Bilkent University

CS 464
Homework 2 Report

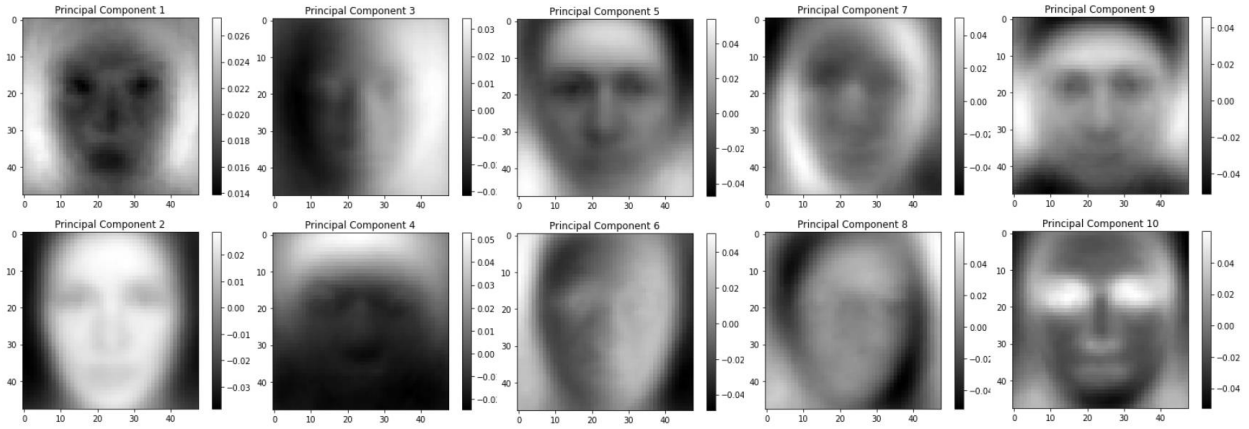
Can Kırımca
ID: 21802271

Question 1.1

For this question, PCA is implemented as three functions in Python. The first function calculates the principal components, the second function applies them to the images and the third one reconstructs the image from the PCA applied version. The proportion of variance explained by each principal component is as follows:

```
PVE for component 1 : 0.2889360592982296
PVE for component 2 : 0.1124551750257083
PVE for component 3 : 0.09959533869406235
PVE for component 4 : 0.0622191000740302
PVE for component 5 : 0.03281326851180241
PVE for component 6 : 0.0291717621381416 37
PVE for component 7 : 0.02136908301160922
PVE for component 8 : 0.0209263097049335 36
PVE for component 9 : 0.01878175059860083
PVE for component 10 : 0.0143692850052702 62
```

After converting the first 10 principal components to 48x48 matrices, the following patterns are obtained:

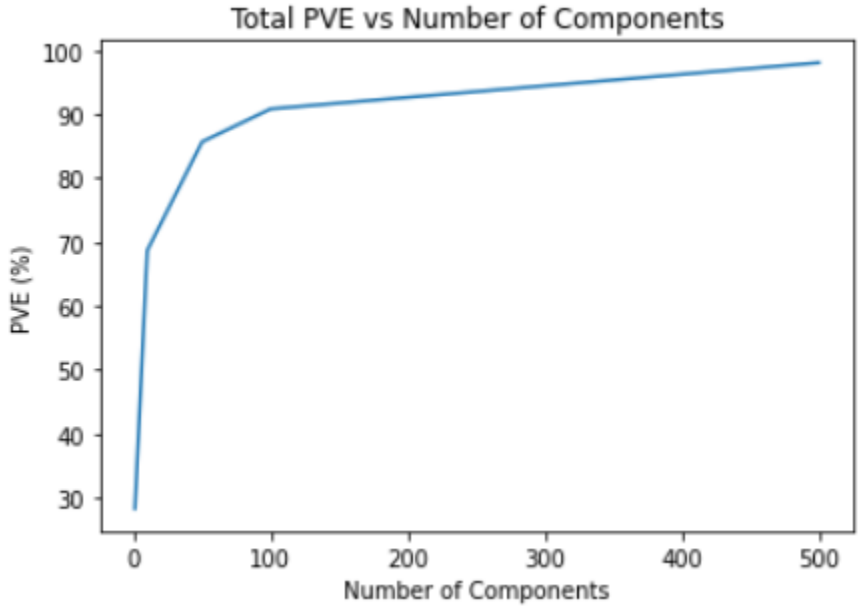


Question 1.2

The total PVE explained by the first k components (k ∈ {1, 10, 50, 100, 500}) is calculated and listed below:

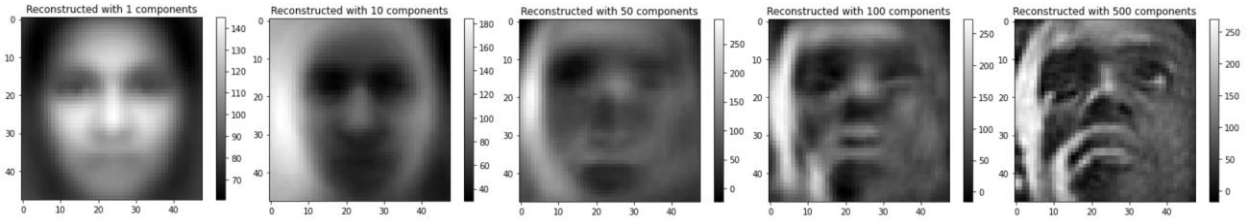
```
Total PVE for the first 1 components: 28.334474895370448 %
Total PVE for the first 10 components: 68.70788394291544 %
Total PVE for the first 50 components: 85.6932188480817 %
Total PVE for the first 100 components: 90.84447232542018 %
Total PVE for the first 500 components: 98.06486239270194 %
```

The following plot is obtained from the previously calculated PVE values:



Question 1.3

After applying PCA, we should multiply the PCA applied matrix with the transpose of the matrix of principal components. This way, we can reconstruct the image. Using more principal components increases the reconstruction quality since more components explain more variance of the dataset, which can be verified from the previous plot. The reconstructed version of the first image is following:



The results verify that the number of principal components is related to the reconstruction quality. Since one principal component cannot explain much variance, the reconstructed version does not look like the

original image. As the number of components increases, the reconstructed image starts to look more similar to the original one.

Question 2.1

In the homework PDF, we are given the following error function:

$$J = (y - X\beta)^T (y - X\beta)$$

In this equation, y is a vector consisting of the ground truth values. X is a matrix consisting of the feature values that will be multiplied with the weights. β is another vector that consists of the values of weights.

To minimize the error, we need to take the derivative with respect to β and set it equal to zero.

$$\frac{dJ}{d\beta} = 2(y - X\beta)^T \frac{d(y - X\beta)}{d\beta} = 0$$

$$= 2(y - X\beta)^T (-X) = 0$$

Which can be simplified to:

$$(y - X\beta)^T X = 0$$

We take the transpose of both sides and get:

$$X^T(y - X\beta) = X^T y - X^T X\beta = 0$$

$$X^T y = X^T X\beta$$

Solving for β, we get:

$$\beta = (X^T X)^{-1} X^T y$$

Question 2.2

$$rank(X^T X) = 14$$

Which is equal to the number of columns. This means X is a full rank matrix and is invertible.

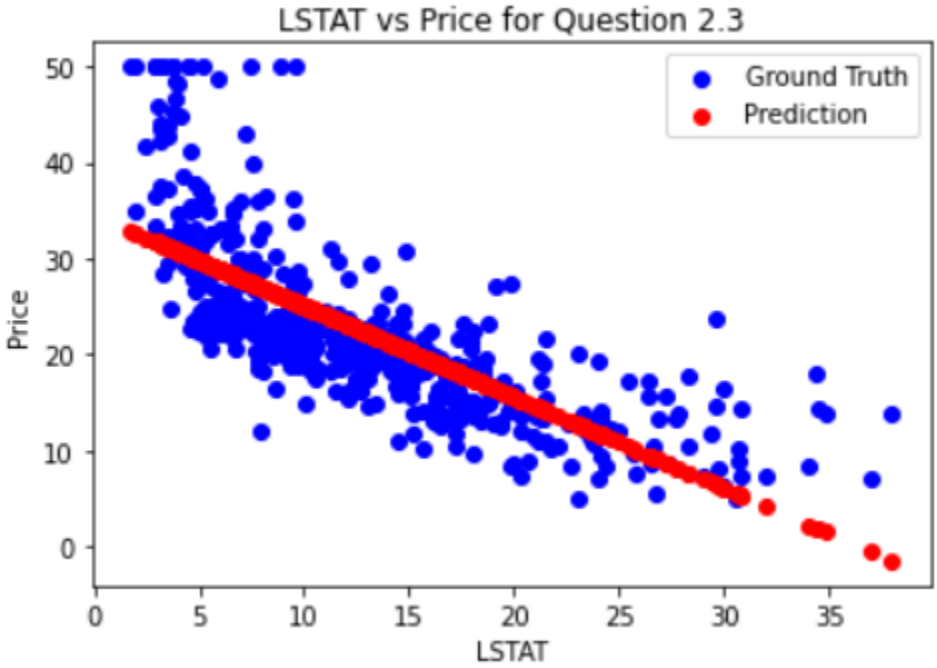
Question 2.3

The coefficients of the model is calculated as below:

```
Coefficients: [[34.55384088]
 [-0.95004935]]
```

The first coefficient, w_0, is multiplied by one and added to the calculation, so it is the constant term of the regression. The second coefficient, w_1, is multiplied with LSTAT feature. Since it is negative, it can be said that LSTAT feature is a negative effect on house prices.

The following plot demonstrates the predictions and ground truth values:



The predictions form a linear pattern, which makes sense for linear regression. It can also be observed that this line has an intercept equal to the first coefficient. The MSE is:

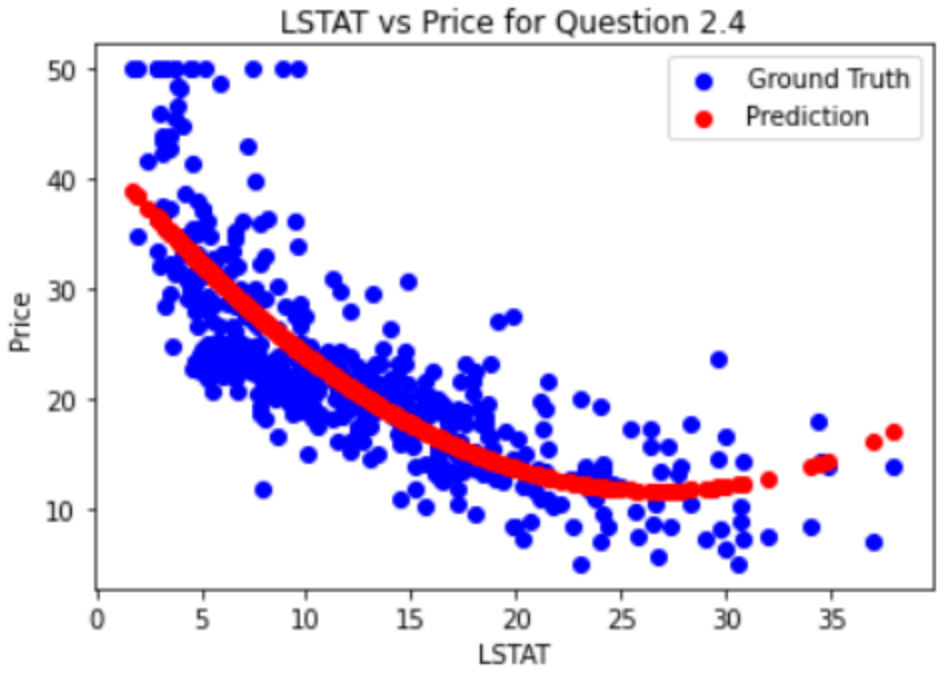
```
MSE: [38.48296723]
```

Question 2.4

When the square of the LSTAT feature is added to the model, the coefficients become:

```
Coefficients: [[42.86200733]
 [-2.3328211 ]
 [ 0.04354689]]
```

Adding the third coefficient increased the constant term and increased the negative effect of the LSTAT feature. The following plot demonstrates the predictions and ground truth values:



MSE becomes:

```
MSE: [30.33052008]
```

After adding the third parameter and applying polynomial regression, MSE decreases. This indicates that the model makes better predictions with polynomial regression.

Question 3.1

In this question, full-batch gradient ascent algorithm is implemented in Python. The algorithm is tested with different learning rates, and 0.01 was chosen since it produced the result with the highest accuracy. The performance metrics of the full batch logistic regression is calculated and the output is given below. The metrics are calculated for both label classes (0 and 1), and their average is also calculated.

```
Accuracy: 0.7039106145251397
True Positives: 50
True Negatives: 76
False Positives: 34
False Negatives: 19

Metrics For Class 0:
        Precision: 0.8
        Recall: 0.6909090909090909
        Negative Predictive Value: 0.5952380952380952
        False Positive Rate: 0.2753623188405797
        False Discovery Rate: 0.2
        F1 Score: 0.7414634146341463
        F2 Score: 0.7102803738317757

Metrics For Class 1:
        Precision: 0.5952380952380952
        Recall: 0.7246376811594203
        Negative Predictive Value: 0.8
        False Positive Rate: 0.3090909090909091
        False Discovery Rate: 0.40476190476190477
        F1 Score: 0.6535947712418301
        F2 Score: 0.6944444444444444
```

Question 3.2

IMPORTANT NOTE: In this part, the instruction was to use 1000 iterations to train the model with mini-batch and stochastic gradient ascent. I implemented the algorithm in a way that it sees only one batch in a single iteration. The weights are updated for each mini-batch (for mini-batch gradient ascent) or for each sample (for stochastic gradient ascent). Therefore, for the mini-batch gradient ascent, the algorithm goes through the entire dataset 1000/8 = 125 times, which corresponds to 125 epochs. Similarly, the stochastic gradient ascent algorithm uses 1000 samples (one sample per iteration), which is between 1 and 2 epochs (since the dataset contains 712 samples).

First, mini-batch gradient ascent algorithm is used to train the model. The performance metrics are calculated for both label classes (0 and 1), and their micro and macro averages are calculated. The output for the mini-batch gradient ascent is as follows:

```
Accuracy: 0.7039106145251397
True Positives: 32
True Negatives: 94
False Positives: 16
False Negatives: 37
Micro Averages:
        Precision: 0.7039106145251397
        Recall: 0.7039106145251397
        Negative Predictive Value: 0.7039106145251397
        False Positive Rate: 0.29608938547486036
        False Discovery Rate: 0.29608938547486036
        F1 Score: 0.6825911097912277
        F2 Score: 0.6727468112349069
Macro Averages:
        Precision: 0.6921119592875318
        Recall: 0.6591567852437418
        Negative Predictive Value: 0.6921119592875318
        False Positive Rate: 0.34084321475625823
        False Discovery Rate: 0.3078880407124682
        F1 Score: 0.6635457672802071
        F2 Score: 0.6584722492486649
```

The output for the stochastic gradient ascent is given below. Since the algorithm considers only 1000 samples, it makes worse predictions compared to the other algorithms. Depending on the random samples that is used to train the model, it makes only negative predictions (0) in some cases. Therefore, if both true positives and false positives are equal to zero, precision and false discovery rate can be 0/0, which generates errors. Therefore, I added a control mechanism to calculate_performance function, which sets 0/0 = 0 in order to avoid errors during the execution. The following output is taken from an execution where the predictions of the model contained some positive labels:

```
Accuracy: 0.6201117318435754
True Positives: 2
True Negatives: 109
False Positives: 1
False Negatives: 67
Micro Averages:
        Precision: 0.6201117318435754
        Recall: 0.6201117318435754
        Negative Predictive Value: 0.6201117318435754
        False Positive Rate: 0.37988826815642457
        False Discovery Rate: 0.37988826815642457
        F1 Score: 0.5490414418228264
        F2 Score: 0.5134885923585465
Macro Averages:
        Precision: 0.6429924242424242
        Recall: 0.5099472990777338
        Negative Predictive Value: 0.6429924242424242
        False Positive Rate: 0.4900527009222661
        False Discovery Rate: 0.35700757575757575
        F1 Score: 0.4088966588966588
        F2 Score: 0.4602912768235349
```

Question 3.3

- NPV and FPR is more informative than precision and recall where correctly predicting the negative samples is the main objective.
- FDR gives the ratio of wrong predictions among all positive predictions. When making positive predictions have major consequences, this ratio becomes more informative about the model's success compared to precision and recall.
- F1 and F2 metrics give various weights to precision and recall. When one of these is more important, F1 and F2 metrics can be sufficiently informative by giving more weight to the more important one. Since these scores consider both false positives and false negatives, they become more informative when we have a imbalanced distribution of classes