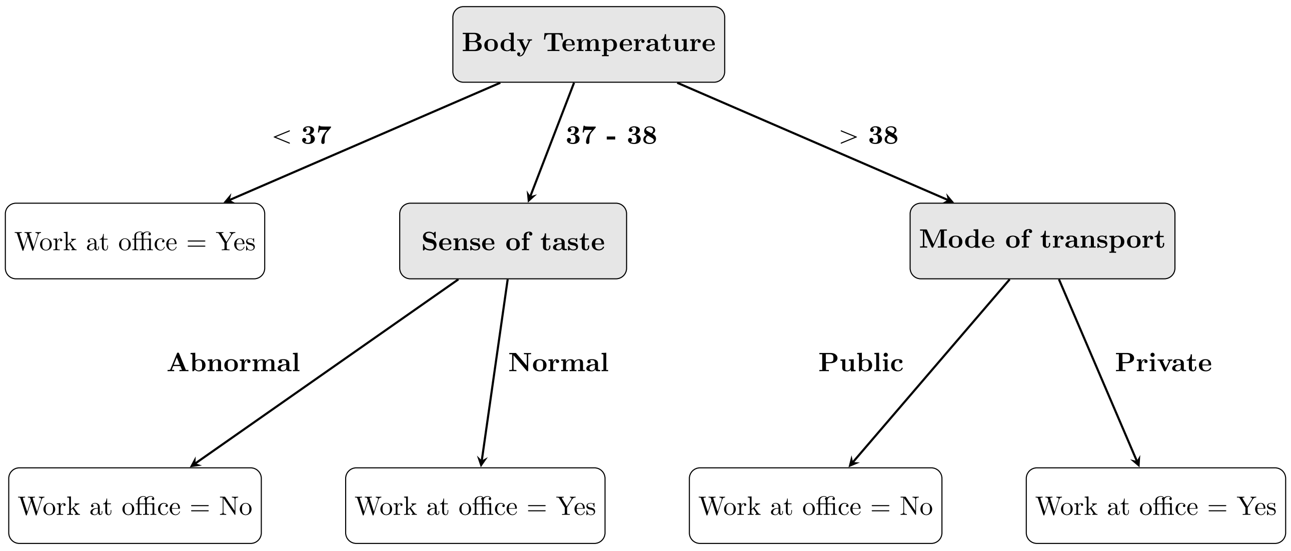
TIF150 / FIM780 Information theory for complex systems
Homework 1

Student: João Carlos Pereira Fernandes
jooc@chalmers.se

For this homework it was a required to implement the ID3 algorithm and apply it to a concrete dataset in order to obtain a decision tree. The implementation of the code was done in Python and the obtained tree is shown below:



1. Decision tree based on ID3 algorithm

This tree will predict correctly every single observation in this dataset. In other words, the entropy of this tree 0, as we wanted, which means there is no 'uncertainty' when classifying an observation.

Let's walk through the code and understand in detail what it does:

The code starts by listing the features/attributes and the target variable. We can immediately calculate the initial entropy of the dataset:

```
Initial entropy of the dataset = 0.940285958670631
```

From there it loops over all attributes in order to find the best root node for the decision tree. For each selected attribute, the program splits the original dataset into smaller ones containing the observations where it matches the value of the attribute, for each possible value it can take. Then, it calculates the total entropy S given the attribute A using the following formula:

$$\mathrm{H}(S|A) = \sum_{t \in v_A} p(t) \sum_{y \in v_S} s(y) \log_2 \frac{1}{s(y)}$$

where v_A and v_S represent the possible values for the attribute A and target, respectively, $p(t)$ denotes the proportion of values in the dataset that match the value t of attribute A and $s(y)$ the proportion of values (from those already filtered) that match the value y of the target variable. Then, the information gain is computed as the initial entropy minus the entropy knowing attribute A, that is, $\mathrm{IG} = \mathrm{H}(S) - \mathrm{H}(S|A)$ and thus the attribute chosen is that which has the smallest conditional entropy. Looking at the prints throughout the code:

```
H( Work at office  |  Body Temperature ) = 0.6935361388961919
H( Work at office  | Crowd at Office ) = 0.9110633930116763
H( Work at office  | Sense of taste ) = 0.7884504573082894
H( Work at office  | Mode of transport ) = 0.8921589282623614
```

So we can conclude that the root node for the decision tree will be the 'Body Temperature' variable. We can calculate again the total entropy of the tree after having chosen this attribute:

```
Chosen attribute (node):  Body Temperature
Total tree entropy: 0.6935361388961919
```

Then, we start creating the 'tree' structure. The tree is encoded through a dictionary whose keys represent ID's (integers) and its values give the possible paths in the form of a list of tuples in the form ('Attribute', 'Value'). Every time a new attribute is chosen, the original branch where it's being appended is removed from the dictionary, and newer ones already containing the ramifications from the new attribute are appended.

For the following attributes a similar thought process follows. However, keep in mind that now when using the formula above our 'target' entropy is now the entropy from the clipped dataset resulting from the observations that match each value of our ramification. This means it will be different for every branch, and we must loop all attributes over all branches in order to determine the optimal one. For the second attribute, we get:

```
H( Work at office  |  Body Temperature =  > 38, Crowd at Office ) = 0.9509775004326937
H( Work at office  |  Body Temperature =  > 38, Sense of taste ) = 0.9509775004326937
H( Work at office  |  Body Temperature =  > 38, Mode of transport ) = 0.0
H( Work at office  |  Body Temperature =  37 - 38, Crowd at Office ) = 0.4
H( Work at office  |  Body Temperature =  37 - 38, Sense of taste ) = 0.0
H( Work at office  |  Body Temperature =  37 - 38, Mode of transport ) = 0.9509775004326937
H( Work at office  |  Body Temperature =  < 37 ) = 0

Chosen attribute: Sense of taste
Total tree entropy: 0.34676806944809596
```

Note that the entropy for the branch where 'Body Temperature = < 37' already has an entropy of zero! The code is built to not iterate over these cases since this means the classification of target variable is unambiguous and we don't need further attributes for the classification. This path then gets removed from the tree dictionary (in order to avoid redundant computations) and added onto our output variable, a dictionary which keys contain the path and the value contains the value of the target variable when following that path.

The code keeps iterating until all attributes have been used or until the entropy of the tree reaches 0, from where no further attributes are required for the classifcation. The final prints and output of the code are:

```
H( Work at office  |  Body Temperature =  > 38, Crowd at Office ) = 0.9509775004326937
H( Work at office  |  Body Temperature =  > 38, Mode of transport ) = 0.0
H( Work at office  |  Body Temperature =  37 - 38, Sense of taste = Abnormal ) = 0
H( Work at office  |  Body Temperature =  37 - 38, Sense of taste = Normal ) = 0

Chosen attribute: Mode of transport
Total tree entropy: 0.0

H( Work at office  |  Body Temperature =  37 - 38, Sense of taste = Abnormal ) = 0
H( Work at office  |  Body Temperature =  > 38, Mode of transport = Public ) = 0
H( Work at office  |  Body Temperature =  > 38, Mode of transport = Private ) = 0

Output:

{"[(' Body Temperature', ' < 37')]": 'Work at office = Yes ',
 "[(' Body Temperature', ' 37 - 38'), ('Sense of taste', 'Abnormal')]": 'Work at office = No ',
 "[(' Body Temperature', ' 37 - 38'), ('Sense of taste', 'Normal')]": 'Work at office = Yes ',
 "[(' Body Temperature', ' > 38'), ('Mode of transport', 'Public')]": 'Work at office = No ',
 "[(' Body Temperature', ' > 38'), ('Mode of transport', 'Private')]": 'Work at office = Yes '}
```

Note that for this dataset it wasn't required the 'Crowd at Office' attribute. The algorithm found a way to correctly classify all observations using fewer attributes than what the dataset presented.

The complete Python code for this homework can be found here.