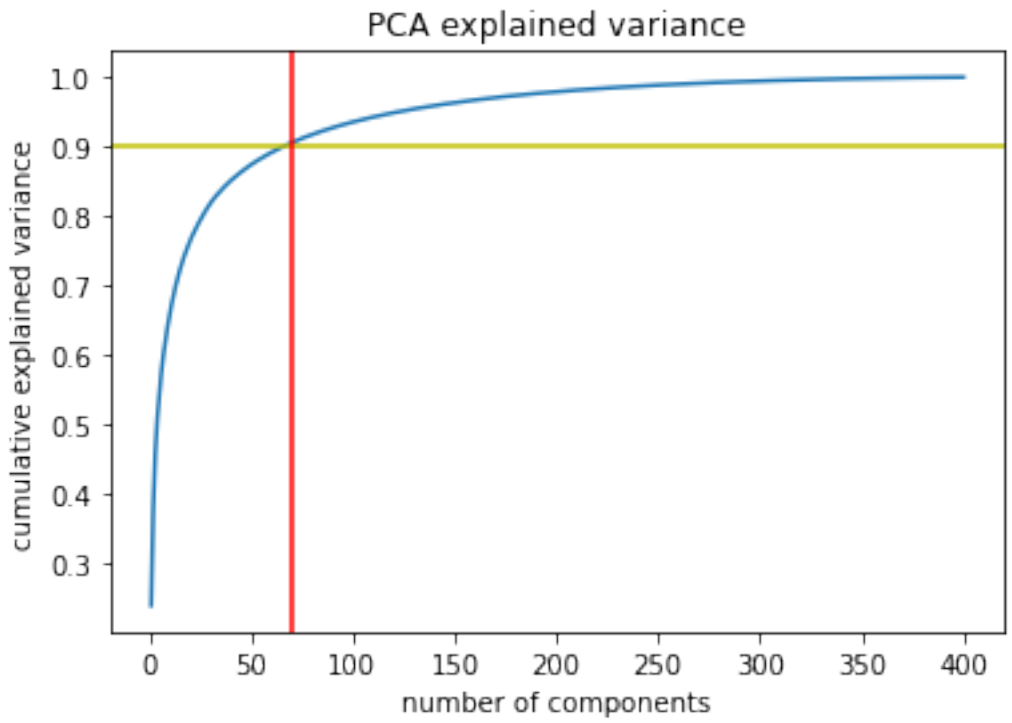
PD6

June 16, 2021

[1]:
```
from sklearn.datasets import fetch_olivetti_faces
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
```

[2]:
```
from numpy.random import RandomState

def plot_gallery(title, images, n_col=3, n_row=2, cmap=plt.cm.gray,␣
 ↪img_shape=(64,64)):
    plt.figure(figsize=(2. * n_col, 2.26 * n_row))
    plt.suptitle(title, size=16)
    for i, comp in enumerate(images):
        plt.subplot(n_row, n_col, i + 1)
        vmax = max(comp.max(), -comp.min())
        plt.imshow(comp.reshape(img_shape), cmap=cmap,
                   interpolation='nearest',
                   vmin=-vmax, vmax=vmax)
        plt.xticks(())
        plt.yticks(())
    plt.subplots_adjust(0.01, 0.05, 0.99, 0.93, 0.04, 0.)

faces, _ = fetch_olivetti_faces(return_X_y=True, shuffle=True,
                                random_state=RandomState(0))
```

[3]:
```
from sklearn.decomposition import PCA

faces, _ = fetch_olivetti_faces(return_X_y=True, shuffle=True)
pca = PCA().fit(faces)
plt.plot(range(1, len(pca.explained_variance_ratio_)+1), np.cumsum(pca.
 ↪explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.title("PCA explained variance")
plt.axhline(y=0.90, color="y")
plt.axvline(x=70, color="r")
plt.show()
```

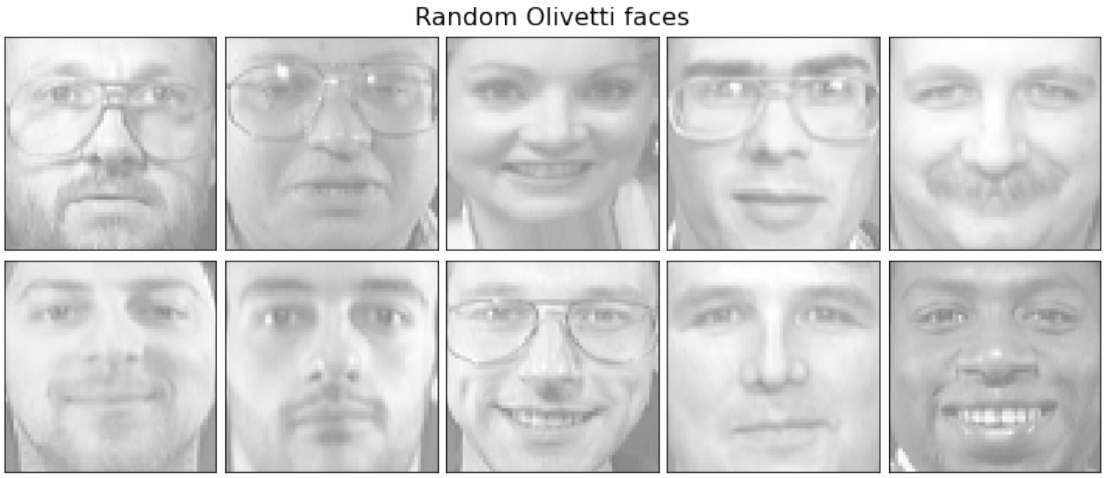


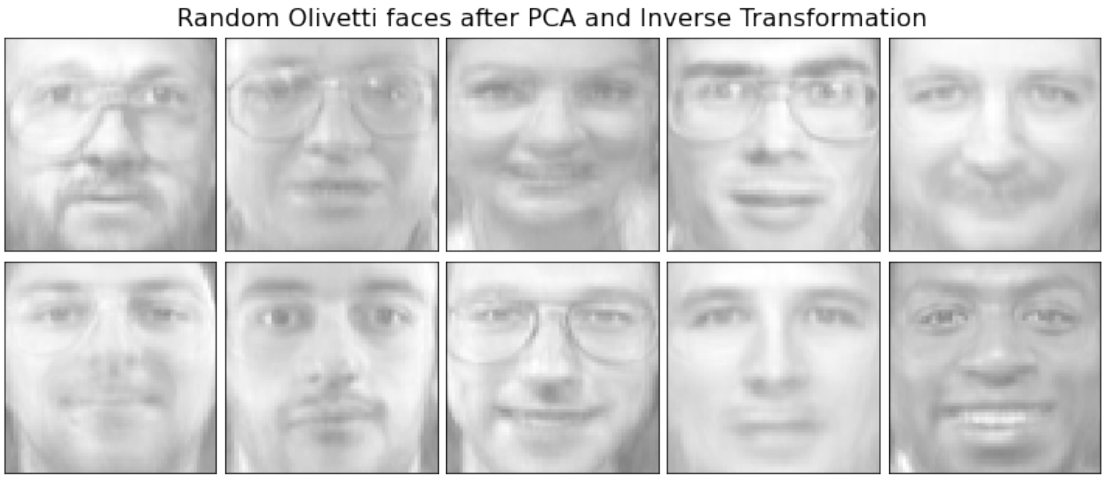
Zastosuję PCA dla 70 komponentów, ponieważ osiągnę wówczas około 90% wyjaśnialnej wariancji.

[4]:
```
pca = PCA(n_components=70)
faces_pca = pca.fit_transform(faces)

print(f"Stopień konwersji: {faces[0].size / faces_pca[0].size}")
IT = pca.inverse_transform(faces_pca)
```

```
Stopień konwersji: 58.51428571428571
```

[5]:
```
plot_gallery("Random Olivetti faces", faces[:10], n_col = 5, n_row=2)
plot_gallery("Random Olivetti faces after PCA and Inverse Transformation", IT[:
↪10], n_col = 5, n_row=2)
```



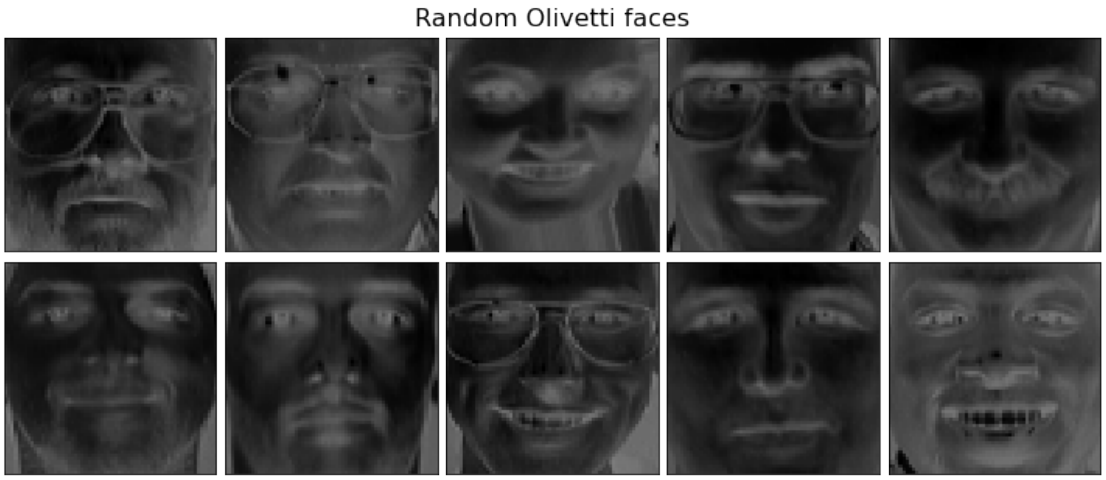


[6]:
```
from sklearn.metrics import mean_squared_error

def RMSE(mod):
    rmse = [0] * 400
    for i, j, k in zip(faces, mod, range(400)):
        rmse[k] = mean_squared_error(i, j, squared=False)

    print(f"Uśredniony błąd RMSE: {np.mean(rmse)}")

RMSE(IT)
```

```
Uśredniony błąd RMSE: 0.042291391640901566
```

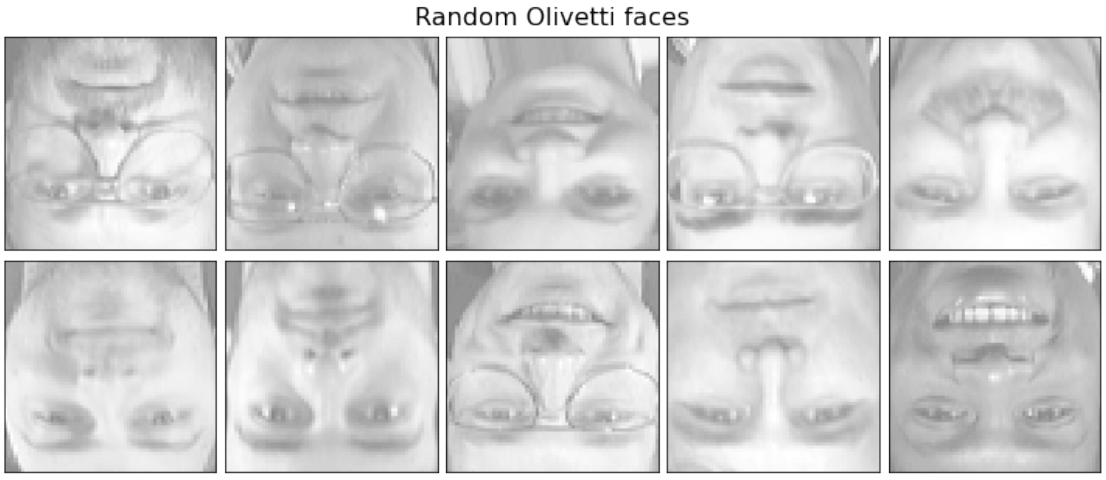
[7]:
```
# zamiana koloru
faces_minus = - faces

plot_gallery("Random Olivetti faces", faces_minus[:10], n_col = 5, n_row=2)
RMSE(faces_minus)
```

```
Uśredniony błąd RMSE: 1.1400033235549927
```



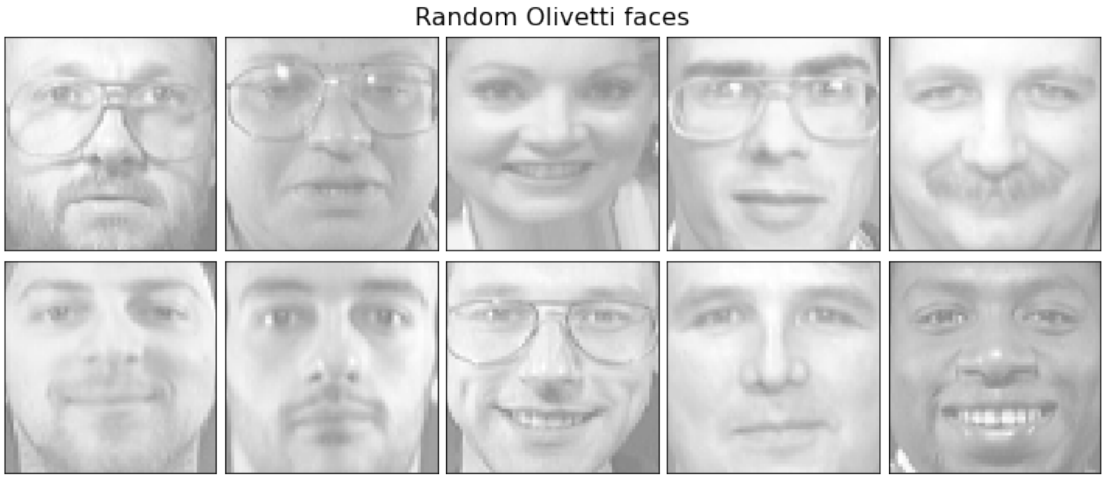
[8]:
```
# Odwrocone do gory nogami
faces_flip = []

for i in faces: faces_flip.append(i[::-1])
plot_gallery("Random Olivetti faces", faces_flip[:10], n_col = 5, n_row=2)
RMSE(faces_flip)
```

```
Uśredniony błąd RMSE: 0.22976413369178772
```



[9]:
```
# Zwiekszony kontrast
faces_mul = 10 * faces

plot_gallery("Random Olivetti faces", faces_mul[:10], n_col = 5, n_row=2)
RMSE(faces_mul)
```

```
Uśredniony błąd RMSE: 5.130014419555664
```

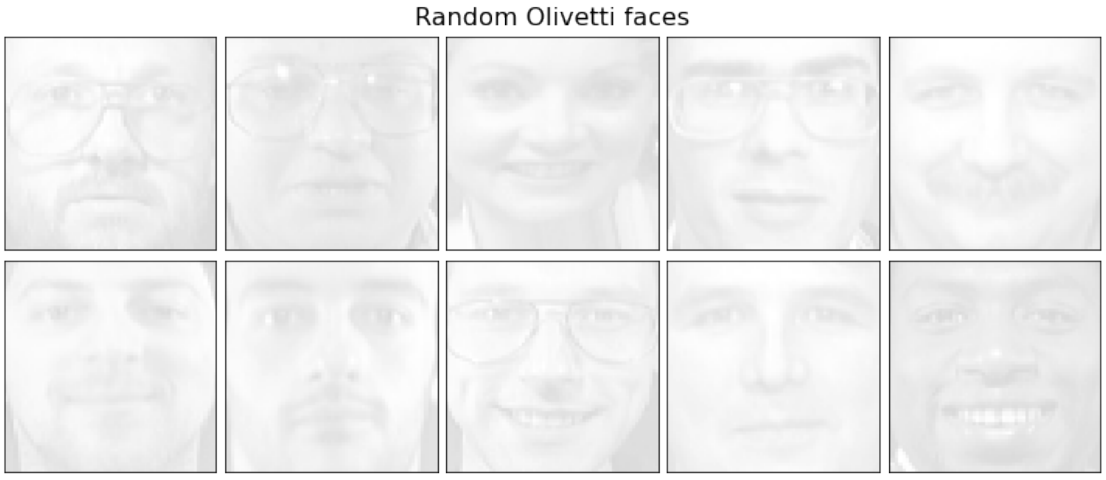


[10]:
```
# Przyciemnienie
faces_dark = faces - 2

plot_gallery("Random Olivetti faces", faces_dark[:10], n_col = 5, n_row=2)
RMSE(faces_dark)
```

```
Uśredniony błąd RMSE: 2.0
```

[11]:
```
# Rozjasnianie
faces_light = faces + 2

plot_gallery("Random Olivetti faces", faces_light[:10], n_col = 5, n_row=2)
RMSE(faces_light)
```

```
Uśredniony błąd RMSE: 2.0
```



Wnioski: Najmniejszy błąd RMSE osiągnęły zdjęcia po transformacji odwrotnej. Mimo znaczącego stopnia kompresji, nie były one dużo bardziej rozmazane. Osoby na tych zdjęciach miały obwódki wokół oczu, jakby okulary.