

TDT4171 Artificial Intelligence Methods

Assignment 4

February 2022

1 Information

- **Delivery deadline: March 18, 2022, by 23:59.** No late delivery will be graded! Deadline extensions will only be considered for extraordinary situations such as family or health-related circumstances. However, these circumstances must be documented, e.g., with a doctor's note ("legeerkl ring"). Having a lot of work in other classes is not a legitimate excuse for late delivery.
- Students can NOT work in groups. Each student can only submit a solution individually.
- This homework counts for 4% of the final grade.
- Cribbing("koking") from other students is not accepted, and if detected, will lead to immediate failure of the course. The consequence will apply to both the source and the one cribbing.
- Required reading for this assignment: Chapter 19 Learning From Examples (the parts in the curriculum found on Blackboard "Sources and syllabus" → "Preliminary syllabus") in **Artificial Intelligence: A Modern Approach, Global Edition 4th Edition, Russell & Norvig**. All the page references follow this version of the book.
- For help and questions related to the assignment, ask the student assistants during the guidance hours or use Piazza. The guidance hours and link to Piazza can be found under "Course work" on Blackboard. For other inquires, an email can be sent to tdt4171@idi.ntnu.no.
- Deliver your solution on Blackboard. Please upload your assignment as one PDF report and one source file containing the code (e.g., a .py file). Please do not put the files into an archive.

ASSIGNMENT SUBMISSION

Text Submission

Attach Files

Attached files



File Name	Link Title	
 my_code.py	<input type="text" value="my_code.py"/>	Do not attach
 my_report.pdf	<input type="text" value="my_report.pdf"/>	Do not attach

Figure 1: Delivery Example

2 Assignment

The main goal of this assignment is to implement the decision tree learning algorithm from scratch. The pseudocode for the decision tree learning algorithm can be found in Figure 19.5 on Page 678. We recommend using Python as the programming language for this assignment. If you choose to use another programming language, we cannot guarantee that we are able to provide you with help if you run into issues with your implementation.

For the programming part of the assignment, the code must be runnable without any modifications after delivery. In addition, the code must be human-readable and contain explaining comments where appropriate. Commenting is especially important if the code does not work.

It is not allowed to use packages such as Scikit-learn to implement the decision tree learning algorithm. Furthermore, copying an implementation of the decision tree learning algorithm from the internet is not allowed.

2.1 Implementation

Two different versions of IMPORTANCE function used by the pseudocode in Figure 19.5 on Page 678 should be implemented.

1. Allocate a random number as importance to each attribute.
2. Allocate a number using the expected information gain as importance to each attribute. For more information about this, see Section 19.3.3 Choosing attribute tests on Page 679.

To compare the two versions of IMPORTANCE, you should examine each of them by doing the following steps:

1. Learn a decision tree from the data in `train.csv`.
2. Document the tree you got in an easy-to-read format in the PDF report. The tree can be documented, for example, by visualizing it similar to Figure 19.6 on Page 678. The visualization can, for example, be produced using the [Graphviz](#) package for Python. See Section 4 for an example made using Graphviz. How to install Graphviz can be found [here](#), and how to use [here](#).
3. Classify all examples in the test set given in the data file `test.csv`. Calculate the accuracy of the learner by comparing them to the correct classification of the examples in the test set. Then, add the result to the PDF report.

2.2 Discussion

Discuss your findings in the PDF report.

- What can you conclude about the results you have obtained? Which IMPORTANCE function is better, and why?
- Do you get the same result if you run the random IMPORTANCE several times?
- What happens when you run the learner based on Information Gain several times?

3 Experiment Data

Two data files (`train.csv` and `test.csv`) can be found on Blackboard along with this assignment text. The two data files have the same format: Each line describes an example, the first seven numbers are the attributes, the last number is the class of that example. All attributes, as well as the class, take values 1 or 2, and you can take advantage of this to simplify your code if you want.

Example

The first line in the training data is:

1,1,2,2,1,1,1,1.

This means that for this example, we have

- Attribute 0 = 1,
- Attribute 1 = 1,
- Attribute 2 = 2,
- Attribute 3 = 2,
- Attribute 4 = 1,
- Attribute 5 = 1,
- Attribute 6 = 1,
- ...and that the example comes from class 1.

The test data has the same format. However, the class label for the test data is not to be used by the decision tree during learning, only to quantify the learning algorithm's accuracy afterward.

4 Example of Visualization

The code in Listing 1 shows how to use Graphviz in Python. Figure 2 illustrates the corresponding tree produced using the code in Listing 1. For more details on how to use the Graphviz package, see [here](#).

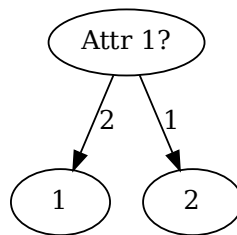


Figure 2: Example of a decision tree visualized using the Graphviz package and the code in Listing 1.

```

1  # Import Graphviz
2  # How to install: https://graphviz.readthedocs.io/en/stable/manual.html#installation
3  import graphviz
4
5  # 1. Create the tree
6  # filename is the filename when the tree is saved to disk
7  tree = graphviz.Digraph(name="Decision Tree", filename="example_dt")
8
9  # 2. Add nodes
10 # name is a unique identifier for the node.
11 # label is the caption to be displayed when the tree is visualized.
12 tree.node(name="attr_1", label="Attr 1?")
13
14 tree.node(name="value_1", label="1")
15 tree.node(name="value_2", label="2")
16
17 # 3. Add edges
18 # tail_name is the start node of the edge.
19 # head_name is the end node of the edge.
20 # label is the caption to be displayed when the tree is visualized.
21 tree.edge(tail_name="attr_1", head_name="value_1", label="2")
22 tree.edge(tail_name="attr_1", head_name="value_2", label="1")
23
24 # 4. Save the tree to disk
25 # Optionally, set the argument
26 # view to True to display the visualization immediately
27 tree.render(view=True)

```

Listing 1: Example on how to use Graphviz.