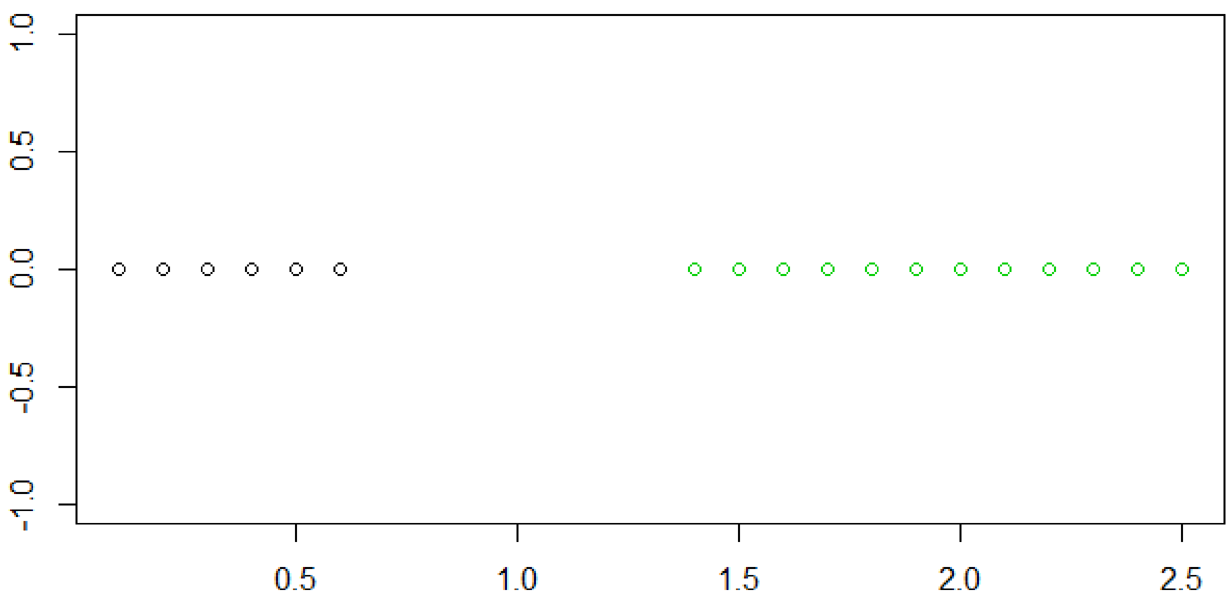
NIRMAL KUMAR RAVI

ASSIGNMENT 2

1D 2-class Gaussian discriminant analysis

Dataset - I have used [Iris] dataset for 1D 2-class Gaussian discriminant analysis.
Features - petal width
Classes- Iris-setosa, Iris-virginica

Let's plot our dataset to see how data looks like



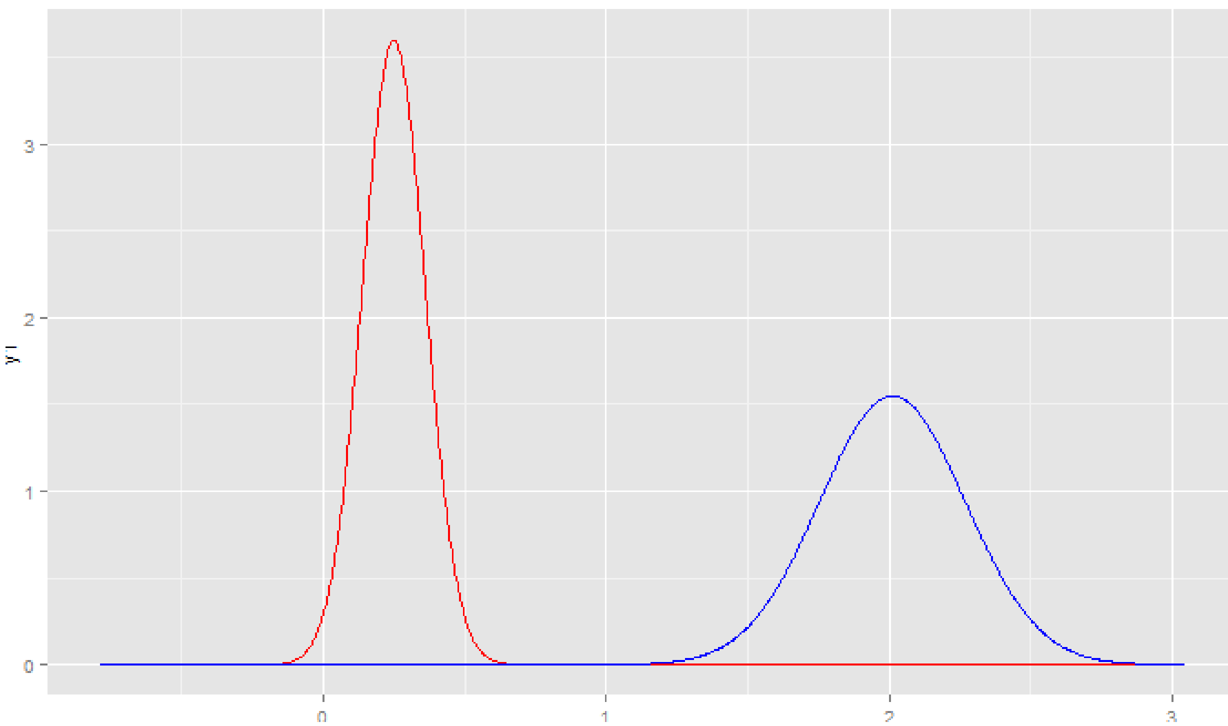
As we can see clear separation between points of two classes we can fit Gaussian in it.

Calculate parameters

To predict class we must calculate it parameters (Mean, variance) of Gaussian distribution from the training set for each class.

Class	Mean	SD
Iris-setosa	0.250000	0.1108409
Iris-virginica	2.010526	0.2576290

Distribution Plot



As we can see from the plot there is little or no overlapping between two Gaussians.

Performance Measure

As you can see there is a clear separation between two Gaussian so
Accuracy – 1 , Presidion – 1 , recall – 1 , F-measure - 1

nD 2-class Gaussian discriminant analysis

Dataset - I have used Iris dataset for nD 2-class Gaussian discriminant analysis.
Features - petal width
Classes- Iris-setosa, Iris-virginica

Calculate parameters

To predict class we must calculate it parameters (Mean, covariance) of Gaussian distribution from the training set for each class.

```
mean
mean `Iris-setosa`

sepal.length 4.991667
sepal.width  3.452778
petal.length 1.477778
petal.width  0.250000

mean `Iris-virginica`
                 [,1]
sepal.length 6.639474
sepal.width  2.973684
petal.length 5.581579
petal.width  2.010526


sigma
sigma `Iris-setosa`
             sepal.length sepal.width petal.length petal.width
sepal.length   0.14764286  0.10616667  0.024095238 0.014714286
sepal.width    0.10616667  0.13113492  0.012063492 0.015000000
petal.length   0.02409524  0.01206349  0.035492063 0.005714286
petal.width    0.01471429  0.01500000  0.005714286 0.012285714

sigma `Iris-virginica`
             sepal.length sepal.width petal.length petal.width
sepal.length   0.39704836  0.08701280   0.30885491  0.05822191
sepal.width    0.08701280  0.10253201   0.06733997  0.03974395
petal.length   0.30885491  0.06733997   0.32965149  0.06100996
petal.width    0.05822191  0.03974395   0.06100996  0.06637269


priors
priors `Iris-setosa`
0.4864865

priors `Iris-virginica`
0.5135135
```

Performance Measure

```
confusion_matrix
                 truelabel
prediction        Iris-setosa Iris-virginica
  Iris-setosa              13              0
  Iris-virginica            0             12

accuarcy
1

percision
   Iris-setosa Iris-virginica
             1              1

recall
   Iris-setosa Iris-virginica
             1              1

fmeasure
   Iris-setosa Iris-virginica
             1              1
```

Cross Validation (5 folds)

```
avgAccuracy
 1

avgPercision
   Iris-setosa Iris-virginica
             1              1

avgRecall
   Iris-setosa Iris-virginica
             1              1

avgFmeasure
   Iris-setosa Iris-virginica
             1              1
```

nD k-class Gaussian discriminant analysis

Dataset - I have used Iris dataset for nD k-class Gaussian discriminant analysis.
Features - petal width
Classes- Iris-setosa, Iris-virginica, Iris-versicolor

Calculate parameters

To predict class we must calculate it parameters (Mean, covariance) of Gaussian distribution from the training set for each class.

Calculate parameters

To predict class we must calculate it parameters (Mean, covariance) of Gaussian distribution from the training set for each class.

```
mean `Iris-setosa`

sepal.length 5.030556
sepal.width  3.430556
petal.length 1.480556
petal.width  0.250000

mean `Iris-virginica`

sepal.length 6.555556
sepal.width  2.952778
petal.length 5.527778
petal.width  2.025000

mean `Iris-versicolor`

sepal.length 5.902564
sepal.width  2.758974
petal.length 4.241026
petal.width  1.317949
```

```
sigma `Iris-setosa`
             sepal.length sepal.width petal.length petal.width
sepal.length   0.13075397 0.100468254  0.010325397 0.010428571
sepal.width    0.10046825 0.134753968  0.004611111 0.016714286
petal.length   0.01032540 0.004611111  0.035325397 0.006142857
petal.width    0.01042857 0.016714286  0.006142857 0.012857143

sigma `Iris-virginica`
             sepal.length sepal.width petal.length petal.width
sepal.length   0.42882540  0.07498413   0.34726984  0.03857143
sepal.width    0.07498413  0.09170635   0.06220635  0.04035714
petal.length   0.34726984  0.06220635   0.36777778  0.05100000
petal.width    0.03857143  0.04035714   0.05100000  0.07278571

sigma `Iris-versicolor`
             sepal.length sepal.width petal.length petal.width
sepal.length   0.23446694  0.08852901   0.16989204  0.05626856
sepal.width    0.08852901  0.10300945   0.09146424  0.04680837
petal.length   0.16989204  0.09146424   0.22616734  0.07819163
petal.width    0.05626856  0.04680837   0.07819163  0.04361673

priors `Iris-setosa`
 0.3243243

priors `Iris-virginica`
 0.3243243

priors `Iris-versicolor`
 0.3513514
```

Performance Measure

```
confusion_matrix
                   truelabel
prediction          Iris-setosa Iris-versicolor Iris-virginica
  Iris-setosa                13               0              0
  Iris-versicolor             0              11              0
  Iris-virginica              0               0             14

accuarcy
 1

percision
    Iris-setosa Iris-versicolor  Iris-virginica
              1               1               1

recall
    Iris-setosa Iris-versicolor  Iris-virginica
              1               1               1

fmeasure
    Iris-setosa Iris-versicolor  Iris-virginica
              1               1               1
```

Cross Validation (5 folds)

```
avgAccuracy
0.9666667

avgPercision
    Iris-setosa Iris-versicolor  Iris-virginica
           1.00            0.94            0.96

avgRecall
    Iris-setosa Iris-versicolor  Iris-virginica
      1.0000000       0.9577778       0.9436364

avgFmeasure
    Iris-setosa Iris-versicolor  Iris-virginica
      1.0000000       0.9484211       0.9514286
```

Naive Bayes with Bernoulli features

Dataset – I have used sms dataset for Naive Bayes with Bernoulli.
Features – number of unique words in document
Classes – spam, ham

In Bernoulli we see whether the word exists or not, we don’t care about number of times the word occurs.

Assumption: features are not correlated

Let’s train our model

Size of training set 70%

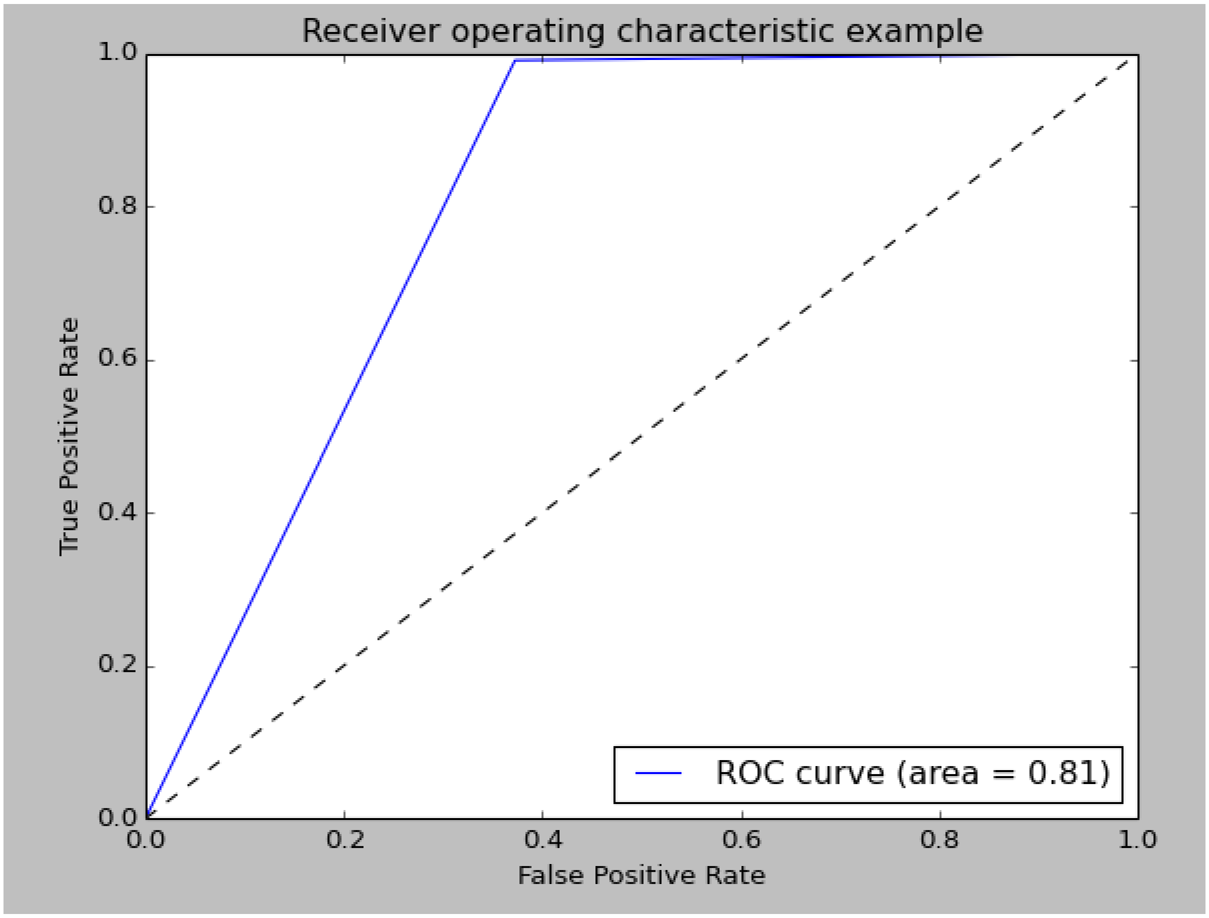
The model is being trained with training set.

Let us test our model

Accuracy 0.67662881052
Precision 0.293577981651
Recall0.991150442478
F measure 0.45298281092
Area under the ROC curve 0.809327812808

As we can see the accuracy is pretty low. This is because in Bernoulli we see the existence of the word not number of occurrence. As we go to binomial where we take word occurrences in to account we can see increase in accuracy

ROC curve



Let’s do **5-fold cross validation** to test our model further

5 fold cross validation
Accuracy 0.707029651158
Precision 0.312750640334
Recall0.990761610652
F measure 0.47520039036
Area under the ROC curve 0.826869546815

<u>Naive Bayes with Binomial features</u>

Dataset – I have used [sms](sms) dataset for Naive Bayes with Binomial features
Features – number of unique words in document

Classes – spam, ham

In Naive Bayes with Binomial features we see the number of occurrence of words

Assumption: features are not correlated

Let's train our model

Size of training set 70%

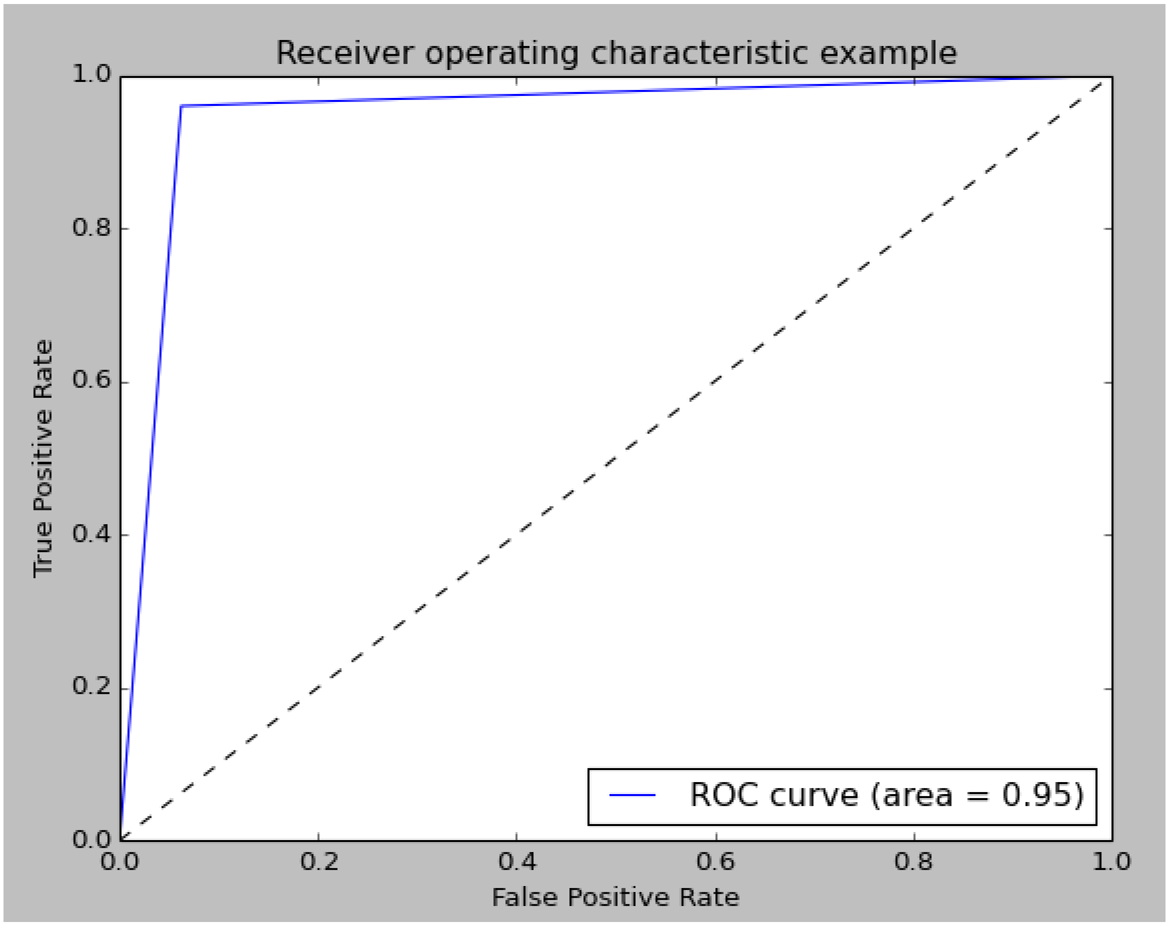
The model is being trained with training set.

Let us test our model

Accuracy 0.940824865511
Precision 0.706840390879
Recall0.96017699115
F measure 0.81425891182
Area under the ROC curve 0.94898967042

As we can see the accuracy increased to 94% (as in Bernoulli its 67%).
This is because we have taken occurrence of words in to account

ROC curve



Let’s do **5-fold cross validation** to test our model further

Accuracy 0.945997858483
Precision 0.724324176368
Recall0.963571706635
F measure 0.826598520684
Area under the ROC curve 0.953400052347

MLE for Binomial distribution

$$f(x) = \left(\frac{n!}{x!(n-x)!}\right) p^x (1-p)^{n-x}$$

$$L(p) = \prod_{i=1}^{n} f(x_i) = \prod_{i=1}^{n} \left(\frac{n!}{x_i!(n-x_i)!}\right) p^{x_i} (1-p)^{n-x_i}$$

$$L(p) = \left(\prod_{i=1}^{n} \left(\frac{n!}{x_i!(n-x)!}\right)\right) p^{\sum_{i=1}^{n} x_i} (1-p)^{n - \sum_{i=1}^{n} x_i}$$

Take log likelihood

$$\ln(L(p)) = \sum_{i=1}^{n} x_i \ln(p) + \left(n - \sum_{i=1}^{n} x_i\right) \ln(1-p)$$

Take derivative to find maximum

$$\frac{d \ln L(p)}{dp} = \frac{1}{p} \sum_{i=1}^{n} x_i + \frac{1}{1-p} \left(n - \sum_{i=1}^{n} x_i\right) = 0$$

$$(1-\hat{p}) \sum_{i=1}^{n} x_i + P\left(n - \sum_{i=1}^{n} x_i\right) = 0$$

$$\hat{p} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{K}{n}$$