Blatt6_Olbrich,Rotgeri

June 15, 2020

1 Aufgabe 11

Führen Sie eine lineare Diskriminazanalyse nach Fisher per Hand durch

```
In [94]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
```

$$\text{Population 0}: (1;1)(2;1)(1,5;2)(2;2)(2;3)(3;3)$$
$$\text{Population 1}: (1.5;1)(2.5;1)(3.5;1)(2.5;2)(3.5;2)(4.5;2)$$

a) Berechnen Sie die Mittelwerte und Streumatrizen , sowie die kombinierte Streumatrix .

Mittelwerte: $\vec{\mu} = \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \frac{1}{N} \begin{pmatrix} \sum_{i=1}^N x_i \\ \sum_{i=1}^N y_i \end{pmatrix} \rightarrow \vec{\mu_0} = \begin{pmatrix} \frac{23}{12} \\ 2 \end{pmatrix}, \vec{\mu_1} = \begin{pmatrix} 3 \\ \frac{3}{2} \end{pmatrix}$
Streumatrizen: $S_i = \sum_{j=1}^{N_i} (\vec{x}_j - \vec{\mu}_j)(\vec{x}_j - \vec{\mu}_j)^T$

$$S_0 = \begin{pmatrix} -\frac{11}{12} \\ -1 \end{pmatrix} \begin{pmatrix} -\frac{11}{12} \\ -1 \end{pmatrix}^T + \begin{pmatrix} \frac{1}{12} \\ -1 \end{pmatrix} \begin{pmatrix} \frac{1}{12} \\ -1 \end{pmatrix}^T + \begin{pmatrix} -\frac{5}{12} \\ 0 \end{pmatrix} \begin{pmatrix} -\frac{5}{12} \\ 0 \end{pmatrix}^T + \begin{pmatrix} \frac{1}{12} \\ 0 \end{pmatrix} \begin{pmatrix} \frac{1}{12} \\ 0 \end{pmatrix}^T + \begin{pmatrix} \frac{1}{12} \\ 1 \end{pmatrix} \begin{pmatrix} \frac{1}{12} \\ 1 \end{pmatrix}^T + \begin{pmatrix} \frac{13}{12} \\ 1 \end{pmatrix} \begin{pmatrix} \frac{13}{12} \\ 1 \end{pmatrix}^\top$$
$$= \begin{pmatrix} \frac{121}{144} & \frac{11}{12} \\ \frac{11}{12} & 1 \end{pmatrix} + \begin{pmatrix} \frac{1}{144} & \frac{-1}{12} \\ -\frac{1}{12} & 1 \end{pmatrix} + \begin{pmatrix} \frac{25}{144} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \frac{1}{144} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \frac{1}{144} & \frac{1}{12} \\ \frac{1}{12} & 1 \end{pmatrix} + \begin{pmatrix} \frac{169}{144} & \frac{13}{12} \\ \frac{13}{12} & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 53/24 & 2 \\ 2 & 4 \end{pmatrix}$$
$$S_1 = \begin{pmatrix} \frac{9}{4} & \frac{3}{4} \\ \frac{3}{4} & \frac{1}{4} \end{pmatrix} + \begin{pmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{pmatrix} + \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} \end{pmatrix} + \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} \end{pmatrix} + \begin{pmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{pmatrix} + \begin{pmatrix} \frac{9}{4} & \frac{3}{4} \\ \frac{3}{4} & \frac{1}{4} \end{pmatrix}$$
$$= \begin{pmatrix} 11/2 & 3/2 \\ 3/2 & 13/2 \end{pmatrix}$$

Gesamtstreuung: $S_W = \sum_i^N S_i = S_0 + S_1 \rightarrow S_W = \begin{pmatrix} 185/24 & 7/2 \\ 7/2 & 11/2 \end{pmatrix}$

b) Wie lautet $\vec{\lambda}$?

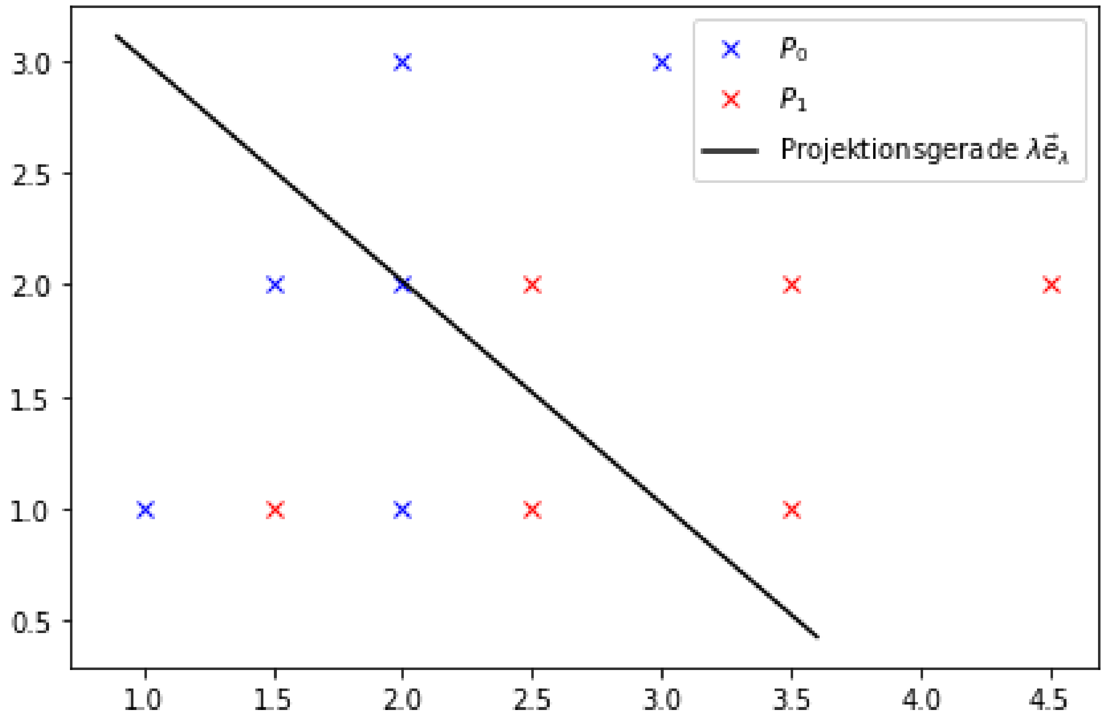
$$\vec{\lambda} = S_W^{-1}(\vec{\mu}_0 - \vec{\mu}_1)$$
$$\text{Mit } S_W^{-1} = \frac{1}{\text{Det}S_W}\begin{pmatrix} \frac{11}{2} & -\frac{7}{2} \\ -\frac{7}{2} & \frac{185}{24} \end{pmatrix} = \frac{48}{1447}\begin{pmatrix} \frac{11}{2} & -\frac{7}{2} \\ -\frac{7}{2} & \frac{185}{24} \end{pmatrix} = \begin{pmatrix} 0,182 & -0,116 \\ -0,116 & 0,256 \end{pmatrix}$$
$$\rightarrow \vec{\lambda} = \begin{pmatrix} -0,256 \\ 0,254 \end{pmatrix}$$
$$\rightarrow \vec{\lambda} = \lambda \vec{e}_\lambda = 0,360 \begin{pmatrix} -0,710 \\ 0,704 \end{pmatrix}$$

c) Zeichnen Sie die Punkte der beiden Populationen in einen Graphen ein, zusammen mit der Projektionsgeraden

In [95]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit
#import uncertainties.unumpy as unp
from uncertainties import ufloat
from scipy.stats import stats
from scipy import linalg



P0x = np.array([1,2,1.5,2,2,3])
P0y = np.array([1,1,2,2,3,3])
P1x = np.array([1.5, 2.5, 3.5, 2.5, 3.5, 4.5])
P1y = np.array([1,1,1,2,2,2])

m=0.254/(-0.256)
def f(x):
  return m*x+4


x = np.linspace(0.9, 3.6,100)




plt.plot(P0x, P0y, 'bx', label=r'$P_0$')
plt.plot(P1x, P1y, 'rx', label=r'$P_1$')
plt.plot(x, f(x), 'k-', label=r'Projektionsgerade $\lambda \vec{e}_\lambda$')



plt.legend()
plt.tight_layout()
plt.show()
```

```python
plt.savefig('c.pdf')
```

```python
plt.clf()
```



(d) Projezieren Sie die einzelnen Punkte auf diese Gerade.

Projektion= $\vec{\lambda}^T \cdot x$

$$\text{Population } 0: -0.006; -0.716; 0.343; -0.012; 0.693; -0.017$$
$$\text{Population } 1: -0.361; -1.071; -1.781; -0.367; -1.076; -1.786$$

(e) Wählen Sie einen geeigneten Parameter cut und berechnen Sie die dazugehörige Effizienz und Reinheit. Warum haben Sie diesen Parameter gewählt?

$\lambda_{cut} = -0.360$ sinnvoll, da hierbei 5/6 von Population 0 und 6/6 von Population 1 richtig zugeordnet werden.

$t_p = 6$
$t_n = 5$
$f_p = 1$
$f_n = 0$
Reinheit $= \frac{\text{tp}}{\text{tp + fp}} = \frac{6}{7}$
Effizienz $= \frac{\text{tp}}{\text{tp + fn}} = \frac{6}{6} = 1$

2 Aufgabe 12

In [ ]:

In [96]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


p0 = pd.read_hdf('zwei_populationen.h5', key='P_0_10000')
p1 = pd.read_hdf('zwei_populationen.h5', key='P_1')
p0_1000 = pd.read_hdf('zwei_populationen.h5', key='P_0_1000')
```

(a) Berechnen Sie die Mittelwerte P 0 und P 1 der beiden Populationen.

In [97]:
```
mu_p0 = p0.mean()
mu_p1 = p1.mean()
print('Mittelwerte:')
print()
print('mu_p0')
print(mu_p0)
print()
print('mu_p1')
print(mu_p1)
```

```
Mittelwerte:

mu_p0
x   -0.007300
y    2.963676
dtype: float64

mu_p1
x    6.096272
y    3.174674
dtype: float64
```

(b) Berechnen Sie die Kovarianzmatrizen V P 0 und V P 1 der beiden Populationen, sowie die kombi- nierte Kovarianzmatrix V P 0,P 1 .

In [98]:
```
V_p0 = p0.cov()
V_p1 = p1.cov()

p0p1 = pd.concat([p0,p1])
V_p01 = p0.append(p1).cov()

print('Kovarianzmatrix:')
print()
```

```
print('V_p0')
print(V_p0)
print()
print('V_p1')
print(V_p1)
print()
print('V_p01')
print(V_p01)
```

```
Kovarianzmatrix:

V_p0
           x         y
x  12.554716  8.360258
y   8.360258  6.838396

V_p1
           x        y
x  12.052492  7.21376
y   7.213760  5.32757

V_p01
           x         y
x  21.616851  8.108595
y   8.108595  6.093809
```

(c) Konstruieren Sie eine lineare Fisher-Diskriminante = e . Geben Sie diese Geradengleichung an.

In [99]:
```
# Definition des Vektorprodukts
def vprod(x):
    y = np.atleast_2d(x)
    return np.dot(y.T, y)

S0 = np.zeros(2)
for index, row in (p0 - mu_p0).iterrows():
    S0 = S0 + vprod(row)

S1 = np.zeros(2)
for index, row in (p1 - mu_p0).iterrows():
    S1 = S1 + vprod(row)

# Addition der Matritzen
SW = S0 + S1
print('Streumatrix:')
print()
print('S0 = ', S0)
```

```python
print()
print('S1 = ', S1)
print()
print('SW = ', SW)
print()

# Berechnung der Fischer-Diskriminante
L = np.dot(np.linalg.inv(SW), (mu_p0-mu_p1))
print('L = ', L)

# Geradengleichung noch ausrechnen!!!
norm = np.linalg.norm(L)
print("Normierung: ", norm)
L_norm = L/norm
print('L_norm = ', L_norm)
```

```
Streumatrix:

S0 =  [[125534.60451066  83594.21951235]
 [ 83594.21951235  68377.1222049 ]]

S1 =  [[493048.72624545  85008.76520258]
 [ 85008.76520258  53715.57106624]]

SW =  [[618583.33075611 168602.98471493]
 [168602.98471493 122092.69327114]]

L =  [-1.50671717e-05  1.90787233e-05]
Normierung:  2.431084837388791e-05
L_norm =  [-0.61977153  0.78478229]
```

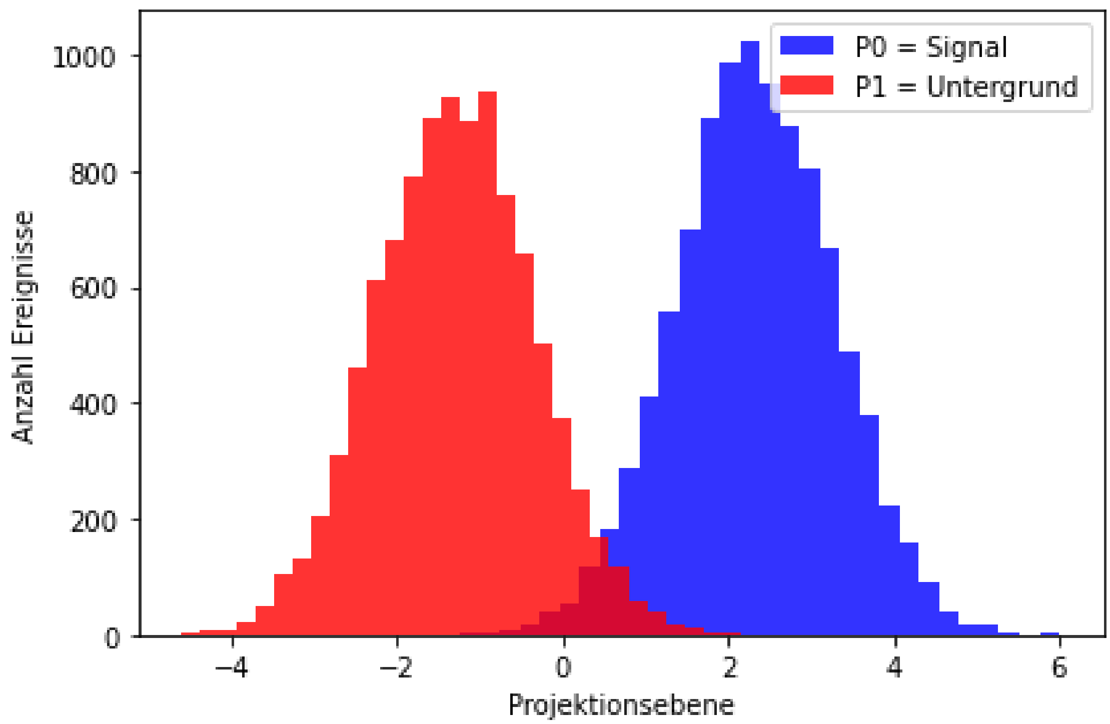
Es ergibt sich somit die Geradengleichung:
$\vec{\lambda} = \lambda \cdot \vec{e}_\lambda = 0,0000243 \cdot \begin{pmatrix} -0,619 \\ 0,785 \end{pmatrix}$

(d) Stellen Sie die Populationen als Projektion auf die Gerade aus (c) in einem eindimensionalen Histogramm dar.

```python
In [100]: Projektion_0 = np.array([])
          for index, row in (p0).iterrows():
              Projektion_0 = np.append(Projektion_0, np.vdot(row, L_norm))

          Projektion_1 = np.array([])
          for index, row in (p1).iterrows():
              Projektion_1 = np.append(Projektion_1, np.vdot(row, L_norm))

          plt.hist(Projektion_0, label='P0 = Signal', bins=30, color='b', alpha=0.8)
          plt.hist(Projektion_1, label='P1 = Untergrund', bins=30, color='r', alpha=0.8)
```

```
plt.xlabel('Projektionsebene')
plt.ylabel('Anzahl Ereignisse')
plt.legend(loc="best")
plt.tight_layout()
plt.savefig('Projektionen.pdf')
plt.show()
plt.clf()
```

```
None
```



(e) Betrachten Sie P 0 als Signal und P 1 als Untergrund. Berechnen Sie die Effizienz und die Reinheit des Signals als Funktion eines Schnittes cut in und stellen Sie die Ergebnisse in einem Plot dar.

(f) Bei welchem Wert von cut wird nach der Trennung das Signal-zu-Untergrundverhältnis S/B maximal? Erstellen Sie auch hierzu einen Plot.

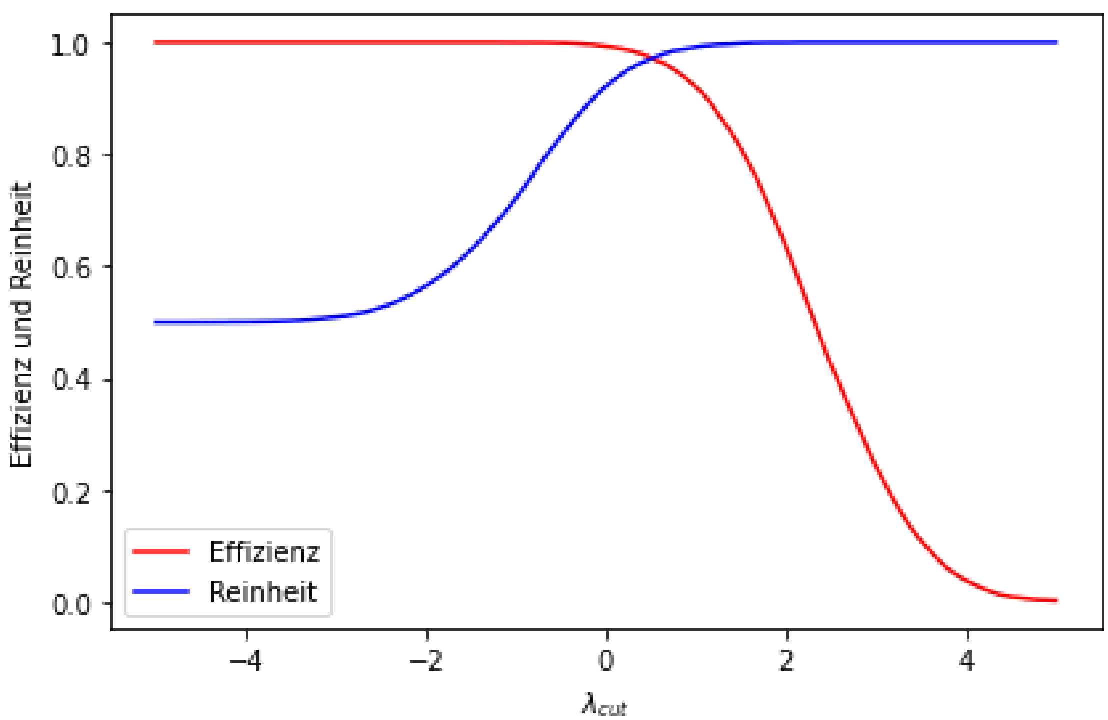
(g) Bei welchem Wert von cut wird nach der Trennung die Signifikanz S/ S + B maximal? Erstellen Sie auch hierzu einen Plot.

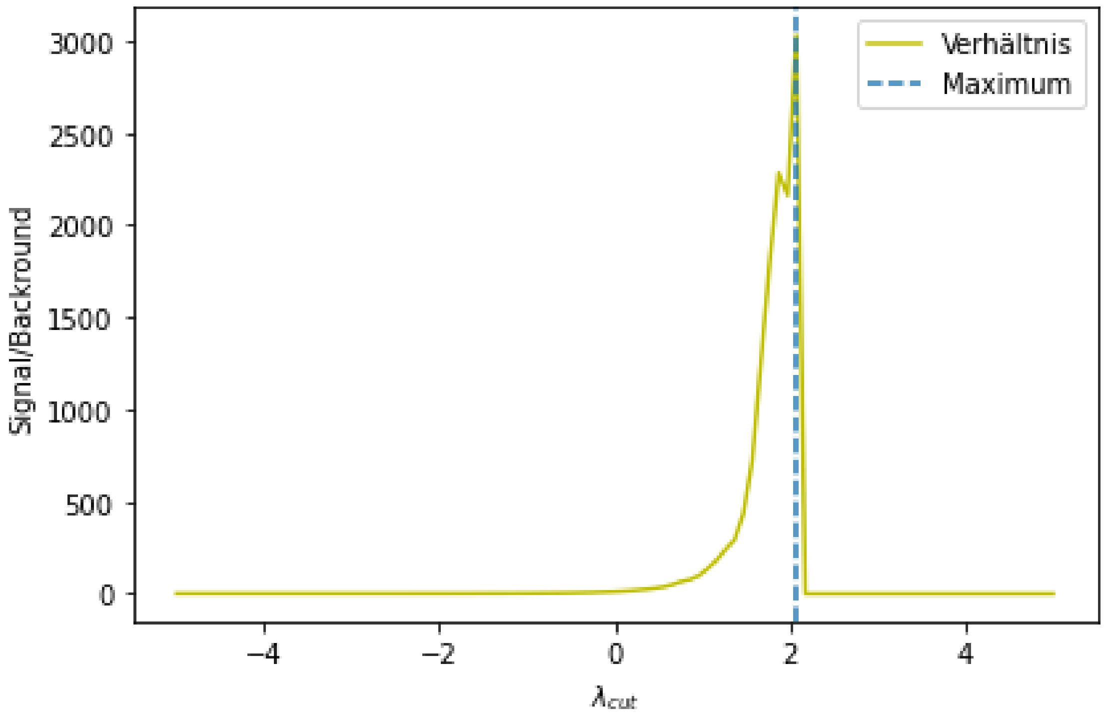
Die folgenden Formeln wurden verwendet:
Reinheit = true positive / (true positive + false positive)
Effizienz = true positive / (true positive + false negative)

In [101]:
```
lcut = np.linspace(-5,5, 100)
Back = Projektion_1
Sig = Projektion_0

#leere Arrays erzeugen
eff = np.zeros(len(lcut))
rein = np.zeros(len(lcut))
ver = np.zeros(len(lcut))
sign = np.zeros(len(lcut))
#Effizienz und Reinheit für jedes l_cut berechnen
for i in range(len(lcut)):
    tp = len(Sig[Sig > lcut[i]])
    fp = len(Back[Back > lcut[i]])
    fn = len(Sig[Sig <= lcut[i]])

    eff[i] = tp/(tp + fn)
    rein[i] = tp/(tp + fp)
    if fp != 0:
            ver[i] = tp/fp
    sign[i] = tp/(np.sqrt(tp + fp))

# Maximum des Verhältnisses berechnen
lcut_maxv = lcut[np.argmax(ver)]
#print('lcut_maxv = ', lcut_maxv )

# Maximum der Signifikanz berechnen
lcut_maxs = lcut[np.argmax(sign)]
#print('lcut_maxs = ', lcut_maxs)


# Plots
plt.plot(lcut, eff, 'r-', label='Effizienz')
plt.plot(lcut, rein, 'b-', label='Reinheit')
plt.xlabel(r"$\lambda_{cut}$")
plt.ylabel('Effizienz und Reinheit')
plt.legend(loc="best")
plt.tight_layout()
plt.savefig('EffizienzReinheit.pdf')
plt.show()
plt.clf()

plt.plot(lcut, ver, 'y-', label='Verhältnis')
plt.xlabel(r"$\lambda_{cut}$")
plt.ylabel('Signal/Backround')
plt.axvline(x=lcut_maxv, linestyle='--', label='Maximum')
plt.legend(loc="best")
plt.tight_layout()
plt.savefig('Verhältnis.pdf')
```

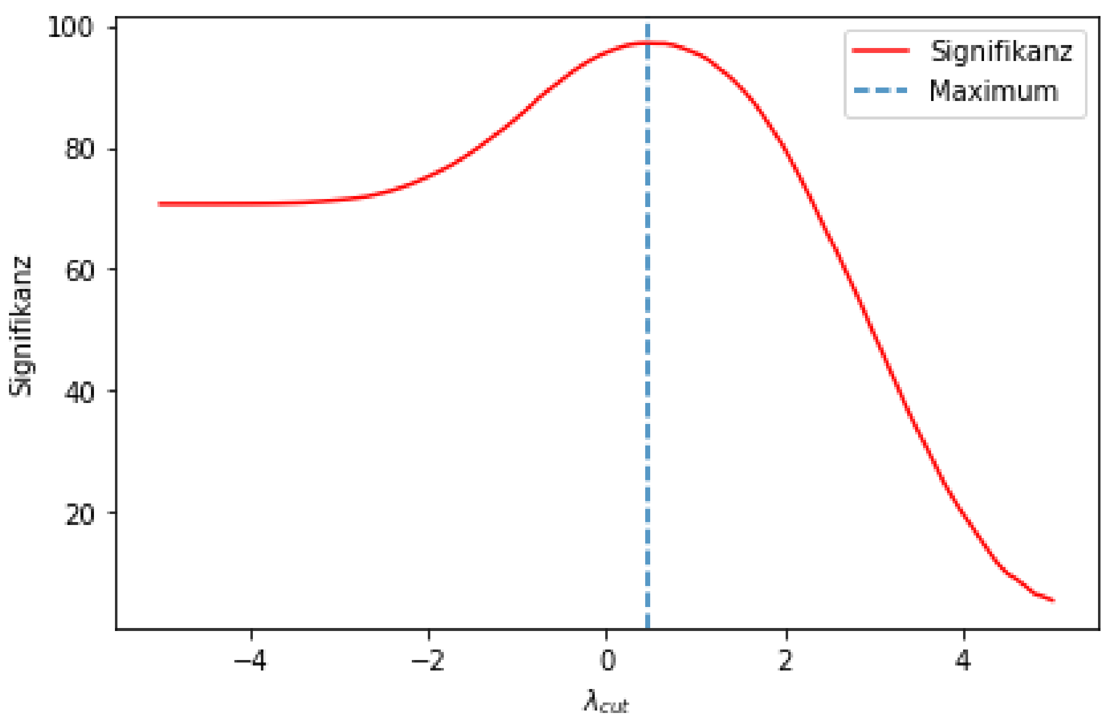
```
plt.show()
plt.clf()

print('Signal/Untergrund maximal bei lambda =', lcut_maxv )

plt.plot(lcut, sign, 'r-', label='Signifikanz')
plt.axvline(x=lcut_maxs, linestyle='--', label='Maximum')
plt.xlabel(r"$\lambda_{cut}$")
plt.ylabel('Signifikanz')
plt.legend(loc="best")
plt.tight_layout()
plt.savefig('Signifikanz.pdf')
plt.show()
plt.clf()

print('Signifikanz maximal bei lambda =', lcut_maxs )
```

None






```
Signal/Untergrund maximal bei lambda = 2.070707070707071
```



```
Signifikanz maximal bei lambda = 0.45454545454545414
```

(h) Wiederholen Sie die Schritte (a) bis (g) für den Fall, dass P 0 nun die Population P_0_1000 bezeichnet. Was fällt Ihnen auf? Interpretieren Sie die Ergebnisse.

```
In [102]: p2 = pd.read_hdf('zwei_populationen.h5', key='P_0_1000')

          mu_p2 = p2.mean()
          print('Mittelwerte:')
          print('mu_2 = ', mu_p2)
          print()

          V_p2 = p2.cov()
          V_p21 = p2.append(p1).cov()

          print('Kovarianzmatrizen:')
          print()
          print('V_p2 = ')
          print(V_p2)
          print()
          print('V_p21 = ')
          print(V_p21)
          print()

          S2 = np.zeros(2)
          for index, row in (p2 - mu_p2).iterrows():
              S2 = S2 + vprod(row)

          # Addition der Matritzen
          SW2 = S2 + S1
          print('Streumatrix:')
          print('S2 = ', S2)
          print()
          print('SW2 = ', SW2)
          print()

          L2 = np.dot(np.linalg.inv(SW2), (mu_p2-mu_p1))
          print('L2 = ', L2)
          # Geradengleichung noch ausrechnen!!!
          norm2 = np.linalg.norm(L2)
          print("Normierung2: ", norm2)
          L2_norm = L2/norm2
          print('L2_norm = ', L2_norm)

          None
```

```
Mittelwerte:
mu_2 =  x   -0.026781
y    3.015787
dtype: float64

Kovarianzmatrizen:

V_p2 =
           x         y
x  12.066306  8.126692
y   8.126692  6.651329

V_p21 =
           x         y
x  15.151426  7.376432
y   7.376432  5.449404

Streumatrix:
S2 =  [[12054.24005952  8118.56549865]
 [ 8118.56549865  6644.67747606]]

SW2 =  [[505102.96630497  93127.33070123]
 [ 93127.33070123  60360.2485423 ]]

L2 =  [-1.62633576e-05  2.24597600e-05]
Normierung2:  2.7729724492448756e-05
L2_norm =  [-0.58649546  0.80995251]
```

Es ergibt sich somit die Geradengleichung:

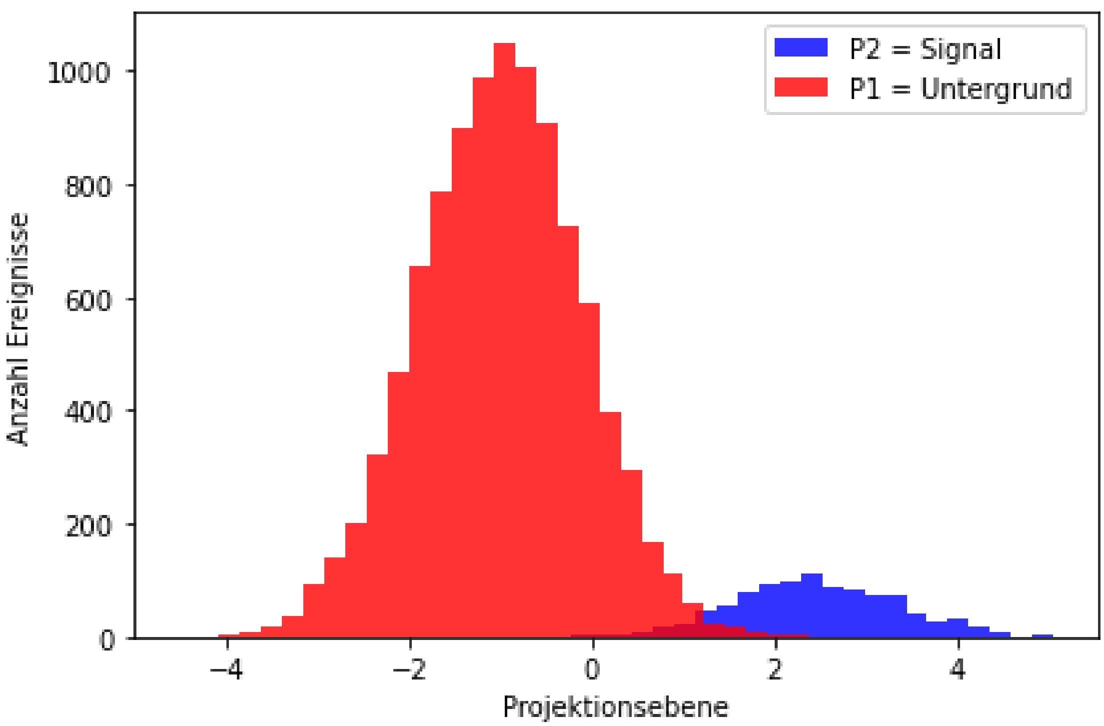
$$\vec{\lambda} = \lambda \cdot \vec{e}_\lambda = 0{,}0000277 \cdot \begin{pmatrix} -0{,}586 \\ 0{,}810 \end{pmatrix}$$

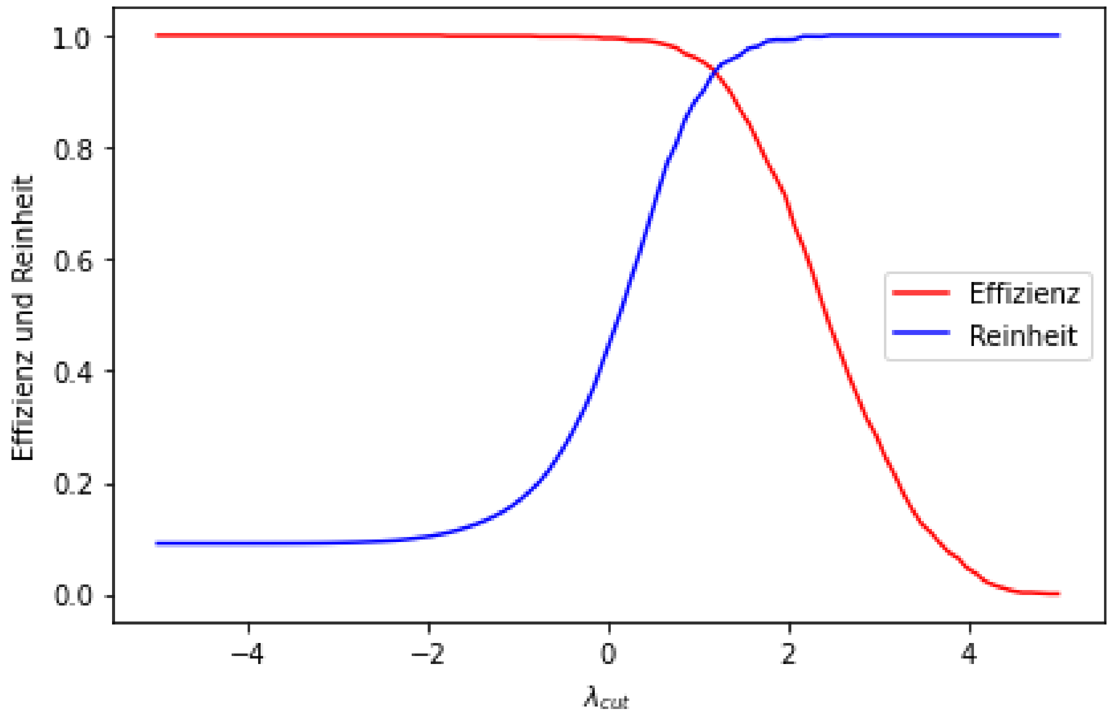
In [103]:
```
Projektion2_2 = np.array([])
for index, row in (p2).iterrows():
    Projektion2_2 = np.append(Projektion2_2, np.vdot(row, L2_norm))

Projektion2_1 = np.array([])
for index, row in (p1).iterrows():
    Projektion2_1 = np.append(Projektion2_1, np.vdot(row, L2_norm))

plt.hist(Projektion2_2, label='P2 = Signal', bins=30, color='b', alpha=0.8)
plt.hist(Projektion2_1, label='P1 = Untergrund', bins=30, color='r', alpha=0.8)
plt.xlabel('Projektionsebene')
plt.ylabel('Anzahl Ereignisse')
plt.legend(loc="best")
plt.tight_layout()
plt.savefig('Projektionen2.pdf')
```

```python
plt.show()
plt.clf()

lcut = np.linspace(-5,5, 100)
Back2 = Projektion2_1
Sig2 = Projektion2_2

#leere Arrays erzeugen
eff2 = np.zeros(len(lcut))
rein2 = np.zeros(len(lcut))
ver2 = np.zeros(len(lcut))
sign2 = np.zeros(len(lcut))
#Effizienz und Reinheit für jedes l_cut berechnen
for i in range(len(lcut)):
    tp2 = len(Sig2[Sig2 > lcut[i]])
    fp2 = len(Back2[Back2 > lcut[i]])
    fn2 = len(Sig2[Sig2 <= lcut[i]])

    eff2[i] = tp2/(tp2 + fn2)
    rein2[i] = tp2/(tp2 + fp2)
    if fp2 != 0:
            ver2[i] = tp2/fp2
    sign2[i] = tp2/(np.sqrt(tp2 + fp2))

# Maximum des Verhältnisses berechnen
lcut2_maxv = lcut[np.argmax(ver2)]
#print('lcut2_maxv = ', lcut2_maxv )

# Maximum der Signifikanz berechnen
lcut2_maxs = lcut[np.argmax(sign2)]
#print('lcut2_maxs = ', lcut2_maxs)


# Plots
plt.plot(lcut, eff2, 'r-', label='Effizienz')
plt.plot(lcut, rein2, 'b-', label='Reinheit')
plt.xlabel(r"$\lambda_{cut}$")
plt.ylabel('Effizienz und Reinheit')
plt.legend(loc="best")
plt.tight_layout()
plt.savefig('EffizienzReinheit2.pdf')
plt.show()
plt.clf()

plt.plot(lcut, ver2, 'y-', label='Verhältnis')
plt.xlabel(r"$\lambda_{cut}$")
plt.ylabel('Signal/Backround')
plt.axvline(x=lcut2_maxv, linestyle='--', label='Maximum')
```

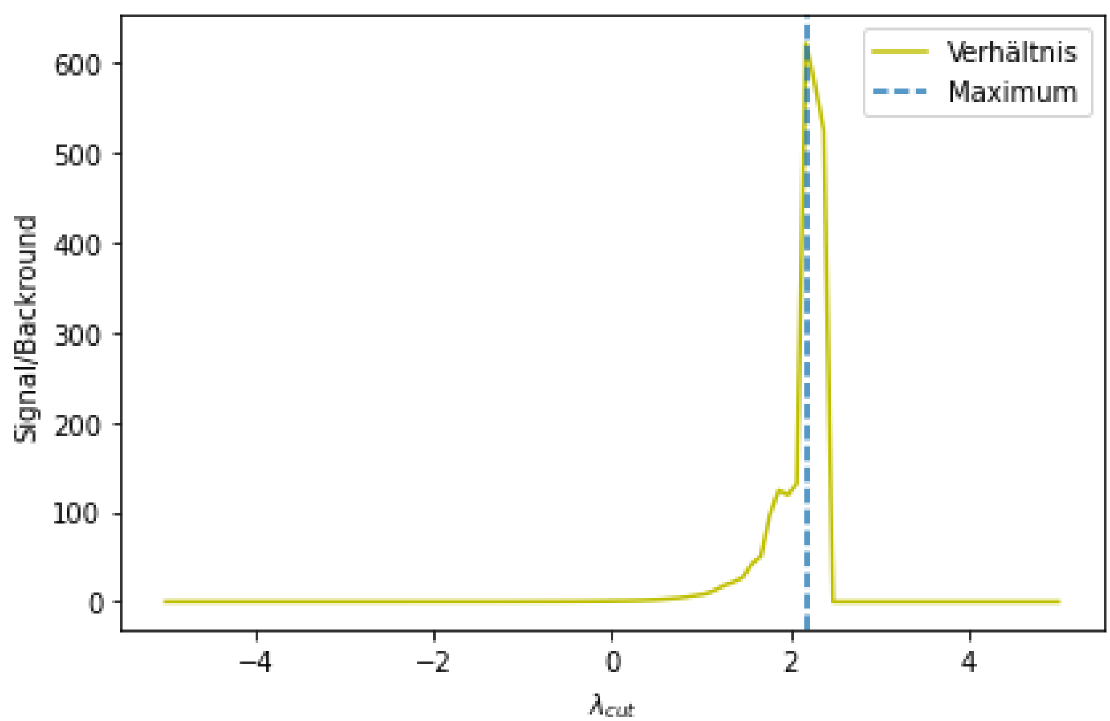
```
plt.legend(loc="best")
plt.tight_layout()
plt.savefig('Verhältnis2.pdf')
plt.show()
plt.clf()

print('Signal/Untergurund maximal bei lambda =', lcut2_maxv )

plt.plot(lcut, sign2, 'r-', label='Signifikanz')
plt.axvline(x=lcut2_maxs, linestyle='--', label='Maximum')
plt.xlabel(r"$\lambda_{cut}$")
plt.ylabel('Signifikanz')
plt.legend(loc="best")
plt.tight_layout()
plt.savefig('Signifikanz2.pdf')
plt.show()
plt.clf()

print('Signifikanz maximal bei lambda =', lcut2_maxs )
```

```
None
```



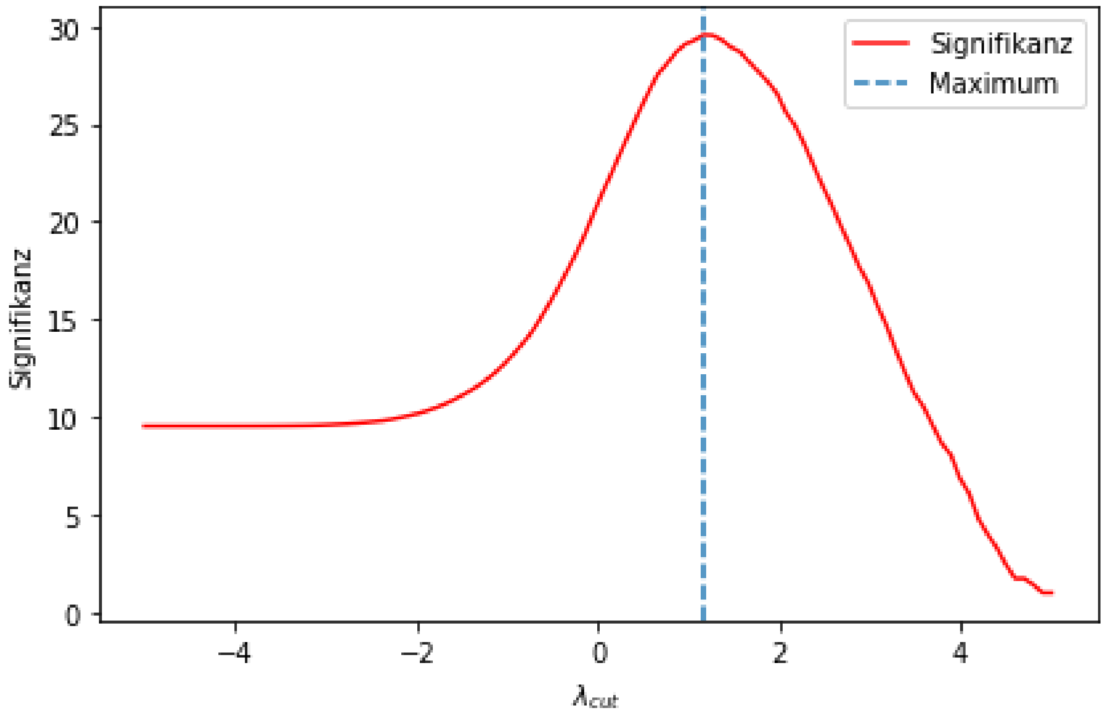


Effizienz und Reinheit
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4
λ_{cut}
Effizienz
Reinheit



Signal/Backround
600
500
400
300
200
100
0
−4
−2
0
2
4
λ_{cut}
Verhältnis
Maximum

```
Signal/Untergurund maximal bei lambda = 2.1717171717171713
```




```
Signifikanz maximal bei lambda = 1.1616161616161618
```

Da das Signal im Vergleich zum Untergrund bei der zweiten Population geringer ist, ist auch die Signifikanz wesentlich geringer. Dies liegt daran, dass bei groSSem Untergrund die Trennung nicht so gut funktioniert.

In [ ]:

In [ ]: