Problem 5

(a) Training data was already loaded in the code given. Training data is as follows:

```
> X
4000 x 51949 sparse Matrix of class "dgCMatrix"

 [1,] . . . . . . . . . . 2 6 1 . . . . . . . . . 12 . 1 . 2 1 8 4 . . 2 . . . . 1 . . . . . . . . . 1 1 . . . ......
 [2,] . . . . . . . . . . . 2 . . . . . . . . . . 2 . . . . . 9 1 . . 2 . . . . . . . . 1 . . . . . 1 . . . . ......
 [3,] . . . . . . . . . . . 1 . . . 1 . . . . . .   . . . . . . 2 . . . . . . . . . . . 1 . . . . . . . . . . ......
 [4,] . . . . . . . . 2 . . 1 . . . 1 . . . . . . 1 . . . . . 4 1 . . 2 . 1 . . . . . . 1 . 1 . . . . . . 1 . ......
 [5,] . . . . . . . . . . . 1 . . . . . . . . . . 5 . 2 . 1 . 4 3 . . 4 . . . . . . . . 2 . 1 . . . 4 . . . 1 ......
 [6,] . . . . . . . . . . . . . . . . . . . . . . 2 . . . . . . 2 . . 1 . . . . . . . . 1 . . . . . 1 . . 1 . ......
 [7,] . . . . . . . . 1 . . . . . . 2 . . . . . . 3 . . . . . 5 2 . . 3 . . . . . . . . 2 . . . . . 2 . . 2 1 ......
 [8,] . . . . . . . . 1 . . 1 . . . . . . . . . . 2 . . . . . 3 1 . . 1 . . . . . . . . 3 . 1 . . . . . . . . ......
 [9,] . . . 3 . . . . . . . 5 . . . . . . . . . . 2 . . . 2 1 6 2 . . 2 . . . . . . . . 1 . . 1 . . 1 . . . 1 ......
[10,] . . . . . 1 . . 2 . . 5 . . . . . . . . . . 6 . 1 . 1 1 6 5 . . 6 . . . . . . . . 1 . . . . . 2 . . . 1 ......

 ..............................
 ........suppressing 51897 columns and 3981 rows in show(); maybe adjust 'options(max.print= *, width = *)'
 ..............................

[3992,] . 1 . . . . . . . . . . 4 . . . 1 . . . . . . . 4 . . . 1 .  5 5 . . 10 . . . . . . . . . 2 . . . . . 1 2 . 1 1 ......
[3993,] . . . . . . . . . . . . 9 . . 4 . . . . . . . . 6 . . . . .  9 2 . .  8 . . . . . . . . . 2 . . . . . . 1 . . 1 ......
[3994,] . . . . . . . . . . . 12 . . 4 2 . . . . . . . 6 . 2 . 4 .  6 5 . .  8 . . . . . . . . . 2 . . . . . 1 2 . 1 2 ......
[3995,] . 2 . . . . . . . . . . 6 . . . . . . . . . . . 1 . . . . . 10 3 . .  4 . . . . . . . . . 1 . 1 . . . . . . . . ......
[3996,] . . . . . . . . . . . . 1 . . 2 3 . . . . . . . 1 . . . 2 .  3 2 . .  5 . . . . . . . . . 1 . . . . . 2 1 . 1 . ......
[3997,] . . . . . . . . . . . . 3 . . . . . . . . . . . 3 . . . . .  5 2 . .  4 . . . . . . . . . . . . . . . . . . 1 ......
[3998,] . . . . . . . . . . . . 2 . . . . . . . . . . . 6 . 1 . 3 . 11 5 3 .  5 . . . . . . . . . 2 . 1 . . . . . . 1 4 ......
[3999,] . . . . . . . . . . . . 5 . . 4 . . . . . . . . 4 . . . 1 .  2 1 . .  7 . . . . . . . . . 7 . . . . . . . . . 2 ......
[4000,] . . . 1 . . . . . . . . 4 . . . 4 . . . . . . . 5 . 1 . 6 . 10 4 4 .  8 . . . . . . . . . 4 . 1 . . . 1 1 . 1 3 ......
>
```

Loaded the test data in an exactly similar manner. It looks like below:

```
> X_Test
2400 x 60636 sparse Matrix of class "dgCMatrix"

 [1,] . . . . . . . . . . . 1 . . . . . . . . . . 1 . . . . .  1 2 . . 7 . 1 . . . . . . . . . 2 . . . . . . . . . . ......
 [2,] . . . . . . . . . . . 1 . . . 1 . . . . . . 4 . . . 2 . 12 1 . . 6 . . . . . . . . . 2 . . . . . 1 . . 2 2 ......
 [3,] . . . . . . . . . . . 3 . . . . . . . . . . 1 . . . . .  1 1 . . 3 . . . . . . . . . 3 . . . . . 1 . . . . ......
 [4,] . . . . . . . . 1 . . 3 . . . . . . . . . . 2 . . . 3 .  6 2 . . 3 . . . . . . . . . 3 . 1 . . . . . . 1 . ......
 [5,] 1 1 . . . . . . 2 . . 3 . . . 1 . . . . . . 7 . 1 . 2 .  7 3 . 1 7 . . . . . . . . . 6 . . . . . 3 . . 3 1 ......
 [6,] . . . . . . . . . . . . . . . 2 . . . . . . 2 . . . . .  6 3 . . 7 . . . . . . . . . . . 3 . . . . . . 1 2 ......
 [7,] . . . . . . . . . . . . . . . . . . . . . . 1 . 1 . 1 .  5 1 . . 6 . . . . . . . . . 1 . . . . . 1 . . 1 . ......
 [8,] . . . . . 1 . . . . . . . . . . . . . . . . . . . . . .  2 . . . . . . . . . . . . . . . . . . . . . . . . ......
 [9,] . . . . . . . . . . . 2 . . . . . . . . . . 4 . 3 . . .  9 3 . . 2 . . . . . . . . . 5 . . . . . 1 . . 2 2 ......
[10,] . . . . . . . . . . . . . . . . . . . . . . 2 . . . 1 .  3 1 . . . . . . . . . . . . 2 . . . . . 2 . . 1 . ......

 ..............................
 ........suppressing 60584 columns and 2381 rows in show(); maybe adjust 'options(max.print= *, width = *)'
 ..............................

[2392,] . . . . . . . . . . . .  7 . . . . 1 . . . . . .  5 . . . . . 13  2 . .  7 . . . . . . . . . 2 . . . . . 1 . . . 1 ......
[2393,] . . . . . . . . . . . .  5 . . . 1 . . . . . . . 11 . 1 . 3 . 12  2 . . 11 . . . . . . . . . 5 . 1 . . . 3 1 . 1 2 ......
[2394,] . . . . . . . . . . . .  8 . . . . 1 . . . . . .  6 . . . 1 . 14  4 1 . 10 . . . . . . . . . 3 . . . . . 1 . . . 2 ......
[2395,] . . . . . . . . . . . . 16 . . . . 2 . . . . . . 19 . . . 8 . 15  9 . .  9 . . . . . . . . . 9 . . . . . 1 1 . 2 1 ......
[2396,] . . . . . . . . . . . .  1 . . . 1 . . . . . . .  4 . . . 1 .  3  2 . .  1 . . . . . . . . . 2 . 1 . . . . . . . . ......
[2397,] . . . . . . . . . . . .  . . . . . . . . . . . .  . . . . . .  2  1 . .  . . . . . . . . . . 1 . . . . . . . . . . ......
[2398,] . . . . . . . . . . . . 14 . . . . 2 . . . . . .  7 . . . 4 .  9  8 . .  8 . . . . . . . . . 8 . . . . . 1 1 . 1 1 ......
[2399,] . . . . . . . . . . . . 14 . . . 4 4 . . . . . .  7 . 1 . 7 . 33 12 . .  3 . . . . . . . . . 3 . . . 1 . 2 1 . 4 . ......
[2400,] . . . . . . . . . . . .  4 . . . 1 . . . . . . .  7 . . . 2 . 12  2 . .  6 . . . . . . . . . 1 . . . . . . 1 . . 1 ......
```

(b) Four binary classified SVM models were constructed. For each model, the following steps were taken:
  - converted the target variable to {-1,1} depending on the class of interest
  - a SVM model with linear kernel was constructed
  - the model was used to predict on both train and test data
  - confusion matrix on each are shown

Classifier 1: (Operating Systems)

Train data:

```
> confusionMatrix(pred_1, Y_Model1, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction   -1    1
        -1 3000    0
        1     0 1000

               Accuracy : 1
                 95% CI : (0.9991, 1)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.00
            Specificity : 1.00
         Pos Pred Value : 1.00
         Neg Pred Value : 1.00
             Prevalence : 0.25
         Detection Rate : 0.25
   Detection Prevalence : 0.25
      Balanced Accuracy : 1.00

       'Positive' Class : 1
```

Test data:

```
> confusionMatrix(pred_Test_1, Y_Test_Model1, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction   -1    1
        -1 1783   84
        1    17  516

               Accuracy : 0.9579
                 95% CI : (0.9491, 0.9656)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.8834

 Mcnemar's Test P-Value : 0.00000000005125

            Sensitivity : 0.8600
            Specificity : 0.9906
         Pos Pred Value : 0.9681
         Neg Pred Value : 0.9550
             Prevalence : 0.2500
         Detection Rate : 0.2150
   Detection Prevalence : 0.2221
      Balanced Accuracy : 0.9253

       'Positive' Class : 1
```

Classifier 2: (Vehicles)

Train Data:

```
> confusionMatrix(pred_2, Y_Model2, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction   -1    1
        -1 3000    0
        1     0 1000

               Accuracy : 1
                 95% CI : (0.9991, 1)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.00
            Specificity : 1.00
         Pos Pred Value : 1.00
         Neg Pred Value : 1.00
             Prevalence : 0.25
         Detection Rate : 0.25
   Detection Prevalence : 0.25
      Balanced Accuracy : 1.00

       'Positive' Class : 1
```

Test Data:

```
> confusionMatrix(pred_Test_2, Y_Test_Model2, positi
Confusion Matrix and Statistics

          Reference
Prediction   -1    1
        -1 1751  101
        1    49  499

               Accuracy : 0.9375
                 95% CI : (0.9271, 0.9469)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.8284

 Mcnemar's Test P-Value : 0.00003125

            Sensitivity : 0.8317
            Specificity : 0.9728
         Pos Pred Value : 0.9106
         Neg Pred Value : 0.9455
             Prevalence : 0.2500
         Detection Rate : 0.2079
   Detection Prevalence : 0.2283
      Balanced Accuracy : 0.9022

       'Positive' Class : 1
```

Classifier 3 (Sports)

Train Data:

```
> confusionMatrix(pred_3, Y_Model3, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction   -1    1
        -1 3000    0
        1     0 1000

               Accuracy : 1
                 95% CI : (0.9991, 1)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.00
            Specificity : 1.00
         Pos Pred Value : 1.00
         Neg Pred Value : 1.00
             Prevalence : 0.25
         Detection Rate : 0.25
   Detection Prevalence : 0.25
      Balanced Accuracy : 1.00

       'Positive' Class : 1
```

Test Data :

```
> confusionMatrix(pred_Test_3, Y_Test_Model3, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction   -1    1
        -1 1759   74
        1    41  526

               Accuracy : 0.9521
                 95% CI : (0.9428, 0.9603)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : < 0.000000000000000022

                  Kappa : 0.8698

 Mcnemar's Test P-Value : 0.002845

            Sensitivity : 0.8767
            Specificity : 0.9772
         Pos Pred Value : 0.9277
         Neg Pred Value : 0.9596
             Prevalence : 0.2500
         Detection Rate : 0.2192
   Detection Prevalence : 0.2362
      Balanced Accuracy : 0.9269

       'Positive' Class : 1
```

Classifier 4 (Politics)

Train Data:

```
> confusionMatrix(pred_4, Y_Model4, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction   -1    1
        -1 3000    0
        1     0 1000

               Accuracy : 1
                 95% CI : (0.9991, 1)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.00
            Specificity : 1.00
         Pos Pred Value : 1.00
         Neg Pred Value : 1.00
             Prevalence : 0.25
         Detection Rate : 0.25
   Detection Prevalence : 0.25
      Balanced Accuracy : 1.00

       'Positive' Class : 1
```

Test Data:

```
> confusionMatrix(pred_Test_4, Y_Test_Model4, positiv
Confusion Matrix and Statistics

          Reference
Prediction   -1    1
        -1 1745   85
        1    55  515

               Accuracy : 0.9417
                 95% CI : (0.9315, 0.9507)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : < 0.0000000000000002

                  Kappa : 0.8418

 Mcnemar's Test P-Value : 0.01425

            Sensitivity : 0.8583
            Specificity : 0.9694
         Pos Pred Value : 0.9035
         Neg Pred Value : 0.9536
             Prevalence : 0.2500
         Detection Rate : 0.2146
   Detection Prevalence : 0.2375
      Balanced Accuracy : 0.9139

       'Positive' Class : 1
```

Train and test errors on each are reported:

	Training error	Test Error
Model1	0	0.04250000
Model2	0	0.06250000
Model3	0	0.04791667
Model4	0	0.05833333

All the binary classifiers were combined to form the multi-classifier model. The confusion matrix for the multi-class classifier is as follows:

```
> confusionMatrix(max_label, Y_Test)
Confusion Matrix and Statistics

          Reference
Prediction   1   2   3   4
         1 523  24  15  22
         2  26 513  24  30
         3  28  27 533  34
         4  23  36  28 514

Overall Statistics

               Accuracy : 0.8679
                 95% CI : (0.8537, 0.8812)
    No Information Rate : 0.25
    P-Value [Acc > NIR] : <0.0000000000000002

                  Kappa : 0.8239

 Mcnemar's Test P-Value : 0.5016

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            0.8717   0.8550   0.8883   0.8567
Specificity            0.9661   0.9556   0.9506   0.9517
Pos Pred Value         0.8955   0.8651   0.8569   0.8552
Neg Pred Value         0.9576   0.9519   0.9623   0.9522
Prevalence             0.2500   0.2500   0.2500   0.2500
Detection Rate         0.2179   0.2137   0.2221   0.2142
Detection Prevalence   0.2433   0.2471   0.2592   0.2504
Balanced Accuracy      0.9189   0.9053   0.9194   0.9042
```

The overall error is 1-0.8679 = 0.1321 = 13.21%

(c) Soft margin classifiers with different C values were trained. Using hold out cross-validation on 25% validation data set (of the training data), the following are the training errors on respective binary models and overall model:

```
> training
```

	0.125	0.25	0.5	1	2	4	8	16	32	64	128	256	512
Model1	0.0000000000	0.0000000000	0.0000000000	0	0	0	0	0	0	0	0	0	0
Model2	0.0013333333	0.0003333333	0.0003333333	0	0	0	0	0	0	0	0	0	0
Model3	0.0006666667	0.0006666667	0.0000000000	0	0	0	0	0	0	0	0	0	0
Model4	0.0000000000	0.0000000000	0.0000000000	0	0	0	0	0	0	0	0	0	0
Overall	0.0010000000	0.0006666667	0.0003333333	0	0	0	0	0	0	0	0	0	0

All the values of cost from 1-512 give the best model on training data, where error = 0.

The error of all the 5 models was seen on validation data error and the following are the results:

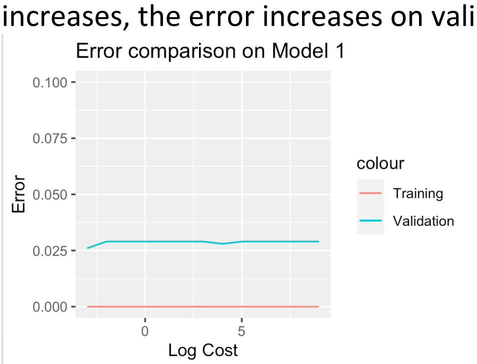
```
> validation
```

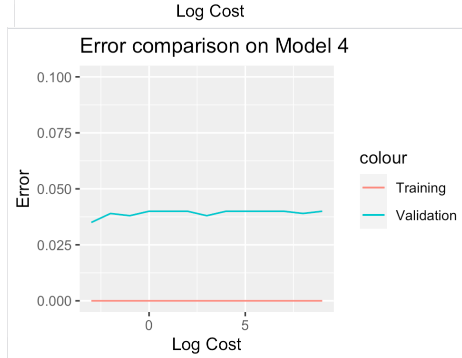
	0.125	0.25	0.5	1	2	4	8	16	32	64	128	256	512
Model1	0.026	0.029	0.029	0.029	0.029	0.029	0.029	0.028	0.029	0.029	0.029	0.029	0.029
Model2	0.044	0.049	0.048	0.049	0.042	0.042	0.042	0.041	0.042	0.043	0.042	0.042	0.042
Model3	0.033	0.039	0.038	0.042	0.042	0.043	0.043	0.043	0.043	0.043	0.043	0.044	0.044
Model4	0.035	0.039	0.038	0.040	0.040	0.040	0.038	0.040	0.040	0.040	0.040	0.039	0.040
Overall	0.083	0.096	0.100	0.098	0.096	0.099	0.092	0.092	0.092	0.095	0.100	0.107	0.099

The overall validation error is the lowest at cost = 0.125. This is in all the binary classifiers and the overall model.

Hence, we choose cost = 0.125 as the optimal cost due to least overall validation error of 0.083.

The following are the graphs of binary classifiers' error rate v/s Log_2(Cost). The error rate on training and validation data can be compared. It can be seen that as cost increases, the error increases on validation data.





This graph is the multi-class overall classifiers' error rate v/s Log_2(Cost). It has similar inference as that of binary classifier, indicating cost = 0.125 to be the best cost for our model.

(d) For the best C chosen = 0.125, 4 binary classifiers were again trained on the entire training data and tested on the test data. Error on binary classifiers:

	Train	Test
Model1	0.00025	0.03750000
Model2	0.00100	0.05541667
Model3	0.00050	0.03458333
Model4	0.00025	0.05791667

The binary classifiers were combined to form a multi-class classifier again, which gave the following results :

```
Confusion Matrix and Statistics

          Reference
Prediction   1   2   3   4
         1 528  34  18  17
         2  25 510  17  34
         3  20  21 546  21
         4  27  35  19 528

Overall Statistics

               Accuracy : 0.88
                 95% CI : (0.8663, 0.8927)
    No Information Rate : 0.25
    P-Value [Acc > NIR] : <0.0000000000000002

                  Kappa : 0.84

 Mcnemar's Test P-Value : 0.638

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            0.8800   0.8500   0.9100   0.8800
Specificity            0.9617   0.9578   0.9656   0.9550
Pos Pred Value         0.8844   0.8703   0.8980   0.8670
Neg Pred Value         0.9601   0.9504   0.9699   0.9598
Prevalence             0.2500   0.2500   0.2500   0.2500
Detection Rate         0.2200   0.2125   0.2275   0.2200
Detection Prevalence   0.2487   0.2442   0.2533   0.2537
Balanced Accuracy      0.9208   0.9039   0.9378   0.9175
```

The accuracy improved to 0.88. The error rate decreased from 13.69 to 12%. As compared to hard margin classifier, soft margin performs better.
Reason for better classification:
In hard margin SVM, even a single outlier can modify the decision boundary, which makes this classifier very sensitive to noise. Hard classifier would hence cause overfitting of the data and soft -margin classifier would perform better on test data (unseen data).

(e) The data was normalized suing wordspace library. The normalized data was used to train binary SVM classifiers, keeping the cost as optimal. The binary classifiers were then combined to perform the multi class classification again on test data and the following confusion matrix was retrieved:

```
> confusionMatrix(final, Y_Test)
Confusion Matrix and Statistics

          Reference
Prediction   1   2   3   4
         1 531  13  10   6
         2  15 524  16  14
         3  13   9 549   7
         4  41  54  25 573

Overall Statistics

               Accuracy : 0.9071
                 95% CI : (0.8948, 0.9184)
    No Information Rate : 0.25
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.8761

 Mcnemar's Test P-Value : 0.00000000001597

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            0.8850   0.8733   0.9150   0.9550
Specificity            0.9839   0.9750   0.9839   0.9333
Pos Pred Value         0.9482   0.9209   0.9498   0.8268
Neg Pred Value         0.9625   0.9585   0.9720   0.9842
Prevalence             0.2500   0.2500   0.2500   0.2500
Detection Rate         0.2213   0.2183   0.2288   0.2387
Detection Prevalence   0.2333   0.2371   0.2408   0.2888
Balanced Accuracy      0.9344   0.9242   0.9494   0.9442
```

The accuracy has now increased to 0.9071, reducing the error to 9.29%. Normalization improves results because initially a higher frequency of words would have resulted in an arbitrary higher weighted classification. Now that the frequency of words are normalized, the classifiers learns better by positioning the articles in the space fairly. The relative frequency of words is now taken into account rather than the absolute frequency.