INF264
Project 1:
Implementing decision trees

Deadline: September 27th, 23.59
Deliver here:
`https://mitt.uib.no/courses/19532/assignments/22074`

Projects are a compulsory part of the course. This project contributes a total of 20 points to the final grade. You need to upload your answer to MittUiB before 23.59 on the 27th of September.

Grading will be based on the following qualities:

- Correctness (your answers/code are correct and clear)
- Clarity of code (documentation, naming of variables, logical formatting)
- Reporting (thoroughness and clarity of the report)

Deliverables. You should deliver exactly two files:

1. a PDF report containing an explanation of your approach and design choices to help us understand how your particular implementation works. You can include snippets of code in the PDF to elaborate any point you are trying make.
2. a zip file of your code. We may want to run your code if we feel necessary to confirm that it works the way it should. Please include a README.txt file in your zip file that explains how we should run your code. In case you have multiple files in your code directory, you must mention in the README.txt file which file is the main file that we need to run to execute your entire algorithm.

Programming languages. The course staff supports Python users. If you want to use some other language, please contact teaching assistants Pierre Gillot (pierre.gillot@uib.no) and Ramin Hasibi (ramin.hasibi@uib.no) before you start coding to check whether they can run and grade the project with your language of preference.

Code of conduct. The goal of the project is learning to implement machine learning algorithms. Therefore, you must write the code yourself. You are not allowed to use existing libraries. The decision tree implementation can import only basic packages like `numpy`; for model selection and evaluation as well as for visualisation you can import packages like `sklearn.metrics.accuracy_score` or `matplotlib`. Furthermore, it is not allowed to copy-paste code from online tutorials. If you are unsure whether something is allowed or not, ask the teaching assistants.

Late submission policy: All late submissions will get a deduction of 2 points. In addition, there is a 2-point deduction for every starting 12-hour period. That is, a project submitted at 00.01 on September 28th will get a 4-point deduction and a project submitted at 12.01 on the same day will get a 6-point deduction (and so on). All projects submitted on September 30th or later are automatically failed. (Executive summary: Submit your project on time.) There will be no possibility to resubmit failed projects so start working early.

1 Tasks

Write a computer program that solves the following tasks. Then write a short report that explain your approach and design choices (Tasks 1.1-1.3) and the results of your experiments as well as the experimental procedure you used (Tasks 1.4-1.5).

1.1 Implement the ID3 algorithm from scratch

Implement the ID3 algorithm using entropy as the impurity measure. Your implementation should have two functions that the users can use:

1. `learn(X, y, impurity_measure='entropy')`
2. `predict(x, tree)`

The function `learn` learns a decision tree classifier from a data matrix $\mathbf{X}$ and a label vector $\mathbf{y}$. We consider the classification task so you can assume that $\mathbf{y}$ consists of categorical variables.

The function `predict` predicts the class label of some new data point $\mathbf{x}$.

Note: If you implement your tree in object-oriented fashion, it is not necessary to include argument `tree`.

Note: For debugging and visualisation purposes, it may be useful to implement a function that prints the tree.

1.2 Add Gini index

Implement Gini index as an alternative impurity measure. To use the Gini index, you should call your learning function like `learn(X, y, impurity_measure='gini')`.

1.3 Add reduced-error pruning

ID3 is prone to overfit. To fix this, extend your algorithm to tackle overfitting using reduced-error pruning.

You will need to extend your function's argument list with an additional parameter called `prune` which should by default be set to `False`.

Since pruning should be done inside the `learn` method, the pruning set is a subset of the training set. You can add an additional parameter that specifies which part of the data should be used as a pruning set.

1.4 Evaluate your algorithm

Load the Abalone dataset from MittUiB; alternatively, you can use `https://archive.ics.uci.edu/ml/datasets/Abalone`. Assess the performance of your algorithm using an appropriate performance measure. Which setting should you select for this data (entropy or gini, pruning or no pruning)? What is your estimate for the performance of the selected model on unseen data points? Report how you arrived at the conclusions.

1.5 Compare to an existing implementation

Compare your implementation to some existing decision tree implementation. How does your implementation fare against this implementation in terms of accuracy and speed? Can you explain the (possible) differences?

Note: You can compare to, for example, `DecisionTreeClassifier` from `sklearn`.