COMP 551 - Applied Machine Learning
Assignment 2

Tiffany Wang - 260684152

February 9, 2018

Acknowledgement

This assignment was fully completely by myself, Tiffany Wang. However, I discussed answers and methodologies with Daniel Lim, John Wu, Frank Ye and Nabil Chowdhury.

Question 1

I randomly generated 2000 class 0 and class 1 examples using the numpy.random.multivariate_normal function. Then I randomly selected 70% samples from both classes to build the training set, and used the remaining 30% of the data for the testing set.

The training and testing datasets are saved separately as $DS1\_test0.csv$, $DS1\_train0.csv$ for class 0, and $DS1\_test1.csv$, $DS1\_train1.csv$ for class 1. $DS1\_test.csv$ and $DS1\_train.csv$ hold the total datasets.

Question 2

It is important to mention that the weights and the accuracy measures differ from run to run, as the datasets are randomly generated.
The weights have the model are found using the following models:

$w_0 = log(p(y_0)) - log(p(y_1)) - \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 + \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1$

$w_1 = x^T\Sigma^{-1}(\mu_0 - \mu_1)$

The following are the results obtained from the LDA model:

$w_0 = 27.8476$

$$w_1 = \begin{bmatrix} 14.708 & -8.830 & -5.522 & -2.621 & -10.030 \\ -4.185 & 16.792 & -25.039 & -29.697 & 9.557 \\ -13.382 & -12.230 & 15.742 & 13.102 & -5.926 \\ 13.480 & 29.5497 & -6.974 & 0.014 & -5.3335 \end{bmatrix}$$

Accuracy = 95.64%
Precision = 95.55%
Recall = 95.70%
F-measure = 0.9562

Question 3

In this section, I used the k-NN model to train the model. The approach is to find the output k nearest points to the test input and classify the test input with the highest probably

class from the k nearest points.

Steps of the algorithm: (S1) Calculate euclidean distance from test input to every single training example.
(S2) Select the points closest to the test input (lowest euclidean distance).
(S3) Calculate the mean of the training example ground truth outputs.
(S4) Set prediction_output = 0 if mean ¡ 0.5
else prediction_output = 1

The k-NN model was trained using k from 1 to 50. Although the higher the k, the more accurate the value, the variation in accuracy from $k = 1$ to $k = 50$ was minimal, all between 51% and 57%. Therefore, I would say that there was not a particular k that performed better. In fact, the performance of k-NN is close to random guessing (50%), which shows that the model is not suited for this specific problem.

The *best* accuracy was obtained with $k = 47$.
Accuracy = 57.32%
Precision = 63.64%
Recall = 56.29%
F-measure = 0.5973

The sample set, generated from a single Gaussian multivariate distribution, is linear. Hence, as expected, the LDA model performed a lot better than k-NN. The accuracy was almost double, it was 50.1% better, to be exact.

Question 4

Essentially, the examples were created in the same way as in Question 1 using the numpy random multivariate normal generator. However, knowing that the mixture probability of the datasets 1, 2, 3 is (10%, 42%, 48%), I built the datasets with:
$training\_set = (10\% \cdot 2000 \cdot Dataset1 + 42\% \cdot 2000 \cdot Dataset3 + 48\% \cdot 2000 \cdot Dataset3) \cdot 70\%$
$testing\_set = (10\% \cdot 2000 \cdot Dataset1 + 42\% \cdot 2000 \cdot Dataset3 + 48\% \cdot 2000 \cdot Dataset3) \cdot 30\%$

The training and testing datasets are saved separately as $DS2\_test0.csv$, $DS2\_train0.csv$ for class 0, and $DS2\_test1.csv$, $DS2\_train1.csv$ for class 1. $DS2\_test.csv$ and $DS2\_train.csv$ hold the total datasets.

Question 5

- LDA model:

$w_0 = -0.0279$

$$w_1 = \begin{matrix} [0.0402 & -0.0017 & 0.052 & -0.0063 & -0.025 \\ 00315 & -0.052 & 0.0254 & -0.0221 & 0.0189 \\ 0.0522 & 0.00537 & 0.0072 & -0.0037 & -0.0084 \\ 0.0633 & -0.0032 & -0.02449 & -0.0776 & 0.0060] \end{matrix}$$

Accuracy = 50.12%
Precision = 48.88%
Recall = 61.115%
F-measure = 0.543

- k-NN model
  The *best* accuracy obtained with $k = 25$.
  Accuracy = 51.23%
  Precision = 55.71%
  Recall = 49.68%
  F-measure = 0.5253

Since the sample is a mixture of three different multivariate distributions, it lost its linearity. The performance of LDA decreased from 95.64% to 50.12%.
However, in the case of the k-NN model, the prediction results are the similar.

Question 6

On the first hand, the accuracy of the LDA model is changes drastically between DS1 and DS2. As mentioned earlier, DS1 is linear, so the predictions using the LDA model is expectedly precise. However, as the DS2 loses linearity with the mixture of three different Gaussian distributions, the performance of the LDA model drops to around 50%, which is close to random guessing. I would like to conclude that the LDA model is powerful for the prediction of linear models.

On the other hand, the k-NN model have similar performances for both DS1 and DS2. As k-NN does not make any assumptions on the dataset, its results are independent of the linearity of the datasets.