15 /20

SMD-Übungsblatt 5

Abgabe: 22.11.18

Yvonne Kasper yvonne.kasper@udo.edu ,
Robert Appel robert.appel@udo.edu ,
Julian Schröer julian.schroeer@udo.edu

1 Aufgabe1

Die erzeugte HDF5 Datei enthält die Daten für die Energie, und Akzeptanz mit mit der Länge 10^5 sowie die Hits, und x-, y-Werte mit einer Länge von etwa 25000, da diese nur auf den akzeptierten Energien berechnet werden. Dabei sind die Werte nicht passend zu den mit *True* makierten Ereignissen. Die Überbleibenden Plätze sind mit *NaN* gefüllt. Beispiel hier: Die Anzahl der Hits in Zeile 0 gehört

	Energy	AcceptanceMask	NumberofHits	x	y
0	3.118022	False	11.0	4.922748	2.665603
1	1.104789	False	122.0	8.274866	3.854395
2	1.482114	True	74.0	7.224836	3.309602
3	1.074785	False	29.0	6.782956	2.297691

Abbildung 1: Beispiel des DataFrames.

zu der ersten mit *True* gezeichneten Energie, also aus Zeile 3.

1.1 Teilaufgabe a)

Um die 10^5 Signalereignisse mit der Transformationsmethode zu gernerieren werden mit

$$E = (1-x)^{\frac{-1}{1,7}}$$ ✓

die Energieereignisse gesamplet. Die dazugehörige Rechnung befindet sich auf dem seperat abgegebenen Zettel.

1.2 Teilaufgabe b)

Mit dem Neumann'schen Rückweisungsverfahren wurden die tatsächlich detektierten Ereignisse gesamplet. Das Ergebnis ist in Abbildung 2 dargestellt. ✓

1.3 Teilaufgabe c)

Die Anzahl der angesprochenen Photomultiplier (Hits) werden aus einer Normalverteilung mit $\mu = 10E$ und $\sigma^2 = 2E$ (E in TeV) gezogen. Dabei gehen nur die Energien der detektierten Ereignisse ein. Dies sind von den Anfangs 10^5 Werten noch etwa 25000 Werte.

1.4 Teilaufgabe d) |-

Die Ortsmessung ist Abhängig von der Anzahl an Hits und damit wiederum Energieabhängig. Das Signal trifft am Punkt (7, 3) auf den quadratische Detektor. Für die x- und y-Richtung wird eine Normalverteilung angenommen mit $\sigma = (\log_{10}(N+1))^{-1}$. Das Ergebnis ist in dem 2D-Histogramm 3 aufgetragen. Dabei wurden jeweils 50 Bins in die beiden Richtungen gewählt.

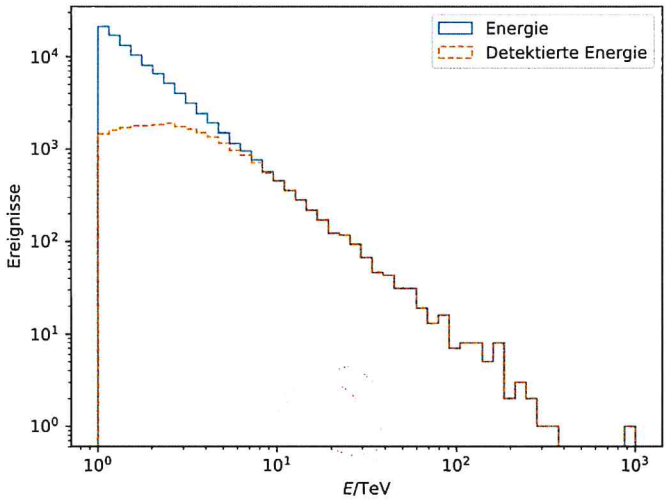
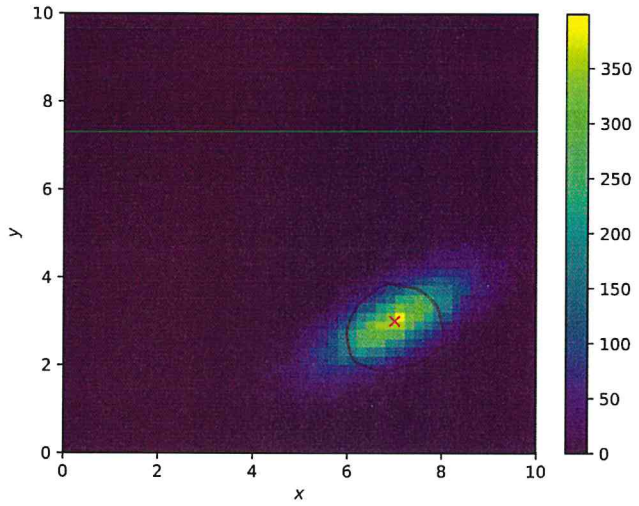


Abbildung 2: Energieereignisse und tatsächlich detektierte Energieereignisse.

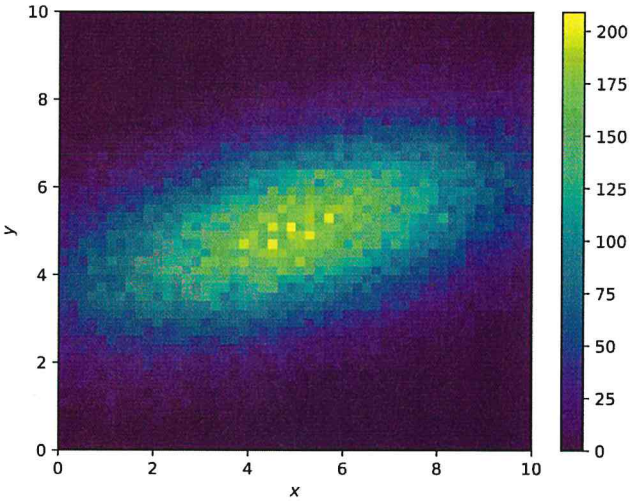
1.5 Aufgabenteil e)

Die hier gezeigten Berechnungen sind nur mit 10^5 Untergrund-Ereignissen durchgeführt, da die verwendete Funktion für die Ortsauflösung laufzeitmäßig ungünstig geschrieben ist. Mit 10^7 Ereignissen würde das Programm ca. 370 Stunden laufen. Leider hatte ich nicht mehr die Zeit die Funktion sinnvoll neu zu schreiben.



Sollte nicht elliptisch sein siehe Code

Abbildung 3: Darstellung der Ortsmessung.



Das hier stimmt auch nicht ganz

Abbildung 4: Darstellung der Ortsmessung für den Untergrund.

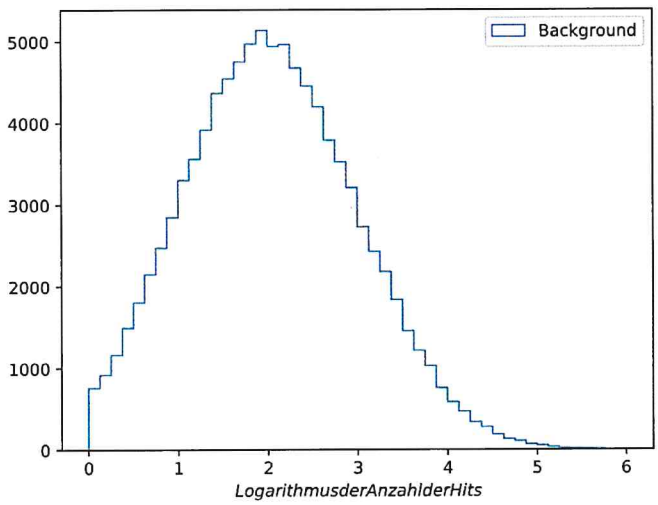
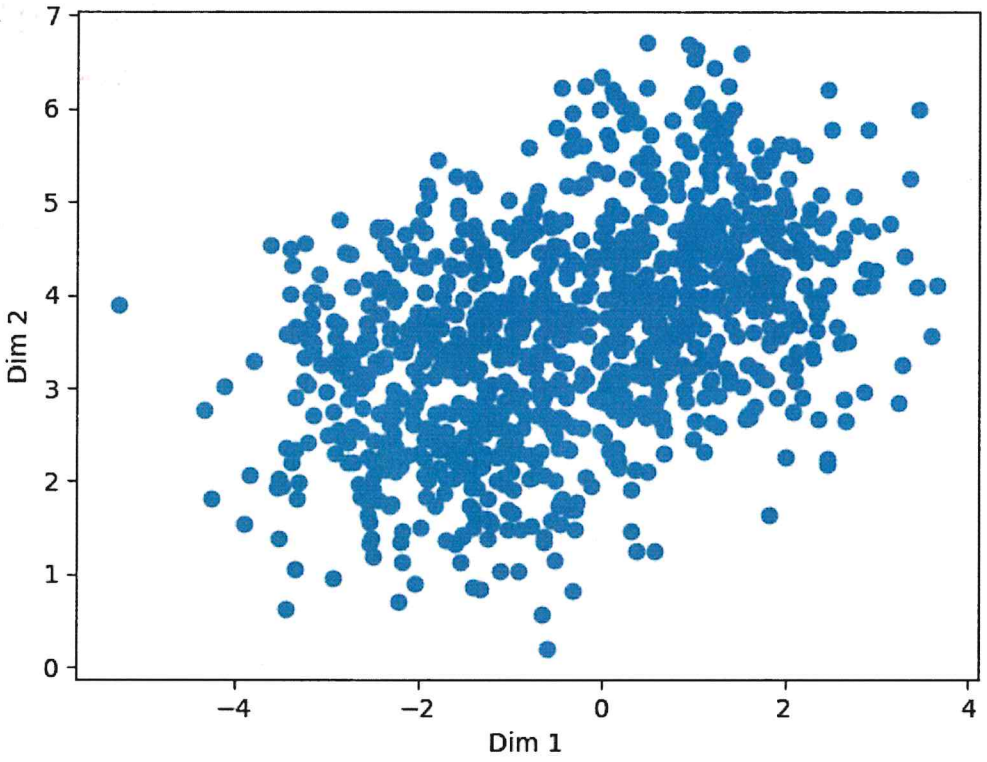


Abbildung 5: Darstellung der Untergrund-Hits.

2 Aufgabe 2

2.1 a)



2 Farben

-1P

(✓)

Abbildung 6: Scatterplot vor PCA

2.2 b)

Die PCA transformiert N Datenpunkte mit d Dimensionen auf k<d DImensionen. Dafür werden die Größten Eigenwerte ausgewählt und aus den zugehörigen Eigenvektoren bilden die Transformationsmatrix in die neue Basis. Die gesuchte neue Basis wird so festgelegt, dass die Varianz der Verteilung entlang der Basisvektoren maximiert wird. Reihenfolge:

1. Zentrierung der Daten X um ihren Mittelwert
2. Bestimmung der Kovarianzmatrix
3. Diagonalisieren der Kovarianzmatrix
4. Sortieren der Eigenwerte in absteigender Reihenfolge.
5. Auswahl der k größten Eigenwerte und der zugehörigen Eigenvektoren.
6. Bildung einer Matrix, in welcher die Eigenvektoren der k Ausgewählten Eigenwerte als Spalten stehen.

7. Transfomation der Daten $X' = WX$ mithilfe der neuen Matrix W um in ihrer Dimension reduzierte Daten zu erhalten.

2.3 c)

Nichtsortierte Eigenwerte (mit linalg ausgerechnet):
0.89875061 0.98813673 0.99958442 17.51933024
Der vierte Eigenwert hebt sich deutlich von den anderen drei Eigenwerten ab.

2.4 d)

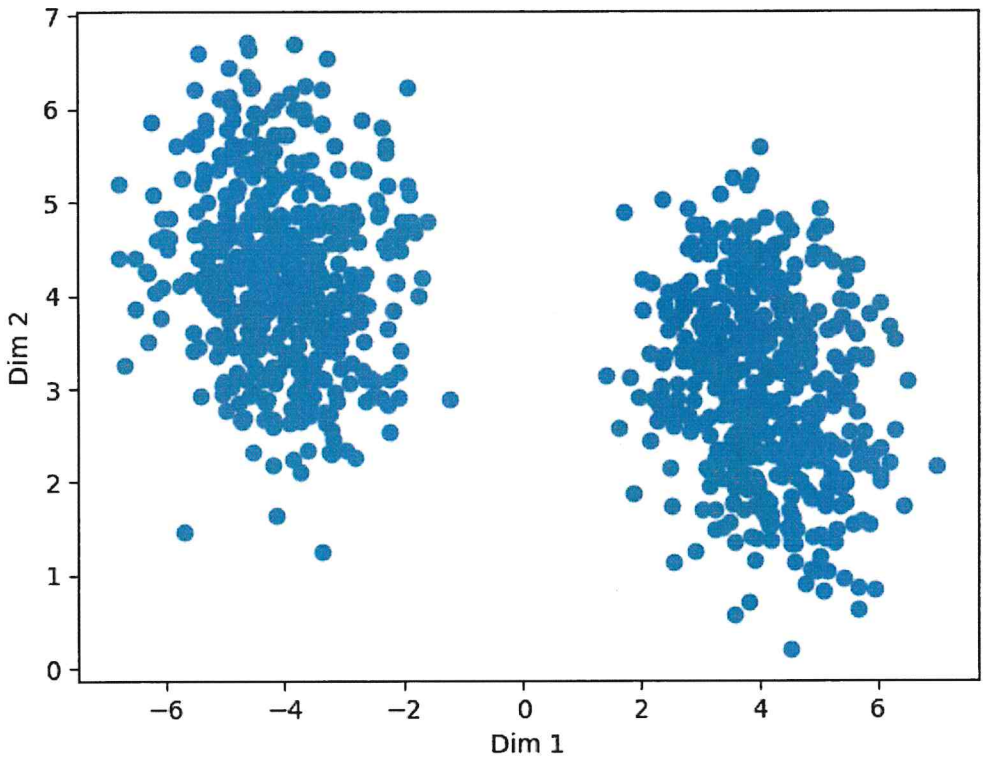


Abbildung 7: Scatterplot nach PCA

2 Farben verwenden

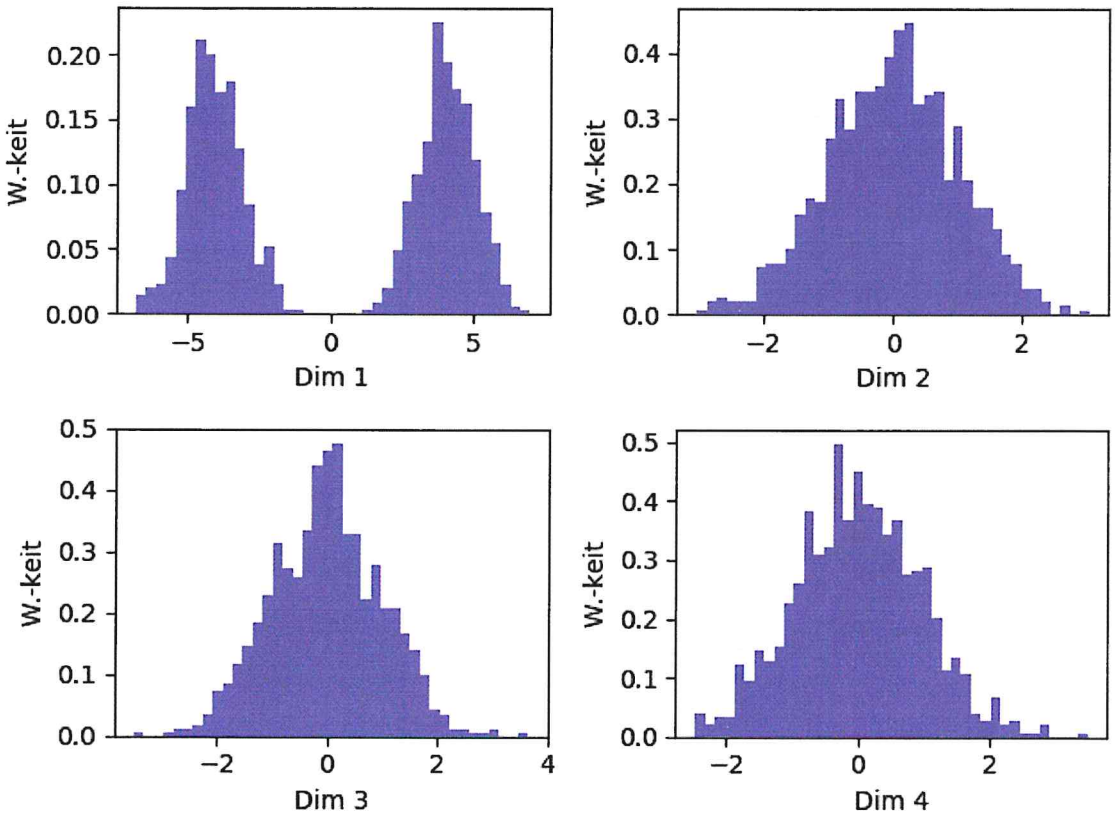


Abbildung 8: Histogramme der verschiedenen Dimensionen

3/5

Code fuer Blatt05

Kasper, Appel, Schroeer

23. November 2018

../B/1/Blatt05_Kasper_Appel_Schroeer/Blatt5.py

```
def Aufgabe1():
    def Signal(x):
        return((1-x)**(-(1.0/1.7)))

    def Neutrinofluss(E):
        return(1.7*(E**(-2.7)))

    def Akzeptanz(E):
        return((1-np.exp(-0.5*E))**3)

    def HitsSim(E):
        i = 0
        x = -1
        while i == 0:
            u1 = np.random.uniform(0, 1)
            u2 = np.random.uniform(0, 1)
            v1 = 2*u1-1
            v2 = 2*u2-1
            s = v1**2+v2**2
            if (s < 1):
                term = np.sqrt(-(2/s)*np.log(s))
                c1 = v1*term
                c2 = v2*term
                x = 2*E*c1+10*E
                x = np.round(x)
                if (x > 0):
                    i = 1
        return(x)

    def OrtsSim(N):
        Treffer = 0
        while Treffer == 0:
            u1 = np.random.uniform(0, 1)
            u2 = np.random.uniform(0, 1)
            v1 = 2*u1-1
            v2 = 2*u2-1
            s = v1**2+v2**2
            if (s < 1):
                term = np.sqrt(-(2/s)*np.log(s))
                sig = 1/(np.log10(N+1))
                c1 = v1*term
                c2 = v2*term
                x = sig*c1+sig*c2+7
                y = sig*c2+3
                if (0 <= x <= 10) and (0 <= y <= 10):
                    Treffer = 1
        return((x, y))

    def OrtBKG(N):
        Treffer = 0
        while Treffer == 0:
            u1 = np.random.uniform(0, 1)
            u2 = np.random.uniform(0, 1)
            v1 = 2*u1-1
            v2 = 2*u2-1
            s = v1**2+v2**2
            if (s < 1):
```

} x, y sind unabhängig

```
            term = np.sqrt(-(2/s)*np.log(s))
            sig = 3
            c1 = v1*term
            c2 = v2*term
            x = np.sqrt(0.75)*sig*c1+0.5*sig*c2+5
            y = 0.5*sig*c2+5
            if (0 <= x <= 10) and (0 <= y <= 10):
                Treffer = 1
    return((x, y))

# Aufgabenteil a)
Esize = 100000
x = np.random.uniform(0, 1, Esize)
Energie = np.empty(Esize)
Energie = Signal(x)

df = pd.DataFrame()
# Ein DataFrame für die Energiemessung und für die AcceptanceMask da diese gleich lang sind
df['Energy'] = Energie[0:]
# Aufgabenteil b)
x2 = np.random.uniform(0, 1, Esize)
Detektiert = np.zeros(Esize, dtype=bool)

for i in np.arange(0, Esize):
    if x2[i] < Akzeptanz(Energie[i]):
        Detektiert[i] = True

DetEnergie = Energie[Detektiert]
plt.hist(Energie, bins=np.logspace(0, 3), histtype='step',
         label='Energie')
plt.hist(DetEnergie, bins=np.logspace(0, 3), histtype='step', linestyle='--',
         label='Detektierte Energie')
plt.loglog()
plt.legend()
plt.xlabel(r'$E/\mathrm{TeV}$')
plt.ylabel(r'Ereignisse')
plt.savefig('Energie.pdf')
plt.clf()
print(len(DetEnergie))
df['AcceptanceMask'] = Detektiert[0:]
# Aufgabenteil c)
Hits = np.zeros(len(DetEnergie), dtype=int)
for j in np.arange(0, len(DetEnergie)):
    Hits[j] = HitsSim(DetEnergie[j])
plt.hist(Hits, bins=50, range=[0, 100], histtype='step')
plt.xlabel(r'$Anzahl der Hits$')
plt.savefig('Hits.pdf')
plt.clf()
df2 = pd.DataFrame()
# Ein weiters DataFrame da die weiteren Daten nur auf den akzeptierten Energien
# berechnet werden.
df2['NumberofHits'] = Hits[0:]
# Aufgabenteil d)

Ort = np.zeros((len(DetEnergie), 2))
for k in np.arange(0, len(DetEnergie)):
    Ort[k:] = OrtsSim(Hits[k])

plt.hist2d(Ort[:, 0], Ort[:, 1], bins=[50, 50],
           range=[[0, 10], [0, 10]], cmap='viridis')
plt.plot(7, 3, 'rx')
plt.colorbar()
plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
plt.savefig('Ort.pdf')
plt.clf()
df2['x'] = Ort[:, 0]
df2['y'] = Ort[:, 1]

# Aufgabenteil e)
```

```
    BKGSize = 100000
    mu = 2
    sigma = 1
    log10_N_BKG = np.random.normal(mu, sigma, BKGSize)
    N_BKG = 10**log10_N_BKG
    Ort_BKG = np.zeros((BKGSize, 2))

    for l in tqdm(np.arange(0, BKGSize)):
        Ort_BKG[l:] = OrtBKG(N_BKG[l])

    df_BKG = pd.DataFrame()
    df_BKG['NumberofHits'] = N_BKG[0:]
    df_BKG['x'] = Ort_BKG[:, 0]
    df_BKG['y'] = Ort_BKG[:, 1]

    plt.hist2d(Ort_BKG[:, 0], Ort_BKG[:, 1], bins=[50, 50],
               range=[[0, 10], [0, 10]], cmap='viridis')
    plt.colorbar()
    plt.xlabel(r'$x$')
    plt.ylabel(r'$y$')
    plt.savefig('Ort_BKG.pdf')
    plt.clf()

    plt.hist(log10_N_BKG, bins=48, range=[0, 6], histtype='step',
             label='Background')
    plt.legend()
    plt.xlabel(r'$Logarithmus der Anzahl der Hits$')
    plt.savefig('Hits_BKG.pdf')
    plt.clf()
    dfSignal = pd.concat([df, df2], axis=1)
    dfSignal.to_hdf('NeutrinoMC.hdf5', key='Energy')
    df_BKG.to_hdf('NeutrinoMC.hdf5', key='Background')


def Aufgabe2():
    # a)
    samples, labels = make_blobs(n_samples=1000, centers=2, n_features=4, random_state=0)
    samples_x = samples[:, 0]
    samples_y = samples[:, 1]
    plt.scatter(samples[:, 0], samples[:, 1])
    plt.xlabel('Dim 1')
    plt.ylabel('Dim 2')
    plt.savefig('scatterplot_a.png')
    plt.clf()
    # c)
    # Zentrieren um Mittelwert
    mean = np.mean(samples)
    samples_zentriert = samples - mean

    # Analyse
    pca = PCA()
    pca.fit(samples_zentriert)
    samples_pca = pca.transform(samples_zentriert)

    covariance = pca.get_covariance()
    print(covariance)
    eigenval, eigenvec = linalg.eigh(covariance) # eigh diagonalisiert mir die covariance matrix
    print('Nicht sortierte Eigenwerte:')
    print(eigenval)

    # d)

    # scatterplot
    plt.scatter(samples_pca[:, 0], samples[:, 1])
    plt.xlabel('Dim 1')
    plt.ylabel('Dim 2')
    plt.savefig('scatterplot_d.png')
    plt.clf()
    # histogram
    plt.subplot(2, 2, 1)
    plt.hist(samples_pca[:, 0], color='b', density=True, bins=40, label='Dim 1', alpha=0.6)
```

```
    plt.xlabel('Dim 1')
    plt.ylabel('W.-keit')
    plt.tight_layout()

    plt.subplot(2, 2, 2)
    plt.hist(samples_pca[:, 1], color='b', density=True, bins=40, label='Dim 2', alpha=0.6)
    plt.xlabel('Dim 2')
    plt.ylabel('W.-keit')
    plt.tight_layout()

    plt.subplot(2, 2, 3)
    plt.hist(samples_pca[:, 2], color='b', density=True, bins=40, label='Dim 3', alpha=0.6)
    plt.xlabel('Dim 3')
    plt.ylabel('W.-keit')
    plt.tight_layout()

    plt.subplot(2, 2, 4)
    plt.hist(samples_pca[:, 3], color='b', density=True, bins=40, label='Dim 4', alpha=0.6)
    plt.xlabel('Dim 4')
    plt.ylabel('W.-keit')
    plt.tight_layout()
    plt.savefig('histogrammi.png')


if __name__ == '__main__':
    from sklearn.datasets import make_blobs
    import matplotlib.pyplot as plt
    from sklearn.decomposition import PCA
    import numpy as np
    import pandas as pd
    from tqdm import tqdm
    from scipy import linalg

    Aufgabe1()
    Aufgabe2()
```

Aufgabe 13

SHD Blatt 5 (Übung bei Simone & Alicia)

Yvonne Kaspor
Robert Appel
Julian Schröer

a) Neutrinofluss: $\phi = \phi_0 \cdot \left(\frac{E}{TeV}\right)^{-\gamma}$

$E_{min} = 1\,TeV$, $E_{max} = \infty$

$$A = \int_{E_{min}=1TeV}^{\infty} \phi_0 \left(\frac{E}{TeV}\right)^{-\gamma} dE$$

$$= \left[\frac{\phi_0}{(-\gamma+1)} \left(\frac{E}{TeV}\right)^{-\gamma+1}\right]_{1TeV}^{\infty}$$

$$= 0 - \frac{\phi_0}{(-\gamma+1)} 1^{-\gamma+1} = \frac{-\phi_0}{(-\gamma+1)}$$

$$A(x) = \int_{E_{min}=1TeV}^{x\,TeV} \phi_0 \left(\frac{E}{TeV}\right)^{-\gamma} dE$$

$$= \left[\frac{\phi_0}{(-\gamma+1)} \left(\frac{E}{TeV}\right)^{-\gamma+1}\right]_{1TeV}^{x\,TeV}$$

$$= \frac{\phi_0}{(-\gamma+1)} x^{-\gamma+1} - \frac{\phi_0}{(-\gamma+1)} = \frac{-\phi_0}{(-\gamma+1)} \left(-x^{-\gamma+1} + 1\right)$$

$$r(x) = \frac{A(x)}{A} = \left(\frac{-\phi_0}{(-\gamma+1)}\right)^{-1} \left(\frac{\phi_0}{(-\gamma+1)} \left(-x^{-\gamma+1} + 1\right)\right)$$

$$= -x^{-\gamma+1} + 1 = r$$

$\Rightarrow$ mit spektralem Index $\gamma = 2{,}7$

$\Rightarrow$ $r - 1 = -x^{-1{,}7}$

$\Rightarrow$ $(1-r)^{\frac{1}{1{,}7}} = x$ //

✓

#Hits

$$\mathcal{N}\left(10 \cdot \frac{E}{TeV}, 2 \cdot \frac{E}{TeV}\right)$$

mittelwert $10 \frac{E}{TeV}$

standardabweichung $2 \frac{E}{TeV}$

mit Polarmethode

$$\mathcal{N} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$\sigma = 2$

$\mu = 10$

$$F(r) = 1 - \frac{-(r-\mu)^2}{2\sigma^2}$$

$$\vec{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} t \\ t \\ t \end{pmatrix} \quad t \in$$

$$S \geq e^{\frac{-(r-\mu)^2}{2\sigma^2}}$$

$$\ln s = \frac{-(r-\mu)^2}{2\sigma^2}$$

$$2\sigma^2 \ln s = -(r-\mu)^2$$

$$\sqrt{-2\sigma^2 \ln s} = r - \mu$$

$$\sqrt{-2\sigma^2 \ln s} + \mu = r$$