UNIVERSIDAD DE BURGOS
Escuela Politécnica Superior
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Decision Tree Simulator

Presentado por Daniel Drefs Fernandes
en Universidad de Burgos — June 12, 2024
Tutores: Carlos López Nozal
Ismael Ramos Pérez

UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

D. Carlos López Nozal y Ismael Ramos Pérez, profesores del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Exponen:

Que el alumno D. Daniel Drefs Fernandes, con DNI L1TLL31X4, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Decision Tree Simulator.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de los que suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, June 12, 2024

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

El objetivo de este proyecto ha sido crear una aplicación web que ayude en el aprendizaje del concepto de árboles de decisión, su creación y todos los requisitos previos que son necesarios para entender el tema, como la entropía y la entropía condicional.

Las calculadoras de entropía y entropía condicional se manejan mediante valores de entrada elegidos por el usuario.

El algoritmo de árbol de decisión que se enseña a través de esta aplicación es el algoritmo ID3. Se proporciona una visualización paso a paso para garantizar una experiencia de aprendizaje sencilla e intuitiva. La visualización incluye un árbol de decisión creado dinámicamente, mostrado con Scalable Vector Graphics (SVG), y dos tablas que representan el conjunto de datos y los valores relevantes en cada paso, respectivamente.

Este proyecto se ha desplegado a través del servicio GitHub Pages en la siguiente URL: `https://danieldf01.github.io`.

Descriptores

Árboles de decisión, entropía, entropía condicional, algoritmo ID3, aprendizaje automático.

Abstract

The aim of this project has been to create a web application that helps in learning the concept of decision trees, their creation and all the prerequisites that are necessary to understand the topic, like the entropy and conditional entropy.

The entropy and conditional entropy calculators are driven by user-chosen input values.

The decision tree algorithm that is taught through this application is the ID3 algorithm. A step-by-step visualization is provided to ensure a simple and intuitive learning experience. The visualization includes a dynamically created decision tree, displayed with Scalable Vector Graphics (SVG), and two tables that represent the dataset and relevant values at each step, respectively.

This project has been deployed through the GitHub Pages service at the following URL: `https://danieldf01.github.io`.

Keywords

Decision trees, entropy, conditional entropy, ID3 algorithm, Machine Learning.

Contents

List of Figures

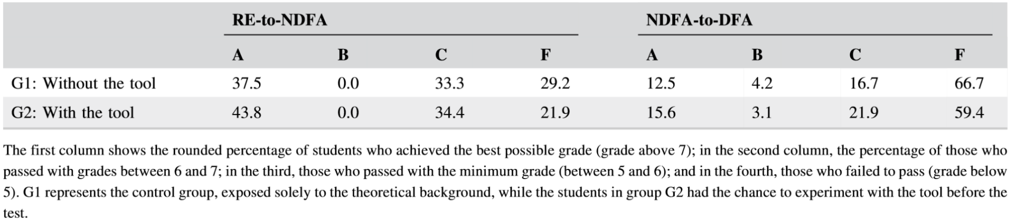
1. Introduction

Decision trees belong to the most popular models in machine learning and data mining, serving as intuitive models for decision-making. The algorithm known as ID3, or Iterative Dichotomiser 3, recursively partitions data based on attribute values with the aim of maximizing information gain at each step. It is one of the most popular decision tree algorithms and has significantly contributed to their widespread use.

This project has drawn great inspiration from the "Seshat Tool"[1]. Its aim is to facilitate users' learning experiences about a concept through intuitive step-by-step simulations of certain algorithms. In the case of Seshat Tool, the target concepts were lexical analysis algorithms. While the main influences are its layout and overall functionality of the presented web application, the related article [5] also carried out a study on its actual effectiveness.

Figure 1.1 shows a table out of the article that presents results of a questionnaire that students were asked to do. They took the questionnaire after they "were taught the main basic concepts of automata and regular expressions, as well as" [5] two algorithms that are also taught in the web application. The students were split into two groups, one that would take the test only based on the knowledge gained by a prior theoretical lecture and another that would additionally be granted access to the Seshat Tool.

[1] Seshat Tool: http://cgosorio.es/Seshat/



	RE-to-NDFA				NDFA-to-DFA			
	A	**B**	**C**	**F**	**A**	**B**	**C**	**F**
G1: Without the tool	37.5	0.0	33.3	29.2	12.5	4.2	16.7	66.7
G2: With the tool	43.8	0.0	34.4	21.9	15.6	3.1	21.9	59.4

The first column shows the rounded percentage of students who achieved the best possible grade (grade above 7); in the second column, the percentage of those who passed with grades between 6 and 7; in the third, those who passed with the minimum grade (between 5 and 6); and in the fourth, those who failed to pass (grade below 5). G1 represents the control group, exposed solely to the theoretical background, while the students in group G2 had the chance to experiment with the tool before the test.

Figure 1.1: Results of students' tests of the Seshat tool [5]

It can be observed that the group that had access to the tool achieved noticeably better results overall than the other group.

Following the success of the Seshat Tool, the aim of this project has been to create a similarly helpful and easy-to-understand decision tree simulator in form of a web application that teaches the ID3 algorithm and all necessary surrounding concepts.

2. Project objectives

The primary objective of this project has been to create an interactive and informative web application focused on educating users about the decision tree algorithm ID3. Even though the result that was aimed for was a rather small application, the concepts of entropy and conditional entropy needed to be explained in it, as well. They are prerequisites for understanding decision trees and their creation, making them necessary components in achieving the goal of educating users about the ID3 algorithm through the application.

That is why the first part of the application serves as a tool with which users can learn about entropy and how it is calculated. Alongside cards that hold general information, the user is provided with a calculator of the entropy for self-chosen values and a visual representation of the binary entropy function.

The conditional entropy calculator works in the same way. Users can input different values that represent the number of instances for an example category of an example attribute and calculate the resulting conditional entropy.

The main focus of the decision tree simulator is the dynamic decision tree creation that makes use of Scalable Vector Graphics (SVG) and whose values are based on an underlying dataset. The user can choose between utilizing one of the several example datasets provided by the application and their own datasets that have to be in a CSV file format. A data table shows the entirety of the currently loaded dataset. The step-by-step visualization of the algorithm provides a table that displays relevant values at each step, e.g., the information gain of an attribute. In addition to that, the application presents visual cues that signal the current state of the algorithm at each step.

3. Theoretical concepts

In the following section, all the theoretical concepts relevant for the understanding of the project will be explained.

3.1 Machine Learning

Machine learning [31] is a field of artificial intelligence which focuses on developing algorithms that are able to learn from given input data and, based on that learned knowledge, generalize to unseen data. Machine learning problems can be categorized into three main tasks: Classification problems, Regression problems, and Clustering problems.

Classification problems

Classification problems [16] in machine learning describe the process of predicting a class or category of a given input. A dataset, which is made up of instances that refer to all the individual data points is given as input. Each instance is characterized by a set of features or attributes associated with a label or class. The input data is learned with all its attributes and values to reach the goal of building a model that can also categorize new data, that has not been seen before, into one of the given classes. One example would be classifying emails as spam or not spam. Another could be a dataset that contains information about iris flowers. Its attributes would include, e.g., the petal length or width. Each instance would represent a flower and the label would tell us the type of iris flower they represent. This type of machine learning problem will be discussed in more detail in the section 3.2.

Regression problems

Regression problems [16] in machine learning describe the process of predicting a continuous value based on labeled input data. Instead of predicting a defined class or category, the goal here is to build a model that can predict the quantity of something. An example would be predicting the salary of a person based on their education degree and previous work experience.

Clustering problems

Clustering problems [4] in machine learning do not aim to predict something, like the previous two. Given some unlabeled input data, the goal is to group similar data points together based on certain attributes to generate an output of clusters inside of which the data of points are more similar to each other than to data points in other clusters.

3.2 Classification algorithms

As mentioned before in 3.1, the goal in classification tasks [16] is to assign a class or a label to some input data. Classification aims to create a model that can also generalize to new data. To achieve this, there are multiple algorithms and evaluation metrics that are used to measure the performance of a model.

K-Nearest Neighbors

K-Nearest Neighbors [14], or short KNN, is an algorithm that uses distance as a measure to determine the K nearest neighbors of a data point. Based on that information, a classification or prediction is made about which of the groups the data point is grouped with. KNN does not go through a training stage, instead it just stores and memorizes the training dataset, which is also referred to as “lazy learning”

Logistic Regression

Logistic regression is recognized as a linear classifier [30], which describes an algorithm that makes its prediction based on a linear combination of input attributes. Its goal is to find a linear decision boundary that separates the classes. Logistic regression [13] is mostly used for binary classification problems. It uses the logistic function to determine the probability of a given input belonging to a certain class.

Support Vector Machine

Support Vector Machine (SVM) [32] is an algorithm that can perform linear as well as non-linear classification. Linear classifiers assume that a dataset is linearly separable while non-linear classifiers are not bound to that restrictive belief. With SVM being able to perform both methods, it is able to find a hyperplane that best separates different classes in a dataset by maximizing the margin between classes.

Decision Trees

Decision trees [12] are versatile supervised learning algorithms used for classification and regression tasks. Their goal is to predict the value of a variable based on previously processed input. Their hierarchical structure includes a root node and several internal and leaf nodes.

It starts at the root, which represents the attribute that best separates the underlying dataset based on a certain criterion. One of the most popular ones is the information gain, which is more thoroughly explained in 3.3. From the root on, branches extend to internal nodes, also called decision nodes. These internal nodes also represent attributes along with a decision rule that tells us how to further split the data. These attributes are continuously evaluated until all classes are of the same type or a predefined maximum depth has been reached. Upon reaching that point, leaf nodes are created. They represent all the possible outcomes of the dataset with each one corresponding to a class label.

Decision tree learning utilizes a divide and conquer approach, iteratively finding the best split points until all or most of the input data is classified.

Figure 3.1 shows an example of a decision tree which evaluates whether a person should buy a game or not.

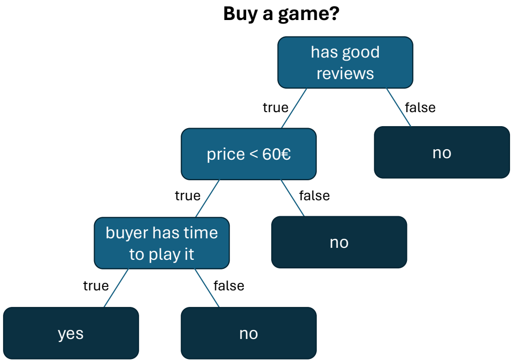


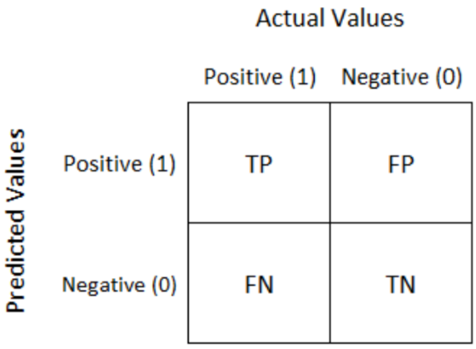
Figure 3.1: A simple decision tree

Evaluation metrics

In the context of classification tasks, evaluation metrics [2] are measures that are used to evaluate the performance of a classification model. They provide information on how well a model performs and are important to ensure its effectiveness. In the following section, some of the most popular evaluation metrics are explained.

Confusion Matrix

A confusion matrix [2] is a table that is used to describe the performance of a classification model. It presents an overview of the predictions on the test dataset against the actual classes. Figure 3.2 shows an example of such a matrix where TP = True Positive, TN = True Negative, FP = False Positive, and FN = False Negative.



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 3.2: Confusion matrix [2]

Accuracy

Accuracy [2] measures how many of all the classified instances were given the correct label. It uses the following formula:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

This metric is useful for well-balanced datasets. In contrast, given a dataset with 99 of its instances belonging to one class and only 1 belonging to the other, likeliness is high that the used model would always predict the class with the higher amount of instances, resulting in a 99% accuracy due to the composure of the dataset.

Precision

Precision [2] describes the ratio of correctly predicted positive cases to the total predicted positive cases. In short words, it measures the accuracy of the positive class predictions and is calculated with the following formula:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision is useful for cases where False Positives are a bigger concern than False Negatives

Recall

Recall [2] describes how many of the actual positive cases our model was able to correctly predict. It measures the ability of the model to find all positive instances and is calculated using the following formula:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Recall is useful when False Negatives are of a higher concern than False Positives.

3.3 Feature Selection

In machine learning, feature selection [19] is a type of preprocessing used to choose the subset of features from the dataset that optimizes the performance of the algorithm. During this process, these algorithms seek to eliminate noisy features or features with redundant information.

But how is determined which feature is better or more important than another? In the context of classification problems, the main idea is to select those features that best prove their ability to differentiate between different target classes. Those ones are most likely to improve the capability of the model to perform prediction tasks and with that, its overall performance.

There are multiple techniques to select the best feature that are focused on different types of tasks and supervision. While each comes with its own advantages and disadvantages, as an example, two supervised techniques are described: Information gain and Fisher's Score. Both have a filter-based approach, so they evaluate the relevance of features by studying their statistical properties or correlation with other features, independently of the learning algorithm itself.

Information gain

Information gain [19] is a method that is commonly used in decision tree algorithms like ID3. It is also used in this project to select the best possible features for the construction of decision trees. It measures the relevance of a feature by evaluating how much information it provides about the class labels. It does so by calculating the reduction in entropy. With that, the higher the information gain of a feature, the more important and useful it is considered for decision-making.

It can be calculated [29] using the following formula:

$$\text{IG}(\text{S}, \text{A}) = \text{E}(\text{S}) - \text{E}(\text{S} \mid \text{A})$$

Where:

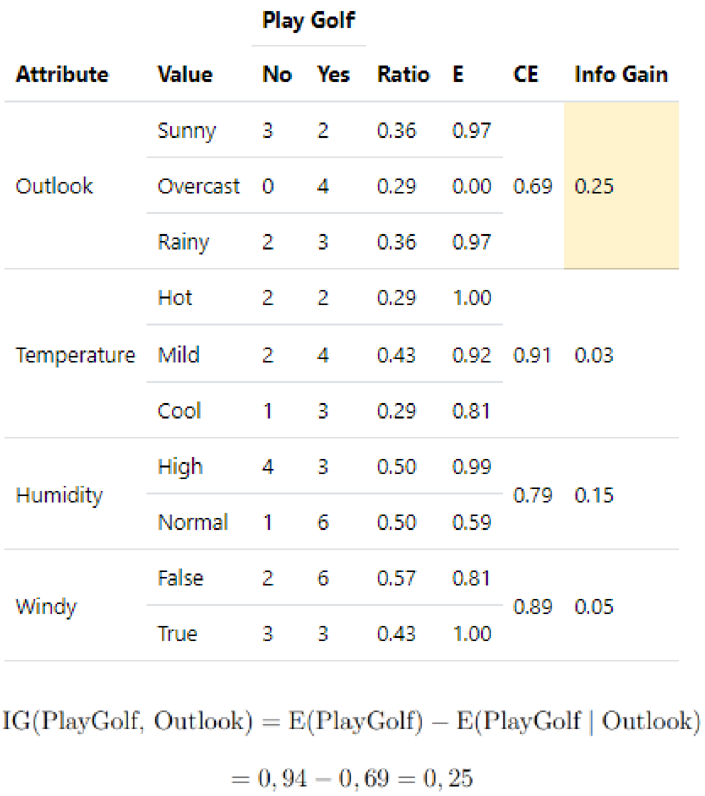
S is the dataset.

A is the attribute/feature for which the information gain is calculated.

E(S) is the entropy (3.4) of the dataset S.

E(S | A) is the conditional entropy (3.4) of dataset S, given attribute/feature A.

Figure 3.3 shows the calculation of information gain based on an example dataset.



		Play Golf					
Attribute	**Value**	**No**	**Yes**	**Ratio**	**E**	**CE**	**Info Gain**
Outlook	Sunny	3	2	0.36	0.97	0.69	0.25
	Overcast	0	4	0.29	0.00		
	Rainy	2	3	0.36	0.97		
Temperature	Hot	2	2	0.29	1.00	0.91	0.03
	Mild	2	4	0.43	0.92		
	Cool	1	3	0.29	0.81		
Humidity	High	4	3	0.50	0.99	0.79	0.15
	Normal	1	6	0.50	0.59		
Windy	False	2	6	0.57	0.81	0.89	0.05
	True	3	3	0.43	1.00		

$$\mathrm{IG(PlayGolf,\ Outlook) = E(PlayGolf) - E(PlayGolf \mid Outlook)}$$

$$= 0,94 - 0,69 = 0,25$$

Figure 3.3: Information gain

The correlation between the entropy of a feature and its information gain values can be observed. The lower the conditional entropy of a feature, the higher the information gain and its usefulness in the process of decision-making. Here, the feature "Outlook" would be chosen as the best one out of the four.

Fisher's Score

Fisher's Score [19], also known as Fisher's Discriminant Ratio, calculates [23] the ability of differentiating classes by determining the ratio of between-class variance and within-class variance. The between-class variance measures the spread of the values of a feature across different classes. A higher value indicates that the means of different classes are further apart, making the feature more discriminative. On the denominator side of the division, the within-class variance determines the spread of the feature values within each class. A lower value means that the data points within each class are closer to the class mean, also making the feature more discriminative.

The calculated value [19] indicates how discriminative and more fitting for classification a feature is. A higher score means higher discrimination and overall usefulness.

3.4 ID3 algorithm

The ID3 algorithm [28] is a machine learning technique used for generating decision trees. Developed by Ross Quinlan in 1986, ID3 is designed to create a model in form of a tree that predicts the value of a target variable based on several input variables of a dataset. It is widely recognized for its simplicity and effectiveness in handling classification problems. Before describing the algorithm in more detail, the concepts of entropy and conditional entropy will be examined as they play an essential role in the creation process of a decision tree.

Entropy

In the context of decision tree algorithms, the entropy [9] can be viewed as a measure of impurity in a dataset. It can also be described as a measure of disorder, uncertainty or the expected surprise of a classification, but going forward, the term impurity will be used to avoid confusion. In datasets with binary classes, where variables can only have two possible outcome values, the entropy value lies between 0 and 1, inclusive. The higher the entropy, the more impure the dataset is. A binary-class dataset that has an equal distribution of, e.g., 5 instances belonging to one class and the other 5 instances belonging to the other class, would have an entropy value of 1. Inversely, a dataset that has all its instances belong to only one class would have an entropy value of 0, making´it a pure dataset. The value is calculated using the following formula:

$$E(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Where:

E(X) is the entropy of dataset X.

n describes the number of classes in the dataset.

$\mathbf{p_i}$ is the proportion of instances in class i or, in other words, the probability of an instance belonging to class i.

Figure 3.4 describes the entropy function in relation to the composition of a dataset and shows clearly how entropy is used to determine its impurity.

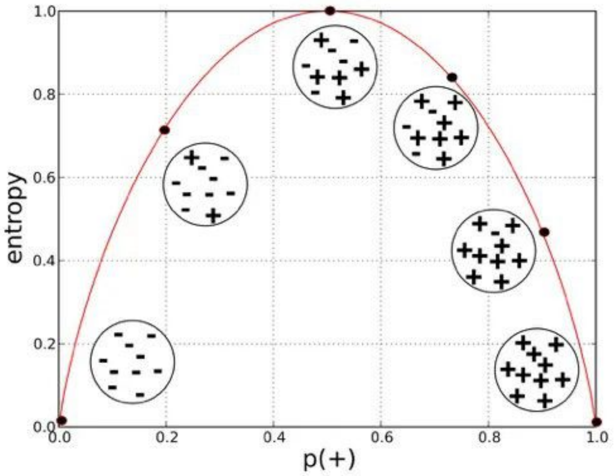


Figure 3.4: Descriptive graph of the entropy function [22]

In a dataset with binary classes, the x-axis describes the amount of instances in a dataset belonging to the positive class while the y-axis measures their entropy values. It records the lowest values at the extremes where there are only or no positive instances recorded in a given dataset while the highest impurity is reached when the numbers of positive and negative instances are equal. When there are more than 2 classes, the scale of the entropy value changes, which makes the visualization more complicated.

Conditional Entropy

In the context of decision trees, the conditional entropy [27] $E(Y \mid X)$ measures the uncertainty in the target variable Y, which usually describes the class labels, given a certain value of the attribute X that is used for splitting a node. It can be described as the weighted sum of $E(Y \mid X = x)$ for every possible value x of X and is calculated using the following formula:

$$\mathrm{E(Y \mid X) = \sum_{x \in X} p(x) E(Y \mid X = x)}$$

The formula sums up the conditional entropies for all the possible values x of attribute X. As the weight, it uses $p(x)$, which is the proportion between the number of instances that have the attribute value x and the size of the dataset. Calculating the conditional entropies for the individual values x of attribute X is possible with the following formula:

$$\mathrm{E(Y \mid X = x) = -\sum_{y \in Y} p(y \mid x) log_2(p(y \mid x))}$$

$p(y \mid x)$ represents the proportion of instances in the dataset with attribute value x that also have the class value y. It sums over all possible values of the target variable Y.

In the end, the conditional entropy evaluates the effectiveness of splitting a node based on the particular attribute X. The lower the conditional entropy, the lower the impurity of the data after splitting the node based on the particular attribute the conditional entropy was calculated for.

Algorithm

The ID3 algorithm [28] constructs decision trees by using a top-down, recursive partitioning approach called "Top-Down Induction of Decision Trees" (TDIDT). With a dataset S and attributes A as input and the tree T as output, the steps are:

1. S will represent the root node of the current tree T.
2. Check if all instances in S belong to the same class.
    - If yes, create a leaf node with the corresponding class label and return the node as T.
3. Check if the attribute set A is empty.
    - If yes, create a leaf node with the most frequent class label and return the node as T.
4. Calculate the information gain for each attribute in A and select A_{best} as the attribute with the highest information gain.
5. Create a decision node containing A_{best}.
6. Split S into subsets based on the different values of A_{best}.
7. For each subset S_i with i starting at 0, do a recursion call with:

    Input: Subset S_i, attributes $A \backslash \{A_{best}\}$.

    Output: Subtree T_i.
8. Append T_i to T's children and return T.

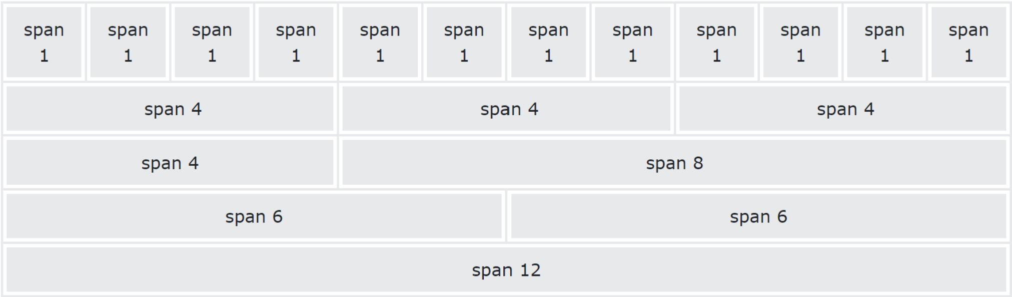
4. Techniques and tools

In this section, all the development tools that have been used to carry out the project are presented. This ranges from frameworks like Bootstrap to version control tools like GitHub. Besides a short introduction, their most notable functions and benefits to the development of the project are documented.

4.1 Bootstrap

Bootstrap [25] is a front-end framework which is known for providing useful and easy-to-use HTML and CSS templates like tables, buttons, forms and many others. It also comes with JavaScript components like modal dialogues and dropdown menus.

One of the key features of Bootstrap is its grid system which lets users divide the contents of their web page into rows and columns. Figure 4.1 shows how the grid system is structured.



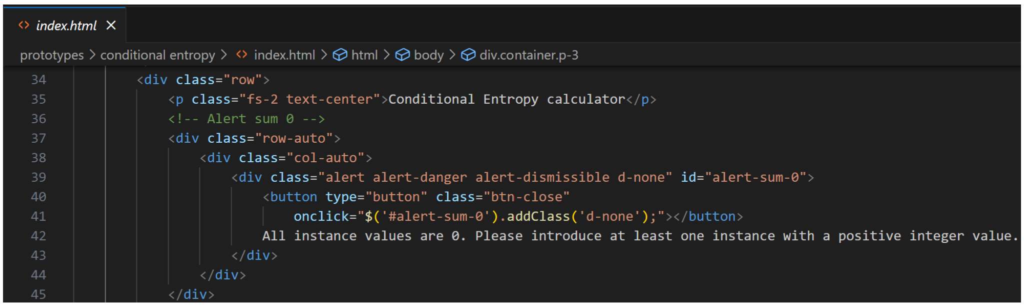
<table>
<tr><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td><td>span 1</td></tr>
<tr><td colspan="4">span 4</td><td colspan="4">span 4</td><td colspan="4">span 4</td></tr>
<tr><td colspan="4">span 4</td><td colspan="8">span 8</td></tr>
<tr><td colspan="6">span 6</td><td colspan="6">span 6</td></tr>
<tr><td colspan="12">span 12</td></tr>
</table>

Figure 4.1: The Bootstrap grid system [24]

Each row possesses 12 columns which the user can freely utilize to organize the contents that are to be displayed. Bootstrap also automatically puts the device on which the web page is displayed into one of six different size categories, based on the screen width of the device. This, in combination with the grid system, allows the developer to make their website responsive to different screens, ranging from large desktop monitors to smartphones.

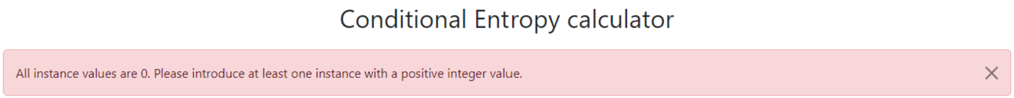
In addition to the ability of Bootstrap to create responsive websites, the wide variety of CSS classes used for common HTML elements, like buttons, offer further simplicity to the web design aspect of creating a page. With this, maintaining a visual consistency throughout the project is also made easier.

In this project, Bootstrap 5 is used. One of the Bootstrap features used is shown in figures 4.2 and 4.3, which displays a custom alert to the user, making use of the grid system of Bootstrap with the "row" and "col" CSS classes and the "alert" class for the alert.


index.html
prototypes > conditional entropy > index.html > html > body > div.container.p-3

```
<div class="row">
    <p class="fs-2 text-center">Conditional Entropy calculator</p>
    <!-- Alert sum 0 -->
    <div class="row-auto">
        <div class="col-auto">
            <div class="alert alert-danger alert-dismissible d-none" id="alert-sum-0">
                <button type="button" class="btn-close"
                    onclick="$('#alert-sum-0').addClass('d-none');"></button>
                All instance values are 0. Please introduce at least one instance with a positive integer value.
            </div>
        </div>
    </div>
```

Figure 4.2: Usage of Bootstrap


Conditional Entropy calculator

All instance values are 0. Please introduce at least one instance with a positive integer value.

Figure 4.3: Bootstrap Alert

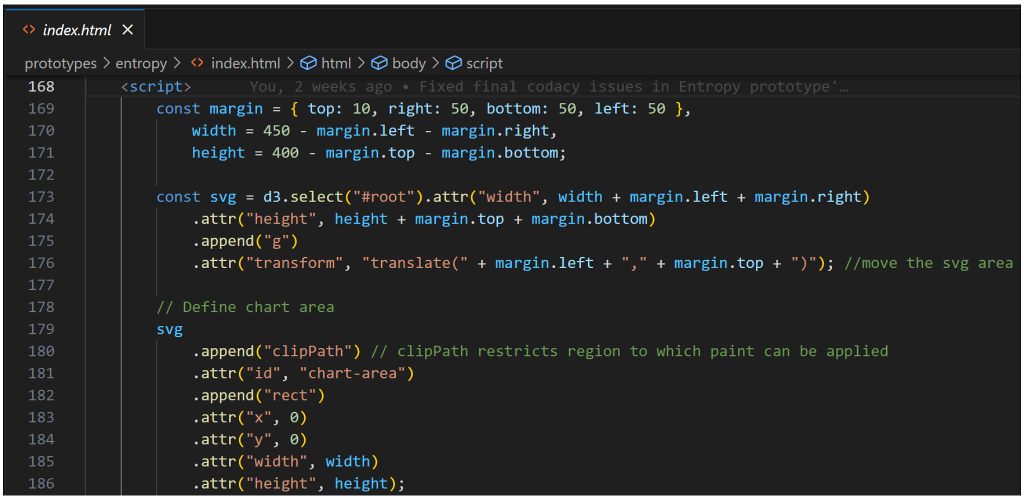
4.2 D3.js

D3 [7] stands for "data-driven documents" and perfectly describes the free, open-source JavaScript library. "Documents" refers to the Document Object Model (DOM). D3 lets the user bind data to its elements. The library works like a toolbox that uses a variety of discrete modules which, e.g., allow selection and transition operations. It binds these modules together so all the necessary tools are at hand, ready to be applied.

D3 makes use of web standards like SVG to display contents. Incorporating these standards, the library also allows the use of external stylesheets which can be employed to change the visual representations of the graphics.

A major feature of D3 is its ability to dynamically change the displayed contents. Whether that change is triggered by user interactions or a change in underlying data, the "data join" concept of the library allows separate operations for entering, updating and exiting existing DOM elements based on a given set of data. Besides filtering and sorting, it lets the user control what happens to their contents in many ways when changes happen and update their website accordingly.

Figures 4.4, 4.5, and 4.6 show the usage of D3 in this project. It was used to create a coordinate system with a graph displaying the binary entropy function. It is also used to dynamically add a point and line to the graph, corresponding to the binary entropy value that was calculated for the input values of the user:

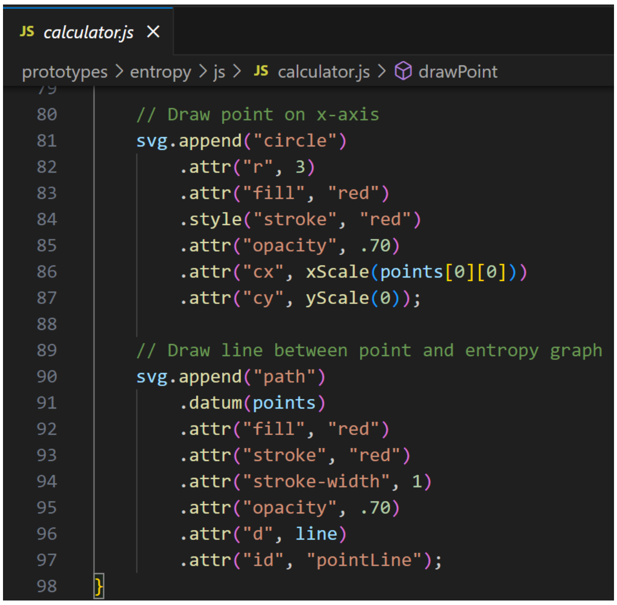


```
<script>        You, 2 weeks ago • Fixed final codacy issues in Entropy prototype'…
    const margin = { top: 10, right: 50, bottom: 50, left: 50 },
        width = 450 - margin.left - margin.right,
        height = 400 - margin.top - margin.bottom;

    const svg = d3.select("#root").attr("width", width + margin.left + margin.right)
        .attr("height", height + margin.top + margin.bottom)
        .append("g")
        .attr("transform", "translate(" + margin.left + "," + margin.top + ")"); //move the svg area

    // Define chart area
    svg
        .append("clipPath") // clipPath restricts region to which paint can be applied
        .attr("id", "chart-area")
        .append("rect")
        .attr("x", 0)
        .attr("y", 0)
        .attr("width", width)
        .attr("height", height);
```

Figure 4.4: Usage of D3: Defining the chart area



```
    // Draw point on x-axis
    svg.append("circle")
        .attr("r", 3)
        .attr("fill", "red")
        .style("stroke", "red")
        .attr("opacity", .70)
        .attr("cx", xScale(points[0][0]))
        .attr("cy", yScale(0));

    // Draw line between point and entropy graph
    svg.append("path")
        .datum(points)
        .attr("fill", "red")
        .attr("stroke", "red")
        .attr("stroke-width", 1)
        .attr("opacity", .70)
        .attr("d", line)
        .attr("id", "pointLine");
}
```

Figure 4.5: Usage of D3: drawing the point and line

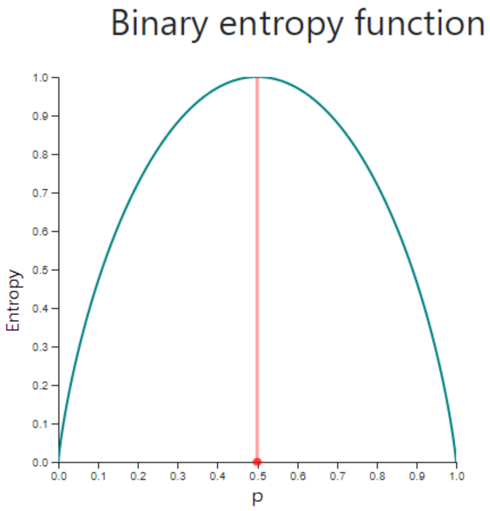
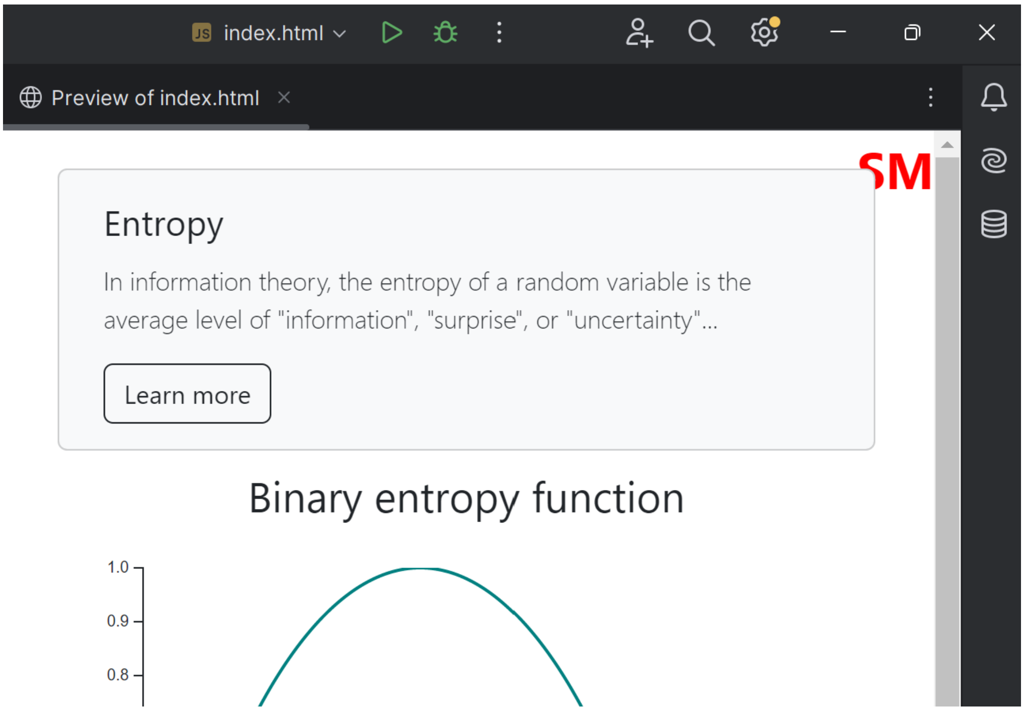


Figure 4.6: D3 graph

4.3 PyCharm

PyCharm [21] is the Integrated Development Environment (IDE) that was used during the first two and part of the third sprint. It is an IDE developed by JetBrains which is specifically designed for Python development. With its support for web development frameworks like Flask and features like code completion for HTML, CSS and JavaScript, it served useful in allowing a quick start into the development of this project. The built-in live preview for HTML files and its interactive debugging feature also made working on the prototypes much simpler. Figure 4.7 shows the usage of that feature.



Entropy

In information theory, the entropy of a random variable is the average level of "information", "surprise", or "uncertainty"...

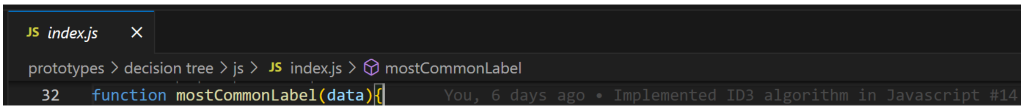
Figure 4.7: Live preview of PyCharm

Its version control integration of systems like Git allowed an easier experience of making local changes remotely available.

However, one of the issues during the third sprint was to start using Bootstrap to make the layout of the website responsive. Due to set-up problems with the IDE, PyCharm was not able to provide auto-completion for the framework. To make use of that and other features, a different program overtook the task of developing the application.

4.4 Visual Studio Code

Visual Studio Code [17] is an open-source code editor created by Microsoft. While it also provides all the benefits mentioned in section 4.3, it comes with a handful of other advantages of which the most significant one would be the vast variety of available extensions. These include Bootstrap IntelliSense which enables CSS class auto-completion, Live Server which launches a local server with a live reload feature, and GitLens which, e.g., allows the user to see inline information about when and in which commit the current line of code was last changed. Figure 4.8 shows this feature.


JS index.js ×
prototypes > decision tree > js > JS index.js > mostCommonLabel

```
function mostCommonLabel(data){        You, 6 days ago • Implemented ID3 algorithm in Javascript #14
```

Figure 4.8: GitLens extension

These extensions allow the user to add features based on what they need. By only including essential coding features upon first installation, the editor take up less space than IDEs like PyCharm. In addition, its lightweight design that is optimized for performance leaves behind a smaller memory footprint.

4.5 GitHub

GitHub [10] is a platform that is mainly used for version control on software development projects. Not only does it provide a location for users to store their files, make and track changes, it also makes it possible to share repositories with other developers and therefore presents an easy way to collaborate on projects.

Project management is made easier with GitHub, too. Providing tools for code review, milestone and issue tracking, it gives participants of the project insight on the current progress. Communication through comments on specific issues, commits, pull requests or code reviews is possible, too, allowing for organized feedback.

In this project, GitHub has been used to store all relevant files and update the development in the repository[2]. Issues were used to set up defined tasks and track progress. In addition to that, a GitHub Pages repository[3] was

[2] GitHub: https://github.com/danieldf01/TFG-decision-trees-sim/

[3] GitHub Pages repository: https://danieldf01.github.io/

used to deploy prototypes for different functionalities of the web application. GitHub Pages is a site hosting service that allows the user to host a website directly from a GitHub repository. It simplified the sprint reviews with the tutors as the deployed web application could simply be opened on any browser without having to launch it from an IDE or a code editor.

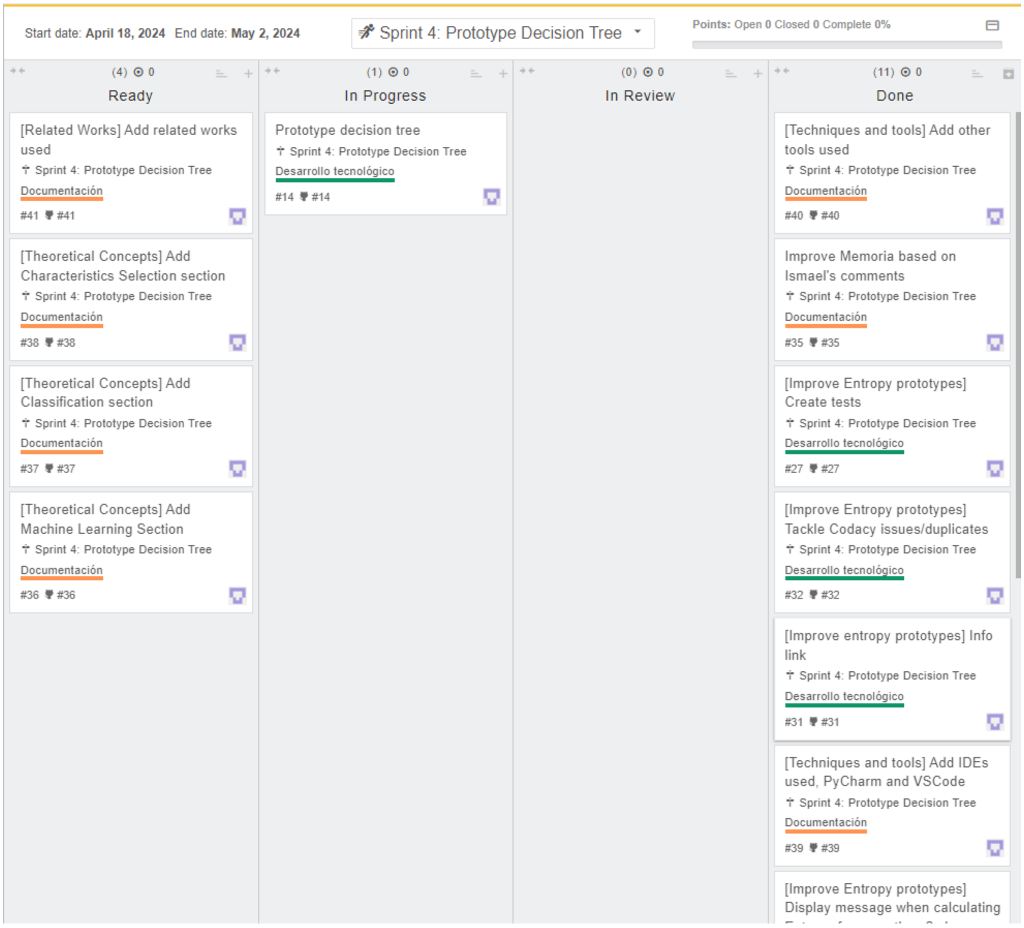
All these functions have been in the present work to ensure an organized and transparent development process.

4.6 Zube

Zube [15] is a project management platform for software development teams that comes with tools that help to plan and manage projects efficiently. Allowing a direct link to GitHub repositories, issues are synced so that when a change is made to those linked issues on GitHub, that information is transferred to Zube, and vice versa.

Tracking progress is made easy with the inclusion of sprints and allowing the user to add issues to a sprint board which can be used for project management methodologies like Scrum. Charts like Burndown and Burnup show the progress of each sprint over the time set for the sprint. The Burndown charts were used to analyze the temporal planning throughout the project in the corresponding section in the annexes.

Figure 4.9 shows the Sprint Board which, among other tools, was used to plan out each sprint thoroughly and document each step in regards to the created issues.



Start date: **April 18, 2024** End date: **May 2, 2024**

Sprint 4: Prototype Decision Tree

Points: Open 0 Closed 0 Complete 0%

(4) ⊙ 0 Ready

- [Related Works] Add related works used — Sprint 4: Prototype Decision Tree — Documentación — #41 #41
- [Theoretical Concepts] Add Characteristics Selection section — Sprint 4: Prototype Decision Tree — Documentación — #38 #38
- [Theoretical Concepts] Add Classification section — Sprint 4: Prototype Decision Tree — Documentación — #37 #37
- [Theoretical Concepts] Add Machine Learning Section — Sprint 4: Prototype Decision Tree — Documentación — #36 #36

(1) ⊙ 0 In Progress

- Prototype decision tree — Sprint 4: Prototype Decision Tree — Desarrollo tecnológico — #14 #14

(0) ⊙ 0 In Review

(11) ⊙ 0 Done

- [Techniques and tools] Add other tools used — Sprint 4: Prototype Decision Tree — Documentación — #40 #40
- Improve Memoria based on Ismael's comments — Sprint 4: Prototype Decision Tree — Documentación — #35 #35
- [Improve Entropy prototypes] Create tests — Sprint 4: Prototype Decision Tree — Desarrollo tecnológico — #27 #27
- [Improve Entropy prototypes] Tackle Codacy issues/duplicates — Sprint 4: Prototype Decision Tree — Desarrollo tecnológico — #32 #32
- [Improve entropy prototypes] Info link — Sprint 4: Prototype Decision Tree — Desarrollo tecnológico — #31 #31
- [Techniques and tools] Add IDEs used, PyCharm and VSCode — Sprint 4: Prototype Decision Tree — Documentación — #39 #39
- [Improve Entropy prototypes] Display message when calculating

Figure 4.9: Sprint Board

4.7 Codacy

Codacy [8] is a code analysis tool that helps software developers improve their code quality. Allowing a link to GitHub, it reviews commits in real-time and checks the quality of the code based on code security, code duplication, coding style and other general issues. The coding standards that are initially introduced by Codacy can be customized to fit the quality standards of the developres by making it possible to, e.g., ignore certain issues that are irrelevant to the project, introduce other security guidelines and enforce specific coding styles of their own.

As seen in figure 4.10, which shows the timeline of existing issues within the code in the remote repository, Codacy has been used to continuously identify and tackle all sorts of problems in the code that arose during the development of the web application.

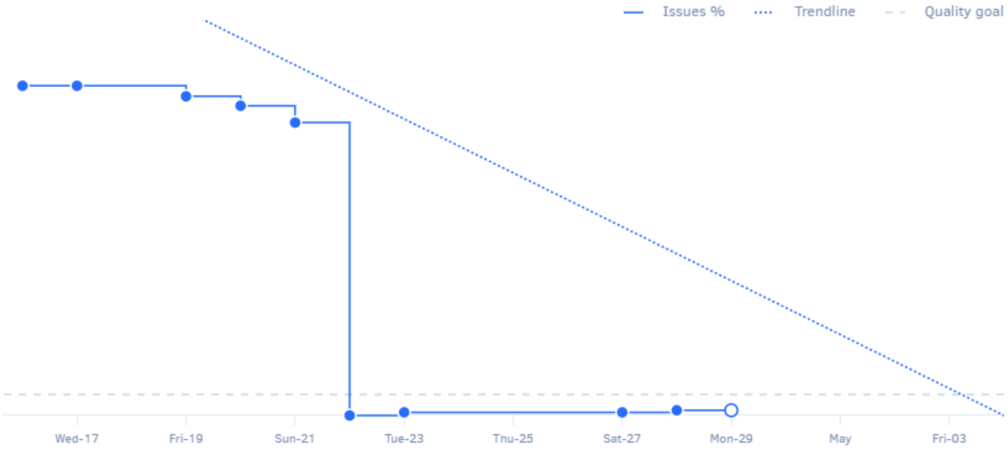


Figure 4.10: Codacy quality evolution

5. Relevant aspects of the project development

This chapter focuses on important parts of the project development that had a significant influence on the outcome of the work.

5.1 Start of the project

At the start of the project, the student was not experienced in web development. For that reason, the first sprints were not only used to learn about the concepts relevant to the topic of decision trees and start implementing first prototypes, but also to simultaneously gain an understanding of web development using HTML, JavaScript, CSS and surrounding tools. The same applies to LaTeX, which also took some time to get used to. That rhythm continued throughout most of the project as almost each step introduced techniques that were new, but had to be learned and then implemented. This slowed down overall development tempo to some extent.

5.2 Decision tree creation without D3

As mentioned in 4.2, D3 is an open-source JavaScript library which, among other things, allows the user to dynamically create and change DOM contents based on underlying data. For that reason, at the beginning of development, it was chosen to help create the graph for the Entropy function and later, dynamically create and change the decision trees examples.

However, as development of the project went on and it was time to create a prototype for the creation of decision trees, the plan of using D3 as the main tool to reach that goal was discarded. This decision was made after an extensive web research on how SVGs displaying trees were created with the library and how they would end up looking. With a few exceptions showing slight deviations, the common structure of D3 trees was as figure 5.1 displays.

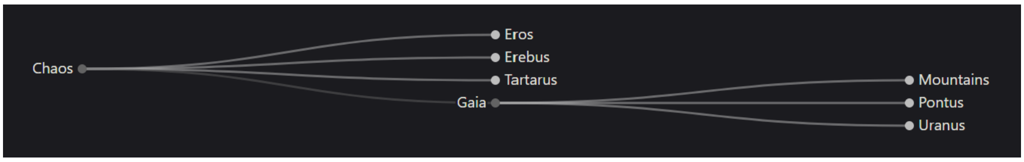


Figure 5.1: SVG displaying a tree, created with D3 [6]

Since this specific library was not exactly easy to use in the first place, trying to make it look the intended way (displaying labels on the edges and other important values connected to the nodes) would have made the creation significantly harder, if it had even been possible at all.

With that in mind, the decision was made not to use D3 to create the decision trees, but to assemble the decision trees from simple SVG elements to form templates for branch paths, decision and leaf nodes, and to develop an algorithm that calculates the positions of elements of a tree in an SVG that is responsive to different screen sizes.

Even though that solution meant more work, it allowed the decision trees to look the intended way in the end, as figure 5.2 shows.

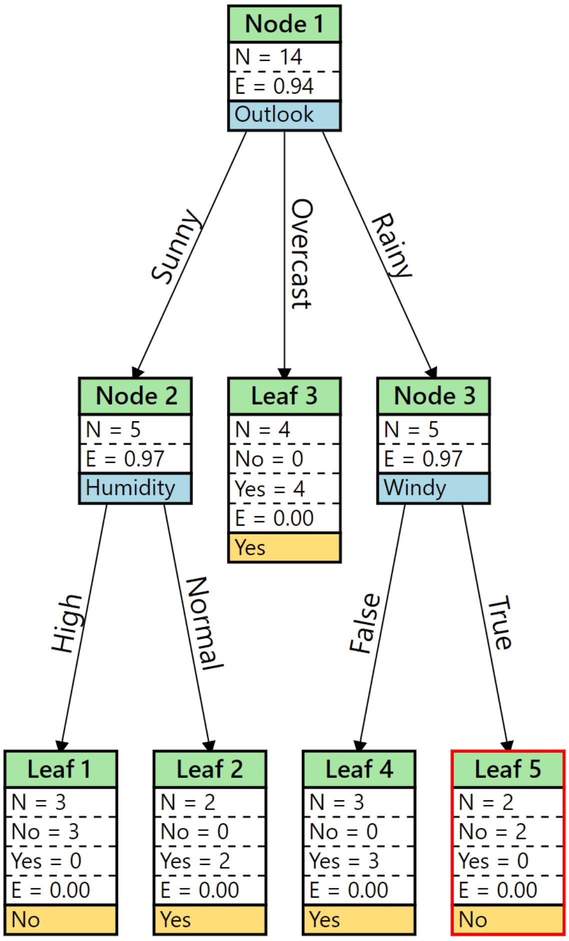


Figure 5.2: Dynamically created decision tree using the web application

5.3 Evaluation of code quality with Codacy before reviews

As mentioned in 4.7, Codacy is a code analysis tool that helps developers to improve the quality of their code. With the biweekly meetings having functioned not only as a way to figure out the tasks for the next sprint, but also as an opportunity to review what had been done so far, there was often no time to check the code quality. Codacy provided everything needed to improve the code condition even before the review with the tutors.

It offered the opportunity of checking the quality of the code without having to wait for a review meeting or ask the tutors whether a certain way of implementing something was good or not. This gave the review meetings more space to focus on what had been done rather than how it had been done. Especially during the meetings towards the end of development, this extra time was needed to discuss additional changes that needed to be made before the project was finalized.

5.4 Decision between two features

Due to time constraints, some features had to be left behind. Two features, however, were still talked about in the final weeks before completion. One was to make the datasets for the decision tree ID3 simulator interactive, meaning that the user would be able to dynamically add columns and rows to a dataset that was already loaded. The other was to let the user load own datasets in the form of Comma-separated values (CSV) files to let the application build a decision tree upon.

With the remaining time, it was clear that it would not be possible to implement both features, so a decision had to be made between them. Ultimately, the feature of letting users upload their own datasets in a CSV file format was given higher priority and implemented. This feature gives the user more freedom by letting them choose a dataset from the internet or create their own and use it to observe how different compositions of datasets influence the resulting decision tree.

5.5 Limitations of a small web application

Towards the end of development, some restrictions had to be taken into account and defined. One of those was a limited size of CSV files uploaded by users. E.g., the file cannot contain more than 150 rows of instances. The main reason for this limitation is the resulting size of the decision trees. With trees that are too big, the user might not be able to read individual the contents of individual nodes and therefore have a hard time understanding the algorithm. That would defeat the whole purpose of this project.

Another drawback that the size of the application brings with it is the possibility of branches or their labels overlapping with other nodes. Figure 5.3 shows how the branch label “very small” overlaps with the leaf node “Leaf 8”.

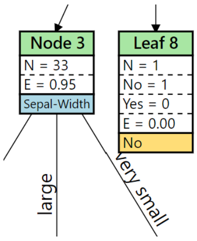


Figure 5.3: Branch label overlapping with leaf node

Due to time constraints, this is a problem that could not be given higher priority for resolution than other necessary refinements, as it would have brought a higher level of workload than was acceptable at the time.

6. Related works

6.1 Web Thoth

Web Thoth [18] is the final project presented by Ismael Ramos Pérez, one of the tutors for this project, as part of his degree in Computer Engineering. It was created in 2016 with the help of tutors Dr. César Ignacio García Osorio and Dr. Álvar Arnaiz González. It is composed of a website that helps users learn lexical analysis algorithms by showing explanations and examples. With its goal being of similar nature to the one of this project, its main influences lie in the layout and overall functionality of the presented website. E.g., one of its main features, the step-by-step simulation of the presented algorithms, has had a significant influence on this project.

6.2 K-Nearest Neighbors Demo

This interactive demo [1] of the K-Nearest Neighbor algorithm displays a decision boundary plot that changes dynamically as the user selects different metrics or alters the number of data points, classes, or neighbors. The dynamic nature of this demo has been another inspiration for the way this decision tree simulator would be implemented in the end.

6.3 SVM demo

The Interactive demo of Support Vector Machines (SVM) [11] is a web application that lets the user input data points of either one of two classes on a canvas which results in the creation of hyperplanes that best separate the two classes. Based on the input of the user and the adjustable parameters, the calculation and display of the hyperplanes changes dynamically. That dynamic change in data is also implemented in this project.

6.4 A Visual and Interactive Look at Basic Neural Network Math

The goal of this work [3] is to teach the user the concepts of basic neural network math. It includes intuitive visual representations of a neural network, its elements and calculations, and the ReLU function. Additionally, the user can change some of the input data, to which the application responds with visual cues as to what kind of changes were made. This case of dynamic visualization of changing data has also been an influence on this project.

7. Conclusions and future lines of work

7.1 Conclusions

One of the hurdles along the way was to acquire knowledge about the technologies and tools that were necessary to carry out this project in a satisfactory way. This was successfully achieved for technologies like JavaScript, CSS, Bootstrap, D3, and others that were relevant for web development. With that, the main objectives were completed in a way that realizes the initial vision of the project. A web application was built that shows the concept of decision trees, the behavior of the ID3 algorithm, and surrounding concepts like entropy. It is a tool that can help students make this topic more tangible and simple to understand during their studies.

7.2 Future lines of work

Resolve shortcomings

Firstly, a future expansion of this project could resolve all the shortcomings. Mentioned in 5.4 and 5.5, these include the overlapping of branch paths with nodes, CSV file limitations for users due to the size of the application, and the lack of the feature that allows for interactive data.

Questionnaires

Much like in the research article [5] about the "Seshat Tool" that was mentioned in the introduction, a similar study could be made about the effectiveness of this decision tree simulator. Two groups of students could be asked to take the same questionnaire about the topics of entropy, decision trees and their creation using the ID3 algorithm. One group would be prepared only through theoretical input in form of a lecture, the other would additionally be given access to this application. The results would be analyzed to see if there was a difference in test scores between the two groups.

C4.5 algorithm

The C4.5 algorithm [26] is an algorithm that extends ID3 by being able to handle both continuous and discrete attributes, handle missing attribute values, and prune trees to avoid overfitting. A future work could expand upon this application and implement a simulator for the C4.5 algorithm.

Furthermore, a scatter plot could be used to present how a tree with numerical values works. In this way, the understanding of the user of that type of decision trees could be strengthened while additionally, the concept of decision boundaries could be explained in an interactive way.

Regression trees

A future addition to this application could implement a simulator that teaches the concept and creation of regression trees. E.g., the Classification And Regression Tree (CART) algorithm [20], which, as the name suggests, is an algorithm that is applicable for classification and regression problems, could be taught interactively.

Bibliography

[1] K-nearest neighbors demo. http://vision.stanford.edu/teaching/cs231n-demos/knn/, n.d. [Internet; visited 09-june-2024].

[2] Sumeet Kumar Agrawal. Metrics to evaluate your classification model to take the right decisions. https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/, 2024. [Internet; visited 02-may-2024].

[3] Jay Alammar. A visual and interactive look at basic neural network math. https://jalammar.github.io/feedforward-neural-networks-visual-interactive/, 2018. [Internet; visited 04-may-2024].

[4] Moez Ali. Clustering in machine learning: 5 essential clustering algorithms. https://www.datacamp.com/blog/clustering-in-machine-learning-5-essential-clustering-algorithms, 2022. [Internet; visited 30-april-2024].

[5] Álvar Arnaiz-González, Jose-Francisco Díez-Pastor, Ismael Ramos-Pérez, and César García-Osorio. Seshat — a web-based educational resource for teaching the most common algorithms of lexical analysis. *Computer Applications in Engineering Education*, 26(6):2255–2265, 2018.

[6] Mike Bostock and Inc. Observable. Tree. https://d3js.org/d3-hierarchy/tree, 2024. [Internet; visited 12-june-2024].

[7] Mike Bostock and Inc. Observable. What is d3? https://d3js.org/what-is-d3, 2024. [Internet; visited 15-april-2024].

[8] Codacy. Codacy docs. https://docs.codacy.com, 2024. [Internet; visited 08-june-2024].

[9] Shailey Dash. Decision trees explained — entropy, information gain, gini index, ccp pruning. https://towardsdatascience.com/decision-trees-explained-entropy-information-gain-gini-index-ccp-pruning-4d78070db36c#:~:text=In%20the%20context%20of%20Decision,only%20pass%20or%20only%20fail., 2022. [Internet; visited 29-march-2024].

[10] Inc. GitHub. About github. https://github.com/about, 2024. [Internet; visited 08-june-2024].

[11] Jonas Greitemann. Interactive demo of support vector machines (svm). https://greitemann.dev/svm-demo, 2018. [Internet; visited 04-may-2024].

[12] IBM. What is a decision tree? https://www.ibm.com/topics/decision-trees#:~:text=A%20decision%20tree%20is%20a,internal%20nodes%20and%20leaf%20nodes., -. [Internet; visited 15-march-2024].

[13] IBM. What is logistic regression? https://www.ibm.com/topics/logistic-regression, 2024. [Internet; visited 02-may-2024].

[14] IBM. What is the k-nearest neighbors (knn) algorithm? https://www.ibm.com/topics/knn#:~:text=The%20k-nearest%20neighbors%20(KNN,used%20in%20machine%20learning%20today., 2024. [Internet; visited 02-may-2024].

[15] Pivit Inc. Zube documentation. https://zube.io/docs, 2024. [Internet; visited 08-june-2024].

[16] Zoumana Keita. Classification in machine learning: An introduction. https://www.datacamp.com/blog/classification-machine-learning, 2022. [Internet; visited 30-april-2024].

[17] Microsoft. Code editing. redefined. https://code.visualstudio.com, 2024. [Internet; visited 21-april-2024].

[18] Ismael Ramos Pérez. Seshat tool. http://cgosorio.es/Seshat/, 2018. [Internet; visited 04-may-2024].

[19] Nate Rosidi. Advanced feature selection techniques for machine learning models. https://www.kdnuggets.com/2023/06/advanced-feature-selection-techniques-machine-learning-models.html, 2023. [Internet; visited 14-may-2024].

[20] Ayush Singhal and Gaurav Shukla. Decision trees : Classification and regression trees (cart). https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf, 2023. [Internet; visited 09-june-2024].

[21] JetBrains s.r.o. The python ide for data science and web development. https://www.jetbrains.com/pycharm/, 2024. [Internet; visited 21-april-2024].

[22] Sam T. Entropy: How decision trees make decisions. https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8, 2019. [Internet; visited 29-march-2024].

[23] Jiliang Tang, Salem Alelyani, and Huan Liu. *Feature selection for classification: A review*, pages 37–64. CRC Press, January 2014. Publisher Copyright: © 2015 by Taylor & Francis Group, LLC.

[24] W3Schools. Bootstrap 5 grids. https://www.w3schools.com/bootstrap5/bootstrap_grid_basic.php, 2024. [Internet; visited 11-june-2024].

[25] W3Schools. Bootstrap 5 tutorial. https://www.w3schools.com/bootstrap5/index.php, 2024. [Internet; visited 15-april-2024].

[26] Wikipedia. C4.5 algorithm. https://en.wikipedia.org/wiki/C4.5_algorithm, 2024. [Internet; visited 08-june-2024].

[27] Wikipedia. Conditional entropy. https://en.wikipedia.org/wiki/Conditional_entropy, 2024. [Internet; visited 29-march-2024].

[28] Wikipedia. Id3 algorithm. https://en.wikipedia.org/wiki/ID3_algorithm, 2024. [Internet; visited 08-june-2024].

[29] Wikipedia. Information gain (decision tree). https://en.wikipedia.org/wiki/Information_gain_(decision_tree), 2024. [Internet; visited 14-may-2024].

[30] Wikipedia. Linear classifier. https://en.wikipedia.org/wiki/Linear_classifier#, 2024. [Internet; visited 11-june-2024].

[31] Wikipedia. Machine learning. https://en.wikipedia.org/wiki/Machine_learning, 2024. [Internet; visited 30-april-2024].

[32] Wikipedia. Support vector machine. https://en.wikipedia.org/wiki/Support_vector_machine, 2024. [Internet; visited 02-may-2024].