भारतीय प्रौद्योगिकी संस्थान धारवाड
Indian Institute of Technology Dharwad

Statistical Pattern Recognition

Assignment 4: Linear Models

Sawan Singh Mahara

M.Sc Research Scholar
191081003

April 17, 2020

Contents

Chapter 1

Model Overview

1 Pocket Perceptron

1.1 Brief

For the pocket perceptron model, the output y is given by linearly combining the d features of a data sample x

$$y = \sum_{i=1}^{d} w_i x_i + w_0 \tag{1.1}$$

If $y \leq 0$ then the prediction is class 0, else class 1.

1.2 Loss Function

When viewed as an optimisation problem, we try and minimise the following cost

$$J(W) = -\sum_{j:W^\top X_j \leq 0} W^\top X_j \tag{1.2}$$

Here, $W \in \mathcal{R}^{d+1}$ and $X_j \in \mathcal{R}^{d+1}$ where j denotes that sample point which is misclassified. The cost is just the summation of all those points misclassified, dot product product with the classifier W

1.3 Gradient of the Loss Function

The gradient of $J(W)$ with respect to W is found to be

$$\Delta J(W) = -\sum_{j:W^\top X_j \leq 0} X_j \tag{1.3}$$

1.4 Implementation

As we would like to reduce the number of misclassifications, our aim would be to minimise $J(W)$ over all W. The gradient descent method was used to do this. At the k'th iteration;

$$W^{k+1} = W_k + \eta \sum_{j:W^\top X_j \leq 0} X_j \tag{1.4}$$

Here, η is the learning rate, which was fixed at 0.1.

This was run multiple times and the W which gave the least misclassifications was taken as the final weight. This is termed as the pocket perceptron algorithm.

2 Linear Least Squares

2.1 Brief

Similar to the perceptron model, the output y is given by the same equation

$$y = \sum_{i=1}^{d} w_i x_i + w_0 \tag{1.5}$$

If $y \leq 0$ then the prediction is class 0, else class 1.

2.2 Loss Function

The difference between the two arises in the loss function considered. Given a dataset $\mathcal{D} = \{(x_i, y_i) : i \in \{0, .., N-1\}\}$ the loss is given by

$$J(W) = \sum_{j=0}^{N-1} (y_j - W^\top x_j)^2 \tag{1.6}$$

Here again, $W \in \mathcal{R}^{d+1}$ and $X_j \in \mathcal{R}^{d+1}$.

2.3 Implementation

A closed form solution W^* exists that minimises $J(W)$

$$W^* = (X^\top X)^{-1} X^\top Y \tag{1.7}$$

This was used to obtain the weights of the classifer.

3 Logistic Regression

3.1 Brief

The logistic regression model is obtained by computing the posterior probability of an assumed Gaussian class conditional density for a feature vector X. That results in the following form for the classifier.

$$y_i^{pred} = \sigma(W_i X_i) \tag{1.8}$$

Where $\sigma(z) = \frac{1}{1+e^{-z}}$
Again, if $y_i^{pred} > 0$, class 0 is picked, else class 1.

3.2 Loss Function

The logistic loss function for a dataset $\mathcal{D} = \{(x_i, y_i) : i \in \{0, .., N-1\}\}$ with $y \in \{0, 1\}$ is given by

$$J(W) = \frac{1}{N} \sum_{i=0}^{N-1} y_i \log(\sigma(W^\top x_i)) + (1 - y_i) \log(1 - \sigma(W^\top x^i)) \tag{1.9}$$

Here again, $W \in \mathcal{R}^{d+1}$ and $x_i \in \mathcal{R}^{d+1}$.

3.3 Implementation

The gradient of $J(W)$ can be shown to be

$$\Delta J(W) = \sum_{i=0}^{N-1} (y^i - \sigma(W^\top x^{(i)}))x^{(i)} \tag{1.10}$$

Using this gradient, the gradient descent algorithm was used to find the optimal W^*

$$W^* = (X^\top X)^{-1} X^\top Y \tag{1.11}$$

This was used to obtain the weights of the classifer.

4 Fischer's Linear Discriminant Analysis

4.1 Brief

The Fischer's Linear Discriminant Analysis method aims to find the direction that maximises the inter class differences, captured by the **between class scatter matrix $\mathbf{S}_B$** and **within class scatter matrix $\mathbf{S}_W$**

4.2 Optimisation Objective

Unlike in the earlier cases, the objective function is not a cost, but a measure of the differences between classes. We aim to maximise this function $J(W)$, given by

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{1.12}$$

Where

$$S_B = \sum_c (\boldsymbol{\mu}_c - \overline{\mathbf{x}}) (\boldsymbol{\mu}_c - \overline{\mathbf{x}})^T$$

$$S_W = \sum_c \sum_{i \in c} (\mathbf{x}_i - \boldsymbol{\mu}_c) (\mathbf{x}_i - \boldsymbol{\mu}_c)^T$$

c is the set of all classes, each of μ_c is that particular class's estimate of the mean and $\overline{\mathbf{x}}$ is the mean of the class means.

4.3 Implementation

The optimal W^* that maximises equation 1.12 is given by

$$W^* = \mathbf{S}_W^{-1}(\mu_1 - \mu_2) \tag{1.13}$$

for the two class case, which was used in the implementation.

Chapter 2

Classification Task Results

1 Synthetic Dataset

1.1 Variation in Priors

The means for class 0 and class 1 were a 10 dimensional vector of zeros and ones, respectively. The covariance matrices were taken as the identity matrix in both cases.

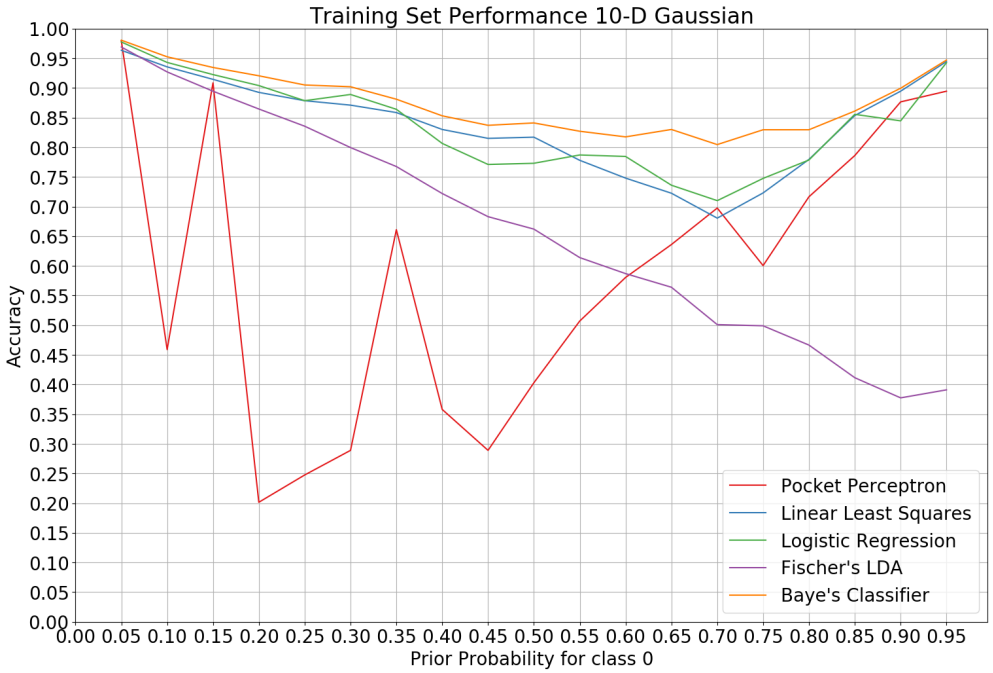


Figure 2.1: The performance of all the classifiers with varying prior probabilities (10-D data)

The same experiment was performed on 2D Gaussians as well.

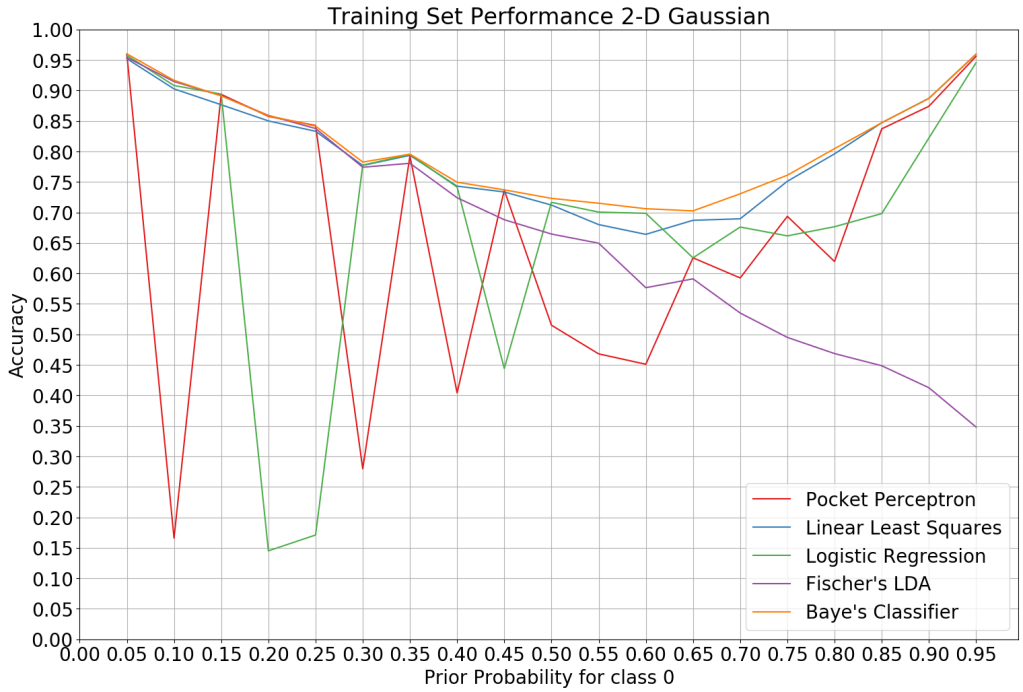
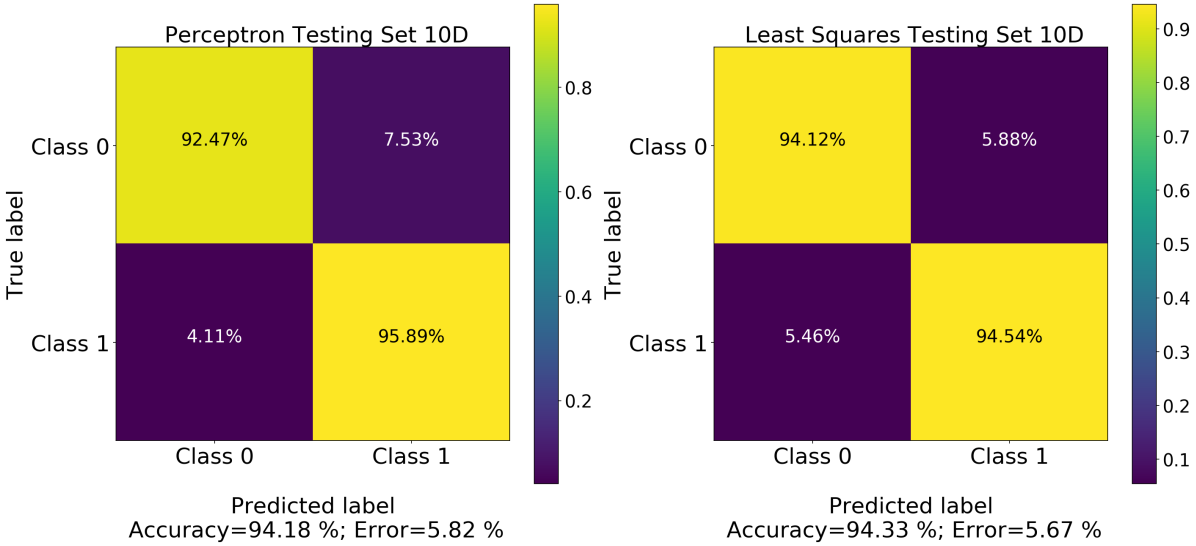
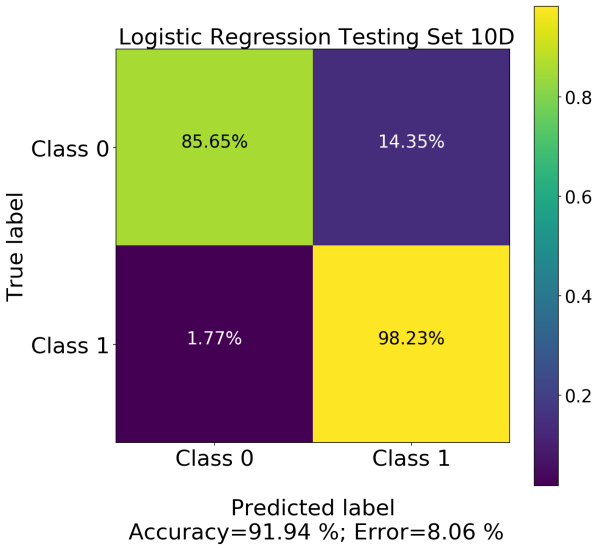
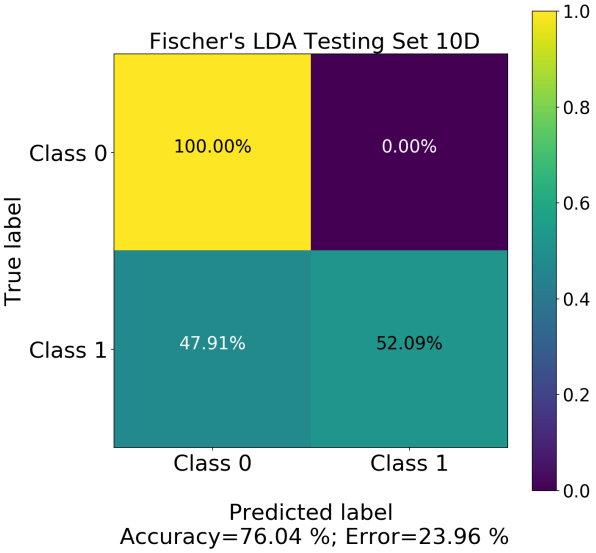


Figure 2.2: The performance of all the classifiers with varying prior probabilities (2-D data)

Confusion Matrices for 10D Case at Equal Priors







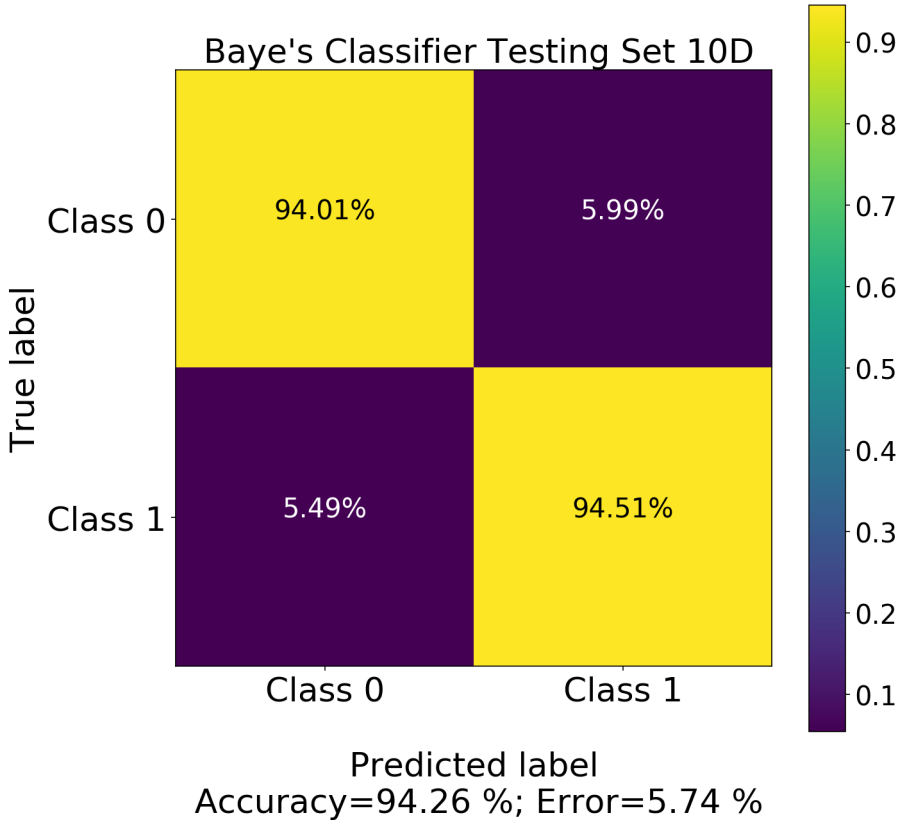


Table of Weights Learned

	Perceptron	Linear Least Squares	Logistic Regression	Fischer's LDA
10 D Weights	1706, 332.055, 331.963, 295.334, 340.97, 326.075, 354.109, 329.483, 286.443, 276.774, 291.386	-0.718344, 0.153846, 0.138732, 0.1342, 0.144079, 0.140264, 0.147045, 0.145608, 0.141364, 0.149058 ,0.143452	-1770.58, 287.996,, 256.583 229.797, 269.531, 267.513, 275.057, 270.162, 242.722, 282.258, 231.4	0.0010137, 0.000218987, 0.000197474, 0.000191023 , 0.000205085, 0.000199655, 0.000209307, 0.000207262, 0.00020122, 0.000212173 0.000204192

Figure 2.5: Weight values learned. All weights were initialised to the zero vector.

The Decision Boundary for 2D Case at Equal Priors

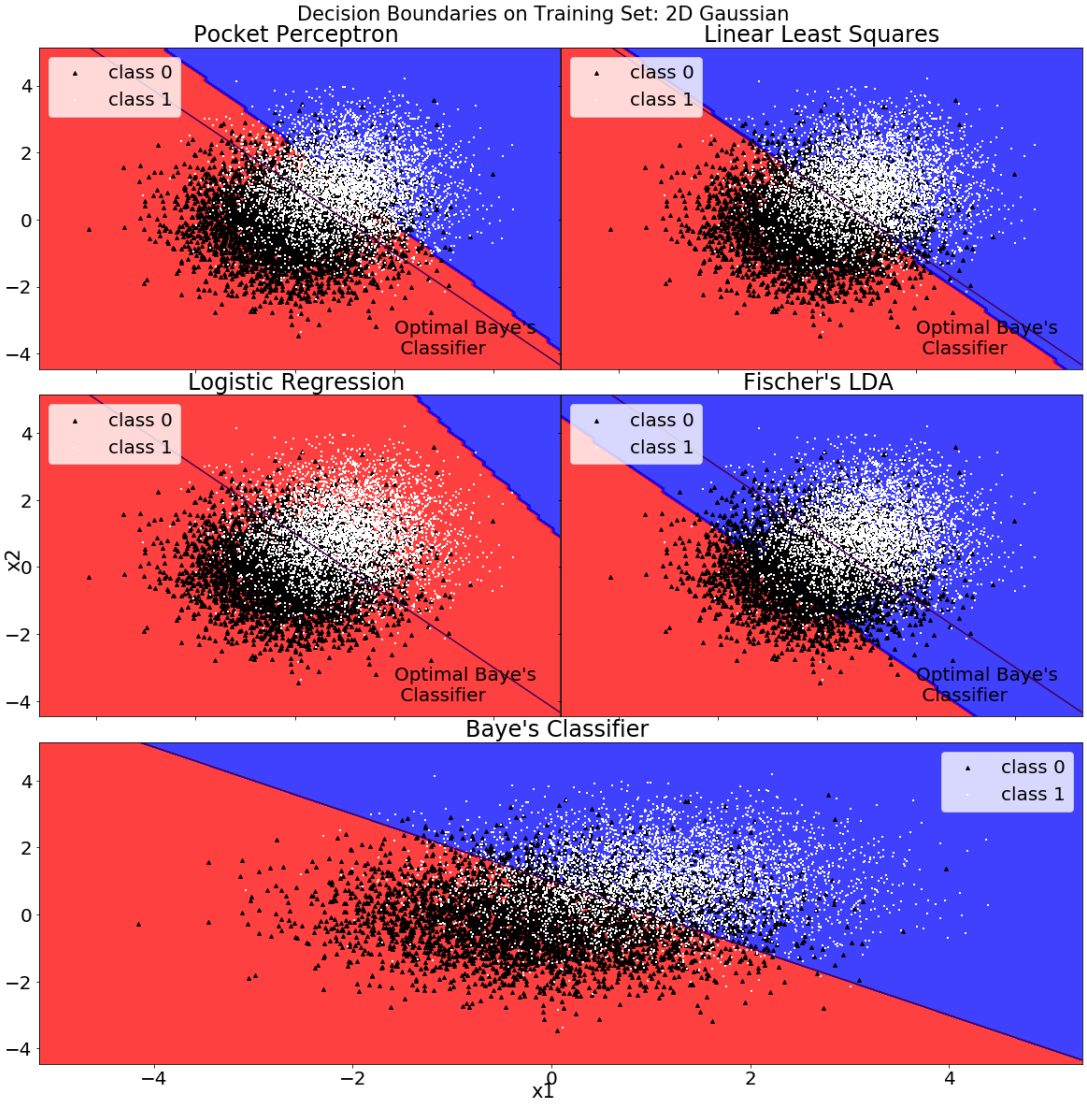
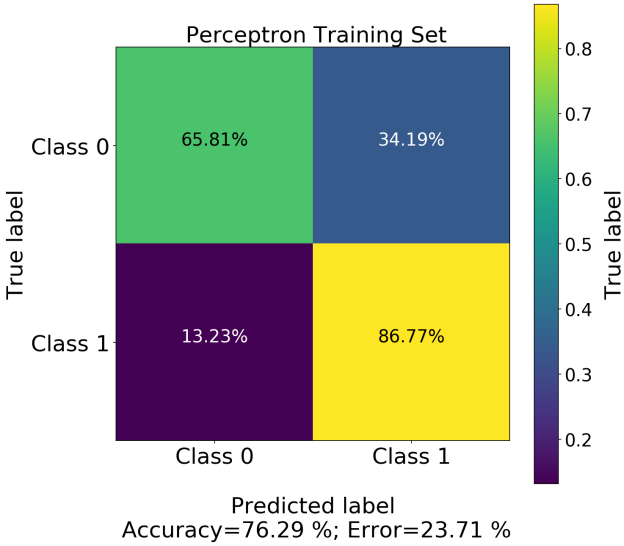
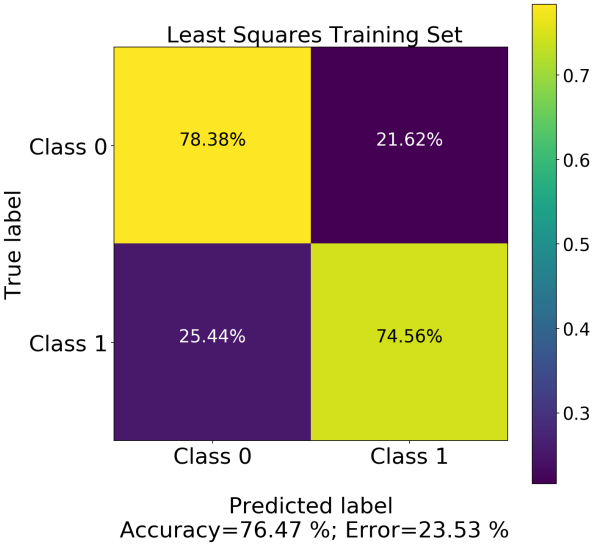
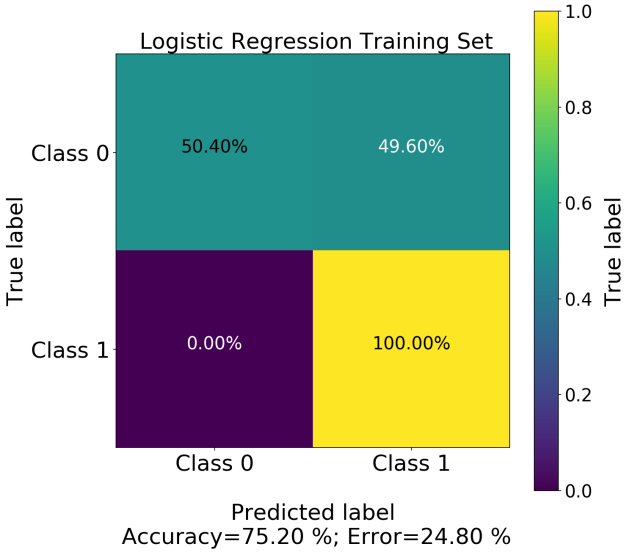


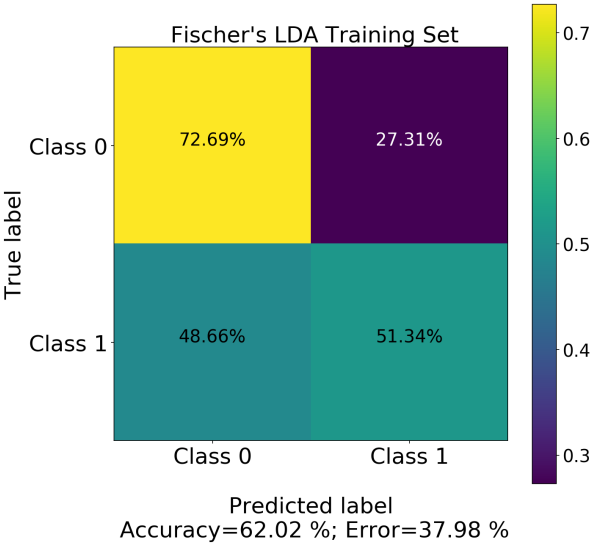
Figure 2.6: For mean 0 and 1 with identity covariance. The black line depicts the ideal baye's decision boundary, and the junction between blue and red shows the respective classifier's decision boundary

Confusion Matrices for 2D Case at Equal Priors









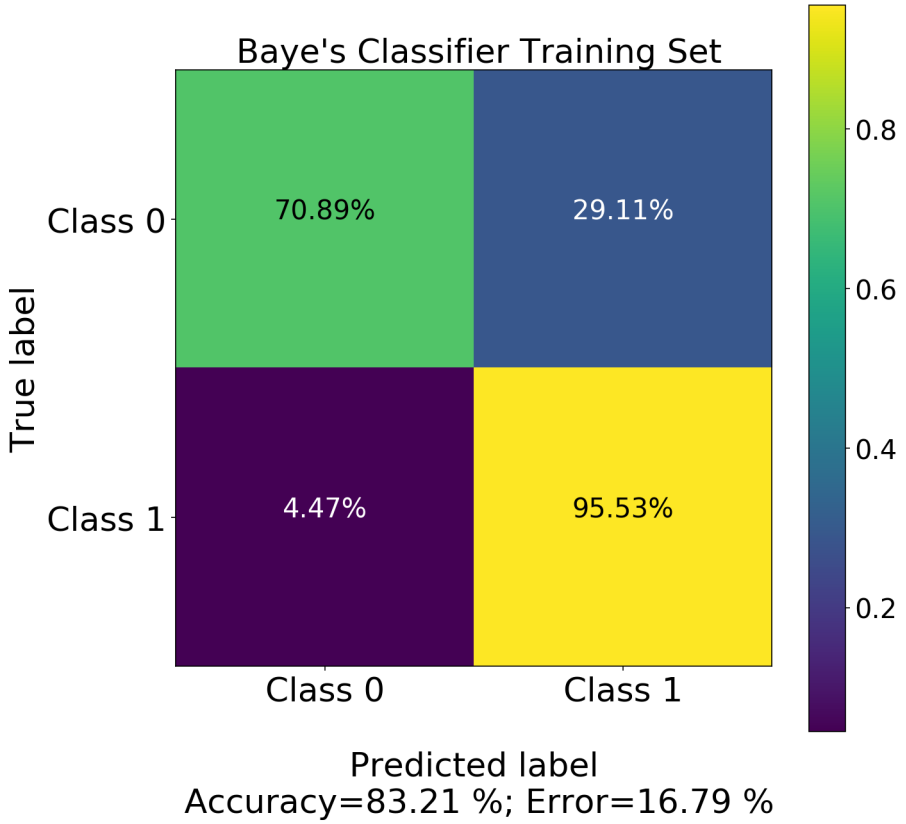


Table of Weights Learned

	Perceptron	Linear Least Squares	Logistic Regression	Fischer's LDA
2 D Weights	[-1145, 622.25, 679.5]	[-0.346461,0.332608,0.342542]	[1495.18,2467.39,2534.83]	[0.000207207,0.000201226,0.000207236]

Figure 2.9: Weight values learned. All weights were initialised to the zero vector.

1.2 2-D Dimensional Dataset Poor Performance and Remedy

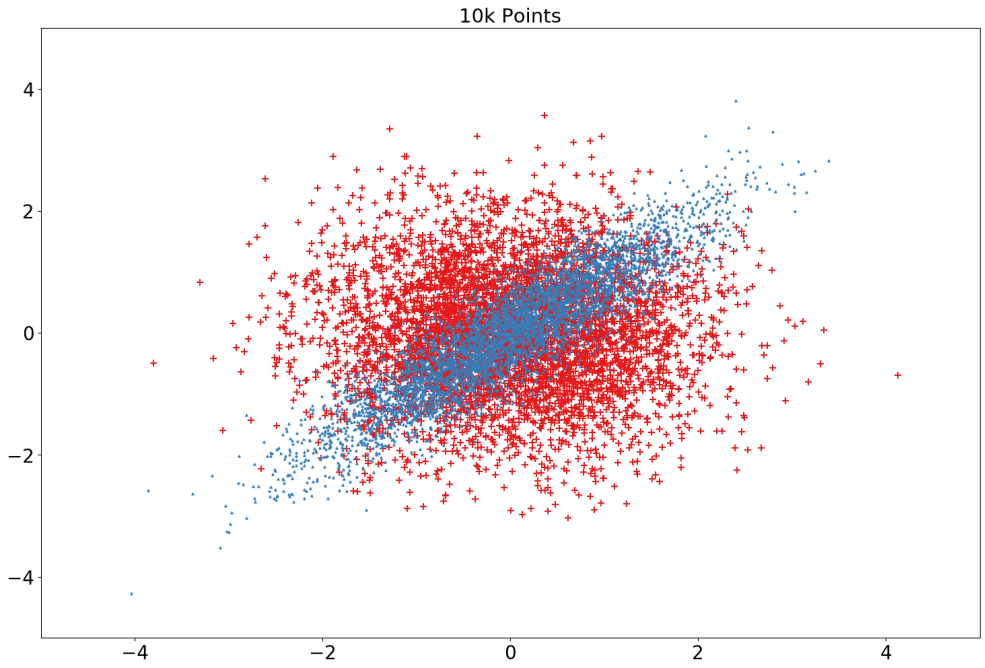


Figure 2.10: The dataset with 0 means and covariance matrices c_0 and c_1

Where,

$$c_0 = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix} c_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The data is not very well separated linearly and class 0 has high correlation between features. The class 1 data, on the other hand, has independent features.
Thus, the performance of the linear classifiers on just the two present features x_1 and sx_2 alone, was poor.
The features were transformed from a 2-D space into a 6-D space, utilising the quadratic transform.

$$G^{old}(X) = w_1 x_1 + w_2 x_2$$

$$G^{new}(X) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_2 x_1 + w_4 x_1^2 + w_5 x_2^2$$

The new objective was then to apply the linear classifiers to learn the weights $w_0 - w_5$

Performance Before Transformation

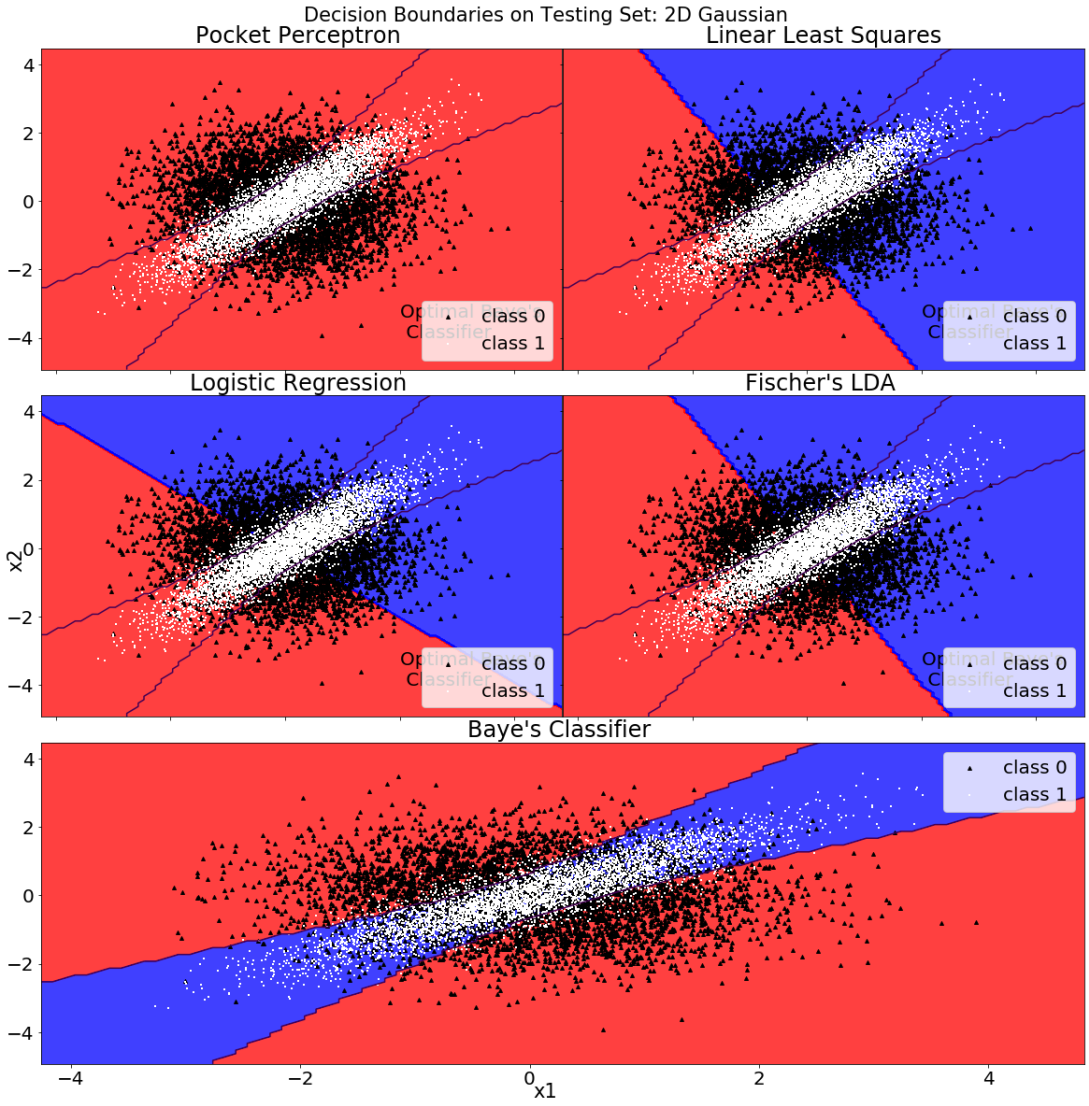
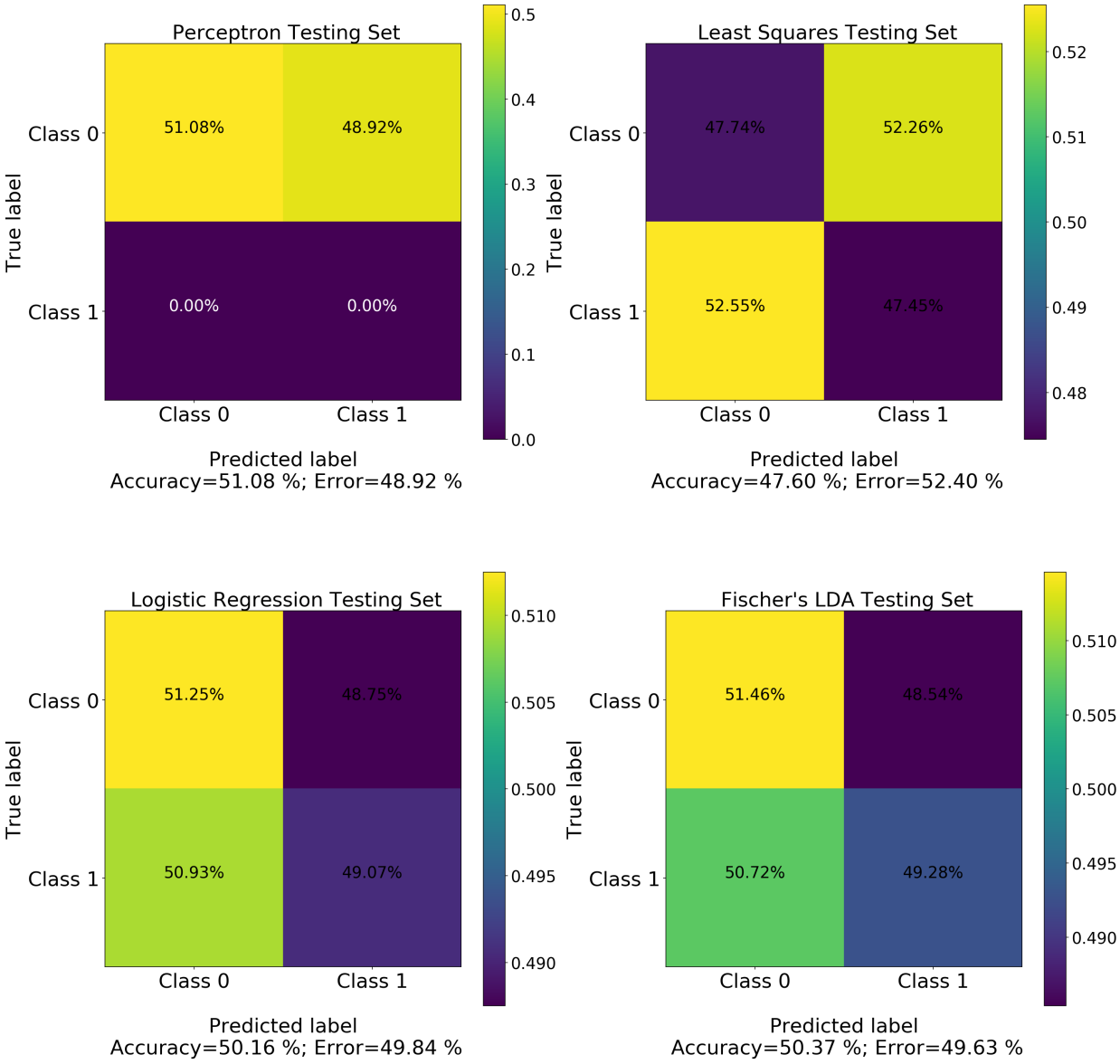


Figure 2.11: The following decision boundaries were obtained for the **non-transformed data**. The black lines in the top 4 figures represent what the optimal baye's classifier boundary would look like, in comparison

Confusion Matrices Before Transformation



The above confusion matrices show the performance of all linear classifiers, when the data was used without any transformation

Performance After Transformation

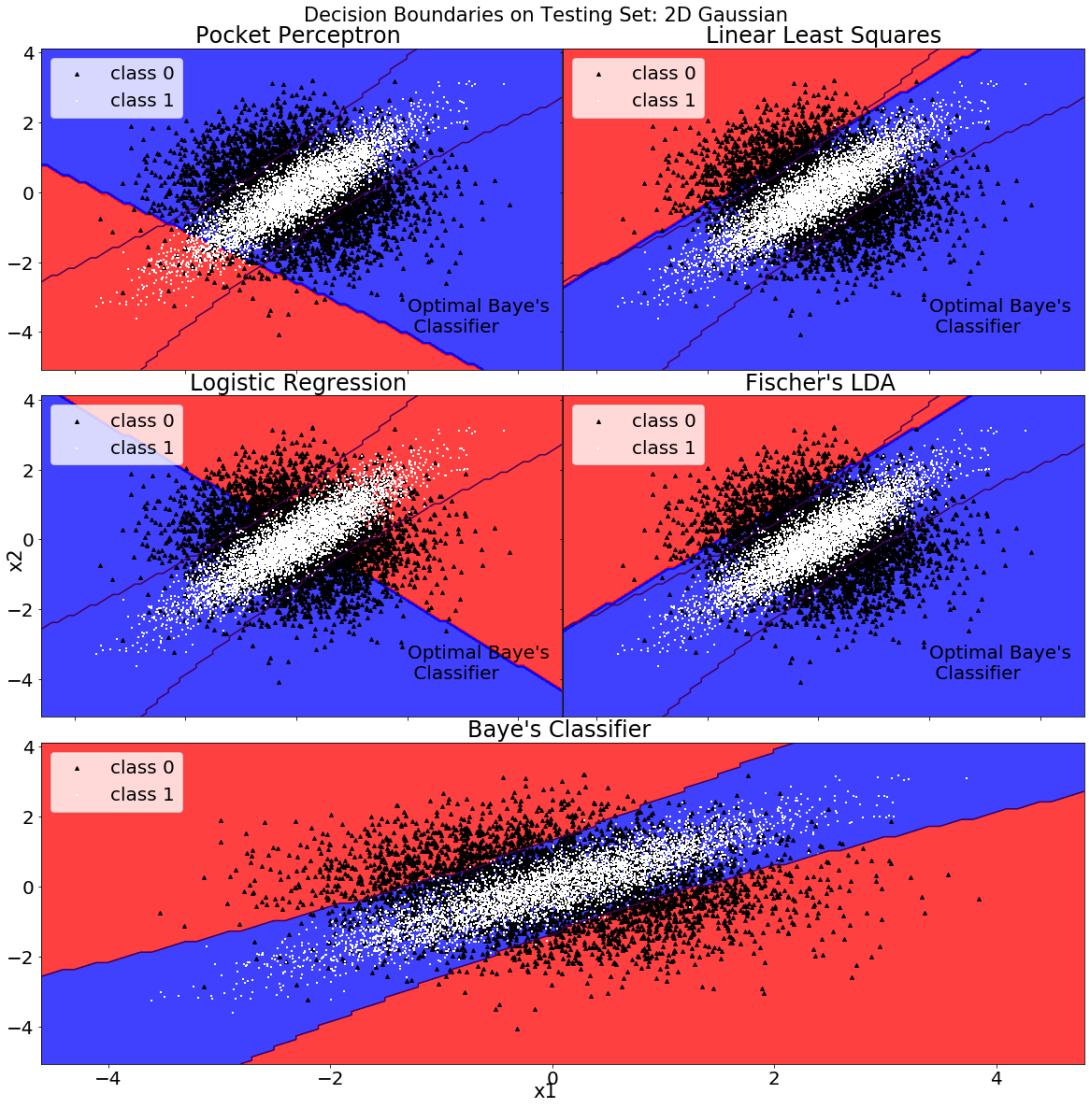
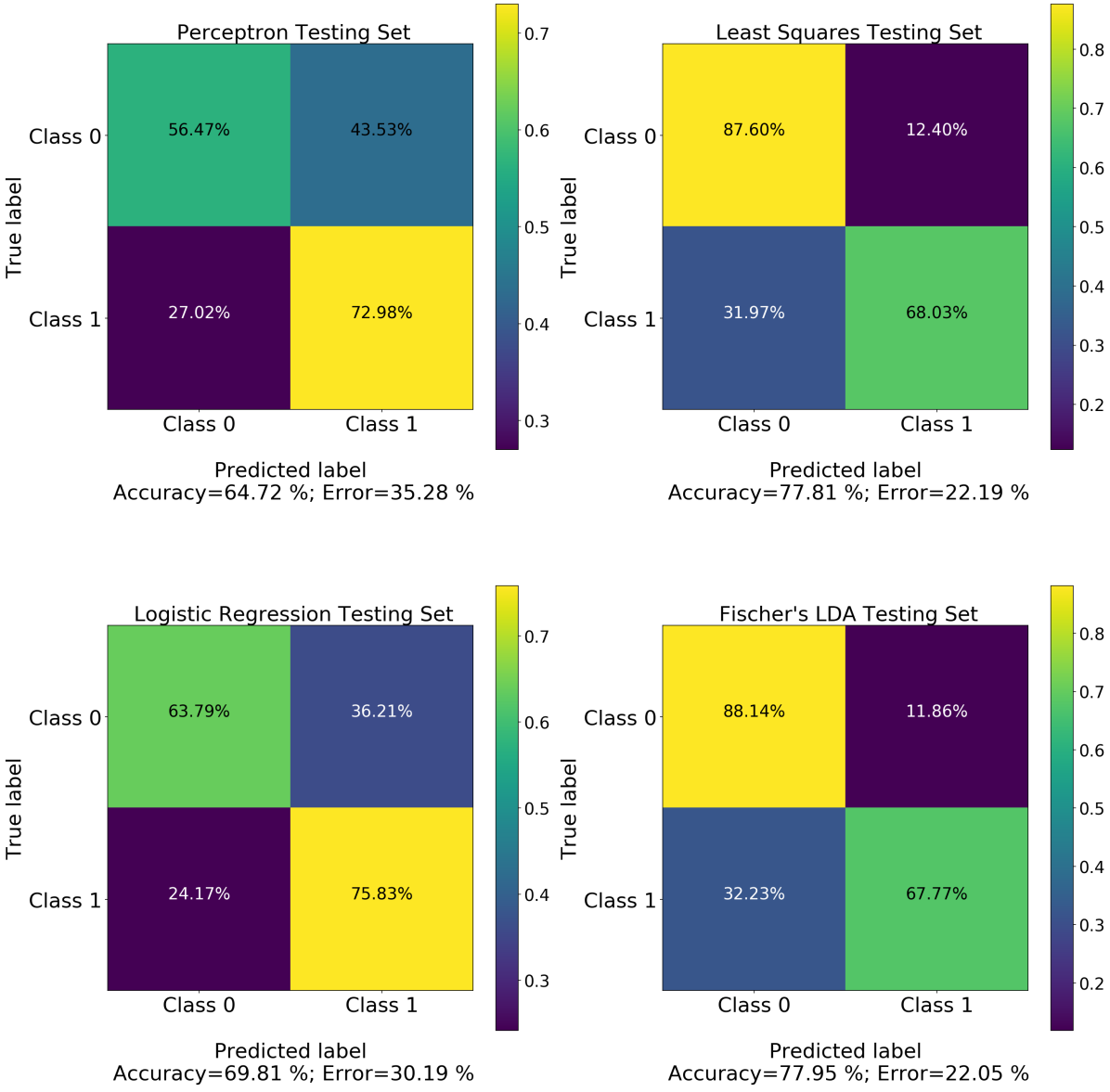


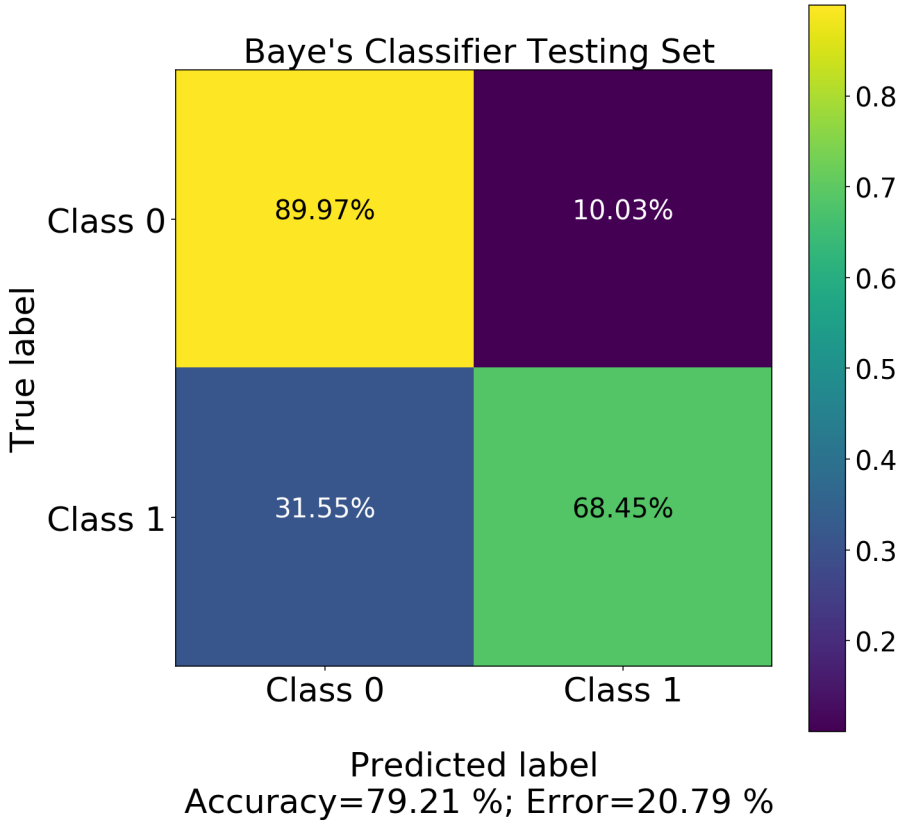
Figure 2.14: The following decision boundaries were obtained for the transformed data. The black lines in the top 4 figures represent what the optimal baye's classifier boundary would look like, in comparison

This boundary has no meaning in 2-D as the true boundary lies in 6-D space. This is just a projection of the 6D boundary onto the plane containing w_1 **and** w_2.

Confusion Matrices After Transformation

The confusion matrices show the performance of all linear classifiers, after applying the quadratic transform.





The Baye's discriminant function, being quadratic inherently, could be thought of as laying in 6-D, thus resulting in the following confusion matrix for the data, agnostic to the transformation.

Table of Weights Learned Before and After Transformation

	Perceptron	Linear Least Squares	Logistic Regression	Fischer's LDA
2 D Weights	[2025,-207.305,-0.959429]	[0.0130219,0.0278356,-0.0221569]	[-1040.7,415.494,-224.643]	[2.27599e-07,1.11442e-05, -8.87066e-06]

Figure 2.17: Weight values learned before transformation. All weights were initialised to the zero vector.

	Perceptron	Linear Least Squares	Logistic Regression	Fischer's LDA
6 D Weights	[-149,21.5679, -39.4491,2520.06,2450.75,-4653.9]	[-0.102186,0.00830815, -0.00725817,0.151352,0.134468, -0.445531]	[-103.731,53.6189,23.4931,2729.86,2632.64,-5236.27]	[4.66941e-05,4.14771e-06, -3.62353e-06,7.556e-05,6.71312e-05,-0.000222425]

Figure 2.18: Weight values learned after transformation. All weights were initialised to the zero vector.

Clear Clusters

Finally, the analysis was done on the following dataset;

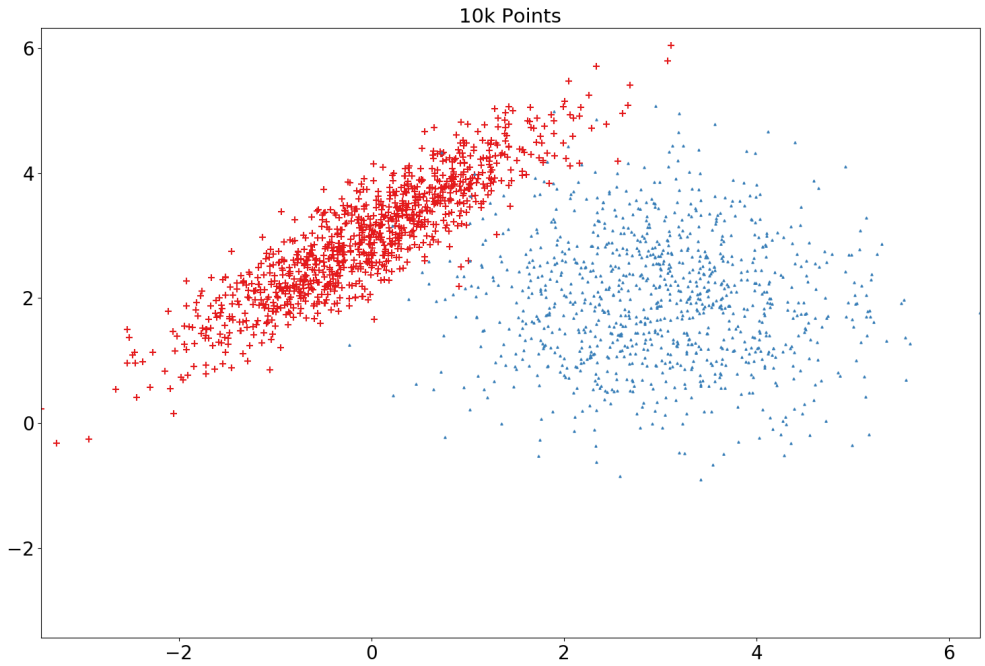


Figure 2.19: The dataset with means m_0 and m_1 and covariance matrices c_0 and c_1

Where,

$$m_0 = \begin{bmatrix} 0 \\ 3 \end{bmatrix} m_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \quad c_0 = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix} c_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The data is somewhat linearly separable and the decision boundaries obtained give fairly good results. The figures show the results of the decision boundaries on the training and testing set.

The Decision Boundaries

The decision boundaries plotted for the training dataset

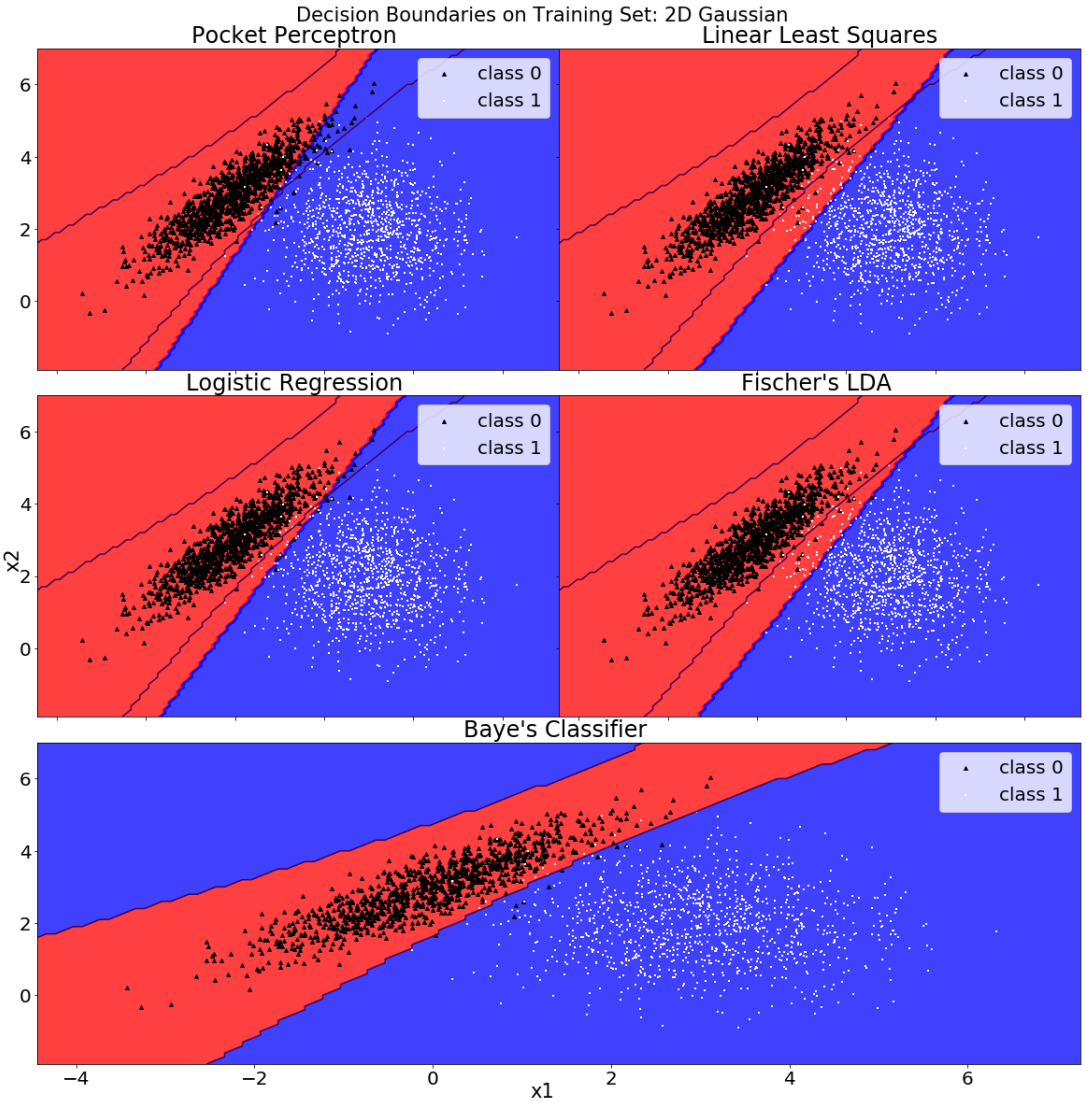


Figure 2.20: Decision Boundaries for the training set

The decision boundaries plotted for the testing dataset

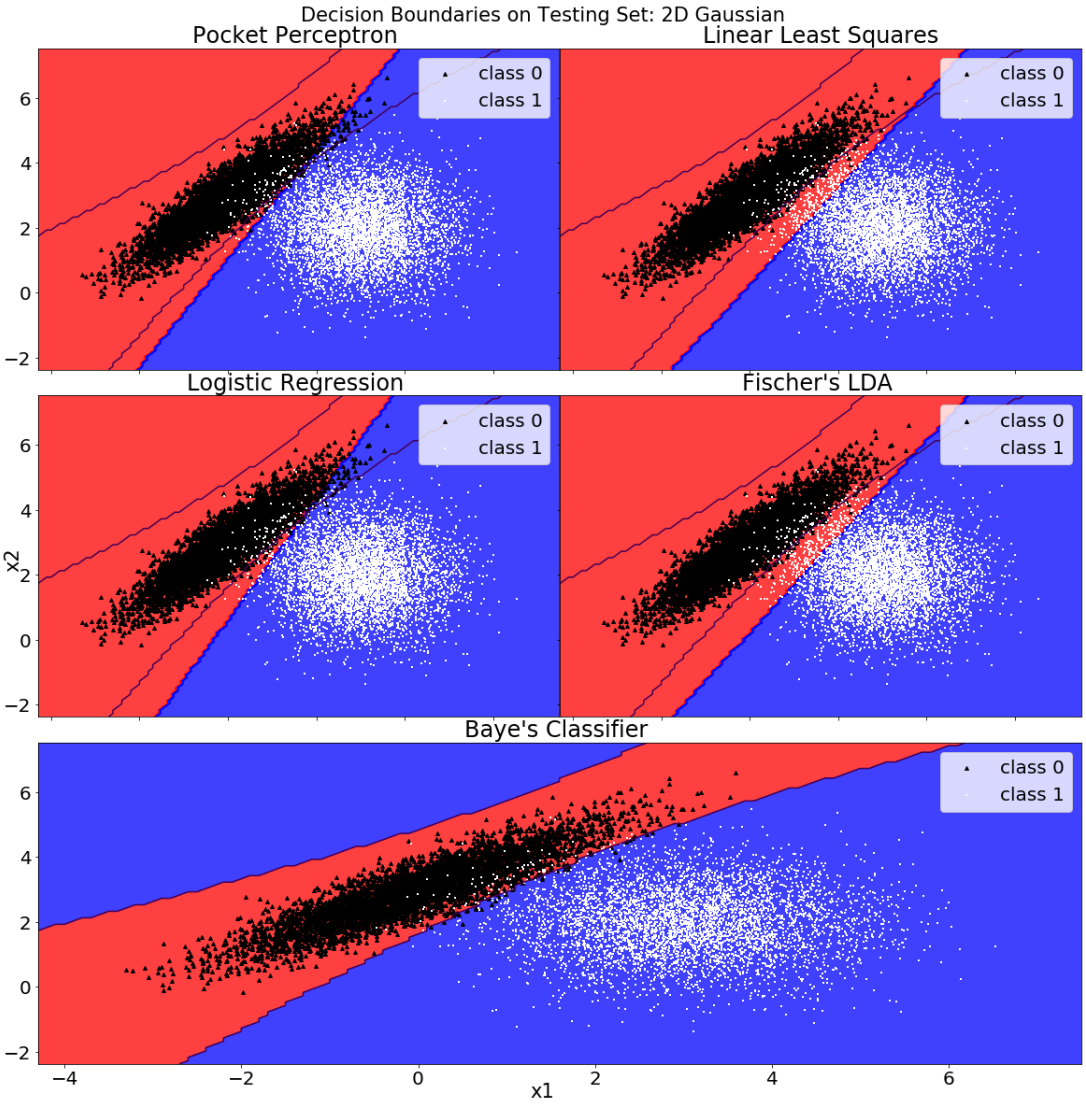
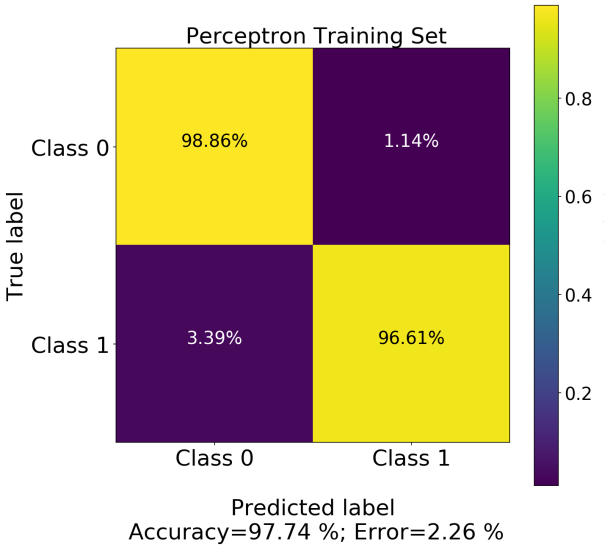
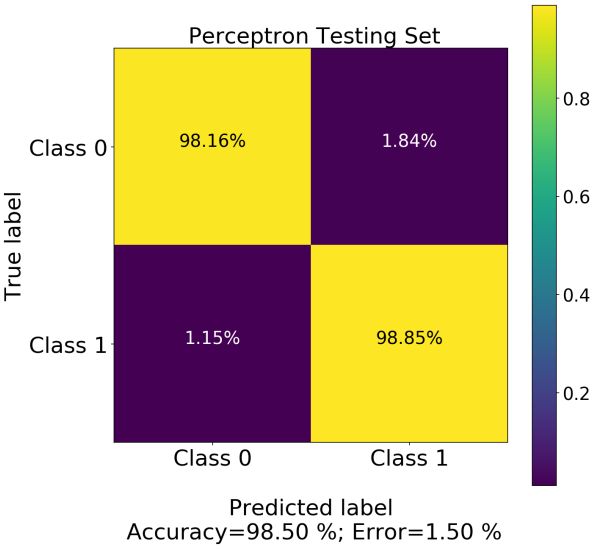
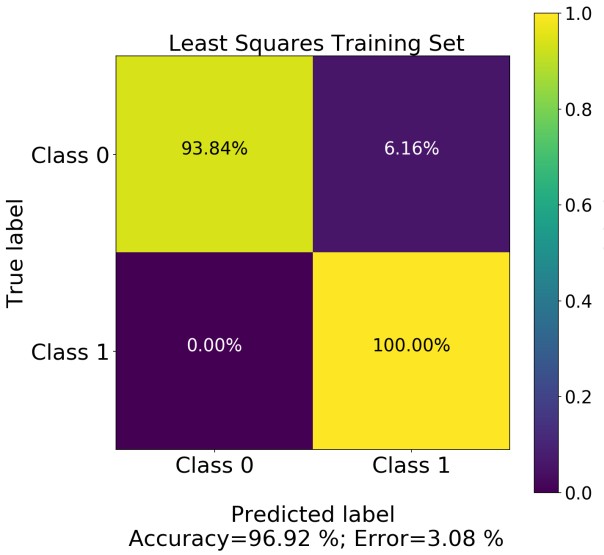


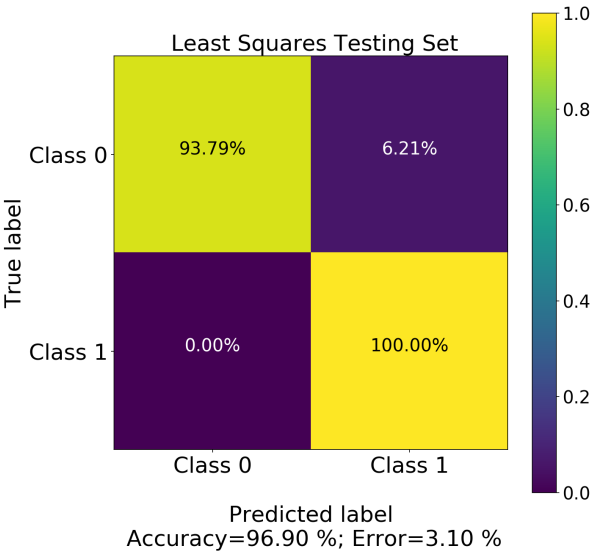
Figure 2.21: Decision Boundaries for the testing set

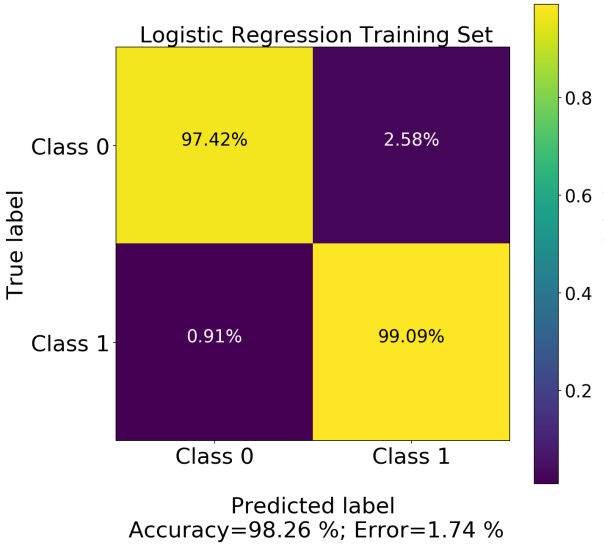
Confusion Matrices Results

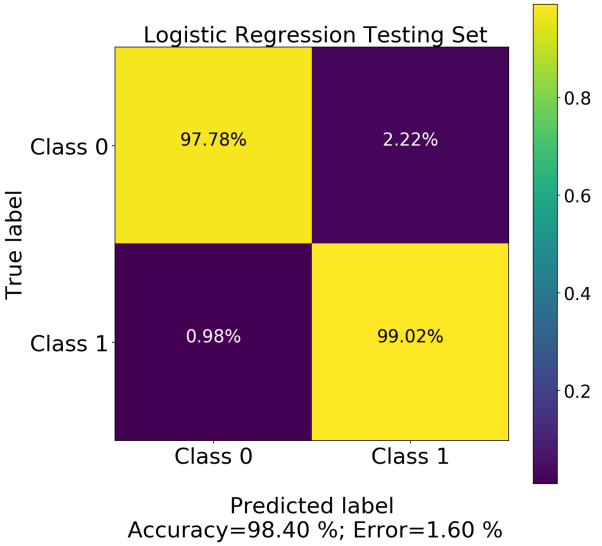


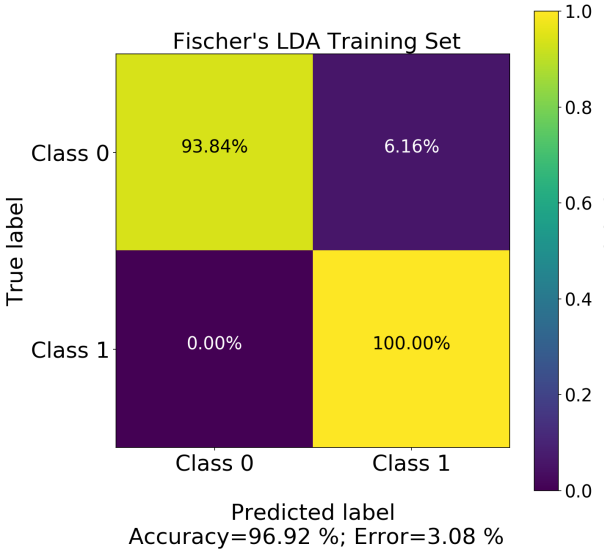


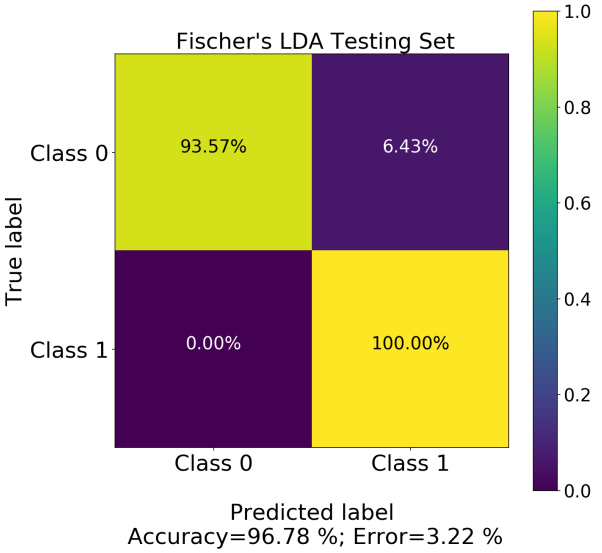





Logistic Regression Training Set
Class 0
Class 1
97.42%
2.58%
0.91%
99.09%
True label
Class 0
Class 1
Predicted label
Accuracy=98.26 %; Error=1.74 %
0.8
0.6
0.4
0.2


Logistic Regression Testing Set
Class 0
Class 1
97.78%
2.22%
0.98%
99.02%
True label
Class 0
Class 1
Predicted label
Accuracy=98.40 %; Error=1.60 %
0.8
0.6
0.4
0.2


Fischer's LDA Training Set
Class 0
Class 1
93.84%
6.16%
0.00%
100.00%
True label
Class 0
Class 1
Predicted label
Accuracy=96.92 %; Error=3.08 %
1.0
0.8
0.6
0.4
0.2
0.0


Fischer's LDA Testing Set
Class 0
Class 1
93.57%
6.43%
0.00%
100.00%
True label
Class 0
Class 1
Predicted label
Accuracy=96.78 %; Error=3.22 %
1.0
0.8
0.6
0.4
0.2
0.0

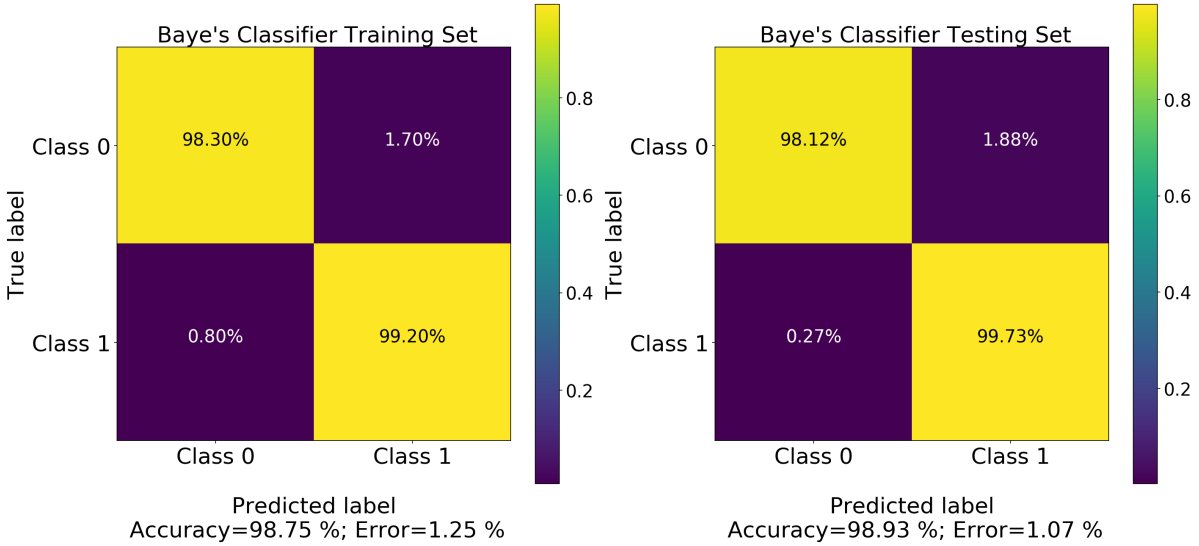


Table of Weights Learned Before Transformation

	Perceptron	Linear Least Squares	Logistic Regression	Fischer's LDA
3 D Weights	[1045,2032.93,-1118.21]	[0.0837468,0.438725,-0.295288]	[1435.9,2301.68,-1474.49]	[0.000172241,0.000880525, -0.000592647]

Figure 2.27: Weight values learned. All weights were initialised to the zero vector.

2 MNIST Handwriting

The multi-class classifier was run on the MNIST digit dataset on digits 5,7,8,9. The entire dataset is sampled in figure 2.28. The algorithms implemented were one vs one and one vs all. They were all hard-coded, except in the case for Fischer's LDA where the sci-kit learn library was used.

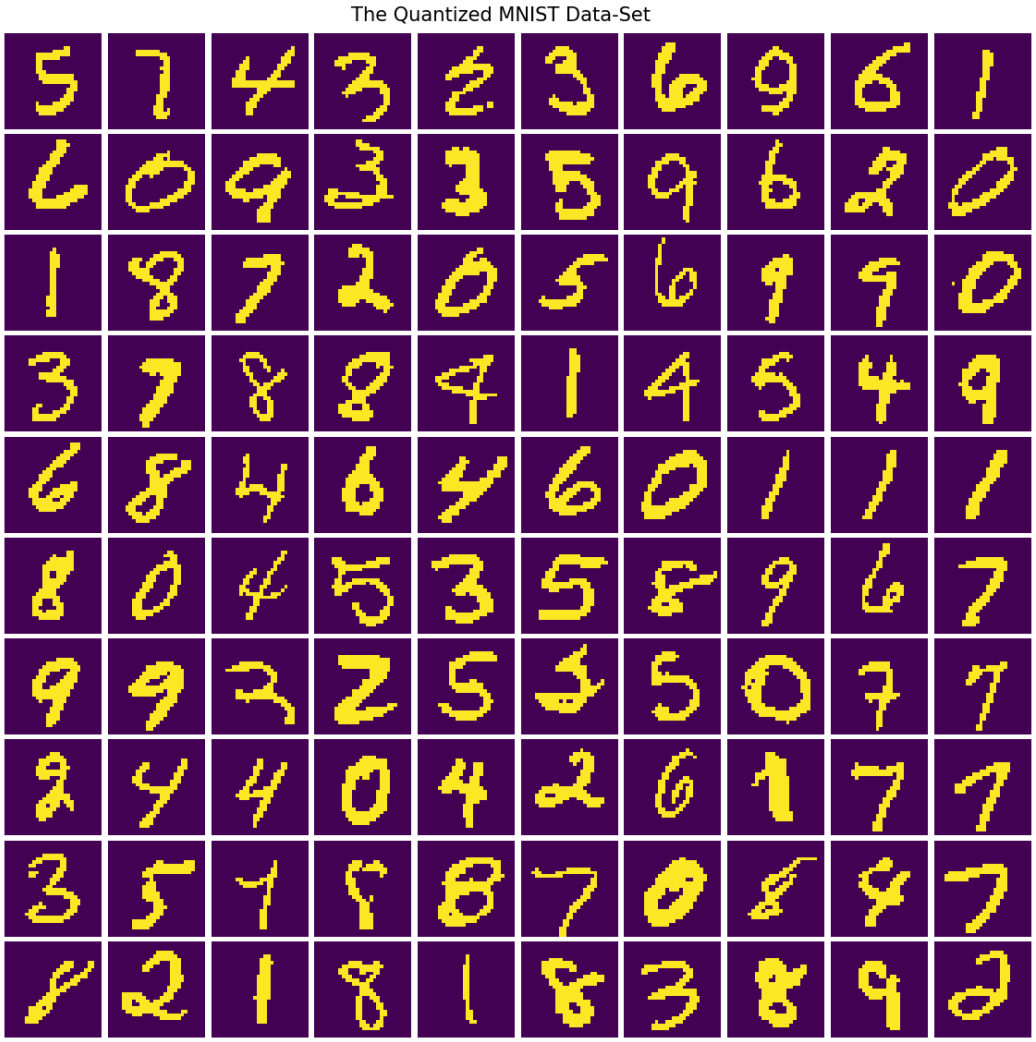
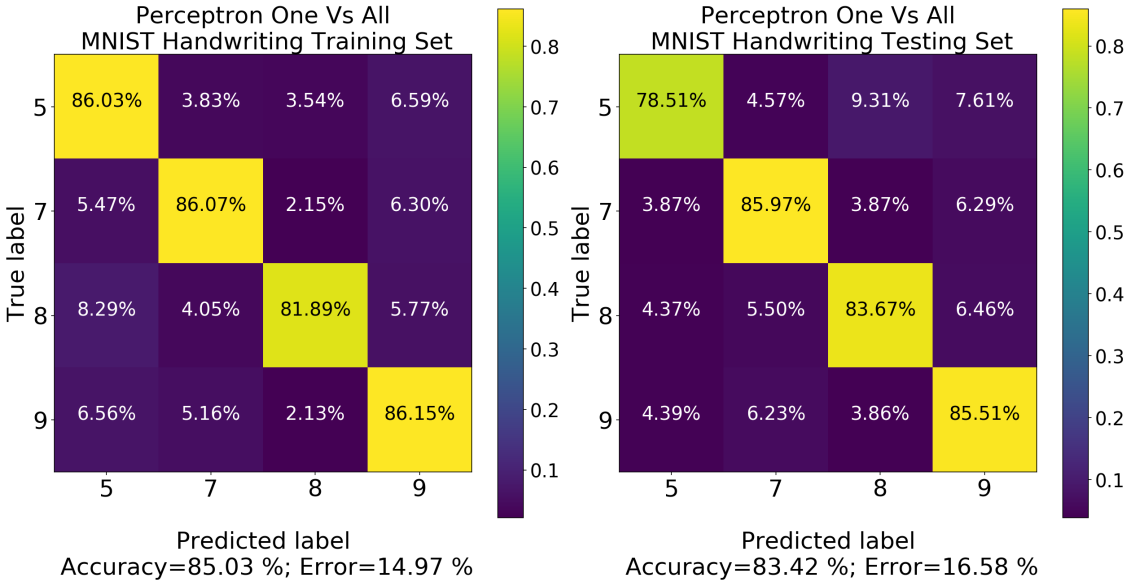
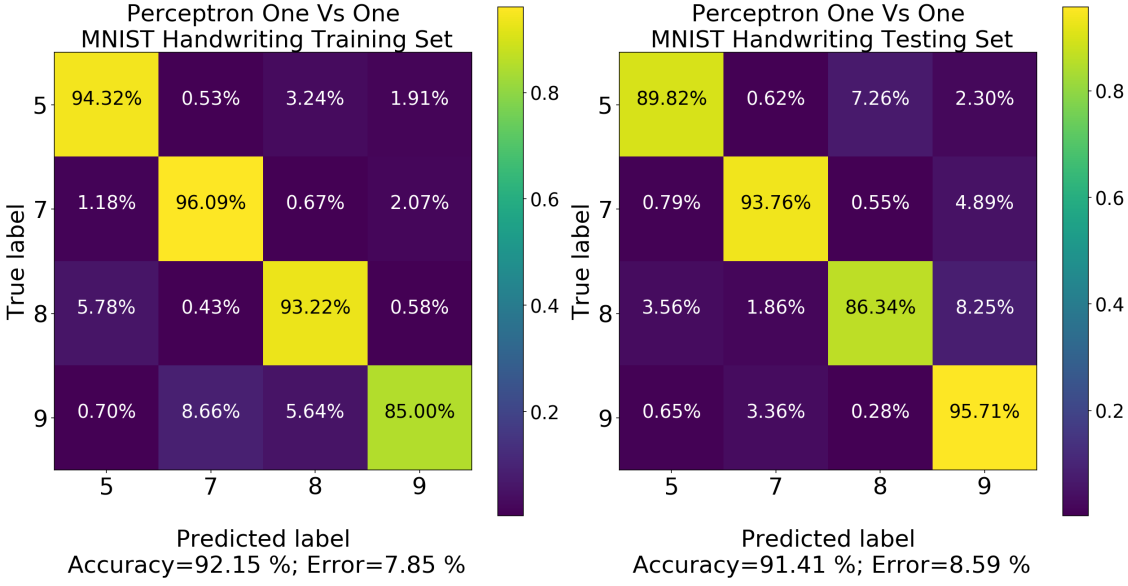
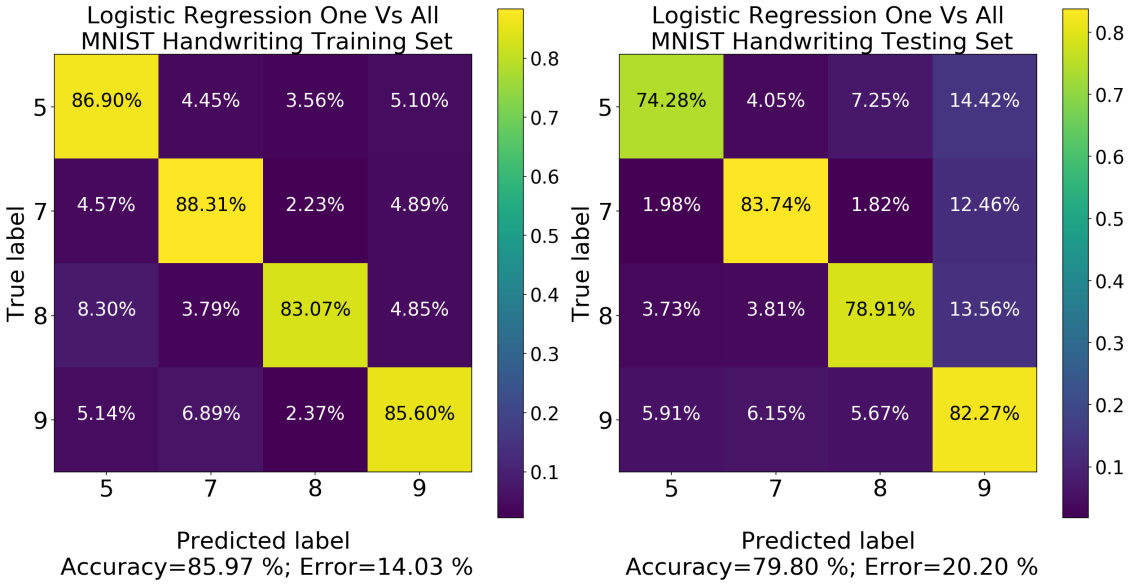


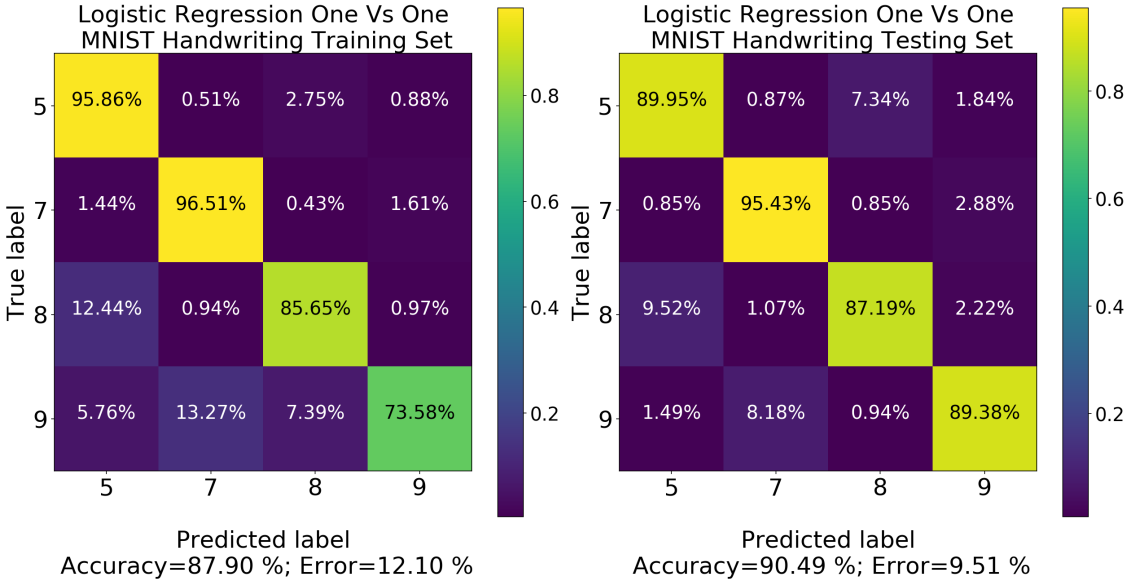
Figure 2.28: MNIST Dataset

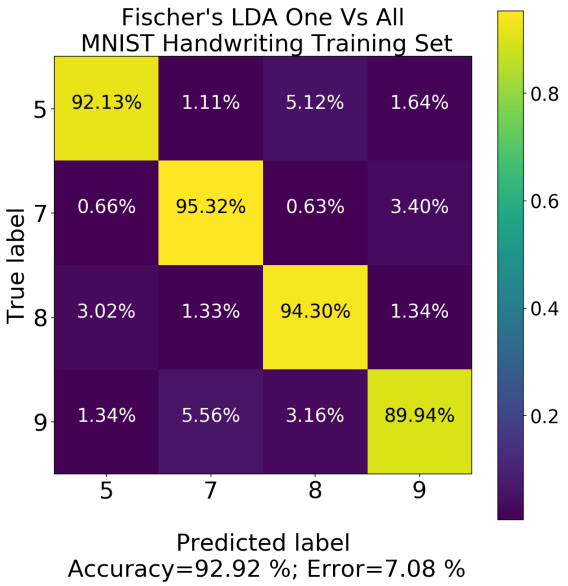
2.1 Confusion Matrices

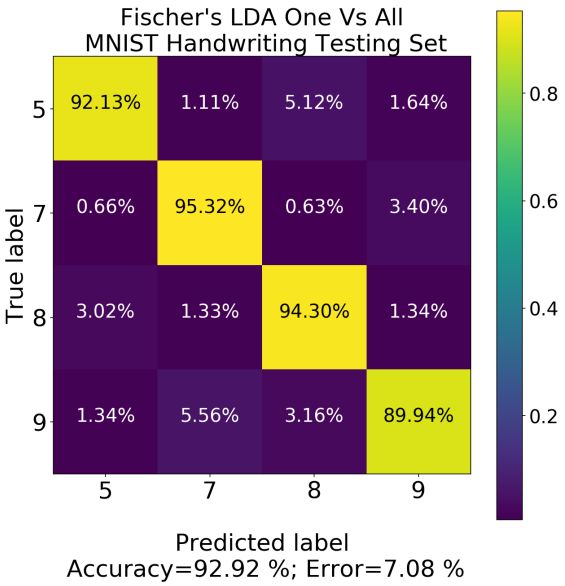


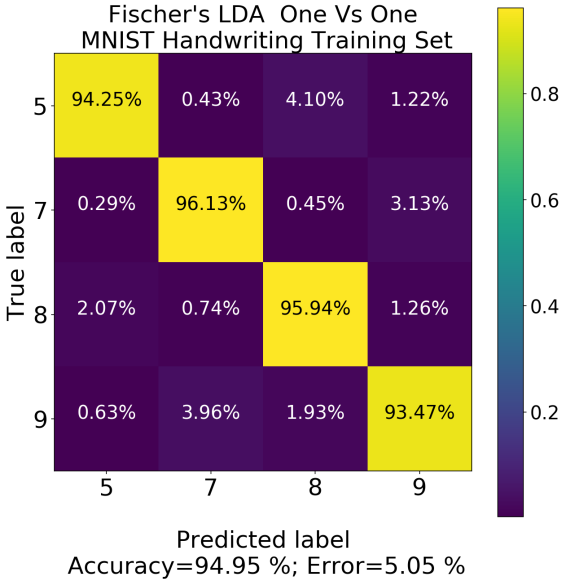


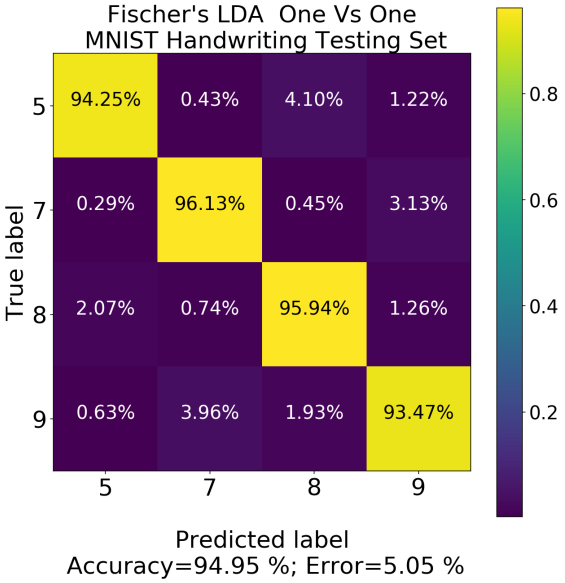

Logistic Regression One Vs All
MNIST Handwriting Training Set
True label
Predicted label
Accuracy=85.97 %; Error=14.03 %
Logistic Regression One Vs All
MNIST Handwriting Testing Set
True label
Predicted label
Accuracy=79.80 %; Error=20.20 %


Logistic Regression One Vs One
MNIST Handwriting Training Set
True label
Predicted label
Accuracy=87.90 %; Error=12.10 %
Logistic Regression One Vs One
MNIST Handwriting Testing Set
True label
Predicted label
Accuracy=90.49 %; Error=9.51 %


Fischer's LDA One Vs All
MNIST Handwriting Training Set
True label
Predicted label
Accuracy=92.92 %; Error=7.08 %
92.13%
1.11%
5.12%
1.64%
0.66%
95.32%
0.63%
3.40%
3.02%
1.33%
94.30%
1.34%
1.34%
5.56%
3.16%
89.94%
5
7
8
9
0.8
0.6
0.4
0.2


Fischer's LDA One Vs All
MNIST Handwriting Testing Set
True label
Predicted label
Accuracy=92.92 %; Error=7.08 %
92.13%
1.11%
5.12%
1.64%
0.66%
95.32%
0.63%
3.40%
3.02%
1.33%
94.30%
1.34%
1.34%
5.56%
3.16%
89.94%
5
7
8
9
0.8
0.6
0.4
0.2


Fischer's LDA One Vs One
MNIST Handwriting Training Set
True label
Predicted label
Accuracy=94.95 %; Error=5.05 %
94.25%
0.43%
4.10%
1.22%
0.29%
96.13%
0.45%
3.13%
2.07%
0.74%
95.94%
1.26%
0.63%
3.96%
1.93%
93.47%
5
7
8
9
0.8
0.6
0.4
0.2


Fischer's LDA One Vs One
MNIST Handwriting Testing Set
True label
Predicted label
Accuracy=94.95 %; Error=5.05 %
94.25%
0.43%
4.10%
1.22%
0.29%
96.13%
0.45%
3.13%
2.07%
0.74%
95.94%
1.26%
0.63%
3.96%
1.93%
93.47%
5
7
8
9
0.8
0.6
0.4
0.2

3 German Credit Dataset

3.1 Dataset Overview and Pre-Processing

Context

The original dataset contains 1000 entries with 20 categorial and symbolic attributes prepared by Prof. Hofmann. Here, each entry represents someone who requests a loan from a bank. Each person is classified as either a high or low risk, according to the defined set of attributes.

Features Present and the Ones Selected

The dataset contains the following attributes of a person:

- Whether a checking account already exists or not (existingchecking)
- The duration for the load (duration)
- The credit history of the person (credithistory)
- The purpose for the loan(purpose)
- The amount of loan being asked for (creditamount)
- The amount in savings present (savings)
- Duration of employment (employmentsince)
- Installment rate (installmentrate)
- Their sex (sex)
- Other loans in their name (otherdebtors)
- Duration of residency (residencesince)
- Information about property owned (property)
- Their age (age)
- Other EMI's in their name(otherinstallmentplans)
- Type of house owned (housing)
- Existing loans with the current bank (existingcredits)
- Employment type (job)

- How many liabilities are possessed (peopleliable)
- If they have a telephone or not (telephone)
- If they are a foreign worker or not (foreignworker)
- Are the a high risk or not (classification)

The blue features are binary, the pink are integers/floats and the green ones are categorical which were looked at as one-hot encoded features.

The age and residence since features were ignored and the numerical columns were normalised.

Results

The following confusion matrices were obtained when the logistic regression classifier was used to classify the data.

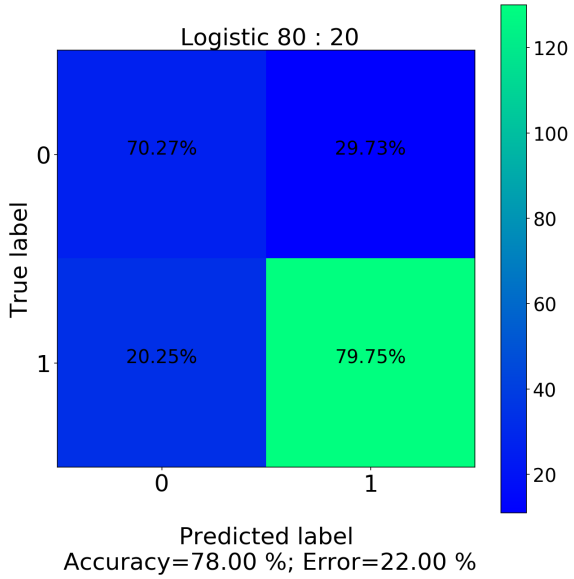


Figure 2.35: Logistic Regression with 80:20 split

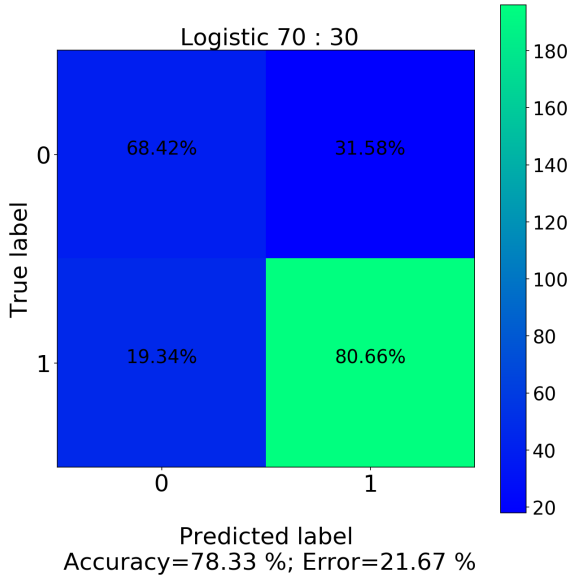


Figure 2.36: Logistic Regression with 70:30 split

Chapter 3

Regression Task

The polynomial is given by $y = 0.25x^3 + 1.25x^2 - 3x - 3$

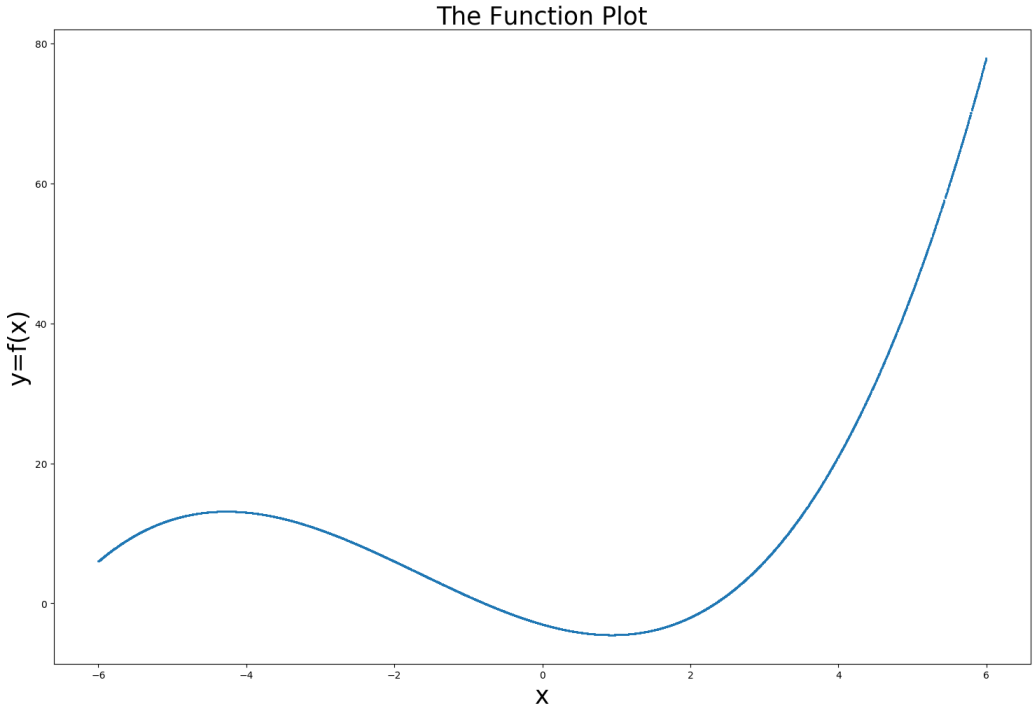


Figure 3.1: Decision boundaries

0.1 Implementation

Uniformly sampling K points $(x, f(x))$, L times, creates L datasets. For these L datasets, a polynomial of some degree $< K$ can be learned. Each polynomial learned is denoted by $g^{(l)}(x) \forall l \in \{1, .., L\}$.

For any point x, in the range of $f(x)$, the estimated value of $f(x)$, using the polynomials learned, is given by $\bar{g}\,(x)$, where

$$\bar{g}(x) = \frac{1}{L} \sum_{l=1}^{L} g^{(l)}(x) \tag{3.1}$$

The degree K of this polynomial can be best chosen by minimising the **expected mean squared error loss** $\mathbf{E}_{out}[(f(x) - \bar{g}(x))^2]$. The out here denotes any *out of sample* point, x.
Computing this requires the follwoing two terms;

$$(\text{bias})^{\,2} = \frac{1}{N} \sum_{n=1}^{N} \{\bar{g}\,(x_n) - f\,(x_n)\}^2 \tag{3.2}$$

$$\text{variance} \;= \frac{1}{N} \sum_{n=1}^{N} \frac{1}{L} \sum_{l=1}^{N} \left\{g^{(l)}\,(x_n) - \bar{g}\,(x_n)\right\}^2 \tag{3.3}$$

In equation 3.3, x_n denotes one point in the range of $f(x)$. Knowing the above terms, if there is some noise of variance σ^2 added to the dataset at hand, $\mathbf{E}_{out}$ can be approximated by;

$$\mathbf{E}_{out} = (\text{bias}^2 + \text{variance} + \sigma^2 \tag{3.4}$$

1 Results

1.1 On the Given Polynomial Function

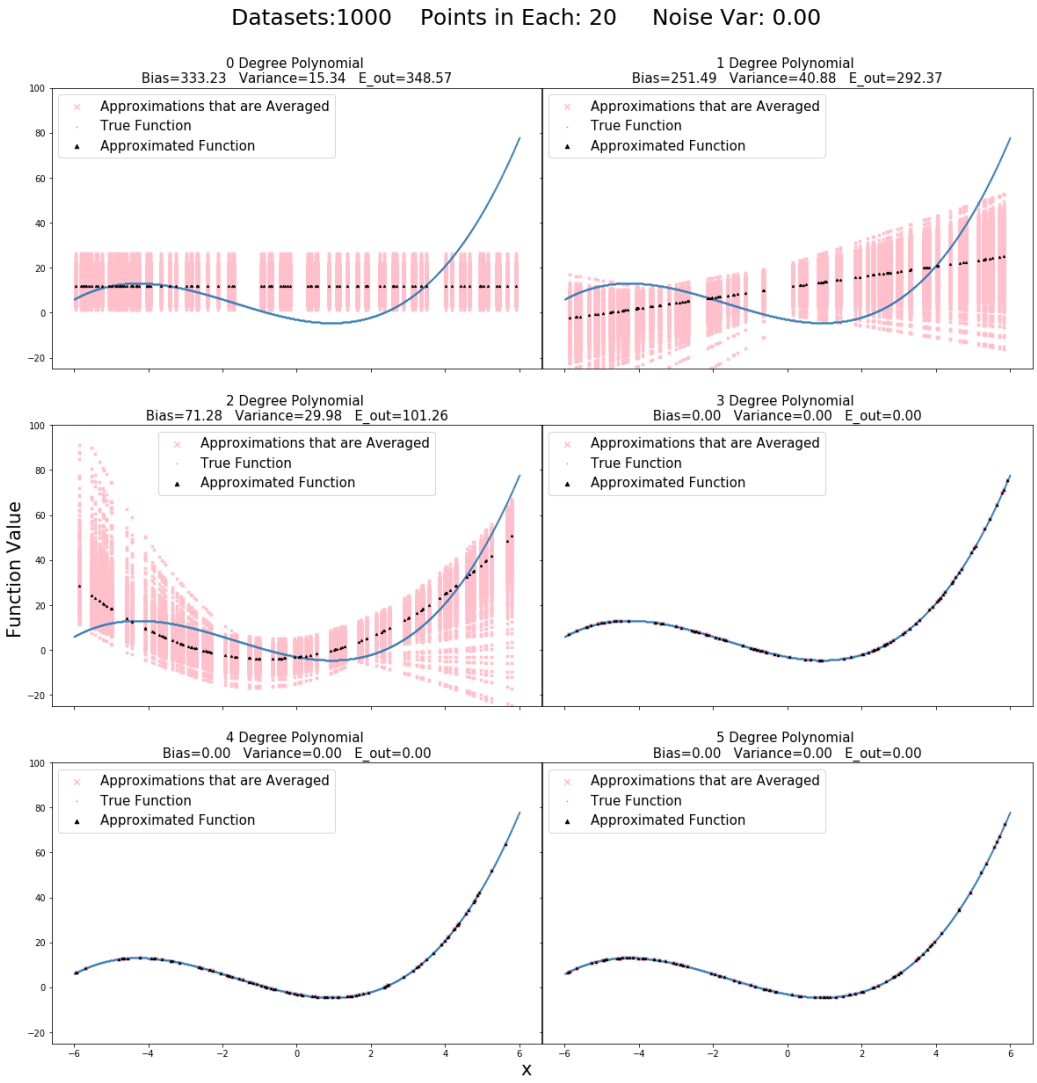


Figure 3.2: The best fit is a 3 degree polynomial

In the figure above, the pink lines are $g^{(l)}(x) \quad \forall l$ all plotted together. The black line depicts $\bar{g}\,(x_n)$ and the blue line depicts $f(x)$. As no noise was present, the function can be perfectly fit with a 3 degree or higher polynomial, up-to 19. This is because there are only 20 data points in each dataset indexed by l.

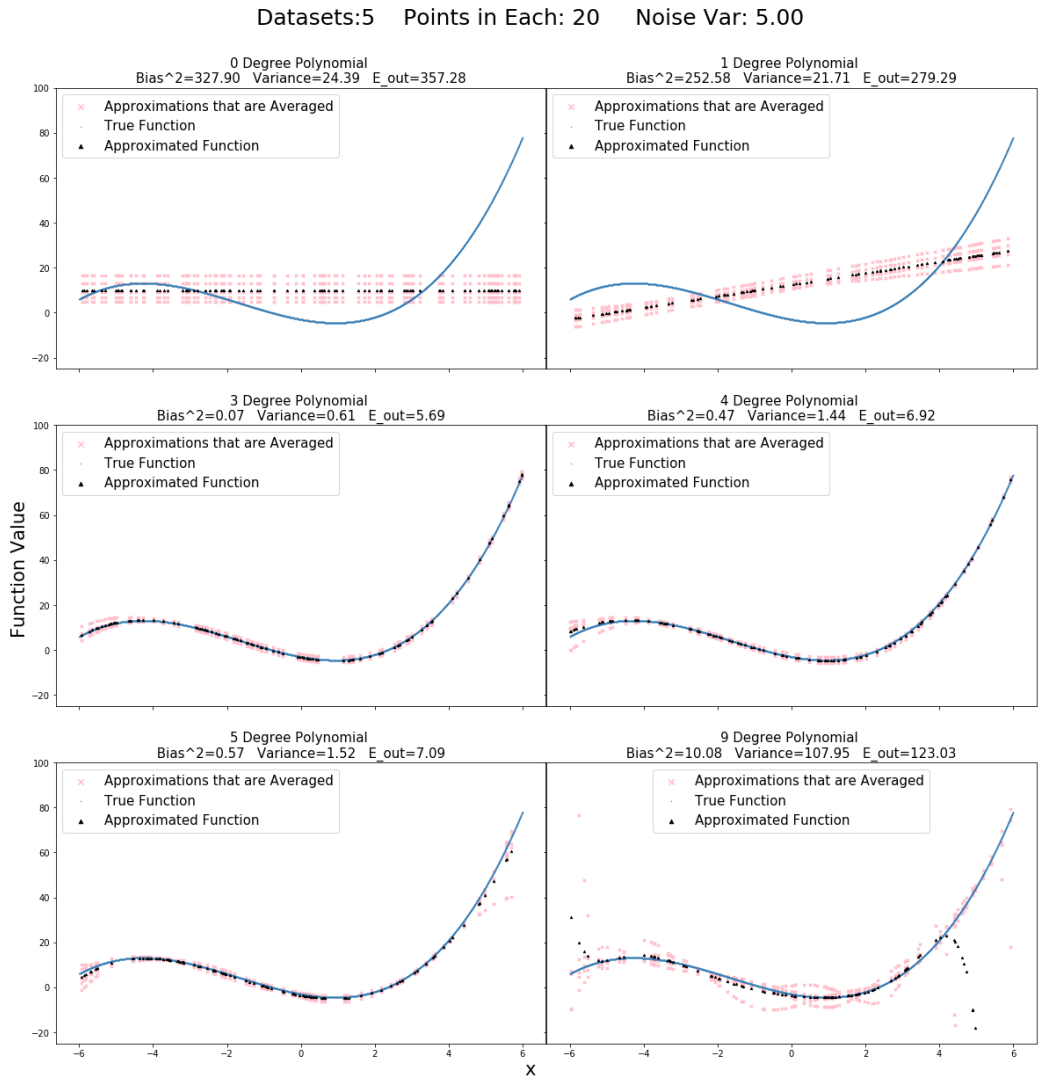


Figure 3.3: The best fit is a 3 degree polynomial

In this case, noise of variance 5 was added to the dataset. This results in a non-zero $\mathbf{E}_{out}$ and the 3 degree polynomial gives the least expected mean square loss.

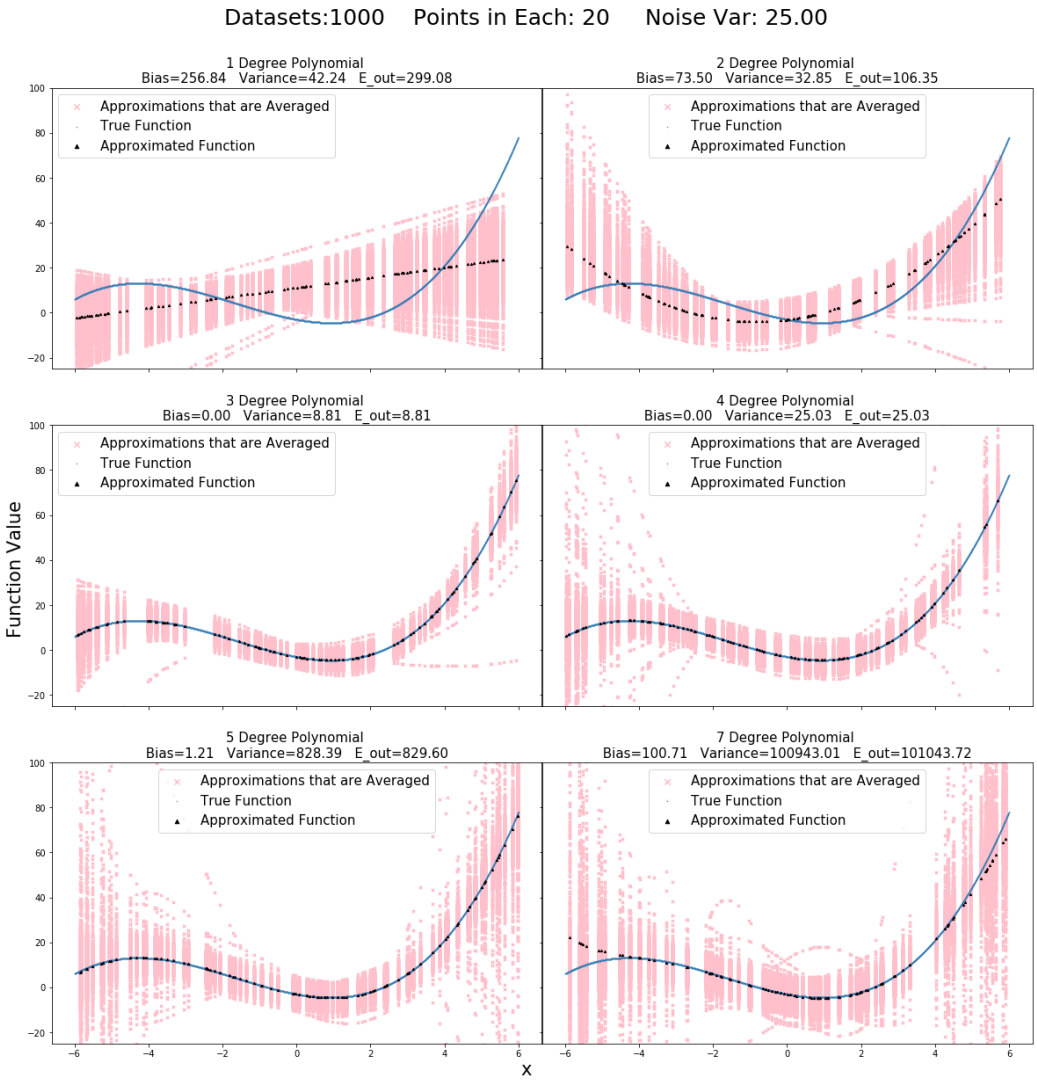


Figure 3.4: The best fit is any 3 degree or higher polynomial

The observation is that as the noise added increases, higher degree polynomials tend to fit the noise more and so lower degree polynomials seem a better fit for any given noisy data.

1.2 On a Sinusoidal Function

Running the algorithm on a sinusoidal function $y = sin(\frac{x\pi}{6})$, the results are as follows

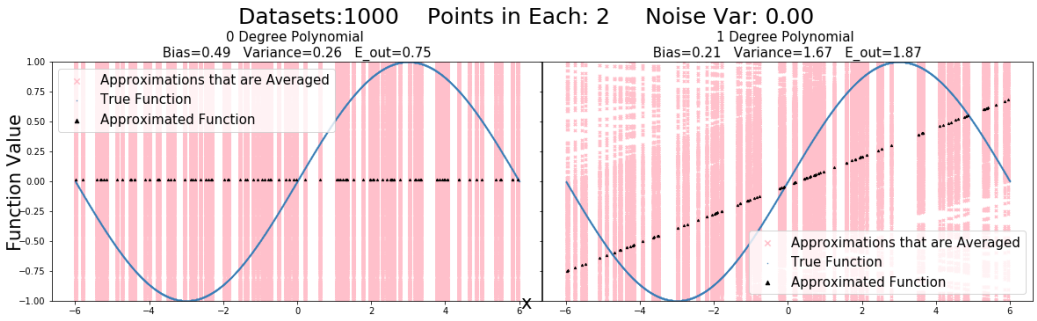


Figure 3.5: The best fit is a 0 degree polynomial

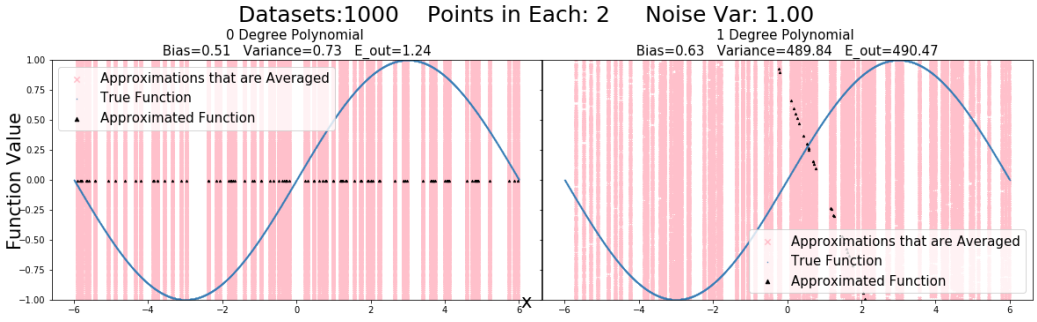


Figure 3.6: The best fit is a 0 degree polynomial

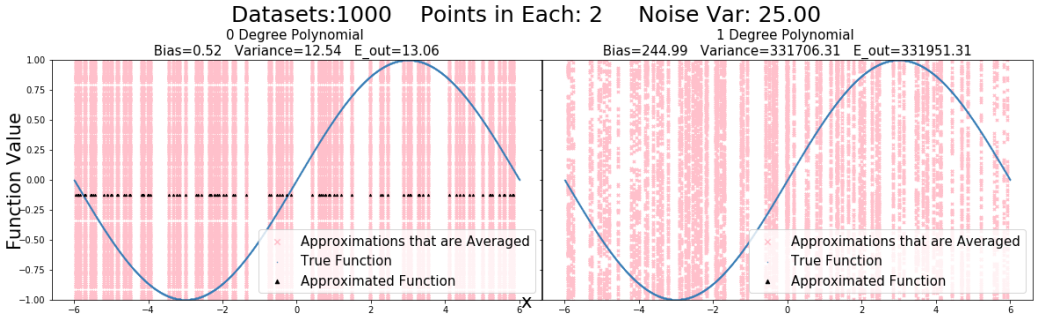


Figure 3.7: The best fit is a 0 degree polynomial

Given a few noisy points, it is best to learn a lower degree polynomial as the approximation for $f(x)$ as that leads to the least expected mean squared

error loss. The model then learns less of the noise, which is usually a very complex function, requiring a large hypothesis set to capture its statistics.