COMP 551 - Assignment 2

Tiffany Wang - 260684152

February 12, 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:

$$w_0 = 27.8476$$

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

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 less than 0.5
else prediction_output = 1

The k-NN model was trained using k from 1 to 50. The variation in accuracy from $k = 1$ to $k = 50$ was minimal, all between 51% and 57%.
The Gaussian distribution is linear, and although the mean used for both classes are different, the sample points are not clustered in a specific regions. The k-NN model is, thus, not suited for this specific problem. In fact, the performance of k-NN is close to random guessing (50%).

K	Accuracy
1	0.513158
2	0.512336
3	0.54523
4	0.523026
5	0.521382
6	0.524671
7	0.528783
8	0.537829
9	0.537007
10	0.53125
11	0.525493
12	0.516447
13	0.527961
14	0.532895
15	0.527138
16	0.524671
17	0.544408
18	0.533717
19	0.537007
20	0.523849
21	0.532072
22	0.541941
23	0.546053
24	0.539474

25	0.552632
26	0.549342
27	0.547697
28	0.550987
29	0.539474
30	0.546053
31	0.556743
32	0.553454
33	0.554276
34	0.560033
35	0.555921
36	0.560033
37	0.553454
38	0.558388
39	0.5625
40	0.5625
41	0.5625
42	0.5625
43	0.560855
44	0.570724
45	0.560855
46	0.570724
47	0.573191
48	0.568257
49	0.571546

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:
  $w_0 = -0.05699$
  $$w_1 = \begin{matrix} [-0.01767 & 0.00082 & -0.01091 & -0.01414 & 0.00181 \\ 0.00082 & -0.01091 & -0.01414 & 0.00181 & 0.01315 \\ -0.01091 & -0.01414 & 0.00181 & 0.01315 & -0.07297 \\ -0.01414 & 0.00181 & 0.01315 & -0.07297 & -0.05553] \end{matrix}$$

  Accuracy = 52.69%
  Precision = 52.05%
  Recall = 51.71%
  F-measure = 0.537

- k-NN model

K	Accuracy
1	0.512175
2	0.525609
3	0.532326
4	0.521411
5	0.531486
6	0.537364
7	0.533165
8	0.523929
9	0.534845
10	0.532326
11	0.528128
12	0.534005
13	0.530647
14	0.532326
15	0.518892
16	0.527288
17	0.515533
18	0.529807
19	0.524769
20	0.528128
21	0.521411
22	0.52309
23	0.52225
24	0.516373
25	0.517212
26	0.523929
27	0.512175
28	0.527288
29	0.525609
30	0.523929
31	0.534005
32	0.526448
33	0.528967
34	0.524769
35	0.526448
36	0.529807
37	0.526448
38	0.528967
39	0.521411
40	0.528967
41	0.534005
42	0.537364
43	0.540722
44	0.539882
45	0.548279
46	0.549958
47	0.541562
48	0.540722
49	0.532326

The *best* accuracy obtained with $k = 46$.
Accuracy = 54.99%
Precision = 49.49%
Recall = 55.28%
F-measure = 0.5222

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.