Information Theory for Complex Systems
Homework 1

Vincent Udén
udenv@student.chalmers.se

Januari - 2023

Below follows my solution for determining the optimal order of attributes using the ID3 algorithm implemented in Python. Before running any code I santised the csv file manually to remove trailing and leading whitespace. The code is attached at the end of the report and is availible on https://github.com/vincent-uden/tif150-information-theory-for-complex-systems.

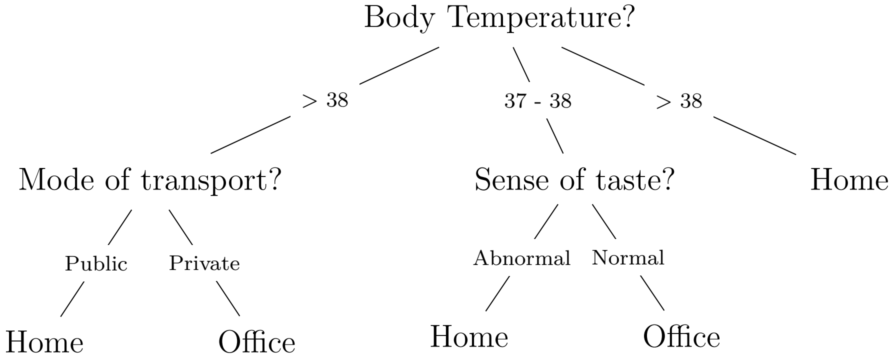

Figure 1: The sequence of optimal attributes for separating individuals working from home or at the office.

Figure 1 shows the optimal attribute order which maximises the information gained at each observation. It also shows the values of the chosen attributes along the edges of the tree. Entropy values are presented below in table 1.

Table 1: Shannon entropy at progressing stages of the ID3 algorithm.

Attribute choice	$H(S\|A)$ at the step	$H(t)$ after step
Start (before any choice)	-	4.39
Body Temperature (> 38)	1.18	1.65
Body Temperature (37 - 38)	1.18	1.65
Body Temperature (< 37)	1.18	0.0
Mode of tranport (Public)	0.0	0.0
Mode of tranport (Private)	0.0	0.0
Sense of taste (Abnormal)	0.0	0.0
Sense of taste (Normal)	0.0	0.0

main.py

```
#!/usr/bin/python

import pandas as pd
import numpy as np

from collections import defaultdict
from typing import List, Tuple

def target_entropy(data: pd.DataFrame):
    t_yes = 0
    t_no = 0
    for r in data[”Work at office”]:
        if r == ”Yes”:
            t_yes += 1
        else:
            t_no += 1

    p_yes = float(t_yes) / (t_yes + t_no)
    p_no = float(t_no) / (t_yes + t_no)

    t_entropy = 0
    for r in data[”Work at office”]:
        if r == ”Yes”:
            t_entropy += p_yes * np.log(1/p_yes)
        else:
            t_entropy += p_no * np.log(1/p_no)
    return t_entropy

def col_entropy(data: pd.DataFrame, t_entropy: float, col: str):
    subset_totals = defaultdict(lambda: 0)
    subset_yes = defaultdict(lambda: 0)

    for i, r in zip(data.index, data[col]):
        subset_totals[r] += 1
        if data[”Work at office”][i] == ”Yes”:
            subset_yes[r] += 1

    total = 0
    for v in subset_totals.values():
        total += v

    entropy = 0
    for k, v in subset_totals.items():
        p_yes = float(subset_yes[k]) / v
        p_no = float(v - subset_yes[k]) / v

        if p_yes == 0:
            H_yes = 0
        else:
            H_yes = subset_yes[k] * p_yes * np.log(1/p_yes)

        if p_no == 0:
            H_no = 0
        else:
            H_no = (v - subset_yes[k]) * p_no * np.log(1/p_no)

        H_t =  H_yes + H_no
        entropy +=  float(v) / total * H_t

    return entropy

def generate_new_data_slices(orig_data: pd.DataFrame, data: pd.DataFrame, col: str):
    possible_values = []
    new_frames = []

    for r in data[col]:
        if not r in possible_values:
            possible_values.append(r)

    for v in possible_values:
        new_frames.append(data[data[col] == v])

    return (possible_values, new_frames)

def eval_data_frames(orig_data: pd.DataFrame, frames: Tuple[str, List[pd.DataFrame]],
   indent_depth=0) -> List[str]:
    question_order = []
    for (name, f) in frames:
        print(”\n” + ” ” * indent_depth + name)
        t_entropy = target_entropy(f)
        print(” ” * indent_depth + f”H(t):{t_entropy}”)
        if t_entropy == 0.0:
            question_order.append(f”{name} - DONE”)
            continue



        col_entropies = []
        for col in f.columns[:-1]:
            col_entropies.append(col_entropy(f, t_entropy, col))

        col_entropies = np.array(col_entropies)

        max_info = np.argmax(t_entropy - col_entropies)
        question_order.append(f”{name} - {f.columns[max_info]}”)
        print(” ” * indent_depth + f”H(S|A):{col_entropies[max_info]}”)

        values, new_frames = generate_new_data_slices(data, f, f.columns[max_info])

        question_order.append(eval_data_frames(orig_data, zip(values, new_frames),
           indent_depth+2))
    return question_order

if __name__ == ”__main__”:
    data = pd.read_csv(”./id3_data-1.csv”)
    t_entropy = target_entropy(data)
    question_order = eval_data_frames(data, [(”Root”, data)])

    print(question_order)
```