Blatt8_Olbrich,Rotgeri

June 29, 2020

1 Aufgabe 16 Naive Bayes: FuSSball

a) Beweisen Sie den Satz von Bayes (1) mit Hilfe der Definition der bedingten Wahrscheinlichkeit.

$$P(F|W) = \frac{P(F \cap W)}{P(W)} = \frac{\frac{P(F \cap W)}{P(F)} \cdot P(F)}{P(W)} = \frac{P(W|F) \cdot P(F)}{P(W)}$$

b) Die Wetterbedingungen des heutigen Tages finden Sie in Tabelle 2. Wie hoch ist die Wahrschein- lichkeit, dass heute FuSSball gespielt wird?

$$P(W|F) = P_{starkerWind,ja} \cdot P_{kalteTemp,ja} \cdot P_{sonnigerAusblick,ja} \cdot P_{hoheFeuchte,ja}$$
$$= \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{2}{9} \cdot \frac{1}{3} = \frac{2}{243}$$
$$P(F) = \frac{9}{14}$$
$$P(W) = P(W|F = ja) \cdot P(F = ja) + P(W|F = nein) \cdot P(F = nein)$$
$$P(W|F_{nein}) = \frac{3}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} \cdot \frac{1}{5} = \frac{36}{625}$$
$$P(W) = \frac{2}{243} \cdot \frac{9}{14} + \frac{36}{625} \cdot \frac{5}{14} = 0,026$$
$$P(F|W) = \frac{P(W|F) \cdot P(F)}{P(W)} = 0,204$$

c) Nehmen Sie an, Sie sollten nun berechnen, wie hoch die Wahrscheinlichkeit ist morgen FuSSball zu spielen (Wetterbedingungen s. Tabelle 3). Welches Problem tritt auf und wie kann man es lösen?

Bei heiSSen Temperaturen wurde noch nie FuSSball gespielt. Daher ergibt sich die Wahrscheinlichkeit berechenet analog zu b von 0. Um dieses Problem zu lösen könnte man die Temperatur nicht mit in die Berechnung einflieSSen lassen. Am schönsten wäre natürlich ein gröSSerer Datensatz.

2 Aufgabe 17

Sie haben einen Datensatz wie er in Tabelle 4 gegeben ist. Hierbei ist Temperatur: Temperatur in Grad Celsius. Wettervorhersage: Wetterqualität (0: schlecht , 1: normal, 2: gut). Luftfeuchtigkeit:

Luftfeuchtigkeit in Prozent. Wind: Aussage, ob es gerade windig ist. FuSSball: Lohnt es sich FuSSball spielen zu gehen? Hierbei ist das Zielattribut, welches man bestimmen will, die Entscheidung, ob es sich lohnt FuSSball spielen zu gehen. In dieser Aufgabe sollen Sie zu diesem Zweck den ersten Schnitt eines binären Entscheidungsbaumes nachvollziehen.

(a) Berechnen Sie per Hand die Entropie der Wurzel (des Baumes).

In [7]:
```
import matplotlib.pyplot as plt
import numpy as np
#import pandas as pd
from scipy.optimize import curve_fit
#from uncertainties import ufloat
from scipy.stats import stats
```

Data.txt sieht wie folgt aus:
Temperatur Wettervorhersage Luftfeuchtigkeit Wind Fussball
29.4 2 85 0 0
26.7 2 90 1 0
28.3 1 78 0 1
21.1 0 96 0 1
20 0 80 0 1
18.3 0 70 1 0
17.8 1 65 1 1
22.2 2 95 0 0
20.6 2 70 0 1
23.9 0 80 0 1
23.9 2 70 1 1
22.2 1 90 1 1
27.2 1 75 0 1
21.7 0 80 1 0

In [2]:
```
Temperatur, Wetter, Luftfeuchtigkeit, Wind, Fuball = np.genfromtxt('Data.txt', unpack='
```

In [3]:
```
def get_Entropie(p_1, p_2):
    if p_1==0:
        S = -p_2*np.log2(p_2)
    elif p_2==0:
        S = -p_1*np.log2(p_1)
    else:
        S = -(p_1*np.log2(p_1) + p_2*np.log2(p_2))
    return S


# Entropie berechnen:
p_Fuball_False = len(Fuball[Fuball==False])/len(Fuball)
p_Fuball_True = 1- p_Fuball_False
S = get_Entropie(p_Fuball_False, p_Fuball_True)
print(S)
```

```
0.9402859586706311
```

Die Entropie wird mit folgender Formel berechnet:

$$\begin{aligned} S &= -\sum_i p_i \log_2(p_i) \\ &= -\left[P(\text{FuSSball=True}) \log_2(P(\text{FuSSball=True})) + P(\text{FuSSball=False}) \log_2(P(\text{FuSSball=False}))\right] \\ &= 0,9402 \end{aligned}$$

Mit

$$\begin{aligned} P(\text{FuSSball=True}) &= \frac{9}{14} \\ P(\text{FuSSball=False}) &= \frac{5}{14} \end{aligned}$$

b) Berechnen Sie per Hand den Informationsgewinn, falls ein Schnitt auf dem Attribut Wind durchgeführt wird.

```
In [4]: P_Wind_false = len(Wind[Wind==False])/len(Wind)
        P_Wind_true =1-P_Wind_false
        print(len(np.logical_and(Wind==True, Fuball==True)))
        P_Windtrue_fuballtrue = len(Wind[np.logical_and(Wind==True, Fuball==True)])/len(Wind[W
        P_Windtrue_fuballfalse = len(Wind[np.logical_and(Wind==True, Fuball==False)])/len(Wind
        P_Windfalse_fuballfalse = len(Wind[np.logical_and(Wind==False, Fuball==False)])/len(Wi
        P_Windfalse_fuballtrue = len(Wind[np.logical_and(Wind==False, Fuball==True)])/len(Wind
        print(P_Windtrue_fuballtrue, P_Windtrue_fuballfalse, P_Windfalse_fuballfalse, P_Windfal


        IG=S-(len(Wind[Wind==True])/14) * (get_Entropie(P_Windtrue_fuballfalse, P_Windtrue_fuba
        print(IG)
```

```
14
0.5 0.5 0.25 0.75
0.04812703040826949
```

Der Informationsgewinn vom Attribut Wind wird mit folgender Formel berechnet:

$$\begin{aligned} IG(Y|X) &= H(Y) - H(Y|X) \\ IG(Wind|Fuball) &= S - \frac{D_{\text{Wind=True}}}{D} \cdot H(D_{\text{Wind=True}}) - \frac{D_{\text{Wind=False}}}{D} \cdot H(D_{\text{Wind=False}}) \\ &= 0,04860 \end{aligned}$$

Hierbei ist S die in a) berechnete Entropie, $D_{\text{Wind=True}}$ die Anzahl der Wind-Einträge mit "True", $D_{\text{Wind=False}}$ die Anzahl der Wind-Einträge mit "False" und $D = 14$ die Gesamtanzahl der Einträge.

H(X) bezeichnen die Entropien und werden wie in a) ausgerechnet. Die benötigten bedingten Wahrscheinlichkeiten zur Berechnung der Entropie lauten:

$$P(\text{FuSSball=True} \mid \text{Wind=True}) = \frac{1}{2}$$
$$P(\text{FuSSball=False} \mid \text{Wind=True}) = \frac{1}{2}$$
$$P(\text{FuSSball=True} \mid \text{Wind=False}) = \frac{3}{4}$$
$$P(\text{FuSSball=False} \mid \text{Wind=False}) = \frac{1}{4}$$

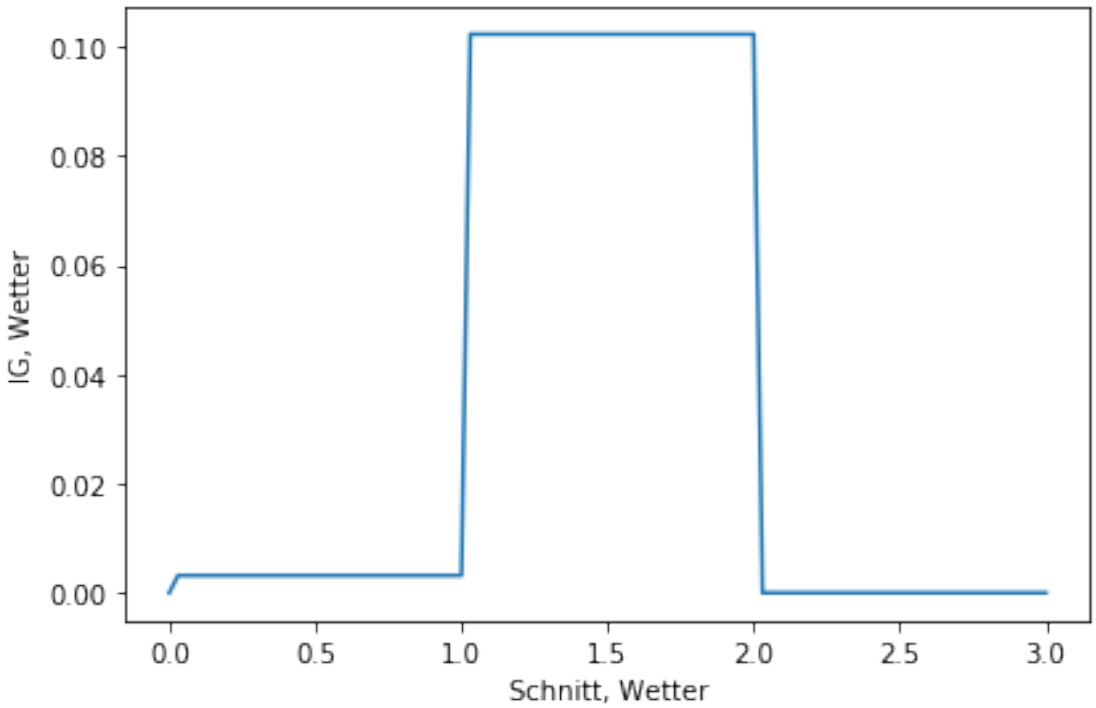
c) Berechnen Sie für die verbleibenden Attribute den Informationsgewinn in Abhängigkeit von verschiedenen Schnitten und plotten Sie den Informationsgewinn in Abhängigkeit der jeweiligen Schnitte

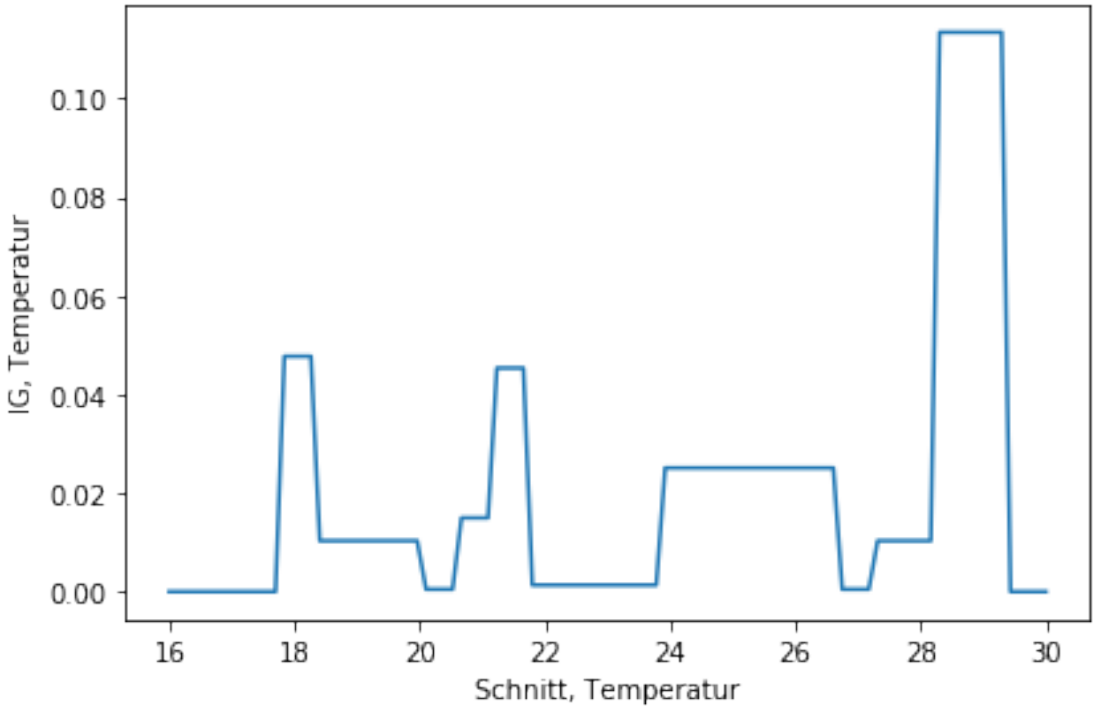
```
In [5]: def get_IG(D, Schnitt):
            IG = np.zeros(len(Schnitt))


            for l in range(len(Schnitt)):
                D_u =  len(D[D < Schnitt[l]]) # Anzahl der Werte unter dem Schnitt
                D_ü = len(D)-D_u # Anzahl der Werte über dem Schnitt


                if D_u==0 or D_u==len(D):
                    IG[l]=0 # Falls alle Werte unter oder über dem Schnitt sind, ist
                            # der Informationsgewinn 0
                else:
                    # Berechnung der bedingten Wahrscheinlichkeiten p_unterSchnitt:
                    Anzahl_unterSchnitt_False = 0


                    for i in range(len(D)):
                        if D[i]<Schnitt[l] and Fuball[i]==False:
                            Anzahl_unterSchnitt_False = Anzahl_unterSchnitt_False+1


                    p_unterSchnitt_False = Anzahl_unterSchnitt_False/D_u
                    p_unterSchnitt_True = 1-p_unterSchnitt_False
                    S_1 = get_Entropie(p_unterSchnitt_False, p_unterSchnitt_True)

                    # Berechnung der Wahrscheinlichkeit p_überSchnitt:
                    Anzahl_überSchnitt_False = 0


                    for i in range(len(D)):
                        if D[i]>=Schnitt[l] and Fuball[i]==False:
```

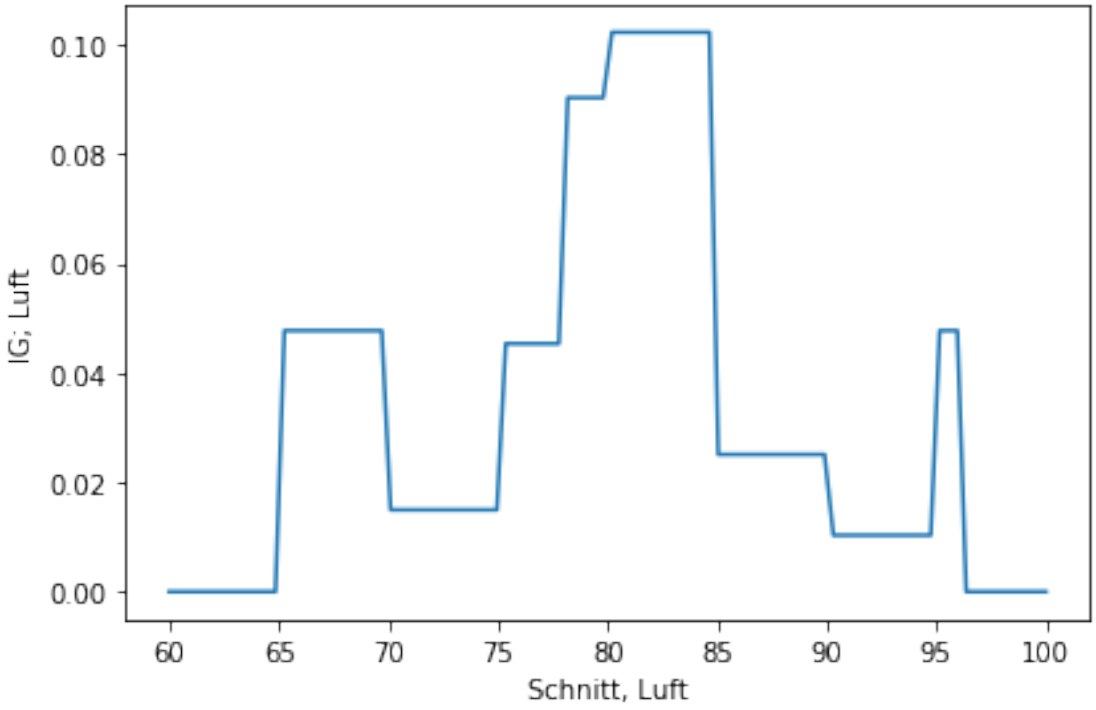
```
                    Anzahl_überSchnitt_False = Anzahl_überSchnitt_False+1


            p_überSchnitt_False=Anzahl_überSchnitt_False/D_ü
            p_überSchnitt_True = 1-p_überSchnitt_False
            S_2 = get_Entropie(p_überSchnitt_False, p_überSchnitt_True)

            IG[l] = S - D_u/len(D)*S_1 - D_ü/len(D)*S_2

    return IG
# IG Wetter
#Schnitt_Wetter = np.array([0,1,2,3])
Schnitt_Wetter = np.linspace(0, 3, 100)
IG_Wetter = get_IG(Wetter, Schnitt_Wetter)
print('WETTER:')
print('Bester Schnitt: ', Schnitt_Wetter[np.argmax(IG_Wetter)])
print('Bester IG: ', np.max(IG_Wetter))

# IG Temperatur:
Schnitt_Temperatur = np.linspace(16, 30, 100)
IG_Temperatur = get_IG(Temperatur, Schnitt_Temperatur)
print('TEMPERATUR:')
print('Bester Schnitt: ', Schnitt_Temperatur[np.argmax(IG_Temperatur)])
print('Bester IG: ', np.max(IG_Temperatur))

# IG Luftfeuchtigkeit
Schnitt_Luft = np.linspace(60, 100, 100)
IG_Luft = get_IG(Luftfeuchtigkeit, Schnitt_Luft)
print('LUFTFEUCHTIGKEIT:')
print('Bester Schnitt: ', Schnitt_Luft[np.argmax(IG_Luft)])
print('Bester IG: ', np.max(IG_Luft))

# Plot: Wetter
plt.plot(Schnitt_Wetter, IG_Wetter)
plt.xlabel('Schnitt, Wetter')
plt.ylabel('IG, Wetter')
plt.tight_layout()
plt.savefig('IG_Wetter.pdf')
plt.show()
plt.clf()

# Plot: Temperatur
plt.plot(Schnitt_Temperatur, IG_Temperatur)
plt.xlabel('Schnitt, Temperatur')
plt.ylabel('IG, Temperatur')
plt.tight_layout()
plt.savefig('IG_Temperatur.pdf')
plt.show()
```

```
        plt.clf()

        # Plot: Luftfeuchtigkeit
        plt.plot(Schnitt_Luft, IG_Luft)
        plt.xlabel('Schnitt, Luft')
        plt.ylabel('IG; Luft')
        plt.tight_layout()
        plt.savefig('IG_Luft.pdf')
        plt.show()
        plt.clf()
```

```
WETTER:
Bester Schnitt:  1.0303030303030303
Bester IG:  0.10224356360985076
TEMPERATUR:
Bester Schnitt:  28.303030303030305
Bester IG:  0.1134008641811034
LUFTFEUCHTIGKEIT:
Bester Schnitt:  80.20202020202021
Bester IG:  0.10224356360985076
```




0.10
0.08
0.06
0.04
0.02
0.00
IG, Temperatur
16
18
20
22
24
26
28
30
Schnitt, Temperatur


0.10
0.08
0.06
0.04
0.02
0.00
IG; Luft
60
65
70
75
80
85
90
95
100
Schnitt, Luft

```
<Figure size 432x288 with 0 Axes>
```

(d) Welches Attribut eignet sich am besten zum Trennen der Daten?

Am besten eignet sich das Attribut Temperatur zum Trennen der Daten, da hier beim besten Schnitt (28,3 Grad) der Informationsgewinn am gröSSten ist. Bei den anderen Attributen ist des maximale Informationsgewinn geringer.

In [ ]: