aufgabe2

November 29, 2018

1 Aufgabe 16

1.1 Teilaufgabe a)

Die Entropie der Wurzel lässt sich nach der Formel

$$H(Y) = -\sum_{z \in Z} P(Y = z) log_2 P(Y = z)$$

berechnen. Dabei sind die möglichen Ereignisse "FuSSball spielen" bzw. "kein FuSSball spielen". Es ergibt sich daher:

$$H(Fuball) = -\left(\frac{n_{Fuball}}{n} log_2 \frac{n_{Fuball}}{n} + \frac{n_{keinFuball}}{n} log_2 \frac{n_{keinFuball}}{n}\right)$$

wobei n die Anzahl der überprüften Fälle ist.

Berechnung:

```
In [1]: import math
        import numpy as np
        import matplotlib.pyplot as plt
```

```
In [2]: p_f = 9/14
        p_kf = 5/14

        H_f = -(p_f*math.log(p_f,2) + p_kf*math.log(p_kf,2))

        print('Wert der Entropie:')
        print(H_f)
```

```
Wert der Entropie:
0.9402859586706309
```

1.2 Teilaufgabe b)

Zur Berechnung des Informationsgewinns wird die Entropie nach einem Schnitt auf dem Attribut Wind benötigt. Die Entropie nach einem Schnitt auf irgendeinem Attribut wird allgemein berechnet nach

$$
\begin{aligned}
H(Y|X) &= \sum_{m \in M} P(X=m) H(Y|X=m) \\
&= -\sum_{m \in M} P(X=m) \sum_{z \in Z} P(Y=z|X=m) log_2 P(Y=z|X=m)
\end{aligned}
$$

in unserem Fall also

$$H(Fuball|X) = \frac{n_{Wind}}{n} H(Fuball|Wind) + \frac{n_{keinWind}}{n} H(keinFuball|Wind)$$

wobei

$$H(Fuball|Wind) = -\left(\frac{n_{Fuball,Wind}}{n_{Wind}} log_2 \frac{n_{Fuball,Wind}}{n_{Wind}} + \frac{n_{keinFuball,Wind}}{n_{Wind}} log_2 \frac{n_{keinFuball,Wind}}{n_{Wind}} \right)$$

$$H(Fuball|keinWind) = -\left(\frac{n_{Fuball,keinWind}}{n_{keinWind}} log_2 \frac{n_{Fuball,keinWind}}{n_{keinWind}} + \frac{n_{keinFuball,keinWind}}{n_{keinWind}} log_2 \frac{n_{keinFuball,keinWind}}{n_{keinWind}} \right)$$

Berechnung:

In [3]:
```
p_w_f = 3/6
p_w_kf = 3/6
p_kw_f = 6/8
p_kw_kf = 2/8
p_w = 6/14
p_kw = 8/14

H_w_f = -p_w*(p_w_f*math.log(p_w_f,2) + p_w_kf*math.log(p_w_kf,2)) - p_kw*(p_kw_f*math
print('Wert der Entropie nach Schnitt auf Attribut Wind:')
print(H_w_f)
```

```
Wert der Entropie nach Schnitt auf Attribut Wind:
0.8921589282623617
```

Der Informationsgewinn lässt sich dann berechnen aus

$$IG(Fuball, Wind) = H(Fuball) - H(Fuball|X)$$

Berechnung:

In [4]:
```
IG = H_f - H_w_f
print('Informationsgewinn nach Schnitt auf Attribut Wind:')
print(IG)
```

```
Informationsgewinn nach Schnitt auf Attribut Wind:
0.04812703040826927
```

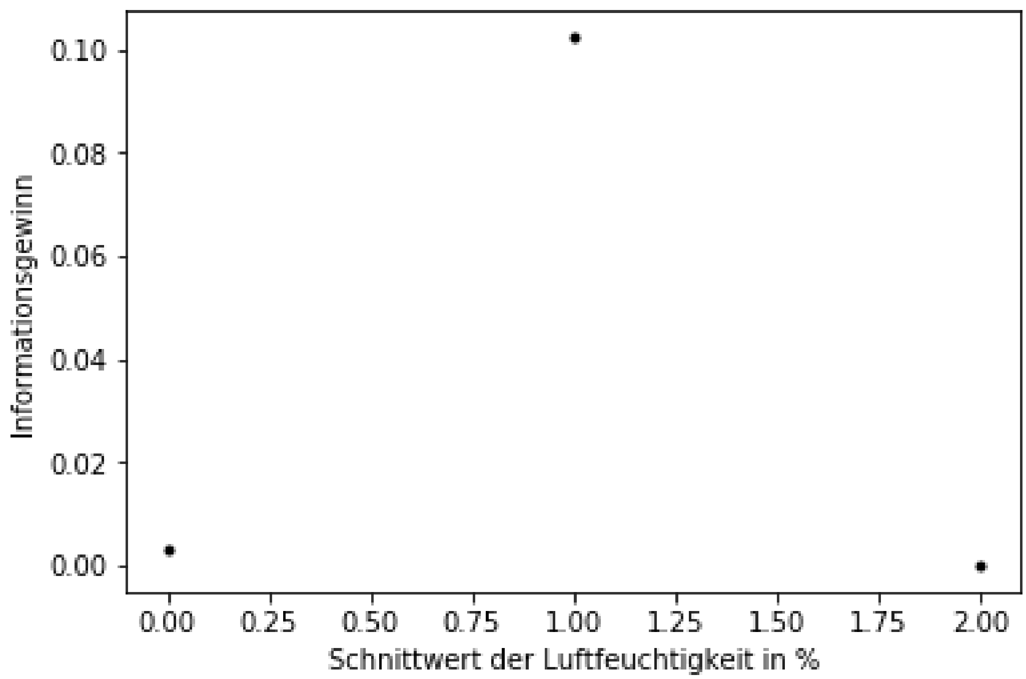
1.3 Teilaufgabe c)

Das gleiche Verfahren wird nun auch auf die anderen Attribute angewandt. Hierbei müssen dann alle möglichen Schnitte überprüft werden.

1.3.1 Wettervorhersage:

In [6]:
```
#Wettervorhersage
W = np.array([2,2,1,0,0,0,1,2,2,0,2,1,1,0])

#streicht doppelte einträge, sind sinnvolle CutWerte (damit sich nichts doppelt)
cutvalue = dict(map(lambda i: (i,1),W)).keys()
#Anzahl Werte
valnum = len(W)
#Matrix aus Werten und Information über Fuball ja(1)/nein(0)
W = np.array([[2,2,1,0,0,0,1,2,2,0,2,1,1,0], [0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0

#Aufteilen der Werte in über oder unter dem CutWert liegende Werte
#Das alles für alle unterschiedlichen CutWerte
for i in cutvalue:
    lowerf = np.array([]) #Array mit Information übers Fuballspielen der tieferen Wert
    higherf = np.array([]) #Array mit Information übers Fuballspielen der höheren Wert
    n = 0
    while n < 14:
        if W[0,n] <= i:
            lowerf = np.append(lowerf, W[1,n])
        else:
            higherf= np.append(higherf, W[1,n])
        n+=1

#Berechnung der Entropien aus den einzelnen Wahrscheinlichkeiten, welche man aus den l
#if-Abfragen damit keine 0 im log2 steht
    H = 0
    if len(lowerf[lowerf==0]) != 0:
        H += -len(lowerf)/valnum*(len(lowerf[lowerf==0])/len(lowerf)*math.log(len(lowe
    if len(lowerf[lowerf==1]) != 0:
        H += -len(lowerf)/valnum*(len(lowerf[lowerf==1])/len(lowerf)*math.log(len(lowe
    if len(higherf[higherf==0]) != 0:
        H += -len(higherf)/valnum*(len(higherf[higherf==0])/len(higherf)*math.log(len(h
    if len(higherf[higherf==1]) != 0:
        H += -len(higherf)/valnum*(len(higherf[higherf==1])/len(higherf)*math.log(len(h

    IG = H_f - H
    print('Schnitt bei:')
    print(i)
    print('Informationsgewinn:')
    print(IG)
    print('-----------------------------')
    plt.plot(i, IG, 'k.')

plt.xlabel('Schnittwert der Wettervorhersage')
plt.ylabel('Informationsgewinn')
plt.show()
```

```
Schnitt bei:
2
Informationsgewinn:
0.0
-----------------------------
Schnitt bei:
1
Informationsgewinn:
0.10224356360985043
-----------------------------
Schnitt bei:
0
Informationsgewinn:
0.003184853044648883
-----------------------------
```



Es ergibt sich bester Inormationsgewinn mit einem Schnitt bei einer Wettervorhersage zwischen 0 und 1.

Dieser beträgt $IG = 0.10224356360985043$

1.3.2 Luftfeuchtigkeit:

```
In [ ]: #Luftfeuchtigkeit
        LF = np.array([85, 90, 78, 96, 80, 70, 65, 95, 70, 80, 70, 90, 75, 80])
```

```
#streicht doppelte einträge, sind sinnvolle CutWerte (damit sich nichts doppelt)
cutvalue = dict(map(lambda i: (i,1),LF)).keys()
#Anzahl Werte
valnum = len(LF)
#Matrix aus Werten und Information über Fuball ja(1)/nein(0)
LF = np.array([[85, 90, 78, 96, 80, 70, 65, 95, 70, 80, 70, 90, 75, 80], [0, 0, 1, 1, 

#Aufteilen der Werte in über oder unter dem CutWert liegende Werte
#Das alles für alle unterschiedlichen CutWerte
for i in cutvalue:
    lowerf = np.array([]) #Array mit Information übers Fuballspielen der tieferen Wert
    higherf = np.array([]) #Array mit Information übers Fuballspielen der höheren Wert
    n = 0
    while n < 14:
        if LF[0,n] <= i:
            lowerf = np.append(lowerf, LF[1,n])
        else:
            higherf= np.append(higherf, LF[1,n])
        n+=1

#Berechnung der Entropien aus den einzelnen Wahrscheinlichkeiten, welche man aus den l
#if-Abfragen damit keine 0 im log2 steht
    H = 0
    if len(lowerf[lowerf==0]) != 0:
        H += -len(lowerf)/valnum*(len(lowerf[lowerf==0])/len(lowerf)*math.log(len(lowe
    if len(lowerf[lowerf==1]) != 0:
        H += -len(lowerf)/valnum*(len(lowerf[lowerf==1])/len(lowerf)*math.log(len(lowe
    if len(higherf[higherf==0]) != 0:
        H += -len(higherf)/valnum*(len(higherf[higherf==0])/len(higherf)*math.log(len(
    if len(higherf[higherf==1]) != 0:
        H += -len(higherf)/valnum*(len(higherf[higherf==1])/len(higherf)*math.log(len(

    IG = H_f - H
    print('Schnitt bei:')
    print(i)
    print('Informationsgewinn:')
    print(IG)
    print('------------------------------')
    plt.plot(i, IG, 'k.')

plt.xlabel('Schnittwert der Luftfeuchtigkeit in %')
plt.ylabel('Informationsgewinn')
plt.show()
```

Es ergibt sich bester Inormationsgewinn mit einem Schnitt bei einem Feuchtigkeitswert zwischen 80% und 85%.

Dieser beträgt $IG = 0.10224356360985043$

1.3.3 Temperatur:

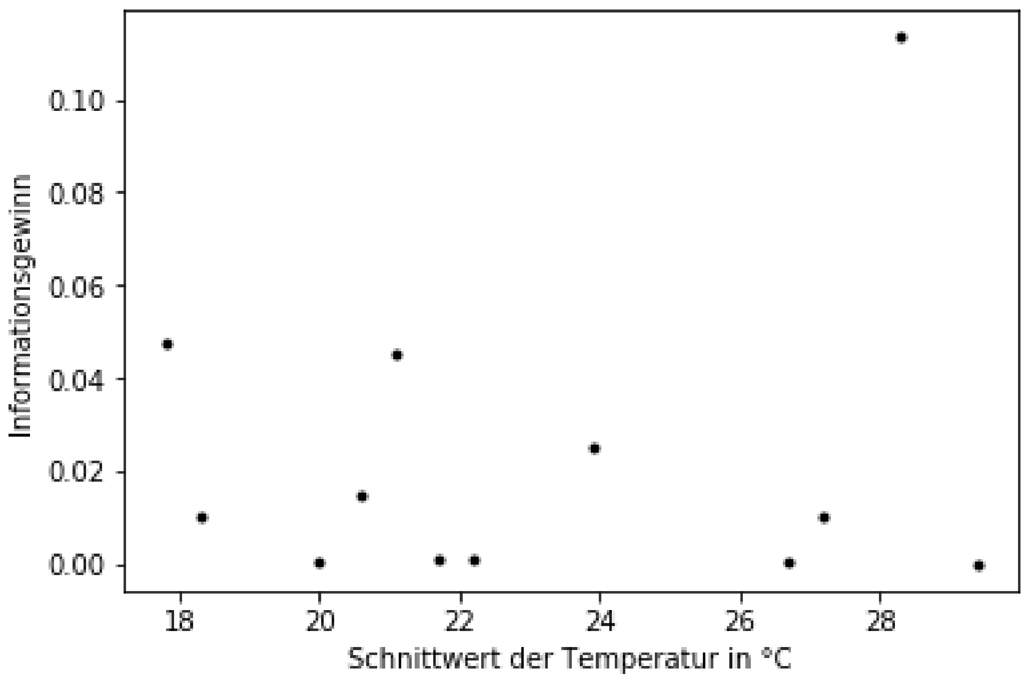
Analog zu Luftfeuchtigkeit

In [7]:
```
#Temperatur
T = np.array([29.4, 26.7, 28.3, 21.1, 20, 18.3, 17.8, 22.2, 20.6, 23.9, 23.9, 22.2, 27

#streicht doppelte einträge, sind sinnvolle CutWerte (damit sich nichts doppelt)
cutvalue = dict(map(lambda i: (i,1),T)).keys()
#Anzahl Werte
valnum = len(T)
#Matrix aus Werten und Information über Fuball ja(1)/nein(0)
T = np.array([[29.4, 26.7, 28.3, 21.1, 20, 18.3, 17.8, 22.2, 20.6, 23.9, 23.9, 22.2, 2

#Aufteilen der Werte in über oder unter dem CutWert liegende Werte
#Das alles für alle unterschiedlichen CutWerte
for i in cutvalue:
    lowerf = np.array([]) #Array mit Information übers Fuballspielen der tieferen Wert
    higherf = np.array([]) #Array mit Information übers Fuballspielen der höheren Wert
    n = 0
    while n < 14:
        if T[0,n] <= i:
            lowerf = np.append(lowerf, T[1,n])
        else:
            higherf= np.append(higherf, T[1,n])
        n+=1

#Berechnung der Entropien aus den einzelnen Wahrscheinlichkeiten, welche man aus den l
#if-Abfragen damit keine 0 im log2 steht
    H = 0
    if len(lowerf[lowerf==0]) != 0:
        H += -len(lowerf)/valnum*(len(lowerf[lowerf==0])/len(lowerf)*math.log(len(lowe
    if len(lowerf[lowerf==1]) != 0:
        H += -len(lowerf)/valnum*(len(lowerf[lowerf==1])/len(lowerf)*math.log(len(lowe
    if len(higherf[higherf==0]) != 0:
        H += -len(higherf)/valnum*(len(higherf[higherf==0])/len(higherf)*math.log(len(
    if len(higherf[higherf==1]) != 0:
        H += -len(higherf)/valnum*(len(higherf[higherf==1])/len(higherf)*math.log(len(

    IG = H_f - H
    print('Schnitt bei:')
    print(i)
    print('Informationsgewinn:')
    print(IG)
    print('------------------------------')
    plt.plot(i, IG, 'k.')

plt.xlabel('Schnittwert der Temperatur in řC')
```

```python
        plt.ylabel('Informationsgewinn')
        plt.show()
```

```
Schnitt bei:
29.4
Informationsgewinn:
0.0
-----------------------------
Schnitt bei:
26.7
Informationsgewinn:
0.0004894691870229728
-----------------------------
Schnitt bei:
28.3
Informationsgewinn:
0.11340086418110329
-----------------------------
Schnitt bei:
21.1
Informationsgewinn:
0.04533417202914436
-----------------------------
Schnitt bei:
20.0
Informationsgewinn:
0.0004894691870229728
-----------------------------
Schnitt bei:
18.3
Informationsgewinn:
0.010318100909640027
-----------------------------
Schnitt bei:
17.8
Informationsgewinn:
0.047709111427960416
-----------------------------
Schnitt bei:
22.2
Informationsgewinn:
0.0013397424044412354
-----------------------------
Schnitt bei:
20.6
Informationsgewinn:
0.01495606992897247
-----------------------------
```

```
Schnitt bei:
23.9
Informationsgewinn:
0.02507817350585051
-----------------------------
Schnitt bei:
27.2
Informationsgewinn:
0.010318100909640027
-----------------------------
Schnitt bei:
21.7
Informationsgewinn:
0.0013397424044412354
-----------------------------
```



Es ergibt sich bester Inormationsgewinn mit einem Schnitt bei einer Temperatur zwischen 28,3řC und 29,4řC.

Dieser beträgt $IG = 0.11340086418110329$

1.4 Teilaufgabe d)

Bei nur einem Schritt eignet sich also die Temperatur am besten zum Trennen der Daten.