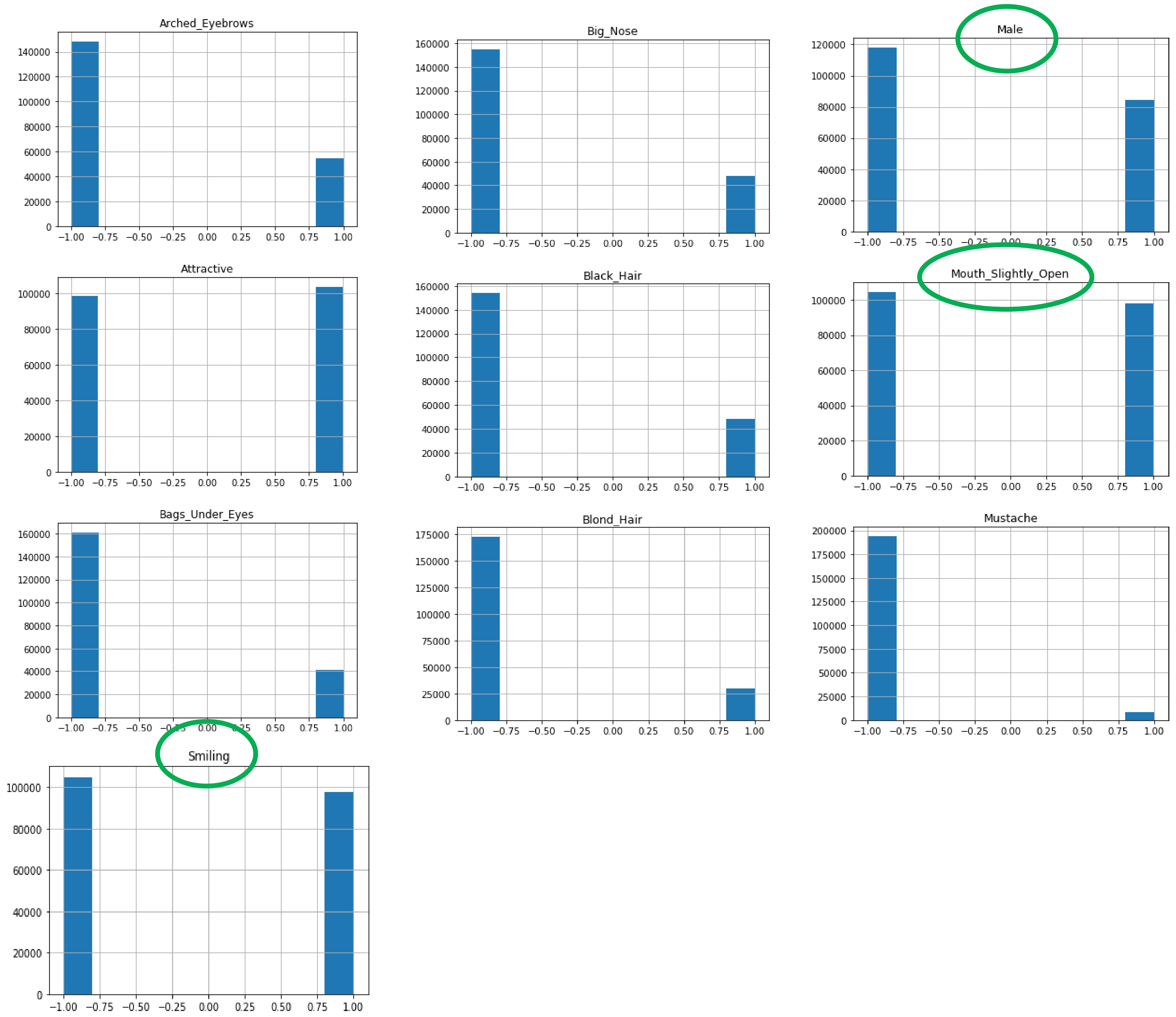
Ausgangslage

- Datensatz mit 202‘599 Gesichtsbilder von Celebrities (JPG, 178x218)
- 40 binäre Features pro Bild
- Augen, Mund etc. normalisiert (gleiche Position)
- Training eines neuronalen Netzes zur Feature-Erkennung

Voranalyse

['5_o_Clock_Shadow', 'Arched_Eyebrows', 'Attractive', 'Bags_Under_Eyes', 'Bald', 'Bangs', 'Big_Lips', 'Big_Nose', 'Black_Hair', 'Blond_Hair', 'Blurry', 'Brown_Hair', 'Bushy_Eyebrows', 'Chubby', 'Double_Chin', 'Eyeglasses', 'Goatee', 'Gray_Hair', 'Heavy_Makeup', 'High_Cheekbones', **'Male'**, **'Mouth_Slightly_Open'**, 'Mustache', 'Narrow_Eyes', 'No_Beard', 'Oval_Face', 'Pale_Skin', 'Pointy_Nose', 'Receding_Hairline', 'Rosy_Cheeks', 'Sideburns', **'Smiling'**, 'Straight_Hair', 'Wavy_Hair', 'Wearing_Earrings', 'Wearing_Hat', 'Wearing_Lipstick', 'Wearing_Necklace', 'Wearing_Necktie', 'Young']



Datenaufbereitung

- Bilder in Graustufen konvertieren zur Dimensionsreduktion
- Reduktion der Bildergrösse zu 89x109
- Transformation der Bilder in ein Dataframe mit Pixelwerten
  - Aufteilung in 1'000er Bins (CSV, ~50MB pro File)

```python
import os
from PIL import Image
import numpy as np
import pandas as pd

faces = pd.DataFrame()

folder = "celeba-dataset"

subfolder = "images"

b=89
h=109

start=1
n=0

while True:
    while True:
        if start<10:
            file="00000"+str(start)+".jpg"
        elif start<100:
            file="0000"+str(start)+".jpg"
        elif start<1000:
            file="000"+str(start)+".jpg"
        elif start<10000:
            file="00"+str(start)+".jpg"
        elif start<100000:
            file="0"+str(start)+".jpg"
        else:
            file=str(start)+".jpg"
        im = Image.open(os.path.abspath(os.path.join(os.getcwd(), folder, subfolder, file)))
        im = im.convert("L")
        im = im.resize((b,h))
        img=np.array(im)
        img=pd.Series(img.flatten())
        faces=faces.append(img,ignore_index=True)
        faces.loc[n,"id"]=file
        n+=1
        if n==1000:
            n=0
            #faces.to_csv("faces_"+str(start)+".csv",header=True,index=False)
            start+=1
            print(faces)
            faces = pd.DataFrame()
            break
        start+=1
```

faces_1000.csv
faces_2000.csv
faces_3000.csv
faces_4000.csv
faces_5000.csv
faces_6000.csv
faces_7000.csv
faces_8000.csv
faces_9000.csv
faces_10000.csv
faces_11000.csv
faces_12000.csv
faces_13000.csv
faces_14000.csv
faces_15000.csv
faces_16000.csv
faces_17000.csv
faces_18000.csv
faces_19000.csv
faces_20000.csv

Beispiel Dataframe

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
2	233	233	233	233	233	233	233	233	232	232	233	233	232	232	232	234	234	236	236	235	236	235	236	237	236	236	236	236	236	237	238	238	236
3	53	55	54	62	71	83	92	102	112	118	122	128	130	130	123	117	110	101	90	79	69	66	61	71	79	87	90	94	96	95	90	85	80
4	255	255	255	255	255	255	255	255	254	254	254	254	252	252	253	254	254	254	254	253	254	254	254	252	249	251	254	255	254	253	252	252	255
5	66	67	40	88	150	178	75	71	61	65	136	97	47	38	37	19	22	28	17	38	10	29	35	21	6	10	20	8	3	20	11	17	9
6	184	185	186	186	184	192	196	213	217	207	199	192	191	205	229	231	232	233	234	234	233	233	233	233	235	235	200	168	188	188	188	189	191
7	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	254	251	252	253	253	254	254	254	254	253	254	254	254	253
8	86	87	87	88	84	87	87	114	160	147	90	96	101	103	102	95	121	165	101	110	100	110	103	98	109	122	169	129	95	106	105	109	104
9	27	27	28	29	34	25	145	167	118	41	49	42	46	40	38	37	34	36	33	93	171	162	45	43	42	40	41	38	40	51	43	166	170
10	77	77	78	80	81	84	87	88	92	96	102	105	109	111	109	111	105	100	97	91	80	67	64	61	67	66	64	64	68	70	77	84	73
11	4	5	6	6	6	6	6	6	6	7	8	8	8	9	10	10	10	11	12	12	13	15	16	17	18	20	23	25	26	30	33	34	36
12	230	230	230	230	231	232	231	230	229	229	229	229	230	231	233	234	232	232	232	232	233	232	231	231	231	232	232	233	234	233	232	232	232
13	2	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	138	138	138	138	148	150	152	151	147	142	141	143	148	152	154	153	151	148	143	140	120	93	71	49	31	30	30	40	37	53	43	45	47
15	175	180	183	188	190	191	195	195	198	199	200	204	205	207	206	205	204	208	201	148	133	130	127	126	124	122	119	121	119	120	117	119	116
16	178	178	178	178	176	176	176	176	176	173	171	160	166	161	148	120	114	123	143	163	172	158	126	118	120	120	123	112	116	118	105	114	124
17	23	23	23	23	23	23	23	23	23	23	23	23	24	24	24	24	23	23	24	26	26	26	26	26	27	27	27	27	27	28	29	29	28
18	197	196	196	195	195	194	194	193	196	195	194	194	194	193	192	192	192	191	190	190	190	190	190	190	190	189	189	188	188	188	188	188	187
19	212	212	212	212	212	214	215	216	216	219	218	220	221	223	223	222	224	224	224	224	224	223	224	225	224	224	224	224	223	223	221	219	222
20	130	132	130	146	180	226	244	248	237	224	216	207	201	192	188	183	193	209	220	228	239	251	254	254	252	250	243	234	204	183	183	184	183
21	1	2	1	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	2	4	3	4	3	2	4	4	1	3	2

Netzarchitektur / Training

```python
model = Sequential()

model.add(Convolution2D(8, (3, 3), input_shape=(109, 89, 1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Convolution2D(8, (3, 3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Convolution2D(16, (3,3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Convolution2D(16, (3, 3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Convolution2D(32, (3,3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Convolution2D(32, (3, 3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dropout(0.2))
model.add(Dense(2240))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Dropout(0.2))
model.add(Dense(1120))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Dropout(0.2))
model.add(Dense(1120))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Dropout(0.2))
model.add(Dense(1))
model.add(BatchNormalization())
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 107, 87, 8)        80
_________________________________________________________________
batch_normalization_1 (Batch (None, 107, 87, 8)        32
_________________________________________________________________
activation_1 (Activation)    (None, 107, 87, 8)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 105, 85, 8)        584
_________________________________________________________________
batch_normalization_2 (Batch (None, 105, 85, 8)        32
_________________________________________________________________
activation_2 (Activation)    (None, 105, 85, 8)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 52, 42, 8)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 50, 40, 16)        1168
_________________________________________________________________
batch_normalization_3 (Batch (None, 50, 40, 16)        64
_________________________________________________________________
activation_3 (Activation)    (None, 50, 40, 16)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 48, 38, 16)        2320
_________________________________________________________________
batch_normalization_4 (Batch (None, 48, 38, 16)        64
_________________________________________________________________
activation_4 (Activation)    (None, 48, 38, 16)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 24, 19, 16)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 22, 17, 32)        4640
_________________________________________________________________
batch_normalization_5 (Batch (None, 22, 17, 32)        128
_________________________________________________________________
activation_5 (Activation)    (None, 22, 17, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 20, 15, 32)        9248
_________________________________________________________________
batch_normalization_6 (Batch (None, 20, 15, 32)        128
_________________________________________________________________
activation_6 (Activation)    (None, 20, 15, 32)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 7, 32)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2240)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2240)              0
_________________________________________________________________
dense_1 (Dense)              (None, 2240)              5019840
_________________________________________________________________
batch_normalization_7 (Batch (None, 2240)              8960
_________________________________________________________________
activation_7 (Activation)    (None, 2240)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 2240)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1120)              2509920
_________________________________________________________________
batch_normalization_8 (Batch (None, 1120)              4480
_________________________________________________________________
activation_8 (Activation)    (None, 1120)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 1120)              0
_________________________________________________________________
dense_3 (Dense)              (None, 1120)              1255520
_________________________________________________________________
batch_normalization_9 (Batch (None, 1120)              4480
_________________________________________________________________
activation_9 (Activation)    (None, 1120)              0
_________________________________________________________________
dropout_4 (Dropout)          (None, 1120)              0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 1121
_________________________________________________________________
batch_normalization_10 (Batc (None, 1)                 4
_________________________________________________________________
activation_10 (Activation)   (None, 1)                 0
=================================================================
Total params: 8,822,813
Trainable params: 8,813,627
Non-trainable params: 9,186
_________________________________
```

```python
#Parameter

goal="Smiling"
anzepoch=1
modelname="defmod"

#Trainingsdateinummern auflisten [2000,3000,4000,5000 etc.]

dateiset=[32000]

from keras.models import load_model as _loadm

try:
    model = _loadm(modelname+".h5")
except:
    print("NO SAVED MODEL FOUND!")

for m in dateiset:
    datei="faces_"+str(m)+".csv"
    file=os.path.abspath(os.path.join(os.getcwd(),datei))
    data=pd.DataFrame(pd.read_csv(file))
    data2=attrs[["image_id",goal]]
    data2=data2.replace(-1, 0)
    datadef=data.merge(data2,left_on="id", right_on="image_id")
    datadef=datadef.drop(columns=["id","image_id"])
    [train,test]=sklms.train_test_split(datadef)
    train_x=train.loc[:, train.columns != goal]
    train_x=np.array(train_x)
    train_x=train_x.reshape(750,109,89,1)
    train_y=np.array(train[goal])
    test_x=test.loc[:, test.columns != goal]
    test_x=np.array(test_x)
    test_x=test_x.reshape(250,109,89,1)
    test_y=np.array(test[goal])
    trained_on=datei
    print("Training on",trained_on,"...")
    model.fit(train_x,train_y,epochs=anzepoch, batch_size=25,validation_data=[test_x, test_y])
    model.save(modelname+".h5")
```

Resultate

Precision für Feature 'Male'

Kat. 0	94.7%
Kat. 1	93.3%

Precision für Feature 'Mouth_Slightly_Open'

Kat. 0	93.4%
Kat. 1	88.2%

Precision für Feature 'Smiling'

Kat. 0	90.7%
Kat. 1	90.9%

Recall für Feature 'Male'

Kat. 0	95.2%
Kat. 1	92.6%

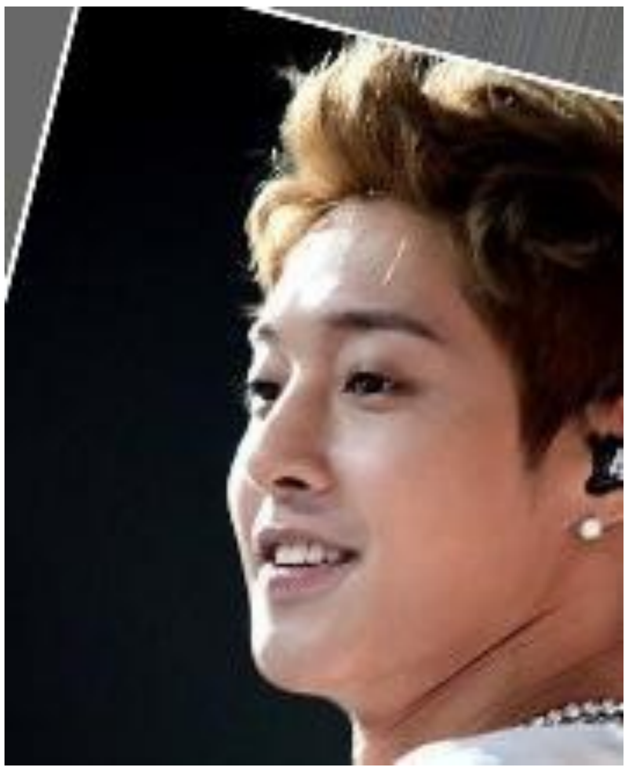
Recall für Feature 'Mouth_Slightly_Open'

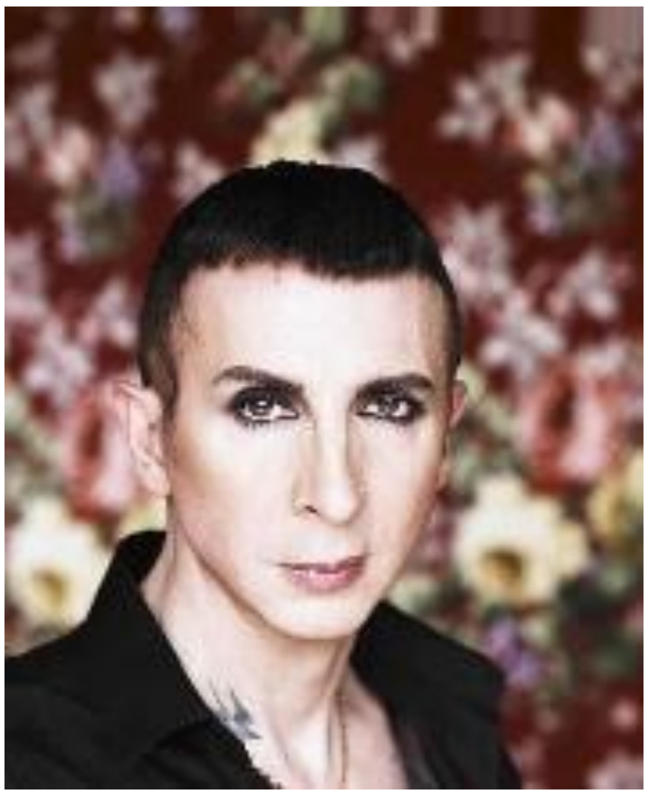
Kat. 0	89.4%
Kat. 1	92.6%

Recall für Feature 'Smiling'

Kat. 0	91.4%
Kat. 1	90.2%

Beispiele von Fehlklassifizierung für Feature „Male“





Beispiele von Fehlklassifizierung für Feature „Smiling“