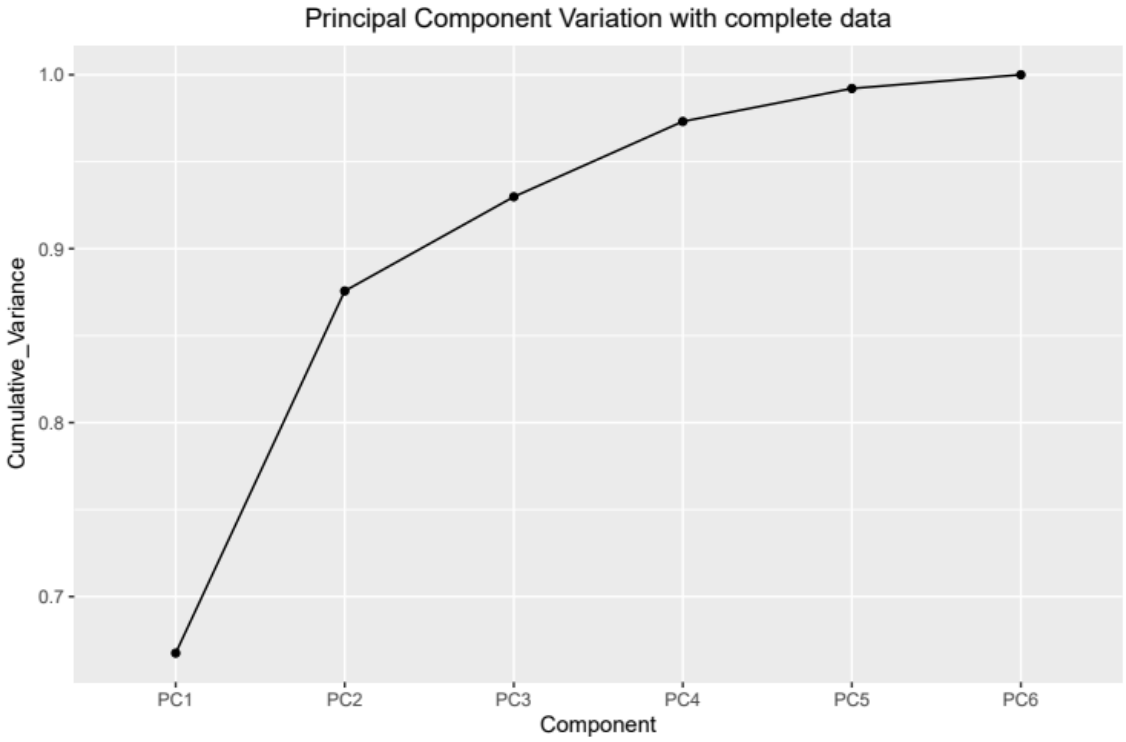
HW3

Sai Kiran Putta

Problem – 1

Performing Principal Component Analysis on the complete dataset and calculating the principal component variance for each component gives the following figure.
X axis shows the principal component and Y axis shows the relative cumulative variance.




```
> pca_importance
    Variance Cumulative_Variance Component
PC1  0.66752             0.66752       PC1
PC2  0.20816             0.87568       PC2
PC3  0.05416             0.92983       PC3
PC4  0.04331             0.97314       PC4
PC5  0.01896             0.99210       PC5
PC6  0.00790             1.00000       PC6
```

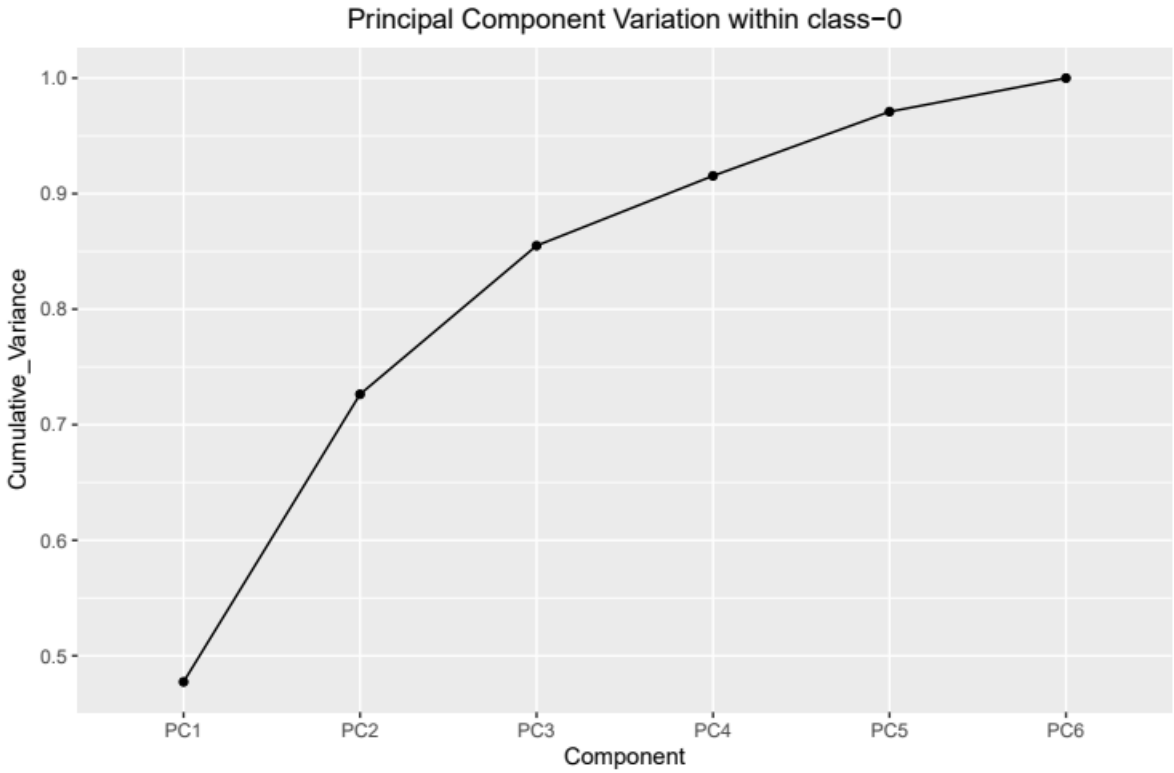
Inference: The first principal component constitutes 67% of variance in the data. The cumulative percent of variance explained till component – 2 is 88%.

In order to better understand the effect of all the features on the target using principal components gives the following figure.

Fig: sep

Inference: There is a clear separation between both the classes.

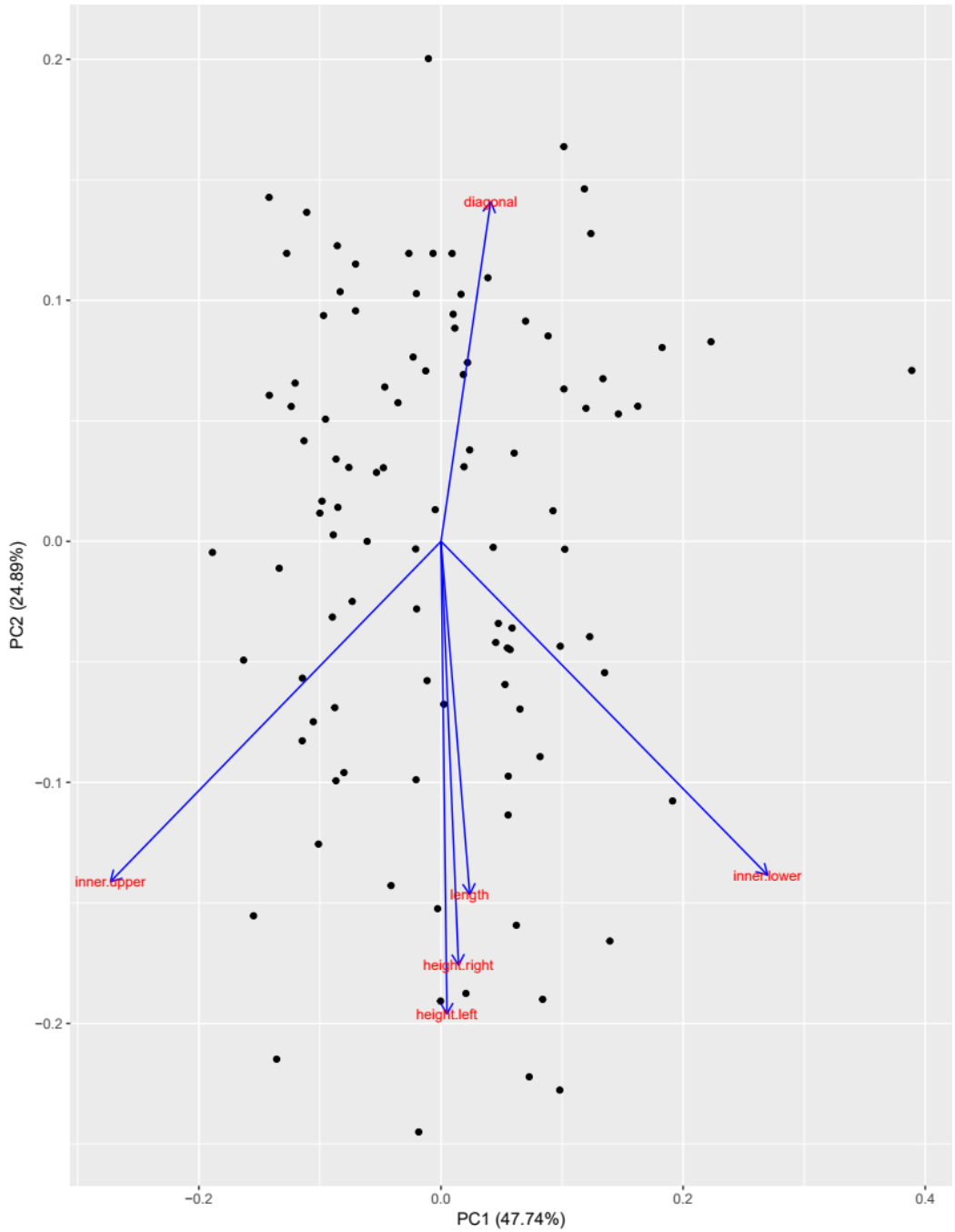
It can be observed from the above figure that from first two principal components it is from the features diagonal and (height.left, height.right) the variation in the classes occur. That is the principal component pertaining to class – 0 has a stronger effect because of feature diagonal and the principal component pertaining to class – 1 has a stronger effect because of features height.left and height.right.



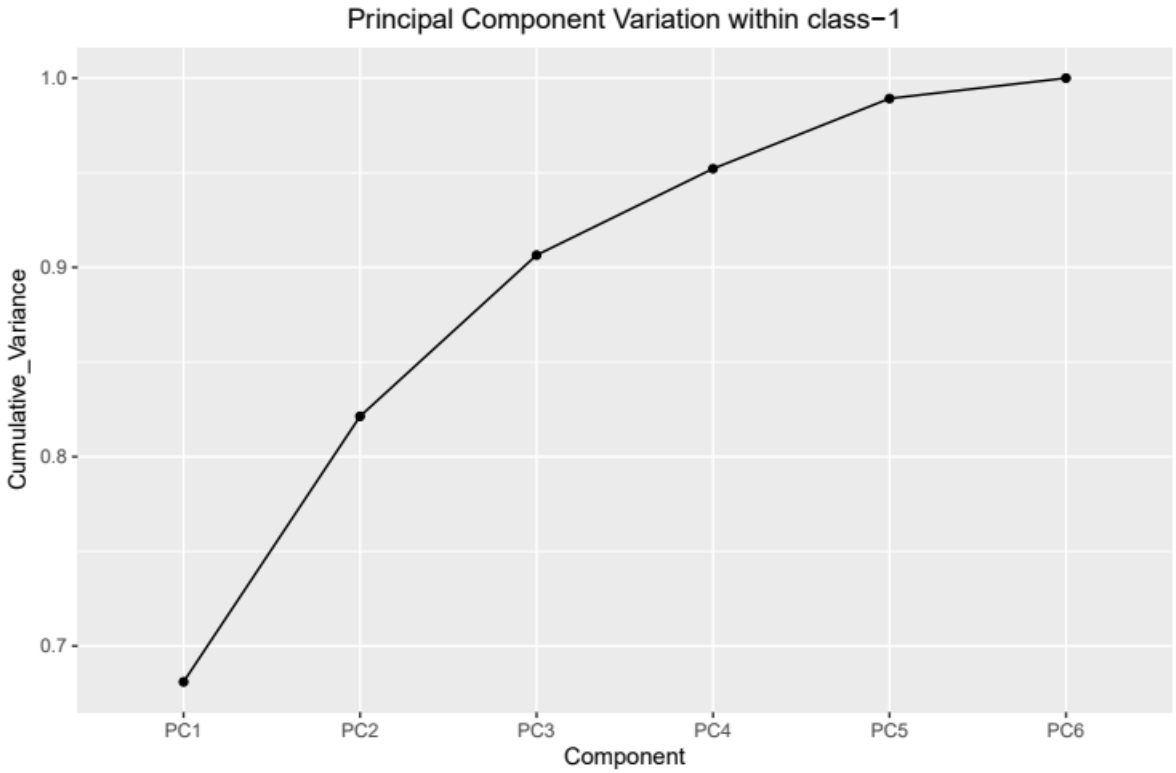

```
> pca_importance_0
    Variance Cumulative_Variance Component
PC1  0.47742              0.47742       PC1
PC2  0.24893              0.72635       PC2
PC3  0.12860              0.85495       PC3
PC4  0.06044              0.91539       PC4
PC5  0.05556              0.97095       PC5
PC6  0.02905              1.00000       PC6
>
```

Inference : Performing a PCA on only data of class – 0 gives us the above graph. The variance explained by the first two principal components have reduced. Indicating that ‘pattern’ in the data has changed – Both classes have different patterns in the data!

Feature variation in PCA for class – 0



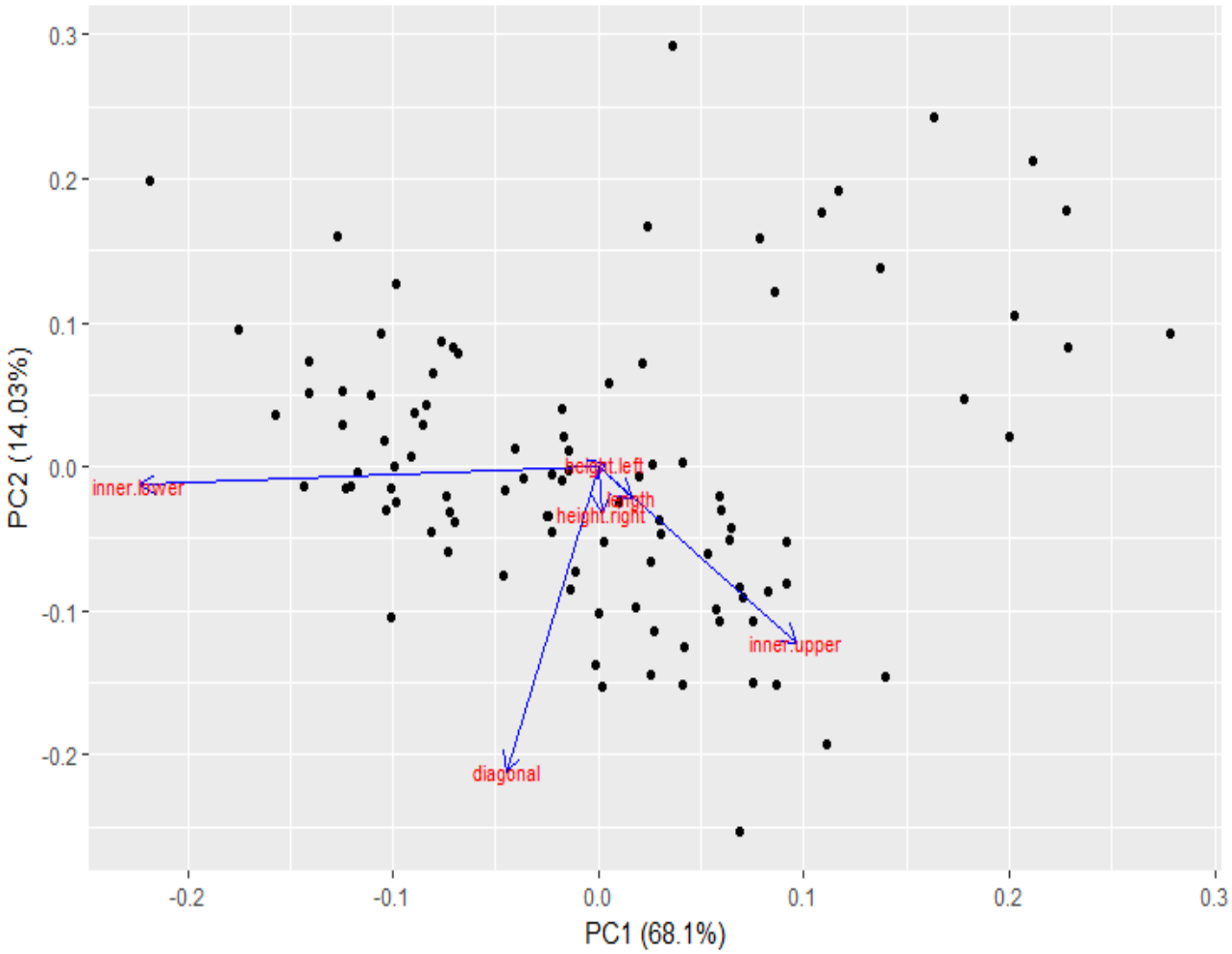
Inference : While length, height.right, height.left are aligned in as features, diagonal seems to be negatively correlated with the features mentioned above.




```
> pca_importance_1
    Variance Cumulative_Variance Component
PC1  0.68103             0.68103       PC1
PC2  0.14031             0.82133       PC2
PC3  0.08515             0.90648       PC3
PC4  0.04565             0.95213       PC4
PC5  0.03709             0.98922       PC5
PC6  0.01078             1.00000       PC6
>
```

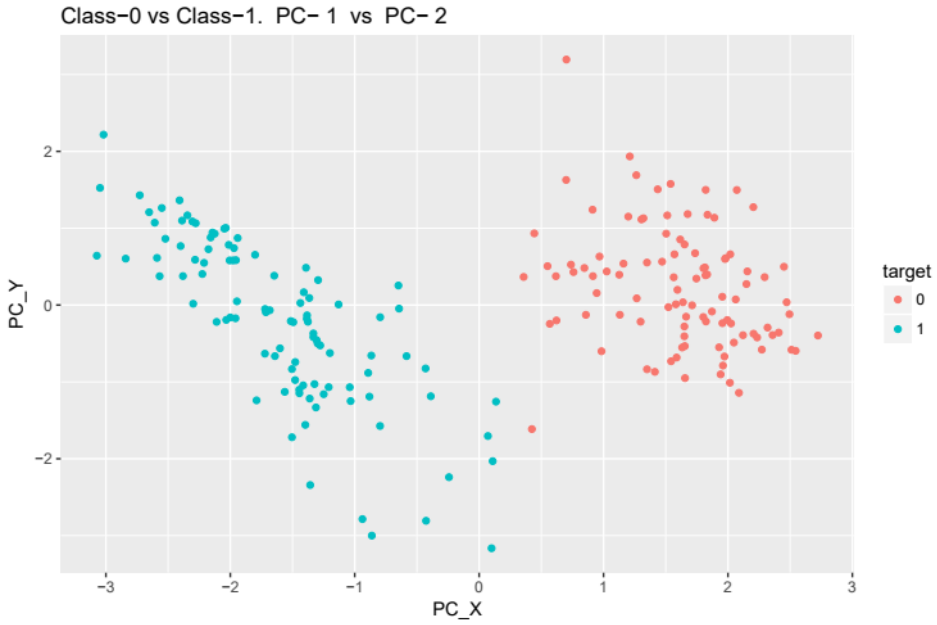
Inference: Performing a PCA on only data of class – 1 gives us the above graph. The variance explained by the first two principal components has differed. Indicating that ‘pattern’ in the data has changed – Both classes have different patterns in the data!

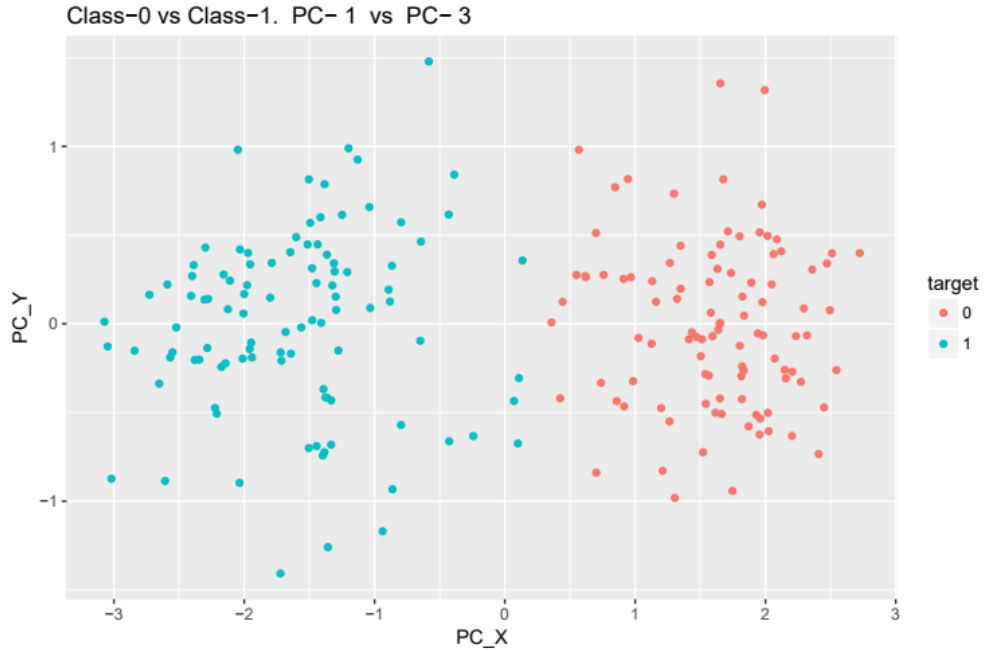
Feature variation in PCA for class – 1

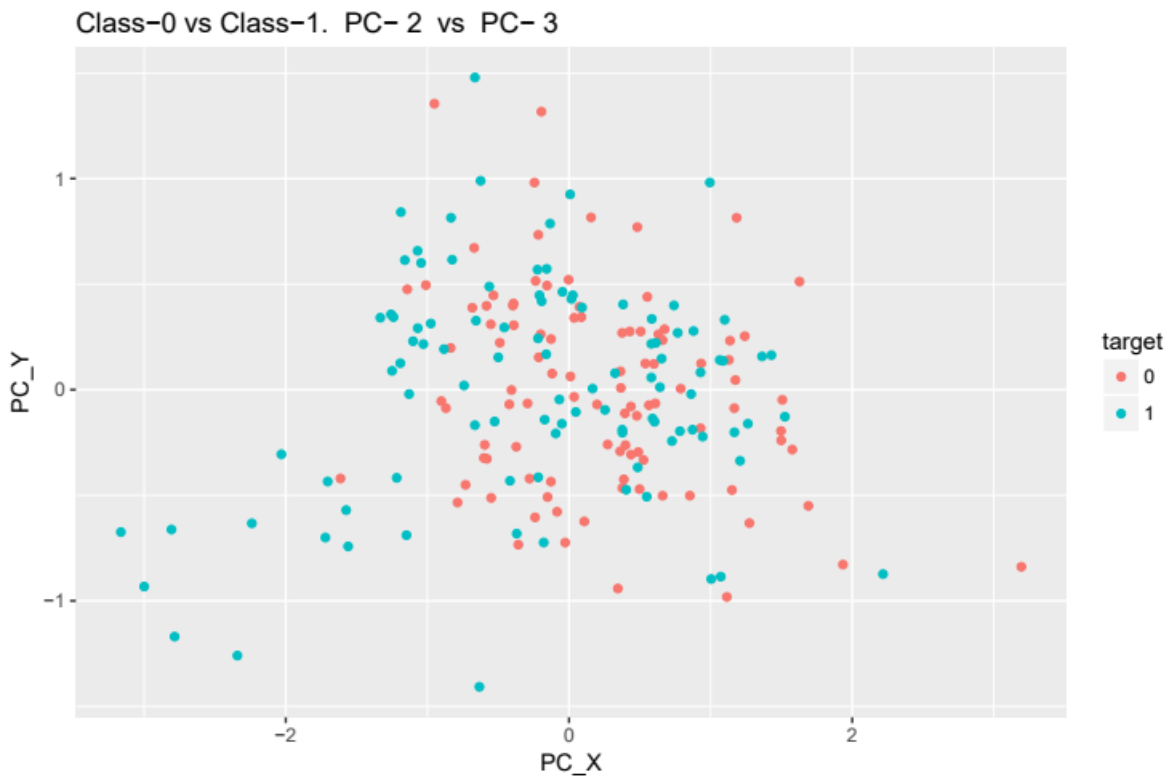


Inference: For further proof that the patterns in both the classes are different, we can clearly see the change in orientation of features for class – 1. They are completely different from what we observed for class – 0.

To better understand the split of classes within PCA, different combinations of principal components were plotted.
Segregation of classes were observed only when PC1 was involved, as below.







This is indicative that PC1 is capturing the information that is needed for separation of both the classes.

So what feature is causing the separation in classes?
Doing a correlation for all the features on PC1 gives us the following result.

```
length :  0.2013605
height.left :  -0.5381321
height.right :  -0.5966697
inner.lower :  -0.9212294
inner.upper :  -0.4352554
diagonal :  0.870232
```

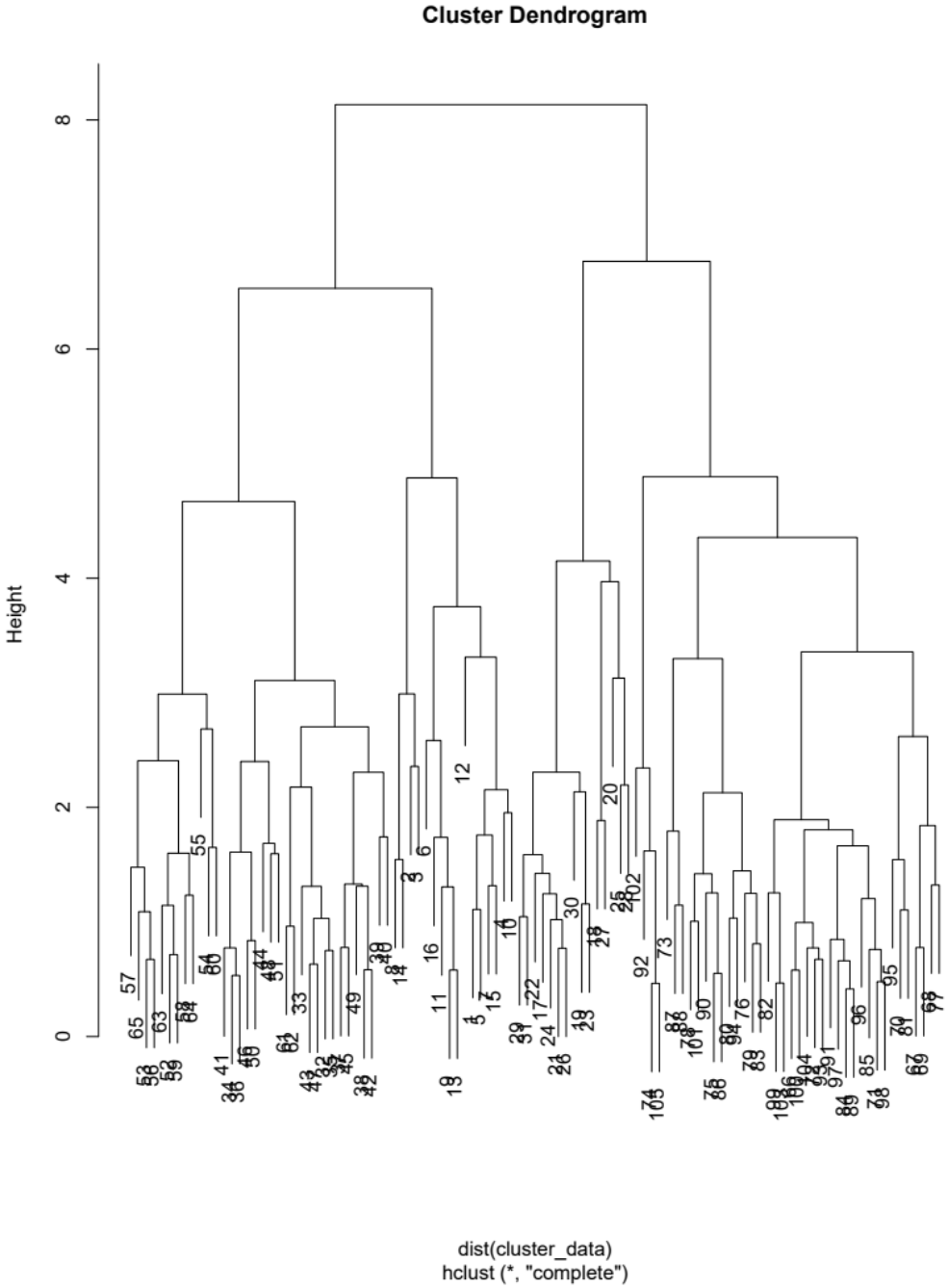
Inference: This is indicating that features inner.lower and diagonal is causing more separation in the classes.
The same can be observed in the fig:sep, where both these features are pulling both the classes apart.

Problem – 2

a)

Checking the sanity of the dataset, we see that there are some NA values in the feature gamma. Hence replacing the NA values in the feature gamma with median of gamma.

Fitting a hierarchical clustering on the dataset using **complete linkage** and plotting the figure gives the following.



We are going to cut the tree in order to get 5 clusters in total, since we already know that the total number of classes in the dataset is 5.

The confusion matrix of the above clustering looks like the following.

```
> table(dataset$classdigit, clusters_complete)
   clusters_complete
     1  2  3  4  5
  1 16  0  0  0  0
  2  0 15  0  0  0
  3  0  0 20  0  0
  4  0  0 14  0  0
  5  0  0  0 36  4
>
```

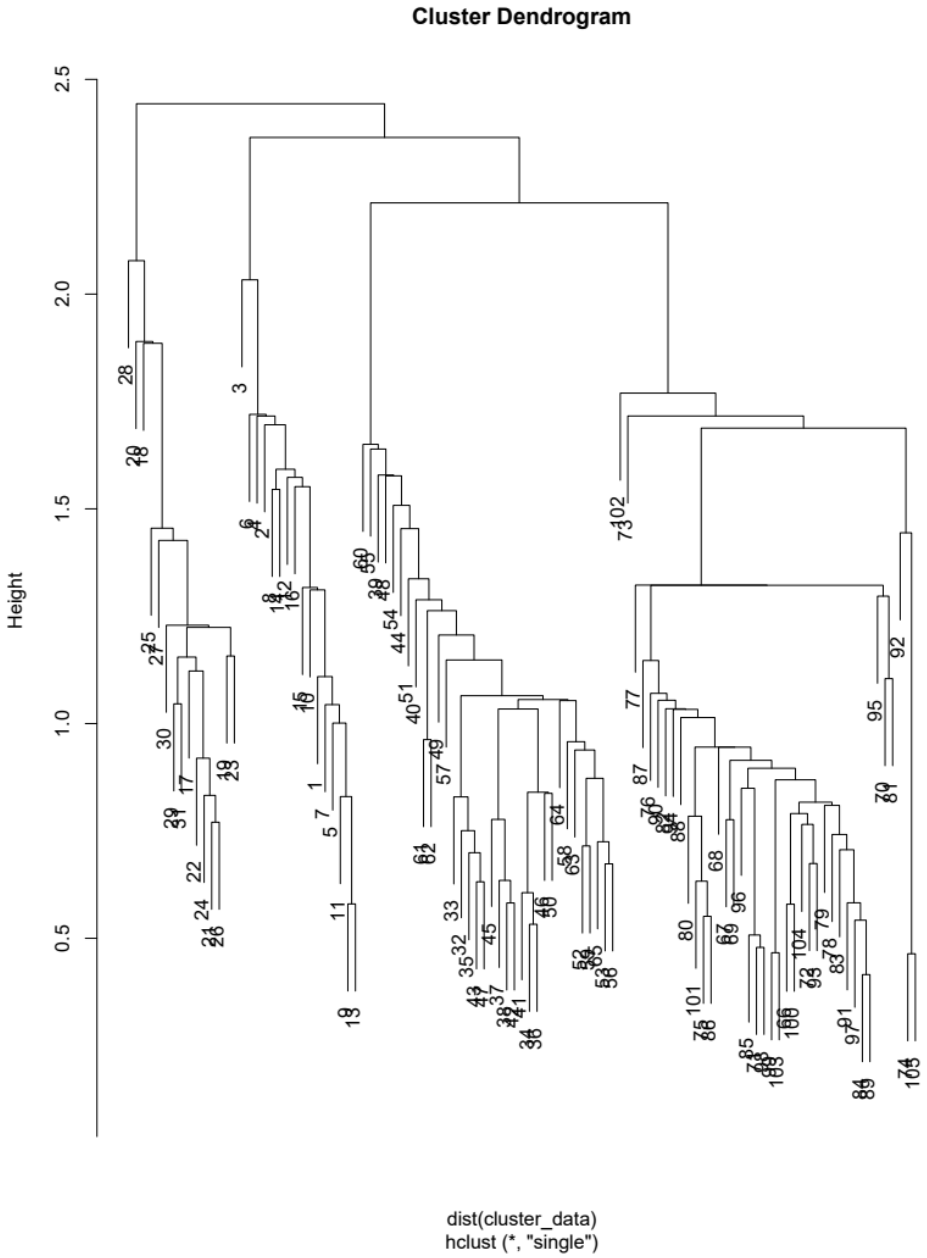
Following have to be noted in the unsupervised learning – in the clustering naming that, after the clusters are formed the naming can be given like we wish. That is for example, in order to optimize the misclassification rate, we would be better off if cluster 4 was named cluster 5 and vice versa.
Hence doing the same would give us the following confusion matrix.

```
> table(dataset$classdigit, clusters_complete)
   clusters_complete
     1  2  3  4  5
  1 16  0  0  0  0
  2  0 15  0  0  0
  3  0  0 20  0  0
  4  0  0 14  0  0
  5  0  0  0  4 36
```

We obtain the misclassification rate of the above confusion matrix as – 17.14%

For further analysis of confusion matrix, the same will be followed. That is the cluster assignments are altered to optimize the misclassification rate.

Fitting a hierarchical clustering on the dataset using **single linkage** and plotting the figure gives the following.

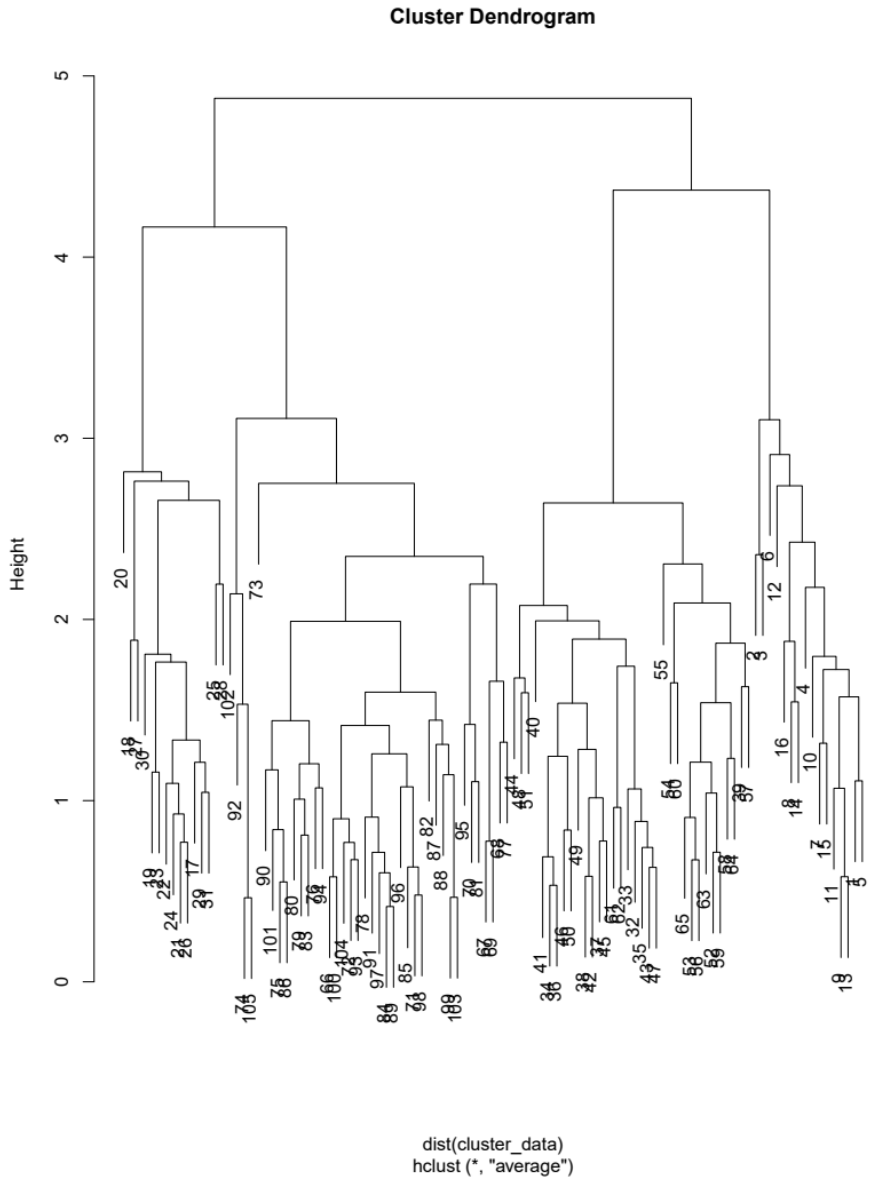


By cutting the above structure in order to get 5 clusters, we get the following confusion matrix.

```
> table(dataset$classdigit, clusters_single)
   clusters_single
      1  2  3  4  5
  1 16  0  0  0  0
  2  0 14  0  1  0
  3  0  0 20  0  0
  4  0  0 14  0  0
  5  0  0  0  0 40
```

The misclassification rate of the above confusion matrix is – 14.28%

Fitting a hierarchical clustering on the dataset using **average linkage** and plotting the figure gives the following.



By cutting the above structure to get 5 clusters we get the following confusion matrix.

```
> table(dataset$classdigit, clusters_average)
   clusters_average
     1  2  3  4  5
  1 16  0  0  0  0
  2  0 15  0  0  0
  3  0  0 20  0  0
  4  0  0 14  0  0
  5  0  0  0  4 36
```

The misclassification rate of the above confusion matrix is – 17.14%

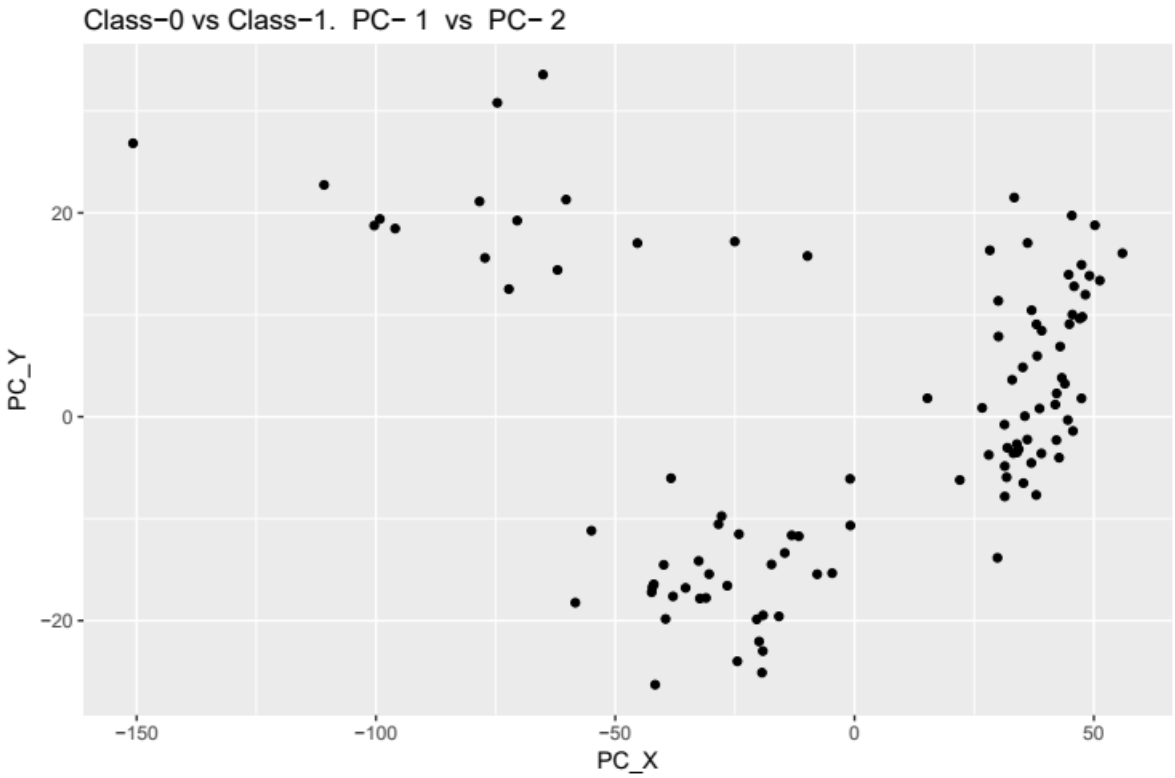
Hence for this dataset, for hierarchical clustering we find single linkage to be the best linkage method to optimize the overall misclassification rate.
The performance of other two linkages are the same.

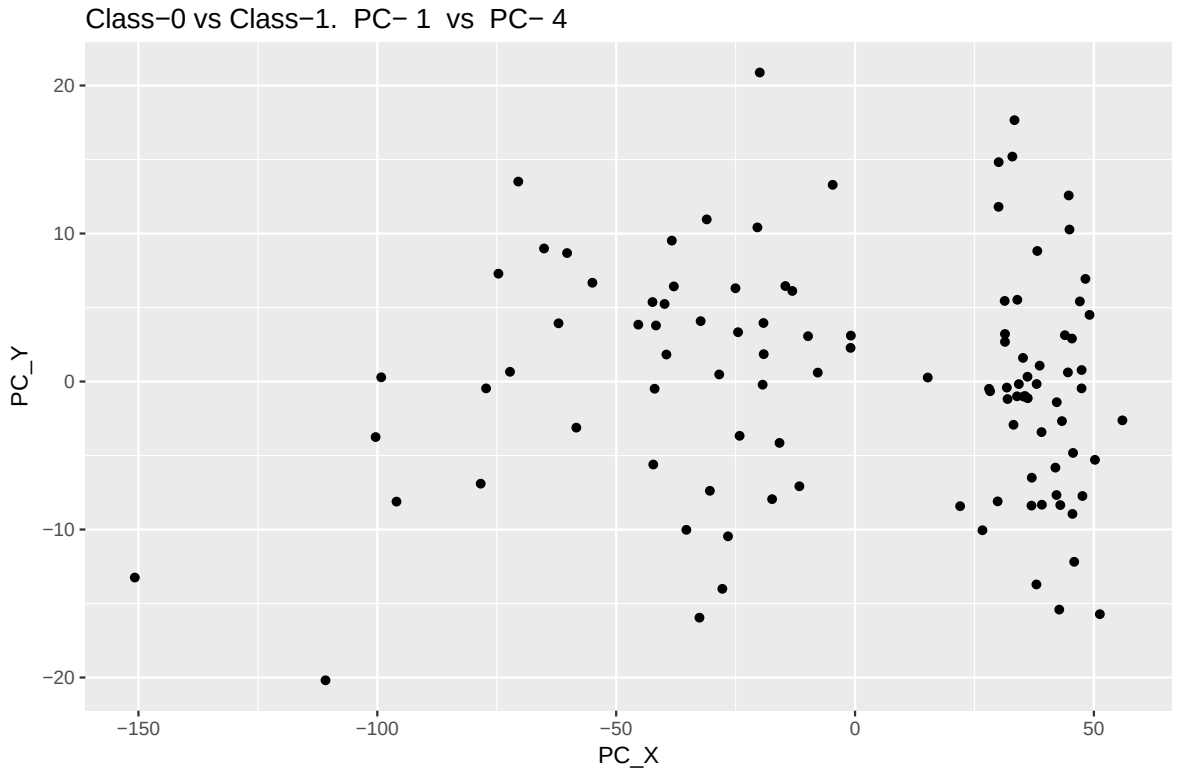
The choice of linkage is always dependent on the dataset we have in hand. Hence we cannot have prejudice about the type of linkage that would work for everything. Hence it is always recommended to try all the linkages while doing hierarchical clustering.

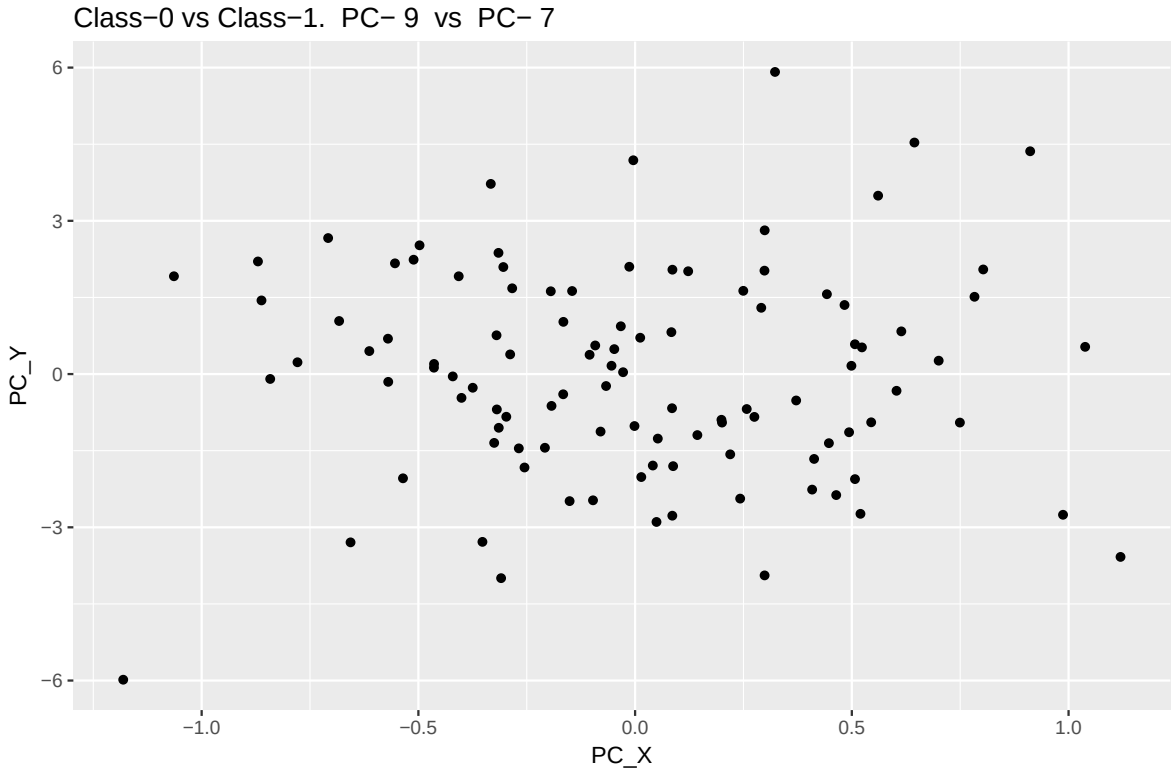
b)
One of the primary reasons we would go for k-mediods rather than k-means is when we suspect if there are any outliers in the dataset since it uses manhatten distance as a distance metric which is more robust to outliers.

Hence let us first try to understand if there are any outliers in the dataset. By running PCA on the dataset and plotting principal components we can arrive at the conclusion if there are outliers.

Checking different combinations of principal component figures like below, we can conclude that there are outliers in the dataset.







By inspecting the plots we can conclude that there are outliers in the dataset. Hence we are choosing to use k-mediods instead of k-means, since it is more robust to outliers.

Following is the confusion matrix we obtain after performing k-mediods on the dataset.

```
> table(dataset$classdigit, cluster_result)
   cluster_result
      1  2  3  4  5
  1   0  1 15  0  0
  2   0 15  0  0  0
  3   0  0 20  0  0
  4   0  1 13  0  0
  5 12  0  0  8 20
```

The misclassification rate of the above confusion matrix is – 47.61%

Hence overall, for this dataset, hierarchical clustering with single linkage gives us the best results.

Problem – 3

Fitting the dataset with k = 2 and iterating over different values (0.1, 1, 10, 50, 100, 500, 1000) of Radius for fitting SOMs, we get the following results.

```
k: 2
[1] "KMeans:"

   2    1
1305 5525
Self Organising Map. Radius:  0.1
   2    1
1319 5511
Self Organising Map. Radius:  1
   2    1
1224 5606
Self Organising Map. Radius:  10
   1    2
1318 5512
Self Organising Map. Radius:  50
   1    2
1311 5519
Self Organising Map. Radius:  100
   2    1
1327 5503
Self Organising Map. Radius:  500
   1    2
1240 5590
Self Organising Map. Radius:  1000
   2    1
1225 5605
```

Fitting the dataset with k = 5 and iterating over different values (0.1, 1, 10, 50, 100, 500, 1000) of Radius for fitting SOMs, we get the following results.

```
k: 5
[1] "KMeans:"

   2    1    5    4    3
 277  391  542 1833 3787
Self Organising Map. Radius:  0.1
   1    5    4    3    2
 272  350  624 1820 3764
Self Organising Map. Radius:  1
   1    5    4    2    3
 268  374  576 1899 3713
Self Organising Map. Radius:  10
   5    1    3    2    4
 219  338  574 1784 3915
Self Organising Map. Radius:  50
   2    5    4    3    1
 279  412  650 1840 3649
Self Organising Map. Radius:  100
   5    1    3    2    4
 342  355  474 1722 3937
Self Organising Map. Radius:  500
   5    1    3    2    4
 271  408  459 1873 3819
Self Organising Map. Radius:  1000
   5    1    2    3    4
 237  323  533 1617 4120
```

Fitting the dataset with k = 10 and iterating over different values (0.1, 1, 10, 50, 100, 500, 1000) of Radius for fitting SOMs, we get the following results.

```
k: 10
[1] "KMeans:"

   6    5    8    1    7    3   10    2    4    9
   8  110  140  221  318  328 1226 1356 1424 1699
Self Organising Map. Radius:  0.1
   3    6    1   10    9    2    7    4    8    5
  71   75  148  214  270  424 1017 1122 1294 2195
Self Organising Map. Radius:  1
   6    8    1   10    2    3    7    4    9    5
  74  104  122  205  273  366 1044 1206 1245 2191
Self Organising Map. Radius:  10
   5    8   10    1    2    7    9    3    4    6
  77   87  122  201  248  538  747 1307 1453 2050
Self Organising Map. Radius:  50
   5    8   10    1    2    9    7    4    3    6
  74   82  125  218  277  375 1103 1206 1253 2117
Self Organising Map. Radius:  100
   3    1   10    8    9    5    2    6    7    4
  74   80  118  209  266  305 1026 1213 1310 2229
Self Organising Map. Radius:  500
   1    5   10    8    9    4    6    2    7    3
  74  104  126  195  248  372  822 1182 1332 2375
Self Organising Map. Radius:  1000
   6    3   10    1    2    9    8    4    7    5
  71   78  139  216  225  402 1080 1217 1297 2105
```

Fitting the dataset with k = 20 and iterating over different values (0.1, 1, 10, 50, 100, 500, 1000) of Radius for fitting SOMs, we get the following results.

```
k: 20
[1] "KMeans:"

 11   8   9  17   1   5  12  16  15  10  20  19  18   7  13   3   6   2  14   4
 16  37  55  66  72  85  90 101 116 118 173 218 249 525 703 781 820 846 878 881
Self Organising Map. Radius:  0.1
   4   20   14    1    2   18    5    8    3    7    6   19   17   15   16    9   10   11   13   12
  35   44   64   70   83   93  107  107  113  128  170  176  272  441  645  707  709  866  886 1114
Self Organising Map. Radius:  1
  1  20  10  15  18   8   4  19   7   3   9   2  17   6  11  16   5  13  12  14
 40  44  46  68  84  95 100 106 135 138 171 206 278 556 675 771 793 794 840 890
Self Organising Map. Radius:  10
  4  19  14  17   1   2   8  18  20   3   7   5   9  10   6  13  15  16  11  12
 43  43  66  70  80  92  98 139 142 163 170 215 306 630 653 694 738 741 797 950
Self Organising Map. Radius:  50
  1  20   7  18   4   2   3  15   5  13  19  16  14   6  17  12   9   8  10  11
 42  49  63  86  93  99 101 102 134 146 146 353 376 382 562 685 807 834 843 927
Self Organising Map. Radius:  100
  4   1  20  11   6  18   3  19   5   2  13  15  14  17  10  12  16   9   7   8
 44  45  45  57  98 113 117 125 135 142 145 309 361 378 652 685 748 773 918 940
Self Organising Map. Radius:  500
  4  20  13   2   1   8  19   5  18  14   3  11   6  16  17  15  10   9   7  12
 29  36  42  47  56  62 101 118 143 152 176 284 389 543 649 728 764 803 850 858
Self Organising Map. Radius:  1000
 20  14   1   7  13   2  16   3  15   5  17  19   4   9  18   6  11  12   8  10
 33  55  60  64  86  93  99 108 129 169 205 209 321 367 683 715 727 859 911 937
```

Like in the previous problem, since the cluster numbering can be randomly assigned, we are sorting the clusters based on number of observations that are there in a cluster.

By inspecting all the tables above we can conclude that the SOM solution becomes more similar to the K-Means solution as the size of SOM neighborhood is taken to be smaller and smaller.