This report contains four major parts, and each major part is for one question in the assignment. As for the last part of survey, its context is in Q5.

Question 1: Beta-binomial Naive Bayes

1. **Algorithm Design**
   In this part ,a classifier based on Beta-Binomial Naive Bayes is built to handle the binarized data from given dataset and finally predict whether an email is or is not a spam.

   (a) **Data Processing**
   In this part, we use binarization method to binarize features: $I(x_{ij} > 0)$. In other words, if a feature is greater than 0, it is simply set to 1. If it is less than or equal to 0, it is set to 0.Here is the formula:

$$\begin{cases} x_{ij} = 1 & x_{ij} > 0 \\ x_{ij} = 0 & x_{ij} \leq 0 \end{cases} \tag{1}$$

   (b) **Key ideas**
   Based on training dataset $D(x_{1:N}, y_{1:N})$, in order to predict the class label y of a specific sample x, we need to compute the posterior possibility of all potential class labels of x, and then choose the highest one:

$$p(\tilde{y} = c \mid \tilde{x}, D) \propto p(\tilde{y} = c \mid y_{1:N}) \prod_{j=1}^{D} p(\tilde{x}_j \mid x_{i \in c, j}, \tilde{y} = c) \tag{2}$$

   We assume all of the 57 features are following Beta distribution:

$$p(\theta, a, b) = \frac{1}{B(a, b)} \theta^{a-1} (1 - \theta)^{b-1} \tag{3}$$

   in which $a = b = \alpha$. With the prior $Beta(\alpha, \alpha)$,we can utilize the posterior of θ

$$p(\theta|D) = Beta(\theta|N_1 + a, N_0 + b) \tag{4}$$

   to calculate $p(\widetilde{x}|D$,which is actually the mean of $p(\theta|D)$:

$$p(\tilde{x} = 1|D) = E(\theta|D) = \frac{N_1 + \alpha}{N + \alpha + \alpha} \tag{5}$$

$$p(\tilde{x} = 0|D) = 1 - p(\tilde{x} = 1|D) \tag{6}$$

To compute the probability of $p(\tilde{y}=1 \mid \tilde{x}, D)$ and $p(\tilde{y}=0 \mid \tilde{x}, D)$, we can implement the following formula:

$$\log p(\tilde{y}=1 \mid \tilde{x}, D) \propto \log p\left(\tilde{y}=1 \mid \lambda_{ML}\right)+\sum_{j=1}^{D} \log p\left(\tilde{x}_j \mid x_{i \in c, j}, \tilde{y}=1\right) \tag{7}$$

$$\log p(\tilde{y}=0 \mid \tilde{x}, D) \propto \log p\left(\tilde{y}=0 \mid \lambda_{ML}\right)+\sum_{j=1}^{D} \log p\left(\tilde{x}_j \mid x_{i \in c, j}, \tilde{y}=0\right) \tag{8}$$

Then to get the reult of probability $p(\tilde{y}=1 \mid \tilde{x}, D)$ and $p(\tilde{y}=0 \mid \tilde{x}, D)$ to make the final prediction.

2. **Result analysis**

   (a) **Training and test error rates versus α**
   Here is the picture 1 shown.

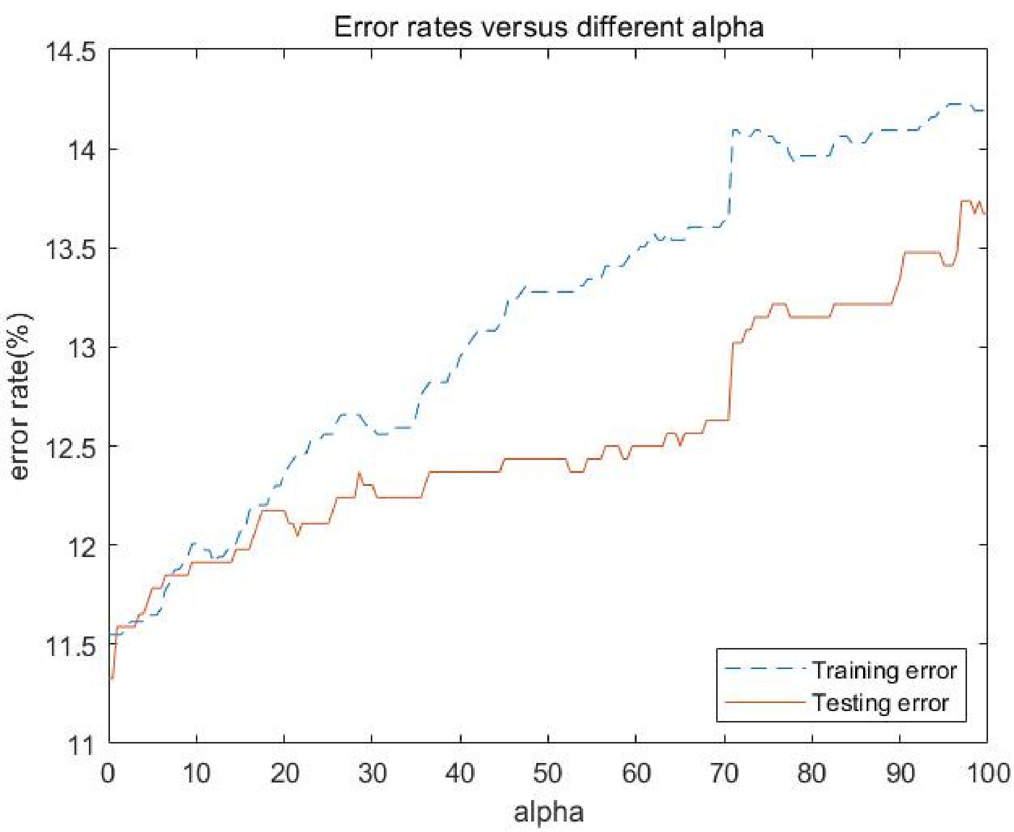

Figure 1: Training and testing error rates versus α in Q1

(b) **What do you observe about the training and test errors as α change?**

i. Firstly, generally both the training and testing errors will increase as α increases. In my view, it is because the prior $Beta(\alpha, \alpha)$ we set on the feature distribution does not match the true feature distribution, moreover as α increases ,the influence of the prior $Beta(\alpha, \alpha)$ on the final prediction will also increase. So as α increases, the error rates also increase.

ii. Secondly, the training error is always higher than testing error given the same number of α. I suppose that the mismatch between the distribution assumption and true feature distribution cause this problem.

iii. Finally, the gap between training error and testing error is gradually bigger when λ grows more.

(c) **Training and testing error rates** for $\alpha = 1$,10 and 100
Here is the final result from simulation in Table 1

α	Training error	Testing error
1	11.5498%	11.5885%
10	12.0065%	11.9141%
100	14.1925%	13.6719%

Table 1: Training and testing error rates for $\alpha = 1$,10 and 100 in Q1

Question 2: Gaussian Naive Bayes

1. **Algorithm Design**
In this part, a classifier based on Gaussian Naive Bayes is built to handle the log-transformed data from given dataset and finally predict whether an email is or is not a spam.

(a) **Data Processing**
All the training and testing data is transformed into logarithm form. As the requirement of the assignment, we edit the feature of dataset in this transformation form using $log(x_{ij} + 0.1)$ (assume natural log)

(b) **Key ideas**
As in this question, we assume features subject to Gaussian distribution, the first step is to compute the ML estimate of conditional mean(μ) and variance(σ^2) of each feature based on training data:

$$\left(\hat{\mu}, \hat{\sigma}^2\right) \triangleq \underset{\mu, \sigma^2}{\operatorname{argmax}} p\left(x_1, \cdots, x_N \mid \mu, \sigma^2\right) \tag{9}$$

Then the probability of each feature can be calculated by:

$$p(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{10}$$

After dividing the training data into 2 classes, we can calculate the Maximum Likelihood estimation of mean μ and variance σ^2 over each features.

$$\begin{aligned} \frac{\partial L}{\partial \mu} &= \frac{\partial}{\partial \mu}\left(\sum_{n=1}^{N} -\frac{(x_n-\mu)^2}{2\sigma^2}\right) = \sum_{n=1}^{N} \frac{(x_n-\mu)}{\sigma^2} = 0 \\ \Longrightarrow \hat{\mu} &= \frac{1}{N}\sum_{n=1}^{N} x_n \end{aligned} \tag{11}$$

$$\begin{aligned} \frac{\partial L}{\partial \sigma} &= \frac{\partial}{\partial \sigma}\left(\sum_{n=1}^{N} -\frac{(x_n-\mu)^2}{2\sigma^2} - N\log\sigma\right) = \sum_{n} \frac{(x_n-\mu)^2}{\sigma^3} - \frac{N}{\sigma} = 0 \\ \Longrightarrow \hat{\sigma}^2 &= \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu)^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n-\hat{\mu})^2 \end{aligned} \tag{12}$$

Finally we implement the following formula to calculate the probability $p(\tilde{y}=1 \mid \tilde{x}, D)$ and $p(\tilde{y}=0 \mid \tilde{x}, D)$ to make the prediction:

$$\log p(\tilde{y}=c \mid \tilde{x}, D) \propto \log p(\tilde{y}=c \mid \lambda_{ML}) + \sum_{j=1}^{D} \log p(\tilde{x}_j \mid x_{i\in c,j}, \tilde{y}=c) \tag{13}$$

2. **Result analysis**

(a) **Training and testing error rates**
Here is the final result from simulation in Table 2

Training error	Test error
16.5742%	16.0156%

Table 2: Training and testing error rates for $\alpha = 1{,}10$ and 100 in Q2

Question 3: Logistic regression

1. **Algorithm Design** In this part, a classifier based on logistic regression with l_2 regularization is built to handle the log-transformed data from given dataset and finally predict whether an email is or is not a spam.

   (a) **Data Processing**
   All the training and testing data is transformed into logarithm.It same as Q2.

   (b) **Key ideas**
   Based on the a given tranining set, we built a discriminative model $p(y|x, w)$, and then we get first extimate w that satidfy:

$$\hat{w} = \underset{w}{\operatorname{argmax}} p\left(y_{1:N} \mid x_{1:N}, w\right) \tag{14}$$

   From the tranformation in the lectures, we assume sampples are independent:

$$\hat{w} = \underset{w}{\operatorname{argmin}} -\sum_{i=1}^{N} \log p\left(y_i \mid x_i, w\right) \triangleq \underset{w}{\operatorname{argmin}} NLL(w) \tag{15}$$

   Now, the next objective is to find a w which can minimize $NLL(W)$, the $NLL(W)$ can be expressed in the following ways:

$$\log p\left(y_i = 1 \mid x_i, w\right) = \log \frac{1}{1 + \exp\left(-w^T x_i\right)} = \log \mu_i \tag{16}$$

$$\log p\left(y_i = 0 \mid x_i, w\right) = \log\left(1 - p\left(y_i = 1 \mid x_i, w\right)\right) = \log\left(1 - \mu_i\right) \tag{17}$$

$$NLL(w) = -\sum_{i=1}^{N} \log p\left(y_i \mid x_i, w\right) = -\sum_{i=1}^{N} \left[y_i \log \mu_i + \left(1 - y_i\right) \log\left(1 - \mu_i\right)\right] \tag{18}$$

   Then to apply derivatives in the above expressions:

$$g = \frac{d}{dw} NLL(w) = \sum_{i=1}^{N} \left(\mu_i - y_i\right) x_i = X^T(\mu - y) \tag{19}$$

$$H = \frac{d}{dw} g(w)^T = \sum_{i=1}^{N} \mu_i \left(1 - \mu_i\right) x_i x_i^T \tag{20}$$

   In addition, we also need to add a bias term and some constrains on the origin model, which can free the decision boundary and greatly reduce overfitting. Therefore, l_2 regularization is necessary. We can rewrite the gradient matrix and hessian matrix:

$$g_{reg}(W) = g(W) + \lambda W \tag{21}$$

$$H_{reg}(W) = H(W) + \lambda I \tag{22}$$

where I is a $(D+1)X(D+1)$ identity matrix,λ is the degree of regularization, the bold W means w with a bias term.
And following the regularization steps, we can get the new version of $NLL(W)$:

$$NLL_{reg}(\mathbf{w}) = NLL(\mathbf{w}) + \frac{1}{2}\lambda \mathbf{w}^T \mathbf{w} \tag{23}$$

We take the Newton's method to find the desired W, so, the first step is to initialize W as a zero vector, after that is to repeat computing until convergence:

$$W_{k+1} = W_k - H_k^{-1} g_k \tag{24}$$

In my code designing, if the difference of $NLL_{reg}(W)$ between two successive iteration is less than a threshold(0.01), the algorithm would judge the result convergences.
Finally, I use W to do prediction in testing data.

2. **Result analysis**

   (a) **Training and testing error rates versus α**
   Here is the picture 2 shown.

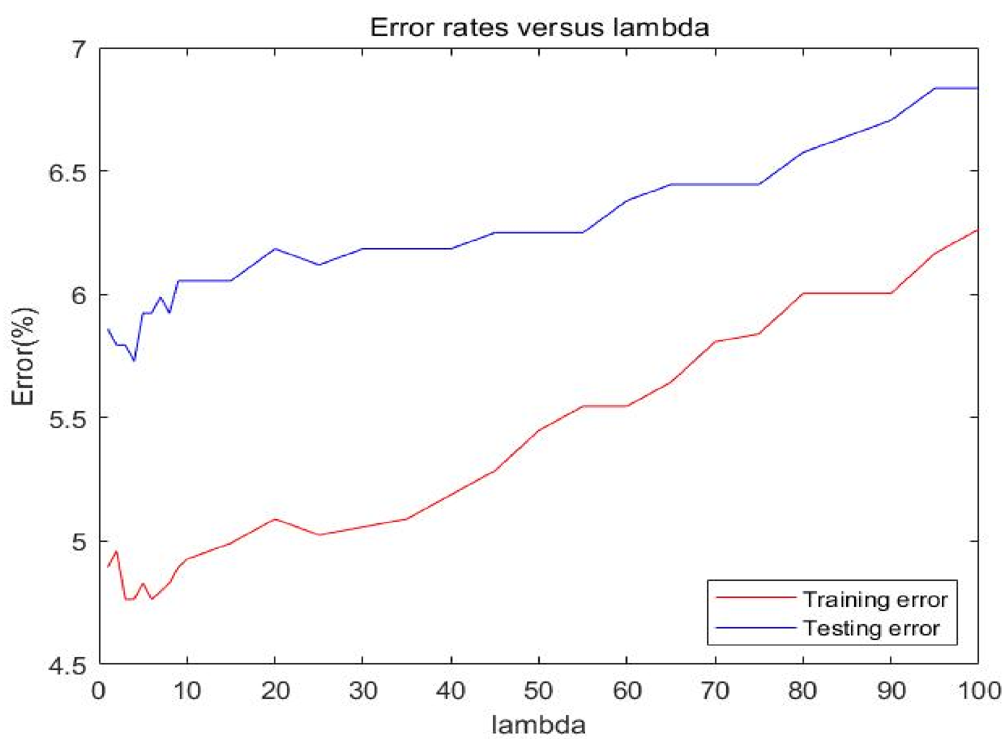

Figure 2: Training and testing error rates versus α in Q3

(b) **What do you observe about the training and test errors as α change?**

i. For training set, when λ is less than 2, the error will decrease as λ increases. However, when λ gets bigger than 2, the error will basically increase as the growth of λ.

ii. As for the testing data, the situation is similar as that of training data. However, the growth speed is slower than that of training data.

iii. Totally, the flow of testing error is always higher than that of training error in that specific range.

(c) **Training and testing error rates** for $\alpha = 1{,}10$ and 100

Here is the final result from simulation in Table 3

α	Training error	Test error
1	4.8940%	5.8594%
10	4.9266%	6.0547%
100	6.2643%	6.8359%

Table 3: Training and testing error rates for $\alpha = 1{,}10$ and 100 in Q3

Question 4: K-Nearest Neighbors

1. **Algorithm Design**

In this part, a KNN classifier based on logistic regression with l_2 regularization is built to handle the log-transformed data from given dataset with the Euclidean distance as the measurement of difference between samples.

(a) **Data Processing**

All the training and testing data is transformed into logarithm form.It same as Q2.

(b) **Key ideas**

The key idea of KNN-classifier is to predict the class label of a specific sample by collecting and analysing its K nearest surrording training samples.Then it use the Euclidean distance to do prediction.

$$\text{Distance }(a, b)=\left(\sum_{i=1}^{D}\left|a_{i}-b_{i}\right|^{2}\right)^{\frac{1}{2}} \tag{25}$$

As for The formula of posterior is derived based on the joint probability $p(x, y = c) = \frac{k_c/N}{V}$, and then we get:

$$p(y=c \mid x)=\frac{p(x, y=c)}{\sum_{c'=1}^{C} p\left(x, y=c'\right)}=\frac{\frac{k_{c}/N}{V}}{\sum_{c=1}^{C} \frac{k_{c'}/N}{V}}=\frac{k_{c}}{K} \tag{26}$$

What we really should pay attention to is that if K is an even number, and among the K nearest neignbors in a test sample, it means only half number of training samples from class 0 and another half number of training samples from class 1. As a result, in my code, I would predict this testing sample as class 1.

2. **Result analysis**

   (a) **Training and testing error rates versus α**
   Here is the picture 3 shown.

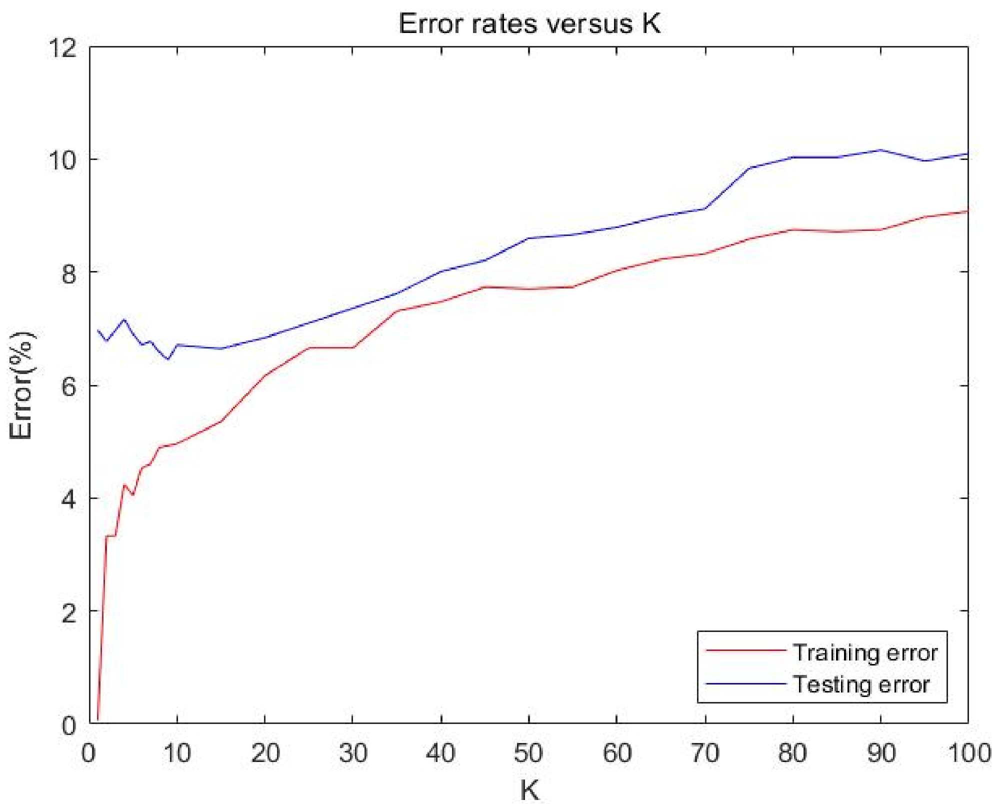

Figure 3: Training and testing error rates versus α in Q4

   (b) **What do you observe about the training and test errors as α change?**
      i. For training set, generally, the error will increase as K increases, and theoretically the error will be 0 when K = 1, because the Euclidean distance between every training sample and itself is 0, or you can say that the 1 nearest neighbor of every training sample is itself, so there will be no error when K = 1.
      ii. For testing set, when K is about less than 10, the error will fluctuate as K increases. When K is bigger than 10, the error will generally increase as K increases.

iii. In most cases, testing error is bigger than training error. But when K is very small, the gap between test error and training error is very big, as K increases, the gap gets smaller and after K is more than 50, the testing error line and training error line gradually separate and the gap between them gets a little bigger.

(c) **Training and testing error rates** for $\alpha = 1,10$ and 100
Here is the final result from simulation in Table 4

α	Training error	Test error
1	0.0653%	6.9661%
10	4.9592%	6.7057%
100	9.0701%	10.0911%

Table 4: Training and testing error rates for $\alpha = 1,10$ and 100 in Q4

Question 5: Survey

Roughly, I spent 5 hours on Q1, 5 hours on Q2, 10 hours on Q3 and 10 hours on Q4, and 10 hours on writing this report. In a word, I spent 40 hours to finish this assignment.