**EE219 Project 2**



Classification Analysis

Winter 2017

Mojtaba Sahraee 204517437

Kia Karbasi 804617014

Heng Zhao 304436191

**Part A)**

In this part we are making sure that number of documents in each subclass is distributed evenly. To do so we are plotting histogram for number of documents in each subclass of Computer Technology and Recreational Activity. As it is obvious in the following table and histogram, all documents are distributed evenly in data set and there are almost 600 samples in each sub-class.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | graphics | wind.misc | ibm.pc.hardware | mac.hardware | auto | motorcycles | baseball | hocky |
| N | 584.0 | 591.0 | 590.0 | 578.0 | 594.0 | 598.0 | 597.0 | 600 |
|  | Number of document in Computer class =2343 | | | | Number of document in recreational class =2389 | | | |

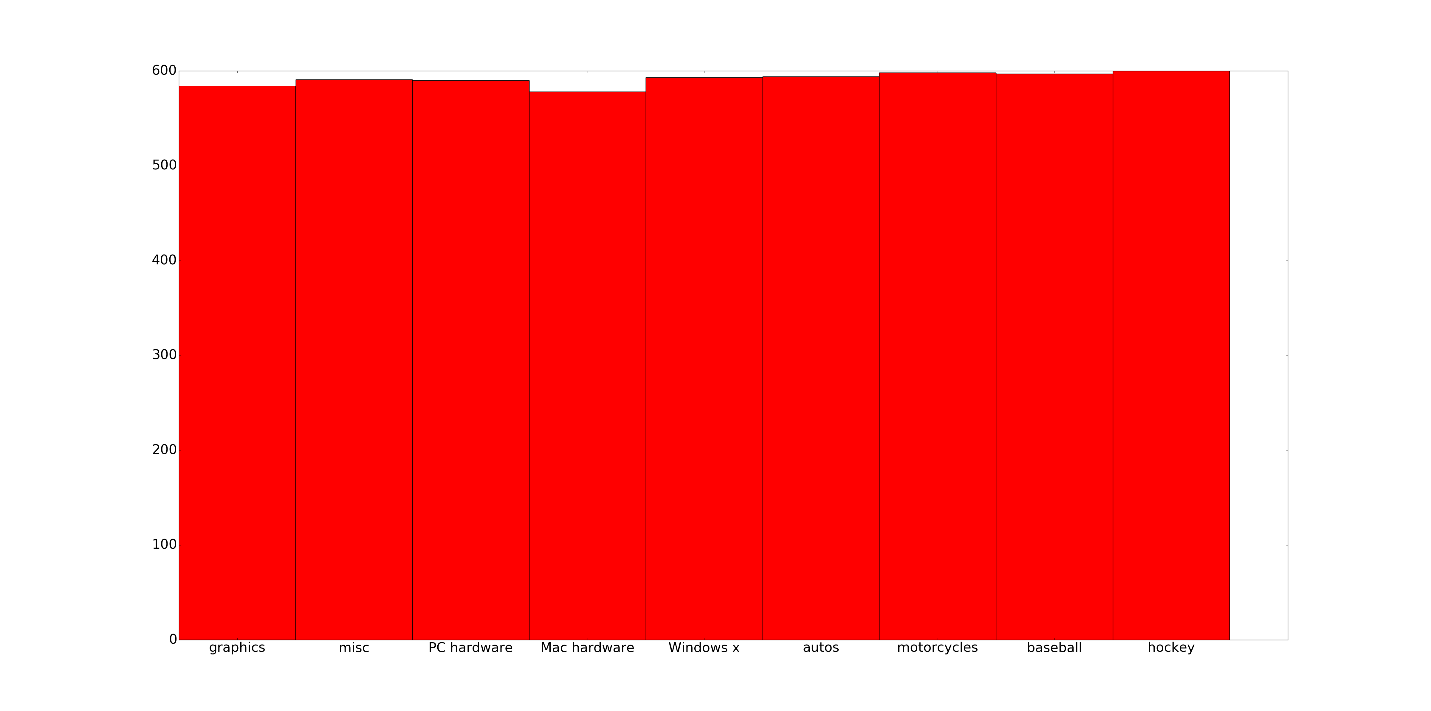


Figure 1

**Part B)**

After vectorizing and take TFxIDF transformation on train data, we have 4732 document with 60513 attribute (term) feature vector:

**X\_train\_tfidf.shape**

(4732, 60513)

**Part C)**

In order to find TFxICF, we first convert document-term matrix to class-term matrix and then find TF-ICF using the formula given. Doing so, we find the following 10 most significant terms (based on this criteria) for each sub-class:

ibm.pc.hardware:

['drive', 'scsi', 'card', 'controller', 'disk', 'drives', 'use', 'ide', 'bus', 'does']

comp.sys.mac.hardware:

['mac', 'apple', 'drive', 'use', 'problem', 'like', 'know', 'does', 'bit', 'just']

misc.forsale:

['00', 'new', 'sale', '50', '10', 'dos', 'offer', 'shipping', '20', 'price']

soc.religion.christian

['god', 'people', 'jesus', 'don', 'bible', 'just', 'think', 'christian', 'say', 'know']

**Part d)**

After applying TFxIDF, still the dimensionality of representation vector is very high. However, this vector is very sparse. Therefore we might not need all the elements in this vector. We know SVD or PCA can provide us with the dimension of maximum or minimum change. They can also tell us what mode has the highest effect on the system. In a control system this is typically done by a Hankel matrix representation. In this project we are using LSI (Latent Semantic Indexing) a dimension reducing technique to find the best representation of vector in a lower dimensional space. The LSI representation is obtained by computing eigenvectors corresponding to the largest eigenvalues of the term-document matrix. LSI is very much similar to Principal Component Analysis (PCA), except in PCA the eigenvalue decomposition is applied to the covariance matrix of the data( Note that here the covariance matrix is referred to the document-term matrix multiplied by its transpose, whose entries represent co-occurring terms in the documents.)

Eigenvectors corresponding to the dominant eigenvalues obtained by LSI are now directions related to dominant combinations of terms occurring in the corpuses, which are referred to as “topics” or “semantic concepts”. Thus, the new low dimensional representations are the magnitudes of the projections of the documents into these latent topics. In this project we are going to apply LSI to the TFxIDF matrix and pick k=50; so each document is mapped to a 50-dimensional vector.

Dk\_train.shape

(4732, 50)

**Part e)**

In this part, we’re using Support Vector Machines which have been proved efficient when dealing with sparse high dimensional datasets, including textual data because they have good generalization accuracy and computational complexity. In this project, we are using the Scikit package in python to train all the classifiers. Since hard margin SVM is essentially the soft margin SVM with a very large , as the objective function in the SVM suggests:

we use a soft margin SVM with large , to get a soft margin SVM classifier. But even when using a very large , we observe that not all the s are zero, which means that the data samples we have are not linearly separable in the feature space. In other words, the constraints in the hard margin SVM are not feasible. Nevertheless, using a soft margin SVM with large is the closest we can get to a hard margin SVM in the sense that it minimizes the number of points misclassified in the training data (even though this results in worse generalization to the test data).

Since the data are subdivided into sub-classes, and we are only interested in the two more general classes -computer and recreation-, we set the class of all the computer related sub-classes to 1 and recreation sub-classes to -1 and then train the classifier. We use for this part which highly penalizes misclassified training samples forcing them to be classified correctly and getting a close to hard margin SVm classifier. Firstly we divide the 8 groups we defined earlier in to two groups of Computer Technology and Recreational Activity and we label data by assigning 1 to the Computer Technology and -1 to Recreational Activity using the code in appendix. Then by importing Scikit-learn packages that we need we calculate the Precision, Accuracy, Confusion matrix, and graph the ROC. We achieve the result in Table 1 and figure 1.

Table 1 Classification Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Confusion Matrix** | **Precision** | **Accuracy** | **Recall** |
|  | 0.9103 | 0.9146 | 0.8592 |

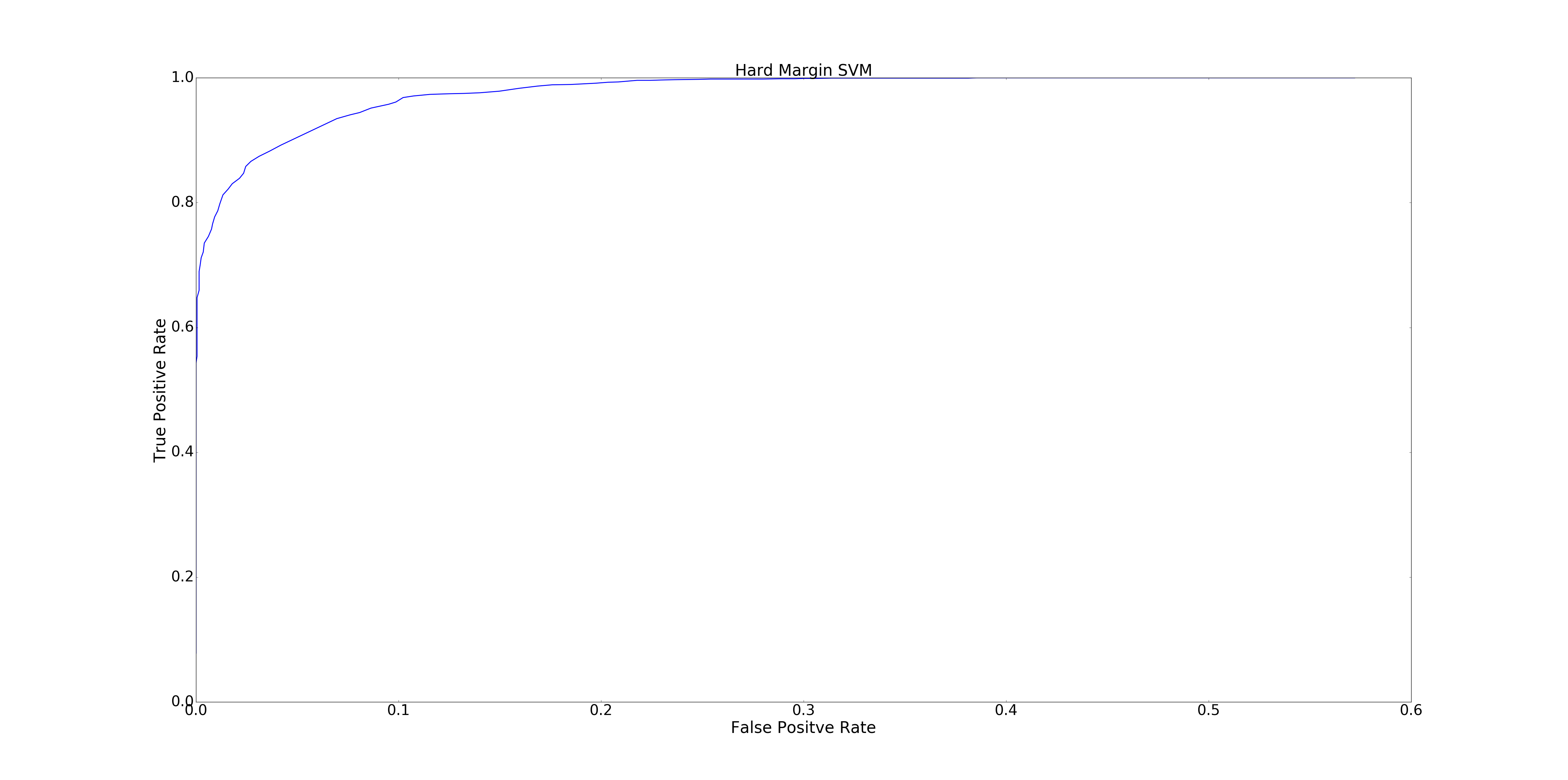
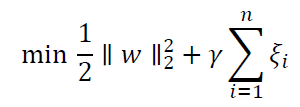


Figure 2 ROC Curve

**Part F)**

In this part, we use the Soft Margin SVM to solve the optimization problem shown below.



Subject to

C:\Users\makan\Desktop\part_f_2.PNG

The parameter γ determines how important each character is and when γ is zero we have Hard Margin SVM and we usually use low value of γ when we’re dealing with noisy samples. In Scikit-learn package we represent γ with the C. In this part we want to use Soft Margin SVM and Cross Fold Validation find and graph the classification parameters. We also want to find the best value of γ in range of [0.001, 1000]. The graph bellow shows the average accuracy in 5-fold cross-validation for the different values of :

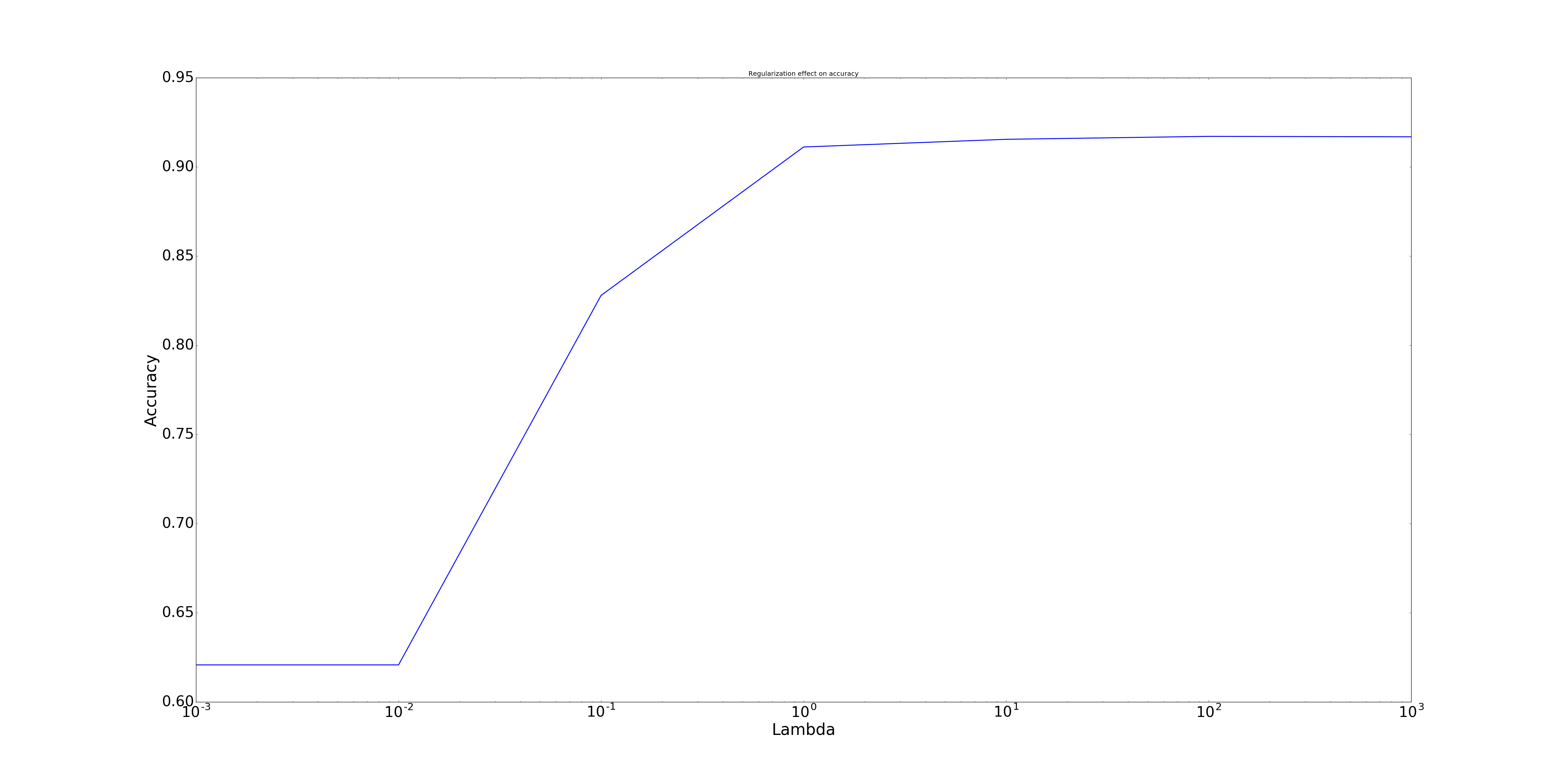


Figure 3

It can be seen that the best accuracy is achieved for larger values of . When is small, the objective function doesn’t care about misclassification and it tries to only maximize the margin which results in poor accuracy.

For which gives us the best accuracy we have the following results for the classifier and ROC for different thresholds and results of the classifier (clearly this is the same as before because the parameter

|  |  |  |  |
| --- | --- | --- | --- |
| **Confusion Matrix** | **Precision** | **Accuracy** | **Recall** |
|  | 0.9103 | 0.9146 | 0.8592 |

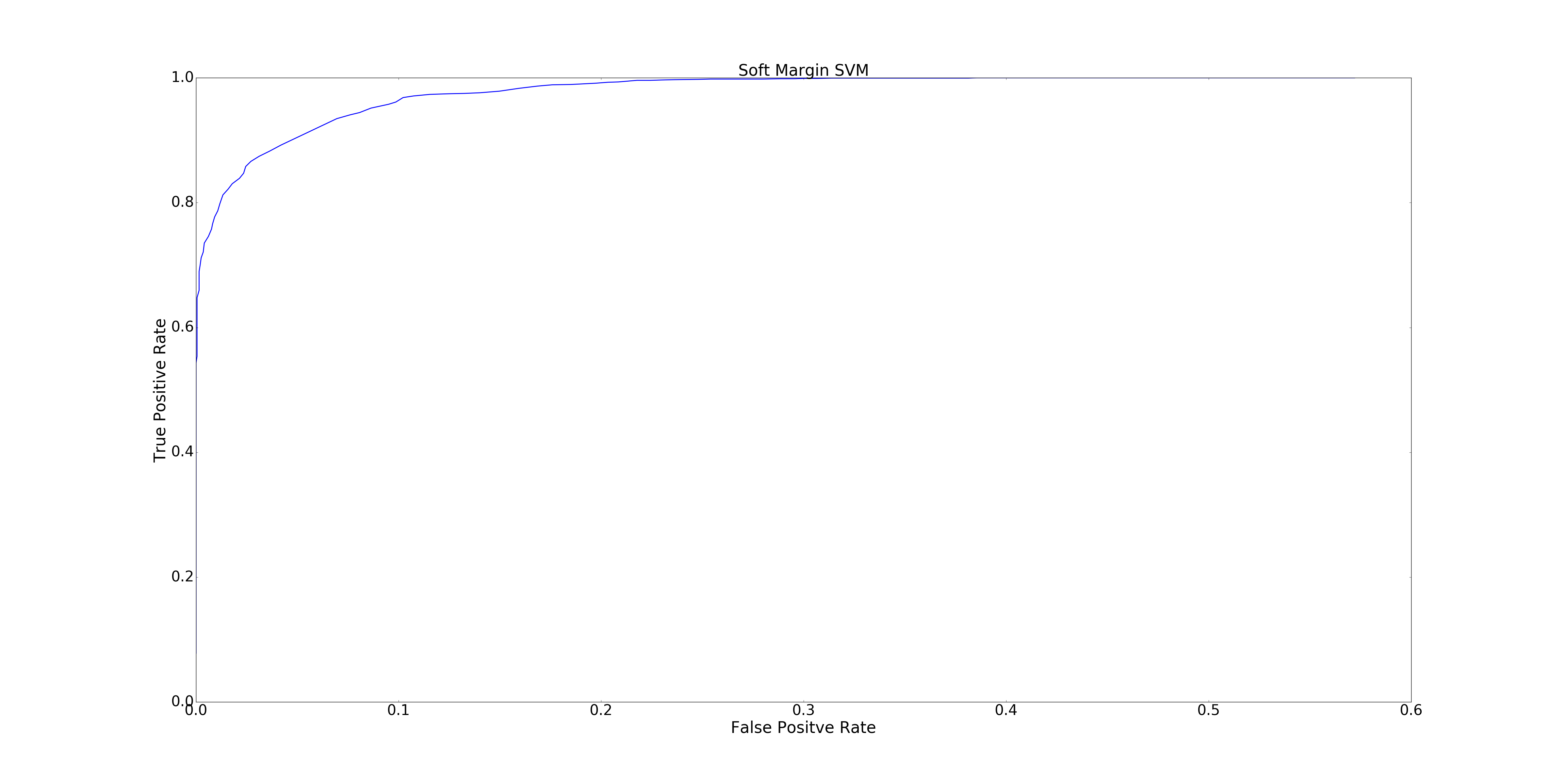


Figure 4

**Part G)**

In this part we use Naïve Bayes algorithm for the same problem as previous parts. This algorithm learns the joint distribution of features given the class assuming that the features are independent given the class and then estimates the maximum likelihood probability of a class given a document for inference. In this part, we can either use the Gaussian Naïve Bayes algorithm or the Multinomial Naïve Bayes algorithm in the Scikit-learn package to calculate the classification parameters that we calculated in previous parts using different algorithm. But since the Multinomial Naïve Bayes assumes that the features are non-negative integers (in fact, from a multinomial distribution), we can only train it on the document term matrix directly and not the lower dimensional feature space that we have found. Fortunately, the learning is fast enough to be able to run it on such a high-dimensional space. Below, are the results for the Multinomial Naïve Bayes classifier trained on the document term matrix.

|  |  |  |  |
| --- | --- | --- | --- |
| **Confusion Matrix** | **Precision** | **Accuracy** | **Recall** |
|  | 0.991546 | 0.879365 | 0.687605 |

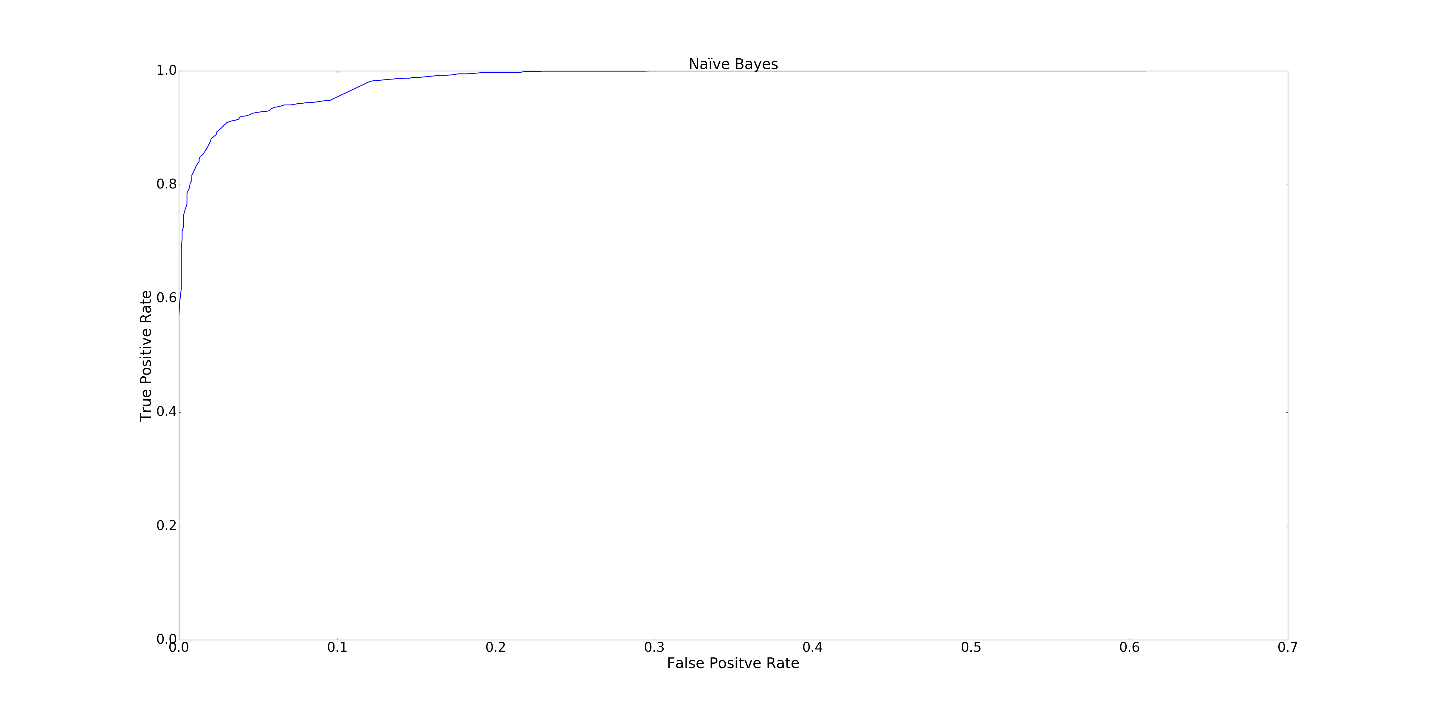


Figure 5 Roc Curve for Multinomial naive classifier

We can also use the Gaussian Naïve Bayes directly on 50-dimensional feature space. Below are the results of this classifier. Clearly the multinomial Naïve Bayes classifier is performing better than the Gaussian Naïve Bayes classifier.

|  |  |  |  |
| --- | --- | --- | --- |
| **Confusion Matrix** | **Precision** | **Accuracy** | **Recall** |
|  | 0.589111 | 0.731111 | 0.960636 |

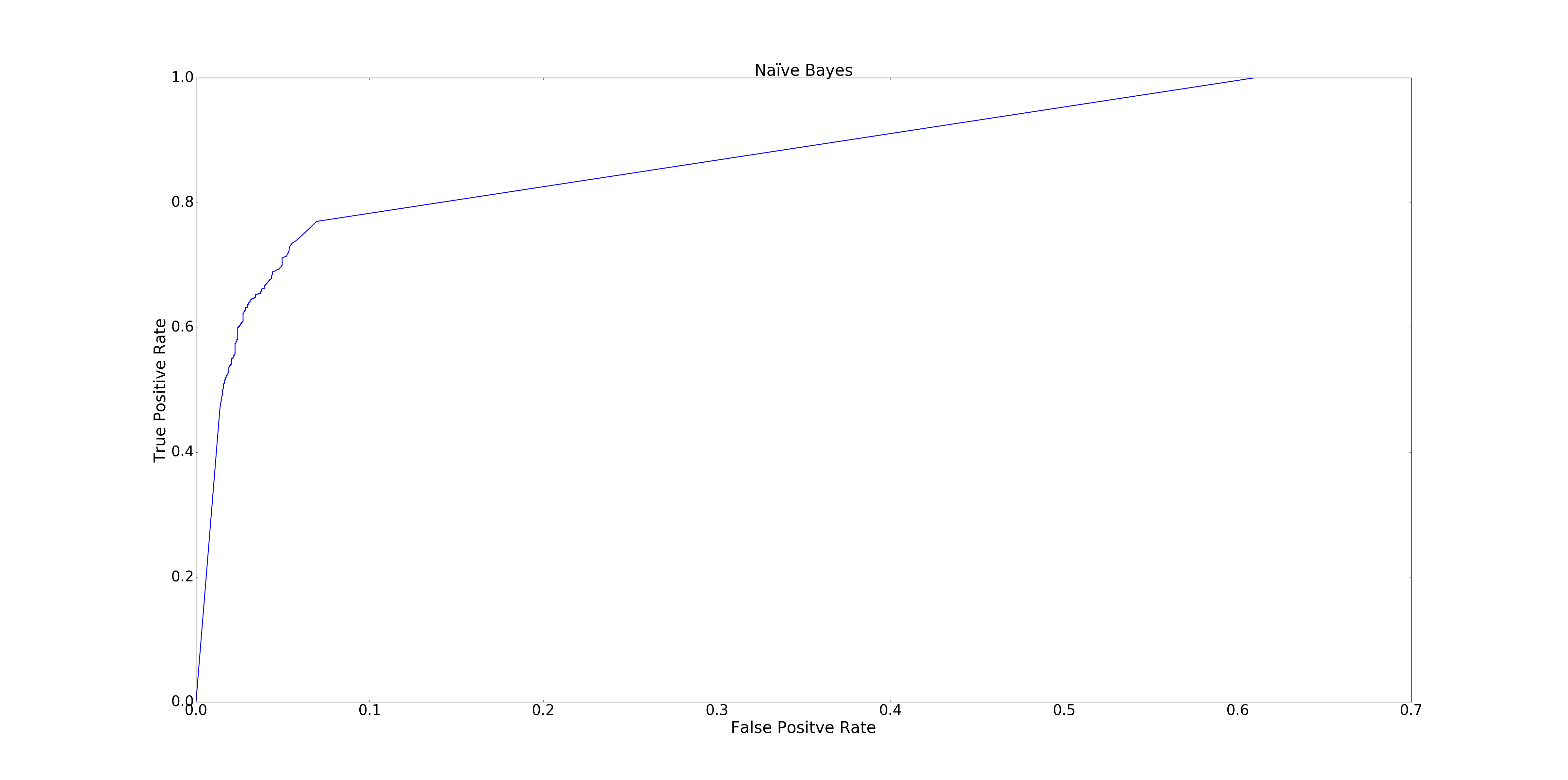


Figure Roc Curve for Gaussian naive classifier

**Part H)**

In this part, we just use another algorithm called Logistic Regression and we repeat the same procedure and part G. First we import the algorithm from Scikit-learn, and then we calculate and graph the classification parameters asked for this part same as the previous part.

|  |  |  |  |
| --- | --- | --- | --- |
| **Confusion Matrix** | **Precision** | **Accuracy** | **Recall** |
|  | 0.912389 | 0.916825 | 0.863484 |

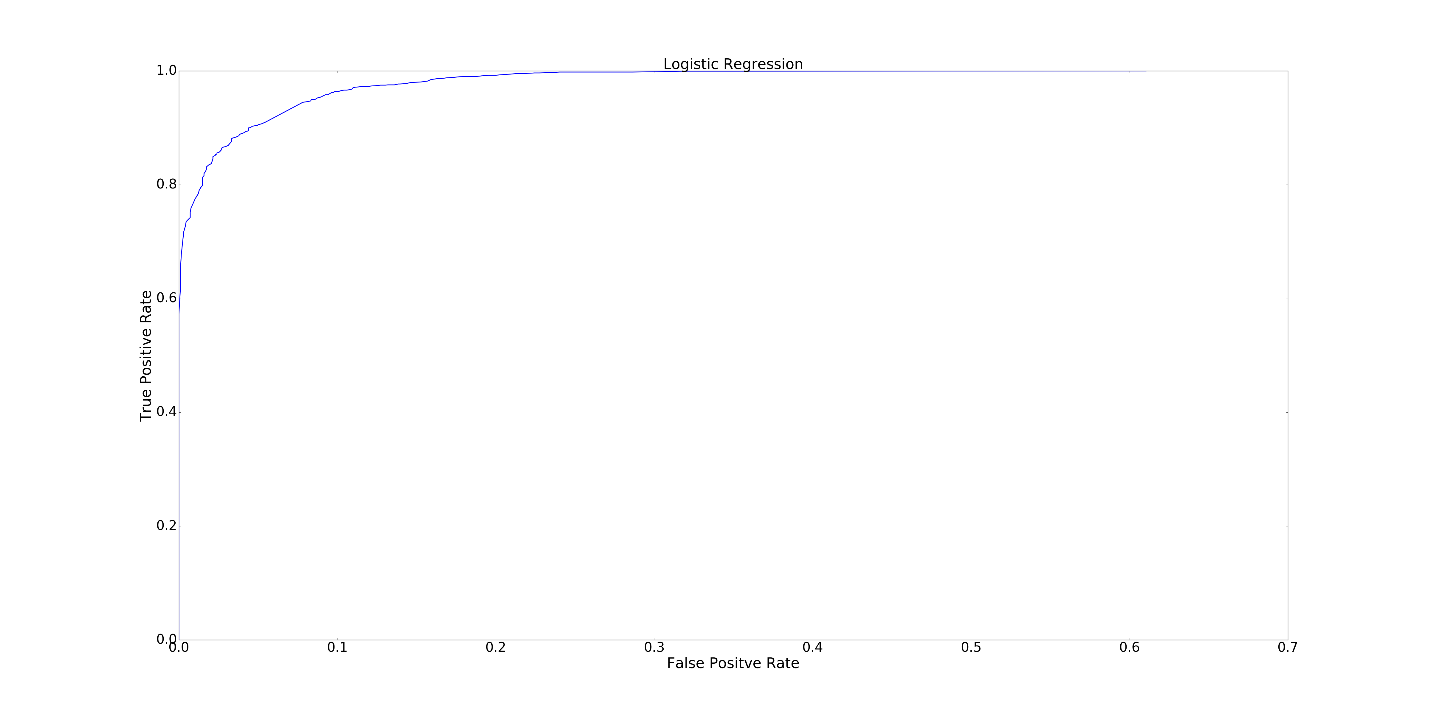


Figure 7 ROC curve for Logistic Regression algorithm

**Part i)**

For the 𝑙1 norm regularizations, we sweep through regularization coefficients and plot accuracy below in Figure 7. It can be seen that for coefficient values greater than “1,” accuracy decreases and for values lower than “1,” the accuracy remains a constant. I.e. any coefficient lesser than “1” should give us the required accuracy.

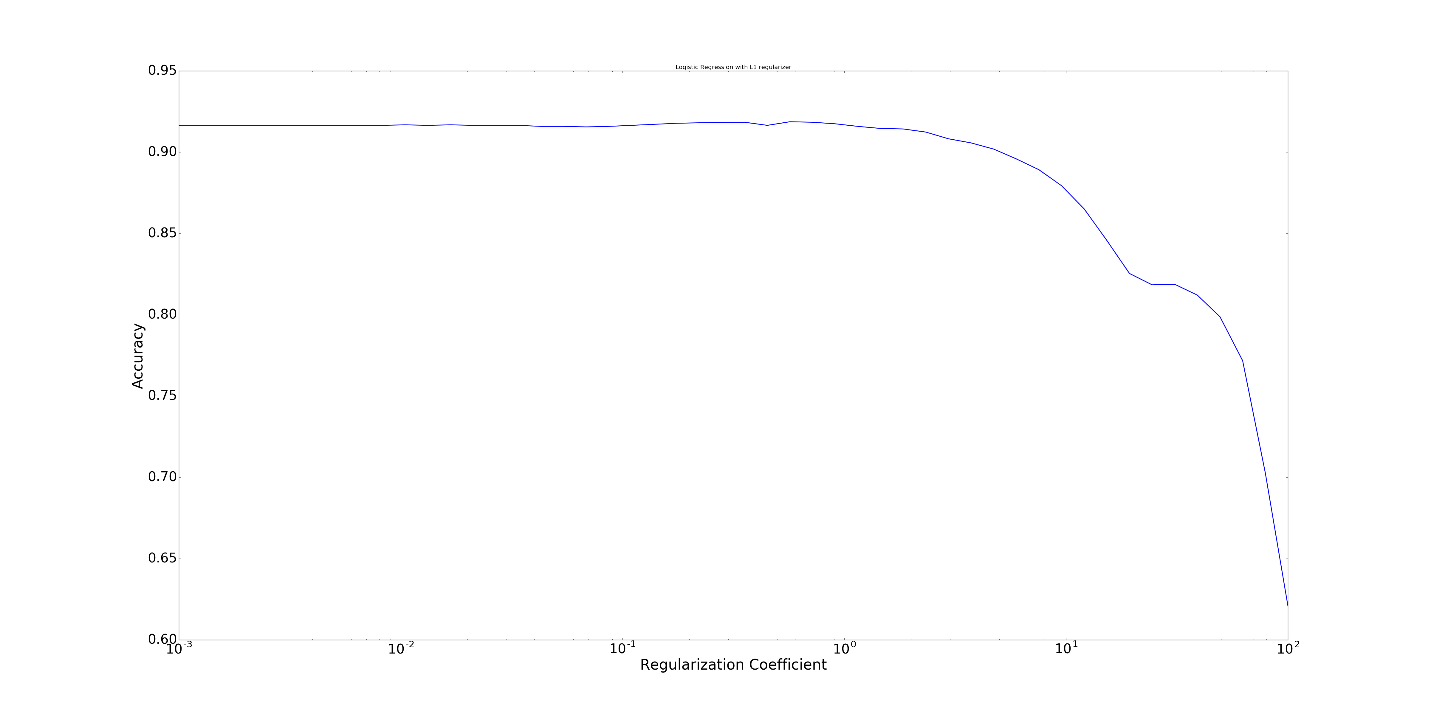
****

Figure 8

Following steps in part h, we plotted the ROC for the 𝑙1 norm regularizations below in figure 8.

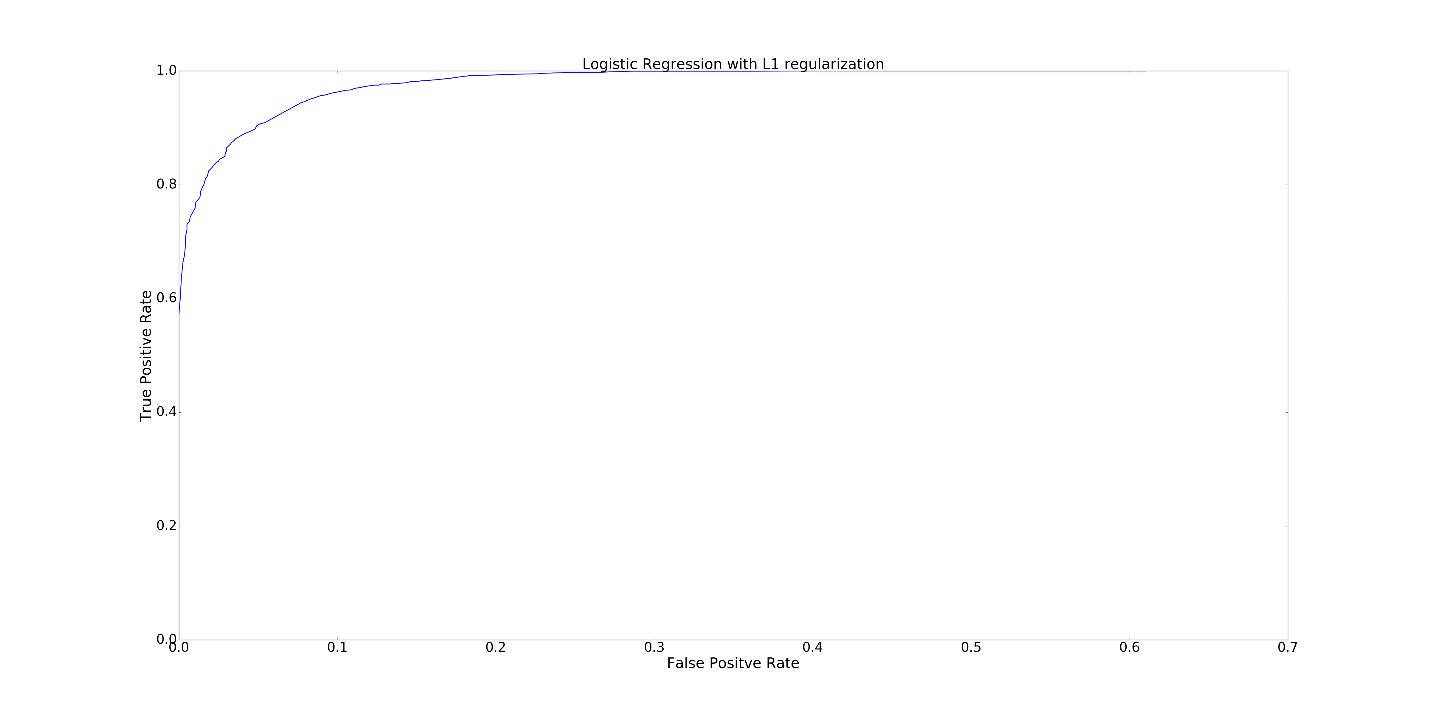
****

Figure 9

Next, we plotted the same graphs for the 𝑙2 norm regularizations and plotted the “Accuracy vs. Regularization” and the ROC in figure 9 and figure 10 respectively. We can see again that accuracy decreases for regularization coefficient values greater than 1. Moreover, for coefficient values less than “1,” the accuracy remains constant. I.e. we can also infer the any value less than “1,” would give us an acceptable accuracy.

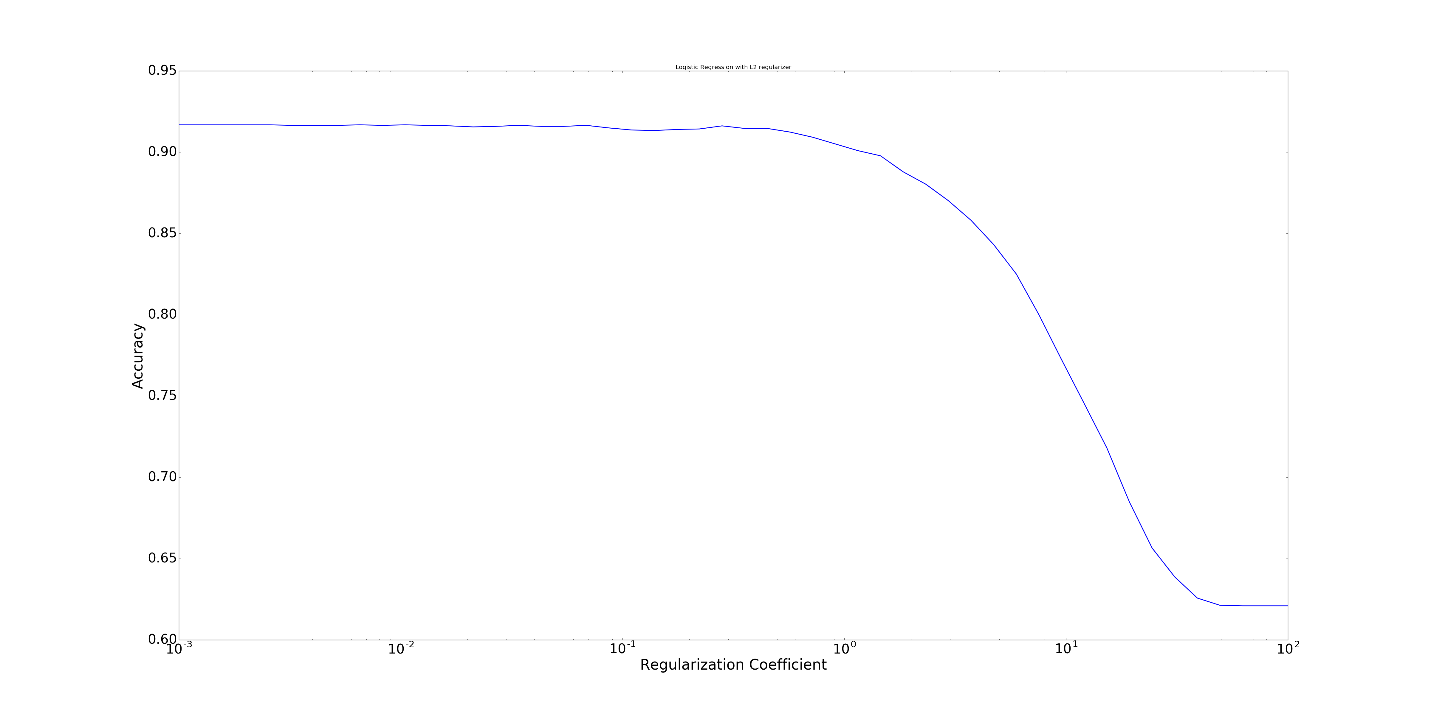
****

Figure 10

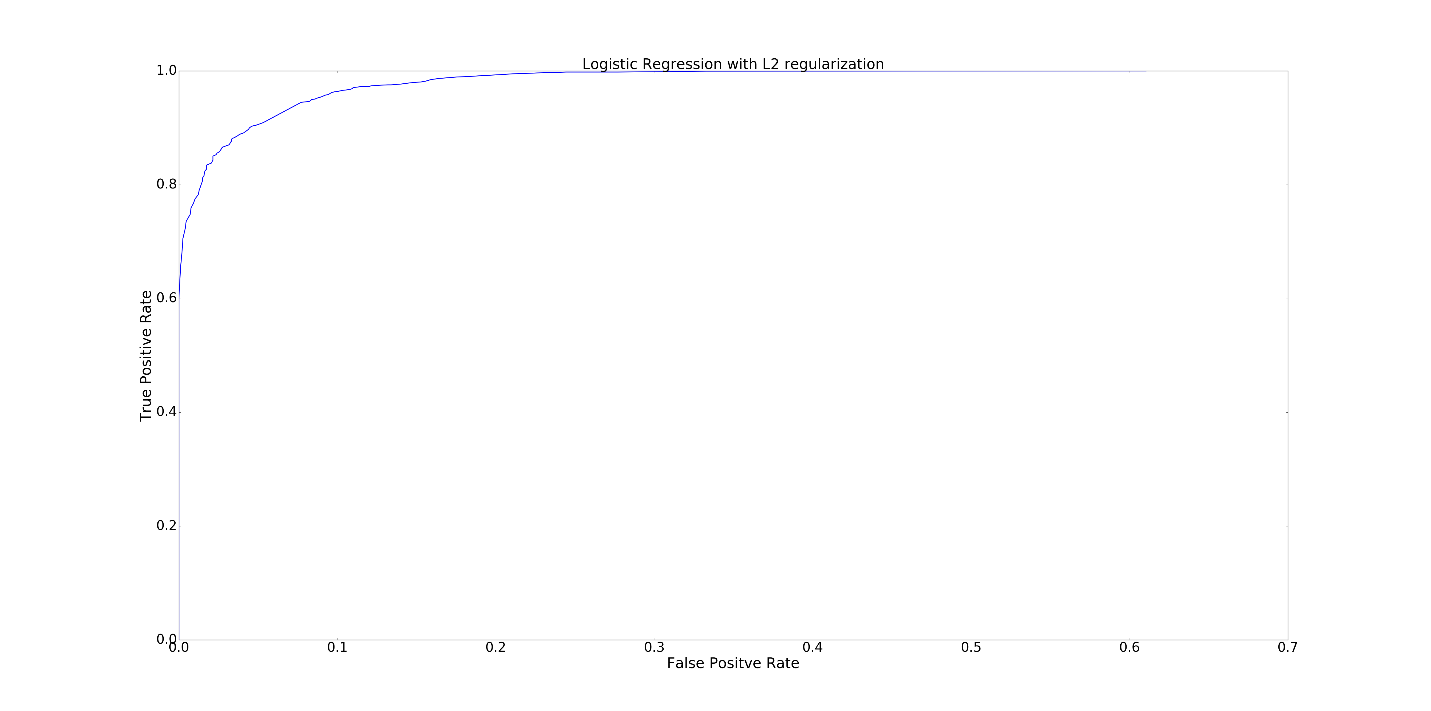
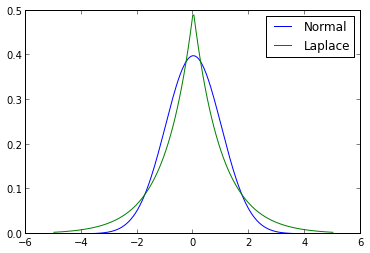
****

Figure 11

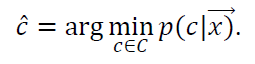
**A discussion on which penalty to use:** using an additive penalty as in the or case is equivalent to using a prior distribution for the parameters of the classifier. penalty means that the prior is Gaussian, so should be used when the coefficient approximately come from a Gaussian. On the other hand, an penalty implies that the coefficients are coming from a Laplacian prior which has the following distribution:



It can be seen that it is concentrated around zero but has heavy tails compared to Gaussian distribution. This prior hence encourages sparse coefficients and is useful when we know the coefficients have to be sparse.

**Part j) (i2?)**

In this part unlike the previous parts we aim to do the classification on more than two classes. We use Naïve Bayes algorithm and SVM to perform classification on the multiple classes. Naïve Bayes algorithm finds the class with maximum likelihood given the data, regardless of the number of classes, and the probability of each class label is computed in a way that the class with the highest probability is picked as depicted below.



For SVM we can use two different methods to handle multiple classes. The first method is to choose the pair of classes and perform a one to one classification technique, and the second method we fit one classifier per class, which reduce the number of classifiers. In this part we use 4 classes instead of 2 classes and we use the package of Scikit-learn to calculate the classification parameters for One VS One and One VS the rest for both SVM and Naïve Bayes.

Before doing classification we should make sure to have evenly distributed data. To do so, same as procedure in part A we find the document numbers in 4 classes which is mentioned.

|  |  |
| --- | --- |
| comp.sys.ibm.pc.hardware : | 590 |
| comp.sys.mac.hardware: | 578 |
| misc.forsale: | 585 |
| soc.religion.christian: | 599 |

As it is obvious from table all document evenly distributed in 4 classes.

Table 2 One VS rest with SVM

|  |  |  |  |
| --- | --- | --- | --- |
| **Confusion Matrix** | **Precision** | **Accuracy** | **Recall** |
|  | 0.8294 | 0.8262 | 0.8257 |

Table 4 One Vs one with SVM

|  |  |  |  |
| --- | --- | --- | --- |
| **Confusion Matrix** | **Precision** | **Accuracy** | **Recall** |
|  | 0.8144 | 0.8096 | 0.8090 |

Table 5 Multiclass Classification with Multinomial Naive Classifier

|  |  |  |  |
| --- | --- | --- | --- |
| **Confusion Matrix** | **Precision** | **Accuracy** | **Recall** |
|  | 0.8465 | 0.8435 | 0.8423 |

Clearly, the Naïve Bayes classifier is performing better than both SVM classifiers. Among the SVM classifiers, One vs Rest is performing slightly better than the classifier trained One vs One but the two are comparable.