Machine Learning Project 1
Predicting Heart Disease

Goal of the Project

The aim of this project is to become more familiar with the notions of underfitting and overfitting, as well as the pipeline to validate a model. To do so, you will solve a simple classification problem using Decision Trees. The next sections of this document will present the data, as well as the different tasks you are expected to perform.

Modalities

After performing the different tasks, you are expected to hand out a report of max. 3 pages, including plots. Additional pages will not be read. When writing your report, keep in mind that we are just waiting for the answers to the different questions we will ask in the next sections (in other words, it is not necessary to give additional information about, e.g., how a model works, what is the dataset, the context, etc.). This report can be written in English or in French, and should be exported in .pdf format.

On the implementation side, you will use Python 3 with the scikit-learn framework (`http://scikit-learn.org`), which provides many tools for machine learning and has a detailed documentation. Please check the instructions in `https://scikit-learn.org/stable/install.html` to install scikit-learn. Your code should be submitted with your report on Webcampus by providing a notebook or a script (.ipynd or .py file(s)).

!!! Please, carefully respect the aforementioned instructions. Due to the high number of reports to evaluate, failure to comply with at least one of the previous instructions will result in a grade of $0/20$ **!!!**

The dataset

The aim of this project is to build a Decision Tree Classifier[1] in order to discriminate 2 different types of patients - those with heart disease (label 2), and those without (label 1) - thanks to several medical features. For each instance, 12 features are available, which are the (i) age, (ii) sex, (iii) chest pain type with 4 possibilities, (iv) resting blood pressure, (v) serum cholesterol in mg/dl, (vi) fasting blood sugar $>$ 120 mg/dl, (vii) resting electrocardiographic results with values 0,1,2, (viii) maximum heart rate achieved, (ix) exercise induced angina, (x) oldpeak = ST depression induced by exercise relative to rest, (xi) the slope of the peak exercise ST segment, and (xii) number of major vessels (0-3) colored by fluoroscopy.

Attached to this project description, you will find two data files named `train.txt` and `test.txt`. The first one constitutes your available training data. It is recommended to load the data using the `loadtxt()` function from numpy[2]. The following python code presents an example of how to load the data in a scikit-learn-ready fashion:

```
import numpy as np
data = np.loadtxt("./train.txt")
X, y = data[:,:-1], data[:,-1]
```

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`
[2] `https://numpy.org/doc/stable/reference/generated/numpy.loadtxt.html`

The second file, `test.txt` only contains additional instances without their corresponding labels (the file only contains 12 columns). In other words, those data cannot be used in your training pipeline and will only be used at the end of the project to validate your model(s) by submitting it in an online leader board (more details in the last section).

Task 0 - Prepare the data

First of all, you will need to split the data in a training and testing set in order to validate your model (at this stage, you should not use the `test.txt` file). To do so, we recommend to use the `train_test_split()` function from scikit-learn[3] with a `test_size` of 0.33 and a `random_state` of 42.

For this task, you just have to split your data in a training and testing set. You will use those same splits all along this project. You do not have to comment this task in your final report.

Task 1 - Train your first Decision Tree

Your first task will be to train and evaluate the performance of a Decision Tree Classifier with the following meta-parameters (see the documentation for more details):

- criterion = "gini"
- max_depth = 1
- min_samples_split = 2
- min_samples_leaf = 1
- max_leaf_nodes = None
- random_state = 42

A companion library (`utils.py`) is also attached to this project. It contains several functions to make your life easier. For example, in order to draw the decision trees, you will have to use the `plot_tree()` function from scikit-learn. However, this function requires the feature names. Those feature names can be easily retrieved by a function called `get_features()` provided in the companion library.

Train the proposed decision tree classifier. Report the decision tree you obtained, as well as the training and testing accuracy. Is your model presenting evidence of under or overfitting? Justify. Discuss the pros and cons of using this decision tree, if any.

Task 2 - Train another Decision Tree

Your second task will be to train and evaluate the performance of a Decision Tree Classifier with the following meta-parameters:

- criterion = "gini"
- max_depth = 10
- min_samples_split = 2
- min_samples_leaf = 1
- max_leaf_nodes = None
- random_state = 42

Train the proposed decision tree classifier. Report the decision tree you obtained, as well as the training and testing accuracy. Is your model presenting evidence of under or overfitting? Justify. Discuss the pros and cons of using this decision tree, if any.

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

Task 3 - Find the best Decision Tree

In this task, you will now try to find the best decision tree. To do so, you will first only try different values between 1 and 10 for the `max_depth` meta-parameter, and report a graph of the training/testing accuracy w.r.t. the corresponding choice for `max_depth`. After that, you are free to play with the other meta-parameters if you want. For each experiment, leave `random_state` = 42.

Discuss this graph by making links with the theory. Spot some choices for `max_depth` that lead to under and overfitting. What would be a good choice for `max_depth`? Justify. Briefly comment the experiments you made if you played with other meta-parameters.

Task 4 - Join the leaderboard!

The final task will be to submit your model into a private Kaggle competition. To do so, we will ask you to first make a Kaggle account (see `https://www.kaggle.com/`). If you do not use your real name when registering to Kaggle, please provide in your report the pseudo you used to make your submissions.

In order to make a submission in the competition, you will need to generate a submission file in a .csv format. This file will contain your predictions for the instances in the `test.txt` data file. As the true labels are known by Kaggle (but not by you), this submission file will allow Kaggle to evaluate the performance of your model, and to rank it in a leaderboard. The required submission file can easily be generated by using the `generateSubmission()` function from the companion library. You are not allowed to use other models than decision trees. Furthermore, you are not allowed to use other external data to train your model(s).

The Kaggle competition for this project can be accessed using the following link: `https://www.kaggle.com/t/a194ff78699e4aaf806abf639d375a47`. Have fun!

For this task, it is mandatory to perform at least one submission to the Kaggle competition (if you use a pseudo, please provide it in your report). Note that your final rank will not be used to evaluate the quality of the project.

Task 5 - Reflect on your Work

Aside from the fun of games and competitions, why did we maintain a separate `test.txt` file? What does your Kaggle score actually represent, and why might it differ from the accuracy you observed on your own test set? Reflect and comment regarding the theory.