

Exercise 15

June 7, 2022

1 Exercise 15

1.0.1 a)

```
[1]: import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

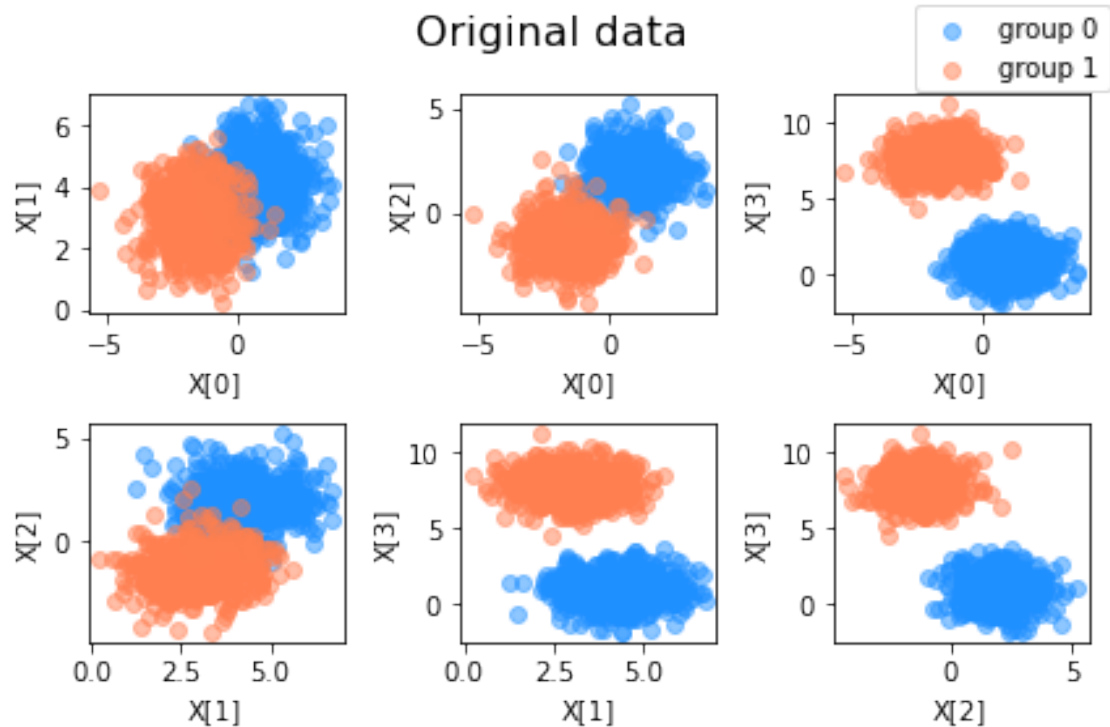
# generating dataset X: data, group:
X, group = make_blobs(n_samples = 1000, centers = 2, n_features = 4,
    ↪random_state = 0)

# Scatter-plot of any 2 dimensions
fig, axs = plt.subplots(2, 3)

# help function to shorten code
def plotfunc(ax1, ax2, row1, row2):
    axs[ax1, ax2].scatter(X[group == 0][:,row1], X[group == 0][:,row2], color =
    ↪"dodgerblue", alpha = 0.5, label = "group 0")
    axs[ax1, ax2].scatter(X[group == 1][:,row1], X[group == 1][:,row2], color =
    ↪"coral", alpha = 0.5, label = "group 1")
    axs[ax1, ax2].set_xlabel("X[{}]".format(row1))
    axs[ax1, ax2].set_ylabel("X[{}]".format(row2))

plotfunc(0, 0, 0, 1)
plotfunc(0, 1, 0, 2)
plotfunc(0, 2, 0, 3)
plotfunc(1, 0, 1, 2)
plotfunc(1, 1, 1, 3)
plotfunc(1, 2, 2, 3)

fig.suptitle('Original data', fontsize=16)
handles, labels = axs[0,0].get_legend_handles_labels()
fig.legend(handles, labels)
plt.tight_layout()
```



1.0.2 b)

```
[2]: from sklearn.decomposition import PCA
```

```
pca = PCA()
```

```
# transformation of X via the pca
```

```
X_tf = pca.fit_transform(X)
```

```
# Eigenvalues of the covariance matrix of transformed data
```

```
eig = pca.explained_variance_
```

```
print("Eigenvalues: ", eig)
```

```
Eigenvalues: [17.51933024  0.99958442  0.98813673  0.89875061]
```

The first eigenvalue is significantly bigger than the others, so only the first feature of the transformed dataset should be kept. All others should be discarded, since their eigenvalues are smaller and similar to each other.

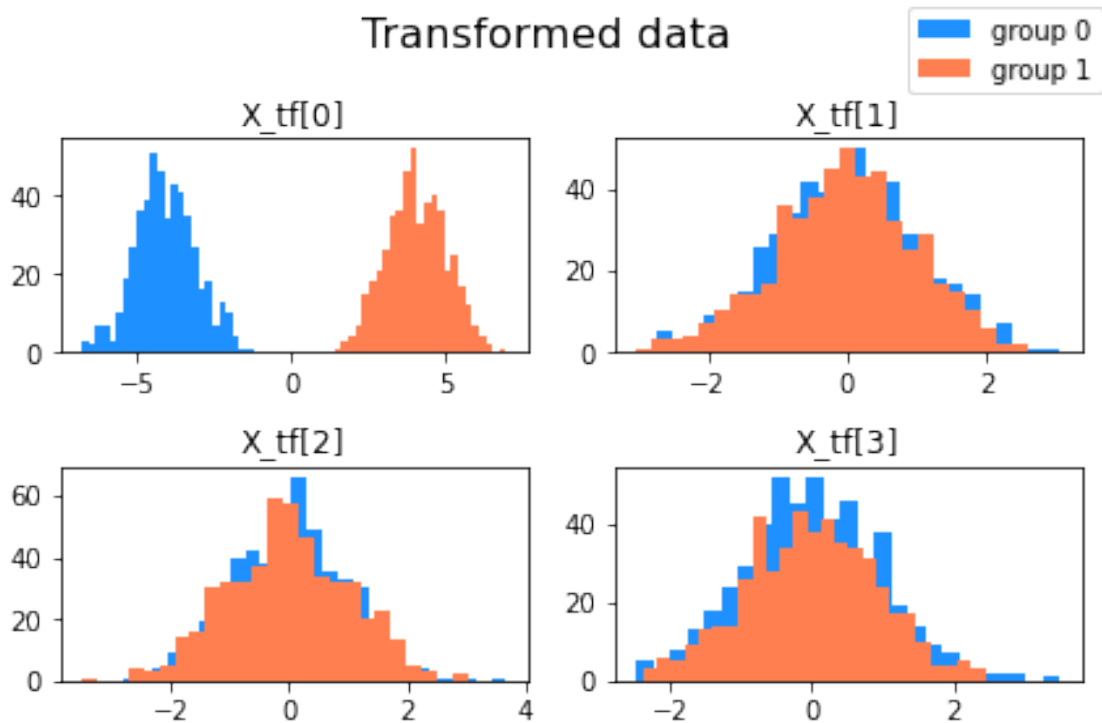
1.0.3 c)

```
[3]: # Histogram plot
fig, axs = plt.subplots(2, 2)

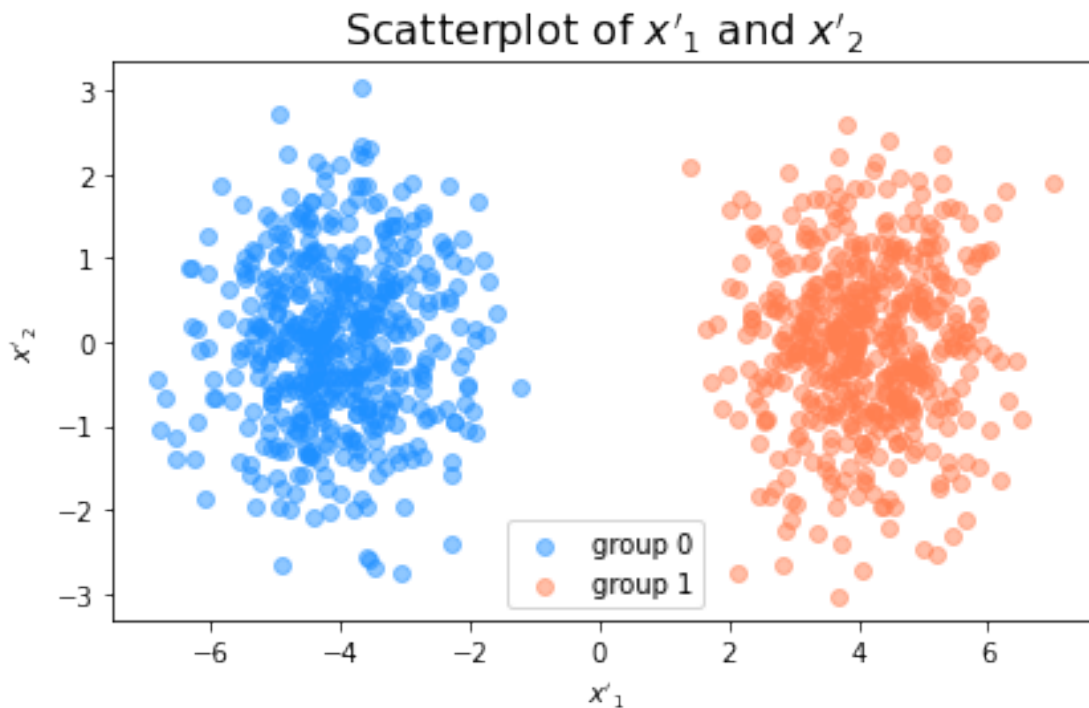
def plotfunc(ax1, ax2, row):
    axs[ax1,ax2].hist(X_tf[group == 0][:,row], bins = 25, color = "dodgerblue",
    ↪label = "group 0")
    axs[ax1,ax2].hist(X_tf[group == 1][:,row], bins = 25, color = "coral",
    ↪label = "group 1")
    axs[ax1,ax2].set_title("X_tf[{}]".format(row))

plotfunc(0, 0, 0)
plotfunc(0, 1, 1)
plotfunc(1, 0, 2)
plotfunc(1, 1, 3)

fig.suptitle('Transformed data', fontsize=16)
handles, labels = axs[0,0].get_legend_handles_labels()
fig.legend(handles, labels)
plt.tight_layout()
```



```
[4]: fig2 = plt.subplot()
fig2.scatter(X_tf[group == 0][:,0], X_tf[group == 0][:,1],
             color = "dodgerblue", alpha = 0.5, label = "group 0")
fig2.scatter(X_tf[group == 1][:,0], X_tf[group == 1][:,1],
             color = "coral", alpha = 0.5, label = "group 1")
fig2.set_title(r"Scatterplot of  $x'_1$  and  $x'_2$ " , fontsize=16)
fig2.set_xlabel(r" $x'_1$ ")
fig2.set_ylabel(r" $x'_2$ ")
handles, labels = fig2.get_legend_handles_labels()
fig2.legend(handles, labels)
plt.tight_layout()
```



```
[5]: # All dimensions of the transformed dataset in a scatter plot

# Plot (same as in a))
fig, axs = plt.subplots(2, 3)

def plotfunc(ax1, ax2, row1, row2):
    axs[ax1, ax2].scatter(X_tf[group == 0][:,row1], X_tf[group == 0][:,row2],
                          color = "dodgerblue", alpha = 0.5, label = "group 0")
    axs[ax1, ax2].scatter(X_tf[group == 1][:,row1], X_tf[group == 1][:,row2],
                          color = "coral", alpha = 0.5, label = "group 1")
    axs[ax1, ax2].set_xlabel("X_tf[{}]" .format(row1))
```

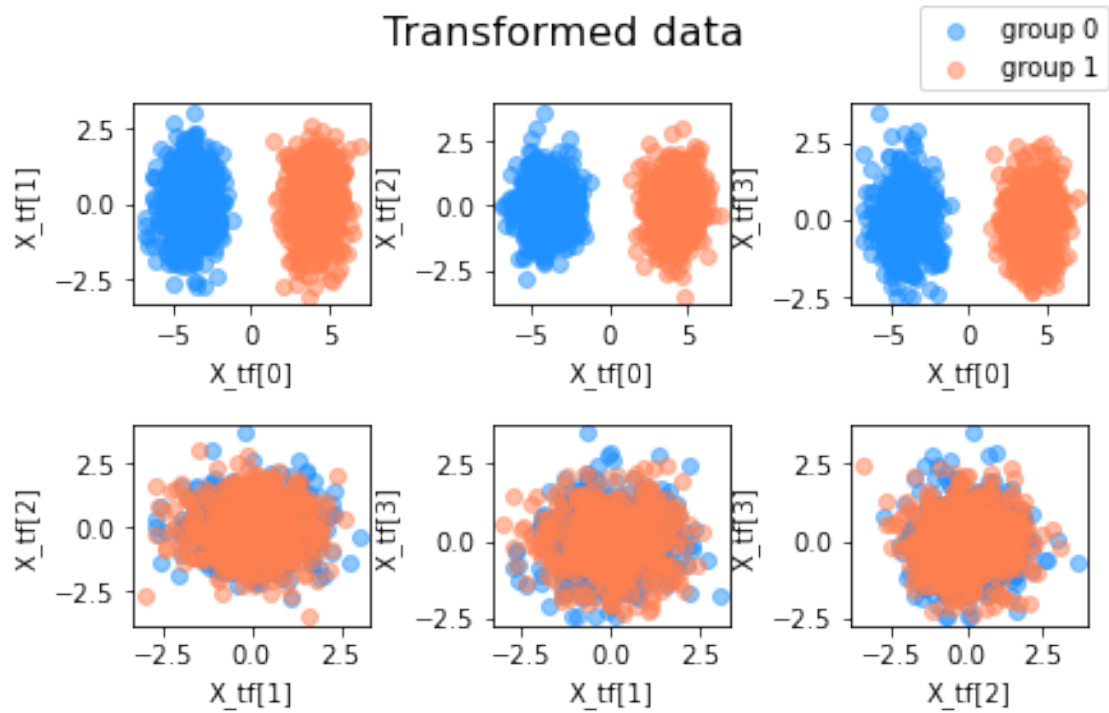
```

    axs[ax1, ax2].set_ylabel("X_tf[{}]" .format(row2))

    plotfunc(0, 0, 0, 1)
    plotfunc(0, 1, 0, 2)
    plotfunc(0, 2, 0, 3)
    plotfunc(1, 0, 1, 2)
    plotfunc(1, 1, 1, 3)
    plotfunc(1, 2, 2, 3)

    fig.suptitle('Transformed data', fontsize=16)
    handles, labels = axs[0,0].get_legend_handles_labels()
    fig.legend(handles, labels)
    plt.tight_layout()

```



[]: