misclassification rate for a specific predictor variable the data is divided by grouping cases according to the respective values and determining the misclassification rate in the partial data sets based on majority class decisions. In a greedy manner, the best-rated predictor variable is placed at the top of the tree and each of the partial data sets is used to continue the process. Recursively the data is further partitioned, and more variables are selected for the following levels of the tree structure until the training data is perfectly classified or no variable is left. This approach often leads to overfitting the training data, which can be revealed using the constructed tree on test data. For instance, Figure 2 shows the evolution of the rates of correct classifications for training and test data during the training process for one of our examples.

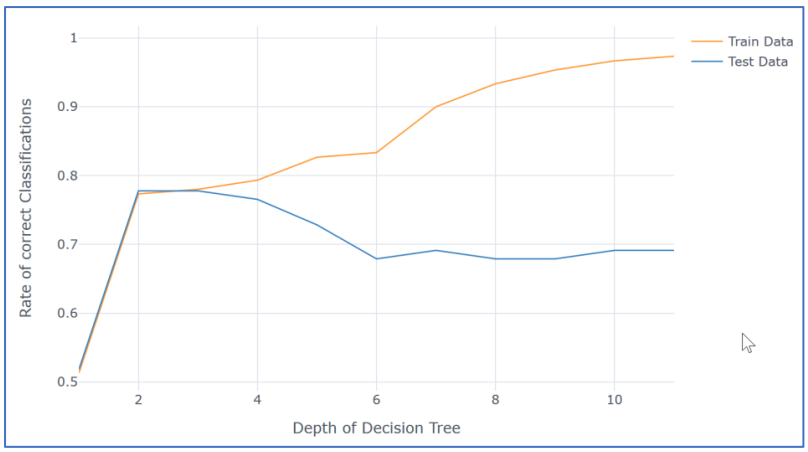


Figure 2. Evolution of rate of correct classifications during the training process of a decision tree (Visualization from the Basic ProDaBi Decision Tree JNB)

While the tree gets more levels during the training process, the training rate increases steadily, and the test rate also increases initially, but tends to decrease from a certain point onwards. This typical behavior is called overfitting, and it occurs because the decision rules added at the end are based on only small subsets of the original data set. Therefore, they are less generalizable to test data than the decision rules added first. Thus, the method of post-pruning (Quinlan, 1993, pp. 37–43), which retrospectively weakens this effect, is also implemented in our Python package. Bottom-up, all decision tree branches are assessed against test data, and if pruning a branch improves the misclassification rate for the test data, it is carried out. A more detailed description of our algorithm and its implementation is provided in the appendix.

The Basic ProDaBi Decision Tree JNB is an example of a menu-based JNB using our PyTree package to explore the concepts of overfitting and pruning. These menu-based notebooks are used in our teaching module for learning the crucial concepts before the students are asked to carry out a machine learning process independently. Students can create a decision tree semi-automatically by specifying the target variable, the criterion (misclassification rate or entropy), and the maximal depth of the tree in an interactive widget (see Figure 3, left). The trees in Figure 3 predict whether a person plays online games frequently or rarely based on data about a person's media behavior (e.g., whether a person watches LetsPlay Videos on YouTube frequently or rarely) and other personal data.

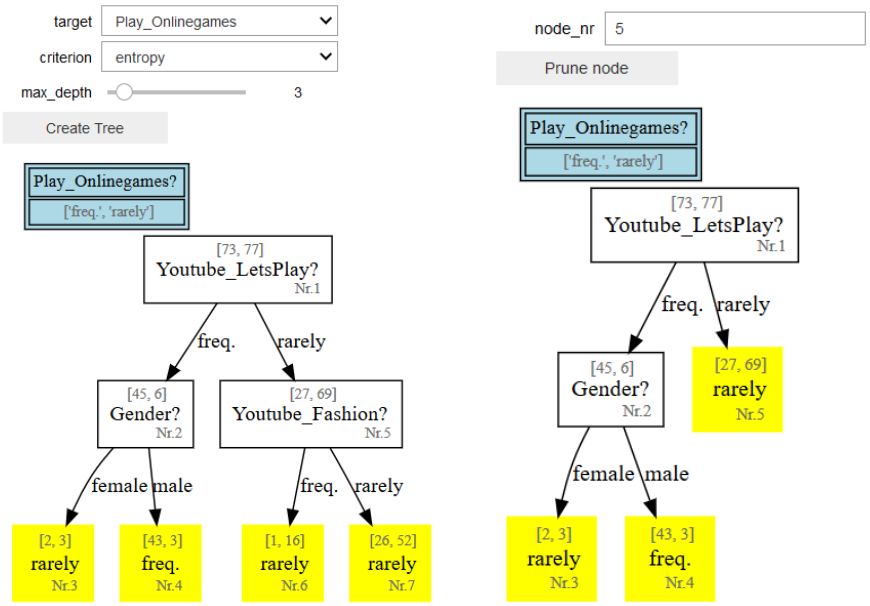


Figure 3. Excerpts of menu-based JNB for students to create decision trees (left) and prune them aftewards (right)

Afterward, the automatically created decision tree is visualized, and additional meaningful values and visualizations that characterize the training process are displayed in the notebook, as shown in Figure 2. The next step is to manually prune single nodes or subtrees with another widget (see Figure 3, right) by referring to node numbers. The effects on the misclassification rates of training and test data are shown after pruning a node. That way, students can combine an approach of pre-pruning (initial setting of a limit for tree growth) and post pruning (retrospective pruning of nodes and subtrees) to find a decision tree that fits the training data well and has no decrease in performance for the test data. After understanding the concept, the students also get access to an automated pruning process in the more advanced notebooks.

4. RESEARCH QUESTION

We intend to study how our teaching module (including our ProDaBi Decision Tree JNBs) and our worked example support students. Since Sulmont et al. (2019a) and Zieffler et al. (2021) found that it

is challenging for students to reason about machine learning models, we are interested in how a worked example supports upper secondary students in creating, evaluating, and reasoning about a decision tree model after attending our teaching module. We wonder if they can use an interactive worked example to transfer a machine learning modeling process to a new problem and reason about it adequately. Sulmont et al. (2019b) found that the students do not have problems with individual concepts concerning creating and evaluating machine learning models. Still, they struggle with the complexity of the overall process. Therefore, we tried to create an environment where handling the complexity of the process is supported by a scaffolding framework in which the students have to reason with individual concepts. We want to provide guidance that gives orientation but not a narrow prescription. We decided to do this by a worked example in the form of a computational essay (see Section 3).

For evaluating students' work, we distinguish between *technical functionality* and *argumentation and narrative quality* of computational essays. The core components of executable code concerning data preparation and application of machine learning algorithms must be technically functional (*technical functionality*). The remaining components, consisting of code for visualizations and explanatory text, extend the essay by adding argumentation and reasoning. One part of argumentation is derived from the context, such as a description of the real-world application and the data source, the formulation of objectives, selecting suitable statistical measures as quality criteria, and a conclusion on whether these have been achieved. Another part of the argumentation is technical and inner-mathematical. It evaluates the created decision tree against the chosen statistical quality criteria and general criteria of machine learning models (*argumentation and narrative quality*).

We are interested in how an interactive worked example can support students in carrying out (technical process support) and documenting their reasoning (narrative support) about a machine learning process. Thus, we study the following aspects:

1. *Technical process support.* In which sense are the students supported by the worked example to technically create a suitable decision tree model by adopting, adapting, enhancing, and adding technical steps of data preparation and applying machine learning algorithms?
2. *Narrative support.* In what sense are the students supported using the worked example to document their reasoning about their decision tree model and form a narrative concerning their statistical analyses? How do they adapt their reasoning to the new problem context by enhancing argumentations of the worked example and adding their own argumentations?
3. In what ways can the observed shortcomings be improved by redesigning the scaffolding worked examples as compared to those which may be due to the learning and teaching in the course?

5. DESIGN AND METHODS OF THE EMPIRICAL STUDY

For analyzing the JNB that students produced, it is necessary to understand the details of the instructional module on decision trees (see Section 5.1) and the details of the worked example with which the students worked (see Section 5.2), which are based on the research literature and our research questions.

5.1. THE MACHINE LEARNING MODULE ABOUT DECISION TREES

Before the students carry out their modeling using machine learning, they attend our more directed teaching module about decision trees. In preceding sessions, the students learned the basics of Python, gained skills in using CODAP and JNBs, examined data sets posing statistical questions, and cleaned and explored the project data with JNBs for getting a first overview of patterns in parking space data. Before the decision tree module, the students reach the first milestone of producing a report of their data exploration. The teaching module consists of 8 lessons of 45 minutes. It consists of 6 sections, as shown in Table 2, and is described in greater detail in Biehler & Fleischer (2021).

Table 2. Structure of the teaching module about decision trees

Section	No. of lessons	Topic
1	1	Introductory example of a decision tree – How to describe classification problems by data and solve them using machine learning
2	2	Exploratory construction of decision trees with CODAP
3	1	Systematizing the construction process and development of formal criteria to understand an automated decision tree algorithm
4	2	Automated creation of trees with the *ProDaBi Decision Tree Jupyter Notebook* and evaluation with test data to acquire the concepts of overfitting and pruning
5	1	Evaluation of decision tree models: precision and sensitivity, weighing of different kinds of errors depending on the context
6	1	Discussion of how using decision trees might impact society.

As the final part of this instructional module, the students get a worked example of the entire modeling process using machine learning as a computational essay. The worked example serves as a learning object where students can deepen their gained knowledge about how a machine learning model is created and evaluated. All individual aspects of the process mentioned in the worked example were discussed in class before. The students use the worked example by comprehending the programming commands and explanations, finding self-explanations, completing gaps in the worked example, and discussing their insight in class. After this transitional phase, the students are prepared to transfer their skills and competences to carry out a modeling process with the project data to be documented as a computational essay.

5.2. DESIGN AND STRUCTURE OF THE WORKED EXAMPLE

We created a worked example in the form of a computational essay in a Jupyter Notebook to provide guidance for two different tasks: carrying out a machine learning process for predictive modeling with decision trees (process-oriented) and creating a computational essay about the modeling process (product-oriented). The context of the predictive modeling is a personalized advertisement, which was widely used during the course. We used multivariate survey data (n = 215; 89 variables) that we collected ourselves based on the JIM study (Behrens & Rathgeb, 2017)Klicken oder tippen Sie hier, um Text einzugeben., which examines adolescents' media use. The concrete task shown in the worked example is creating a decision tree for predicting whether a person plays online games frequently or rarely, based on other media behavior data (social network use, YouTube video preferences, personal data, etc.).

We considered seven criteria in for designing our worked example. We derived three criteria with regard to the purpose of the worked example. We have chosen a (1) different context than the project context and different data so that students have enough room for their contributions when adapting the worked example. The worked example (2) provides the necessary code as code building blocks for all individual steps of the process. Additionally, (3) explanations give additional information about the reasoning of the concrete machine learning process and these explanations exemplify the degree of explanation expected from the students in their computational essays. Moreover, Atkinson et al. (2000) propose four main design criteria for worked examples, concerning the integration of example parts, the use of multiple modalities, the clarity of the subgoal structure, and completeness/incompleteness. (4) *Integrating of example parts* and using (5) *multiple modalities* are realized naturally because the code, the associated outputs, and text are integrated in JNBs. The (6) *clarity of the subgoal structure* is established by dividing the worked example JNB into seven different sections marked by meaningful headlines that indicate the respective sections' aim. The Notebook structure (see Table 3) demonstrates the idealized structure of a machine learning modeling process. In terms of (7) *Completeness/incompleteness*, Atkinson et al. (2000) suggest that at specific points, the incompleteness of descriptions and explanations can be beneficial because students have to fill the gap with self-explanations. Such self-explanations have been found to improve the learning of students (Wylie &

Chi, 2014). Therefore, besides completely formulated comments, certain comments in our worked example are incomplete.

The passages with incomplete comments are the formulation of quality criteria (section four), some aspects of the decision tree evaluation (sections five and six), and the final conclusion (section seven). In section four, the incomplete text cells are enriched with questions as orientation, while the other incomplete text cells only contain headlines as a hint for what should be commented on. The code cells in the worked example are all filled so that it is technically executable.

Table 3. Structure of the worked example (Computational essay in a JNB)

Section	Content	Cells	Completeness
1. Preparation of environment	Import necessary Python packages and the data to set up the Jupyter Notebook; Short explanation of the data source and content	2 code cells 4 text cells	complete
2. Explanation of the context	Explanation of the context of the predictive model that is to be created	1 code cell 3 text cells	complete
3. Data Preparation	Final data preparations steps like dropping, adding, or recoding variables and dividing data into training- and test data	7 code cells 9 text cells	complete
4. Formulation of quality criteria	Formulation of quality criteria that are to be reached by the model derived from the context	2 text cells	1 text cell incomplete (questions)
5. Training of decision tree and evaluation	Creation and evaluation of a first decision tree model with the PyTree package	10 code cells 5 text cells	1 text cell incomplete (only headline)
6. Optimization of decision tree and evaluation	Optimization of the decision tree by post-pruning and evaluation of a second decision tree model with the PyTree package	7 code cells 5 text cells	3 text cells incomplete (only headline)
7. Conclusion and export	Conclusion about the usability of the decision tree model and export of results	1 code cell 3 text cells	1 text cell incomplete (only headline)

The worked example contains 28 cells with executable code and 31 cells with text. The code cells execute the technical steps of preparing the environment, preparing the data, building the decision model using a decision tree algorithm, and visualizing meaningful intermediate results for evaluation and reasoning. The text cells are integrated in between the code cells to enhance the reproducibility of human decision-making by comments and to establish a whole narrative around the modeling process. Text cells are used for headlines and different types of comments. Figure 4 (right) displays an excerpt of the data preparation section of the worked example. Two text cells and one code cell are shown containing the Headline “Recode Variables”, Python code that carries out recoding of the target variable to a binary scale, and the explanations of why the recoding was done.

5.3. DATA COLLECTION AND METHODS OF DATA ANALYSIS

In the non-directed project phase, after the decision tree module, the students transfer the machine learning modeling process to the project context of predicting parking space occupancy. The students build on their previous work. They use the worked example JNB they explored before as an interactive template for creating their own computational essay JNB. They import the cleaned and prepared dataset from their milestone 1 products to continue working on it. The worked example JNB delivers all necessary steps of an idealized modeling process and provides examples for how detailed the different aspects of the process can be documented. The worked example JNB serves as basis for students’ computational essays.

Nevertheless, there is enough room for students’ own contributions because the project data and its context are new and require much adaption and enhancement. Creative personal contribution and well-founded argumentation based on the context are necessary for the adaption process. The students’ insights from the milestone 1 product are helpful for reasoning and the choice of predictor variables.

We collected the students' computational essay JNBs where they document their modeling and reasoning.

For characterizing and assessing students' contributions, we first categorized the individual cells of the students' computational essay JNBs in three ways: *type of content*, *type of adaption*, and *quality of content*. The first categorization of the cells is based on the distinction of *technical functionality* and *argumentation and narrative.* The category technical functionality included core components of executable code concerning data preparation and application of machine learning algorithms. In contrast, the category argumentation and narrative contained explanatory text and code for visualizations supporting argumentation. Each cell was coded as *adopted, adapted, enhanced,* or *added* in the second categorization. Adopting means just copying a cell from the worked example. Adapting means making simple adjustments to transfer the code or comment to another context without significantly changing the function or content. Enhancing means that parts of a cell are incorporated while own contributions are decisive. Added cells are completely new cells made up by a student. A third categorization assesses the quality of the content for enhanced and added cells. We use the categories *very good, minor weaknesses, major weaknesses,* and *deficient*. These categorizations are used for quantifying the quality of the students' essays. We use the category *type of adaption* to focus on the students' contributions that go beyond just adapting or adopting the worked example. The adoptions and adaptions were also necessary, but we wanted to assess the students' more sophisticated contributions, so we focused on the enhanced and added cells of the students' JNBs. To compare the different computational essays quantitatively, points were given for every enhanced or added cell using the category *quality of content* (very good ≙ 3 points, minor weaknesses ≙ 2 points, major weaknesses ≙ 1 point, deficient ≙ 0 points). We call the sum of these points for all cells the *enhancement score* of the computational essay. That way enhancements and additions of the students are counted if they contribute to the overall process in a meaningful way. Just adding random cells would not increase the score since these cells were qualified as deficient.

In Figure 4, three cells of a student solution (left) and the corresponding part of the worked example (right) are shown. This example can describe different categories of types of adaption. The first cell with the title was just adopted. The second cell consisting of code was enhanced because the commands from the worked example were taken and incorporated into a bigger code block that solved a different kind of recoding of a variable from the project data. The third cell with the comment was adapted because the arguments and explanations were taken from the worked example and adapted for the new context without adding argumentations. Adding further argumentations would have been desirable because the student chose to recode the numerical target variable of parking space occupancy into a categorical variable of four not equally spaced intervals, which become increasingly smaller the higher the occupancy rate became. The reason for this might be that in this context, it was more interesting to distinguish whether the parking space was completely full or not than to distinguish between 10 or 20 percent occupancy. If such reasoning had been documented, the text cell with the comment would have been categorized as enhanced.

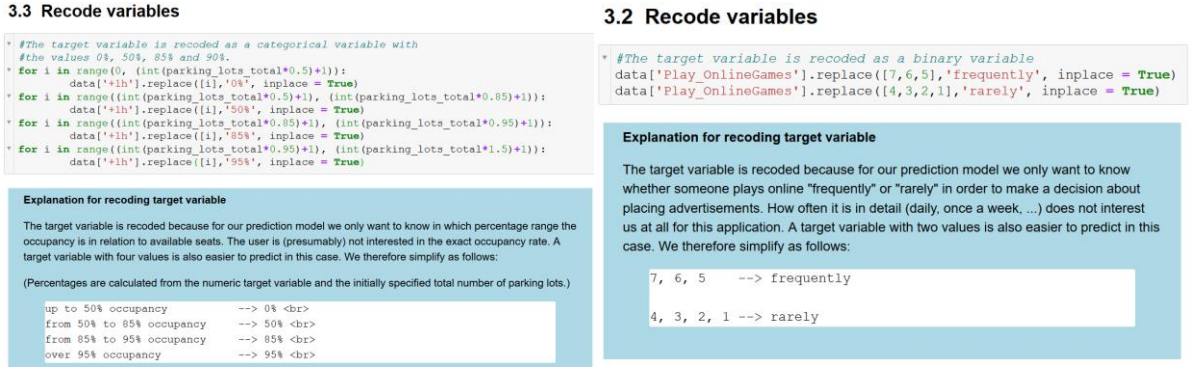


Figure 4. Excerpt of student's computational essay JNB (left) and corresponding part of worked example (right)

In this example, only the code cell is considered for the enhancement score because adopted and adapted cells are disregarded. The quality of the code cell is very good because the code does the intended recoding so that this excerpt of a student solution has an enhancement score of 3.

A s*tandard student solution* of adaption might be a computational essay with the same structure as the worked example. We expect 23 cells to be adopted (e.g., set up of the environment, headlines), 20 cells to be adapted (e.g., creating visualizations, applying the algorithm), and 16 cells to be enhanced. The enhancements of these 16 cells represent the main tasks for the students. Of course, completely different solutions are possible and permissible if students provide creative contributions of any kind. In a good standard solution, we expect the majority of enhancements to be very good or have minor weaknesses. If all enhancements (16 cells) are very good (minor weaknesses), a score of 48 points (32 points) would result. All scores in between would be seen as successful. All scores beyond 48 would be seen as exceptionally successful.

6. RESULTS

We examined the computational essays of our three students. Student 1 and Student 2 both created a solution very close to the good standard solution described above, while Student 3 created a more advanced solution. For the readability of this paper, one essay is presented in detail. In contrast, for the other two essays, the differences and unique contributions are highlighted compared to the first detailed description.

We focus here on the enhanced and added cells in the following to assess the students' advanced contributions. The more basic activities like adopting and adapting cells were very good for all three students. This is not self-evident because adoptions and adaptions need several insights and skills that are not naturally given. The students need to understand the overall process and also the code provided to decide whether to reuse the code. Additionally, they have to be able to adapt code by identifying and changing the crucial part of the code to get a technically functional computational essay. We do not expect that this was going to be easy for every student, but the three of our students who had an affinity for programming were able to do that equally well. Extensive parts of the code (technical functionality) needed adaptions. In the following, the technical functionality is discussed less than argumentations and narrative because enhancements and added cells were primarily found in argumentation.

6.1. COMPUTATIONAL ESSAY OF STUDENT 1

The raw data sets that the students got contained information about the occupancy of nine different parking spaces. For the actual task, only one parking space occupancy is focused on. The data set was cleaned and prepared in the previous phases of the course so that it then contained the occupancy of the parking spaces at hourly intervals over three years (n = 26280). The task is modeled as a classification problem that aims to predict occupancy one hour in advance as a target variable. One challenge that students had to solve was the recoding of the target variables. The numerical target variable of parking space occupancy had to be creatively recoded as a categorical variable for applying the classification algorithm. As predictors, seven variables such as actual occupancy, time of day, day of the week, and rate of change in the last hour were chosen. The ultimate goal of the modeling was to implement the created decision tree in a web app that helps users plan which parking space they go to. The students had to consider this ultimate goal for weighting different types of model errors. The computational essay of *Student 1* is structurally similar to the worked example because all cells containing headlines are adopted. The essay consists of seven sections with 58 cells (24 adopted, 19 adapted, 15 enhanced, 0 added). In Table 4, the contributions of Student 1 and the respective quality are displayed.

Table 4. Computational essay of Student 1

Section	Cells (enhanced/added)	Added or enhanced contributions (quality judgment)
1. Preparation of environment	2 code cells (0/0) 4 text cells (1/0)	• basic information about the data like the source and the meaning of the variables (very good) • basic information about the owning company and the method of data collection is given (very good)
2. Explanation of the context	1 code cells (0/0) 3 text cells (2/0)	• application context of predicting parking space occupancy is explained; app context missed (minor weaknesses) • classification problem is described; more than one target variable is defined without explanation (minor weaknesses)
3. Data Preparation	7 code cells (2/0) 9 text cells (2/0)	• solved the essential task of recoding the numerical target variable categorically using 10 equally spaced intervals to indicate the occupancy (very good) • further data preparations: recoding a predictor variable (actual occupancy), dropping two variables (very good) • additional explanations are not complete, certain steps are not explained while other steps are explained appropriately. This might be the result of an iterative trial and error process. (major weaknesses)
4. Formulation of quality criteria	2 text cells (1/0)	• addresses all questions and reasons about the context and the possible predictions of the decision tree. (minor weaknesses) o introduces a new and reasonable distinction for errors (deviation over 30 percent points is considered a critical error) o error of falsely predicting that the parking space is not full is declared the most relevant error o declares the statistical measures sensitivity and precision as irrelevant and the overall misclassification rate as only slightly meaningful o rejection of precision and sensitivity are not substantially explained
5. Training of decision tree and evaluation	10 code cells (0/0) 5 text cells (3/0)	• explains in a differentiated way and based on the quality criteria and the confusion matrix which crucial errors are found (very good) • reflects and adapts the quality criteria (very good) • explains the evolution of the training process and overfitting using visualizations. The argumentation shows some uncertainty about the assessment of overfitting (minor weaknesses)
6. Optimization of decision tree and evaluation	7 code cells (0/0) 4 text cells (3/0)	• explains the post-pruning method well (very good) • describes the different effects of the pruning correctly but remains insecure about overfitting (minor weaknesses) • The overall rate of correct classifications of only about 70 % seems not to be very high, but this impression is put into perspective (target variable with 10 small-step classes, better assessment by regarding numerical deviations) (very good) • critical errors are counted and presented with details so that only 90 out of 10960 classifications are identified as critical errors. (minor weaknesses)
7. Conclusion and export	1 code cell (0/0) 3 text cells (1/0)	• draws a conclusion that summarises the previous findings in a differentiated manner and recommends actual use of the application; suggests possible steps in better preparing the data for the future to improve the existing model (very good)

The student recoded the numerical target variable as a categorical variable with ten values representing 10 percent intervals of occupancy ratio each. The final decision tree of Student 1 consisted of 75 nodes and has a depth of 5. Figure 5 gives an impression of the complexity of the tree. The overall misclassification rate is around 30 percent, which seems to be high at first glance. However, for predicting 10 different classes of an ordinal scale, it is helpful to make a more differentiated evaluation than just regarding the misclassification rate.

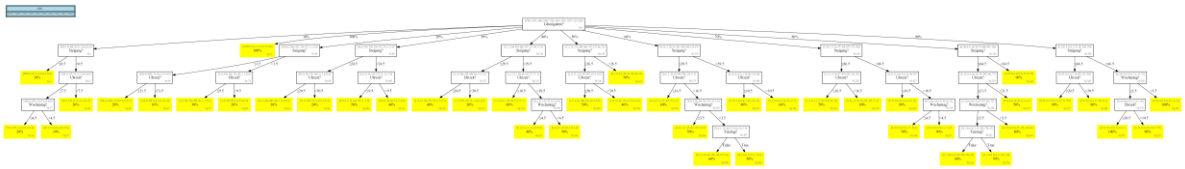


Figure 5. Final decision tree of Student 1

Student 1 puts this finding into perspective in the evaluation, as shown in the following in more detail. This example does not only illustrate our grading but shows a detailed example of how the student argued. Figure 6 shows two confusion matrices that the student visualized in the JNB. A confusion matrix indicates how many cases are classified correctly and wrongly by the decision tree and which types of errors occur in which frequency according to data. The confusion matrix has the dimensions 10 x 10 because the target variable has 10 different classes. The matrix rows represent the correct classes, and the columns represent the predicted classes. The classes each represent an interval of occupancy rate (e.g., 10% represents the interval between 0% and 10%). The right matrix in Figure 6 uses absolute numbers while the left uses row percentages. For instance, the right matrix indicated that 466 cases had the correct class 100% and also the predicted class 100% so that these cases were correctly classified as 100% by the tree. The matrix also indicated that 11 cases with the correct class 10% were falsely predicted as 30% (first row, third column). The student adapted these two visualizations from the worked example and gave an enhanced explanation (see Figure 6 bottom).

prediction correct	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
10%	89.1%	10.3%	0.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
20%	8.0%	79.4%	11.9%	0.7%	0.1%	0.0%	0.0%	0.0%	0.0%	0.0%
30%	0.2%	15.1%	63.6%	17.9%	2.8%	0.3%	0.0%	0.0%	0.0%	0.0%
40%	0.4%	0.8%	24.6%	44.8%	27.4%	1.8%	0.1%	0.0%	0.0%	0.1%
50%	0.4%	0.1%	3.9%	10.7%	69.4%	14.4%	0.7%	0.0%	0.4%	0.0%
60%	0.1%	0.0%	0.8%	0.8%	24.3%	50.3%	20.8%	1.7%	1.2%	0.1%
70%	0.4%	0.0%	0.0%	0.0%	4.2%	20.5%	54.2%	14.3%	6.1%	0.4%
80%	0.3%	0.0%	0.0%	0.2%	1.0%	4.9%	33.4%	26.3%	28.7%	5.3%
90%	0.6%	0.0%	0.0%	0.0%	0.2%	1.9%	8.1%	11.2%	47.7%	30.2%
100%	4.2%	0.0%	0.2%	0.2%	0.0%	0.0%	0.6%	2.3%	19.1%	73.5%

prediction correct	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
10%	2119	246	11	1	0	1	0	1	0	0
20%	138	1373	205	12	1	0	0	0	0	0
30%	2	144	607	171	27	3	0	0	0	0
40%	3	7	206	376	230	15	1	0	0	1
50%	3	1	32	87	565	117	6	0	3	0
60%	1	0	6	6	189	391	162	13	9	1
70%	3	0	0	0	34	165	436	115	49	3
80%	2	0	0	1	6	30	206	162	177	33
90%	3	0	0	0	1	10	42	58	246	156
100%	28	0	1	1	0	0	4	15	126	486

Check quality criteria

As described above, I first look at the most devastating misclassifications that lead to high percentage deviations of 30 % an more. There are not many such errors and this is a very good, but there are errors that were not initially considered and must be weighted much more critically and heavily. These include two types of errors that must be avoided at all costs: the car park is predicted to be 90% to 100% full but is relatively empty (below 40%). The car park is predicted to be 10% to 60% full, but is full (at least 90%).

These cases are also very rare, but should be weighted accordingly higher, because ultimately our system should only be able to predict whether a parking space is still usable or not.

According to my own quality criteria, we have a total of 87 critical errors, in absolute terms, which represents the success of the decision tree in the subject context, which is a very good rate considering how many thousands of data we work with.

The errors are made up of the 100% predictions where 80% or less are correct (38), and the reverse case where 80% or less is predicted although 100% is correct (49). Thus, for the test data (10960 records), my false classifications are only about 0.8%.

Figure 6. Excerpt of the evaluation of Student 1

The student plausibly resumes that there are not many strong errors (deviation of at least 30%). If classifications with at least 30% deviation are considered critical, the rate of critical classification errors would be below 1%. However, the student did not further justify this choice of critical errors. Furthermore, the student defined two types of crucial errors depending on the context (see Figure 6). The explanation provided a differentiated evaluation with creative use of the confusion matrix considering the context, but also with slight inconsistencies in describing the crucial errors in the first and last paragraphs of the comment. That is why this enhancement was categorized as having minor weaknesses.

In Table 5, the quantitative quality of the students' notebook is displayed, taking into account all enhanced and added cells and calculating the enhancement score as described above (see Section 5.3). His overall enhancement score was 37. The solution of Student 1 was overall successful and fulfilled all aspects of a good standard solution. The differentiated narrative was highlighted as particularly good. The technical functionality was represented to a lower degree in this score because most of the technical parts did not need enhancement but adaptions. This lower representation does not mean that the adaptions were not a crucial part of the solution.

Table 5. Student 1: Assessment of all enhanced or added cells of the computational essay

	deficient	major weaknesses	minor weaknesses	very good	(points)
technical functionality	0	0	0	2	(6)
argumentation and narrative	0	1	6	6	(31)

6.2. COMPUTATIONAL ESSAY OF STUDENT 2

The computational essay of Student 2 was also structurally similar to the worked example and, therefore, also similar to the solution of Student 1. It consisted of 59 cells, 24 adopted from the worked example, 19 adapted, and 16 enhanced from the worked example. In the following, only the differences to the solution of Student 1 are presented.

In section 2, Student 2 mentioned the intended web app and user behavior (this aspect of the narrative: very good). In section 3, he converted the numerical target variable that indicated the parking space's occupancy into a four-stage categorical variable (0–33%, 34–66%, 67–90%, 91–100%). The same was carried out for the predictor variable of the actual occupancy, but he did not comment on this step adequately, so argumentation was deficient. The student could have given a reason for this recoding by referring to a contextual argument or just to a result of a trial-and-error approach. The technical functionality concerning data preparation aspects was very good. Section 4 was solved similarly to Student 1, i.e., the statistical measures "sensitivity" and "precision" were also considered as not relevant (minor weaknesses). Student 2 also assessed the error of predicting the parking space as falsely not complete as the most severe error (very good). In section 5, the argumentation about overfitting was very clear (very good), and the student was confident in his judgment. However, assessment of the different types of errors using the confusion matrix was very short, in bullet points, and not very differentiated (major weaknesses). The conclusion in section 7 was also relatively quick and less differentiated than the conclusion of Student 1. The final rate of correct classifications was higher for Student 2 with about 86%. However, Student 2 recommended not using the model in the actual web app because the number of critical errors he found was too high, which was a reasonable conclusion.

In Table 6, the assessment of the computational essay of Student 2 is documented. Student 2 shows some major weaknesses in using the confusion matrix and in the conclusion, which marked this computational essay weaker than the good standard solution but nevertheless successful according to the enhancement score of 34. A positive highlight was his argumentation about overfitting, which was the best of the three essays.

Table 6. Student 2: Assessment of all enhanced or added cells of the computational essay

	deficient	major weaknesses	minor weaknesses	very good	(points)
technical functionality	0	0	0	2	(6)
argumentation and narrative	1	2	4	6	(28)

6.3. COMPUTATIONAL ESSAY OF STUDENT 3

The computational essay of Student 3 was structurally inspired by the worked example but extended by several sections and cells. The JNB consisted of 96 cells, 23 of which were adopted, 36 were adapted, 23 were enhanced, and 14 were added. Compared to the solution of Student 1 and the worked example, section 5 was duplicated twice. The training and assessment of three different decision trees were carried out and compared in the solution of Student 3. The data were prepared in different ways (explained below) for each of these trees to examine which preparation was the most successful. A short section was added where the trees were compared, but the rest of the computational essay was structured as a worked example so that it consisted of 10 sections. Sections 6 to 8 were added while 9 and 10 corresponded to 7 and 8 from Student 1. In the following, the differences to the solution of Student 1 are presented. Similar aspects are left out.

In section 1, Student 3 showed his unique contribution by importing not only the prepared data of the project course but also additional data about weather conditions in Paderborn to improve the predictions. He acquired the data from the internet on his own. This section had two other unique enhancements. He reduced the data so that no data after February 2020 were used because of the lockdown due to COVID-19 began. The data about traffic seemed biased and not helpful in predicting traffic in times without lockdown. Furthermore, he prepared the environment with certain variables to make his work reproducible. He prepared his code in a way that it was easy to adapt for other parking spaces. This showed foresight and an understanding of the benefits of reproducibility.

In section 3**,** he converted the numerical target variable that indicated the parking space's occupancy into a categorical variable with four not equally spaced levels (0-50%, 51-85%, 86-95%, 96-100%). The same was carried out for the respective actual occupancy predictor variable. Student 3 commented on all steps adequately, referring to the iterative trial and error approach, which showed that this recoding improved the performance of the tree (argumentation: very good). The technical implementation of this data preparation was very good as well.

Section 4 was solved similarly to Student 1 (minor weaknesses). In section 5, the argumentation about overfitting was restrained because he communicated explicitly some uncertainties (argumentation: minor weaknesses). Another weakness was assessing different types of errors using the confusion matrix because this part is too short, but the aspects mentioned were reasonable and comprehensible.

In section 6, the predictor variable *time of day* was duplicated in the data. That made sense because the algorithm used the variable twice in one path of the tree. The corresponding comment was deficient because it did not explain the duplication.

In section 7, the duplicated time of day variable was incorporated while the predictor variables referring to the weather were dropped. However, the comment was deficient again. The decision to do that might be reasonable in a trial-and-error approach.

In section 8, Student 3 compared the three decision trees based on the assessments from sections 5, 6, and 7 concluding that the decision tree from section 7 was the best, which is reasonable based on a comparison of the misclassification rates.

In section 9, the chosen decision tree was optimized. Some minor weaknesses were found in describing the pruning process where the terms training- and test data were confused. A final rate of correct classification of about 92% was found, which is the best of all decision trees created. The assessment of different types of errors is adequately detailed, and the number of critical errors (1,7%) and minor critical errors (6%) was presented in a differentiated way (argumentation: very good.) The conclusion in section 10 (former 7) was relatively short but was based on the former assessments and recommended using the decision tree for a test in the actual web app because the number of critical errors found was low enough, which was reasonable.

In Table 7, the results are summarized, indicating the enhancement score. The computational essay of Student 3 (84 points) was highlighted as exceptionally successful because it contained many exceptional contributions that were not expected beforehand improving the outcome and showing a deeper understanding of crucial aspects. Particular highlights were his acquisition of additional data, excluding parts of the data because of bias, and the detailed documentation of an iterative trial and error approach in the modeling process. Some shortcomings were identified in argumentation and narrative

because many comments were very short or just given in bullet points. This weakened the clarity of several explanations.

Table 7. Student 3: Assessment of all enhanced or added cells of the computational essay

	deficient	major weaknesses	minor weaknesses	very good	(points)
technical functionality	0	0	2	5	(19)
argumentation and narrative	2	2	15	11	(65)

6.4. SUMMARY OF RESULTS FROM THE PERSPECTIVE OF THE RESEARCH QUESTIONS

Technical process support. Our findings show that the technical functionality is successful for all of our students. They were able to adopt and adapt the code from the worked example to create a decision tree model and create visualizations for evaluation based on test data. Correctly adapting the code and knowing when just to adopt code indicates both, the understanding of the machine learning process and the understanding of the purpose and structure of the various code building blocks. Moreover, the students were able to enhance the code in crucial parts. A necessary enhancement was the recoding of the target variable, which required reasoned considerations about shaping the classes of the variable and the ability to use *Python* code to implement the recoding technically. Student 2 and Student 3 created four not equally spaced intervals of occupancy, while Student 1 created ten equally spaced intervals. All resulting decision trees had a satisfactory performance. For Student 1 and Student 2, not many contributions beyond the necessary steps were found, which is already an excellent achievement. However, Student 3 went beyond this and added several technical steps indicating his more profound understanding of the impact of data preparation for machine learning. Different data preparations were carried out in a trial-and-error approach to find data preparations that have beneficial effects on the predictive performance of the decision tree. As a result, his decision tree could be evaluated as the best-performing model among all students. Since none of the students had problems adapting and enhancing the code, we do not derive the necessity to revise the worked example with regard to the technical process support.

Narrative support. In investigating the students' reasoning, we found that all of our students were able to establish a narrative for their machine learning process. The argumentation was consistent and reasonable in large parts, while all students showed some weaknesses. The main topics to reason about were the explanation of data and context, data preparation, formulation of quality criteria (based on the confusion matrix and statistical measures), evaluation of overfitting, and evaluation of the model performance with own quality criteria. These topics vary in difficulty and degree of guidance from the worked example. The data, context, and data preparations were adequately described by all students. In the reasoning on the latter three topics, weaknesses occurred. These topics were addressed beforehand in the teaching module to enable students to find self-explanations, and therefore, due to incomplete comments, the topics were less guided by the worked example (see Table 3). However, the weaknesses that students showed in these areas were very diverse and did not dominate to a degree that would make the overall narrative deficient. While Student 2 excellently argued about overfitting, Student 1 had significant problems with this topic. It was the other way round for the evaluation of the model performance. Moreover, all students were to formulate their own quality criteria for their decision tree, derived from the context and based on the confusion matrix and statistical measures the students have learned about in the teaching module. Student 1 and Student 3 were very creative in their use of the confusion matrix for formulating reasonable quality criteria, while Student 2 also had reasonable criteria with a lower degree of creativity. However, a weakness that all students have in common in this part is the lack of including the statistical measures sensitivity and precision in their quality criteria. Although these measures were considered in the teaching module and the worked example, all students argued to

disregard them without a convincing reason. We assume that the students could not transfer their knowledge about these measures to the project context because the examples used in the teaching module were less complex (binary target variable). We will address this in the redesign of the course by also considering non-binary target variables in the teaching sequences. Since all students were able to establish a reasonable narrative but showed different weaknesses in their reasoning, it might be helpful to partly revise the design of the worked example with regard to narrative support. More details about the revisions are given below.

Hypotheses about causes for shortcomings and redesign. We observed some shortcomings in the explanation parts of the computational essays. Explanations with content weaknesses often also suffer from inadequate structure and form, e.g., they are written entirely in bullet points or bulleted sentences. In contrast, the majority of comments rated as very good are written in coherent, continuous text. The interpretation of why these weaknesses occur is difficult in many cases. It could be due to lack of understanding of the content, lack of ability to express thoughts adequately in written form, or not knowing the expectations for a particular comment. The first interpretation (lack of understanding) would suggest a revision of the instructional module, while the latter two (expression in written form, clarity of expectations) would suggest a revision of the worked example.

As described above, only one aspect (considering the specific statistical measures sensitivity and precision) was a weakness of all students, which will be addressed in the redesign of the teaching module. For each of the other contents, at least one student provided adequate explanation and reasoning. This might suggest that it was rather a weakness in the design of the worked example, which did not provide enough support for the documentation, than a weakness in the teaching module. The design criteria for creating the worked example, of which we have considered seven (see Section 5.2), might be decisive for its success so that they have to be critically revised. As our findings indicate, at least one criterion should be used with caution. Leaving gaps in explanations (criterion 7) for students to find self-explanations should be well dosed. The students' weaker comments were observed primarily where the worked example provided incomplete explanations. Comments with weaknesses in content and for rather occurred when the worked example did not provide an exemplary comment as orientation. In this study, it remains unclear whether the weaknesses were due to a lack of understanding or inadequate documentation of the reasoning. Nevertheless, we assume that more comprehensive explanations in these parts of the worked example would support students even more while still leaving them enough room for their own arguments and creative contributions since the actual model, confusion matrix, and context differ from the worked example. We will take this into account when redesigning the worked example and include some more detailed comments. In addition, we plan to add a second layer of explanation that includes meta-commentary on content and form expectations of the arguments: So far, the worked example explains how to do things; in future, we plan to add explanations on how to explain things. We will not only provide examples of good explanations but also meta-comments in which we then make explicit the expectations for adequate explanations. This includes the definition of the target audience, the expected content and form, and the actual purpose of reproducible research.

7. DISCUSSION

Using machine learning for predictive modeling requires two things: the technical ability to create a predictive model and the ability to reason about the process to decide whether the resulting model is useful or just wrong, to refer again to the aphorism from the beginning of this paper. Different studies similarly report that difficulties often occur with reasoning about machine learning processes (Sulmont et al., 2019b; Zieffler et al., 2021) and with documentation in computational notebooks (Rule et al., 2018). Our findings suggest that our approach consisting of three aspects (teaching module, study a worked example, use the worked example as a scaffold) to address these difficulties seems to be promising to support students in applying, documenting, and reasoning about a machine learning process; here exemplified by the machine learning method of decision trees.

In investigating our students' documentations, we distinguish between the technical creation of a model and the documentation of narrative and argumentation. The technical creation of a decision tree with competitive predictive performance and adequate visualizations was particularly successful since our students were able to do it in the expected way or even better. Furthermore, our students could

adequately reason about many aspects from data preparation, formulation of quality criteria, evaluation of overfitting, and the evaluation of the model performance on test data. Nevertheless, the documentation of the argumentation seemed to be more difficult for the students, as they each also showed certain weaknesses.

There are two possible interpretations of the students' weaknesses: either they had a lack of understanding, or they were not able to adequately document their reasoning in written form. This study leaves the question open which of these two interpretations is correct. However, we collected further data that might help to reveal the correct interpretation. The computational essays were also used as a presentation medium where the students explained their thoughts. The presentations were recorded and later, interviews on the students' impressions were carried out. Some informal investigations of the students' presentations seemed to indicate that some weaknesses rather occurred due to inadequate documentation than due to lack of understanding. In many cases, in the presentations, students elaborated on their reasoning that was more rudimentary in their written documentation. We are planning to investigate the recorded presentations systematically in another paper to analyse this impression in depth. For now, it cautiously indicates that a meta-commentary on expectations for comments (content and form) could strengthen the design of the worked example and better support students in documenting their reasoning.

Overall, the study indicated that the task of carrying out a modeling process of a decision tree using machine learning and documenting the reasoning in a computational essay is possible for students when supported by a worked example. However, we have only seen three students who have an affinity with the subject, so we do not know whether such results can be expected of all students of that age. Future research should investigate how this approach can work on a larger scale and with more diverse groups.

REFERENCES

Atkinson, R. K., Derry, S. J., Renkl, A., & Wortham, D. (2000). Learning from examples: Instructional principles from the worked examples research. *Review of Educational Research*, *70*(2), 181–214. https://doi.org/10/csm67w

Bargagliotti, A., Franklin, C., Arnold, P., Gould, R., Johnson, S., Perez, L., & Spangler, D. A. (2020). *Pre-K-12 guidelines for assessment and instruction in statistics education II (GAISE II)* (Second edition). American Statistical Association.

Behrens, P., & Rathgeb, T. (2017). *JIM-Studie 2017: Jugend, Information, (Multi-)Media, Basisstudie zum Medienumgang 12- bis 19-jähriger in Deutschland*. Medienpädagogischer Forschungsverbund Südwest.

Biehler, R., & Fleischer, Y. (2021). Introducing students to machine learning with decision trees using CODAP and Jupyter Notebooks. *Teaching Statistics*, *43*(S1), 133–142. https://doi.org/10.1111/test.12279

Biehler, R., & Schulte, C. (2018). Perspectives for an interdisciplinary data science curriculum at German secondary schools. In R. Biehler, L. Budde, D. Frischemeier, B. Heinemann, S. Podworny, C. Schulte, & T. Wassong (Eds.), *Paderborn Symposium on Data Science Education at School Level 2017: The Collected Extended Abstracts* (pp. 2–14). Universitätsbibliothek Paderborn.

Box, G. E. P., & Draper, N. R. (1987). *Empirical model-building and response surface*. Wiley.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1998). *Classification and regression trees*. Chapman & Hall/CRC.

DiSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. MIT Press.

Engel, J. (2017). Statistical Literacy for Active Citizenship: A Call for Data Science Education. *Statistics Education Research Journal*, *16*(1), 44–49. https://doi.org/10.52041/serj.v16i1.213

Engel, J., Biehler, R., Frischemeier, D., Podworny, S., Schiller, A., & Martignon, L. (2019). Zivilstatistik: Konzept einer neuen Perspektive auf data literacy und statistical literacy. *AStA Wirtschafts- und Sozialstatistisches Archiv*, *13*(3–4), 213–244. https://doi.org/10/gjmwhb

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. Springer New York. https://doi.org/10.1007/978-0-387-84858-7

IDSSP Curriculum Team. (2019). *Curriculum frameworks for Introductory Data Science*. http://idssp.org/files/IDSSP_Frameworks_1.0.pdf

Long, D., & Magerko, B. (2020). What is AI literacy? Competencies and design considerations. In R. Bernhaupt, F. "Floyd" Mueller, D. Verweij, J. Andres, J. McGrenere, A. Cockburn, I. Avellino, A. Goguey, P. Bjørn, S. (Shen) Zhao, B. P. Samson, & R. Kocielnik (Eds.), *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–16). ACM. https://doi.org/10/ghbz2q

McNamara, A. (2019). Key attributes of a modern statistical computing tool. *The American Statistician*, *73*(4), 375–384. https://doi.org/10/ghnfwp

Philip, T. M., Schuler-Brown, S., & Way, W. (2013). A framework for learning about big data with mobile technologies for democratic participation: Possibilities, limitations, and unanticipated obstacles. *Technology, Knowledge and Learning*, *18*(3), 103–120. https://doi.org/10/gmvbkw

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers.

Register, Y., & Ko, A. J. (2020). Learning machine learning with personal data helps stakeholders ground advocacy arguments in model mechanics. *Proceedings of the 2020 ACM Conference on International Computing Education Research*, 67–78. https://doi.org/10/ghnpbc

Ridgway, J. (2016). Implications of the data revolution for statistics education: The data revolution and statistics education. *International Statistical Review*, *84*(3), 528–549. https://doi.org/10/f3q6f6

Ridgway, J., Ridgway, R., & Nicholson, J. (2018). Data science for all: A stroll in the foothills. In *M. A. Sorto, A. White, & L. Guyot (Eds.), Looking back, looking forward: Proceedings of the Tenth International Conference on Teaching Statistics (ICOTS10 in, July, 2018), Kyoto, Japan* (pp. 1–6). International Statistical Institute.

Rule, A., Tabard, A., & Hollan, J. D. (2018). Exploration and explanation in computational notebooks. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–12. https://doi.org/10/gfw4vk

Shearer, C. (2000). The CRISP-DM model: The new blueprint for data mining. *Journal of Data Warehousing*, *5*(4), 13–22.

Sulmont, E., Patitsas, E., & Cooperstock, J. R. (2019a). Can you teach me to machine learn? *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 948–954. https://doi.org/10/ghnpcs

Sulmont, E., Patitsas, E., & Cooperstock, J. R. (2019b). What is hard about teaching machine learning to non-majors? Insights from classifying instructors' learning goals. *ACM Transactions on Computing Education*, *19*(4), 1–16. https://doi.org/10/ghnpbm

Toomey, D. (2017). *Jupyter for data science: Exploratory analysis, statistical modeling, machine learning, and data visualization with Jupyter*. Packt Publishing.

van Gog, T., Paas, F., & van Merriënboer, J. J. G. (2006). Effects of process-oriented worked examples on troubleshooting transfer performance. *Learning and Instruction*, *16*(2), 154–164. https://doi.org/10/chqf28

van Gog, T., Paas, F., & van Merriënboer, J. J. G. (2008). Effects of studying sequences of process-oriented and product-oriented worked examples on troubleshooting transfer efficiency. *Learning and Instruction*, *18*(3), 211–222. https://doi.org/10/brmdfx

Wylie, R., & Chi, M. T. H. (2014). The self-explanation principle in multimedia learning. In R. Mayer (Ed.), *The Cambridge handbook of multimedia learning* (pp. 413–432). Cambridge University Press. https://doi.org/10.1017/CBO9781139547369.021

Zieffler, A., Justice, N., delMas, R., & Huberty, M. D. (2021). The use of algorithmic models to develop secondary teachers' understanding of the statistical modeling process. *Journal of Statistics and Data Science Education*, *29*(1), 131–147. https://doi.org/10.1080/26939169.2021.1900759

A. APPENDIX

In this appendix we briefly explain the relationship of decision trees to Machine Learning (ML), and locate them in the larger area of ML. Some passages are based on explanations in Biehler and Fleischer (2021). ML consists of various fields, each containing different methods that solve similar tasks. These ML domains are called supervised learning, unsupervised learning, and reinforcement learning. Unsupervised learning aims to find patterns in data; supervised learning aims to explain given patterns in data and make predictions based on that. Reinforcement learning aims to enable artificial agents (e., g., a robot, a player of a game, etc.) to act based on experience represented as data.

In supervised learning, one specific task is classification. Classification means that objects described by a set of characteristics are assigned to a class out of different classes. In other words, for an object, the values of a group of predictor variables (characteristics) are known. Based on this, the value of a categorical target variable (class) is to be predicted by a so-called classifier. The classifier is a rule system (e.g., a mathematical function, a decision tree, etc.) that serves as a prediction model and provides an individual output as a prediction of a class for each possible combination of values for the predictor variables.

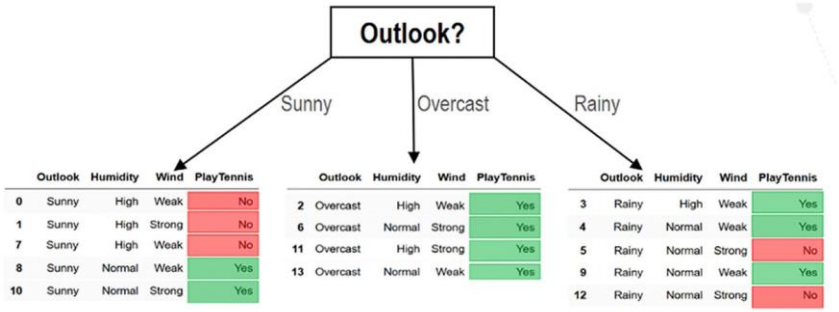
Decision trees are classification (or regression) models that predict a target variable from other variables (e.g., predicting the presence disease from diagnostic features). We can display a decision tree as a literal, hierarchical tree. Using a tree structure to make a decision is highly transparent and understandable, especially when the tree is not too large. As with other model types used in ML, such as artificial neural networks, the application is embedded in a context of predictive modeling: a first model is constructed by training it using training data (fitting it to data in more statistical terms) and then checking it on test data to adjust for overfitting and enhance the generality of the model. Different criteria are used to evaluate the quality of the final predictive model, for example, by calculating misclassification rates. The basic idea is to predict future data from past data, assuming that the underlying system is stable enough to make valid predictions. Automating the model creation based on data is a crucial feature of ML. Creating a model in statistics usually means that human is essential in setting up a statistical model. Humans are still necessary in ML but at a different place. They have to supervise the process, provide valid and useful data, and evaluate the model's usefulness delivered by the ML algorithm.

In the following, we describe the key aspects of the somewhat heterogeneous field of decision tree algorithms. We illustrate our approach based on a conceptual analysis of the domain of decision trees to show our rationale when designing the teaching module. Creating a decision tree can be divided into two parts - the initial growing process with training data and the downstream process of optimizing the initially grown tree by evaluating it with validation data. We describe these two parts separately in the following two subsections.

A.1. THE INITIAL GROWING PROCESS OF A DECISION TREE BASED ON TRAINING DATA

Before describing an algorithm respectively an approach for automatically creating a decision tree, we will illustrate how to manually create a decision tree based on data using an often used example from Quinlan (1986), which is well suited to conveying an idea of the approach. Let us look at a categorical target variable, "Will John play tennis?" The data set in Figure 7 (left) shows data for 14 days in the past, each with values for different weather variables (outlook, humidity, wind) along with John's decisions to play tennis or not. The objective is to predict whether John will play tennis on a given day in the future (target variable: PlayTennis), taking the weather conditions as predictor variables. The goal is to find combinations of values of the predictor variables that can "predict" the target variable in the past data with only a few misclassifications.

	Outlook	Humidity	Wind	PlayTennis
0	Sunny	High	Weak	No
1	Sunny	High	Strong	No
2	Overcast	High	Weak	Yes
3	Rainy	High	Weak	Yes
4	Rainy	Normal	Weak	Yes
5	Rainy	Normal	Strong	No
6	Overcast	Normal	Strong	Yes
7	Sunny	High	Weak	No
8	Sunny	Normal	Weak	Yes
9	Rainy	Normal	Weak	Yes
10	Sunny	Normal	Strong	Yes
11	Overcast	High	Strong	Yes
12	Rainy	Normal	Strong	No
13	Overcast	Normal	Weak	Yes



	Outlook	Humidity	Wind	PlayTennis
0	Sunny	High	Weak	No
1	Sunny	High	Strong	No
7	Sunny	High	Weak	No
8	Sunny	Normal	Weak	Yes
10	Sunny	Normal	Strong	Yes

	Outlook	Humidity	Wind	PlayTennis
2	Overcast	High	Weak	Yes
6	Overcast	Normal	Strong	Yes
11	Overcast	High	Strong	Yes
13	Overcast	Normal	Weak	Yes

	Outlook	Humidity	Wind	PlayTennis
3	Rainy	High	Weak	Yes
4	Rainy	Normal	Weak	Yes
5	Rainy	Normal	Strong	No
9	Rainy	Normal	Weak	Yes
12	Rainy	Normal	Strong	No

Figure 7. Data (left) and data split (right) for a toy example "Does John play Tennis?"

We have to start with one of the predictor variables. For example, in Figure 7 (right), the predictor variable outlook splits the data into three distinct subsets, in each of which we see a specific distribution of the target variable. As the prediction for a subset, we choose the value of the target variable that has the highest frequency in that subset (majority vote principle), and in the case of a tie, random choice decides. If we apply this to the example in Figure 7, we will predict that John plays on days where it is rainy or overcast and does not play on sunny days. We can already correctly predict 10 out of 14 cases with this decision rule. In the next step, we involve further variables to improve the percentage of correct predictions. We can see in the data that on sunny days, splitting by humidity results in no more misclassifications in this branch. We analogously use the variable "wind" for rainy days to split the data set one step further. If these two data splits are carried out, we get the decision tree in Figure 8, and all training examples are classified correctly. In the above example, we have intuitively—and by careful observation—chosen variables for the data splits and sharp observation to split the original data set into so-called pure subsets. Each only contains one class of objects having the same value for the target variable.

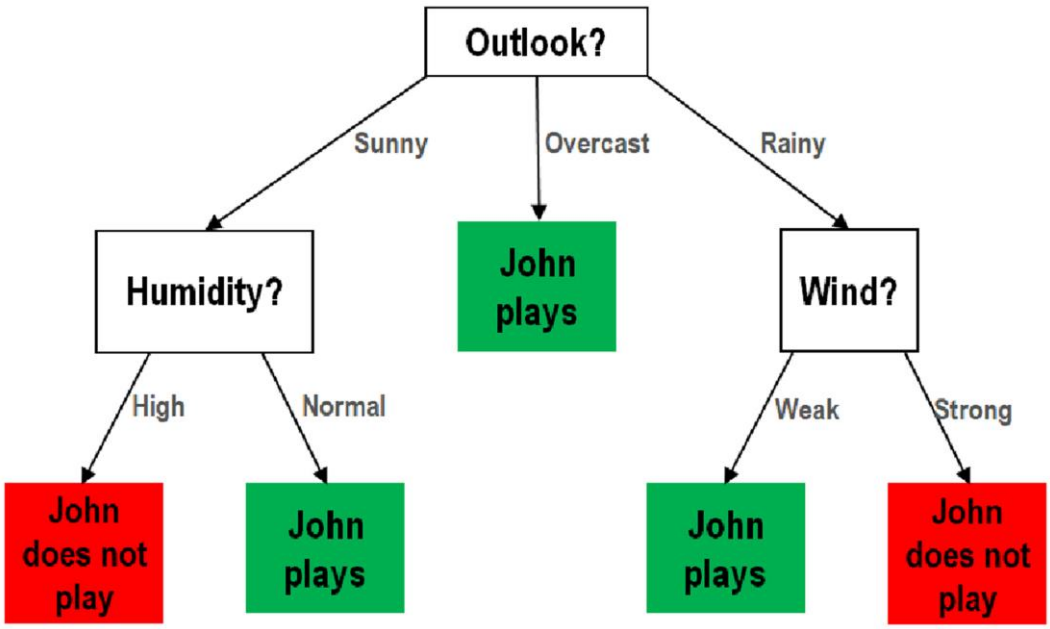


Figure 8. Decision Tree for the toy example "Does John play Tennis?"

Decision tree algorithms create decision trees automatically for given data. Such decision tree algorithms go back to the work of Quinlan, who developed ID3 (Quinlan, 1986) and C4.5 (Quinlan,

1993), or the work of Breiman et al., who created CART (Breiman et al., 1998). These algorithms follow the same recursive greedy approach of building decision trees top-down, aiming to correctly classify all objects in the data. Before choosing a variable and a data split to define a decision rule, two termination criteria are checked:

- **Is another data split necessary?**
- **Is another data split possible?**

Another data split is necessary if the data set is not pure, which means there are objects with different values for the target variable. If the data set is pure, the algorithm terminates at this point. Another data split is possible if a predictor variable is left that can split the data set. If there is no predictor variable to split the data, the algorithm terminates at this point.

If none of the termination criteria is fulfilled, a predictor variable and a data split must be chosen to define a decision rule. Decision tree algorithms are automated by systematically selecting the best-suited variable according to a specific criterion. All predictor variables and possible data splits are taken into account to find the variable and split with the lowest rate of misclassifications (or other measures like entropy, Gini index, gain ratio, etc.). The general steps that have to be carried out recursively for all subsets created during the process are:

- **Define all possible splits** by using all variables
- **Identify the best split** according to a pre-defined criterion
- **Apply the best split** to create subsets

As mentioned above, many different decision tree algorithms (ID3, C4.5, CART, etc.) share these general steps while they differ in detail. They differ in the exact definition of possible splits; for example, CART uses only binary splits while C4.5 allows wider splits. The measure used to identify the best split (misclassification rate, entropy, Gini index, etc.) differs between algorithms. The misclassification rate is the most intuitive measure for assessing and comparing the predictive quality of different data splits. Other less intuitive measures (e.g., entropy, Gini index, gain ratio, etc.) are widely used because, in practice, they have slight advantages over the misclassification rate (Hastie et al., 2009, pp. 309–310). We recommend starting with the misclassification rate at the secondary level because the less intuitive measures might hinder students' understanding of the overall approach. Applying a split also differs between algorithms - for example, with regard to the handling of missing values: In some algorithms, cases with missing values are discarded during the split, while other algorithms attempt to predict missing values to use all cases throughout the process. After the best split is identified and applied, different branches with different subsets of the original data set result. Then, the procedure is repeated recursively regarding all subsets of the data set created in the previous step until a termination criterion (pure subset reached or no unused variable left) is reached.

A.2. THE EVALUATION AND OPTIMIZATION OF A DECISION TREE BASED ON VALIDATION DATA

The primary assumption that a model which correctly predicts past data (represented by training data) will also correctly predict future data (represented by test data) to the same degree is not self-evident, nor is it usually the case. In practice, the model of Figure 8 would be adjusted against overfitting by using validation data (intermediate test data to optimize the model) of John's tennis-playing records. Then the final predictive model would be evaluated using test data. As the example in Figure 8 is fictitious and we have no test data, we leave that example behind now. Typically, an initially grown tree (as described above) does not perform as well for test data as for training data. The approach of seeking pure subsets at any cost to perfectly classify the training data leads to so-called overfitting to the training data. That causes the overall misclassification rate for test data to be higher than for training data. This can be revealed by using validation data for evaluating and optimizing the initially grown tree. For instance, Figure 9 shows the evolution of the rates of correct classifications for training and test data during the automated training process for one of our examples in the teaching module. On the x-axis, the depth of the tree is indicated, and on the y-axis, the rate of correct classifications is shown. The depth of a tree is the number of nodes in the most extended branch. The tree on which this illustration is based has a depth of 11. The final rate of correct classifications is about 97% for training

data and about 69% for test data. The other indicated rates of correct classifications for lower depths of the tree are calculated with the intermediate conditions of the tree during the training process.

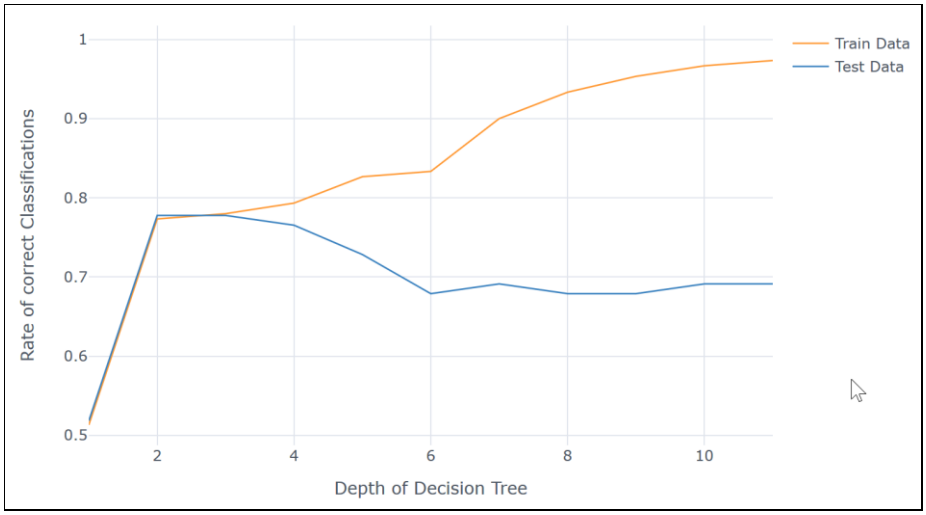


Figure 9. Evolution of the rate of correct classifications during the training process of a decision tree

The evolution of the successful classification rates illustrates overfitting to the training data: the rate of correct classifications increases steadily for training data as the tree gets deeper, but the rate of correct classifications for test data tends to decrease from a certain point onwards. This typical behavior occurs because the decision rules added at the end are based on only small subsets of the original data set. Therefore, they are less generalizable to test data than the decision rules added first. The approach of so-called pruning is to reduce the depth of the tree to reduce overfitting. The objective is to find the most appropriate intermediate state of the tree. Breiman et al. (1998) suggest cost-complexity pruning (pre-pruning), which reduces the depth (complexity) of a tree by weighing the benefit of a node against the increase in complexity as another termination criterion when building the tree. The post-pruning method suggested by Quinlan (1993, pp. 37–43) prunes all nodes in all branches of the decision tree from the bottom up in a trial and error approach to evaluate the respective effect on the prediction for the validation data. If pruning a branch improves the misclassification rate for the validation data, it is carried out. We suggest the latter method of post-pruning for school teaching because it is intuitive, sufficiently productive, and there is no need to introduce a measure for complexity.

A.3. THE ALGORITHM WE IMPLEMENTED IN PYTREE FOR TEACHING PURPOSE

The algorithm we use for teaching (and in our Python library PyTree) is oriented to the existing algorithms mentioned above, consisting of the termination criteria and the greedy approach with the three steps of defining all possible data splits, identifying the best data split by a certain measure, and applying the best data split. The aim of implementing and using our own algorithm was to have a somewhat competitive algorithm that is preferably simple and suitable for school teaching. Each of the single steps of the algorithm should not be too complex, nor should they afford too much mathematical pre-knowledge. For all algorithm steps, we considered the different approaches that different professional algorithms offer and aimed to choose the optimal solution from a didactical point of view. The overall process of our algorithm is the same as described above.

The definition of all splits includes different approaches for categorical and numerical predictor variables. For categorical predictor variables, only one data split is considered splitting the data set in as many subsets as values are given for the predictor variable. For numerical predictor variables, only binary data splits are considered grouping the object greater than a threshold and less than or equal to the threshold. For numerical variables, different data splits are regarded concerning different thresholds. For ascending sorted values of the predictor variable, a threshold is considered in each interval between two neighboring values.

Identifying the best data split is done by evaluating the resulting misclassification rate for each data split. Optionally students can use entropy, but we do not want to force entropy to be taught because it is a far more complex and less intuitive measure.

The application of a split raises the question of how to deal with missing values. Our algorithm drops the cases with a missing value for a predictor variable used in the decision tree. Some algorithms have complex methods of dealing with these values. Still, we aimed to keep the algorithm rather simple when a possible feature is relatively complex and brings just minor performance improvements.

For pruning, our approach offers simple forms of pre-pruning (e.g., a limit for maximum depth of the tree) or the post-pruning method suggested by Quinlan (1993, pp. 37–43) that prunes all nodes in all branches of the decision tree from the bottom up in a trial and error approach.