Introducing students to machine learning with decision trees using CODAP and Jupyter Notebooks

Rolf Biehler | Yannik Fleischer

Department of Mathematics, Paderborn University, Paderborn, Germany

Correspondence
Rolf Biehler, Paderborn University, Paderborn, Germany.
Email: biehler@math.upb.de

Funding information
Deutsche Telekom Stiftung

Abstract
This paper reports on progress in the development of a teaching module on machine learning with decision trees for secondary-school students, in which students use survey data about media use to predict who plays online games frequently. This context is familiar to students and provides a link between school and everyday experience. In this module, they use CODAP’s “Arbor” plug-in to manually build decision trees and understand how to systematically build trees based on data. Further on, the students use a menu-based environment in a Jupyter Notebook to apply an algorithm that automatically generates decision trees and to evaluate and optimize the performance of these. Students acquire technical and conceptual skills but also reflect on personal and social aspects of the uses of algorithms from machine learning.

KEYWORDS
teaching, data science, data science education, decision trees, Jupyter Notebooks, machine learning

1 | INTRODUCTION

Automated decision-making processes based on machine learning methods are relevant in many societal applications. Applications such as personalized advertisements on online platforms, diagnoses in healthcare, evaluation of legal issues [5], or even support for election campaigns [13] are implemented by data-driven decision models that are based on machine learning. The social discourse around automated decision-making is often characterized by a euphoric exaggeration of the performance and value on the one hand and anxious rejection of so-called “franken-algorithms” [18] on the other. With increasing societal relevance, there is a growing demand for data-driven procedures to be taken up in school education [3,6,7,16,17]. Students’ everyday experiences include automated decision models (eg, personalized advertising) that they encounter as black boxes. We created a teaching module for the upper secondary level that aims at making the machine learning method of decision trees transparent to students. The teaching module was created in the ProDaBi project (www.prodabi.de), an interdisciplinary collaboration between researchers in mathematics education and computer science education at the Paderborn University. The project began by hosting an international symposium, based on which a theoretical foundation for curricular goals and contents, such as teaching decision trees, was developed [2]. Supervised learning, and in particular decision trees, also occupies a significant place in the internationally respected IDSSP curriculum for data science in schools [12]. However, teaching it still is a challenge. As Sulmont et al [20] found...
for non-majors at the university level, it is more difficult to teach reasoning about machine learning models than it is to teach the algorithms, yet it is possible [19]. Regarding teacher education, Zieffler et al [22] recently noted that, analogously for mathematics teachers, decision tree algorithms can be taught well in in-service training, but the evaluation of decision trees is a greater challenge. In our module, we focus on both decision tree algorithms and their evaluation. We consider this as necessary for a well-founded reflection on the opportunities and limitations of automated decision models. Practical testing and evaluation take place in a so-called “project course” (a non-compulsory course in grade 12), which has been run in cooperation with two secondary schools in Paderborn for the third time in the school year 2020/21. It consists of a system of modules that are related but which can also be used in a stand-alone manner with some adaptation. We present such a stand-alone module on the machine learning method of decision trees.

2 | DECISION TREES IN THE CONTEXT OF MACHINE LEARNING

Decision trees are classification or regression models that predict a target variable from other variables (e.g., predicting 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 machine learning, such as artificial neural networks, the application is embedded in a context of predictive modeling: a first model is constructed by training it via 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 such predictions valid. The automation of the model creation is a key feature of machine learning, whereas creating a model in statistics usually means that the human is essential in setting up a statistical model. Humans are still essential in machine learning but at a different place. They have to supervise the process in terms of providing valid and useful data and evaluating the usefulness of the model delivered by the machine learning 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 [10] to show our rationale when designing the teaching module. As mentioned above, the process of 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 test data. We describe these two parts separately in the following two subsections.

2.1 | The initial growing process of a decision tree based on training data

The decision tree algorithms go back to the work of Quinlan who developed ID3 [14] and C4.5 [15], or the work of Breiman et al who developed CART [4]. We will illustrate how to manually create a decision tree with an example from Quinlan [14], 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 1 (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.

We have to start with one of the predictor variables. For example, in Figure 1 (right), the predictor variable Outlook splits the data into three distinct subsets, in each of which we see the distribution of the target variable. As the prediction for a subset, we choose the value of the predictor 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 1, we will predict that John plays on days where it is rainy or overcast and does not play on sunny days. With this decision rule, we can already correctly predict 10 out of 14 cases. 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 according to humidity will lead to no further misclassifications, and for the rainy days, we use the variable “wind” to split the data set one step further. If that is done to this example, we get the decision tree in Figure 2, and all training examples are classified correctly—which, however, is unusual in practice. In the above example, we have chosen the first variable for the data split intuitively. In the decision tree algorithms, this process is automated. Choosing the first branch of the
tree is done by looking at each predictor variable and all possible data splits in turn and choosing the variable and split with the lowest rate of misclassifications (or other criteria that define “information gain”). Then, the procedure is repeated recursively regarding all subsets with non-pure frequency distributions. This algorithm terminates when pure subsets are achieved or no unused variables for further splitting exist.

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** in terms of predictive quality
- **Apply the best split** to create subsets

These steps are conducted in every branch for every subset of the data set that was created in the previous step set until a termination criterion (pure subset reached or no unused variable left) is reached.

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 also allows wider splits. The measure used to identify the best split (misclassification rate, entropy, Gini index, etc.) also 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 [11, 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, in terms of handling missing values: In some algorithms, cases with missing values are simply discarded during the split, while other algorithms attempt to predict missing values to use all cases throughout the process.

### 2.2 The evaluation and optimization of a decision tree based on test 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 2 would be adjusted against overfitting by using a further data set of John’s tennis playing records, and then the final predictive model would be evaluated using different evaluation criteria, for instance, the overall misclassification rate (which is 0 for training data in this example). As the example in Figure 2 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 it does 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 test data for evaluating and optimizing the initially grown tree. For instance, Figure 3 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. The training data are used in this training and all intermediate steps are also applied to the test data. On the x-axis, the depth of the tree is indicated, and on the y-axis, the rate of correct classifications is indicated. The depth of a tree is the number of nodes in the longest branch. The tree on which this illustration is based has a depth of 10. The final rate of correct classifications is about 97% for training data and about 73% 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.

The evolution of the successful classification rates illustrates overfitting to the training data; the rate of success increases steadily with training data as the tree gets deeper, while the rate of success with the 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 the overfitting. The objective is to find the most appropriate intermediate state of the tree. Breiman et al [4] 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 [15, 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 test data. If pruning a branch improves the misclassification rate for the test 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.

After these optimization steps, the final decision tree is to be evaluated with another set of new test data. In addition to misclassification rates, this more detailed evaluation includes (in case of a binary target variable) the examination of the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), which are typically displayed in a so-called confusion matrix. Further relevant statistical measures based on these four are sensitivity and precision. In evaluating the performance of a tree, one may wish to have these distinctions because the different errors may have very different consequences and costs (eg, a false positive test for Covid is less problematic than a false negative test in terms of likely testee behavior).

3 COMPONENTS OF THE BASIC TEACHING UNIT FOR DECISION TREES

In this section, we describe the essential components of the introductory teaching unit: the example data, how the Arbor plug-in works in the web-based free data exploration tool CODAP (codap.concord.org), and how the Jupyter Notebook we designed works. In Section 5, we will describe how these components were embedded in a teaching sequence.

3.1 The example data

In this introductory unit, students explore a dataset about how adolescents use media. We collected this data (n = 215, 94 variables) using a questionnaire (adding some more questions) from the annually conducted JIM study (Jugend [Youth]-Information-Media, [1]), in which only highly aggregated data are published. With our questionnaire, we have collected our own non-aggregated data. We will refer to this dataset as JIM-Paderborn or JIM-PB. In addition to demographics such as gender, JIM-PB asks adolescents how they engage with digital media using questions to ask, for example, how often they play online games (Playing_OnlineGames), or Use_Instagram, whether they own a computer.

![FIGURE 3 Evolution of the rate of correct classifications during the training process of a decision tree [Colour figure can be viewed at wileyonlinelibrary.com]](image-url)
(Own_Computer), how many apps are on their smartphones (Number_of_Apps), and many others. This complete data set can be used in exploratory data analyses before starting with decision trees. We use it also in the second part of the decision tree module, when using Jupyter Notebooks. For the first part with CODAP, we made some strategic modifications and used a reduced form (Figure 4):

- We included only 53 cases out of 215 to constitute the training set students are to work with. With this smaller data set, students can build a tree that perfectly classifies the training data (target variable: Playing_OnlineGames)
- We reduced the data from the original 94 variables to 15 variables to make things easier for students - and increase the usability of CODAP. Some variables, such as Own_Computer, are plausible predictors; others, such as Using_Twitter seem not to be
- Many responses are collected on Likert scales about the frequency of use, with seven levels ranging from “never” to “daily”; these we recoded to two values: “rarely” or “frequently.” Although the Arbor plug-in lets students assign specific values to different tree branches, it’s much simpler for beginners if the variables are already binary.

The reduction of cases from 215 to 53 was a pedagogical decision because we think that it is desirable to enable the students to manually create a tree that classifies the training data “perfectly” with reasonable effort. We want to use the experiences from this phase later to motivate a discussion about overfitting. Testing such a “perfect” tree and finding that it might perform worse for test data is vital for motivating a discussion about overfitting the training data and pruning.

Furthermore, when intuitively creating a decision tree, the students can, at first, select a predictor variable based on their context and data knowledge concerning which variables may be influential. However, this approach may become difficult for variable and split selection in later stages. Students might just follow trial and error for creating a “perfect” tree and include non-plausible variables. We want to use this experience later for introducing automated methods.

### 3.2 The digital tool CODAP with the decision tree plug-in Arbor

In the first part of the module, the students are introduced to the web-based data analysis tool CODAP [9]. CODAP includes Arbor, an interactive decision tree plug-in [8] shown in Figure 5. With this tool, it is possible to create a decision tree for a data set manually; the numbers of correct and incorrect classifications are calculated and displayed by Arbor. The illustration in Figure 5 is based on the reduced JIM-PB data set illustrated in Figure 4. The 15 variables are shown at the bottom. The selected target variable is “Playing_OnlineGames” with the values “frequently” and “rarely”. The user can choose predictor variables from the bottom section and introduce them into the tree via drag and drop. The tool automatically splits the data set using the predictor variable and evaluates the resulting subsets.

In Figure 5 (left), the predictor variable “gender” is selected. The illustration shows that out of 28 male persons, 24 frequently play online games. Out of 25 female persons, only eight play frequently. The user can determine a prediction for every branch. The decision trees shown in Figure 5 were set to classify the cases according to the majority in the corresponding subsets of the training data (majority vote principle). Therefore, the tree in Figure 5 left predicts that male persons play online games frequently, and female persons play rarely. Overall, 12 cases are misclassified. To improve the tree, the user can add additional predictors at the end of the branches (see Figure 5, right). That way, a decision tree grows gradually. The user can evaluate the accuracy of the tree by examining the numbers of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), which are shown at the bottom of the display. Unlike other decision tree tools, CODAP does not automatically calculate an overall misclassification rate but distinguishes false positives and false negatives. As an
additional feature, the different values (TP, TN, FP, FN) can optionally be displayed as a confusion matrix. For further exploration, we recommend the reader to visit the prepared environment with the JIM_PB data and the tree plug-in (codap.concord.org/releases/latest/static/dg/en/cert/index.html#shared=158650).

3.3 | The digital tool Jupyter Notebook prepared to apply a decision tree algorithm

Jupyter Notebook [21] is a cell-based programming environment which offers the possibility to write and execute code, visualize the output of the code just below, include hyperlinks, and take notes and incorporate pictures in between. Python code can be executed in this environment, and different Python packages (pandas, plotly, numpy, etc.) can be imported to allow access to powerful data analysis and machine learning tools.

A special benefit is that Python code can be hidden behind interactive widgets so that students can work with powerful Python tools in a menu-based environment without even seeing the code. We designed such a menu-based notebook to apply a decision tree algorithm and automatically create and optimize decision trees to ease access for students. In this notebook, we use a Python package that we especially developed for teaching. It creates decision trees automatically by following the approach described in Section 2. Our package was developed with Python and has several didactical benefits in comparison with other Python packages for decision trees (eg, sci-kit learn). It allows students to create trees automatically, but also manual editing/pruning and automated pruning are possible. Decision trees are visualized in a simple but meaningful way (see Figure 6). In every node the frequency distribution of the respective sub-dataset with regard to the target variable is indicated.

Our Notebook, which we will refer to as the ProDaBi Decision Tree Jupyter Notebook offers different options to evaluate and prune the resulting trees with test data, provides meaningful visualizations (eg, Figure 3, from which it is plausible to work with a tree of depth 4), and calculates different statistical measures. Figure 6 shows the result of an automated initial growing process (left) and the final result after automated post-pruning of the initial tree (right).

4 | STRUCTURE OF THE TEACHING MODULE FOR DECISION TREES

Our module on decision trees consists of two parts. In the first part, the students use the digital tool CODAP with the Arbor decision tree plug-in to create and change decision trees manually via drag and drop. The main goal for students in this part is to understand how a decision tree is used and interpreted, and to discuss a systematic approach to showing how an algorithm can create decision trees automatically. In the second part, the students leave CODAP behind and begin working in a menu-based environment in a Jupyter Notebook (Toomey, 2017). This includes applying an algorithm that automatically creates a decision tree based on data, evaluating the resulting tree with test data - data that was not available as the tree was being built - and optimizing the performance of the tree by systematically pruning branches.

The overall task is as follows: students are to take the perspective of owners of an online platform that collects data on the media behavior of their users, such as the behaviors assessed in the JIM-PB data set. The data are to be used to identify targets (ie, platform users) for...
advertising a new online game. This task motivates the creation of a decision tree to predict how often people are playing (“frequently” or “rarely”). The context is somewhat contrived because online platforms usually do not use survey data for such purposes. It is realistic in that they do have access to similar data that is collected by different means. This point can be discussed with the students at the beginning. This task context makes a connection to students’ everyday experiences (e.g., personalized advertisements, and social media news feeds).

4.1 Phases of the teaching module

The teaching unit consists of different phases:

| Phase | No. of lessons | Topic |
|-------|----------------|-------|
| 1     | 1              | An introductory example of a decision tree and the concept of data split |
| 2     | 2              | Exploratory creation of decision trees by hand with CODAP |
| 3     | 1              | Systematizing the process of creating decision trees and the development of formal criteria to understand a decision tree algorithm |

(Continues)

In phase 1, the students are introduced to decision trees with the small data example “Does John play Tennis?” (Figure 1) to understand the structure of a decision tree diagram and the concept of data split. The introduction is an interactive presentation consisting of different steps.

In phase 2, the students deal with the reduced JIM-PB data. This phase aims to develop competencies using the CODAP tree tool and get familiar with the growing process of a decision tree and the concept of misclassification. The students create a multi-level decision tree for the first time without detailed instructions from the teacher, who only gives technical instructions on how to operate the
tool. Specifically, there is no instruction on how to select predictor variables or when to stop the growing process. The students are given the task of creating a decision tree by exploring and using intuitive criteria until they are satisfied with the result (termination criteria are not specified). Some students may stop the growing process early, some may just try variables at random, and some may even create and use formal criteria. All these approaches are legitimate in this context.

In phase 3, the central goal is to systematize the students’ earlier activities and to transform them into an algorithmic procedure that can be performed by a machine that has no context knowledge. One point is to understand how to systematically assess and evaluate different data splits by using the misclassification rate. At the end of this phase, the teacher guides a discussion to formulate an algorithm as pseudocode that builds on the students’ experiences and includes all essential steps.

In phase 4, the students leave CODAP behind, apply the decision tree algorithm within our Jupyter Notebook, and evaluate the initially grown tree with test data. At this point, the concept of overfitting is discussed in class and explained based on students’ experiences in the previous phases. While trying to find a “perfect” tree in the first place, from a certain point on, the students had the experience of doing arbitrary splits with no reason in terms of context. This experience does not deliver the technical explanation of overfitting, but it makes it plausible why the undesirable overfitting occurs. Afterward, the students use our Jupyter Notebook to manually prune the tree and systematize the process of post-pruning.

In phase 5, first, different examples of prediction models in different contexts are examined with regard to possible types of errors (FP vs FN) and the costs they incur. Afterwards, the students grow and prune trees automatically in Jupyter Notebook, and then they are asked to evaluate the results with different measures that are based on the confusion matrix. They compare different types of mistakes concerning the task context and examine the sensitivity and precision of the decision tree. Furthermore, they interpret the decision tree content-wise and check whether the automatically chosen decision rules are plausible.

In the final phase 6, the differences between humans and machines in decision making are discussed. Humans use their knowledge about the world and about the previously explored data set, and further plausible considerations to find well-performing predictor variables without being able to test many variables. Machines lack knowledge about the world but can test and evaluate many data splits using formal criteria in a short time. Another discussion point is the use of automated decision-making in different contexts (with different costs of errors). A single medical decision might be more impactful than a single decision about advertising, while on a large scale, targeted advertising may be very influential, for example, in election campaigns. Another point that can be addressed is the impact of effective but undesirable decision rules. For instance, in Figure 6, the first second is based on gender; this can be problematic in many contexts.

5 | EXAMPLES OF STUDENTS’ WORK

We show two examples of the results of student work from phase 2. We collected the trees that were built without instruction and using students’ intuitive approaches as screenshots. Two examples (in German) are shown in Figure 7. The values “Häufig” and “Selten” mean frequently and rarely respectively. We also recorded how these two students presented their trees and how they described their approach to choosing predictor variables and deciding when to stop their growing process.

*FIGURE 7* Students’ solutions creating a decision tree with CODAP [Colour figure can be viewed at wileyonlinelibrary.com]
The left tree includes far more variables than the right tree, yet performs only slightly better than the right tree (7 misclassifications vs 8 misclassifications). In the presentation, the student who created the left tree argued for the plausibility of the different variables and explained contextually why the different variables might be appropriate to predict the target variable. The student first selected the plausible variables by intuitive reasoning and then tried different combinations. The student that created the right tree argued in a different way. He tested various variables via “trial and error” as the top decision rule and searched for subsets where the relative frequency of the target value is “near 100% or near 0%” to evaluate which variables might be useful as predictors of the target variable. Furthermore, he stated that he stopped the growing process very early so that the final subsets remain “representative.” In summary, the student of the left tree decided intuitively like humans usually do, while the student of the right behaved more “like a machine” without contextual knowledge and anticipated the decision tree algorithm that was later developed in step 3 and 4.

6 DISCUSSION

We described a basic teaching unit about machine learning and decision trees, based on CODAP in the first part and on our ProDaBi Decision Tree Jupyter Notebook in the second part. This teaching module can be used as a stand-alone module to get essential insights into the concept of a decision tree and the automated creation of decision trees from data. The combination of these two tools in one module was beneficial in different ways. CODAP is used for an easy introduction to decision trees via an explorative approach. As described in Section 5, the work with CODAP brings up different approaches to creating decision trees that provide an occasion to discuss the most effective approach, and to help understand how a machine can automatically create decision trees. After grasping the concepts of how a machine creates trees, it seems beneficial to use Jupyter Notebooks to create trees semi-automatically. Other aspects such as overfitting and pruning can now become the teaching focus. The core of machine learning is automation. It seems to motivate students to see decision trees pop up in seconds after previously spending hours creating them manually. Both tools complement each other very well by compensating for their respective disadvantages.

ACKNOWLEDGEMENTS

This paper emerged as part of the ProDaBi project, which was initiated and is funded by the Deutsche Telekom Stiftung. We are grateful for the continuous support of the whole ProDaBi team during the development of our teaching unit. Our thanks go to Jim Ridgway and Tim Erickson who intensively commented on an earlier version and made many suggestions for considerably improving the language, the structure and the clearness of our paper. Open Access funding enabled and organized by Projekt DEAL. WOA Institution: UNIVERSITAET PADERBORN Blended DEAL: Projekt DEAL.

REFERENCES

1. P. Behrens and T. Rathgeb, JIM-Studie 2017 - Jugend, Information, (Multi-)Media, Basisstudie zum Medienumgang 12- bis 19-jähriger in Deutschland, Medienpädagogischer Forschungsverbund Südwest, Stuttgart, 2017.
2. R. Biehler, L. Budde, D. Frischemeier, B. Heinemann, S. Podworny, C. Schulte, and T. Wassong, Eds., Paderborn Symposium on Data Science Education at School Level 2017: The Collected Extended Abstracts, Paderborn, Universitätsbibliothek Paderborn, 2018. https://doi.org/10.17619/UNIPB/1-374.
3. R. Biehler and C. Schulte, Perspectives for an interdisciplinary data science curriculum at German secondary schools, in Paderborn Symposium on Data Science Education at School Level 2017: The Collected Extended Abstracts, R. Biehler, L. Budde, D. Frischemeier, B. Heinemann, S. Podworny, C. Schulte, et al., Eds., Universitätsbibliothek Paderborn, Paderborn, 2018, 2–14.
4. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification and regression trees. Repr, Chapman & Hall, Boca Raton, 1998.
5. J. Dressel and H. Farid, The accuracy, fairness, and limits of predicting recidivism, Sci Adv 4(1) (2018), eaao5580.
6. J. Engel, Statistical literacy for active citizenship: A call for data science education, Stat Educ Res J 16(1) (2017), 44–49.
7. J. Engel, R. Biehler, D. Frischemeier, S. Podworny, A. Schiller, and L. Martignon, Zivilstatistik: Konzept einer neuen Perspektive auf Data Literacy und Statistical Literacy, ASTA Wirtschafts- und Sozialstatistisches Archiv. 13(3-4) (2019), 213–244. http://dx.doi.org/10.1007/s11943-019-00260-w.
8. J. Engel, T. Erickson, and L. Martignon, Teaching and learning about tree-based methods for exploratory data analysis, in Looking back, looking forward. Proceedings of the Tenth International Conference on Teaching Statistics (ICOTS10, July, 2018), Kyoto, Japan, M. A. Sorto, A. White, and L. Guyot, Eds., Internat. Statistical Institute, Voorburg, The Netherlands, 2018.
9. W. Finzer, CODAP: Common online data analysis platform, 2017, available at codap.concord.org.
10. Y. Fleischer, Elementarisation and didactical analysis of the topic “decision trees in machine learning” for implementation in mathematics classroom, Master Thesis, Paderborn: Universität Paderborn Institut für Mathematik, 2019.
11. T. Hastie, R. Tibshirani, and J. H. Friedman, The elements of statistical learning data mining, inference, and prediction, 2nd ed., Springer, New York, 2009.
12. IDSSP Curriculum Team, Curriculum frameworks for introductory data science, 2019, available at http://www.idssp.org/files/IDSSP_Data_Science_Curriculum_Frameworks_for_Schools_Edition_1.0.pdf.

13. S. Issenberg, How President Obama’s campaign used big data to rally individual voters, MIT Technol Rev 116(1) (2012), 38–49.

14. J. R. Quinlan, Induction of decision trees, Mach Learn 1(1986), 81–106.

15. J. R. Quinlan, C4.5: Programs for machine learning, Morgan Kaufmann Publishers Inc, San Francisco, 1993.

16. J. Ridgway, Implications of the data revolution for statistics education, Int Stat Rev 84(3) (2016), 528–549.

17. J. Ridgway, R. Ridgway, and J. Nicholson, Data science for all: A stroll in the foothills, in M. A. Sorto, A. White, L. Guyot, Looking back, looking forward: Proceedings of the Tenth International Conference on Teaching Statistics (ICOTS10 in, July, 2018), Kyoto, Japan, Voorburg, The Netherlands, 2018, 1–6.

18. A. Smith, Franken-algorithms: The deadly consequences of unpredictable code, The Guardian (August 30 2018), available at https://www.theguardian.com/technology/2018/aug/29/coding-algorithms-frankenalgos-program-danger.

19. E. Sulmont, E. Patitsas, and J. R. Cooperstock, Can you teach me to machine learn? in Proceedings of the 50th ACM Technical Symposium on Computer Science Education, E. K. Hawthorne, M. A. Pérez-Quiñones, S. Heckman, and J. Zhang, Eds., ACM, New York, 2019, 948–954.

20. E. Sulmont, E. Patitsas, and J. R. Cooperstock, What is hard about teaching machine learning to non-majors? Insights from classifying instructors’ learning goals, ACM Trans. Comput. Educ. 19(4) (2019), 1–16.

21. D. Toomey, Jupyter for Data Science - Exploratory analysis, statistical modeling, machine learning, and data visualization with Jupyter, Packt Publishing, Birmingham, 2017.

22. A. Zieffler, N. Justice, R. del Masa, and M. D. Huberty, The use of algorithmic models to develop secondary teachers’ understanding of the statistical modeling process, J Stat Data Sci Educ 1 (2021), 131–147.

How to cite this article: R. Biehler and Y. Fleischer, Introducing students to machine learning with decision trees using CODAP and Jupyter Notebooks, Teaching Statistics 43 (2021), S133–S142. https://doi.org/10.1111/test.12279