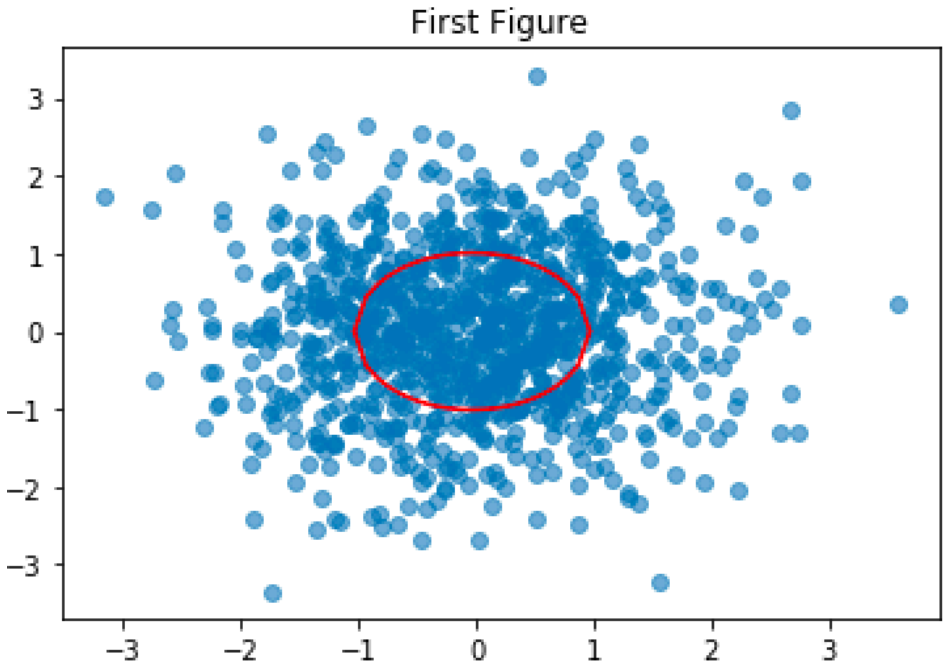
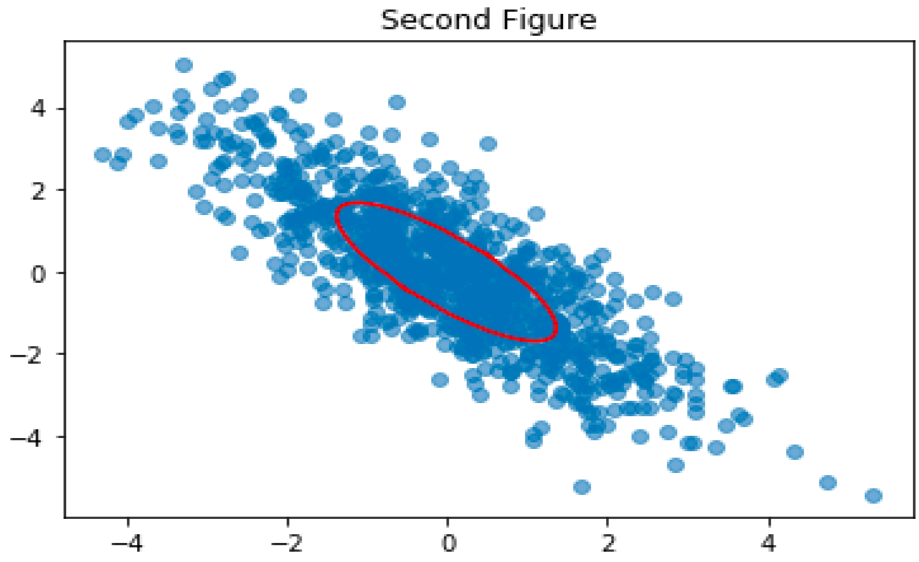
Q1

(a).

For the class described by covariance matrix $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$



For the class described by covariance matrix $\Sigma = \begin{bmatrix} 2 & -2 \\ -2 & 3 \end{bmatrix}$



(b).

The plot result can be seen in question(a) figure.

The calculation process:

1.We need to call np.linalg.eig function to get the eigenvalue and eigenvector for given each covariance.

2.Then get the diagonal matrix of eigenvalue, using np.diag(eigenvalue) function.

3.To get $\Lambda^{-\frac{1}{2}}$, we use np.sqrt(eigenvalue), call np.linalg.inv function to get the result.

4.With the given whiten transformation function $\phi^T \cdot \Lambda^{-\frac{1}{2}}$ to whiten the sample generated in question (a), we get the circle contour, with mean as the circle center

5.At last, use the inverse whitening process to get the final result.

(c).

For the class described by covariance matrix $\sum = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

```
[[1.01626323 0.06352617]
 [0.06352617 1.05841334]]
```

For the class described by covariance matrix $\sum = \begin{bmatrix} 2 & -2 \\ -2 & 3 \end{bmatrix}$

```
[[ 2.07382516 -1.99047658]
 [-1.99047658  2.91635328]]
```

The calculating process is shown in source code

(d).

we generate the dataset in Gaussian distribution, each sample in the dataset is generated randomly according to the covariance matrix and mean. Most sample meet the class data distribution with certain covariance matrix in the question (a). However, there are still some outliers which lead to the covariance matrix is not the same as the one in question (a). If we can generate enough large number of samples in our dataset, the result will be much closer to the covariance matrix in the question (a).

Q2

(a).

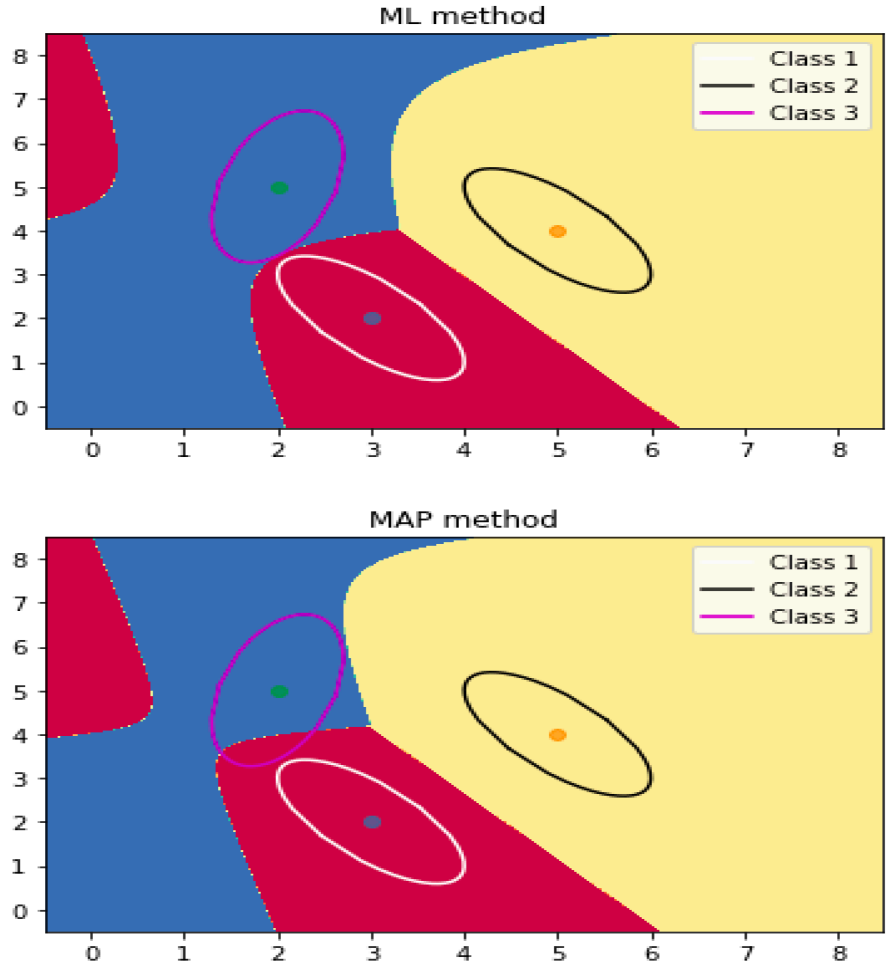


Figure 1

In the Figure 1, the mean is the dot in each ellipse.

From the figure, although the results are similar for MAP and ML methods, there are still some difference between ML method and MAP methods, because the ML method assume the priors of these three classes are the same; however, the difference is subtle. MAP method classifies more class 1 space compared with ML method. First standard deviation contours in two methods are same, because they use the same covariance matrix for each class.

(b).

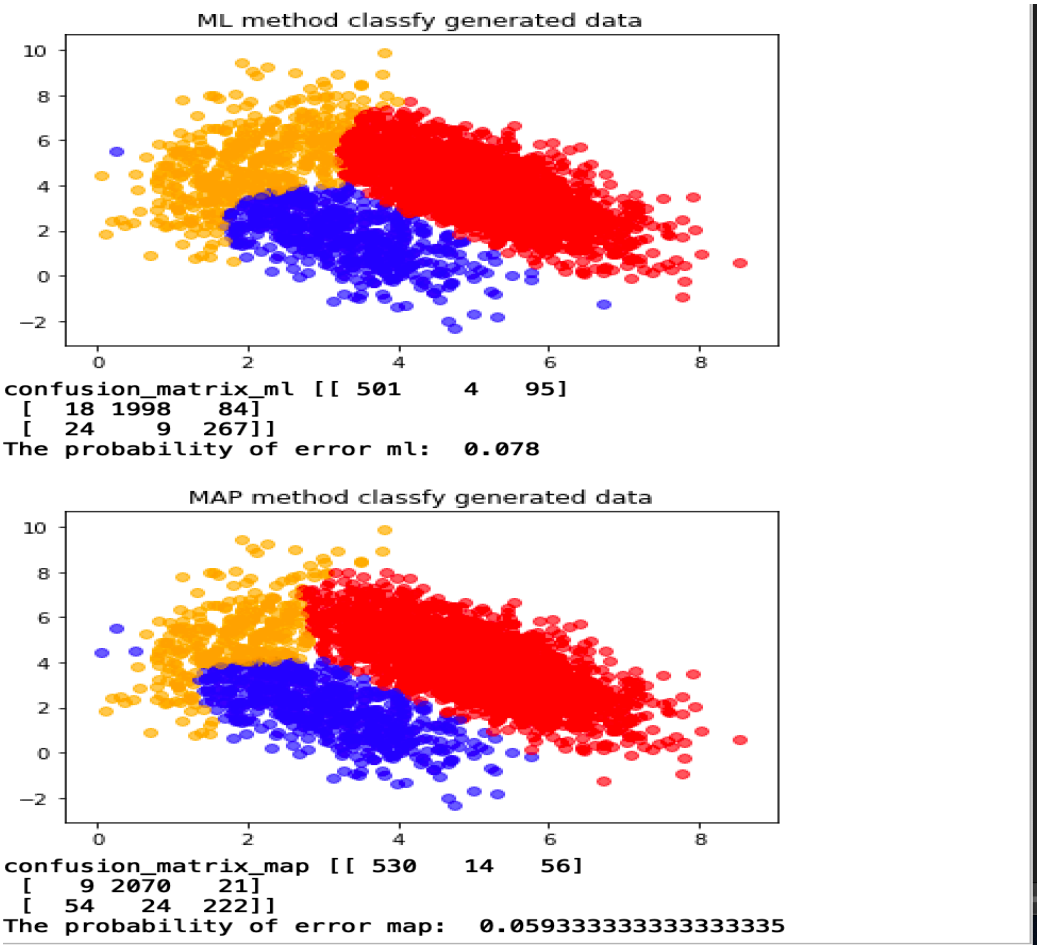


Figure 2

Confusion matrix, experimental P(ε) and the generated data according to prior probabilities and covariance matrices for each class are shown in the Figure 2.

We can see that the classifying results are similar to the question(a). And the experimental P(ε) of MAP is less than that of ML.

From lecture, we know that Maximum A Posteriori (MAP) minimize the Bayes error. My result for this question makes sense.

Q3

(a). Shown in source code.

(b). proposed d = 62. Calculated process can be seen in source code

(c).

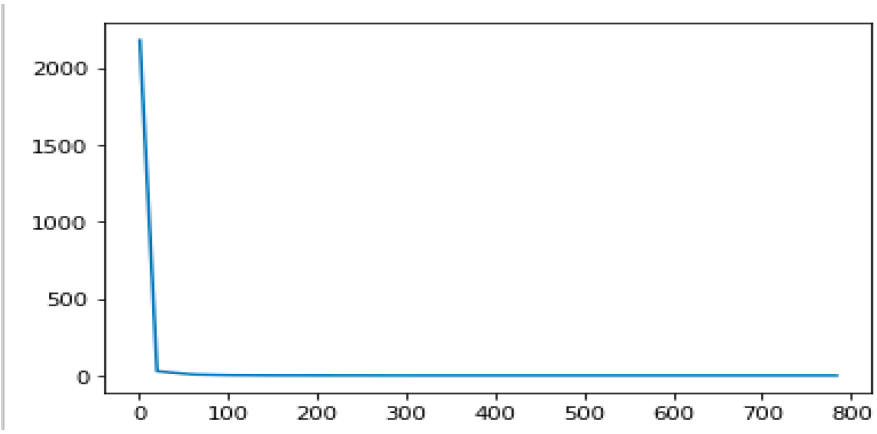


Figure 3

From the Figure 3, we can get that as the number of d increases, the MSE is decreasing. Between 0 and around 20, it decreases rapidly, after which the gap between different number of d is subtle.

We can also know that from the question b, the MSE is small enough at around 62(POV =95%).

(d).

Figure 4

I select the first 8 in the dataset to construct. From figure 4, we can see that the difference between these .png file is clear. With the increases, the picture of 8 is clearer. Seen from 1.png and 10.png, the clarity of 8 increases a lot. Seen from 250.png and 784.png, they are almost same.

The results of this question meet the result of question (c), at the start of d (e.g. 1.png, 10.png, 50.png), the clarity increases rapidly. After around 50 the clarity increases little.

(e).

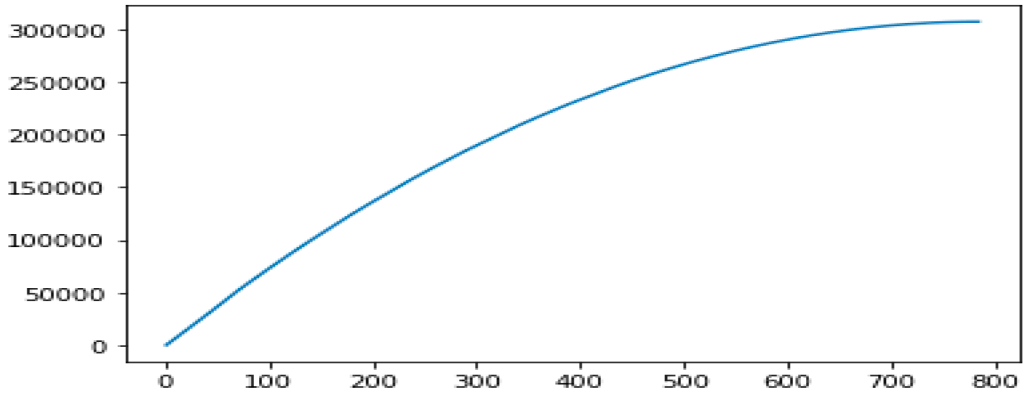


Figure 5

From Figure 5, we can know that with the d increases, the eigenvalues increase steadily. And the speed of increasing decreases. From the above discussion, we can get that with the sum of eigenvalues of sample increase, the accuracy also increases.

Q4

(a).

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - h_\theta(x)\right)^{(i)}\right]$$

(b).

Shown in source code

(c).

When the learning rate is 0.00001 and the epoch number is 5000

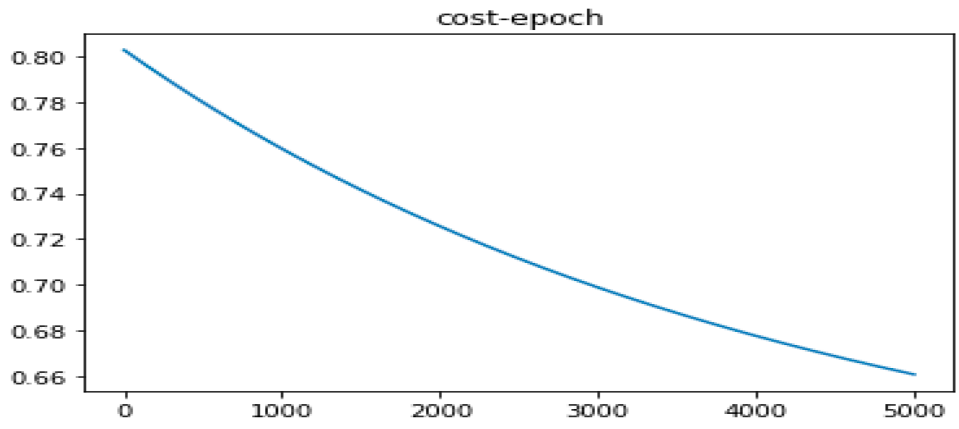


Figure 6

When the learning rate is 0.001 and the epoch number is 5000

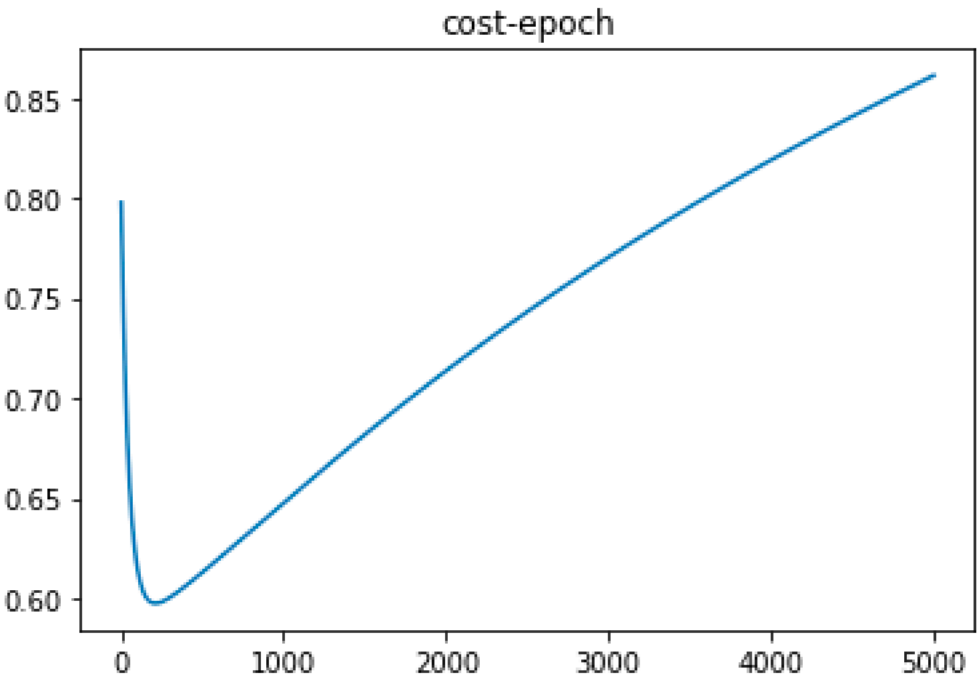


Figure 7

(d).

When the learning rate is 0.001 and the epoch number is 5000

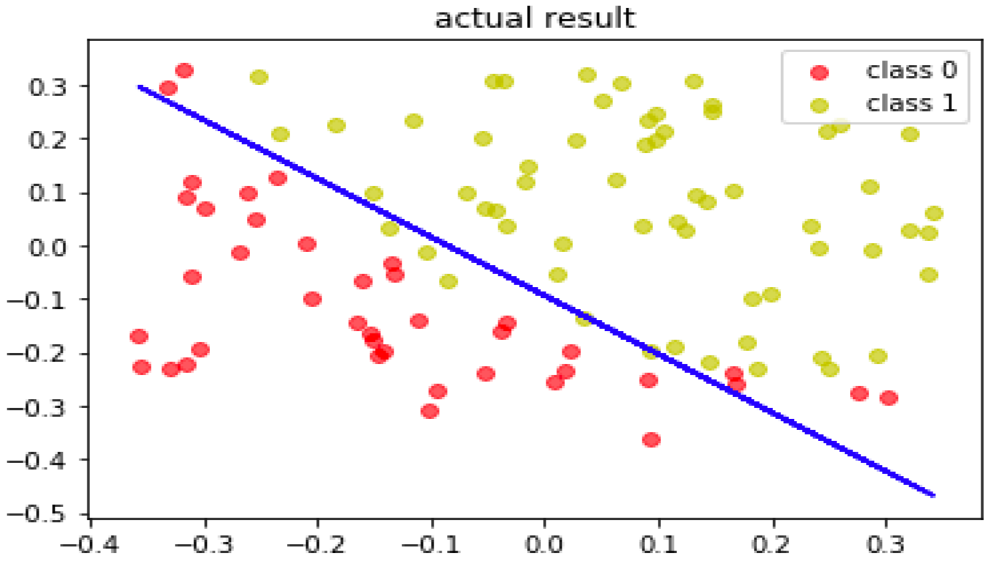


Figure 8

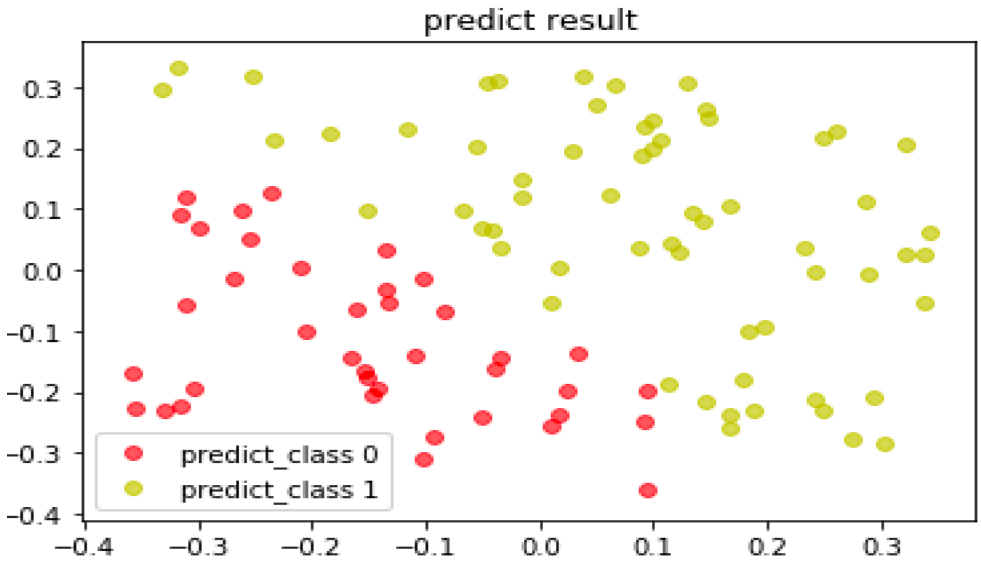


Figure 9

```
error rate:  0.11
```

(e).

Shown Figure 9

(f).

Shown Figure 8

Q5

(a).

All the features in the feature space are independent to each other in Naïve Bayes. However, in the Bayes classifiers, we do not have this assumption.

(b).

When the features we consider for learning are independent to each other.

(c).

When the number of features is small, and we can easily calculate corresponding parameters we need, such like likelihood and prior probability, to get the posterior probability.

(d).

For Bayes Approach, the number of parameters for modelling likelihood, we need to calculate $2(2^n - 1)$, when we consider two classes label and binary feature. Therefore, it is very expensive to compute.

For Naïve Bayes to calculate the likelihood, we just need to calculate 2n parameters in this situation.