Problem 2

a) Naïve bayes model only matches existence of words in the train data and classifies an unseen document into a category using conditional probability and naïve bayes theorem. For a text classification problem, the main focus should be on the key words and sentiment of the text. For example: In texts: 'the' occurs the maximum in category 'Sports' in train data and based on multinomial naïve bayes model created, an unseen data with more 'the' may have a tendency to be classified to category 'Sports'. The precedence of 'the' and 'soccer' in such a situation is same. The (pi) product of all such conditional probabilities account are invariant to order and hence rank.

b) For creating binomial naïve bayes model, after loading the train and test data; I first created a binomial frequency sparse matrix using the following transformation on train and test data:
Train data:

```
# Get a sparse data matrix X (rows: training exmaples, columns: # of occurrences
bin = ifelse(X_data[,3] >=1, 1, 0)
X_binomial = sparseMatrix(x=bin, i=X_data[,1], j=X_data[,2])
X = sparseMatrix(x=X_data[,3], i=X_data[,1], j=X_data[,2])
```

Test data:

```
X_data_binomial = ifelse(X_data_t[,3] >=1, 1, 0)
X_test_binomial = sparseMatrix(x=X_data_binomial, i=X_data_t[,1], j=X_data_t[,2])
X_test = sparseMatrix(x=X_data_t[,3], i=X_data_t[,1], j=X_data_t[,2])
```

For training the model on the basis of MLE. Calculating the priori probability as follows

```
> prop.table(freq)
Y
   1    2    3    4
0.25 0.25 0.25 0.25
```

Since priori probability is the same for each class, we need to worry about it in our further calculations. Just as we do not consider p(x_1, x_2, ….. x_n) in the denominator while calculating naïve bayes probability.
Next, conditional probabilities of each word per category of class was calculated using:

```
#p(x_j / y= k)
#Selecting rows which belong to respective categories
art_1 <- which(Y==1)
art_2 <- which(Y==2)
art_3 <- which(Y==3)
art_4 <- which(Y==4)

#Articles for Category 1
#Selecting the part of matrix for each category:
art_X1 <- X_binomial[art_1,]
#Calculating sum of existence (1/0) of each word given the category:
c1 = colSums (art_X1, na.rm = FALSE, dims = 1)
c1 = c1/1000

#Articles for Category 2
#Selecting the part of matrix for each category:
art_X2 <- X_binomial[art_2,]
#Calculating sum of existence (1/0) of each word given the category:
c2 = colSums (art_X2, na.rm = FALSE, dims = 1)
c2 = c2/1000

#Articles for Category 3
#Selecting the part of matrix for each category:
art_X3 <- X_binomial[art_3,]
#Calculating sum of existence (1/0) of each word given the category:
c3 = colSums (art_X3, na.rm = FALSE, dims = 1)
c3 = c3/1000

#Articles for Category 4
#Selecting the part of matrix for each category:
art_X4 <- X_binomial[art_4,]
#Calculating sum of existence (1/0) of each word given the category:
c4 = colSums (art_X4, na.rm = FALSE, dims = 1)
```

Conditional probabilities for each word, given categorical class are as follows:

Category 1:

```
> #Displaying conditional probabilities:
> c1
  [1] 0.022 0.084 0.000 0.028 0.021 0.058 0.013 0.004 0.138 0.000 0.009 0.727 0.002 0.017
 [15] 0.000 0.368 0.000 0.006 0.003 0.002 0.000 0.000 0.755 0.001 0.192 0.000 0.394 0.121
 [29] 0.888 0.773 0.048 0.068 0.857 0.000 0.070 0.000 0.009 0.010 0.000 0.003 0.006 0.708
 [43] 0.019 0.292 0.017 0.000 0.003 0.550 0.097 0.000 0.482 0.543 0.003 0.033 0.048 0.024
 [57] 0.000 0.000 0.000 0.734 0.000 0.002 0.000 0.010 0.002 0.092 0.075 0.008 0.006 0.003
 [71] 0.034 0.481 0.294 0.000 0.000 0.128 0.016 0.000 0.013 0.305 0.671 0.066 0.091 0.064
 [85] 0.030 0.023 0.027 0.005 0.016 0.000 0.004 0.021 0.024 0.000 0.002 0.012 0.000 0.000
```

Category 2:

```
> c2
  [1] 0.010 0.066 0.000 0.002 0.008 0.121 0.010 0.003 0.019 0.000 0.006 0.798 0.003 0.038
 [15] 0.012 0.355 0.002 0.004 0.000 0.002 0.014 0.010 0.826 0.000 0.221 0.000 0.472 0.048
 [29] 0.956 0.858 0.137 0.010 0.864 0.000 0.017 0.000 0.001 0.003 0.000 0.003 0.038 0.743
 [43] 0.001 0.376 0.038 0.000 0.038 0.671 0.159 0.175 0.498 0.543 0.031 0.022 0.008 0.028
 [57] 0.001 0.000 0.020 0.715 0.000 0.000 0.006 0.023 0.000 0.140 0.175 0.027 0.010 0.009
 [71] 0.052 0.446 0.301 0.002 0.011 0.077 0.017 0.000 0.018 0.473 0.681 0.047 0.217 0.189
 [85] 0.018 0.076 0.024 0.022 0.016 0.000 0.001 0.016 0.010 0.000 0.000 0.014 0.000 0.000
```

Category 3:

```
> c3
  [1] 0.002 0.055 0.000 0.002 0.002 0.255 0.002 0.005 0.004 0.000 0.003 0.755 0.001 0.032
 [15] 0.001 0.336 0.002 0.002 0.000 0.007 0.000 0.000 0.776 0.001 0.197 0.000 0.475 0.022
 [29] 0.936 0.849 0.069 0.004 0.825 0.000 0.035 0.003 0.002 0.000 0.002 0.003 0.017 0.618
 [43] 0.001 0.291 0.022 0.000 0.026 0.569 0.241 0.000 0.515 0.503 0.012 0.029 0.010 0.009
 [57] 0.000 0.002 0.002 0.681 0.000 0.001 0.001 0.041 0.000 0.182 0.115 0.062 0.016 0.027
 [71] 0.021 0.370 0.221 0.000 0.016 0.042 0.032 0.000 0.035 0.204 0.672 0.065 0.350 0.254
```

Category 4:

```
> c4
  [1] 0.008 0.090 0.000 0.013 0.012 0.179 0.007 0.020 0.023 0.001 0.002 0.937 0.037 0.034
 [15] 0.065 0.579 0.050 0.006 0.001 0.006 0.005 0.001 0.891 0.001 0.340 0.000 0.680 0.036
 [29] 0.961 0.923 0.234 0.079 0.924 0.000 0.045 0.006 0.000 0.010 0.000 0.008 0.027 0.776
 [43] 0.000 0.345 0.049 0.022 0.011 0.671 0.429 0.019 0.550 0.616 0.013 0.058 0.047 0.044
 [57] 0.000 0.000 0.008 0.812 0.000 0.001 0.000 0.028 0.001 0.081 0.476 0.018 0.013 0.006
 [71] 0.090 0.459 0.319 0.000 0.012 0.103 0.004 0.000 0.021 0.357 0.752 0.034 0.494 0.166
```

(Each class has 51949 probabilities because there are 51949 words in the train data)

Then, a binomial naïve bayes theorem was constructed using the following code:

```
#Naive Bayes Model:
existence_per_article = apply(X_binomial, 1, function(x) {return(which(x==1))})
str(existence_per_article)
rm(r)
result <- vector()
for(i in 1:4000)
{
  ind = unlist(existence_per_article[i])
  p1 = prod(c1[ind])
  p2 = prod(c2[ind])
  p3 = prod(c3[ind])
  p4 = prod(c4[ind])
  r = which.max(c(p1, p2, p3, p4))
  result = c(result, r)
}

result
classes = as.factor(result)

#Classification on training data:
confusionMatrix(as.factor(Y), classes)
```

The following is the classification on training data:

```
> confusionMatrix(as.factor(Y), classes)
Confusion Matrix and Statistics

          Reference
Prediction   1   2   3   4
         1 999   0   0   1
         2  61 932   3   4
         3  87   2 910   1
         4 200   0   0 800

Overall Statistics

               Accuracy : 0.9102
                 95% CI : (0.901, 0.9189)
    No Information Rate : 0.3368
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8803

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            0.7416   0.9979   0.9967   0.9926
Specificity            0.9996   0.9778   0.9708   0.9374
Pos Pred Value         0.9990   0.9320   0.9100   0.8000
Neg Pred Value         0.8840   0.9993   0.9990   0.9980
Prevalence             0.3367   0.2335   0.2283   0.2015
Detection Rate         0.2497   0.2330   0.2275   0.2000
Detection Prevalence   0.2500   0.2500   0.2500   0.2500
Balanced Accuracy      0.8706   0.9878   0.9838   0.9650
```

Training Accuracy: 91.02%

c) Then Laplace smoothening was performed before predicting on the unseen data, given the possibility of unknown words. The following is the classification result of Laplace smoothening Bernoulli naïve bayes model on the training data:

```
> confusionMatrix(as.factor(Y), classes)
Confusion Matrix and Statistics

          Reference
Prediction   1   2   3   4
         1 981   1   0  18
         2  63 854   1  82
         3  88   2 887  23
         4 200   0   0 800

Overall Statistics

               Accuracy : 0.8805
                 95% CI : (0.87, 0.8904)
    No Information Rate : 0.333
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.8407

 Mcnemar's Test P-Value : < 0.00000000000000022

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            0.7365   0.9965   0.9989   0.8667
Specificity            0.9929   0.9535   0.9637   0.9350
Pos Pred Value         0.9810   0.8540   0.8870   0.8000
Neg Pred Value         0.8830   0.9990   0.9997   0.9590
Prevalence             0.3330   0.2142   0.2220   0.2308
Detection Rate         0.2452   0.2135   0.2218   0.2000
Detection Prevalence   0.2500   0.2500   0.2500   0.2500
Balanced Accuracy      0.8647   0.9750   0.9813   0.9009
```

Training Accuracy: 88.05%

Classification on test data:

```
> confusionMatrix(as.factor(Y_test), test_classes)
Confusion Matrix and Statistics

          Reference
Prediction   1   2   3   4
         1 493   2   2 103
         2  32 416   1 151
         3  59   0 456  85
         4  91   1   2 506

Overall Statistics

               Accuracy : 0.7796
                 95% CI : (0.7625, 0.796)
    No Information Rate : 0.3521
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7061

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            0.7304   0.9928   0.9892   0.5988
Specificity            0.9380   0.9071   0.9257   0.9395
Pos Pred Value         0.8217   0.6933   0.7600   0.8433
Neg Pred Value         0.8989   0.9983   0.9972   0.8117
Prevalence             0.2812   0.1746   0.1921   0.3521
Detection Rate         0.2054   0.1733   0.1900   0.2108
Detection Prevalence   0.2500   0.2500   0.2500   0.2500
Balanced Accuracy      0.8342   0.9500   0.9574   0.7692
```

Testing Accuracy: 77.97%

d) Multinomial naïve bayes model results using created algorithm:

Train data results:

```
> confusionMatrix(as.factor(Y), classes)
Confusion Matrix and Statistics

          Reference
Prediction   1   2   3   4
         1 983   1   1  15
         2 121 806   1  72
         3 171   2 807  20
         4 313   0   0 687

Overall Statistics

               Accuracy : 0.8208
                 95% CI : (0.8085, 0.8325)
    No Information Rate : 0.397
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.761

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            0.6190   0.9963   0.9975   0.8652
Specificity            0.9930   0.9392   0.9395   0.9024
Pos Pred Value         0.9830   0.8060   0.8070   0.6870
Neg Pred Value         0.7983   0.9990   0.9993   0.9643
Prevalence             0.3970   0.2023   0.2023   0.1985
Detection Rate         0.2457   0.2015   0.2018   0.1718
Detection Prevalence   0.2500   0.2500   0.2500   0.2500
Balanced Accuracy      0.8060   0.9677   0.9685   0.8838
```

Training accuracy: 82.08%

Test data:

```
> confusionMatrix(as.factor(Y_test), classes)
Confusion Matrix and Statistics

          Reference
Prediction   1   2   3   4
         1 467   2   2 129
         2  18 392   1 189
         3  28   1 429 142
         4  65   2   2 531

Overall Statistics

               Accuracy : 0.7579
                 95% CI : (0.7403, 0.7749)
    No Information Rate : 0.4129
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6772

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            0.8080   0.9874   0.9885   0.5358
Specificity            0.9270   0.8962   0.9130   0.9510
Pos Pred Value         0.7783   0.6533   0.7150   0.8850
Neg Pred Value         0.9383   0.9972   0.9972   0.7444
Prevalence             0.2408   0.1654   0.1808   0.4129
Detection Rate         0.1946   0.1633   0.1787   0.2213
Detection Prevalence   0.2500   0.2500   0.2500   0.2500
Balanced Accuracy      0.8675   0.9418   0.9508   0.7434
```

Testing accuracy: 75.79%

e) Our dataset has small text articles, with lesser number of words and hence existence/ non-existence of words (binomial) model works better than the frequency of words (multinomial) model. When comparing Bernoulli and Multinomial model, in general: both of their respective pros and cons. Bernoulli model will work better with short documents because the frequency of one word is not much and hence may/ may not affect the decision. Whereas, on the other hand; multinomial model will work better if we happen to have a word occur multiple times.