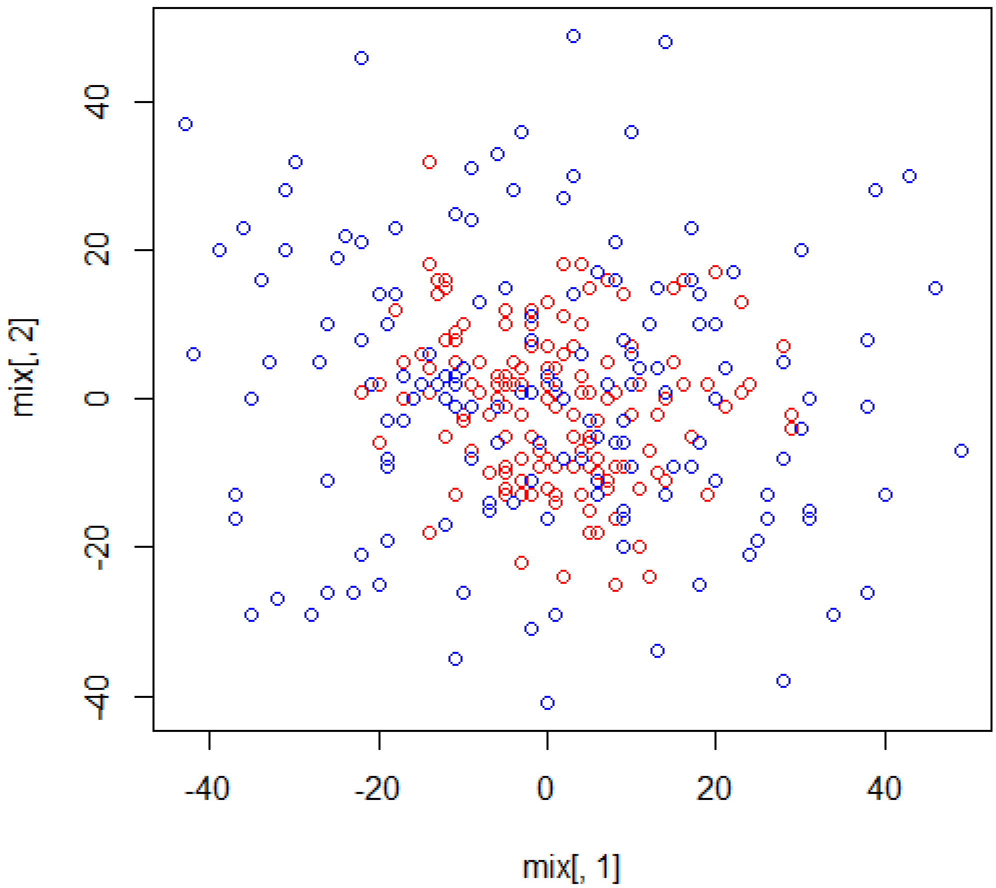
CSC 791 Graph Data Mining
Mahek Chheda (mschheda)

Project-3

Q.1] Plot for bad kmeans data set, SVM and PCA are same as this dataset acts bad for all 3 of them.



Q.2]

a. Metric for bad kmeans data

$Accuracy

[1] 0.4433333

$Error_Rate

[1] 0.5566667

$TPR

[1] 0.3666667

$TNR

[1] 0.52

$FPR

[1] 0.48

$FNR

[1] 0.6333333

$`Precision-1`

[1] 0.4330709

$`Precision-2`

[1] 0.4508671

$`F-measure-1`

[1] 0.3971119

$`F-measure-2`

[1] 0.4829721

b. Metric for Bad SVM data

```
n= 300

node), split, n, deviance, yval
      * denotes terminal node

 1) root 300 75.0000000 1.500000
   2) y< 18.5 262 64.3969500 1.435115
     4) x< 23.5 247 59.3198400 1.400810
       8) x>=-15.5 213 47.9812200 1.342723
        16) y>=-7.5 154 29.6103900 1.259740
          32) x< 9.5 117 19.6581200 1.213675 *
          33) x>=9.5 37  8.9189190 1.405405 *
        17) y< -7.5 59 14.5423700 1.559322
          34) x< 11.5 45 11.2444400 1.488889
            68) y>=-22.5 38  9.2631580 1.421053 *
            69) y< -22.5 7  0.8571429 1.857143 *
          35) x>=11.5 14  2.3571430 1.785714 *
       9) x< -15.5 34  6.1176470 1.764706
        18) x>=-26 23  5.2173910 1.652174
          36) x< -19.5 14  3.5000000 1.500000 *
          37) x>=-19.5 9  0.8888889 1.888889 *
        19) x< -26 11  0.0000000 2.000000 *
     5) x>=23.5 15  0.0000000 2.000000 *
   3) y>=18.5 38  1.8947370 1.947368 *
[1] "Confusion matrix for SVM Data prediction"

rpart.pred  1  2
```

```
    1 36 15
    2  1 23
[1] "Accuracy of SVM"
[1] 0.7866667
```

c. "Metric for Bad PCA data"

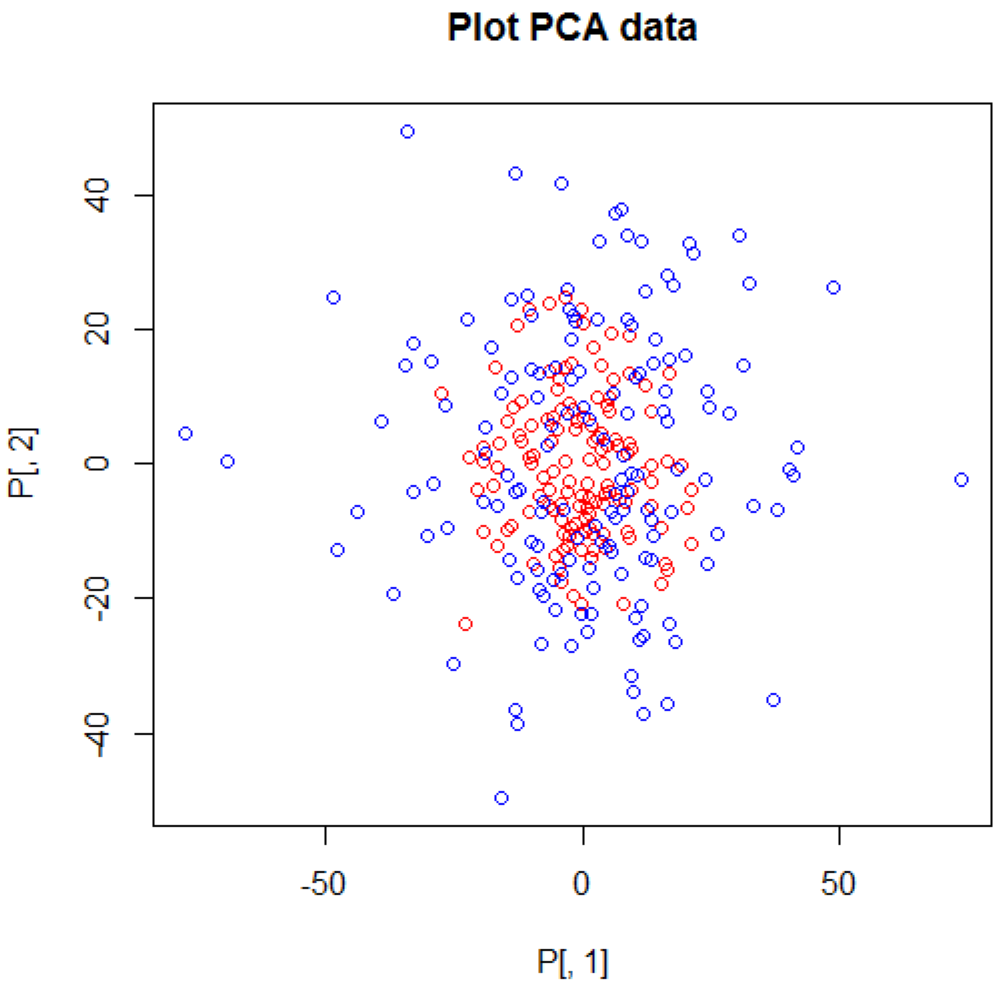
```
$Accuracy
[1] 0.5266667

$Error_Rate
[1] 0.4733333

$TPR
[1] 0.56

$TNR
[1] 0.4933333

$FPR
[1] 0.5066667

$FNR
[1] 0.44

$`Precision-1`
[1] 0.525

$`Precision-2`
[1] 0.5285714
```

$`F-measure-1`

[1] 0.5419355

$`F-measure-2`

[1] 0.5103448



Q.3]

[1] "Applying Radial basis kernel function on kmeans data sets"

Using automatic sigma estimation (sigest) for RBF or laplace kernel

$Accuracy

[1] 0.6466667

$Error_Rate

[1] 0.3533333

$TPR

[1] 0.54

$TNR

[1] 0.7533333

$FPR

[1] 0.2466667

$FNR

[1] 0.46

$`Precision-1`

[1] 0.6864407

$`Precision-2`

[1] 0.6208791

$`F-measure-1`

[1] 0.6044776

$`F-measure-2`

[1] 0.6807229

[1] "Applying Laplace Kernel Function on kmeans data sets"

Using automatic sigma estimation (sigest) for RBF or laplace kernel

$Accuracy

[1] 0.5433333

$Error_Rate

[1] 0.4566667

$TPR

[1] 0.5866667

$TNR

[1] 0.5

$FPR

[1] 0.5

$FNR

[1] 0.4133333

$`Precision-1`

[1] 0.5398773

$`Precision-2`

[1] 0.5474453

$`F-measure-1`

```
[1] 0.5623003

$`F-measure-2`
[1] 0.5226481

[1] "Applying Linear kernel function on bad data sets"
 Setting default kernel parameters
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Linear (vanilla) kernel function.

Number of Support Vectors : 174

Objective Function Value : -173.7002
Training error : 0.438889
[1] "Predict labels on test"
     ypred
ytest  2
    1 58
    2 62
[1] "Accuracy of the prediction"
[1] 0.5166667
[1] "Prediction scores"
      ypred
         2
  TRUE 120
[1] "Applying Gaussian kernel function on bad data sets"
```

```
Using automatic sigma estimation (sigest) for RBF or laplace kernel
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  1.9296580844319

Number of Support Vectors : 131

Objective Function Value : -99.5321
Training error : 0.211111
[1] "Predict labels on test"
     ypred
ytest  2
    1 58
    2 62
[1] "Accuracy of the prediction"
[1] 0.5166667
[1] "Prediction scores"
      ypred
         2
  TRUE 120
[1] "Applying kernel tricks on PCA bad data sets"
[1] "Applying Linear Gaussian radial basis function"
$Accuracy
[1] 0.56

$Error_Rate
```

[1] 0.44

$TPR

[1] 0.1866667

$TNR

[1] 0.9333333

$FPR

[1] 0.06666667

$FNR

[1] 0.8133333

$`Precision-1`

[1] 0.7368421

$`Precision-2`

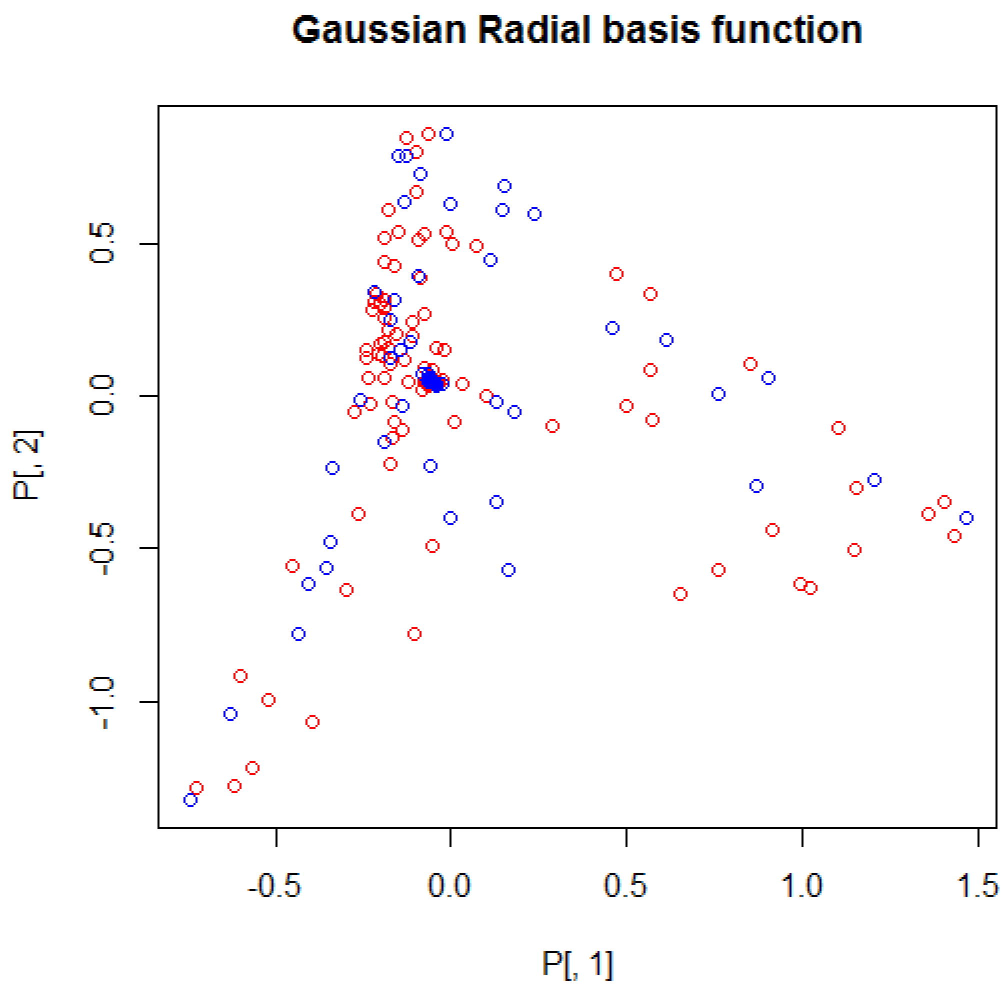
[1] 0.5343511

$`F-measure-1`

[1] 0.2978723

$`F-measure-2`

[1] 0.6796117



[1] "Applying Laplacian kernel basis function"

$Accuracy

[1] 0.5033333

$Error_Rate

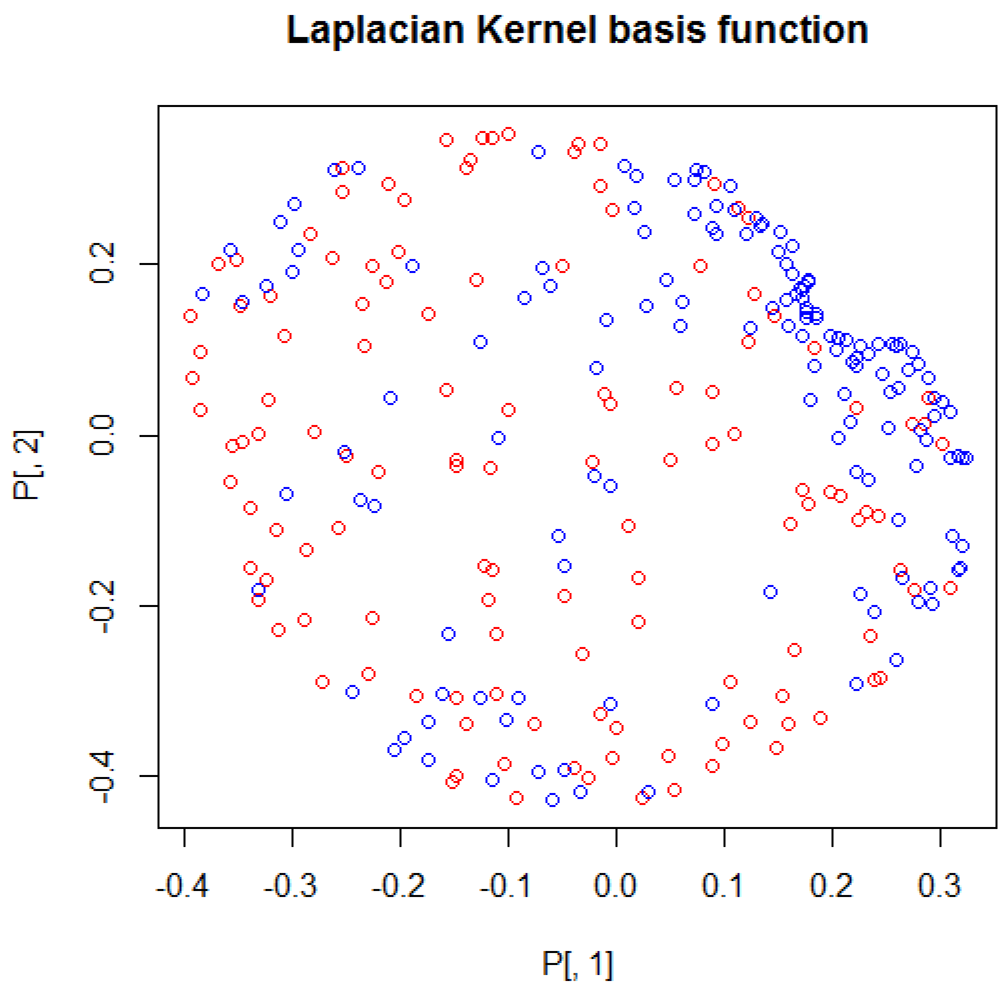
[1] 0.4966667

$TPR

[1] 0.006666667

$TNR

```
[1] 1

$FPR
[1] 0

$FNR
[1] 0.9933333

$`Precision-1`
[1] 1

$`Precision-2`
[1] 0.5016722

$`F-measure-1`
[1] 0.01324503

$`F-measure-2`
[1] 0.6681514
```



c.)

Yes, the performance of the functions increases by a good margin when we apply kernel tricks especially Gaussian radial basis function. The accuracy of the methods increases and this helps in getting better clusters of data set.

d.)

By playing with different kernel tricks in case of Kmeans there is a difference in Accuracy of the method. In Radial basis function accuracy comes out to be around 65% and for Laplacian dot function it comes out to be around 55%. In case of SVM, when I apply Linear kernel function I get around 72.5% support vector but for Radial basis function I get around 54%. In PCA, the value of Recall with True Negative rate is higher ~~1.

Q.4]

b. Performance of kmeans on high dimensional data.

[1] "Metric for kmeans data"

$Accuracy

[1] 0.45

$Error_Rate

[1] 0.55

$TPR

[1] 0.3533333

$TNR

[1] 0.5466667

$FPR

[1] 0.4533333

$FNR

[1] 0.6466667

$`Precision-1`

[1] 0.4380165

$`Precision-2`

[1] 0.4581006

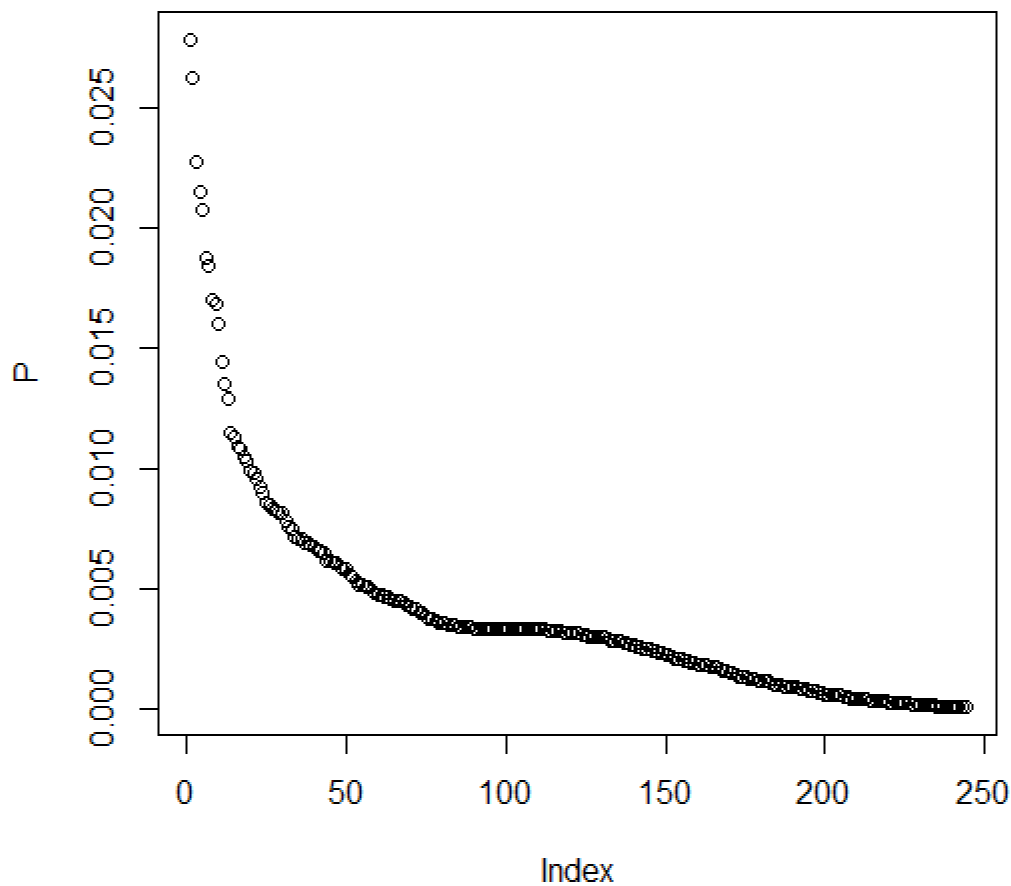
$`F-measure-1`

[1] 0.3911439

$`F-measure-2`

[1] 0.4984802

c.



The above graph is for getting plot of eigenvalues for each of the points. I took 75 point out of 244 which could cover energy or variance of upto 66%

[1] "Number of principal components"

[1] 75

[1] "Variability of data"

[1] 0.6626439

e.

Performance of d-dimension kmeans algorithm:

[1] "Metric for kmeans data"

$Accuracy

[1] 0.5366667

$Error_Rate

[1] 0.4633333

$TPR

[1] 0.52

$TNR

[1] 0.5533333

$FPR

[1] 0.4466667

$FNR

[1] 0.48

$`Precision-1`

[1] 0.537931

$`Precision-2`

[1] 0.5354839

$`F-measure-1`

[1] 0.5288136

$`F-measure-2`

[1] 0.5442623

Performance of m-dimension k-means algorithm:

$Accuracy

[1] 0.5066667

$Error_Rate

[1] 0.4933333

$TPR

[1] 0.98

$TNR

[1] 0.03333333

$FPR

[1] 0.9666667

$FNR

[1] 0.02

$`Precision-1`

[1] 0.5034247

$`Precision-2`

[1] 0.625

$`F-measure-1`

[1] 0.6651584

$`F-measure-2`

[1] 0.06329114

We see here the performance is almost the same but generally it would increase by good amount.

f.

The performance of kkmeans is:

$Accuracy

[1] 0.4866667

$Error_Rate

[1] 0.5133333

$TPR

[1] 0.5266667

$TNR

[1] 0.4466667

$FPR

[1] 0.5533333

$FNR

[1] 0.4733333

$`Precision-1`

[1] 0.4876543

$`Precision-2`

[1] 0.4855072

$`F-measure-1`

[1] 0.5064103

$`F-measure-2`

[1] 0.4652778

We see that the performance here of kkmeans is less in terms of Precision and Accuracy compared to kpca and then kmeans. Kkmeans can be better for data sets having less dimensions so there is less complexity in finding clusters but if the dimension is huge then in that case kpca would fetch better results by reducing the curse of dimensionality and then making it easier to perform kmeans clustering.

References:

1. http://cbio.ensmp.fr/~jvert/svn/tutorials/practical/svmbasic/svmbasic_notes.pdf
2. http://www.inside-r.org/packages/cran/kernlab/docs/kkmeans
3. http://www.jstatsoft.org/v15/i09/paper
4. http://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf
5. http://cbio.ensmp.fr/~jvert/svn/tutorials/practical/svmbasic/svmbasic_notes.pdf
6. http://cran.r-project.org/web/packages/kernlab/vignettes/kernlab.pdf