Aufgabe 2

In [1]:

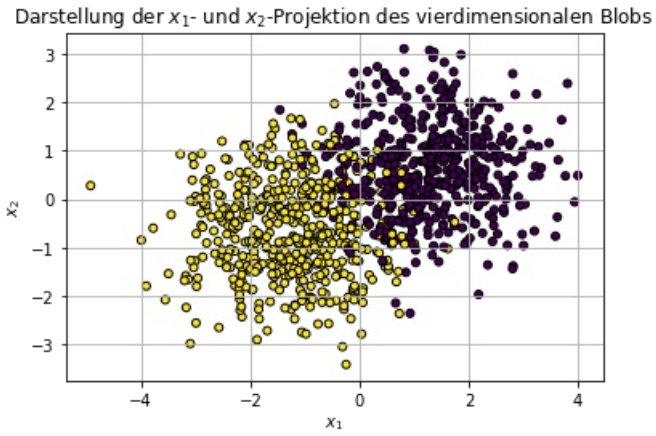
```
from sklearn import datasets
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np
```

Teilaufgabe a)

Wir erstellen einen Datensatz mit vier Merkmalen und zwei Klassen. Dabei wird die eine gelb und die andere schwarzblau dargestellt. Zuerst erstellen wir einen Scatterplot der x_1- und der x_2-Projektion der Daten.

In [2]:

```
x,y = datasets.make_blobs(n_samples=1000, centers=2,
                          n_features=4, random_state=0)
x = x - x.mean(axis=0) # Zentrierung
plt.scatter(x[:, 0], x[:, 1], marker='o', c=y,
            s=25, edgecolor='k')
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')
plt.title('Darstellung der $x_1$- und $x_2$-Projektion des vierdimensionalen Blobs')
plt.grid()
plt.show()
plt.clf()
```



Teilaufgabe b)

Beschreibung der Hauptkomponentenanalyse:

Die Hauptkomponentenanalyse dient der Strukturierung, Vereinfachung und Veranschaulichung von umfangreichen Datensätzen, indem eine Vielzahl von Merkmalen durch eine geringere Anzahl aussagekräftiger Linearkombinationen genähert wird. Ein Datensatz in Form einer Matirx bzw. N Datenpunkte in d Dimensionen werden mit einer Hauptachsentransformation auf einen k dimensionalen Unterraum projeziert, sodass möglichst wenig Information verloren geht und bestehende Redundanz in Form von Korrelation zusammengefasst wird.

1. Zentrierung der Daten auf ihren Mittelwert.
2. Berechnung der Kovarianzmatrix aus der Datenmatrix A.
3. Berechnung der Eigenwerte und Eigenvektoren der Kovarianzmatrix.
4. Auswahl der k größten Eigenwerte und zugehörigen Eigenvektoren.
5. Aufstellen einer $d \times k$ Matrix W mit den k Eigenvektoren als Spalten.
6. Anwendung von W auf die Datenmatrix X.

Teilaufgabe c)

Wir führen eine Hauptkomponentenanalyse (PCA) des gegebenen Datensatzes mit dem Paket sklearn durch. Außerdem bestimmen wir die Kovarianzmatrix des Datensatzes und geben die Eigenwerte (EW) aus. Der größte EW ist so zu interpretieren, dass der dazugehörige Eigenvektor (EV) die Daten sehr gut trennen kann. Dabei wird eine Linearkombination gebildet; da die anderen drei EW jedoch klein sind im Vergleich zum ersten EW gibt es kaum eine Beimischung der anderen EV. Im Allgemeinen kann man sagen, dass der EV zum größten EW die Intervarianz nach Transformation maximiert, also die Klassen möglichst gut trennt.

In [3]:

```
pca = PCA(n_components = 4)
pca.fit(x)
x = pca.fit_transform(x)

c = np.cov(x, rowvar=False)
eigw = np.linalg.eigvals(c)
print(eigw)
```

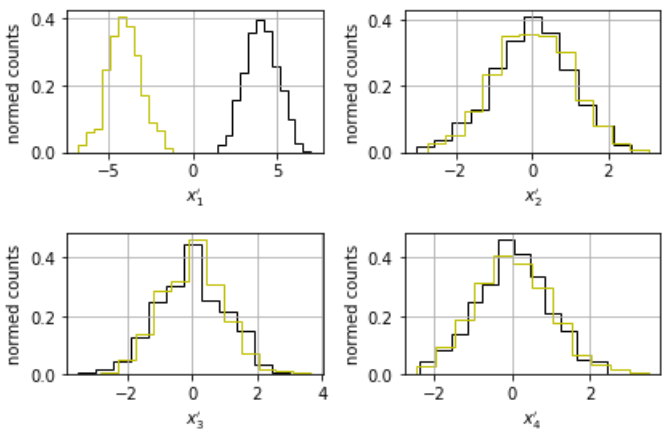
```
[ 17.51933024   0.89875061   0.99958442   0.98813673]
```

Teilaufgabe d)

Der durch die PCA transformierte Datensatz wird x' genannt. Dieser Datensatz wird nun in jeder Dimension histogrammiert. Die Daten werden wieder nach Klasse farblich gekennzeichnet. Es ist klar zu erkennen, dass die erste Hauptkomponente x'_1 die Klassen am besten trennt. Dies wird auch klar durch den darunter angefertigten Scatterplot der Projektion auf die beiden ersten Dimensionen bzw. Hauptkomponenten. In den anderen Hauptkomponenten ist keine Trennung zu erkennen.

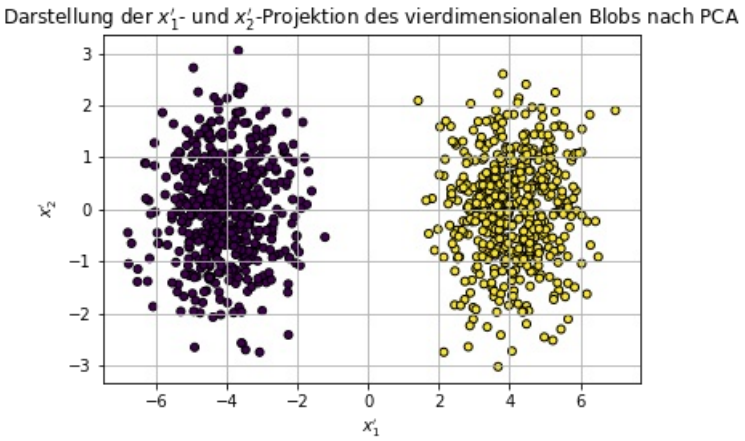
In [4]:

```
x1, x2, x3, x4 = zip(*x)
x_array = [x1, x2, x3, x4]

i = 0
for count in x_array:
    i += 1
    xMembersFirst = count*y
    xMembersFirst = xMembersFirst[xMembersFirst != 0]
    xMembersSecond = count*(((y+1)-2)*(-1))
    xMembersSecond = xMembersSecond[xMembersSecond != 0]

    plt.subplot(2,2,i)
    plt.hist(xMembersFirst, normed=True, bins = 12, histtype='step', color='black')
    plt.hist(xMembersSecond, normed=True, bins = 12, histtype='step', color='y')
    plt.ylabel('normed counts')
    plt.xlabel(r'$x_{%i}^{\prime}$' %i)
    plt.grid()

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



In [5]:

```
plt.scatter(x[:, 0], x[:, 1], marker='o', c=y,
            s=25, edgecolor='k')
plt.xlabel(r'$x_{1}^{\prime}$')
plt.ylabel(r'$x_{2}^{\prime}$')
plt.title('Darstellung der $x_{1}^{\prime}$- und $x_{2}^{\prime}$-Projektion des vierdimensionalen
Blobs nach PCA')
plt.grid()
plt.show()
```



In [ ]: