In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from glob import glob

%matplotlib inline
```

In [3]:
```
# load the data

data = np.load('./dataa/data_10000_norm.npz')
```

In [4]:
```
data.files
```

Out[4]:
```
['arr_0', 'arr_1']
```

In [5]:
```
X = data['arr_0'] # Independent features
y = data['arr_1'] # dependent features
```

In [6]:
```
X.shape, y.shape
```

Out[6]:
```
((5458, 10000), (5458,))
```

Eigen Images

In [7]:
```
X1 = X - X.mean(axis=0)
```

In [8]:
```
from sklearn.decomposition import PCA
```

In [9]:
```
pca = PCA(n_components=None,whiten=True,svd_solver='auto')
```

In [10]:
```
x_pca = pca.fit_transform(X1)
```

In [11]:
```
x_pca.shape
```

Out[11]:
```
(5458, 5458)
```

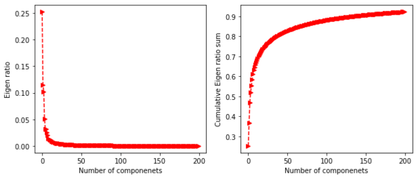
In [12]:
```
eigen_ratio = pca.explained_variance_ratio_
eigen_ratio_cum = np.cumsum(eigen_ratio)
```

In [13]:
```
plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.plot(eigen_ratio[:200],'r>--')
plt.xlabel('Number of componenets')
plt.ylabel('Eigen ratio')

plt.subplot(1,2,2)
plt.plot(eigen_ratio_cum[:200],'r>--')
plt.xlabel('Number of componenets')
plt.ylabel('Cumulative Eigen ratio sum')
```

Out[13]:
```
Text(0, 0.5, 'Cumulative Eigen ratio sum')
```



Conclusion: Using elbow method, consider number of components between 25-30

> since if I consider component between 25-30 the explained variance is around 75% so, inorder to get min 80% variance I am considering 50 components

In [14]:
```
pca_50 = PCA(n_components=50,whiten=True,svd_solver='auto')
x_pca_50 = pca_50.fit_transform(X1)
```

In [15]:
```
# saving pca
import pickle

pickle.dump(pca_50,open('./dataa/pca_50.pickle','wb'))
```

In [16]:
```
# consider 50 components and inverse transform


x_pca_inv = pca_50.inverse_transform(x_pca_50)
```

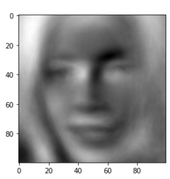
In [17]:
```
x_pca_inv.shape
```

Out[17]:
```
(5458, 10000)
```

In [20]:
```
#checking one image

eigen_img = x_pca_inv[0,:]
eigen_img = eigen_img.reshape((100,100))
plt.imshow(eigen_img,cmap='gray')
```

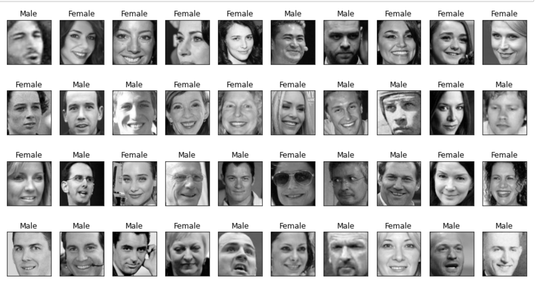
Out[20]:
```
<matplotlib.image.AxesImage at 0x2801aaf6108>
```



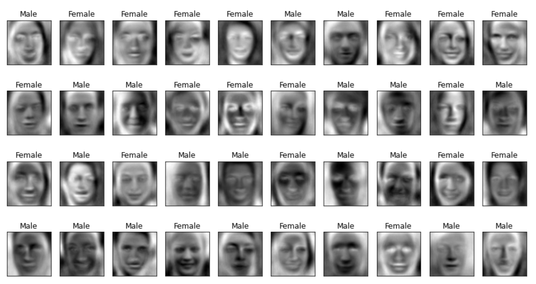
In [22]:
```
def label(y):
    if y==0:
        return 'Male'
    else:
        return 'Female'

np.random.randint(1001)
pics = np.random.randint(0,5458,40)
plt.figure(figsize=(15,8))
for i,pic in enumerate(pics):
    plt.subplot(4,10,i+1)
    img = X[pic:pic+1].reshape(100,100)
    plt.imshow(img,cmap='gray')
    plt.title('{}'.format(label(y[pic])))
    plt.xticks([])
    plt.yticks([])
plt.show()

print("="*20+'Eigen Images'+"="*20)
plt.figure(figsize=(15,8))
for i,pic in enumerate(pics):
    plt.subplot(4,10,i+1)
    img = x_pca_inv[pic:pic+1].reshape(100,100)
    plt.imshow(img,cmap='gray')
    plt.title('{}'.format(label(y[pic])))
    plt.xticks([])
    plt.yticks([])
plt.show()
```




```
====================Eigen Images====================
```



In [23]:
```
# saving

np.savez('./dataa/Data_pca_mean_50.pickle',x_pca_50,y,X.mean())
```

In [ ]: