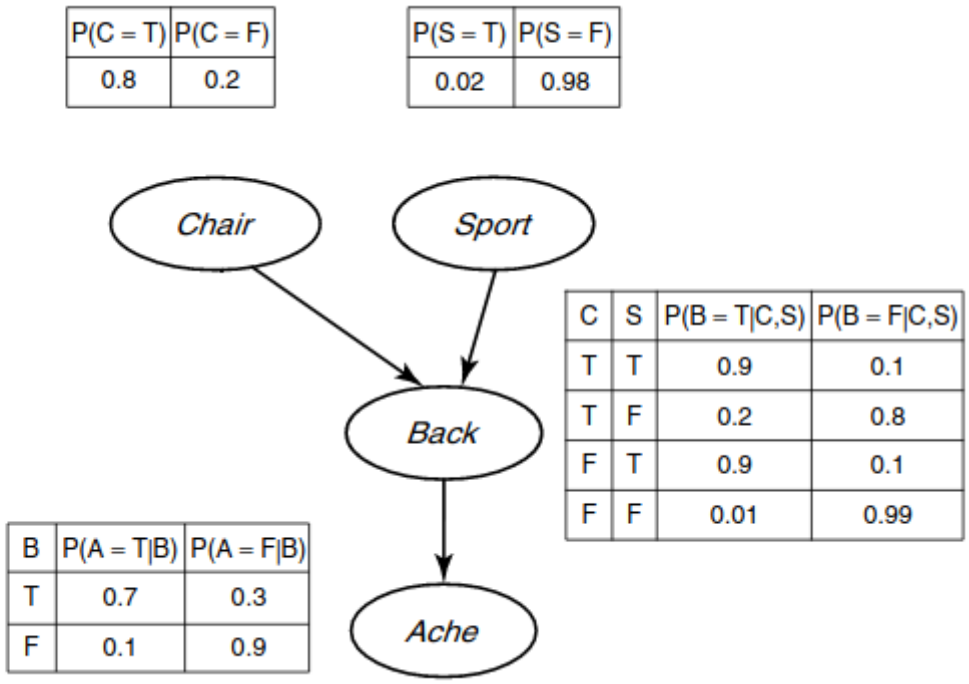
Ćw. 7 - Sieci Bayesowskie. Aleksander Drwal.

Tematem zadania jest implementacja losowego generatora danych działającego zgodnie z rozkładem reprezentowanym przez poniższą sieć bayesowską



Implementacja

Sieć jest wczytywana z pliku `.json`. Dla tej konkretnej sieci ma on poniższą strukturę.

```
[
    {
        "name": "Chair",
        "parents": [],
        "probabilities": [
            {
                "parentsValues": [],
                "value": 0.8
            }
        ]
    },
    {
        "name": "Sport",
        "parents": [],
        "probabilities": [
            {
                "parentsValues": [],
```

```
                "value": 0.02
            }
        ]
    },
    {
        "name": "Back",
        "parents": [
            "Chair",
            "Sport"
        ],
        "probabilities": [
            {
                "parentsValues": [
                    true,
                    true
                ],
                "value": 0.90
            },
            {
                "parentsValues": [
                    true,
                    false
                ],
                "value": 0.2
            },
            {
                "parentsValues": [
                    false,
                    true
                ],
                "value": 0.9
            },
            {
                "parentsValues": [
                    false,
                    false
                ],
                "value": 0.01
            }
        ]
    },
    {
        "name": "Ache",
        "parents": [
            "Back"
        ],
        "probabilities": [
```

```
            {
                "parentsValues": [
                    true
                ],
                "value": 0.7
            },
            {
                "parentsValues": [
                    false
                ],
                "value": 0.1
            }
        ]
    }
]
```

Algorytm jest uruchamiany jako aplikacja konsolowa:

```
usage: main.py [-h] [--count COUNT] [--output OUTPUT] [--col-headers]
               network_data_path
```

Wyniki

Dane wygenerowane przez sieć są używane do trenowania drzewa ID3 z ćwiczenia nr. 4.
Dla zestawu danych składającego się z 1000 losowo wygenerowanych zbiorów, algorytm osiąga następujące wyniki:

$$Macierz\ pomyłek \approx \begin{bmatrix} 294.55 & 36.3 \\ 23.45 & 46.85 \end{bmatrix}$$

$$Dokładność \approx 85\%$$

$$Czułość \approx 92.6\%$$

$$Swoistość \approx 55\%$$

$$Precyzja \approx 89\%$$

Dla porównania w poprzednim zadaniu dla zbiorów Breast cancer i mushroom, wyniki były następujące:

Breast cancer:

$$Macierz\ pomyłek \approx \begin{bmatrix} 10.7 & 17.25 \\ 21.35 & 63.2 \end{bmatrix}$$

$$Dokładność \approx 66\%$$

$$Czułość \approx 79\%$$

$$Swoistość \approx 33\%$$

$$Precyzja \approx 72\%$$

mushroom:

$$Macierz\ pomyłek \approx \begin{bmatrix} 1672.4 & 0 \\ 0 & 1555.6 \end{bmatrix}$$

$$Dokładność \approx 100\%$$

$$Czułość \approx 100\%$$

$$Swoistość \approx 100\%$$

$$Precyzja \approx 100\%$$

Obserwacje

Algorytm radzi sobie lepiej niż w przypadku zbioru **Breast Cancer**, natomiast nie jest w stanie w pełni odwzorować zbioru jak jest to w przypadku danych z zestawu **mushroom**.

Może to być spowodane tym, że taki sam zbiór danych nie oznacza zawsze takiego samego wyniku. Np. istnieje możliwość, że dla `chair=true sport=true back=true` wynikiem będzie `ache=true` lub `ache=false`.

Oprócz tego, szansa na to, że `ache=true` jest prawie 5 razy mniejsza niż `ache=true`. Z tego powodu algorytm może być bardziej skłonny do przypisania wartości `ache=false` w miejscach gdzie nie powinien tego zrobić.
Taki stan rzeczy widać w wynikach, gdzie `ache=true` jest klasą negatywną a `ache=false` klasą pozytywną. Przy takim stanie rzeczy mamy:

$$\frac{FP}{TP+FP} \approx 0.11\ ,\ \frac{FN}{TN+FN} \approx 0.33$$

Czyli wartości z klasy negatywnej są niepoprawnie zgadywane 3 razy częściej niż te z pozytywnej.