Programming Assignment 1
NA17B014
ME17B076

CS5691 Pattern Recognition and Machine Learning

Programming Assignment 1

Report submitted by

Mizhaan Maniyar (NA17B014)

Arnesh Kumar Bose (ME17B076)

Group 12

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
CHENNAI 600036, INDIA.
Jan-May 2020

Question 1

*(Note: **MATLAB** was used for this problem)*

Assumptions:

(For Model 2 and Model 4, the covariance matrices were calculated by centering the data points with respect to their own means pertaining to their class and not by the total mean of the dataset. This covariance matrix was thus calculated by taking the weighted average of the covariance matrices for each of the 3 classes.)

The accuracy of each model was measured as the ratio of the number of correct classifications by the total sample number of the dataset. Given below are the accuracies of each model for both the datasets:

Dataset 1					
Accuracy %	**Model 1**	**Model 2**	**Model 3**	**Model 4**	**Model 5**
Training Set	69.94%	96.92%	96.92%	96.92%	**96.92%**
Test Set	71.33%	96.22%	96.22%	96.22%	**96.22%**

Dataset 2					
Accuracy %	**Model 1**	**Model 2**	**Model 3**	**Model 4**	**Model 5**
Training Set	53.64%	85.11%	87.28%	86.39%	**90.83%**
Test Set	52.33%	85.22%	87.11%	86.39%	**90.00%**

Although, for the Dataset 1 the accuracies for all models except Model 1 are the same for test as well as the training set, we can see that for Dataset 2 the accuracies for Model 5 is the highest. Hence, we shall take Model 5 as our 'best model' for classification.

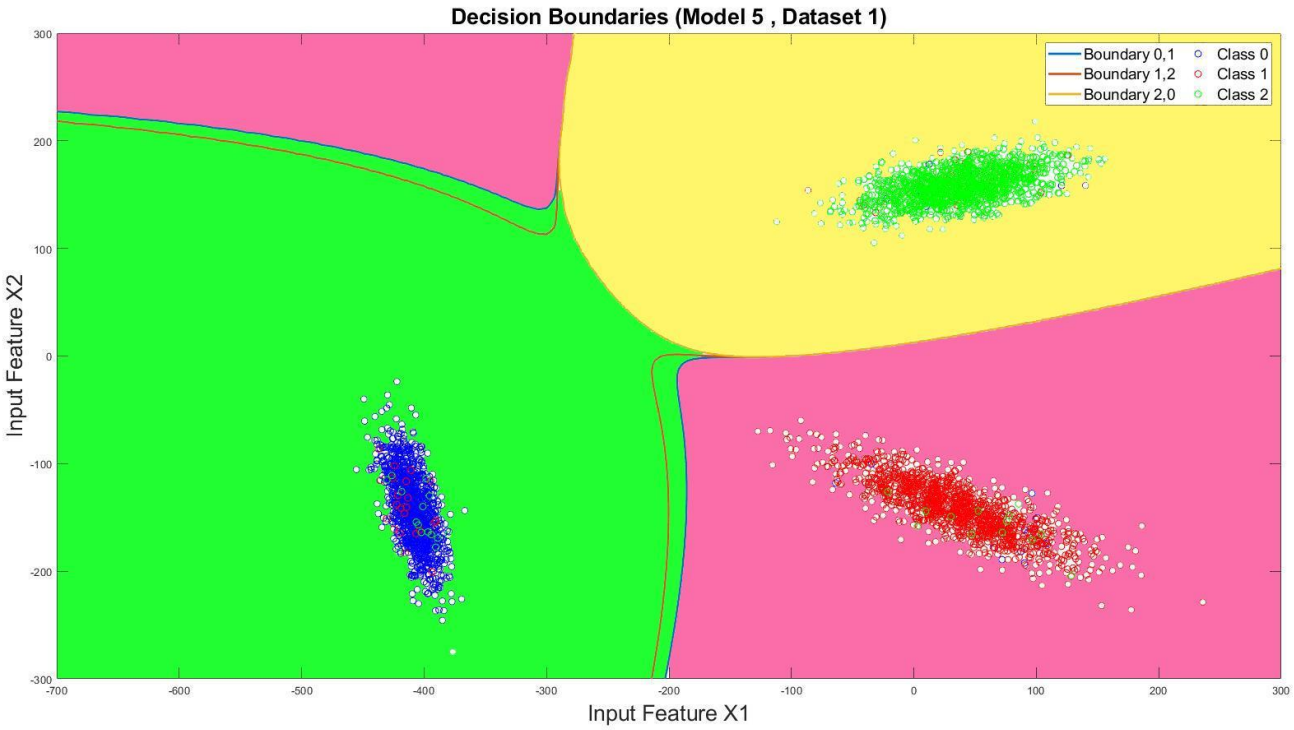
After calculation of the accuracy table we formulated the Confusion matrix for the best model, in this case the best model being Model 5. *(Note that the confusion matrix was based on the test set.)*

Dataset 1		**Confusion Matrix for Model 5 with accuracy 96.22%**			
Output Class	**Class 0**	**286**	**4**	**7**	96.30%
		31.78%	0.44%	0.78%	3.70%
	Class 1	**5**	**292**	**5**	96.69%
		0.56%	32.44%	0.56%	3.31%
	Class 2	**7**	**6**	**288**	95.68%
		0.78%	0.67%	32.00%	4.32%
		95.97%	96.69%	96.00%	**96.22%**
		4.03%	3.31%	4.00%	**3.78%**
		Class 0	**Class 1**	**Class 2**	
		Target Class			

Dataset 2		Confusion Matrix for Model 5 with accuracy 90%			
Output Class	Class 0	283	5	16	93.09%
		31.44%	0.56%	1.78%	6.91%
	Class 1	9	281	20	90.65%
		1.00%	31.22%	2.22%	9.35%
	Class 2	6	34	246	86.01%
		0.67%	3.78%	27.33%	13.99%
		94.97%	87.81%	87.23%	**90.00%**
		5.03%	12.19%	12.77%	**10.00%**
		Class 0	**Class 1**	**Class 2**	
		Target Class			

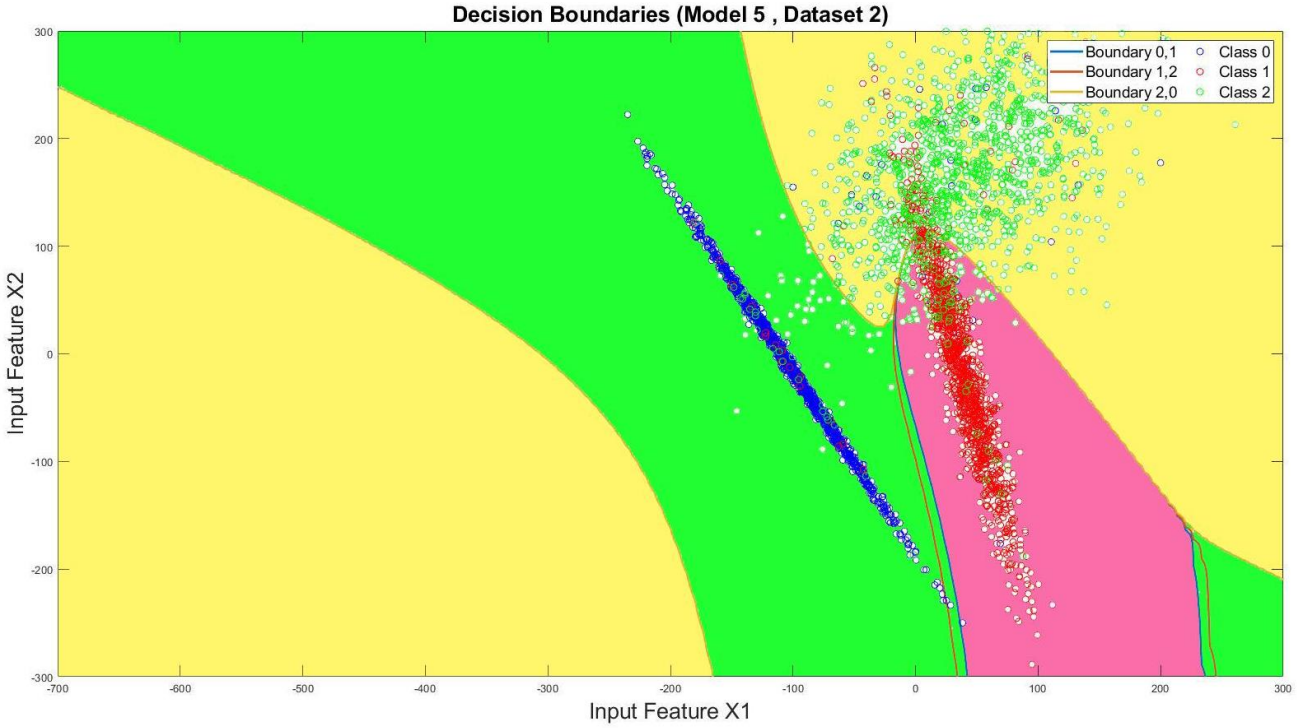
To visualize how this model classifies the training dataset, we can define *decision boundaries* which shall define regions corresponding to each of the classes. Hence, a point shall be classified into a particular class depending on the region it lands on.

The decision boundaries were defined according to the Bayes Classifier taking into consideration the loss function. Given below is the result:

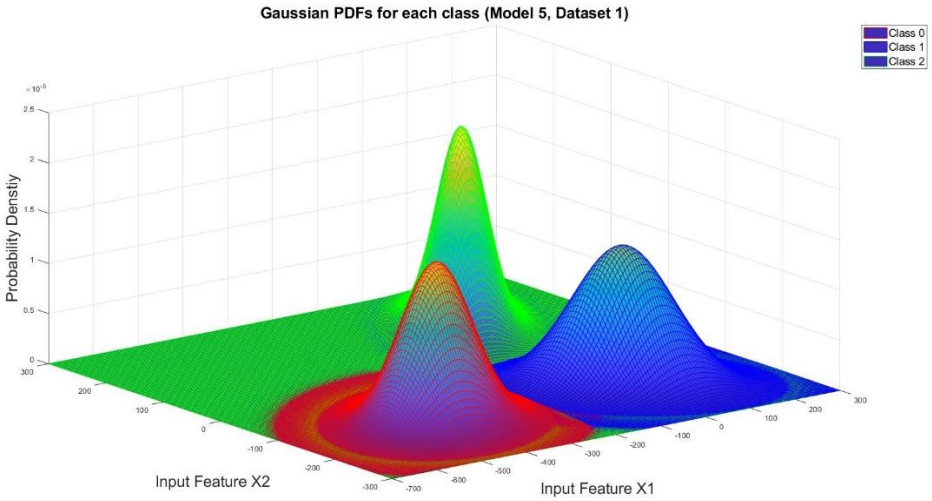


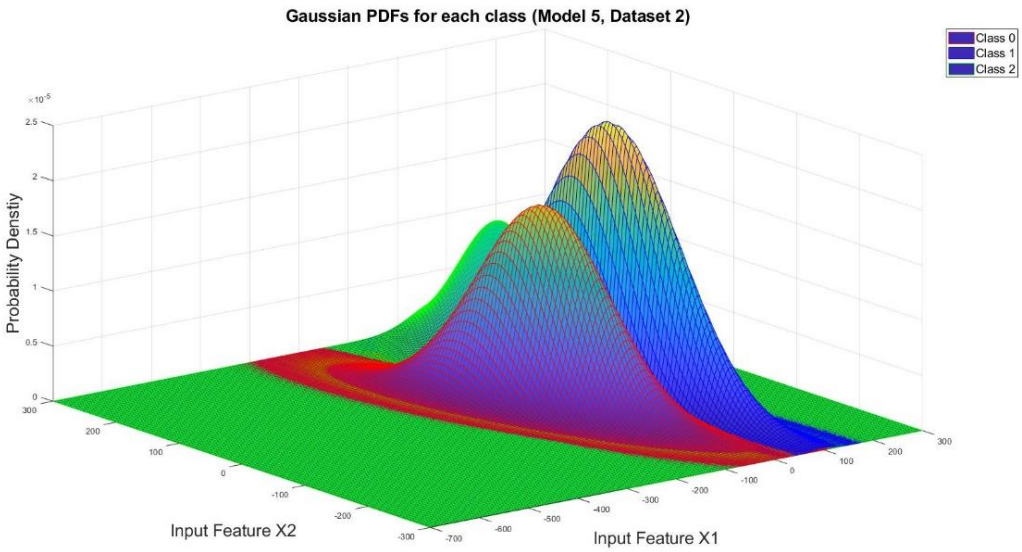
(The region in 'Green' is classified as 'Class 0', the region in 'Pink' is classified as 'Class 1' and the region in 'Yellow' is classified as 'Class 2'.)

As we can see above that the region on the top left of the map in pink is classified as Class 1 even though it is furthest away from the mean of the points belonging to that class, because of our inclusion of loss function, which states that the loss associated by classifying that region as Class 1 is lesser than the other two.

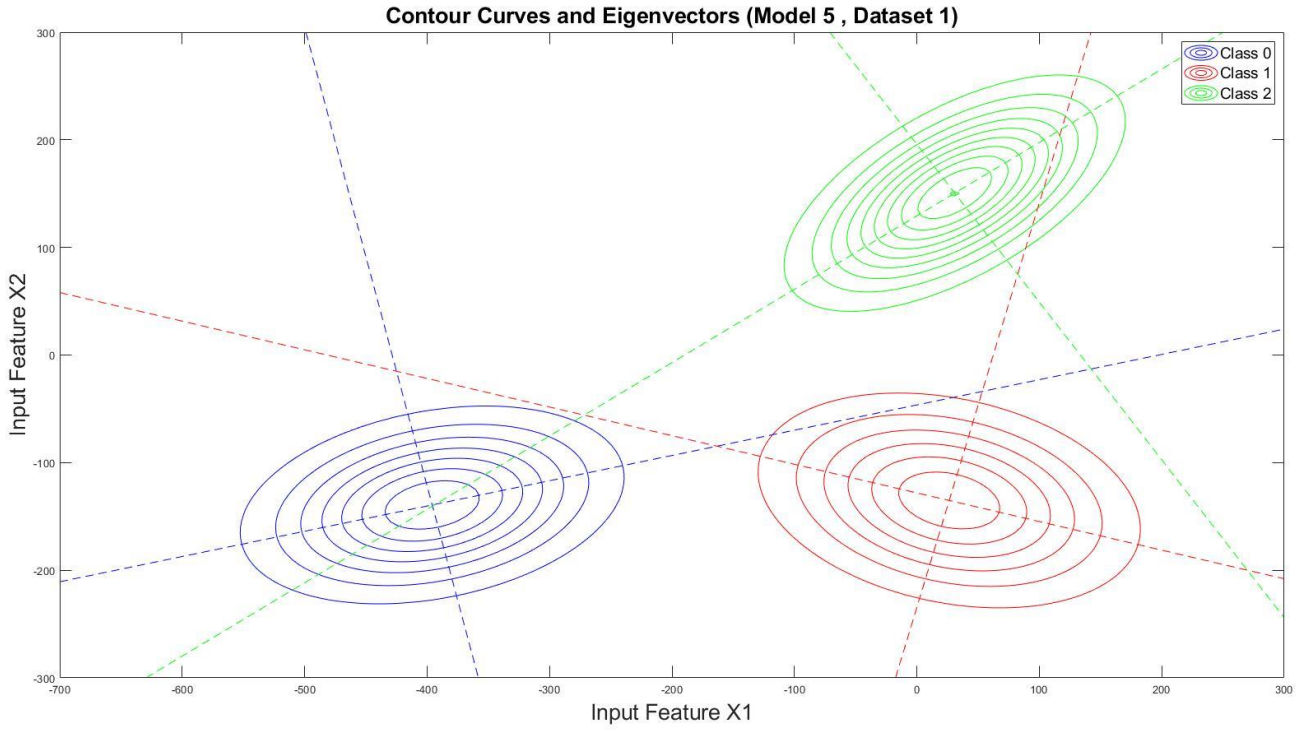


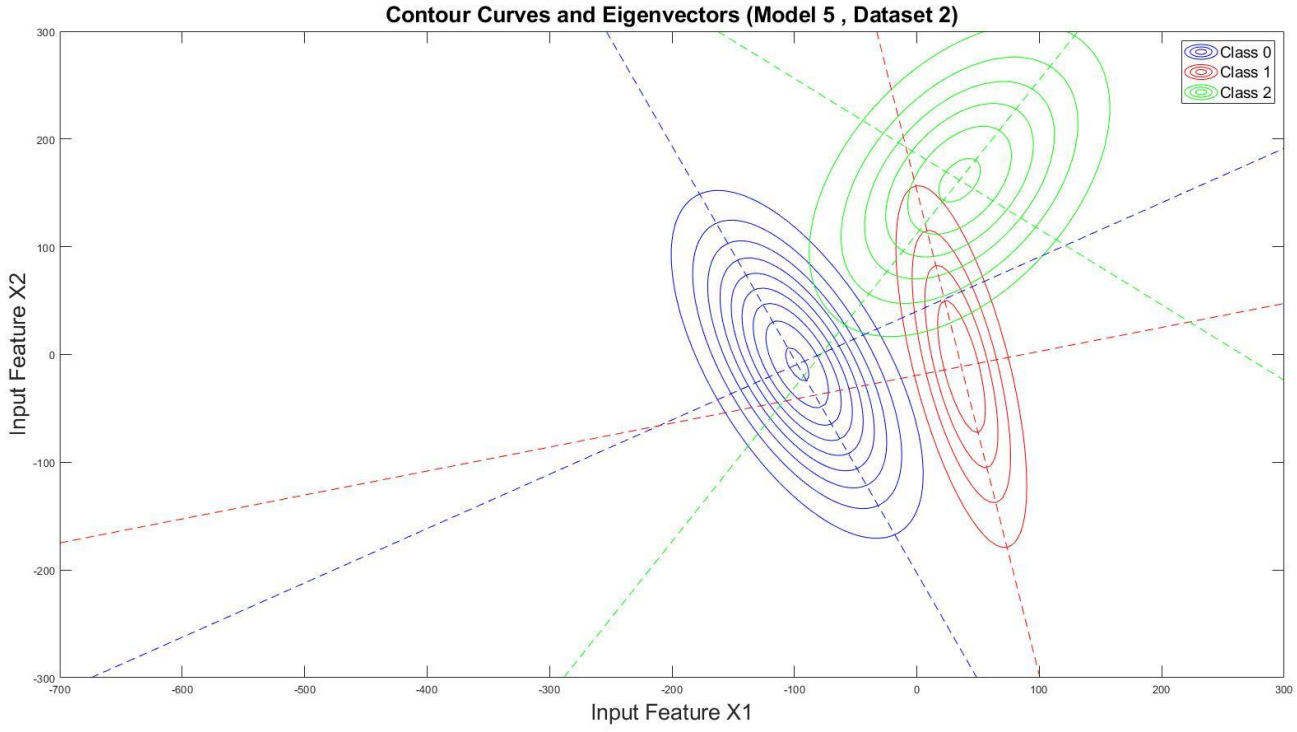
Note that for both the datasets, the region in between *Boundary 0,1* and *Boundary 1,2* is classified as Class 0 as the loss associated here is minimum for Class 0. Hence, we can say that the inclusion of *Boundary 1,2* for this classification is of no use. Following are the three dimensional p





Contour Curves and Eigenvectors of the covariance matrix for Model 5 were plotted together and the results thus obtained are given in the following figure:





As we can see from the above plots, the eigenvectors are perpendicular to one another and run along the major and minor axes of their respective contour curves.

Conclusions:

When we compare the results of the two datasets, we can see that unlike dataset 1, dataset 2 has points that are not that well separated. There is a lot of overlap between the points especially if we see for the points belonging to Class 1 and Class 2. Therefore, even the decision boundaries for the second dataset seem a little ‘squished’ due to high variance along one eigenvector and low variance along the other.

Question 2:

(a) On performing the Maximum Likelihood Estimation on the one-dimensional gaussian dataset 3, the estimate of mean and sigma are as follows:

$$\mu = 1.1196 \qquad \sigma = 4.1537$$

(b) Under the assumption that the prior mean $\mu_0 = -1$, the posterior probabilities are calculated for random sample sizes of 10, 100 and 1000.

Inferences:

- In figure 1, where the sample size is small, the posterior probability distributions are tending to have a mean very near to the prior mean which is -1, because we do not have enough samples to influence the prior distribution.
- As we go on increasing the sample size, the posterior mean shifts away from the prior mean to the sample mean which is 1.1196, as if we have a huge number of samples, the prior distribution becomes insignificant. This can be seen in figure 2 and 3.
- For a fixed number of samples, if we make the prior variance very small compared to the sample variance, then the posterior variance also tends to be small because even if observe large variance in dataset, we attribute it to the large sample variance, and not the posterior variance.
- As we go on decreasing the sample variance compared to the prior variance, the posterior variance increases, as now we attribute the variance in dataset to the variance of the distribution of mean.

The plots for the posterior probabilities are as

Figure 1

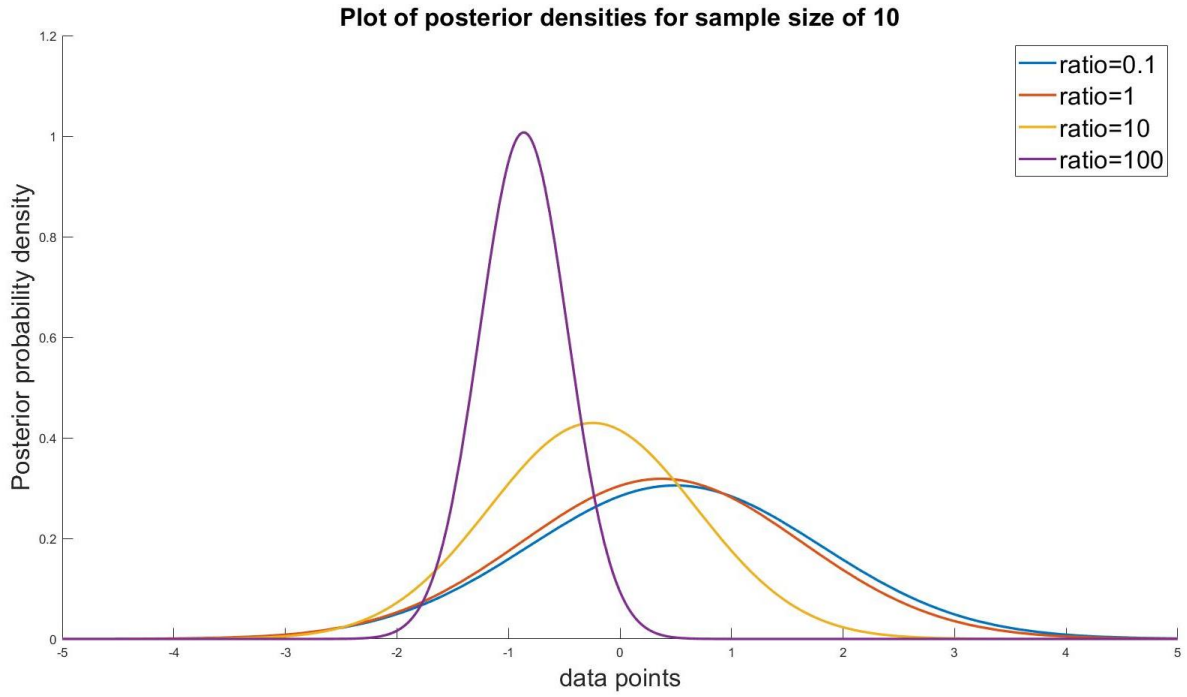


Figure 2

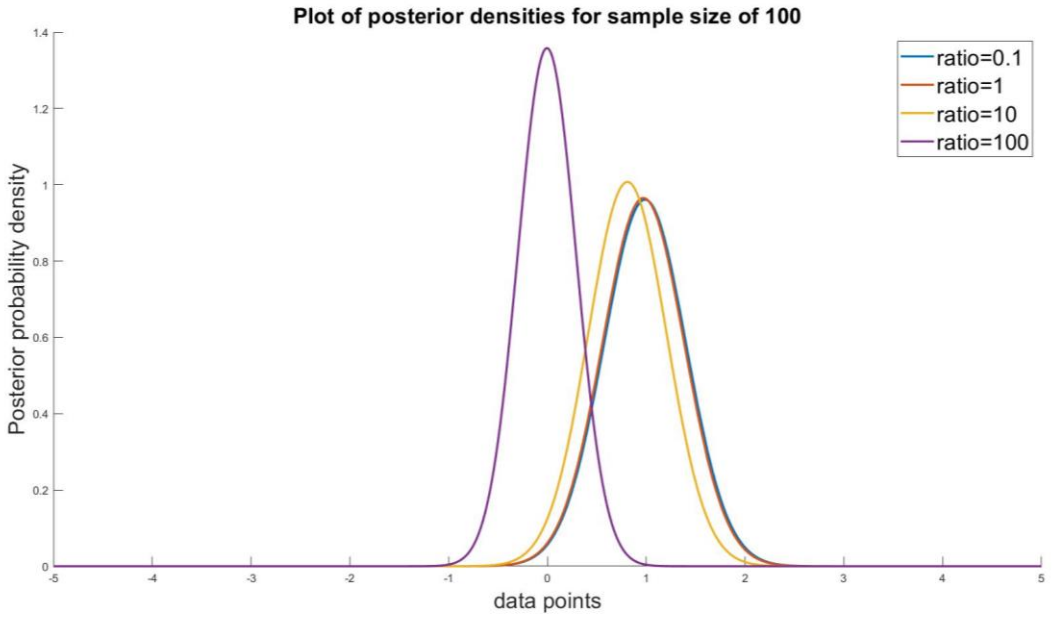
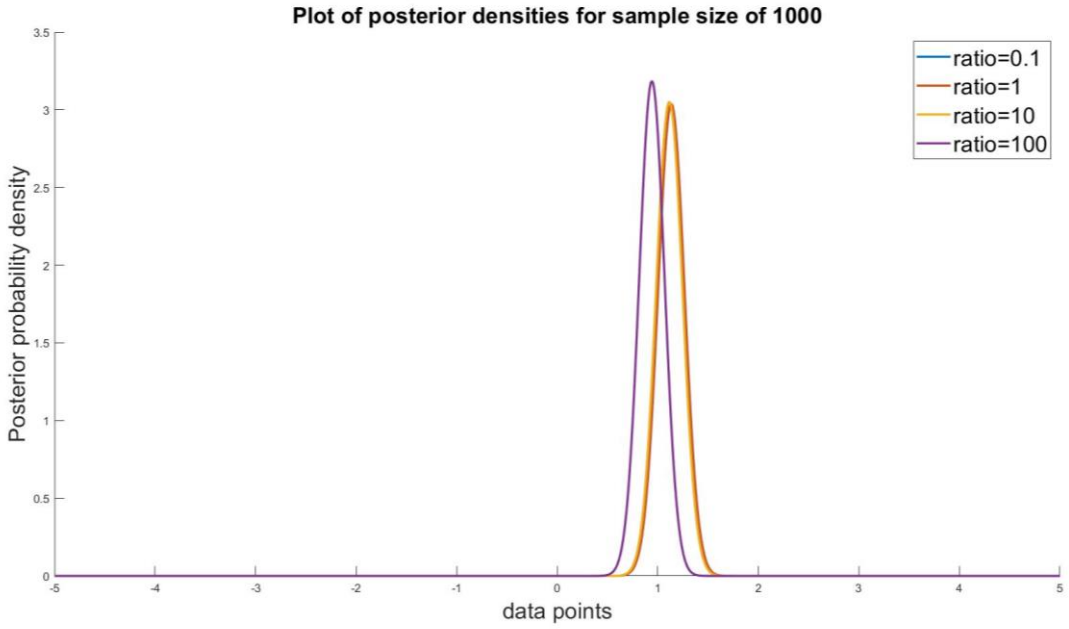


Figure 3



Formulas used: The following formulas are used to find the posterior probability

$$\frac{1}{\sigma_n^2} = \frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}$$

$$\frac{\mu_n}{\sigma_n^2} = \frac{S_n}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}$$

Question 3:

(a) The three sets of 20 observations were generated from the three equally probable three-dimensional Gaussian distributions. The maximum likelihood estimators of mean and covariance are the sample mean and covariance, which are as follows:

$$\hat{\mu}_1 = (-0.0529, 0.1691, 0.1683)^T$$

$$\hat{\Sigma}_1 = \begin{pmatrix} 3.2372 & 0.833 & 0.5594 \\ 0.833 & 7.7250 & -0.8348 \\ 0.5594 & -0.8348 & 1.3947 \end{pmatrix}$$

$$\hat{\mu}_2 = (0.4389, 4.5014, -4.644)^T$$

$$\hat{\Sigma}_2 = \begin{pmatrix} 1.3244 & 0.1464 & -0.7796 \\ 0.1464 & 4.6416 & 0.3335 \\ -0.7796 & 0.3335 & 3.5294 \end{pmatrix}$$

$$\hat{\mu}_3 = (-0.4091, 0.2494, -0.2693)^T$$

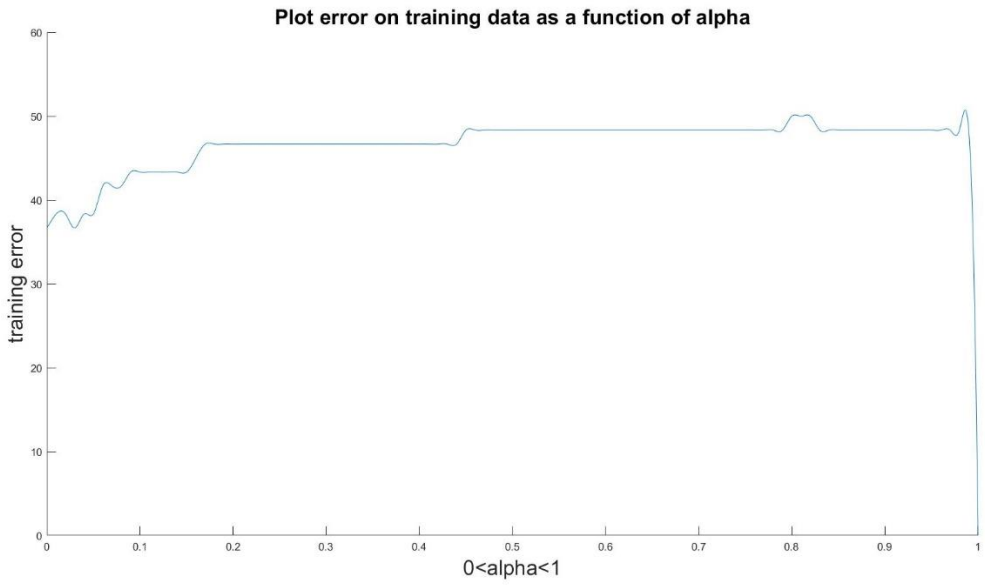
$$\hat{\Sigma}_3 = \begin{pmatrix} 8.9219 & -1.9264 & -1.8772 \\ -1.9264 & 7.1163 & 0.2498 \\ -1.8772 & 0.2498 & 5.9927 \end{pmatrix}$$

(b) The new covariance matrices for each class are generated using the formula

$$\Sigma_i(\alpha) = \frac{(1-\alpha)n_i\Sigma_i + \alpha n\Sigma}{(1-\alpha)n_i + \alpha n}, 0 < \alpha < 1$$

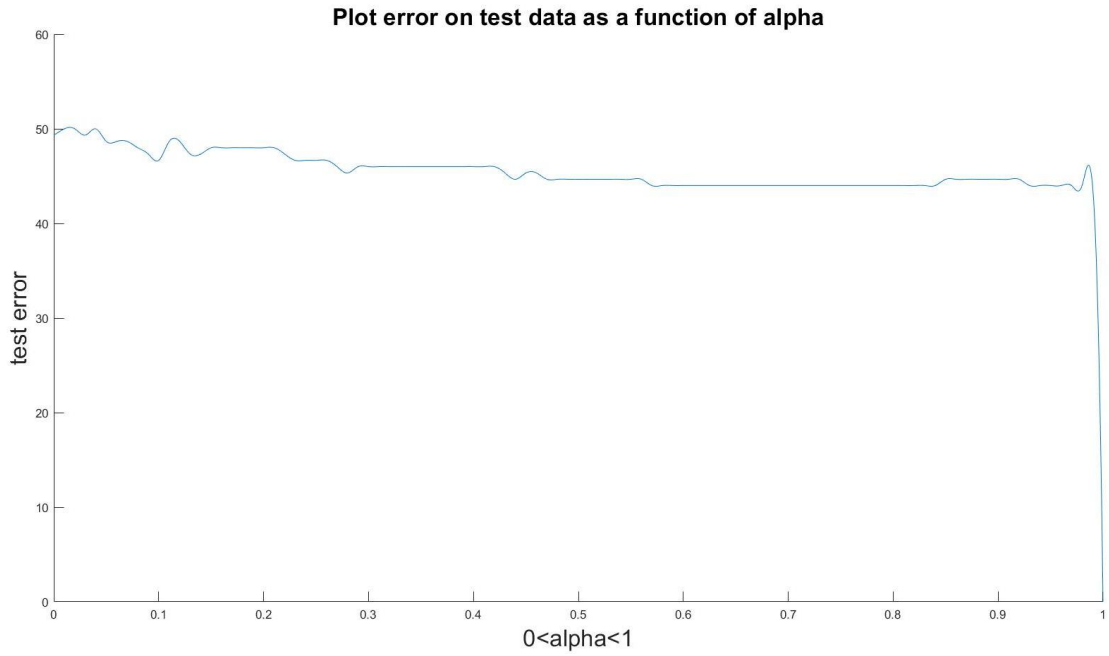
The error on training data as a function of α is as follows:

Figure 1



(c) Then, 50 more test points were generated using the same original distribution so as find the error on test data as a function of alpha.

Figure 2



Inferences:

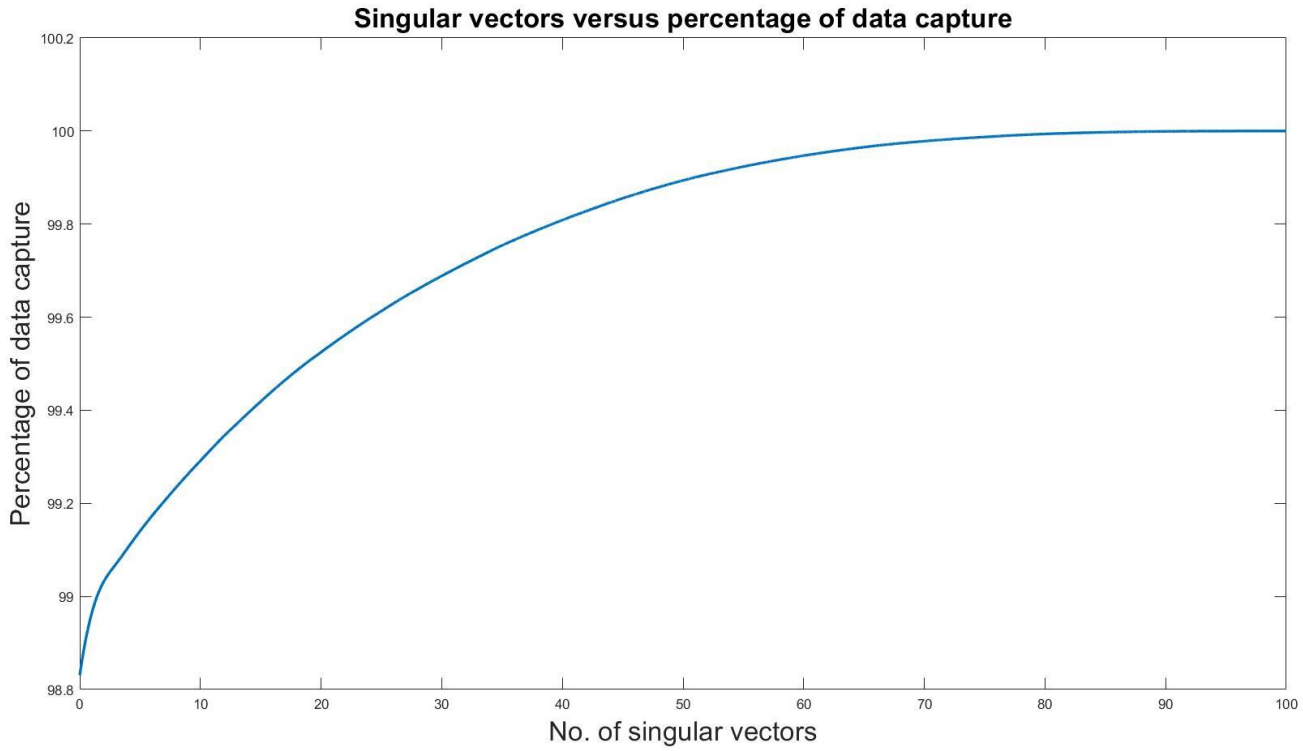
- The estimated means and covariances of the randomly generated samples are not as accurate because we have generated a small number of samples, and hence we see the deviation.
- The shrinkage formula is used to reduce overfitting of the training error and hence we can see that the training error goes up as we increase alpha in figure 1.
- The test error goes down as the value of alpha increases and hence the overfitting is reduced.

Question 4:

In order to create highly correlated columns with entries between 0 and 1, we first generated a single column with values from a uniform random variable between 0 and 1. The other columns were generated by adding a gaussian noise of variance 0.1 and then the interval is scaled down to (0,1). Hence we got a 100*100 matrix A.

(a) The Frobenius norm of the matrix A is 57.7559.
(b) The top 10 singular vectors capture about 99.2915% of the Frobenius norm of the matrix A.
(c) The random 10 singular vectors capture only about 3.9928% of the Frobenius norm.
(d)

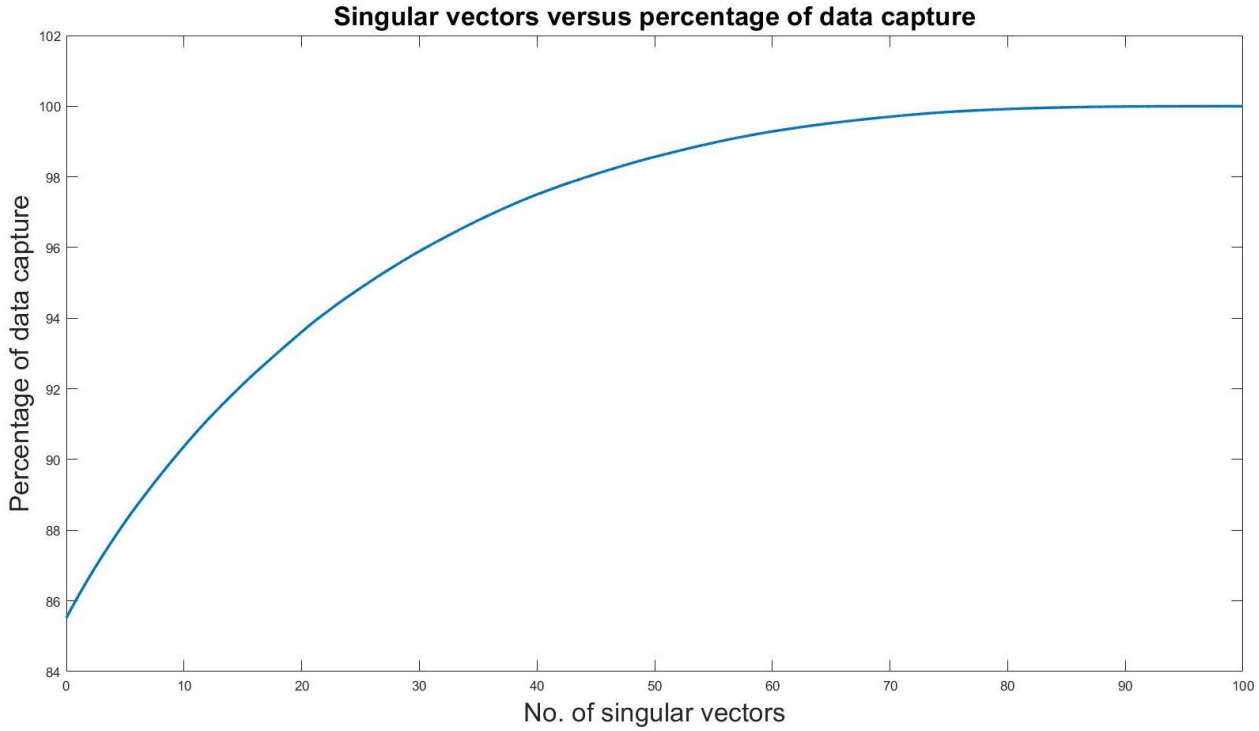
Figure 1



(e) Since the columns are highly correlated, the highest singular value captures about 98.8% of the Frobenius norm as there is typically a single column, and other columns are highly dependent on it. That is why the remaining 99 singular values capture only 1.2% of the Frobenius norm. This can be seen in Figure 1.

(f) Now we have generated a statistically independent matrix with entries derived from a uniform distribution between 0 and 1.

The Frobenius norm of the matrix A is 58.9156.
The top 10 singular vectors capture about 90.3723% of the Frobenius norm of the matrix A.
The random 10 singular vectors capture only about 12.5656% of the Frobenius norm.



Although the columns are statistically independent, since the range of values is small, we are getting a high correlation and the highest singular value is capturing around 85.8% of the Frobenius norm which is less than the previous case.

Question 2

*(Note: **MATLAB** was used for this problem)*

Algorithm: The colored image matrix was imported using **imread()** function and was then converted to grayscale by using **rgb2gray().** The rows of the original image matrix *X*, were taken as the feature vectors on which we are supposed to apply PCA. The reconstructed image matrix *Y* was then formed by taking the approximations of the row vectors of *X* as its row vectors, by using the following formula :

$$y_n = \bar{x} + \sum_{i=1}^{M} (x_n^T u_i - \bar{x}^T u_i)$$

(x_n are the rows of the input matrix, y_n are the rows of the reconstructed image matrix and u_i are the principal components.)

Given below are the plots obtained by taking the top N eigenvectors as our principal components. Here we can visually observe that with increasing % , the quality of our reconstructed image gets better.
We measure the quality of our reconstructed image as:

$$Q = \frac{||Y||_F}{||X||_F} \times 100\%$$

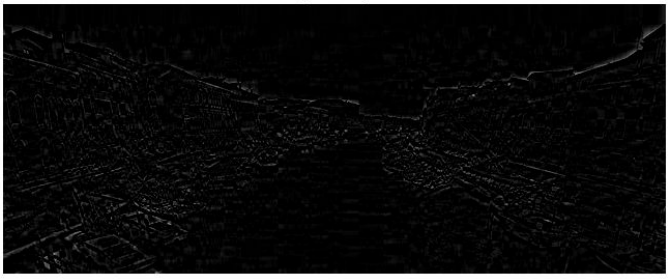
(Here $||A||_F$ is the frobenius norm of the matrix A; and X and Y denote the original image matrix and reconstructed image matrix respectively.)

The quality of the the matrix *Y* reconstructed by using the top N eigenvectors is as follows:

Top N eigenvectors (%)	10	25	50
Quality of *Y* (%)	99.21	99.75	99.95

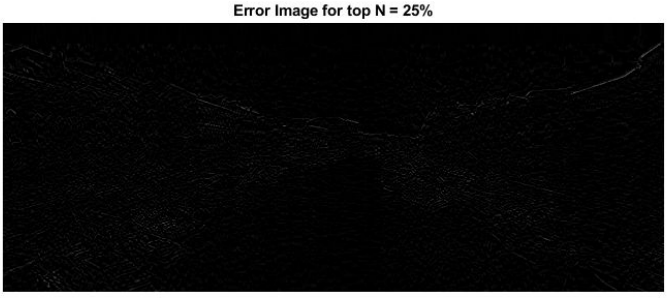
Reconstructed Image for top N = 10%

Error Image for top N = 10%



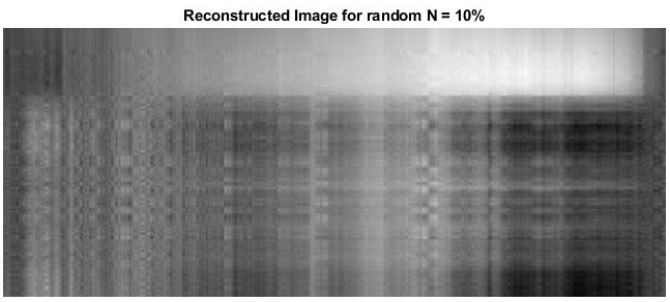
Reconstructed Image for top N = 25%

Error Image for top N = 25%



As we can observe, the error image keeps getting darker for higher % of top eigenvectors, which implies that the difference between the matrix *X* and matrix *Y* decreases with increasing % of top N.

Now if we take random 10% of the eigenvectors as our principal components and apply the same algorithm, we get the following images along with their respective error images:

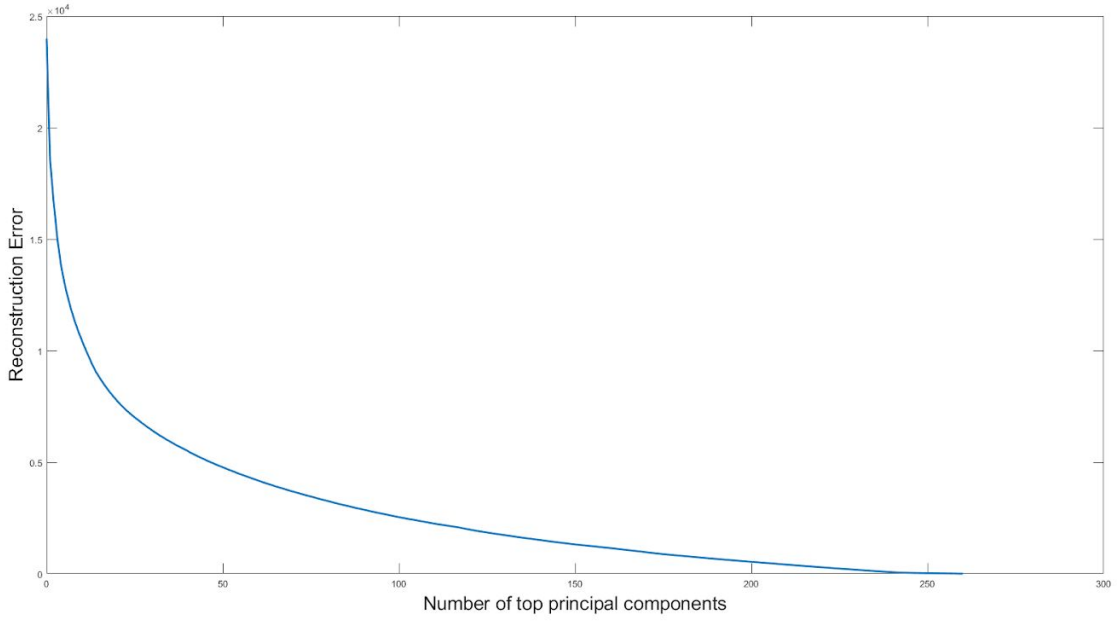


We can see that the error image resembles the original image more than the reconstructed image. This is because there's a high chance of missing out the top 10% eigenvectors in our randomized analysis.

A comparative graph of the reconstruction error v/s number of principal components was then graphed by plotting error for each N ∈ (0,1,2,3…...260) , here 'N' stands for the number of top eigenvectors as our principal components. The error was measured by the following formula:

$$Error = ||Y - X||_F$$

The plot is as follows:



As it is obvious and we can observe that for increasing N, our reconstruction error decreases. However it is crucial to note that at N = 260, the error vanishes to 0.

Inference:

We infer that by increasing the number of top eigenvectors used in our analysis, the visual quality of the reconstructed image increases. We also observed that by taking random eigenvectors, we get a distorted image.

Question 6:

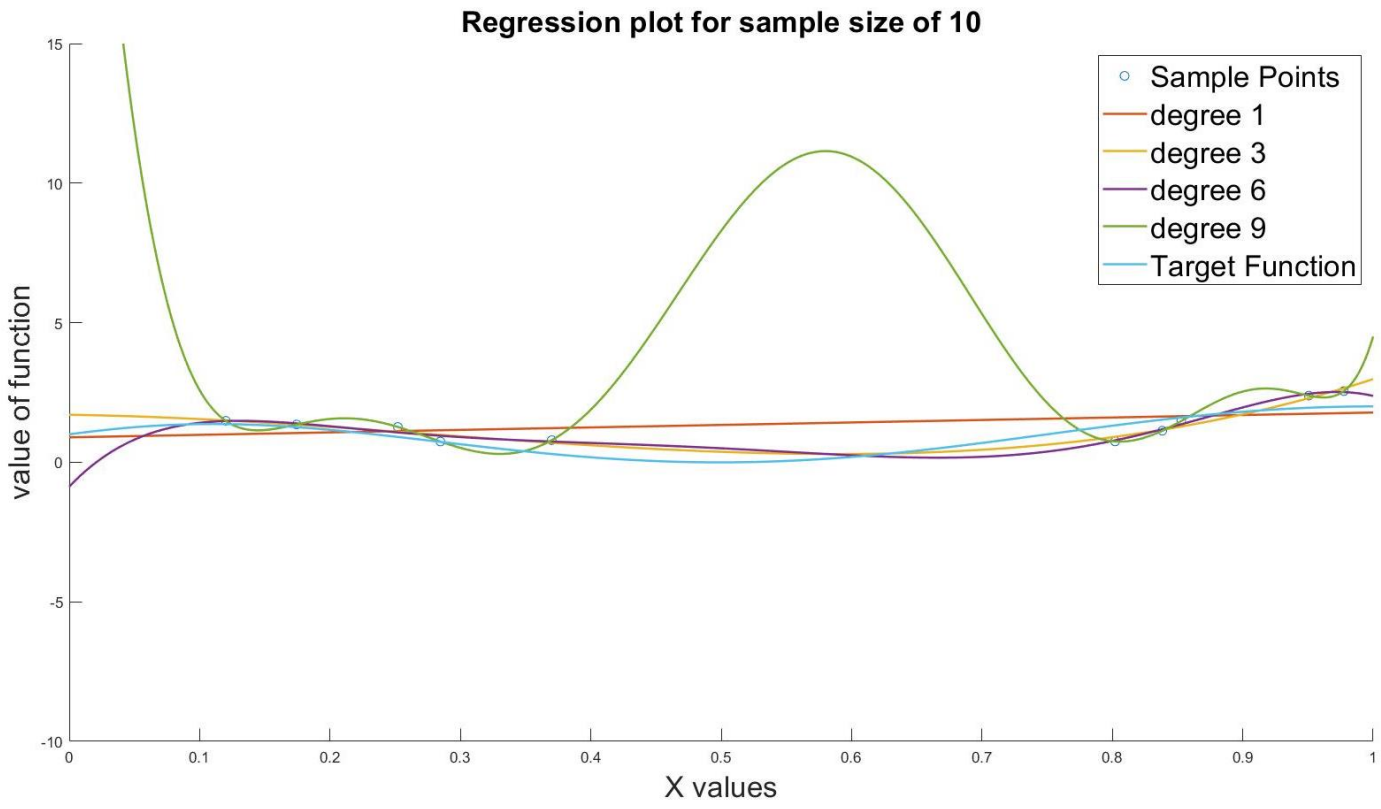
The points in X were generated using a uniform distribution [0,1]. The function assigned to us was y=cos(2πx)+tanh(2πx). The Y vector was generated by applying the function on X and adding a gaussian noise of mean 0 and variance 0.2. The 100 samples were divided into train and test as 80:20.

(a) Since the points in X are generated randomly, we have used the first ten points for fitting the given polynomials. Following is the table of coefficients.

Table 1

Coefficients	Degree 1	Degree 3	Degree 6	Degree 9
W_0	0.8883887	11.80627	-586.39941	131393.17
W_1	0.8923693	-9.8492959	1797.1741	-561605.78
W_2		-0.6800043	-2124.814	969991.31
W_3		1.7005273	1235.5599	-855962.59
W_4			-367.46168	393709.49
W_5			49.189601	-76115.366
W_6			-0.872499	-5647.7521
W_7				4954.1434
W_8				-750.27315
W_9				38.155061

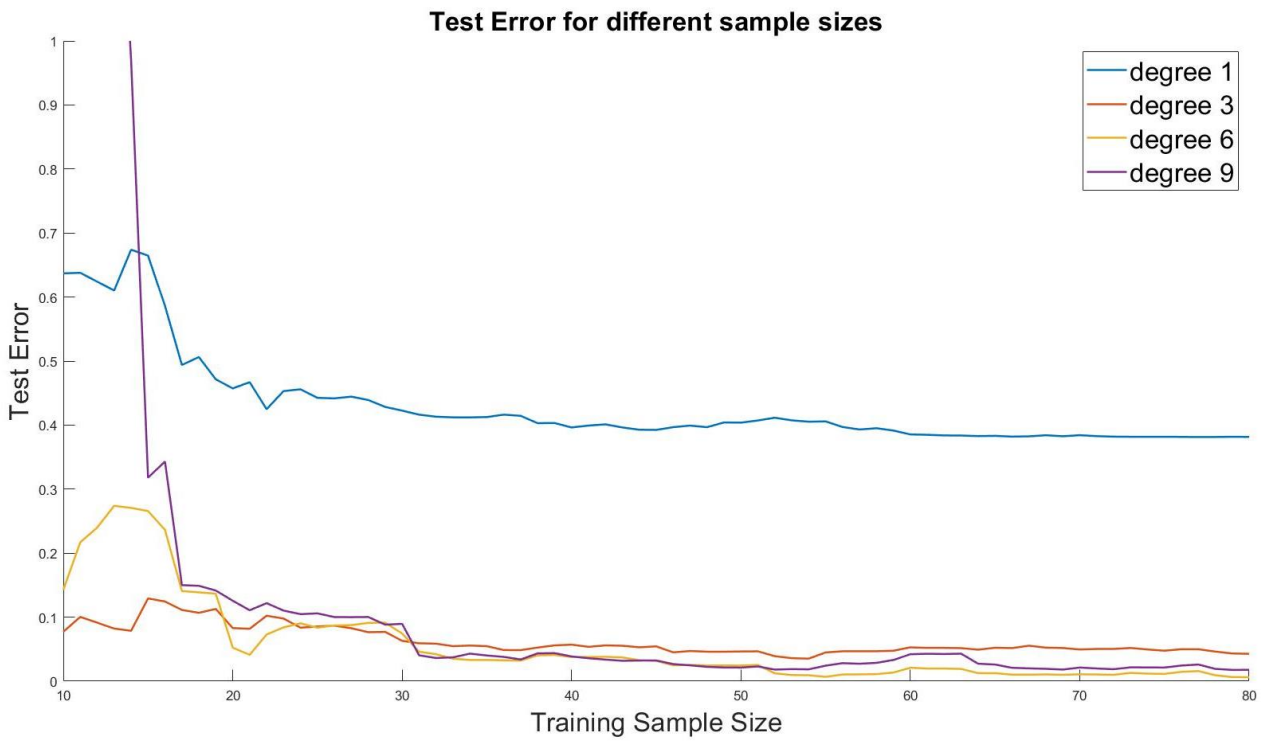
Figure 1



(b) All the four models were trained on increasing data set sizes and then the test errors were calculated. As can be seen from figure 2, when the sample size is small, degree 9 polynomial has a very high error which decreases drastically as the sample size increases. This shows that degree 9 polynomial was highly overfit

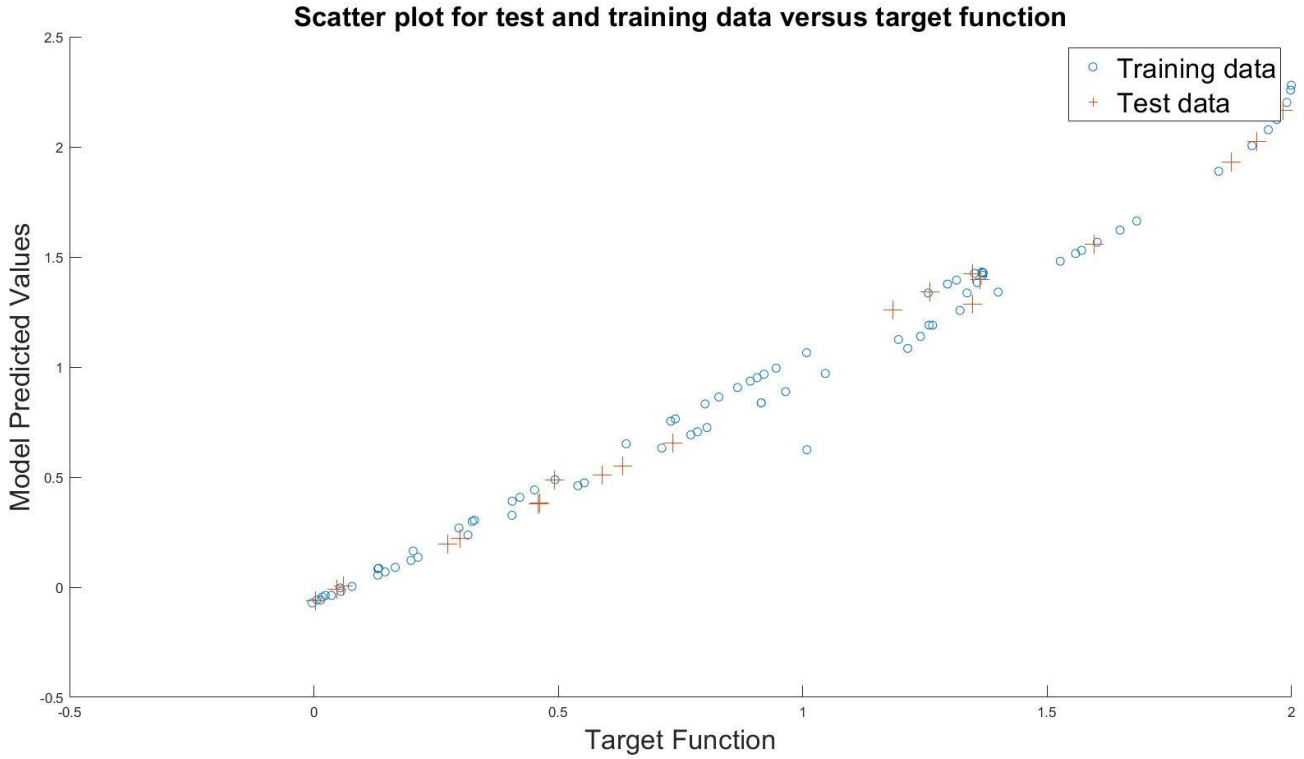
when the sample size was 10. Degree 1 polynomial is underfit so even on increasing the sample size the test error does not decrease. Degree 3 and 6 polynomials has very less test error but initially degree 6 was overfit so there is a bump in the graph.

Figure 2



(c) As can be seen the test error is minimum for degree 6 polynomial when the entire training set is used for curve fitting. Hence this is the best model. Figure 3 is the scatter plot for the target function versus training and test data. As can be seen from the figure, the points roughly lie on the 45^0 line.

Figure 3



(d) Figure 4 shows the plot of RMS error for training and test dataset for various degrees of polynomials