سوال سوم

1 . هر دو الگوریتم PCA و LDA از این جهت که به دنبال یک سری بردار پایه برای تصویر کردن داده ها هستند به یکدیگر شباهت دارند . اما در الگوریتم LDA هدف این است که با توجه به کلاس داده ها، داده ها طوری تصویر شوند که میانگین آن ها از هم فاصله بگیرند و واریانس کلاس های آن ها نیز همین طور . اما در الگوریتم PCA کلاس داده ها اهمیت ندارند و فقط این مهم است که واریانس داده ها در چه جهتی بیشتر خواهد شد . بنابراین می توان گفت هنگامی که هدف کلاس بندی داده هاست و البته تعداد داده ها از ابعادی که می خواهیم داده ها را تصویر کنیم بیشتر است بهتر است از الگوریتم LDA استفاده کنیم . ولی اگر تعداد ابعادی که می خواهیم بعد داده را به آن کاهش دهیم از تعداد کلاس ها بیشتر بود باید از الگوریتم PCA استفاده کنیم .

2 . با توجه به اینکه در الگوریتم PCA داده ها مستقل از اینکه چه برچسبی دارند بررسی می شوند ممکن است این الگوریتم برای حالت supervised به خوبی عمل نکند و حتی داده هایی که مربوط به کلاس های مختلف هستند را اطراف یکدیگر قرار دهد .

سوال چهارم

ابتدا کتابخانه های مورد نیاز و سپس داده ها را وارد می کنیم .

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

In [2]:
```
from sklearn.decomposition import PCA
from sklearn.decomposition import TruncatedSVD
import sklearn.metrics
```

In [3]:
```
train_set = pd.read_csv('train.csv')
```

در این قسمت هر کدام از تصاویر به شکل یک ماتریس 8*8 در آمده تا در ادامه فرآیند مورد سوال روی آن انجام گیرد .

In [5]:
```
```

In [6]:

Out[6]:
```
array([[[ 52,  42,  34, ..., 127, 127, 123],
        [ 69,  67,  67, ..., 149, 132, 130],
        [104, 103, 105, ..., 156, 160, 142],
        ...,
        [119, 120, 120, ...,  55,  51,  59],
        [111, 108, 114, ...,  97,  93,  93],
        [ 89,  93,  87, ...,  75,  82,  80]],

       [[ 79,  80,  68, ...,  49,  50,  51],
        [ 78,  81,  69, ...,  54,  54,  51],
        [ 76,  82,  70, ...,  50,  49,  53],
        ...,
        [133, 132, 131, ..., 123, 128, 127],
        [ 98,  85,  79, ..., 127, 129, 132],
        [ 74,  79,  72, ..., 127, 127, 131]],

       [[138, 133, 136, ..., 129, 132, 134],
        [145, 138, 141, ..., 128, 130, 131],
        [157, 150, 144, ..., 125, 130, 129],
        ...,
        [ 24,  23,  24, ...,  17,  17,  16],
        [ 23,  22,  22, ...,  17,  19,  20],
        [ 25,  25,  25, ...,  20,  22,  23]],

       ...,

       [[ 58,  62,  64, ...,  45,  46,  76],
        [ 64,  62,  63, ...,  95, 139, 182],
        [108, 116, 130, ..., 210, 229, 230],
        ...,
        [224, 222, 220, ...,  72, 121, 148],
        [213, 217, 223, ...,  45,  44,  54],
        [160, 159, 163, ...,  57,  55,  48]],

       [[175, 173, 175, ..., 155, 152, 154],
        [172, 175, 175, ..., 154, 153, 155],
        [173, 170, 171, ..., 144, 150, 150],
        ...,
        [154, 156, 154, ..., 116, 117, 117],
        [155, 154, 154, ..., 116, 118, 115],
        [122, 132, 142, ..., 117, 114, 115]],

       [[144, 154, 156, ...,  32,  31,  27],
        [160, 162, 164, ...,  62,  55,  49],
        [167, 165, 167, ...,  81,  80,  77],
        ...,
        [ 83,  75,  88, ...,  35,  42,  42],
        [ 71,  81,  47, ...,  39,  37,  50],
        [ 98,  89,  91, ...,  34,  36,  42]]])
```

در این قسمت 32 تا از عکس های داده های آموزشی برای نمونه رسم شده است .

In [7]:
```
plt.figure(figsize=(16, 8))

for i in range(0, 32):
  plt.subplot(4, 8, i + 1)
  plt.imshow(train_faces[i], cmap = 'gray')
  plt.axis('off')
```

در این قسمت برای اینکه داده ها به PCA داده شوند آماده شده است و سپس یک PCA ساخته شده و داده ها به آن فیت شده اند و میانگین و عناصر آن نشان داده شده است .

In [9]:
```
```

In [10]:
```
pca = PCA()
```

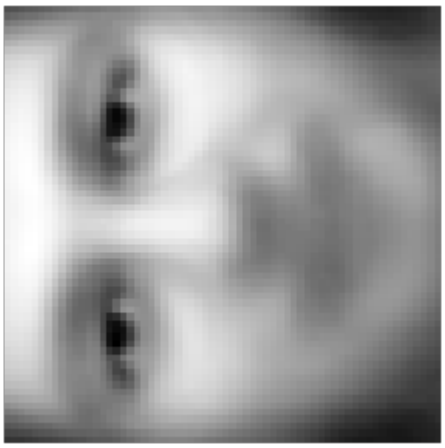
Out[10]:
```
PCA(copy=True, iterated_power='auto', n_components=None, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

میانگین عکس ها به صورت زیر است .

In [11]:
```
```

Out[11]:
```
array([95.52333333, 96.94333333, 98.63      , ..., 71.82666667,
       73.66666667, 74.73      ])
```

In [12]:
```
plt.imshow(pca.mean_.reshape(64, 64), cmap = 'gray')
plt.axis('off')
```



PC ها نیز در زیر نشان داده شده اند .

In [13]:
```
```

Out[13]:
```
array([[ 0.00312273,  0.00316557,  0.00363146, ...,  0.00150902,
        -0.00015355, -0.00147422],
       [ 0.02758496,  0.02871165,  0.03012808, ..., -0.02694858,
        -0.02770146, -0.02844937],
       [ 0.00127559, -0.00170876, -0.00539665, ..., -0.00856413,
        -0.00898205, -0.00759856],
       ...,
       [-0.00052242, -0.00856238, -0.00863223, ...,  0.03759114,
         0.01803062, -0.0788451 ],
       [ 0.00993009, -0.01954237, -0.01092584, ..., -0.01269036,
        -0.02701062,  0.00606819],
       [ 0.03122904,  0.02470276, -0.02552069, ..., -0.00171424,
         0.00375668,  0.00318187]])
```

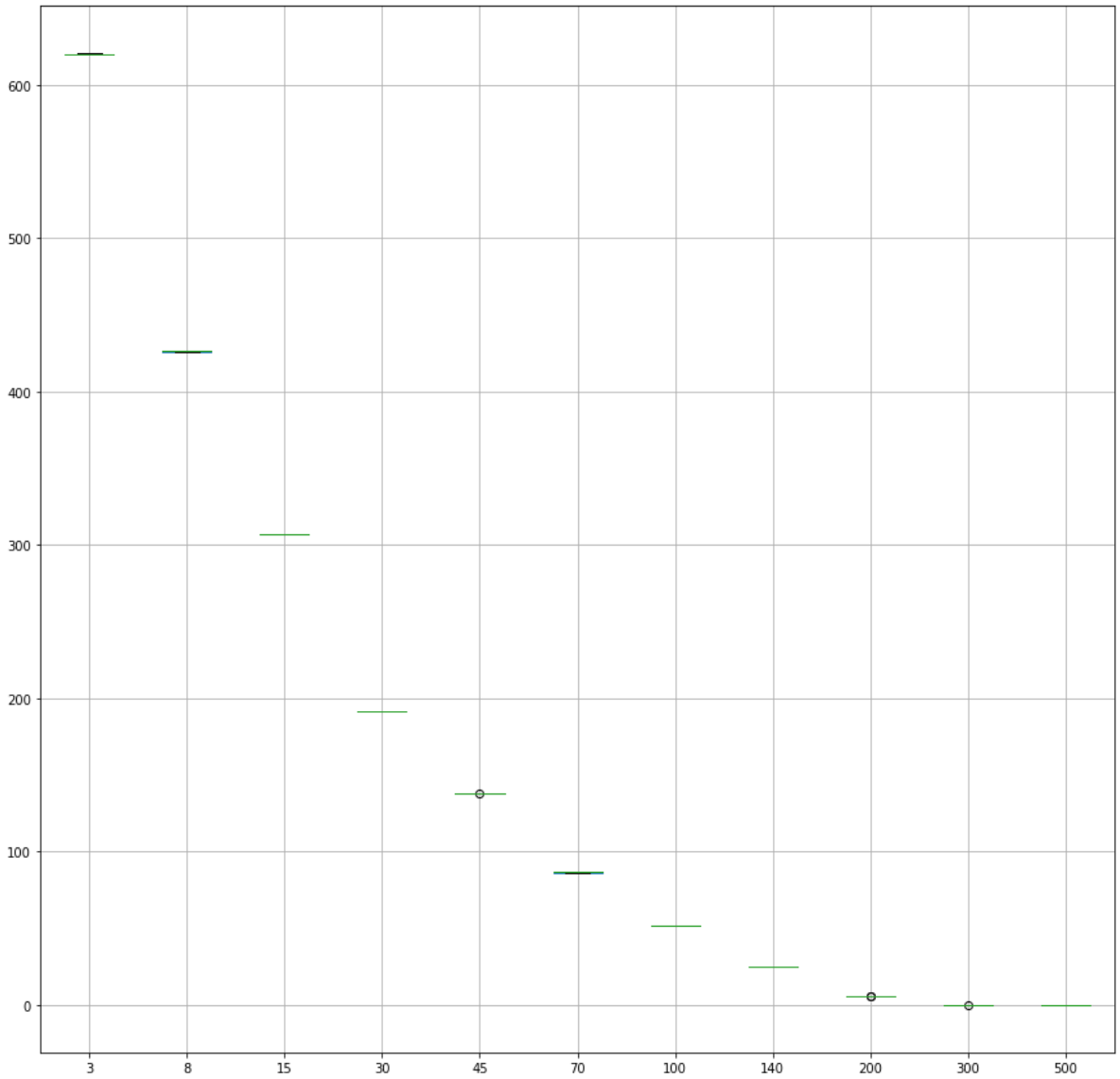
در این قسمت ارزیابی مدل ها که بر اساس تعداد ابعاد می باشد صورت گرفته است . مدل ها TruncatedSVD به جای PCA می باشند که عملکرد آن ها به هم شبیه است .

In [15]:
```
dimension = np.array([3, 8, 15, 30, 45, 70, 100, 140, 200, 300, 500])

mse_train = np.empty((len(dimension), 10))
mse_test = np.empty((len(dimension), 10))
k = 0
for d in dimension:
  train_mse = np.empty((10, 5))
  test_mse = np.empty((10, 5))

  for i in range(0, 10):
    X = np.arange(0, len(x_train))
    kf = KFold(n_splits = 5, shuffle = True)
    j = 0
    for train, test in kf.split(X):

      model = TruncatedSVD(n_components = d)
      model.fit(x_train[train])

      train_transformed = model.transform(x_train[train])
      train_inverse_transformed = model.inverse_transform(train_transformed)
      train_mse[i][j] = sklearn.metrics.mean_squared_error(x_train[train], tra

      test_transformed = model.transform(x_train[test])
      test_inverse_transformed = model.inverse_transform(test_transformed)
      test_mse[i][j] = sklearn.metrics.mean_squared_error(x_train[test], test_

      j = j + 1

  for i in range(0, 10):
    mse_train[k][i] = train_mse[i].mean()
    mse_test[k][i] = test_mse[i].mean()
```

در اینجا boxplot مربوط به خطای بازسازی داده های آموزشی رسم شده است

In [16]:
```
train_validation = pd.DataFrame({'3': mse_train[0],
                                 '8': mse_train[1],
                                 '15': mse_train[2],
                                 '30': mse_train[3],
                                 '45': mse_train[4],
                                 '70': mse_train[5],
                                 '100': mse_train[6],
                                 '140': mse_train[7],
                                 '200': mse_train[8],
                                 '300': mse_train[9],
                                 '500': mse_train[10]})
```

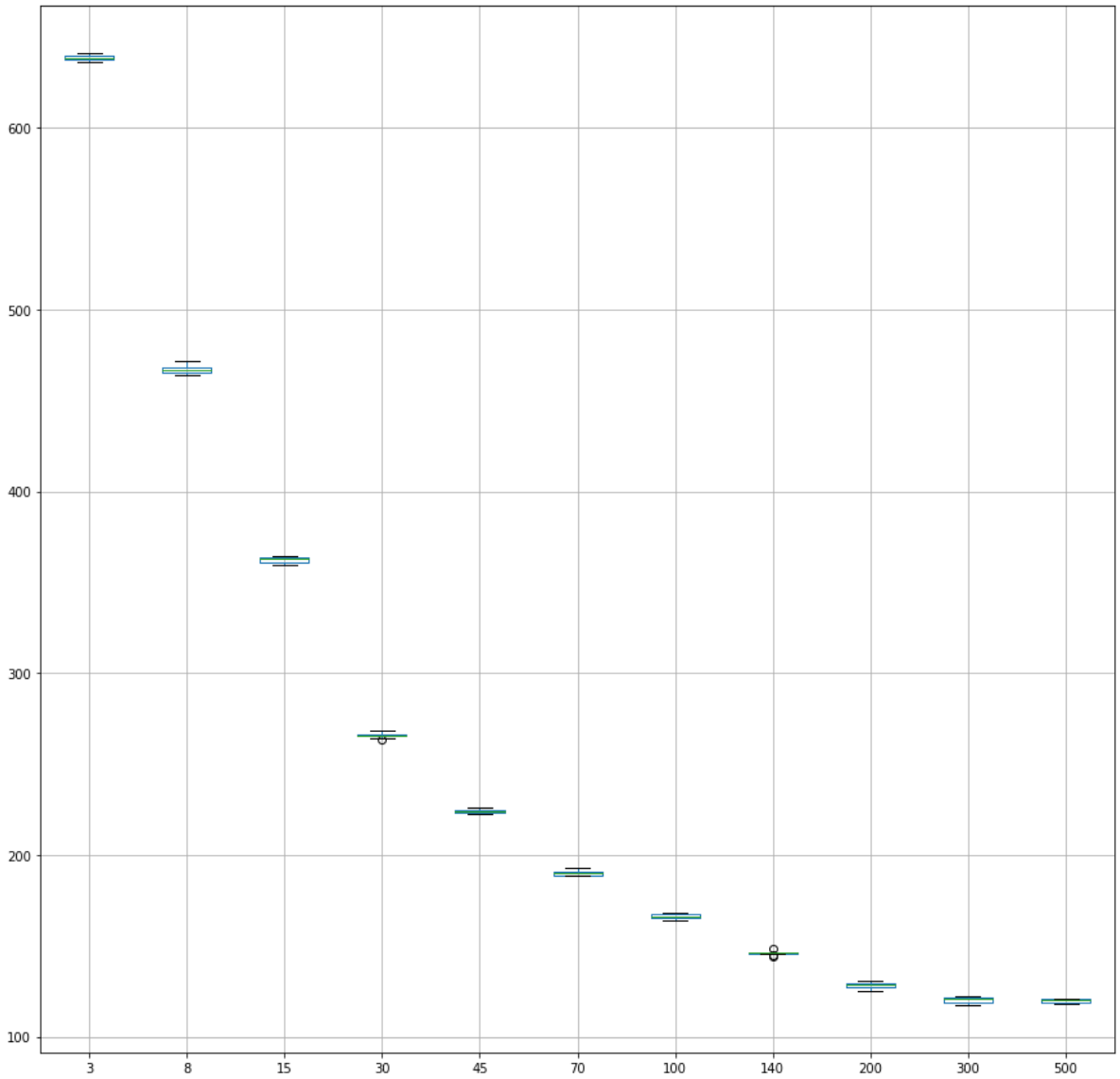
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffc28274190>



در اینجا boxplot مربوط به خطای بازسازی داده های تست رسم شده است .

In [17]:
```
test_validation = pd.DataFrame({'3': mse_test[0],
                                '8': mse_test[1],
                                '15': mse_test[2],
                                '30': mse_test[3],
                                '45': mse_test[4],
                                '70': mse_test[5],
                                '100': mse_test[6],
                                '140': mse_test[7],
                                '200': mse_test[8],
                                '300': mse_test[9],
                                '500': mse_test[10]})
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffc280fe0d0>

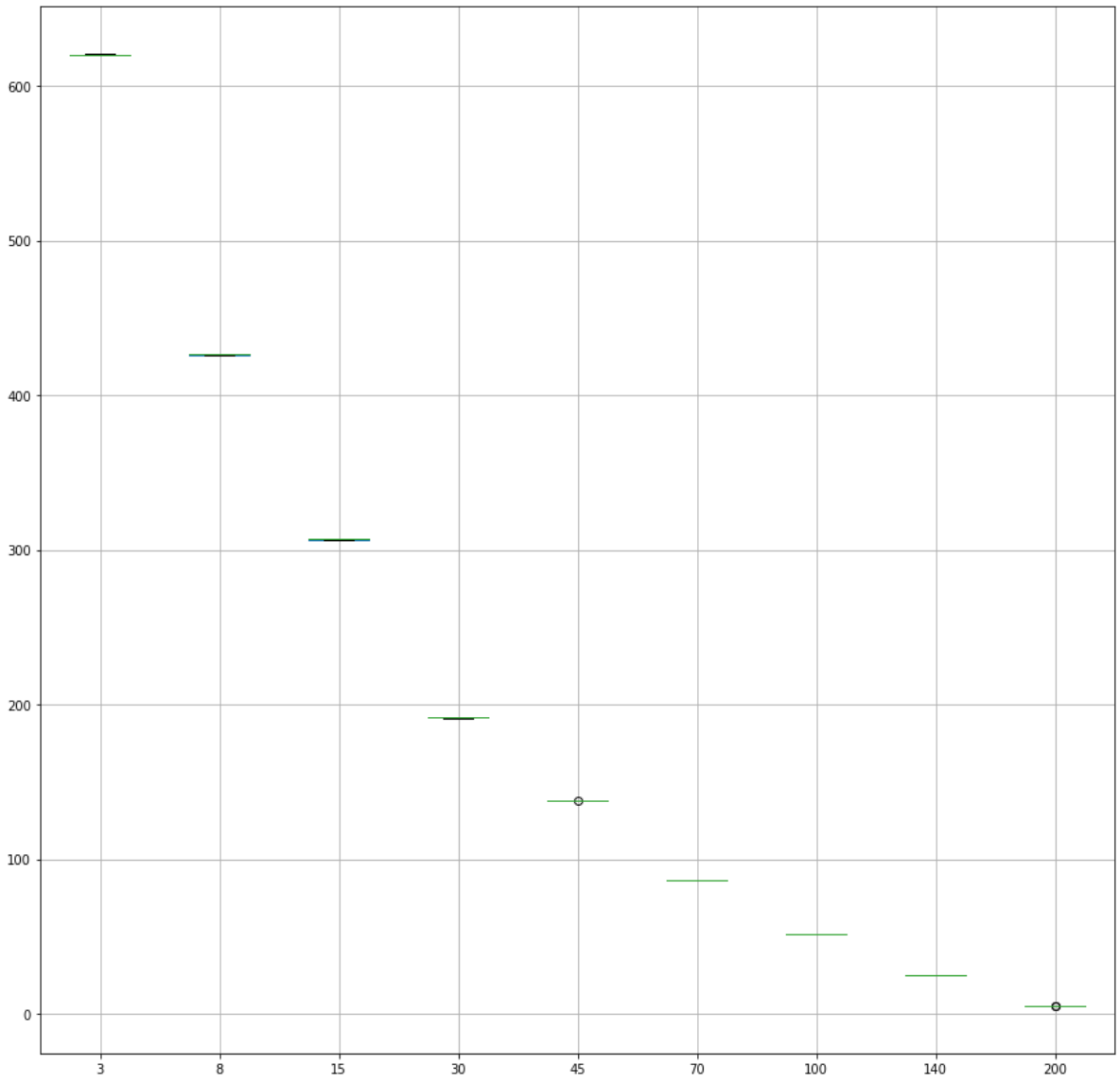


در اینجا نیز PCA بررسی شده است ولی ابعاد آن را نمی توان بیشتر از 240 بالا برد .

In [19]:
```
dimension = np.array([3, 8, 15, 30, 45, 70, 100, 140, 200])

mse_train1 = np.empty((len(dimension), 10))
mse_test1 = np.empty((len(dimension), 10))
k = 0
for d in dimension:
  train_mse1 = np.empty((10, 5))
  test_mse1 = np.empty((10, 5))

  for i in range(0, 10):
    X = np.arange(0, len(x_train))
    kf = KFold(n_splits = 5, shuffle = True)
    j = 0
    for train, test in kf.split(X):

      model = PCA(n_components = d)
      model.fit(x_train[train])

      train_transformed = model.transform(x_train[train])
      train_inverse_transformed = model.inverse_transform(train_transformed)
      train_mse1[i][j] = sklearn.metrics.mean_squared_error(x_train[train], tr

      test_transformed = model.transform(x_train[test])
      test_inverse_transformed = model.inverse_transform(test_transformed)
      test_mse1[i][j] = sklearn.metrics.mean_squared_error(x_train[test], test

      j = j + 1

  for i in range(0, 10):
    mse_train1[k][i] = train_mse1[i].mean()
    mse_test1[k][i] = test_mse1[i].mean()
```

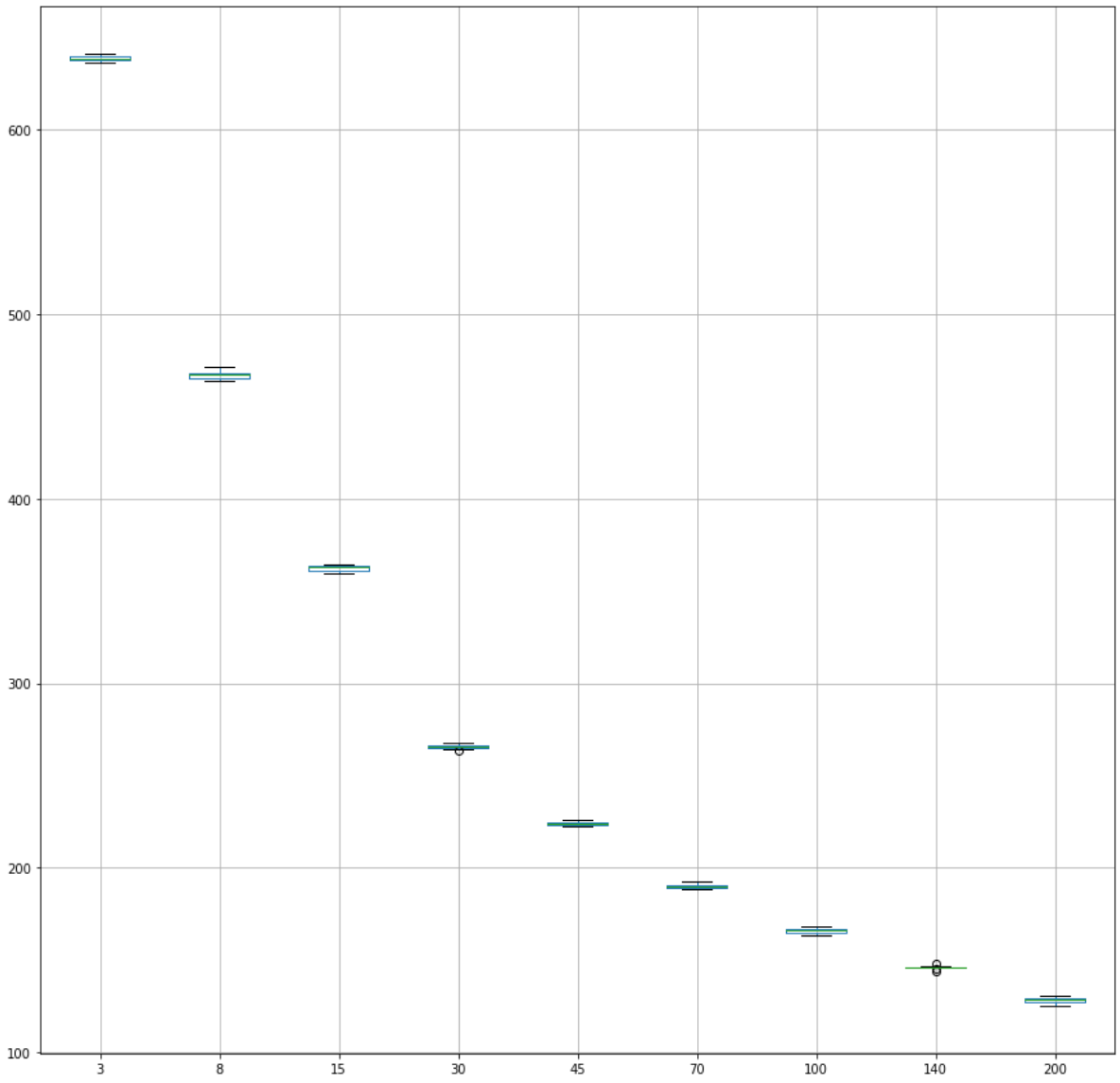
In [20]:
```
train_validation1 = pd.DataFrame({'3': mse_train[0],
                                  '8': mse_train[1],
                                  '15': mse_train[2],
                                  '30': mse_train[3],
                                  '45': mse_train[4],
                                  '70': mse_train[5],
                                  '100': mse_train[6],
                                  '140': mse_train[7],
                                  '200': mse_train[8]})
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffc24da86d0>
```



In [21]:
```
test_validation1 = pd.DataFrame({'3': mse_test[0],
                                 '8': mse_test[1],
                                 '15': mse_test[2],
                                 '30': mse_test[3],
                                 '45': mse_test[4],
                                 '70': mse_test[5],
                                 '100': mse_test[6],
                                 '140': mse_test[7],
                                 '200': mse_test[8]})
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffc24c20c10>
```



با توجه به mse در حالت های 200, 300, 500 می توان گفت که در حالت ابعاد برابر با 200 عملکرد مطلوبی خواهیم داشت .

در ادامه مدل ساخته شده است و سپس روی داده های تست transform شده است و در نهایت نیز بازسازی شده است .

In [22]:
```
final_model = TruncatedSVD(n_components=200)
```

Out[22]:
```
TruncatedSVD(algorithm='randomized', n_components=200, n_iter=5,
             random_state=None, tol=0.0)
```

In [23]:
```
x_test = np.array(test_set)
x_test_transformed = final_model.transform(x_test)
```

خطای mse مدل نیز در زیر محاسبه شده است .

In [24]:

در ادامه شکل های مربوط به تصاویر اصلی و تصاویر بازسازی شده رسم شده است .

تصاویر اصلی :

In [25]:

In [26]:
```
plt.figure(figsize=(20, 20))

for i in range(0, 100):
  plt.subplot(10, 10, i + 1)
  plt.imshow(test_face[i], cmap = 'gray')
```

تصاویر بازسازی شده :

In [27]:
```
```

In [28]:
```
plt.figure(figsize=(20, 20))

for i in range(0, 100):
  plt.subplot(10, 10, i + 1)
  plt.imshow(reconstructed_test_face[i], cmap = 'gray')
```