EE219 PROJECT 2
Classification Analysis

Guanchu Ling 904590047
Yingchao Tang 404592020
Yang Guo 304681050

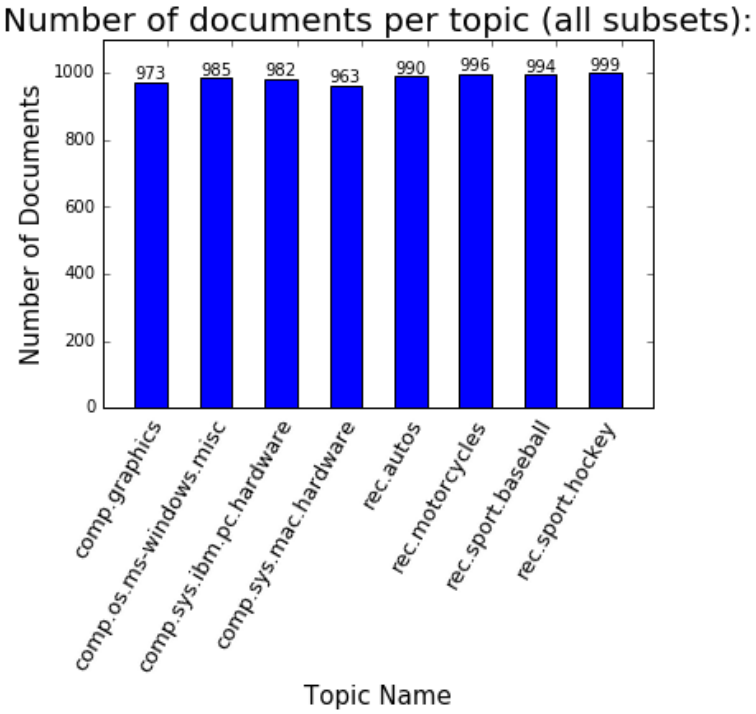
Introduction

In this project, we did classification analysis on the dataset “20 newsgroups”. It is a collection of approximately 20,000 newsgroup documents partitioned evenly across 20 different newsgroups. Each newsgroup document is a textual string that contains raw information. Our goal is to correctly classify all the documents into their corresponding classification. The data can be obtained based on their corresponding subset, namely “train” or “test”. We use the training dataset to build the classifier and then use the testing dataset to verify the effectiveness of the classifier. While we train the classifier, features of the newsgroup documents are extracted after some preprocessing of the raw data. Several techniques are utilized during this process, which will be elaborated in detail.

Question (a)

Before we carry out the classification analysis, first we have to make sure that the dataset is properly balanced with respect to every collection of documents that we are interested in. It is super important for the classifier to build accurately. To get started, we plotted a histogram of the number of documents per topic to make sure they are evenly distributed. We specifically get the statistics for “all subsets”, “training subsets” and “testing subsets”. The histograms and results are shown below:

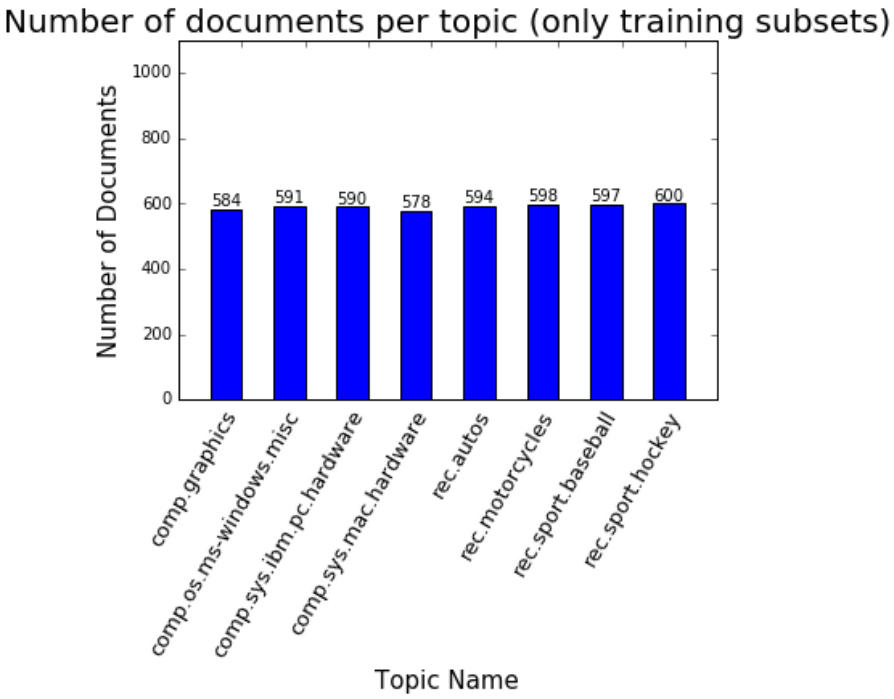
```
Number of documents per topic (all subsets):
             comp.graphics : 973
   comp.os.ms-windows.misc : 985
  comp.sys.ibm.pc.hardware : 982
     comp.sys.mac.hardware : 963
                 rec.autos : 990
           rec.motorcycles : 996
        rec.sport.baseball : 994
          rec.sport.hockey : 999

Number of documents in Computer Technology: 3903
Number of documents in Recreational Activity: 3979
```



Number of documents for all subsets

When we look at all subsets, it is obvious to see that the number of documents in each topic are very close, ranging from 950 to 1000. It is reasonable to guess that the numbers of documents are properly and evenly distributed.

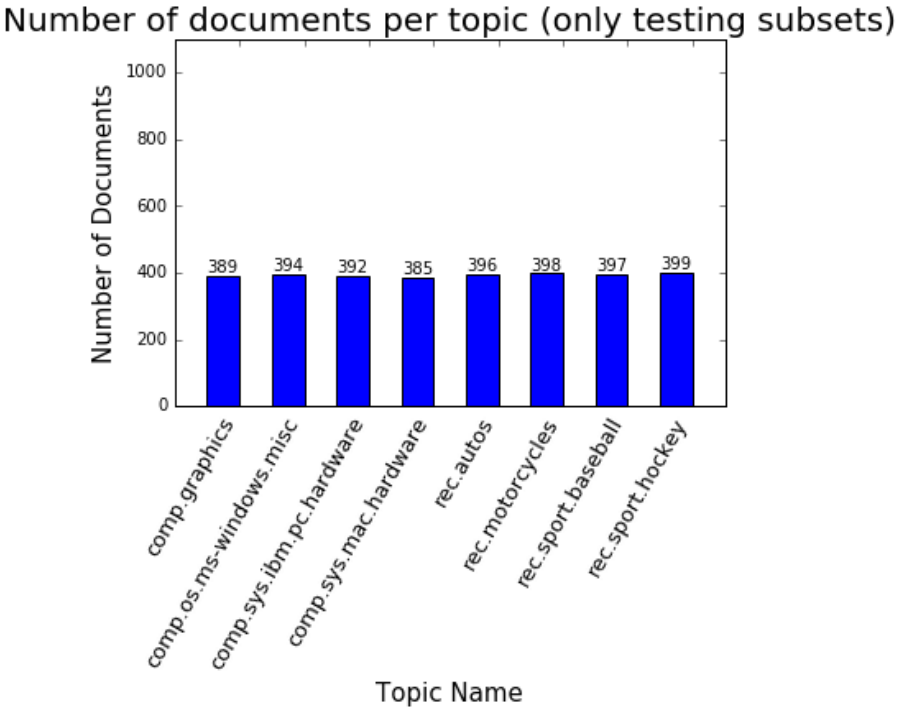
```
Number of documents per topic (only training subsets)
             comp.graphics : 584
   comp.os.ms-windows.misc : 591
  comp.sys.ibm.pc.hardware : 590
     comp.sys.mac.hardware : 578
                 rec.autos : 594
           rec.motorcycles : 598
        rec.sport.baseball : 597
          rec.sport.hockey : 600

Number of documents in Computer Technology: 2343
Number of documents in Recreational Activity: 2389
```



Number of documents for training subsets only

When we focus on training date only, the number of documents for topic ranges from 570 to 600 with very little variance. This verifies our guess the dataset is evenly distributed.

```
Number of documents per topic (only testing subsets)
             comp.graphics : 389
   comp.os.ms-windows.misc : 394
  comp.sys.ibm.pc.hardware : 392
     comp.sys.mac.hardware : 385
                 rec.autos : 396
           rec.motorcycles : 398
        rec.sport.baseball : 397
          rec.sport.hockey : 399

Number of documents in Computer Technology: 1560
Number of documents in Recreational Activity: 1590
```



Number of documents for testing subsets only

Finally, we look at testing subsets only. The number of documents for each topic ranges from 380 to 400, again with very small variance. Therefore we can draw the conclusion that this is a balanced dataset and the number of documents is evenly distributed for every topic.

Question (b)

In this part, we are going to model the text data and extract key features of the dataset. Since the representation should be succinct as not to include too much irrelevant information, we will try to summarize the dataset into a term-document matrix. The entries in the matrix are some statistics representing the significance of a word in the document. The approach for doing this is by using Term frequency - Inverse Document Frequency (TFxIDF) metric.

Before transforming the data into TFxIDF, we have to standardize the original data into a specific form that is much better and easier for the algorithm to process. The first step is to remove old the punctuations in the original data. Punctuations contribute very little to the key meaning of texts, but they can add a lot of noise when we use a computer algorithm to process the texts, which is why we have to remove them before we go on. Referring to ASCII codes, we provided all possible punctuations English for the algorithm to detect and delete from the original text. Next we have to remove newsgroup headers, footer signatures and quotes of every post in the dataset, because means information is surely irrelevant to the content of the post. The third step is very important – we will process all stem words. For example, “computers” and “computer” are actually the same word of different type. In our algorithm, if we regard these two words as different words, it would definitely cause big error in our final count of the frequency of words. Therefore we have to transform all stem words into the same form, which makes the computer easy to recognize. In the meantime we will also transform all uppercase letters into lowercase letters. The last step is to remove all stop words in the dataset because they can appear in all texts and don’t contribute to recognizing the classification of the text.

After doing all the pre-processing, we can now use a CountVectorizer to get the frequency counts of the words that we think are meaningful. There is a parameter that sets the limit for the minimum allowed frequency. If a word appears rarely, we are confident to regard it as irrelevant word. This parameter exactly decides below what frequency should the word be disgarded. So we know that as this parameter becomes larger, the number of words remaining will be less. Some results are listed below:

```
Min Frequency: 0
      Number of Terms: 102646
Min Frequency: 10
      Number of Terms: 5245
Min Frequency: 20
      Number of Terms: 2876
Min Frequency: 30
      Number of Terms: 2071
Min Frequency: 40
      Number of Terms: 1643
Min Frequency: 50
      Number of Terms: 1362
Min Frequency: 60
      Number of Terms: 1145
Min Frequency: 70
      Number of Terms: 970
Min Frequency: 80
      Number of Terms: 842
Min Frequency: 90
      Number of Terms: 759
Min Frequency: 100
      Number of Terms: 693
```

Question (c)

For this question, our task is to find the 10 most significant terms in the following classes:

```
“comp.sys.ibm.pc.hardware”
“comp.sys.mac.hardware”
“misc.forsale”
“soc.religion.christian”
```

We will use the techniques we just discussed above to pre-process the dataset and then count the frequencies of the words. For each class, we specifically calculated the statistics for “all subsets”, “training subset” and “testing subset”. The 10 most significant terms and their corresponding frequencies are listed below:

"comp.sys.ibm.pc.hardware"		
Data subset: All subsets included	Data subset: Only training subsets	Data subset: Only testing subsets
10 most significant terms:	10 most significant terms:	10 most significant terms:
"ani" | 448	"ani" | 279	"ani" | 169
"card" | 505	"bus" | 194	"card" | 174
"control" | 397	"card" | 331	"drive" | 217
"disk" | 367	"control" | 287	"know" | 119
"drive" | 897	"disk" | 277	"modem" | 115
"know" | 299	"drive" | 680	"pc" | 111
"problem" | 332	"ide" | 193	"problem" | 126
"scsi" | 580	"problem" | 206	"scsi" | 139
"use" | 726	"scsi" | 441	"use" | 254
"work" | 335	"use" | 472	"work" | 142

We can distinguish some words that are strongly related to PC hardware such as “disk”, “drive”, “scsi”, “bus”, “ide”, “pc”, etc. It makes sense that these words are the most significant ones.

"comp.sys.mac.hardware"		
Data subset: All subsets included	Data subset: Only training subsets	Data subset: Only testing subsets
10 most significant terms:	10 most significant terms:	10 most significant terms:
"ani" | 341	"ani" | 177	"ani" | 164
"appl" | 391	"appl" | 243	"appl" | 148
"doe" | 255	"card" | 159	"disk" | 118
"drive" | 385	"drive" | 250	"drive" | 135
"know" | 271	"know" | 163	"file" | 119
"like" | 267	"like" | 164	"know" | 108
"mac" | 629	"mac" | 362	"mac" | 267
"problem" | 355	"problem" | 219	"problem" | 136
"use" | 564	"use" | 330	"use" | 234
"work" | 293	"work" | 159	"work" | 134

We can distinguish some words that are strongly related to Mac hardware such as “appl”, “drive”, “mac”, “file”, “disk”, etc. It makes sense that these words are the most significant ones.

"misc.forsale"		
Data subset: All subsets included	Data subset: Only training subsets	Data subset: Only testing subsets
10 most significant terms:	10 most significant terms:	10 most significant terms:
"00" | 630	"00" | 284	"00" | 346
"10" | 303	"10" | 160	"10" | 143
"includ" | 302	"dos" | 179	"20" | 135
"new" | 394	"includ" | 209	"appear" | 138
"offer" | 353	"new" | 249	"drive" | 153
"pleas" | 276	"offer" | 228	"new" | 145
"price" | 306	"price" | 190	"offer" | 125
"sale" | 390	"sale" | 239	"pleas" | 122
"ship" | 284	"ship" | 161	"sale" | 151
"use" | 339	"use" | 222	"ship" | 123

We can distinguish some words that are strongly related to market and sale such as “offer”, “price”, “sale”, “ship”, “new”, “pleas”, etc. It makes sense that these words are the most significant ones.

"soc.religion.christian"		
Data subset: All subsets included	Data subset: Only training subsets	Data subset: Only testing subsets
10 most significant terms:	10 most significant terms:	10 most significant terms:
"believ" | 599	"ani" | 296	"christ" | 230
"christian" | 925	"believ" | 374	"christian" | 400
"church" | 653	"christian" | 525	"church" | 310
"god" | 1787	"church" | 343	"god" | 742
"jesus" | 626	"god" | 1045	"homosexu" | 351
"know" | 564	"jesus" | 423	"know" | 230
"peopl" | 684	"know" | 334	"peopl" | 273
"say" | 680	"peopl" | 411	"say" | 264
"sin" | 534	"say" | 416	"sin" | 322
"think" | 536	"think" | 340	"word" | 236

We can distinguish some words that are strongly related to Christian such as “believ”, “christian”, “church”, “god”, “jesus”, “sin”, etc. It makes sense that these words are the most significant ones.

Another observation of the results is that the 10 most significant terms in the training subsets and the testing subsets are very similar, which indicates that it is reasonable to build the classifier from the training subsets and then use it to classify the testing subsets.

Question (d)

After all the operations done so far, the dimensionality of the TFxIDF matrix is still very large. It could be computationally intensive for the computer to evaluate it. Thus we have to find a way to reduce the dimensionality of the matrix. In this project, we will be using Latent Semantic Indexing (LSI). Generally speaking, words used in the same context tend to have similar (or close) meanings. LSI could build the relationship between words in the same context and use this to identify the meaning of context.

The LSI technique is based on sigular value decomposition (SVD) of matrices. Let D be the TFxIDF matrix, then the SVD of the matrix could be expressed as $D = U\Sigma V^T$ where U and V are both unitary matrices and Σ is the singular value matrix. Since larger singular values contains more information than smaller singular values, we can discard those smaller singular values and truncate the singular value matrix Σ. Then we have the truncated SVD $D_k = U_k \Sigma_k V_k^T$, which is a good approximation of the original matrix. Next the matrix U_k can be used to project document column vectors into a 50-dimentional representation.

Question (e)

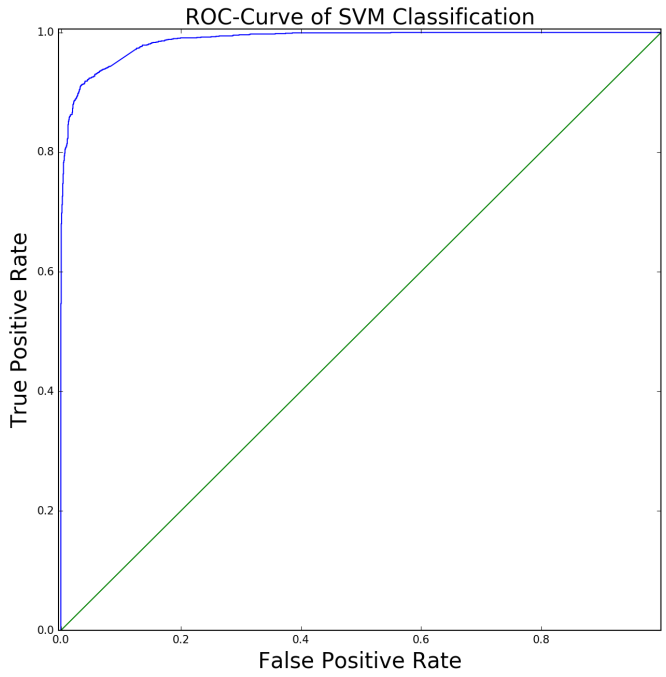
In the process of classification, the Support Vector Machine method gave us the hyperplane with the maximized margin, where support vector means the data samples that are closest to the hyperplane. In the scikit-learn kit, the SVM classification model has been provided as a built-in class type SVC, so we can easily implement the provided LinearSVC class and do the classification.

As what's done in the Question(d), we already tokenized the input textual documents, transformed them into the TF-IDF term-document matrix and reduced the dimension to 50 by feature selection. One more thing we need to do before training the classification model is to label the target data. We chose to use the [+1, -1] classification method, where "+1" represents Computer Technology and "-1" represents Recreational Activity.

So far we the separated training and testing set of design matrix along with the target vector. And we can just implement the built-in LinearSVC class for classification. The LinearSVC class is implemented in terms of liblinear, which offers more flexibility in the choice of penalties and loss functions and should scale better to a large number of sample data.

Here are the results of the LinearSVC classification:

```
                Classification report:
============================================================
             precision    recall  f1-score   support

   Com Tech       0.91      0.97      0.94      1590
    Rec Act       0.97      0.91      0.93      1560

avg / total       0.94      0.94      0.94      3150
============================================================

Confusion Matrix:
==============
[[1539   51]
 [ 148 1412]]
==============

Total accuracy:
0.936825396825
```



ROC-Curve of SVM Classification

As we can see from the data results above, the linear SVM classification could give us

the accuracy of 0.9368. And from the confusion matrix, we can see that our classification model performs slightly better at classifying documents about computer technology than the recreational activity.

Question (f)

Compared with the linear SVM where we define the closest data sample exactly satisfy the presentation of the canonical hyperplane, in the soft margin SVM, we follow the optimization solution that has a limit tolerance for a few weird noise sample data. For the very high dimensional problems, especially like the textual classification, sometimes the data are not linearly separable, so we expect to see better classification result with the soft margin classification. With the SVC class in the scikit-learn kit, we can perform the soft margin SVM with tunable trade-off parameter gamma. And we do the prediction with 5-fold cross validation trying to see whether the classification accuracy would be improved compared with the linear SVM. Here are the results (With trade-off parameter tuned from 1e-3 to 1e+3):

```
######################
#                    #
# Gamma value: 0.001 #
######################

              Classification report:
=========================================================
             precision    recall  f1-score   support

   Com Tech       0.50      1.00      0.67      3979
    Rec Act       0.00      0.00      0.00      3903

avg / total       0.25      0.50      0.34      7882
=========================================================

Confusion Matrix:
==============
[[3979    0]
 [3903    0]]
==============

Total accuracy:
0.504821111393
```

```
####################
#                  #
# Gamma value: 0.01 #
####################

             Classification report:
=========================================================
             precision    recall  f1-score   support

   Com Tech       0.82      0.98      0.89      3979
    Rec Act       0.98      0.78      0.87      3903

avg / total       0.90      0.88      0.88      7882
=========================================================

Confusion Matrix:
==============
[[3913   66]
 [ 873 3030]]
==============

Total accuracy:
0.880867800051
```

```
####################
#                  #
# Gamma value: 0.1 #
####################

             Classification report:
=========================================================
             precision    recall  f1-score   support

   Com Tech       0.88      0.98      0.92      3979
    Rec Act       0.97      0.86      0.91      3903

avg / total       0.92      0.92      0.92      7882
=========================================================

Confusion Matrix:
==============
[[3883   96]
 [ 554 3349]]
==============

Total accuracy:
0.917533620908
```

```
##################
#                #
# Gamma value: 1 #
##################

             Classification report:
=========================================================
             precision    recall  f1-score   support

   Com Tech       0.90      0.97      0.93      3979
    Rec Act       0.97      0.89      0.93      3903

avg / total       0.93      0.93      0.93      7882
=========================================================

Confusion Matrix:
==============
[[3861  118]
 [ 419 3484]]
==============

Total accuracy:
0.931870083735
```

```
###################
#                 #
# Gamma value: 10 #
###################

             Classification report:
=========================================================
             precision    recall  f1-score   support

   Com Tech       0.92      0.96      0.94      3979
    Rec Act       0.96      0.91      0.93      3903

avg / total       0.94      0.94      0.94      7882
=========================================================

Confusion Matrix:
==============
[[3816  163]
 [ 345 3558]]
==============

Total accuracy:
0.935549352956
```

```
####################
#                  #
# Gamma value: 100 #
####################

             Classification report:
=========================================================
             precision    recall  f1-score   support

   Com Tech       0.90      0.54      0.67      3979
    Rec Act       0.67      0.94      0.78      3903

avg / total       0.78      0.74      0.73      7882
=========================================================

Confusion Matrix:
==============
[[2152 1827]
 [ 247 3656]]
==============

Total accuracy:
0.736868815022
```

```
####################
#                  #
# Gamma value: 1000 #
####################

             Classification report:
=========================================================
             precision    recall  f1-score   support

   Com Tech       0.51      0.82      0.63      3979
    Rec Act       0.53      0.21      0.30      3903

avg / total       0.52      0.52      0.47      7882
=========================================================

Confusion Matrix:
==============
[[3256  723]
 [3092  811]]
==============

Total accuracy:
0.515985790409
```

From the numerical result above, we can see that as we are tuning the trade-off parameter, the classification accuracy increases firstly then decreases. When gamma is

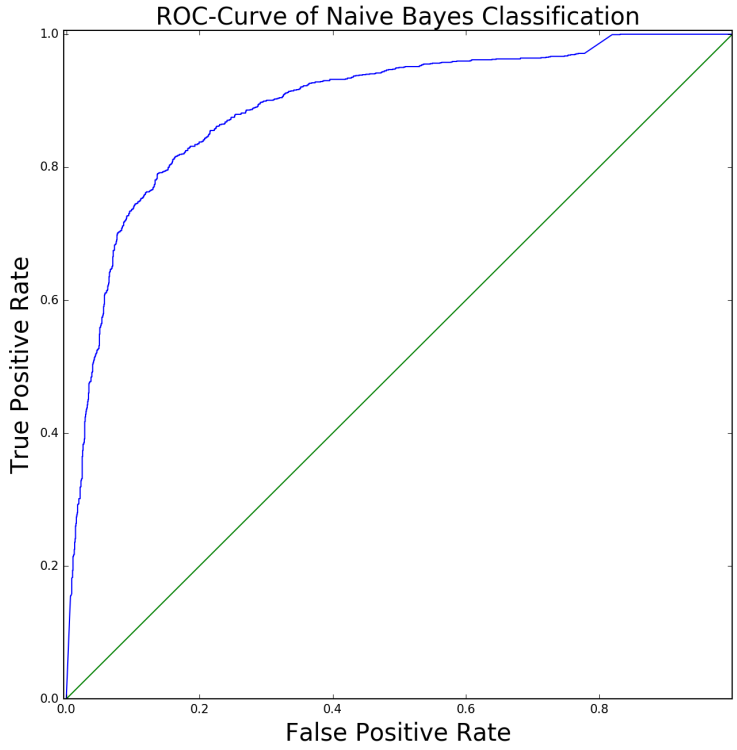
extremely small, all testing data samples are classified to the Computer Technology category. And this is the extreme case of under-fitting. While as the gamma increases, we found the classification accuracy falls within an acceptable range when gamma is in the range of 0.1 to 10, where accuracy peaks when gamma values 10. And from the confusion matrix, we can see that the optimized gamma in classifying Computer Technology and Recreational Activity are not same.

Among the different gamma-values we explored, the optimized value should be gamma = 10, which gives the accuracy of 0.9355. The peak accuracy is still slightly lower than the linear SVM. But as explained at the beginning of this part, soft margin SVM usually have better general performance due to that it could produce a more generalizable model when applied to new data and avoid overfitting that is likely to happen in linear SVM. So we suppose that there might be an optimized trade-off value between 0.1 and 10 that could generate a better model compared with the linear SVM.

Question (g)

In this part of the project, we perform the classification on the same data set using the Naïve Bayes algorithm. Using the built-in GaussianNB class, we can easily train the model and do the prediction on the test data set. Here are the results:

```
                Classification report:
=========================================================
             precision    recall  f1-score   support

   Com Tech       0.78      0.90      0.83      1590
    Rec Act       0.88      0.74      0.80      1560

avg / total       0.83      0.82      0.82      3150
=========================================================

Confusion Matrix:
==============
[[1428  162]
 [ 403 1157]]
==============

Total accuracy:
0.820634920635
```



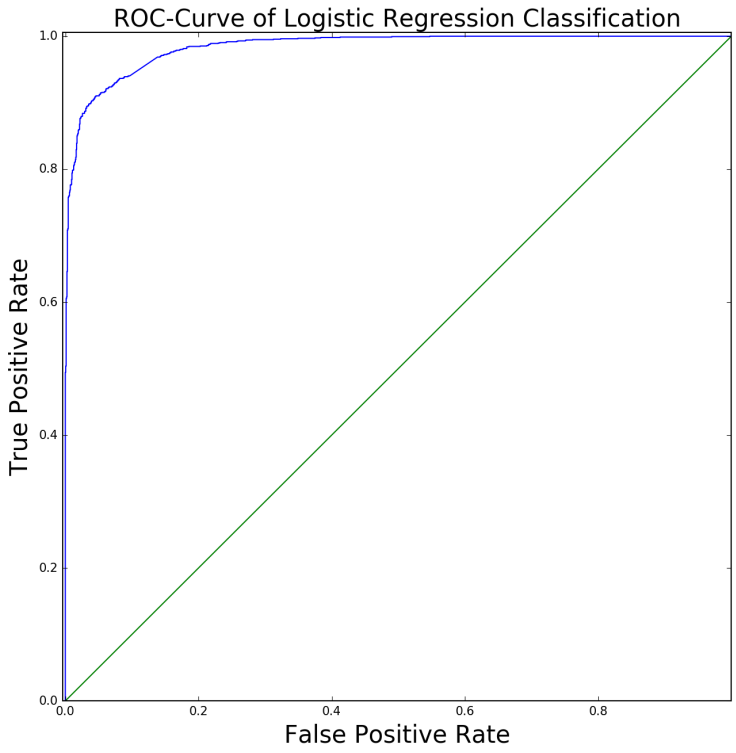
ROC-Curve of Naïve Bayes Classification

By comparing the ROC curves, we can see that the classification with naïve Bayes algorithm performs worse than the soft margin and the optimized soft margin SVM.

Question (h)

As we have already explored in the last project. The Logistic Regression have very good performance in linear regression. And now we want to see how well it can handle the classification problem. Firstly, we explored the general L2 regularized Logistic Regression with liblinear as the solver. And here are the results:

```
                Classification report:
=========================================================
             precision    recall  f1-score   support

   Com Tech       0.90      0.97      0.93      1590
    Rec Act       0.97      0.89      0.93      1560

avg / total       0.93      0.93      0.93      3150
=========================================================

Confusion Matrix:
==============
[[1541   49]
 [ 169 1391]]
==============

Total accuracy:
0.930793650794
```



ROC-Curve of Logistic Regression Classification

From the comparison of the ROC curve and numerical results, we can see that the Logistic Regression still performs very well in the classification case, giving the total accuracy of 0.9308, which is almost as good as the linear SVM. And we can see that for all the classification method we explored so far, the computer technology documents seem to have a higher probability to be recognized correctly.

Question (i)

In the last part, we explored the classification performance of the general logistic regression which is default using L2 regularization with penalty parameter C equals 1. In this part, we will explore the classification performance of L1 and L2 norm regularized logistic regression with different penalty parameter from very small value as 0.000001 to a very large value as 100000. L1 regularization and L2 regularization are two closely related techniques that can be used by machine learning (ML) training algorithms to reduce model overfitting. Eliminating overfitting leads to a model that makes better predictions. Here are some of the representative results:

```
#######################################
#                                     #
# PENALTY TYPE: l1 norm regularization #
#                                     #
# REGULARIZATION TERM: 0.0001         #
#######################################

                Classification report:
=====================================================
             precision    recall  f1-score   support

   Com Tech       0.50      1.00      0.67      1590
    Rec Act       0.00      0.00      0.00      1560

avg / total       0.25      0.50      0.34      3150
=====================================================

Confusion Matrix:
==============
[[1590    0]
 [1560    0]]
==============

Total accuracy:
0.504761904762
```

```
#######################################
#                                     #
# PENALTY TYPE: l2 norm regularization #
#                                     #
# REGULARIZATION TERM: 0.0001         #
#######################################

                Classification report:
=====================================================
             precision    recall  f1-score   support

   Com Tech       0.57      1.00      0.73      1590
    Rec Act       1.00      0.23      0.38      1560

avg / total       0.78      0.62      0.55      3150
=====================================================

Confusion Matrix:
==============
[[1589    1]
 [1199  361]]
==============

Total accuracy:
0.619047619048
```

```
#######################################
#                                     #
# PENALTY TYPE: l1 norm regularization #
#                                     #
# REGULARIZATION TERM: 0.001          #
#######################################

                Classification report:
=====================================================
             precision    recall  f1-score   support

   Com Tech       0.50      1.00      0.67      1590
    Rec Act       0.00      0.00      0.00      1560

avg / total       0.25      0.50      0.34      3150
=====================================================

Confusion Matrix:
==============
[[1590    0]
 [1560    0]]
==============

Total accuracy:
0.504761904762
```

```
#######################################
#                                     #
# PENALTY TYPE: l2 norm regularization #
#                                     #
# REGULARIZATION TERM: 0.001          #
#######################################

                Classification report:
=====================================================
             precision    recall  f1-score   support

   Com Tech       0.69      0.99      0.81      1590
    Rec Act       0.99      0.54      0.70      1560

avg / total       0.84      0.77      0.76      3150
=====================================================

Confusion Matrix:
==============
[[1582    8]
 [ 716  844]]
==============

Total accuracy:
0.770158730159
```

```
#######################################
#                                     #
# PENALTY TYPE: l1 norm regularization #
#                                     #
# REGULARIZATION TERM: 0.01           #
#######################################

             Classification report:
=====================================================
             precision    recall  f1-score   support

   Com Tech       0.85      0.93      0.89      1590
    Rec Act       0.92      0.83      0.87      1560

avg / total       0.88      0.88      0.88      3150
=====================================================

Confusion Matrix:
==============
[[1478  112]
 [ 271 1289]]
==============

Total accuracy:
0.878412698413
```

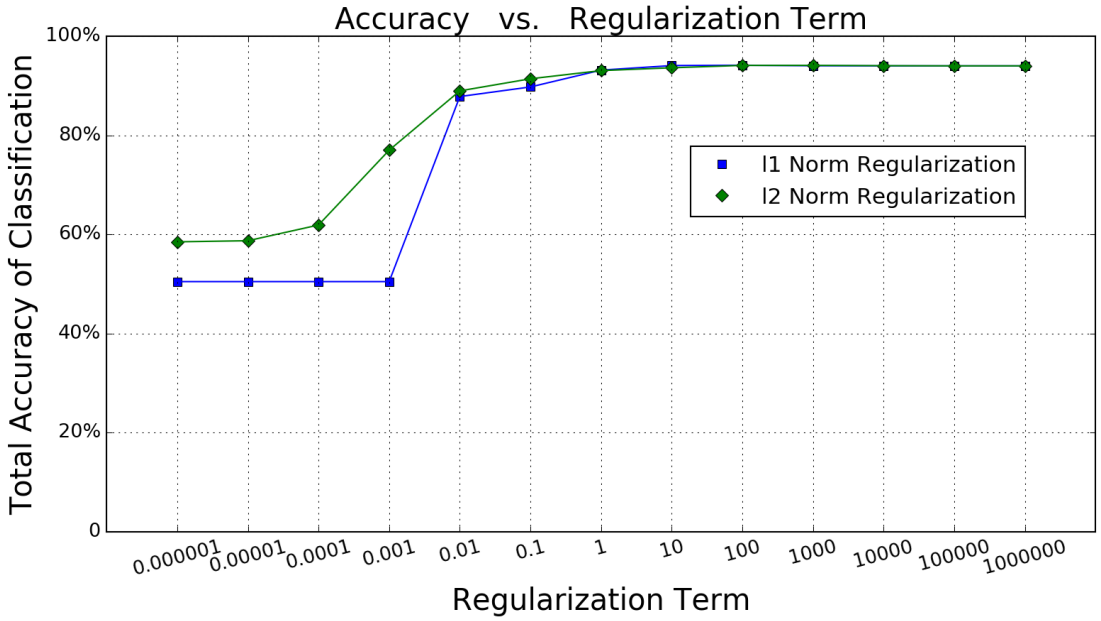
```
#######################################
#                                     #
# PENALTY TYPE: l2 norm regularization #
#                                     #
# REGULARIZATION TERM: 0.01           #
#######################################

             Classification report:
=====================================================
             precision    recall  f1-score   support

   Com Tech       0.83      0.98      0.90      1590
    Rec Act       0.97      0.80      0.88      1560

avg / total       0.90      0.89      0.89      3150
=====================================================

Confusion Matrix:
==============
[[1553   37]
 [ 310 1250]]
==============

Total accuracy:
0.889841269841
```

```
#######################################
#                                     #
# PENALTY TYPE: l1 norm regularization #
#                                     #
# REGULARIZATION TERM: 0.1            #
#######################################

             Classification report:
=====================================================
             precision    recall  f1-score   support

   Com Tech       0.87      0.94      0.90      1590
    Rec Act       0.93      0.86      0.89      1560

avg / total       0.90      0.90      0.90      3150
=====================================================

Confusion Matrix:
==============
[[1488  102]
 [ 220 1340]]
==============

Total accuracy:
0.897777777778
```

```
#######################################
#                                     #
# PENALTY TYPE: l2 norm regularization #
#                                     #
# REGULARIZATION TERM: 0.1            #
#######################################

             Classification report:
=====================================================
             precision    recall  f1-score   support

   Com Tech       0.88      0.97      0.92      1590
    Rec Act       0.96      0.86      0.91      1560

avg / total       0.92      0.91      0.91      3150
=====================================================

Confusion Matrix:
==============
[[1538   52]
 [ 218 1342]]
==============

Total accuracy:
0.914285714286
```

```
#######################################
#                                     #
# PENALTY TYPE: l1 norm regularization #
#                                     #
# REGULARIZATION TERM: 1              #
#######################################

             Classification report:
=====================================================
             precision    recall  f1-score   support

   Com Tech       0.91      0.96      0.93      1590
    Rec Act       0.96      0.90      0.93      1560

avg / total       0.93      0.93      0.93      3150
=====================================================

Confusion Matrix:
==============
[[1524   66]
 [ 149 1411]]
==============

Total accuracy:
0.931746031746
```

```
#######################################
#                                     #
# PENALTY TYPE: l2 norm regularization #
#                                     #
# REGULARIZATION TERM: 1              #
#######################################

             Classification report:
=====================================================
             precision    recall  f1-score   support

   Com Tech       0.90      0.97      0.93      1590
    Rec Act       0.97      0.89      0.93      1560

avg / total       0.93      0.93      0.93      3150
=====================================================

Confusion Matrix:
==============
[[1541   49]
 [ 169 1391]]
==============

Total accuracy:
0.930793650794
```



Accuracy vs. Regularization Term

From the results above we can see that, at very small values of penalty parameter the classification almost doesn't work when all test cases are classified as the computer technology. But as the penalty parameter increases, the accuracy of the classification grows as well. As we can see from the accuracy-regularization term plot, when the penalty parameter grows larger than 0.01, the classification accuracy increases to an acceptable range. And the accuracy grows to a stable level at around 0.93 for both L1 and L2 regularization term.

But it is very hard to tell which one is better. Generally, L1 regularization is more preferable because with L1 regularization, it is possible to drive one or more weight values to zero so that some insignificant features would be unselected, while L2 regularization can suppress the weight value but not entirely to zero. In our case, we can see that the classification accuracy with L1-regularized logistic regression is slightly higher than the L2-regularized case. So that's why in most machine learning cases L1 regularization is more preferable.

Multiclass Classification

We train classifiers on the documents belonging to the classes:

```
"comp.sys.ibm.pc.hardware"
"comp.sys.mac.hardware"
"misc.forsale"
"soc.religion.christian"
```

Since this is a multi-class problem, we use One Vs One and One Vs Rest classification techniques to train our classifier, based on Naive Bayes SVM. The steps followed to train the dataset are the same as analyzed in Question (b) and Question (c). The results show below:

```
#####################################
#                                   #
# One vs One Classifier - Naive Bayes #
#####################################

                    Classification Report:
=================================================================
                          precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.63      0.70      0.66       392
   comp.sys.mac.hardware       0.67      0.61      0.64       385
            misc.forsale       0.76      0.72      0.74       390
  soc.religion.christian       0.94      0.97      0.95       398

             avg / total       0.75      0.75      0.75      1565
=================================================================

Confusion Matrix:
===================
[[273  72  40   7]
 [100 234  43   8]
 [ 59  41 280  10]
 [  3   4   5 386]]
===================

Total Accuracy:
0.749520766773
```

```
##############################
#                            #
# One vs One Classifier - SVM #
##############################

                     Classification Report:
=================================================================
                          precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.81      0.85      0.83       392
   comp.sys.mac.hardware       0.86      0.81      0.83       385
            misc.forsale       0.86      0.88      0.87       390
  soc.religion.christian       0.99      0.97      0.98       398

             avg / total       0.88      0.88      0.88      1565
=================================================================

Confusion Matrix:
==================
[[334  34  23   1]
 [ 43 310  31   1]
 [ 28  17 344   1]
 [  7   1   3 387]]
==================

Total Accuracy:
0.878594249201
```

```
########################################
#                                      #
# One vs Rest Classifier - Naive Bayes #
########################################

                     Classification Report:
=================================================================
                          precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.63      0.68      0.65       392
   comp.sys.mac.hardware       0.66      0.61      0.63       385
            misc.forsale       0.75      0.71      0.73       390
  soc.religion.christian       0.93      0.98      0.95       398

             avg / total       0.74      0.75      0.74      1565
=================================================================

Confusion Matrix:
==================
[[266  77  43   6]
 [ 96 234  43  12]
 [ 58  43 277  12]
 [  1   1   6 390]]
==================

Total Accuracy:
0.745686900958
```

```
################################
#                              #
# One vs Rest Classifier - SVM #
################################

                     Classification Report:
=================================================================
                          precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.83      0.85      0.84       392
   comp.sys.mac.hardware       0.86      0.81      0.83       385
            misc.forsale       0.86      0.90      0.88       390
  soc.religion.christian       0.99      0.98      0.98       398

             avg / total       0.88      0.88      0.88      1565
=================================================================

Confusion Matrix:
==================
[[334  32  24   2]
 [ 43 310  30   2]
 [ 20  19 350   1]
 [  4   1   4 389]]
==================

Total Accuracy:
0.883706070288
```

Obviously, we can see that SVM Classification is always more accurate than Naïve Bayes Classification, no matter we use OneVsOne or OneVsRest.

References:

1. Wikipedia – Support Vector Machine

   https://en.wikipedia.org/wiki/Support_vector_machine

2. Wikipedia – Receiver operating characteristic

   https://en.wikipedia.org/wiki/Receiver_operating_characteristic

3. Official website of scikit-learn package

   http://scikit-learn.org/stable/documentation.html

4. Quora – What is a good explanation of Latent Semantic Indexing (LSI)?

   https://www.quora.com/What-is-a-good-explanation-of-Latent-Semantic-Indexing-LSI