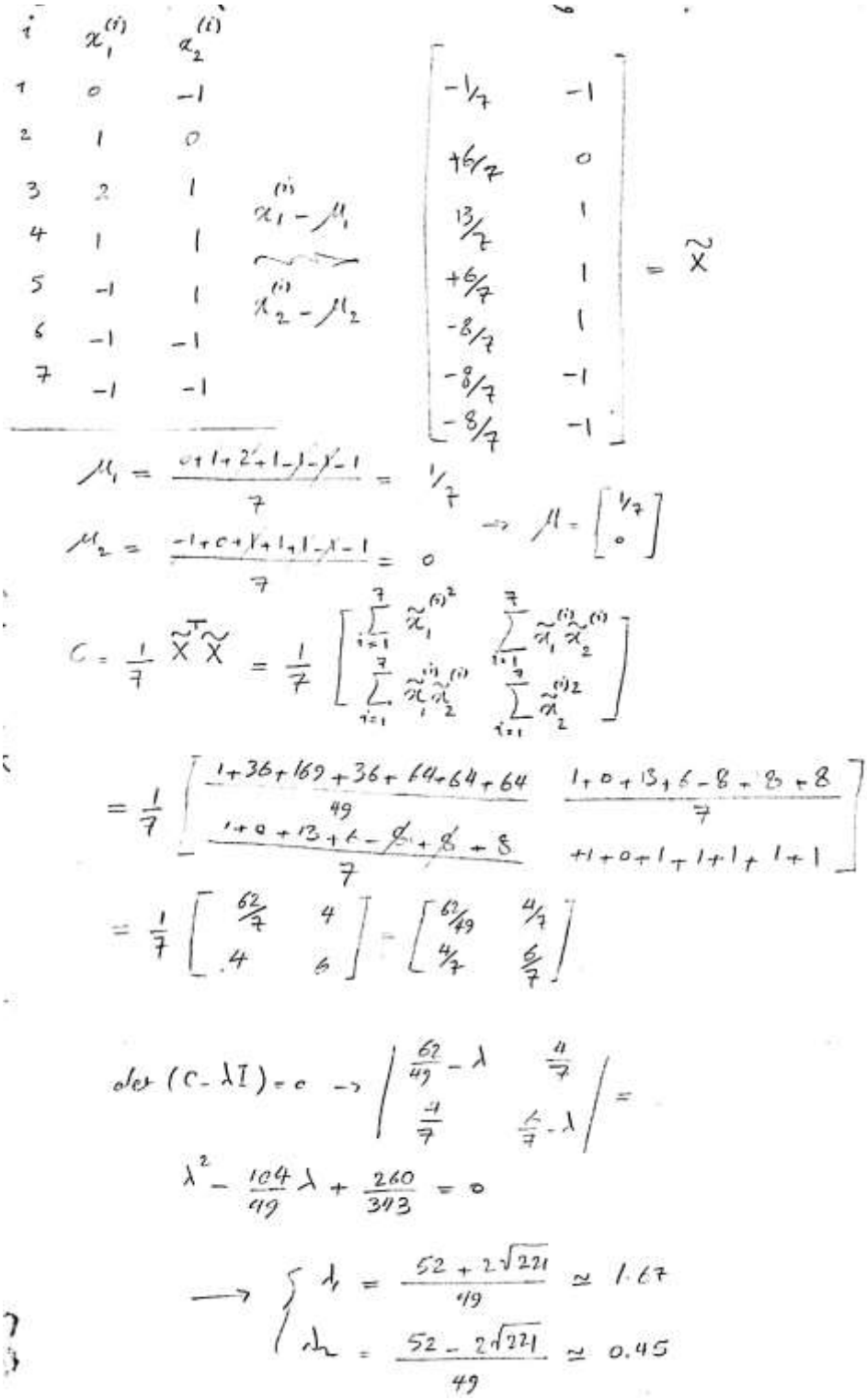
Attached file: hw4.ipynb

Question 1:



i	$x_1^{(i)}$	$x_2^{(i)}$
1	0	-1
2	1	0
3	2	1
4	1	1
5	-1	1
6	-1	-1
7	-1	-1

$x_1^{(i)} - \mu_1$

$x_2^{(i)} - \mu_2$

$$= \tilde{X}$$

$$C\vec{v}_1 = \lambda_1 \vec{v}_1$$

$$\begin{bmatrix} \frac{12}{49} v_1 + \frac{4}{7} v_2 \\ \frac{4}{7} v_1 + \frac{6}{7} v_2 \end{bmatrix} = \begin{bmatrix} 1.67\, v_1 \\ 1.67\, v_2 \end{bmatrix}$$

$$\downarrow$$

$$\simeq \begin{bmatrix} 1.26 v_1 + 0.57 v_2 \\ 0.57 v_1 + 0.86 v_2 \end{bmatrix} \Rightarrow \begin{array}{l} -0.41 v_1 = -0.57 v_2 \\ 0.57 v_1 = +0.81 v_2 \end{array}$$

$$\Rightarrow \quad v_2 = 0.91\, v_1 \quad \Rightarrow \quad \begin{bmatrix} 1.4 \\ 1 \end{bmatrix} = \vec{v}_1 \rightsquigarrow \text{1st PC}$$

$$C\vec{v}_2 = \lambda_2 \vec{v}_2$$

$$\begin{bmatrix} 1.26 v_1 + 0.57 v_2 \\ 0.57 v_1 + 0.86 v_2 \end{bmatrix} = \begin{bmatrix} 0.45 v_1 \\ 0.45 v_2 \end{bmatrix} \Rightarrow \begin{array}{l} 1.81 v_1 = -057 v_2 \\ 0.57 v_1 = -0.41 v_2 \end{array}$$

$$\Rightarrow \quad v_2 = -1.9\, v_1 \quad \Rightarrow \quad \begin{bmatrix} -0.7 \\ 1 \end{bmatrix} = \vec{v} \rightsquigarrow \text{2nd PC}$$

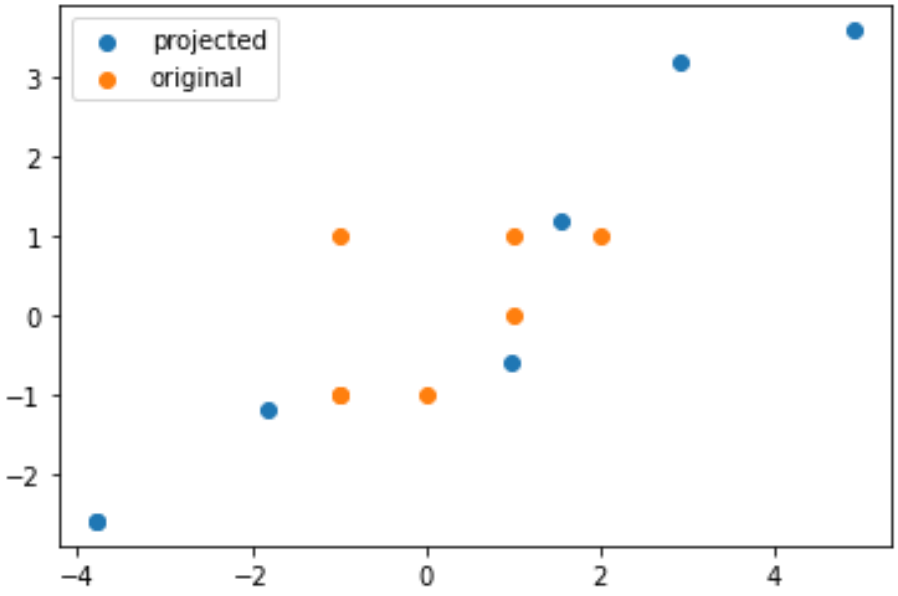
$$A = \vec{v}_1$$

$$X' = \tilde{X} A = \begin{bmatrix} -\frac{1}{7} \times \frac{14}{10} - 1 = -1.2 \\ \frac{6}{7} \times \frac{14}{10} - 0 = 1.2 \\ \frac{13}{7} \times \frac{14}{10} + 1 = 3.6 \\ \frac{6}{7} \times \frac{14}{10} + 1 = 3.2 \\ -\frac{8}{7} \times \frac{14}{10} + 1 = -0.6 \\ -\frac{8}{7} \times \frac{14}{10} - 1 = -2.6 \\ -\frac{8}{7} \times \frac{14}{10} - 1 = -2.6 \end{bmatrix}$$

$$A^T = \vec{v}_1^{\,T} = \begin{bmatrix} 1.4 & 1 \end{bmatrix}$$

$$X_{re} = X' A^T - \begin{bmatrix} \mu_1 & \mu_2 \\ \mu_1 & \mu_2 \\ \mu_1 & \mu_2 \end{bmatrix}_{7\times 2}$$

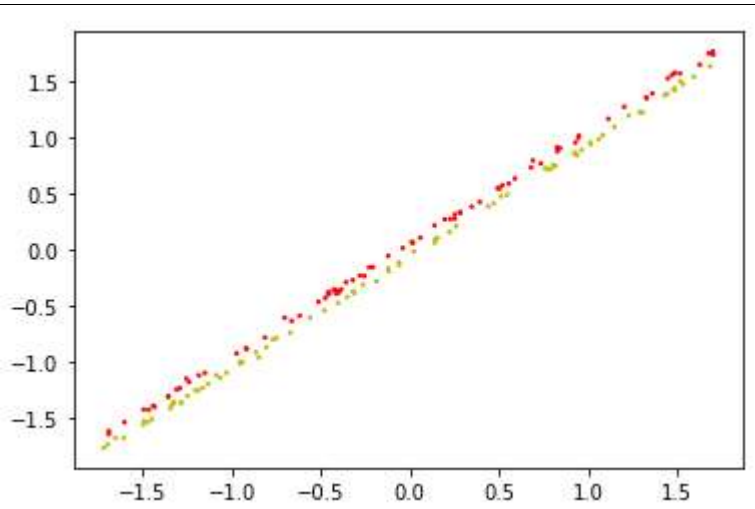
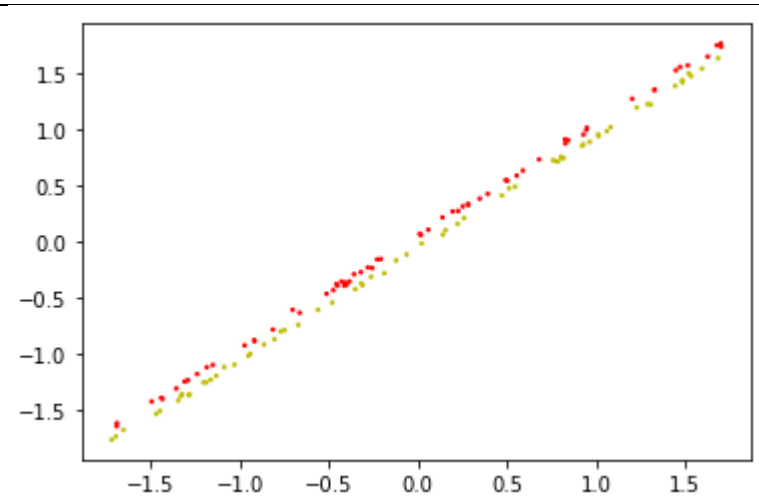
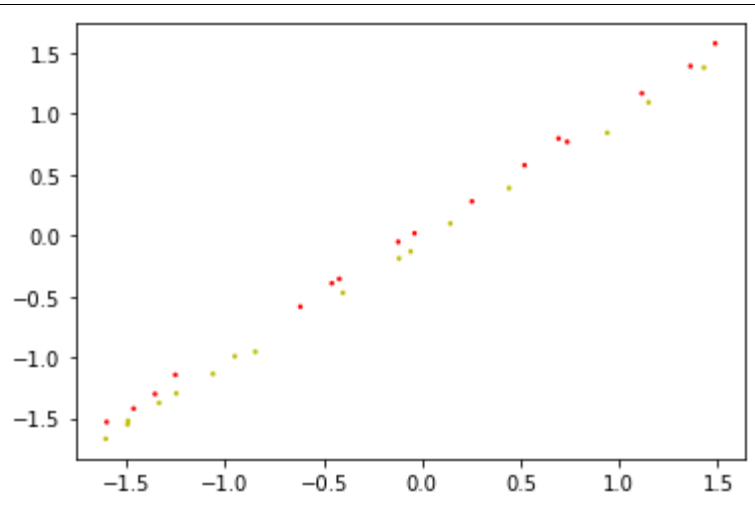
$$= \begin{bmatrix} -1.68-\mu_1 & -1.2-\mu_2 \\ 1.68-\mu_1 & 1.2-\mu_2 \\ 5.04-\mu_1 & 3.6-\mu_2 \\ 3.08-\mu_1 & 3.2-\mu_2 \\ -0.84-\mu_1 & -0.6-\mu_2 \\ -3.64-\mu_1 & -2.6-\mu_2 \\ -3.64-\mu_1 & -2.6-\mu_2 \end{bmatrix} = \begin{bmatrix} -1.82 & -1.2 \\ 1.54 & 1.2 \\ 4.9 & 3.6 \\ 2.93 & 3.2 \\ 0.98 & -0.6 \\ -3.78 & -2.6 \\ -3.78 & -2.6 \end{bmatrix}$$



Question 2:

I read the features and labels of both test and training data sets as numpy data matrices. I changed the labels 0 and 1 to -1 and 1 (just for the convenience of addressing the data) and I call the -1 class the red class and the +1 class the yellow class. Before doing anything I standardize the data with StandardScaler from sklearn. This standardization is done by estimating the mean and variance of the training data as the mean and variance of the unknown distribution. And the same model is used on the test data.

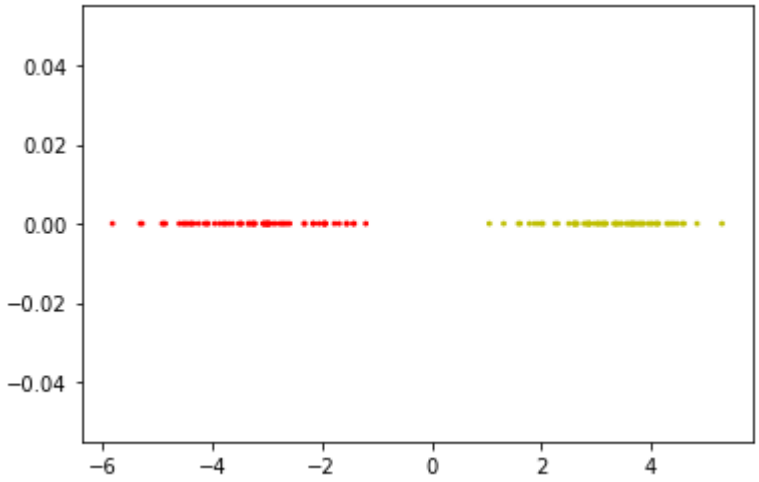
In the first step, display test and training data and both together:

All data	
Train data	
Test data	

Using the LinearDiscriminantAnalysis function from the sklearn library, I fit an LDA model on the training data. The direction vector of LDA is obtained:

```
[ 386.49722, -386.96594]
```

which takes two-dimensional data into one-dimensional space. Training data on this one-dimensional space:



It is obvious that the data can be completely separated by setting a threshold number. I use the sklearn library to determine this threshold value from a perceptron. This perc1 model, after fitting on the training data, becomes equal to the following vector:

[2.49924668 1.]

The number 1 is the bias of the model. Now, by fitting this model vector on the training data, the training accuracy is equal to:

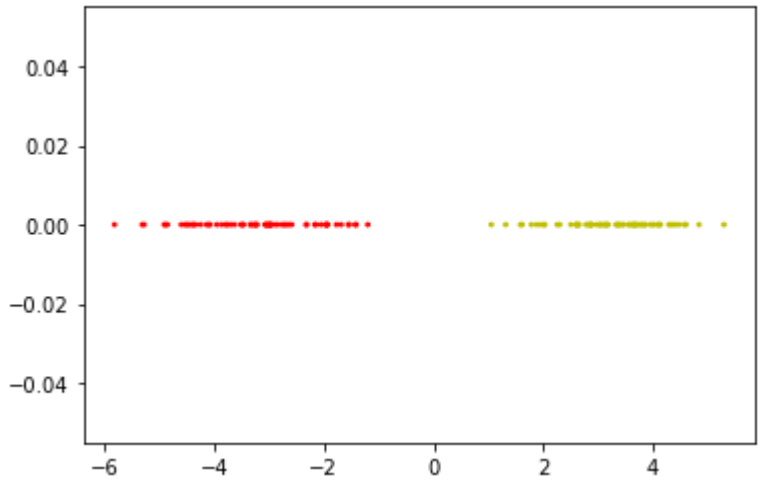
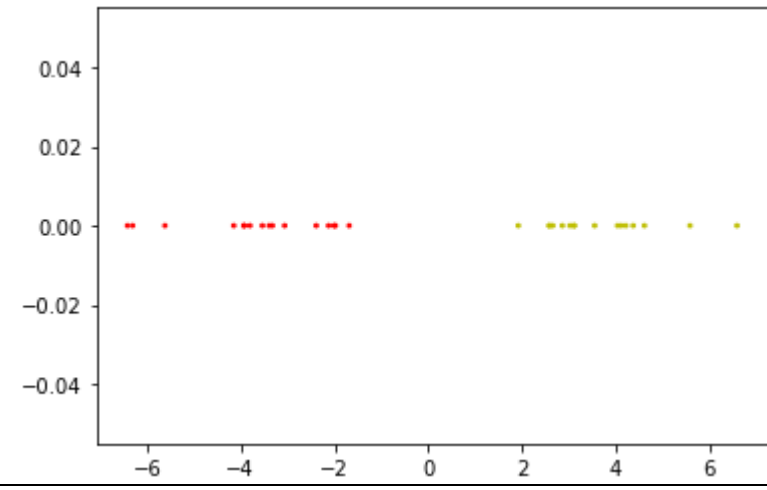
	precision	recall	f1-score	support
-1	1.00	1.00	1.00	64
1	1.00	1.00	1.00	64
accuracy			1.00	128
macro avg	1.00	1.00	1.00	128
weighted avg	1.00	1.00	1.00	128

And after transforming the test data with the LDA alignment vector and classifying the results with perc1 on the test data, I have:

	precision	recall	f1-score	support
-1	1.00	1.00	1.00	16
1	1.00	1.00	1.00	16
accuracy			1.00	32
macro avg	1.00	1.00	1.00	32
weighted avg	1.00	1.00	1.00	32

The accuracy of the two-class classification on both test and training data is 100% and the model error on this data is zero.

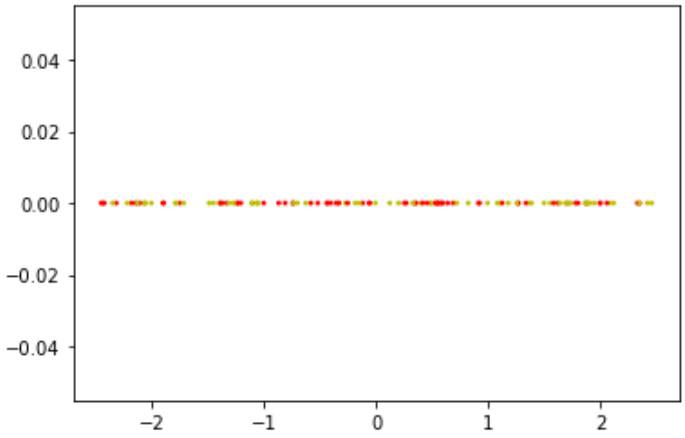
If I show the 1D classified data on the figure, I have no misclassified data (black color). The two classes are far apart.

Train data	
Test data	

Using the PCA model from the sklearn library, I transfer the training data to a one-dimensional space with the PCA criterion. The covariance matrix of this transformation:

```
[[1.0078739  1.0062541],
 [1.0062541  1.0078735]]
```

These obtained data are on the axis based on class color:



This transformation does not separate the class data, but rather merges them together. Again, I model the training data with a perceptron model. obtained perc2 model:

```
[-0.5916291          0.]
```

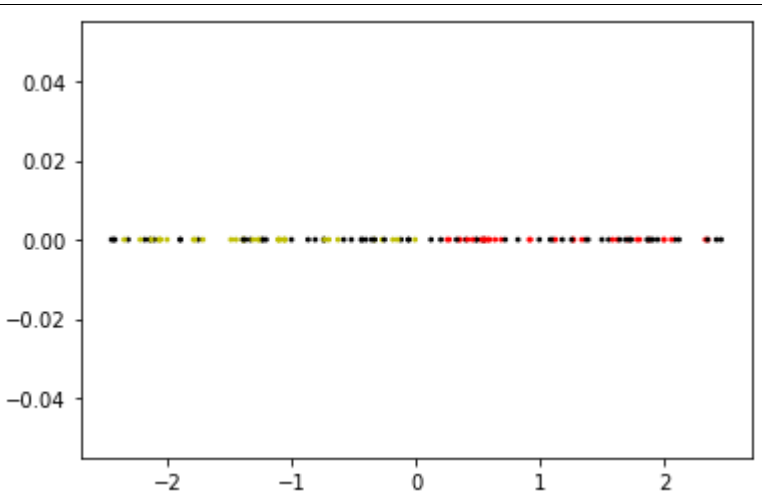
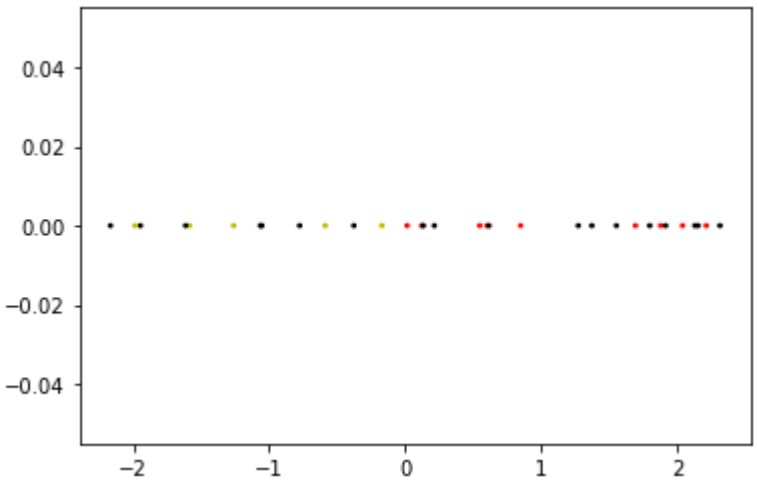
I classify the one-dimensional training data with the PCA component with this model. The following results are obtained.

	precision	recall	f1-score	support
-1	0.50	0.50	0.50	64
1	0.50	0.50	0.50	64
accuracy			0.50	128
macro avg	0.50	0.50	0.50	128
weighted avg	0.50	0.50	0.50	128

The accuracy of the model on the training set is low and only half of the data are classified in the correct class. Also, if I apply the PCA criterion to the test data and then classify with the same model:

	precision	recall	f1-score	support
-1	0.45	0.56	0.50	16
1	0.42	0.31	0.36	16
accuracy			0.44	32
macro avg	0.43	0.44	0.43	32
weighted avg	0.43	0.44	0.43	32

The accuracy of the model is lower than the 50%. Also on the figure:

Train data	
Test data	

If we look again at the first representation of the data in two dimensions, on the same plot a better separation between the data could be imagined than on the data transferred with PCA. This model lacks efficiency.

PCA transformation for data transfer and dimensionality reduction proceeds with the aim of maximizing the variance of the remaining data and tries to map the data in the direction where the maximum variance exists and on the other hand to remove the dimensions that have an almost uniform and flat graph. . PCA is completely unconcerned with data labels (unsupervised method) while LDA measures specifically on labeled data with the aim of sharpening the distinction between two clusters in the lower dimension of the transition. Therefore, it can be expected that if the goal is to categorize the data, the LDA criterion will have a more favorable result. This result was well defined on the models that were made above. If we return to the first display of data (two-dimensional), these two classes are distinguishable on the same two dimensions, but the distance between the two classes is small from the hypothetical dividing line. What LDA did was intensifying this distance, so that on the axis of numbers that LDA made, the distance between the last data of two classes is about 2, while it also increased the

concentration of the data of each class, which was almost not seen on the two-dimensional display. . The LDA criterion reduces or increases the variance of the data selectively according to their class label, while PCA tries to preserve the total variance of the data, which results in a complete integration of the categories on these closely related (but separable) data. because the highest variance of the primary data was not in the direction that distinguished the two classes (hypothetical line between the two classes), but in the same direction, almost perpendicular to it. That is, the variance of the data was preserved in the wrong direction for the purpose of separation because PCA selects the vector of the first component while what is needed in this classification is to preserve the variance in the direction of the vector of the second component and that is why in PCA the classes became inseparable.

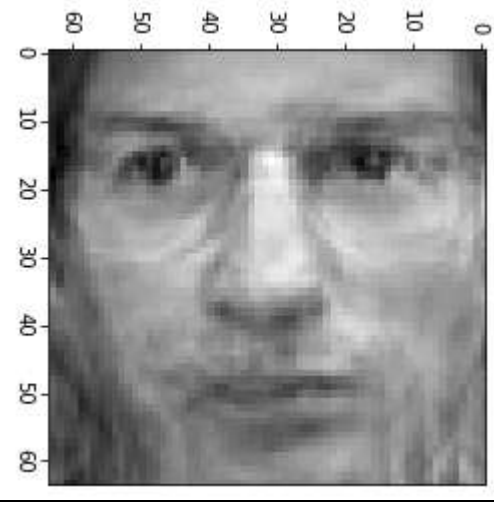
Question 3:

I read the file with scipy. In order to work with the data more easily, I saved the data part that I need, i.e. faces, which had a total of 400 photos and each photo had 64 x 64 components, as a numpy data matrix of 400 rows and 64 x 64 columns. These photos are 10 to 10 photos of 40 people. I also created a vector of 400 for the labels, which, of course, was not used because classification and face recognition was not required! I split these 400 rows without shuffling with train_test_split from sklearn into two groups of 280 and 120 as training and test respectively.

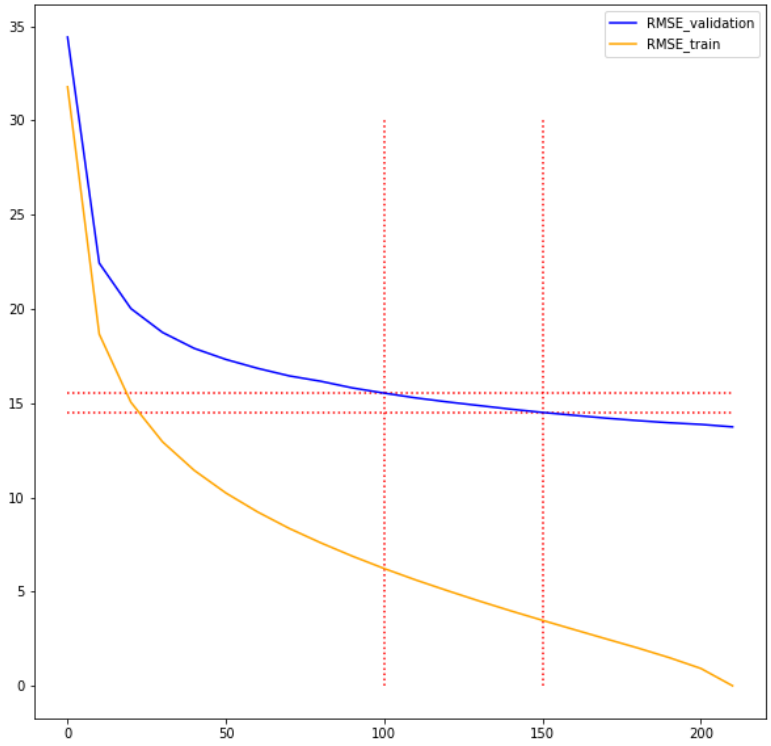
With a PCA model from sklearn 5, I take this data from the 64x64 space to 45 dimensions. The model is only trained with the first 280 data. Then I take the training and test data with the obtained components to the 45-dimensional space and then reconstruct it again to the same original space. The RMSE value obtained between the initial data of each set with the reconstructed space is equal to:

```
RMSE on train data = 11.525403972573551
RMSE on test data = 18.023206734576966
```

Below we have an arbitrary image of the test set (image like test row 1) in two basic and reconstructed states with 45 main components.

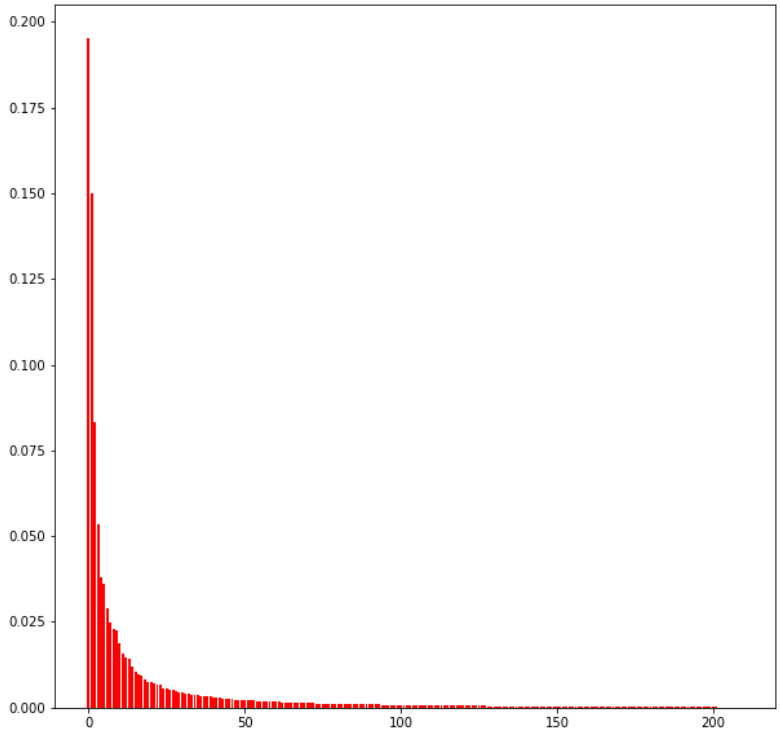
original	projected
	

To choose the optimal dimension, I divide the training set into 75 and 25. And I keep the last part for validation. For the secondary space of the entire test set of 210, I can choose a large set from 1 to 210 as the dimension number (if there were more data, I could take up to 64x64 as the secondary dimension, but here the number limit for I have a data matrix) starting from zero and going forward from 10 to 10 each time I make a PCA transformation on the test set. For each of these transformations made on the sets of 210 training data and 70 validation data, I calculate the transformation value and then reconstruct the initial theorem with the same vectors. I calculate the RMSE. The graph of these obtained numbers:

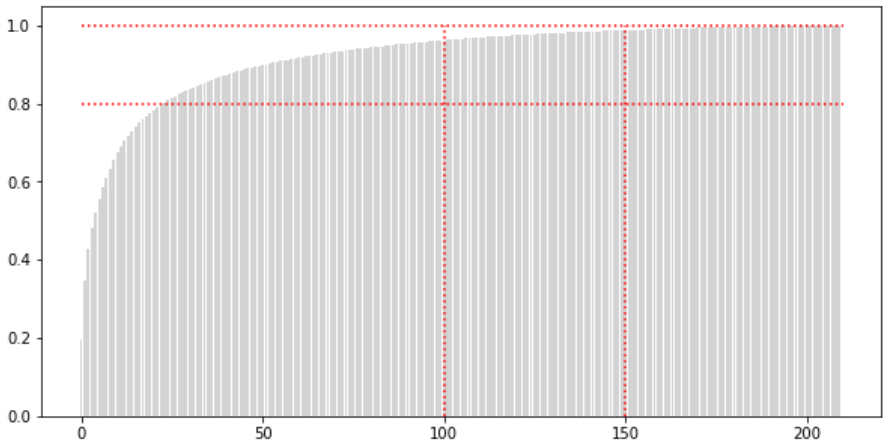


If I pay attention to the part of the diagram with red dashed lines, firstly, from 100 onwards, the slope of the validation blue line becomes gentler and the improvement is more on the test data. So the next number can be chosen after 100. On the other hand, after 150, the slope becomes very gentle. And the remaining 50 dimensions only affect the RMSE by about 1 unit. This amount can be ignored. So I choose my number somewhere between 100 and 150.

If we look carefully at the bar graph of the ratio of distributed variances for each component vector from 1 to 210, the reduction of the ratio from 100 onwards is hardly visible.



For a better view of the graph, I draw the same ratios cumulatively. From vector 150 onwards, the effect is very small. And the last vectors that have a noticeable effect are 100 to 150. On the same interval that I was considering, I choose the middle point as the secondary dimension. Of course, 100 or 150 is also suitable for selection, because in terms of accuracy, we still do not have much difference on this interval for this amount of data. I choose 125.



I expect the result of the final PCA transformation that I make with dimension 125 on the test data to have a value of about 15 from the first graph like the validation data. I make the last model with 125 main components. Results :

```
RMSE on train data = 4.775306976555008
RMSE on validation data = 14.968495895510301
RMSE on test data = 16.10800194377165
```

The reconstruction accuracy on the test is also close to the expected value. The error between the initial training data and the reconstructed data on the training data is very small.