CS464 Introduction to Machine Learning

Homework 2 Report

İrem Ecem Yelkanat
21702624
Section 2

1 PCA & Digits

To apply PCA and obtain principal components, I first mean centered the data and compute covariance matrix of mean centered data. Then, I compute eigenvalues and eigenvectors of covariance matrix. I sort principal components with respect to their eigenvalues in order to get first k principal components easily. I compute and store proportion of explained variance of each principal component. Before implementing parts of Question 1, I had the values of eigenvalues, eigenvectors, and PVE's in sorted manner.

Question 1.1 Apply PCA to obtain the first 10 principal components. Report the proportion of variance explained (PVE) for each of the principal components. Reshape each of the principal component to a 28x28 matrix and show them. Discuss your results.

Proportion of variance explained for the first 10 principal components are:

- 0.10047663329207118
- 0.07544486615471581
- 0.06140516190545804
- 0.05425807392725399
- 0.05031248900777848
- 0.042463633647547204
- 0.033114037683358695
- 0.029502883483561806
- 0.027298577357821947
- 0.022780414032228625

First principal component can explain 10% of variance in the data. Summation of the PVEs of the first 10 principal components is 0.49705677049179575. It can be said that first 10 principal components are accounted for approximately 50% of variance in the dataset. Considering there are total of 784 principal components, first ten principal components have significant effect regarding explaining half of the variance of the data. Remaining 774 principal components have approximately %50 of proportion of explained variance in total.

First 10 principal components are (in grayscale):

Figure 1: First 10 Principal Components in Grayscale

First 10 principal components are (with color):

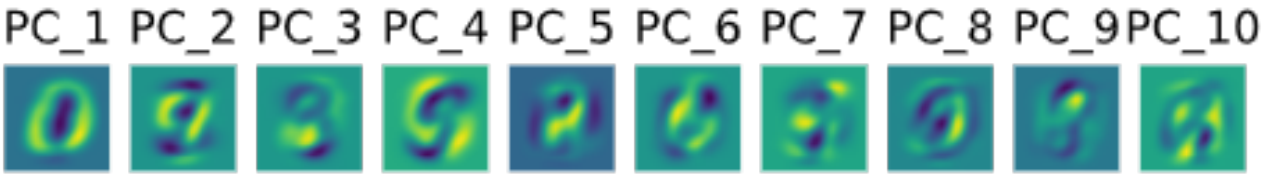

Figure 2: First 10 Principal Components with Color

These principal components show key characteristics of digits. First 6 principal components can be recognized as digits, for example first one is similar to 0, third one is similar to 3 etc. However, the other principal components are hard to recognize to identify a digit.

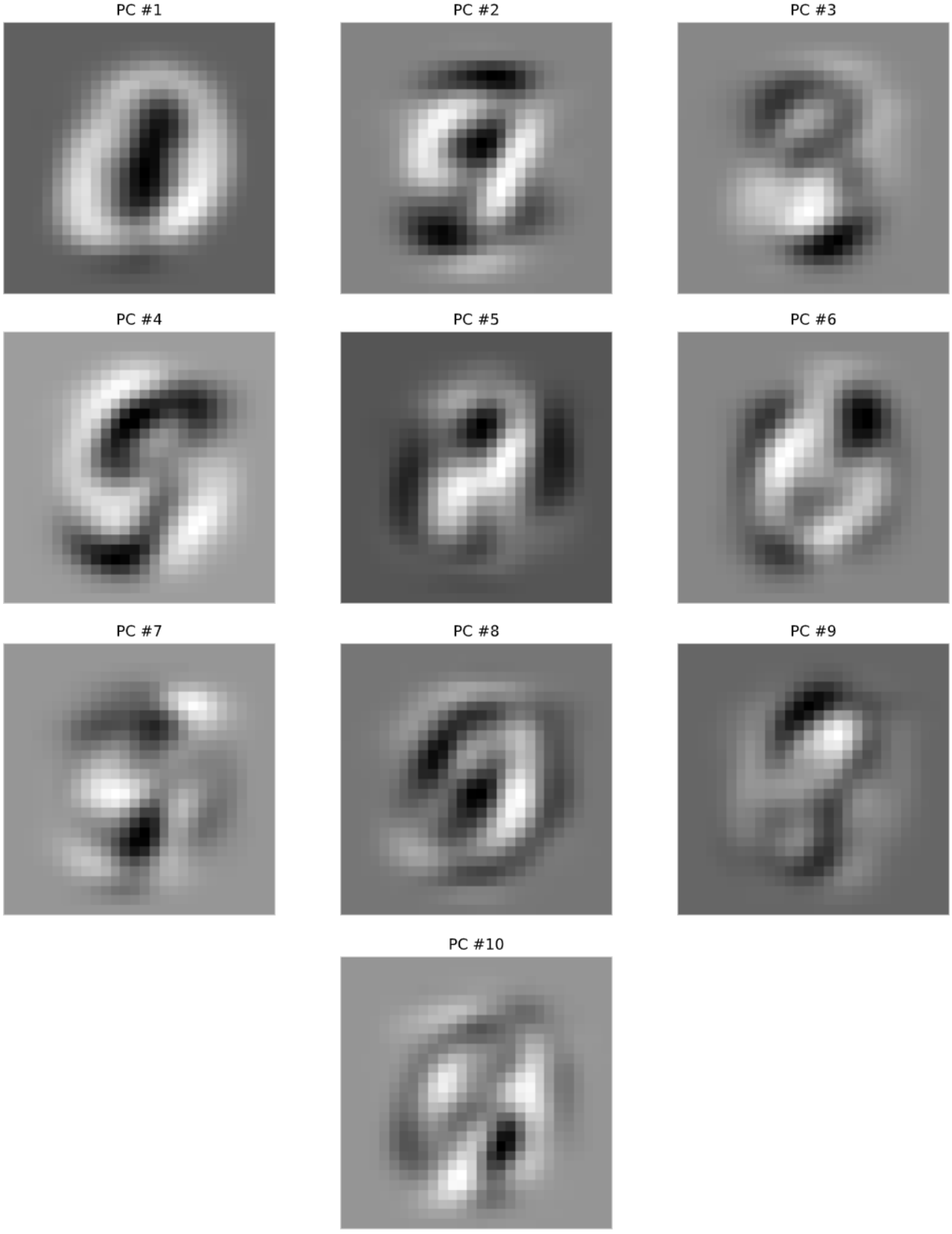

Figure 3: First 10 Principal Components in Grayscale in Detail

Question 1.2 Obtain first k principal components and report PVE for k ∈ {8,16,32,64,128,256}. Plot k vs. PVE and comment on it.

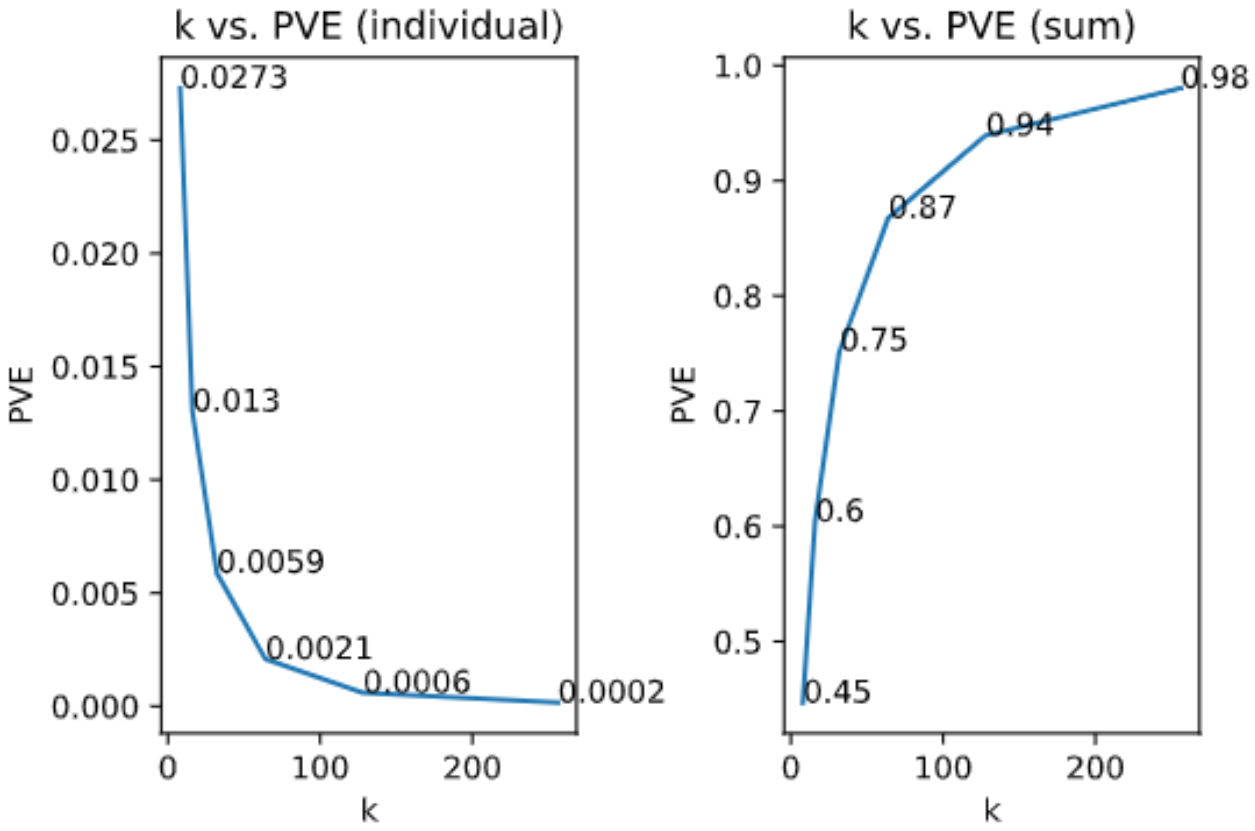

Figure 4: k vs. PVE(Proportion of Variance Explained)

In the first plot in *Figure 4* it can be seen that, with the increased number of k, the proportion of variance explained decreases. The slope of the curve is large for small k (in magnitude), whereas the slope of the curve is small for large k (in magnitude). Proportion of variance explained with respect to principal components decreases from smaller k to larger k. For example, PVE for k=8 is approximately 2.73% and for k=256 PVE is approximately 0.02%.

In the second plot in *Figure 4* it can be seen that increased number of principal components also increases the proportion of variance explained. The slope of the curve is large for small k, whereas the slope of the curve is small for large k. This is the result of proportion of variance explained with respect to principal components decreases from first principal component to last principal component in sorted order.

Using first 8 principal components, approximately half of variance of the data can be explained. Using first 256 principal components, 98% of variance, approximately all variance in the data can be explained. Excluding outliers, first 256 principal components have significant effect regarding explaining the variance in the data in total.

Question 1.3 Describe how you can reconstruct an image using the principal components you obtained in question 1.1. Use first k principal components to analyze and reconstruct the first image in the dataset where k ∈ 1, 3, 5, 10, 50, 100, 200, 300. Discuss your results.

If our data is m x n matrix consisting of m samples and n features, to find first k principal components, we first center data by subtracting mean of the data from every sample. Call mean centered data as X. And let V be n x k matrix consisting of first k eigenvectors where each column corresponds to a eigenvector. Then projections of the data to principal components is given by Z = XV, where Z is a m x k matrix. Z is also called PCA scores. To reconstruct original features from principal components, we can make the second dimension of Z as n by using transpose of V. Each PC should be placed same as the vector used for projection. This can be done by multiplying Z with V^T, $ZV^T = XVV^T$. Finally, we must add mean of the original data. PCA reconstruction formula is:

$$ReconstructedX = X * V * V^T + Mean$$

In our case,

- X is 10000 x 784 matrix consisting of 10000 samples and 784 features.
- V is 784 x k matrix where $k \in 1, 3, 5, 10, 50, 100, 200, 300$
- Z (X*V) is 10000 x k matrix where $k \in 1, 3, 5, 10, 50, 100, 200, 300$
- Reconstructed data is 10000 x 784 matrix

The results of reconstructing first image in the dataset with $k \in 1, 3, 5, 10, 50, 100, 200, 300$ are:

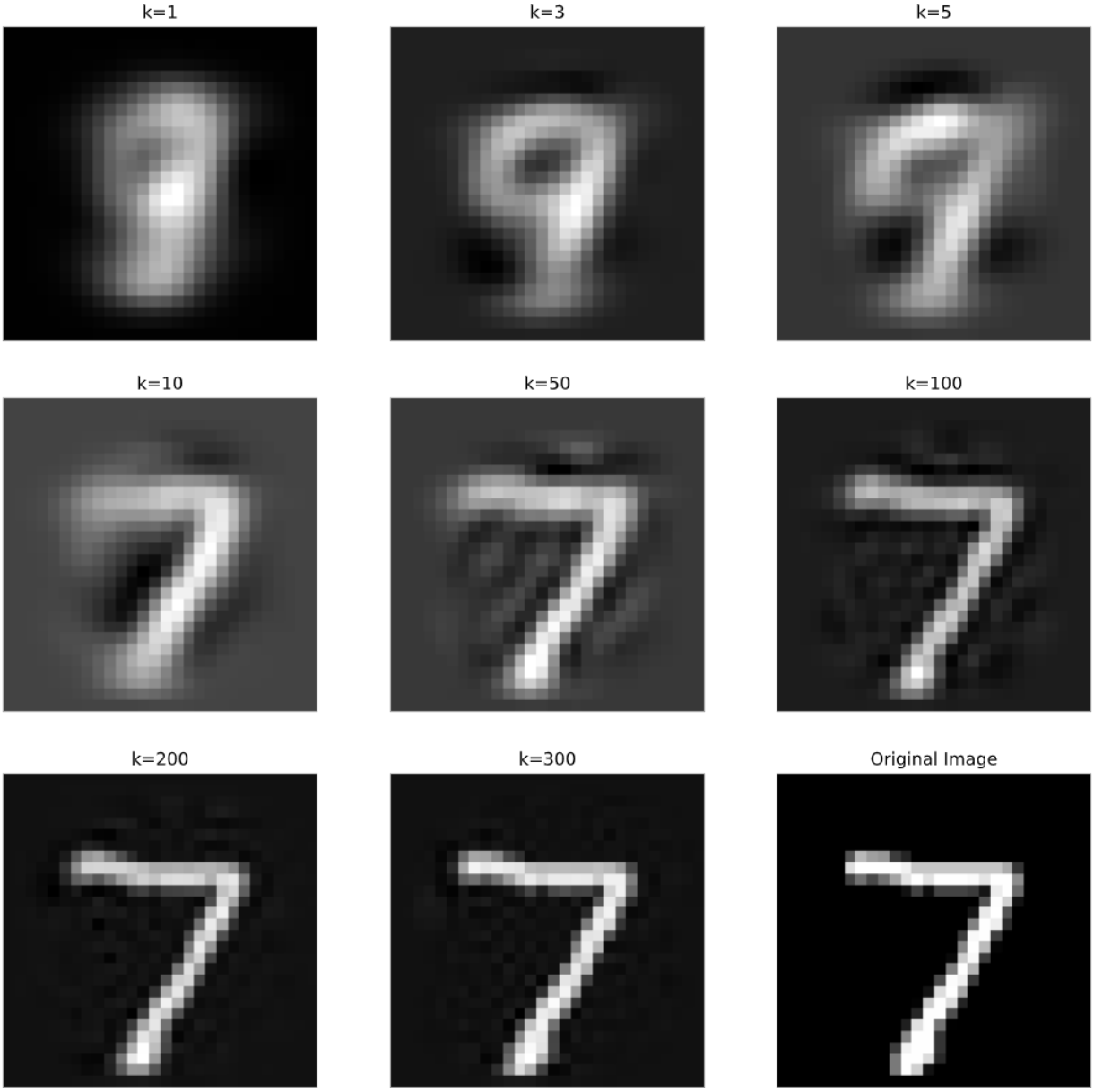

Figure 5: Reconstruction of the First Image in the Dataset

As it can be seen in *Figure 5*, from smaller k to largest k, the reconstructed image becomes more recognizable and more similar to the original image, which is the first image in the dataset. Because as k increases, proportion of variance explained also increases. In the previous parts of the question, it was shown that PVE for k=1 is approximately 10%, and by inspection, PVE for k=300 is close to 98% as PVE for k=256 is approximately 98%. Using the formula for reconstructed X described above, the result becomes more closer to original data.

- For k=1, the image cannot be identified as neither of digits.
- For k=3, the image can be recognized as 9, but it is so blurry.
- For k=5, the image seems like 9 again, however, it has some characteristics of 7.
- For k=10, the image can be recognized as 7 although it is blurry and it has notch at top left.
- For k=50, the image can be recognized as 7, it is blurry but it is less blurry and notch at top left decreases compared to k=10.
- For k=100, the image is similar to original image meanwhile it has some noise compared to original image and pixel are close to gray unlike perfect white and black pixels in the original image.
- For k=200, the image has less noise compared, and black pixels becomes more darker and white pixels become lighter compared to k=100.
- For k=300, the image is the most similar reconstructed image among all reconstructed images to original image. The background color is closer to gray unlike original image and pixels are not perfect black and white compared to original image, however, there is no significant difference between shapes of 7 in the reconstructed image and in the original image. Reducing the second dimension of data from 784 to 300, we still have most of information about the original image and we can discard features or dimensions that have less significant variance. And we discarded more than half of the features.

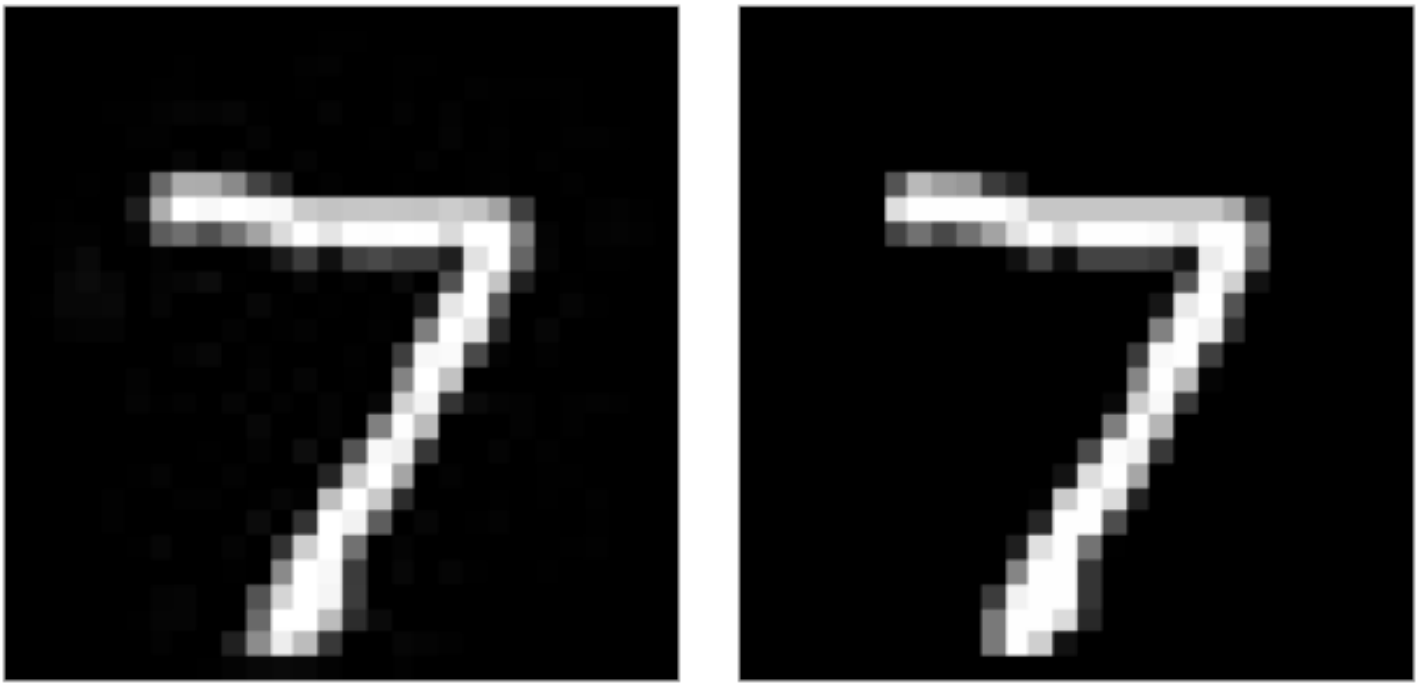

Figure 6: Reconstruction of Image for k=300 (rescaled) vs Original Image

Applying PCA and reducing dimensions would hopefully improve classification.

The original data have grayscale images and pixel values were between 0 and 255. When I reverse PCA and reconstruct images, the pixel values are not in the scale of 0-255, instead there might be pixel values that are less than 0 and greater than 255 hence the reconstructed values are not perfectly scaled. In addition to reconstructed images above in *Figure 5*, I also show the reconstructed images when they are scaled between 0 and 255. As it can be seen in *Figure 7*, the images are more clear compared to the images in *Figure 5*, and they have more clear black and white pixels.

For example, for k=300, the reconstructed image can be considered same as the original image at first glance. The image is almost identical to original image, however, it contains some noise as there are some extra gray pixels compared to original image. The difference can be seen more clearly in *Figure 6*, where images are larger.

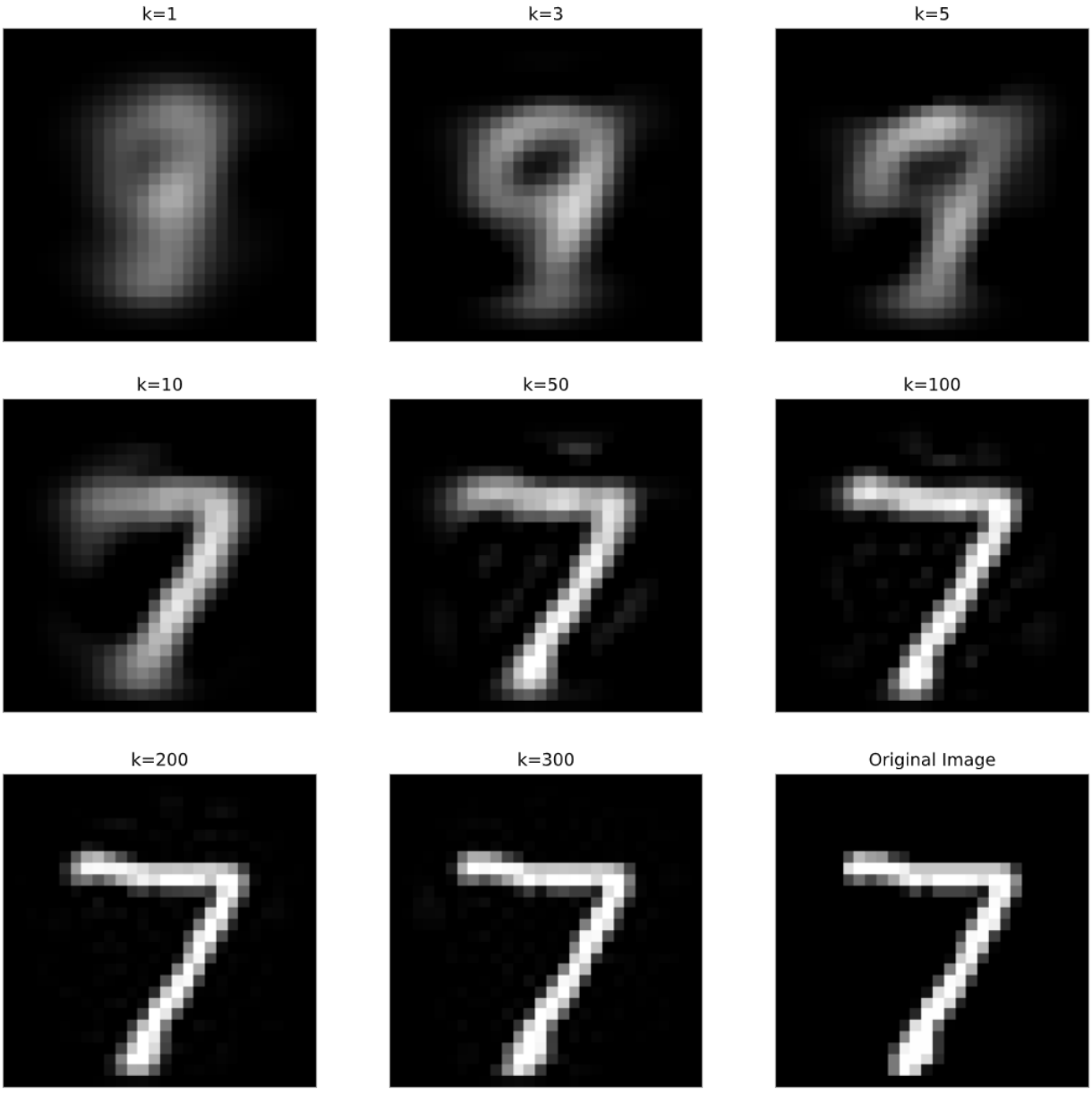

Figure 7: Reconstruction of the First Image in the Dataset (rescaling applied)

2 Linear & Polynomial Regression

Question 2.1 Derive the general closed form solution for multivariate regression model using ordinary least squares loss function. Briefly explain each matrix involved in calculation and how they are constructed.

- y is an 1 x n column vector consisting of ground truth value in the training set. Each y_k corresponds to value of k^{th} (k = 1…n) sample in the training set.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- X is a n x (m + 1) matrix consisting of values of training set, where each row corresponds to a sample in the dataset and each column corresponds to a feature. For example, x_{kl} corresponds to l^{th} feature value of k^{th} sample in the dataset.

$$\begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \dots & x_{1m} \\ 1 & x_{21} & x_{22} & x_{23} & \dots & x_{2m} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & x_{n3} & \dots & x_{nm} \end{bmatrix}$$

- β is an 1 x (m + 1) column vector consisting of weights of features. Each β_k corresponds to weight value of k^{th} (k = 1…m) feature in the training set.

$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix}$$

$$\frac{d\,||Y - X\beta||^2}{d\beta} = \frac{d(Y - X\beta)^T(Y - X\beta)}{d\beta}$$

$$= -2X^TY + 2X^TX\beta$$

Setting the derivative to 0,

$$2X^TY = 2X^TX\beta$$

$$\beta = (X^TX)^{-1}X^TY$$

Question 2.2 Find the rank of $X^T X$ for the given dataset using built-in library functions of your language (rank() for MATLAB, numpy.linalg.matrix rank() for numpy etc.). What does the rank tell you about the solution you have found for Question 2.1.

For the given dataset, the rank of $X^T X$ is found to be 5. Also, shape of $X^T X$ is (5,5). Considering the solution found for Question 2.1, for the given dataset, all rows and columns of $X^T X$ are linearly independent, and $X^T X$ is invertible. In addition, it can be said that the solution found for Question 2.1 can only be applied only if $X^T X$ is invertible.

Question 2.3 Using the formula you have derived for Question2.1, train a linear regression model by using only "sqftliving" feature which is given to you in the dataset. Consider all of the dataset as training set. Report and interpret the coefficients of the trained model. In addition, plot price vs. "sqftliving" along with your model's predictions on the same plot. Finally, calculate MSE using your model's predictions and ground truth labels. Comment on your results.

Coefficients of the trained model are:

- β_0 is -43580.74309447433
- β_1 is 280.62356789744837

The value for a test sample given to the model, the prediction is calculated as y_hat = $\beta_0 + x_1 \beta_1$ where x1 is sqftliving value of the sample. Fitted line of the trained model intercepts with the y axis at -43580.74309447433, and slope of the line is 280.62356789744837. This means, the predicted value for house price starts from -43580.74309447433 where sqft_living is 0, and for each extra square feet value of a house, sqft_living, 280.62356789744837 is added.

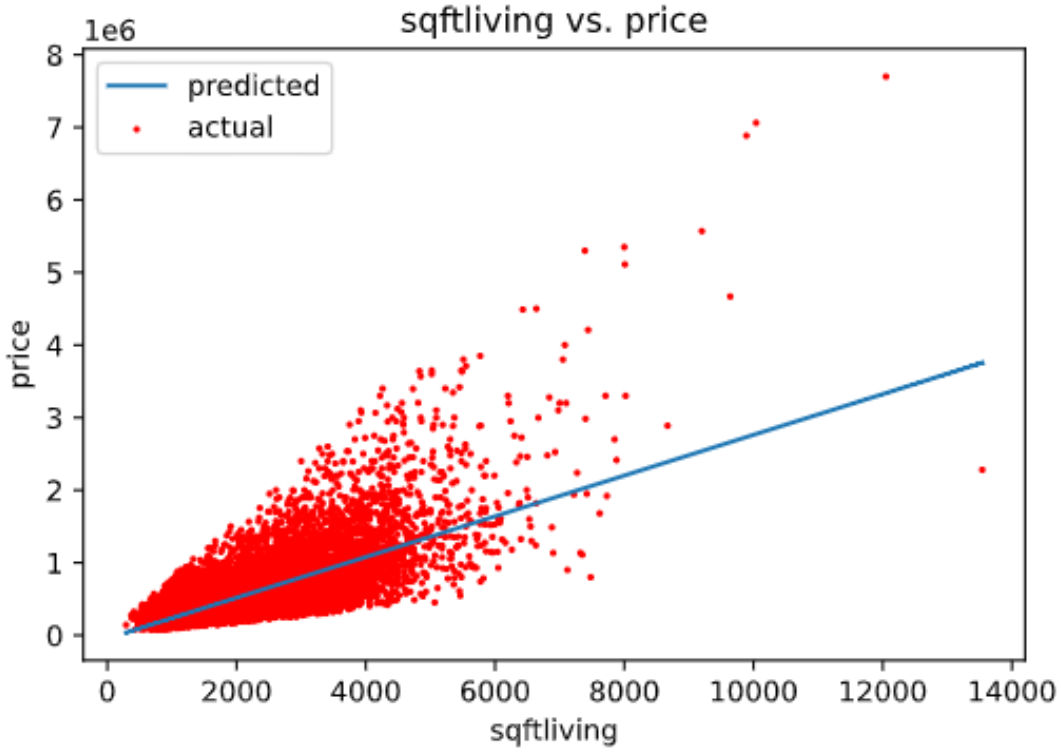

Figure 8: Sqftliving vs. Price

Mean Square Error is 68351286833.039825.

As it can be seen in *Figure 8*, the model is a linear line between the data points with minimum residual error possible in total. The data is scattered around the fitted line and the price varies significantly around the line. For example, for approximately 500 sqftliving, the minimum of prices is approximately than 500000 and the maximum of prices is approximately 3500000. Since most of the data resides between sqftliving values less than 5000, for the values for sqftliving more than 5000, the model will mostly predict a lower price compared to actual price. Except for the sample that is in the most right of the graph, there is huge difference between the predicted value and actual price for larger sqftliving. In addition, since data is sparse around the fitted line, mean squared error is very large although it is lowest error possible. Since we are using only one feature to train the model and there might me other factors or features that can affect the house prices, this large error might be normal.

Question 2.4 In this part, again assume that you are only provided with ”sqftliving” as a feature. You will use polynomial regression to train your model. In this part, you will be using the feature x_i and its powers x_{i^2} . In addition, plot the graph of price vs. ”sqftliving” along with your model’s predictions on the same plot. Finally, provide the coefficients and the training MSE for your model. Assume the whole dataset is your training set. Comment on your results.

Coefficients of the trained model are:

- β_0 is 199222.2793054853
- β_1 is 67.99409468579412
- β_2 is 0.03858126093720183

The value for a test sample given to model, the prediction is calculated as y_hat = $\beta_0 + x_1 \beta_1 + x_2 \beta_2$ where x_1 is sqftliving value of the sample and x_2 is sqftliving value of the sample squared. Fitted line of the trained model intercepts with the y axis at 199222.2793054853. This means, the predicted value for house price starts from 199222.2793054853 where sqftliving is 0, for each extra square feet value of a house, sqftliving, 67.99409468579412 is added, and for each extra square of square feet value of a house, 0.03858126093720183.

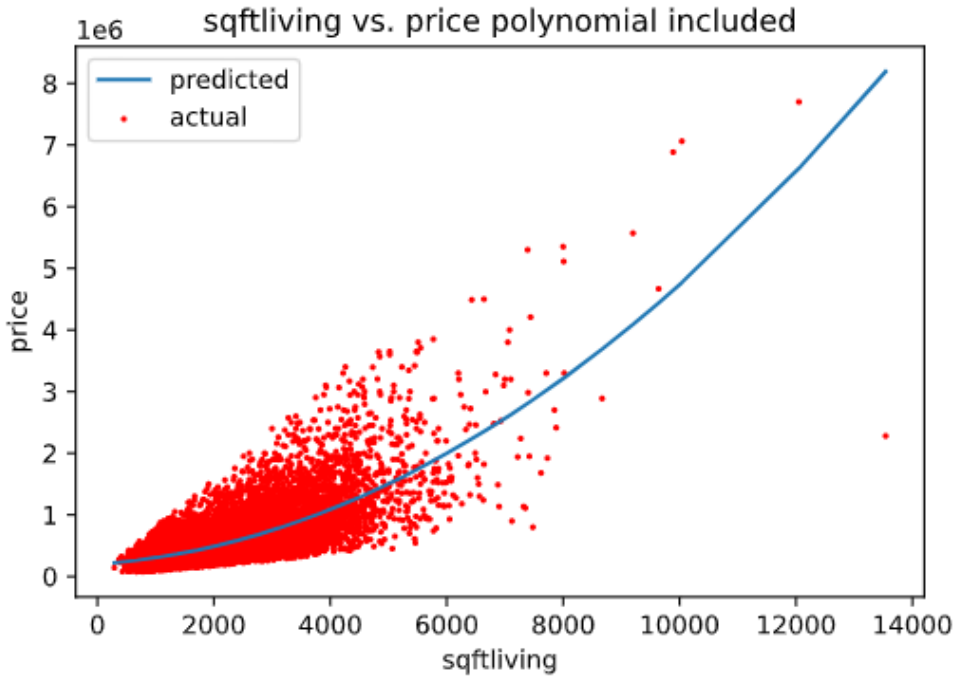

Figure 9: Sqftliving vs. Price Polynomial Included

Mean Square Error is 62975083210.96574.

As it can be seen in *Figure 9*, the model is a polynomial line between the data points with minimum residual error possible in total. The data is scattered around the fitted line and the price varies significantly around the line. For example, for approximately 500 sqftliving, the minimum of prices is approximately than 500000 and the maximum of prices is approximately 3500000. Since most of the data resides between sqftliving values less than 5000, for the values for sqftliving more than 5000, the model will mostly predict a lower price compared to actual price, similar to first model. However, this model predicts larger value for larger sqftliving compared to the first model Except for the sample that is in the most right of the graph, the residual error is reduced compared to first model, however, for the sample that is in the most right of the graph, the residual error increased significantly. In addition, since data is sparse around the fitted line, mean squared error is very large although it is lowest error possible, however, it is reduced 10% when comparing it to first model. Since we are using two features to train the model and there might me other factors or features that can affect the house prices, this large error might be normal.

3 Logistic regression

Question 3.1 You will implement full batch gradient ascent algorithm to train your logistic regression model. Initialize all weights to 0. Try different learning rates from the given logarithmic scale [$10^{-}5$, $10^{-}4$, $10^{-}3$, $10^{-}2$, $10^{-}1$] and choose the one which works best for you. Use 1000 iterations to train your model. Report the accuracy and the confusion matrix using your model on the test set given. Calculate and report averages of precision, recall, negative predictive value (NPV), false positive rate (FPR), false discovery rate (FDR), F1 and F2 scores. (Hint: Check the Amount feature. It needs to be normalized before moving to logistic regression)

Before training my model with training and testing it with test data, I normalized Amount feature with min-max scaling.

To select the best learning rate for the model, I first calculated loss for each leaning rate during each epoch. Losses for each learning rate can be seen in *Figure 10*. As it can be seen in the plot, 1e-1

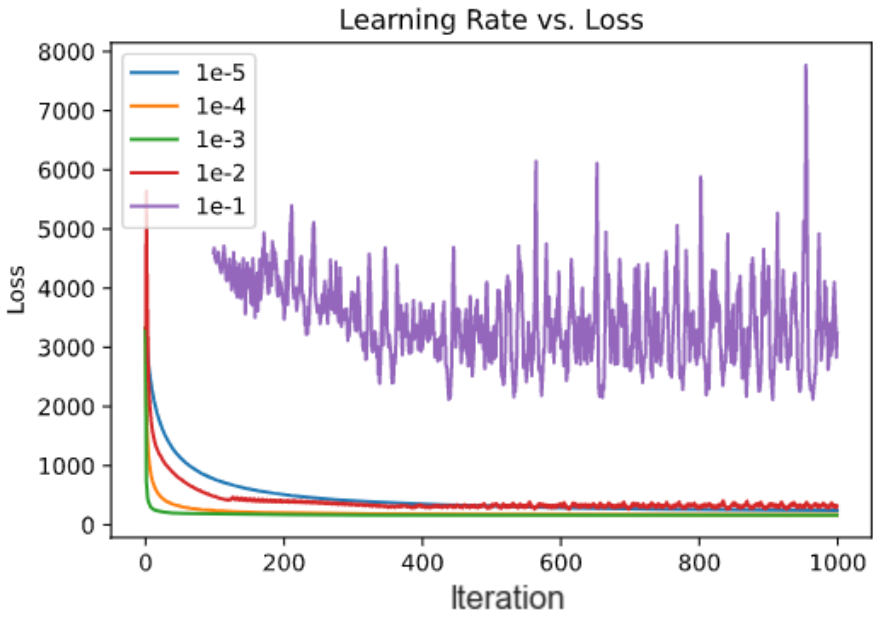

Figure 10: Learning Rate vs. Loss

is not an optimal learning rate for the model as it jumps around the curve many times. Same is can be said about 1e-2, however, the jumps are less significant and are close to optimal point. Comparing 1e-3, 1e-4 and 1e-5, 1e-3 seemed to be the best learning rate, however, I also tested each learning rate with the test data.

Later, I tested the model with all learning rates to see the differences of learning rates except for 1e-1 since it is not a candidate for being the best learning rate. The results can be seen below:

Metrics for learning rate = 1e-2
- Accuracy is: 0.98
- Precision is: 0.8627450980392157
- Recall is: 0.8979591836734694
- Negative Predictive Value is: 0.9908925318761385
- False Positive Rate is: 0.012704174228675136
- False Discovery Rate is: 0.13725490196078433
- F1 Score is: 0.8799999999999999
- F2 Score is: 0.8906882591093118

Metrics for learning rate = 1e-3
- Accuracy is: 0.9858333333333333
- Precision is: 0.9655172413793104
- Recall is: 0.8571428571428571
- Negative Predictive Value is: 0.9874213836477987
- False Positive Rate is: 0.0027223230490018148
- False Discovery Rate is: 0.034482758620689655
- F1 Score is: 0.908108108108108
- F2 Score is: 0.8768267223382045

Metrics for learning rate = 1e-4
- Accuracy is: 0.9858333333333333
- Precision is: 0.9764705882352941
- Recall is: 0.8469387755102041
- Negative Predictive Value is: 0.9865470852017937
- False Positive Rate is: 0.0018148820326678765
- False Discovery Rate is: 0.023529411764705882
- F1 Score is: 0.907103825136612
- F2 Score is: 0.870020964360587

Metrics for learning rate = 1e-5
- Accuracy is: 0.9858333333333333
- Precision is: 0.9764705882352941
- Recall is: 0.8469387755102041
- Negative Predictive Value is: 0.9865470852017937
- False Positive Rate is: 0.0018148820326678765
- False Discovery Rate is: 0.023529411764705882
- F1 Score is: 0.907103825136612
- F2 Score is: 0.870020964360587

The values are close to each other, in fact, for 1e-4 and 1e-5, the values are exactly the same. Weighting precision and recall equally, I decided to use 1e-3 as the learning rate as 1e-3 has the largest F1 score among all learning rates.

In addition, although results are not shown, 1e-1 as the learning rate, the expressions were not correctly calculated as some results resulted in overflow and lead model to learn nothing.

As stated before, the learning rate was chosen to be 1e-3, to clearly show the results and remain consistent, I will restate the results for 1e-3 and display confusion matrix.

- Accuracy is: 0.9858333333333333
- Precision is: 0.9655172413793104
- Recall is: 0.8571428571428571
- Negative Predictive Value is: 0.9874213836477987
- False Positive Rate is: 0.0027223230490018148
- False Discovery Rate is: 0.034482758620689655
- F1 Score is: 0.908108108108108
- F2 Score is: 0.8768267223382045

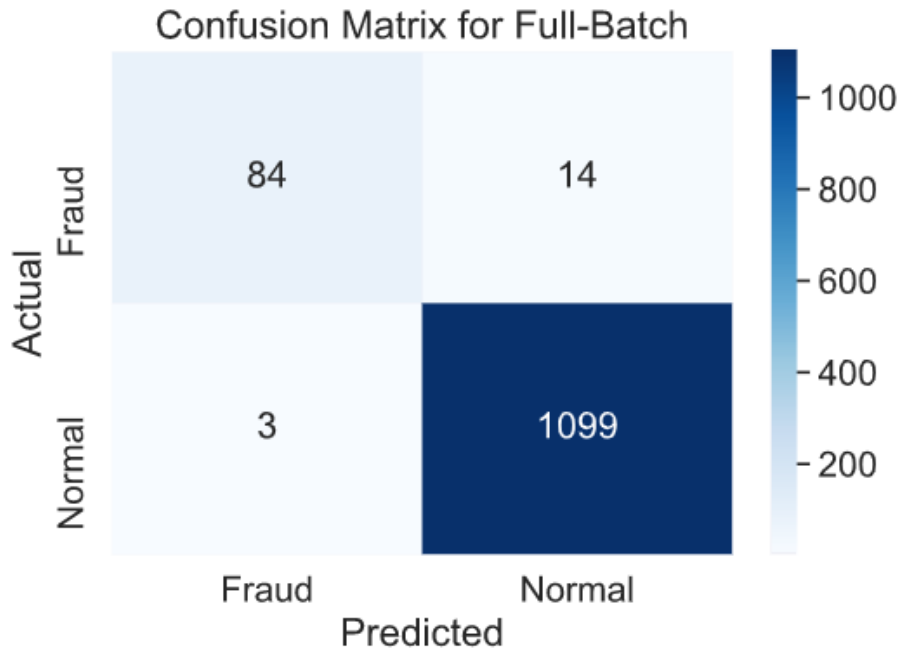

Figure 11: Confusion Matrix for Full Batch Gradient Ascent Algorithm

Question 3.2 You will implement mini-batch gradient ascent algorithm with batch size = 100 and stochastic gradient ascent algorithm to train your logistic regression model. Initialize all weights to random numbers drawn from a Gaussian distribution N(0, 0.01). Use the learning rate you have chosen in Question 3.1 and perform 1000 iterations to train your model. Report the accuracies and the confusion matrices using your models on the given test set. Calculate and report averages of precision, recall, negative predictive value (NPV), false positive rate(FPR), false discovery rate (FDR), F1 and F2 scores.

Training the mini-batch model with learning rate of 1e-3 the metrics for the test sets are stated below:

- Accuracy is: 0.9858333333333333
- Precision is: 0.9655172413793104
- Recall is: 0.8571428571428571
- Negative Predictive Value is: 0.9874213836477987
- False Positive Rate is: 0.0027223230490018148
- False Discovery Rate is: 0.034482758620689655
- F1 Score is: 0.908108108108108
- F2 Score is: 0.8768267223382045

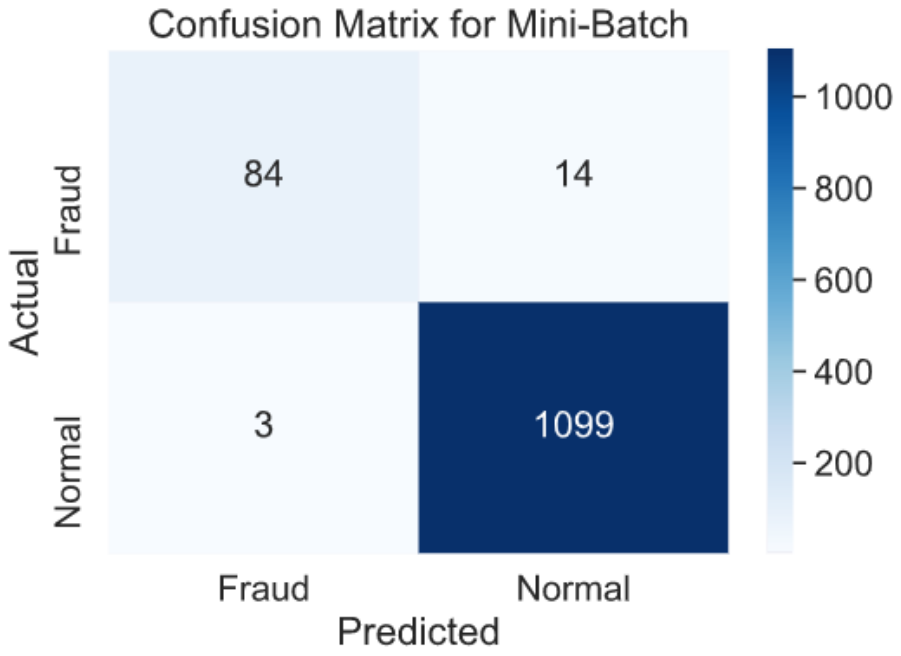

Figure 12: Confusion Matrix for Mini Batch Gradient Ascent Algorithm

Training the mini-batch model with learning rate of 1e-3 the metrics for the test sets are stated below:

- Accuracy is: 0.9858333333333333
- Precision is: 0.9655172413793104
- Recall is: 0.8571428571428571
- Negative Predictive Value is: 0.9874213836477987
- False Positive Rate is: 0.0027223230490018148
- False Discovery Rate is: 0.034482758620689655
- F1 Score is: 0.908108108108108
- F2 Score is: 0.8768267223382045

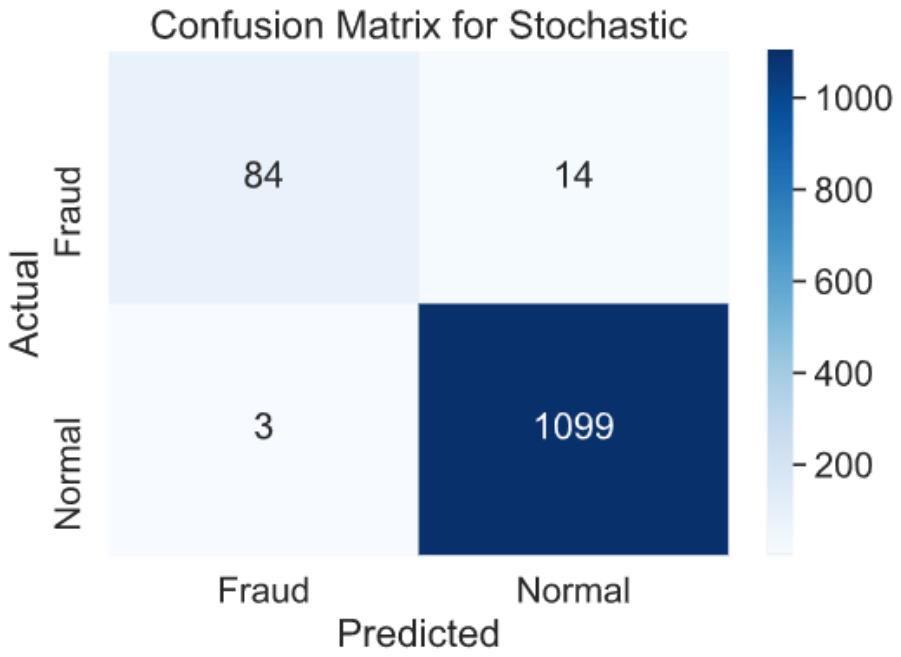

Figure 13: Confusion Matrix for Stochastic Gradient Ascent Algorithm

The metrics for full batch gradient ascent algorithm, mini batch gradient ascent algorithm, and stochastic gradient ascent algorithm are exactly the same. That means, applying the same learning rate to all full batch, mini batch and stochastic algorithm performed the same as given test set. For the given train and test sets, applying full batch is better as mini-batch and stochastic models require more computational power compared to full-batch model.

When examining the weights for full batch model, mini batch model and stochastic model, the coefficients for weights are:

- [-4.88078572 0.50673719 -0.46359559 -0.94188556 0.79194953 0.35276887 -0.30641786 0.08700687 -0.38630198 -0.6100511 -1.05732005 -0.45694303 -0.68458513 -0.72097379 -0.65238005 -0.83997412 -0.23490648 -0.32976182 0.11244955 -0.09132011 -0.76031052 0.82889118 1.64489153 -0.18208467 0.04689058 -0.59679754 0.17359531 -1.11861848 -0.47283791 0.00329194] for full batch model,
- [-4.92740995 0.53208584 -0.44366008 -0.93738618 0.79731864 0.37442663 -0.31992129 0.04089908 -0.37333825 -0.63999791 -1.1056036 -0.46593518 -0.68098308 -0.71988963 -0.6373202 -0.84110425 -0.22884766 -0.32297112 0.11597575 -0.07245454 -0.84765609 0.83008987 1.67376207 -0.15352204 0.05063158 -0.61011588 0.15955669 -1.265053 -0.58053646 3.05336341] for mini batch model and
- [-4.88227549 0.50423504 -0.45825686 -0.9383081 0.7898167 0.35425579 -0.30746583 0.07493112 -0.39124983 -0.60673561 -1.05033753 -0.45891365 -0.68715016 -0.72654883 -0.65454142 -0.83598952 -0.23693706 -0.32695851 0.10940543 -0.09184227 -0.74967809 0.82176798 1.63837861 -0.18102494 0.04498324 -0.59338597 0.17661386 -1.11175865 -0.50302899 0.00180515] for stochastic model.

As it can be seen above, coefficients are really close to each other, differing approximately 5% at most. Since they are so similar, they might performed same in the test data. However, we cannot be sure that if they predict the test data exactly the same, we only know the numbers.

Question 3.3 In what cases, NPV, FPR, FDR, F1 and F2 would be more informative metrics compared to accuracy, precision and recall alone? Explain.

- Accuracy is the proportion of true positives and true negatives among all predictions
- Precision is the proportion of true positives among all positive predictions
- Recall is the proportion of true positive predictions among all positive samples
- NPV is the proportion of true negatives among all negative predictions
- FPR is the proportion of false positives among all negative samples
- FDR is the proportion of false positives among all positive predictions
- F-Measure is the harmonic mean of precision and recall
- F_1-Measure equally weights precision and recall
- F_2-Measure weights recall more than precision

Negative predictive value (NPV) is associated prevalence, and it is the measure of how likely the sample is really negative if prediction is negative. Keeping recall and specificity of the test the same, NPV increases as prevalence decreases. NPV is important when percentage of true negatives among all negative predictions is important. It is calculated as TN / (TN + FN). It shows how accurate are negative predictions. If not missing positives is a significant matter, i.e positive samples should not be predicted as negative, we would want to minimize false negatives and maximize NPV, like Covid-19. In this case, NPV rate is important.

False positive rate (FPR) is the measure of how likely the sample will be predicted as positive if sample is really negative. FPR is important when percentage of false positives among all negative samples is important . It is calculated as FP / (FP + TN). It shows how misleading are positive predictions. If not missing negatives is a significant matter, i.e negative samples should not be

predicted as positive, we would want to minimize false positives and minimize FPR, like pregnancy test. We wouldn't want to make false alarms. In this case, FPR rate is important.

False discovery rate (FDR) is the measure of how likely the sample is really negative if prediction is positive. FDR is important when percentage of false positives among all positive predictions is important. It is calculated as FP / (FP + TP). It shows how misleading are positive predictions. If not missing negatives is a significant manner, i.e positive predictions should resemble the truth, we would want to minimize false positives and minimize FDR. Like mail labeling, we wouldn't want to label a ham mail as spam, because important mails might be discarded. In this case, FDR is important.

F-measure harmonically finds the mean of precision and recall. F-measure is important when we want to use precision and recall together. It is the harmonic mean and arithmetic mean because it also considers outliers. It is calculated as $(1 + \beta^2)PR / (\beta^2P + R)$. F score is important when data is distributed non-uniformly. When $\beta=1$, i.e F1, recall and precision are weighted equally, when $\beta=2$, i.e F2, F-measure favors recall.