1 Question 1

The data set presented contains 50 data points consisting of 5 continuous features and 1 continuous response. Let's consider some key summary statistics to explore the data.

	x_1	x_2	x_3	x_4	x_5	y
Mean	0.1703	0.2314	-0.1747	0.1562	0.4527	7.5067
Min	-4.8867	-3.4736	-8.6123	-4.2361	-2.8289	-10.1835
Max	7.9114	6.4195	4.5124	2.6414	4.4669	72.9078
Standard Deviation	1.8269	1.8620	1.8386	1.2561	1.4496	12.0048
Pearson Correlation with y	0.4862	-0.0178	0.0156	-0.1414	0.1575	1

Table 1: Summary statistics of data set

Looking at Table 1, it can be seen that all 5 features have their mean near the y-axis. x_1, x_2, x_3, and x_4 all appear to have high leverage points with either their minimum or maximum values standing 3 to 4 standard deviations away from their means. These high leverage points may be classified as outliers depending on the corresponding y values or other dimensions. Looking at the Pearson correlation coefficients, we see that x_1 has medium positive correlation with y. The other 4 variables display very weak to weak correlation and will likely contain little to no information about y. It should be noted that Pearson's correlation coefficient only measures correlation on a linear scale. The Spearman correlation coefficients can detect monotonic relationships, however in this case the Spearman correlation values are similarly low.

We are tasked with fitting four regression models to the above data and compare their prediction performance. Then we will select the best performing model and use it to predict the response of 4 unseen data points. The measure of performance that will be used for comparison is the Mean Square error which is defined as:

$$\text{MSE} = \frac{1}{n}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \tag{1}$$

Before fitting, our data will be split into training and testing sets. The training set will consist of 80% of the data, and the testing sets will contain the remaining 20%. The models will first be fit to the training set and performance comparison will be carried out on the testing set. This means performance comparison will be conducted on unseen data and so should represent the models performance on new data points, provided they come from the same original population. After comparison, the final model will be fit to the entire data set and then used to predict the 4 new data points.

The first model to be fitted is a simple linear model. The simple linear model is of the form:

$$y_i = \beta_0 + \beta_1 x_{i,1} + \cdots + \beta_5 x_{i,5} + \epsilon_i \quad i = 1, ..., n$$

where $\epsilon \sim N(0, \sigma^2)$

The simple linear model is fit using the ordinary least squares approach. This approach aims to minimise the following loss function:

$$L(\boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$$

which using calculus, results in the following optimal value of $\boldsymbol{\beta}$:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{2}$$

This equation is used to construct the coefficients for the simple linear model. Based on the data exploration conducted above, it is unlikely that a simple linear model will model this data well. This is due to, with the exception of x_1, the very weak Pearson Correlation seen between the features and the response. Therefore, β_0x and β_1 will be the main contributors to this model.

The second model to be fitted is an extension of the simple linear model. A regularisation term will be added to the loss function which will restrict the values which $\hat{\boldsymbol{\beta}}$ can take. The new loss function will be of the form:

$$L(\boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta_{-0}}\|_2^2 \tag{3}$$

where $\boldsymbol{\beta_{-0}}$ denotes the vector of coefficients excluding the intercept term. The intercept term is excluded as described in Section 3.4.1 of The Elements of Statistical Learning by Hastie et al.

This modified loss function is minimised by the following value of $\boldsymbol{\beta}$:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{4}$$

where $\boldsymbol{I}$ is a modified identity matrix with a 0 in the first diagonal and the ridge parameter λ is as a positive constant. The identity is modified to allow for the lack of regularization on the intercept term as mentioned above.

The ridge parameter λ controls the regularisation and the coefficients of the ridge regression shrink towards 0 as $\lambda \to \infty$. It should be noted that the coefficients can never be shrunk to exactly 0. This regularisation term is usually employed to reduce multi-collinearity between the features seen in the design matrix $\boldsymbol{X}$. However, it can also be used to prevent over-fitting to the training data, which will be it's purpose in this case.

The ridge parameter will be tuned using two different optimisation methods with k-fold cross validation. The first optimisation method will be using a grid search of possible values of λ . For each value of λ in the grid, the mean MSE across the k folds will be calculated. The value of λ which minimises the mean MSE across the k folds will be chosen as the optimal value. The second approach, although similar, will use an optimizer to search for the value of λ which minimises the mean MSE across the k folds. This should produce a more accurate and optimal value of λ though may be more prone to over fitting.

The third model to be fitted will be a quadratic model containing all quadratic interaction terms:

$$y_i = \beta_0 + \beta_1\, x_{i,1} + \cdots + \beta_5\, x_{i,5} + \beta_6\, x_{i,1}^2 + \cdots + \beta_{10}\, x_{i,5}^2 + \beta_{11}\, x_{i,1}\, x_{i,2} + \cdots + \beta_{20}\, x_{i,4}\, x_{i,5} + \epsilon_i \quad i = 1, ..., n$$

where $\epsilon \sim N(0, \sigma^2)$.

The loss function is unchanged and so the optimal value of $\hat{\boldsymbol{\beta}}$ remains as in Eq. (2). To account for the quadratic interaction terms the design matrix, $\boldsymbol{X}$, has corresponding columns added which have the relevant transformations applied. This model will be very prone to over fitting as the number of features, 21, is quite large relative to the number of data points in the training set, 40. A model which has the number of features equal to the number of data points will predict the training set perfectly but fail to generalise to unseen data. Therefore, the quadratic model should preform well on the training set but have relatively poor performance on the test set.

The fourth model to be fitted will be a quadratic ridge regression model. This model will have the form seen for the quadratic model but will use the loss function as in Eq. (3). Therefore, the optimal value of $\hat{\boldsymbol{\beta}}$ remains as in Eq.(4). The optimal ridge parameter is found using both methods as described for the linear ridge regression model.

Fitting these models to the training data set generates the follow results:

As can be seen in Table 4, we first note that the difference between the grid search and direct optimisation of the ridge parameter, λ, is negligible. This means either can be used, provided

Coefficients	Linear Model	Linear Ridge Optimised	Linear Ridge Grid-Search	Quadratic model	Quadratic Ridge Optimised	Quadratic Ridge Grid-Search
Intercept	5.4406	7.4711	7.4746	0.0299	1.3938	1.4064
x_1	4.4898	0.7266	0.7187	-0.858	-0.0909	-0.0862
x_2	0.6862	-0.0205	-0.0205	-0.7187	-0.2706	-0.266
x_3	0.0294	0.0493	0.0488	0.9392	0.1008	0.0999
x_4	-1.2633	-0.1044	-0.1032	-0.5851	-0.3248	-0.322
x_5	1.8304	0.1756	0.1735	-0.2599	0.0335	0.0363
x_1^2	0	0	0	1.1451	0.9915	0.9913
x_2^2	0	0	0	-0.153	0.0498	0.0504
x_3^2	0	0	0	-0.0238	-0.0418	-0.0418
x_4^2	0	0	0	0.586	0.0762	0.0746
x_5^2	0	0	0	1.0231	0.9531	0.9497
$x_1\,x_2$	0	0	0	0.3908	0.4664	0.4633
$x_1\,x_3$	0	0	0	0.4015	0.3005	0.2987
$x_1\,x_4$	0	0	0	1.4239	0.2692	0.263
$x_1\,x_5$	0	0	0	2.4512	1.7015	1.6862
$x_2\,x_3$	0	0	0	-2.2695	-0.8301	-0.8209
$x_2\,x_4$	0	0	0	1.4299	0.555	0.5484
$x_2\,x_5$	0	0	0	-0.3754	0.1059	0.1065
$x_3\,x_4$	0	0	0	2.0805	0.2361	0.2278
$x_3\,x_5$	0	0	0	-0.5787	-0.0421	-0.0414
$x_4\,x_5$	0	0	0	0.402	0.156	0.1557
Ridge Parameter	0	641.5165	650	0	48.7167	50
MSE (train)	97.0559	147.2619	147.4635	12.9594	18.202	18.3223
MSE (test)	141.0924	43.4943	43.3992	65.8084	13.5647	13.4977

Table 2: Coefficients of models fitted

sufficient coverage of the grid being searched. From here on, when referring to ridge regression, the direct optimisation procedure will be used as it does not require a range of potential values. Looking at the MSE of the ridge regression models, the MSE was higher on the training data but lower on the testing data than that of the non-regularised models. This is an indicator that the regularisation was successful in preventing over-fitting. This can been seen more clearly by examining the differences between the first and the second model. With a ridge parameter of 641, the linear ridge model has shrunk all of the non intercept terms towards 0. This will prevent the over-fitting to the high leverage points mention earlier in the initial data exploration. The large intercept term is intuitive due to the very weak correlation coefficients as mentioned previously.

Surprisingly, the linear ridge model has a much larger ridge parameter than the quadratic ridge model. This is probably due to quadratic interaction effects modelling non-linear trends in the data. This would need to be confirmed, potentially graphically using a pairs plot. Due to the very low MSE on the training set, we can conclude that a quadratic model fits the data best. However, as we wish to pick the best model for prediction purposes rather than inference, a quadratic ridge model is most appropriate. This is due to the lower MSE seen on the test set when compared with the non-ridge quadratic model.

Therefore, we refit the quadratic ridge model using the direct optimisation of λ to the entire data set. This final model is as follows:

Coefficients	Final model
Intercept	1.4806
x_1	-0.0679
x_2	-0.1212
x_3	0.0005
x_4	-0.3546
x_5	-0.0672
x_1^2	0.9612
x_2^2	0.0294
x_3^2	-0.0557
x_4^2	0.0761
x_5^2	1.0775
$x_1\,x_2$	0.4825
$x_1\,x_3$	0.4106
$x_1\,x_4$	0.306
$x_1\,x_5$	1.9723
$x_2\,x_3$	-0.7601
$x_2\,x_4$	0.5193
$x_2\,x_5$	0.217
$x_3\,x_4$	0.0814
$x_3\,x_5$	-0.051
$x_4\,x_5$	0.0209
Ridge Parameter	32.53906
MSE (full data)	16.0946

Table 3: Coefficients of final model

Using this model to predict the 4 unseen data points provided:

x_1	x_2	x_3	x_4	x_5	Predicted y
-0.95	1.76	-2.04	0.82	-0.85	6.9109
-1.27	2.58	-1.08	2.96	-1.83	13.2833
-0.17	1.3	0.78	-2.79	-0.28	0.2585
-3.48	0.99	0.78	5.18	1.45	0.1755

Table 4: Coefficients of final model

2 Question 2

We have been presented with a dataset of SMS messages with the aim of creating two Naive Bayes spam detection models. This dataset consists of 44,066 words across 5,572 messages, with a vocabulary length of 7,903 words. The data has been presented in its raw form, a pre-processed form, and a sparse matrix, $\boldsymbol{X}$, with 5,572 rows and 7,903 columns. The x_{ij} entry of this sparse matrix represents the number of occurrences of the jth word in our vocabulary in the ith message in our dataset. Data pre-processing of this form is difficult and time consuming and so we shall

work directly with the provided sparse matrix, $\boldsymbol{X}$. Before we begin the derivations of the Naive Bayes models, we must conduct some initial data exploration. There are 747 spam messages and the remaining 4,825 messages are classified as ham (not spam). Therefore, we are working with an unbalanced dataset and sub-sampling may be used to address this in a smaller dataset. However, it should not be necessary due to the number of messages contained in this dataset. A small number of words occur very frequently in this dataset, with 10%, aprrox 4,400 words, coming from just 16 values in the vocabulary. The most frequently occurring word in the vocabulary occurred 463 times across all messages. This can be compared to the fact that 4,308 words in our vocabulary occur just once in the entire dataset. This leads to an interesting question, whether the sparse matrix should be converted to a dense matrix. This could be done by simply removing all columns which do not occur more than a chosen number or do not occur in more than a chosen number of messages. As just seen, removing vocabulary which occurs only once, reduces our total vocabulary by more than half. Similarly, removing words which do not occur in more than 20 messages, reduces our vocabulary to just 406 with little penalty to performance. This poses a problem of Naive Bayes classification and sparsity. If a word in our vocabulary $\boldsymbol{x_j}$ never occurs in a spam message, then the probability that a message containing that word is classified as spam is exactly 0. This can be dealt with using additive smoothing (Laplace smoothing) or by converting our matrix to a dense form. The easiest way to convert the matrix to a dense form is to remove words from our vocabulary. However, reducing our vocabulary, will reduce future coverage of words in new unseen messages. Including them in this instance does not have a large computational penalty. Therefore, we will implement Laplace smoothing on our Naive Bayes models which will be explained in more detail later. However, if our vocabulary was large enough such that it posed a computational problem, as can be seen by the above examples, reducing the matrix can be implemented with difficulty. Both of the two examples above, had little to no effect on classification performance.

It should be noted that there are 17 rows in $\boldsymbol{X}$ which appear to have no non-zero entries. However, removing these data points has a large negative effect on the performance of any Naive Bayes models fit, so there is some value provided by these rows. It is unclear why these rows are equal to zero, as the messages in the original, and pre-processed data are non empty. If the number of empty rows was higher, it may be beneficial to reconstruct $\boldsymbol{X}$ from the pre-processed dataset. Due to the value being so low relative to the total amount of messages, and the large dip in performance when removed, we will proceed with these rows included.

A Naive Bayes classifier with K classes utilises Bayes theorem to fit to the data which is given as the following:

$$P(c_i|\boldsymbol{x_i}) = \frac{P(c_i)P(\boldsymbol{x_i}|c_i)}{P(\boldsymbol{x_i})} = \frac{P(c_i)P(\boldsymbol{x_i}|c_i)}{\sum_{k=1}^{K} P(\boldsymbol{x_i}|c_i = k)} \tag{5}$$

where $\boldsymbol{x_i}$ is a data point and c_i is the class of the data point.

The Naive Bayes classifier assigns a new data point $\boldsymbol{x^*}$ to the class k which maximises the value seen in Eq. (5). Due to this, we can drop the constant of proportionality seen in the denominator as the posterior will be maximised when the numerator is maximised. Therefore, for the remainder of the report we will be working with:

$$P(c_i = k|\boldsymbol{x_i}) \propto P(c_i = k)P(\boldsymbol{x_i}|c_i = k)$$

In order to fit the Naive Bayes models, we will need to derive the the priors and any parameters the likelihood requires. We will be using the MLE estimates of these values, which we will derive now. The Naive part of Naive Bayes is the assumption of independence of word combinations seen

in messages, i.e.:

$$P(\boldsymbol{x_i}|c_i = k) = \prod_{j=1}^{M} P(x_{ij}|c_i = k)$$

where j indicates the column of $\boldsymbol{x_i}$, and M is the length of our vocabulary.

This assumption although naive makes Naive Bayes a powerful classifier, though it is not a good assumption for modelling the data. This assumption leads to:

$$\begin{aligned} P(\boldsymbol{x_i}, c_i) &= \prod_{k=1}^{K} \left[P(c_i = k) P(\boldsymbol{x_i}|c_i = k) \right] \\ &= \prod_{k=1}^{K} \left[P(c_i = k) \prod_{j=1}^{M} P(x_{ij}|c_i = k) \right] \\ &= \prod_{k=1}^{K} P(c_i = k) \prod_{k=1}^{K} \prod_{j=1}^{M} P(x_{ij}|c_i = k) \end{aligned} \tag{6}$$

Using indicator functions, which will be denoted by $\mathbb{1}$, and Eq. (6), we can derive the likelihood of our data:

$$P(\mathcal{D}) = \prod_{i=1}^{N} \left[\prod_{k=1}^{K} P(c_i = k)^{\mathbb{1}\{c_i = k\}} \prod_{k=1}^{K} \left(\prod_{j=1}^{M} P(x_{ij}|c_i = k) \right)^{\mathbb{1}\{c_i = k\}} \right]$$

We then take the log of the likelihood to construct the log likelihood as:

$$\log(P(\mathcal{D})) = \sum_{i=1}^{N} \left[\sum_{k=1}^{K} \log\Big(P(c_i = k)\Big) \mathbb{1}\{c_i = k\} \sum_{k=1}^{K} \left(\sum_{j=1}^{M} \mathbb{1}\{c_i = k\} \log\Big(P(x_{ij}|c_i = k)\Big) \right) \right]$$

where N is the number of data points.

Two Naive Bayes classifiers will be fit to this data. The first of which will be a Bernoulli Naive Bayes classifier and the second will be a Multinomial Naive Bayes Classifier. The difference between both of these models is the assumed form of the data and the assumed form of the probability distribution of data points. First, shall look at the Bernoulli model and derive the relevant MLE estimates. The Bernoulli model assumes that $x_{ij} \in \{0, 1\}$, i.e. the data represents the jth word in our vocab occurring at least once in message i and that:

$$P(x_{ij}|c_i = k) \propto (\theta_{jk})^{x_{ij}} (1 - \theta_{jk})^{1 - x_{ij}} \tag{7}$$

where θ_{jk} is the probability of a success, in this case the jth word occurring at least once in a message of class k.

We wish to derive the MLE estimates for a general class k and so will drop the summation over k for the following steps: Subbing Eq. (7) into the log likelihood we get:

$$\begin{aligned} \log(P(\mathcal{D}|c_i = k)) &= \sum_{i=1}^{N} \left[\log\Big(P(c_i = k)\Big) \mathbb{1}\{c_i = k\} \left(\mathbb{1}\{c_i = k\} \log\Big((\theta_{jk})^{x_{ij}} (1 - \theta_{jk})^{1 - x_{ij}}\Big) \right) \right] \\ &= \sum_{i=1}^{N} \left[\log\Big(P(c_i = k)\Big) \mathbb{1}\{c_i = k\} \left(\mathbb{1}\{c_i = k\} \Big(x_{ij} \log(\theta_{jk}) + (1 - x_{ij}) \log(1 - \theta_{jk}) \Big) \right) \right] \end{aligned}$$

It is clear we need MLE estimates for θ_{jk} and $P(c_i = k)$. For the estimation of $P(c_i = k)$, we will need to use Lagrangian multipliers to solve subject to the constraint: $\sum_{k=1}^{K} P(c_i = k) = 1$. First, for θ_{jk} we will take the derivative of the log likelihood and set equal to 0:

$$\begin{aligned}
\frac{\partial \log(P(\mathcal{D}|c_i = k))}{\partial \theta_{jk}} &= \sum_{i=1}^{N} \left(\mathbb{1}\{c_i = k\} \Big(\frac{x_{ij}}{\theta_{jk}} - \frac{1 - x_{ij}}{1 - \theta_{jk}} \Big) \right) = 0 \\
&= \sum_{i=1}^{N} \left(\mathbb{1}\{c_i = k\} \Big(x_{ij} - \theta_{jk} \Big) \right) = 0 \\
\hat{\theta}_{jk} = \frac{\sum_{i=1}^{N} \mathbb{1}\{c_i = k\} x_{ij}}{\sum_{i=1}^{N} \mathbb{1}\{c_i = k\}} &= \frac{\text{Number of messages containing vocab } j \text{ in Class k}}{\text{Number of messages in Class k}}
\end{aligned} \tag{8}$$

Using Lagrangian multipliers for $P(c_i = k)$ with the constraint $\sum_{k=1}^{K} P(c_i = k) = 1$, we get the follow function:

$$\Lambda = \sum_{i=1}^{N} \log \Big(P(c_i = k) \Big) \mathbb{1}\{c_i = k\} - \lambda \left(\sum_{k=1}^{K} P(c_i = k) - 1 \right)$$

This leads to:

$$\begin{aligned}
\frac{\partial \Lambda}{\partial P(c_i = k)} &= \frac{\sum_{i=1}^{N} \mathbb{1}\{c_i = k\}}{P(c_i = k)} - \lambda = 0 \\
\hat{P}(c_i = k) &= \frac{\sum_{i=1}^{N} \mathbb{1}\{c_i = k\}}{\lambda}
\end{aligned}$$

We solve for the Lagrangian multiplier using the constraint:

$$\begin{aligned}
\sum_{k=1}^{K} \hat{P}(c_i = k) = \frac{\sum_{i=1}^{N} \mathbb{1}\{c_i = k\}}{\lambda} &= 1 \\
\lambda = \sum_{i=1}^{N} \mathbb{1}\{c_i = k\} &= N
\end{aligned}$$

Therefore, the MLE estimate of the prior is:

$$\hat{P}(c_i = k) = \frac{\sum_{i=1}^{N} \mathbb{1}\{c_i = k\}}{N} = \frac{\text{Number of messages in Class k}}{\text{Total number of messages}} \tag{9}$$

And thus concludes the derivation of the MLE estimates for the Bernoulli model given by Eq. (8), and Eq. (9). The Multinomial MLE estimates are calculated in a similar manner. However, both require use of Lagrangian multipliers. The MLE estimate for the prior is the same as that of the Bernoulli model. Therefore, we are only required to derive the new MLE estimate for θ_{jk}.

The Multinomial model assumes that $x_{ij} \in \{0, 1, \dots\}$, i.e. the data represents the number of occurrences of the jth word in our vocab in message i and that:

$$P(x_{ij}|c_i = k) \propto \prod_{j=1}^{M} (\theta_{jk})^{x_{ij}} \tag{10}$$

where θ_{jk} is the probability of a success, in this case the jth word occurring x_{jk} times in a message of class k.

We wish to derive the MLE estimates for a general class k and so will drop the summation over k for the following steps: Subbing Eq. (10) into the log likelihood, we construct a Lagrangian function to solve subject to the constraint: $\sum_{j=1}^{M} \theta_{jk} = 1$:

$$\Lambda = \sum_{i=1}^{N} \mathbb{1}\{c_i = k\} \left(\sum_{j=1}^{M} x_{ij} \log(\theta_{jk}) \right) - \lambda \left(\sum_{j=1}^{M} \theta_{jk} - 1 \right)$$

This leads to:

$$\frac{\partial \Lambda}{\partial \theta_{jk}} = \sum_{i=1}^{N} \mathbb{1}\{c_i = k\} \left(\frac{x_{ij}}{\theta_{jk}} \right) - \lambda = 0$$

$$\hat{\theta}_{jk} = \frac{\sum_{i=1}^{N} \mathbb{1}\{c_i = k\} x_{ij}}{\lambda}$$

We solve for the Lagrangian multiplier using the constraint:

$$\sum_{j=1}^{M} \hat{\theta}_{jk} = \frac{\sum_{j=1}^{M} \sum_{i=1}^{N} \mathbb{1}\{c_i = k\} x_{ij}}{\lambda} = 1$$

$$\lambda = \sum_{j=1}^{M} \sum_{i=1}^{N} \mathbb{1}\{c_i = k\} x_{ij}$$

Therefore, the MLE estimate for θjk is:

$$\hat{\theta}_{jk} = \frac{\sum_{i=1}^{N} \mathbb{1}\{c_i = k\} x_{ij}}{\sum_{j=1}^{M} \sum_{i=1}^{N} \mathbb{1}\{c_i = k\} x_{ij}} = \frac{\text{Total occurrences of vocab } j \text{ in messages of class k}}{\text{Total number of words in messages of class k}} \tag{11}$$

And thus concludes the derivation of the MLE estimates for the Multinomial model given by Eq. (11), and Eq. (9).

Fitting both of these models is a simple case of calculating both the priors and the thetas using simple count and sum functions according to the relevant MLE estimates for each class, spam and ham. As a final note, we shall discuss Laplacian smoothing as a method to overcome the sparsity of the matrix $\boldsymbol{X}$. Laplacian smoothing requires the addition of 1 to the numerator of the MLE estimate of θ_{jk} and the addition of M to the denominator. This provides a base probability of message of class k containing a word that has never occurred in a message of class k before. This base probability is equal to $\frac{1}{M}$.

To measure both models performance on unseen data, we split the given data set into training and testing sets. 80% of the data, 4,458 messages are placed into a training set and the remaining 1,114 are placed into a test set. Both models are fit to the training set by using the data points in the training set to calculate the MLE estimates. We then predict the classification of the data points in the test set. As mentioned before, the class predicted for a data point is the class which maximises the posterior of that data point. To avoid computational errors, we in fact calculate the log posterior. Classification is carried out for both models on all points in the test sets and a confusion matrix is constructed using the known true classes of the test data points. In this example, we have taken ham as a positive response encoded to class 0, and spam as a negative response, encoded to class 1. From the confusion matrix, we can calculate a range of metrics to

compare the models. As it is more important to correctly classify ham messages, as incorrectly classifying a ham message as spam could cause important messages to not be delivered, we will focus on the sensitivity of the classifiers as our main point of comparison. The sensitivity is calculated as the total number of correctly predicted ham messages divided by the total number of actual ham messages.

The confusion matrix for the Bernoulli Naive Bayes classifier is as follows:

Predicted \Actual	Ham	Spam
Ham	957	14
Spam	11	132

Table 5: Confusion Matrix for Bernoulli NB

As can be seen in Table 5, the Bernoulli Naive Bayes correctly classified 957 Ham messages out of a possible 968. This leads to a sensitivity of 0.9886. It incorrectly allowed 14 Spam messages through. The confusion matrix for the Multinomial Naive Bayes classifier is as follows:

Predicted \Actual	Ham	Spam
Ham	956	7
Spam	12	139

Table 6: Confusion Matrix for Multinomial NB

As can be seen in Table 6, the Multinomial Naive Bayes correctly classified 956 Ham messages out of a possible 968. This leads to a sensitivity of 0.9876. The Bernoulli model only correctly classified 1 more Ham message than the Multinomial. But the Multinomial model stopped 7 more spam messages. It can be argued that the single extra Ham message that the Bernoulli model classified correctly is not worth the 7 Spam messages it missed. In this case, due the minute difference in sensitivity, the overall accuracy of the classifiers as our metric. The accuracy of the Bernoulli model was 0.9776 vs the Multinomial's 0.9829. Therefore, we shall choose the Multinomial Naive Bayes classifier as our model of choice.