ASSIGNMENT 4

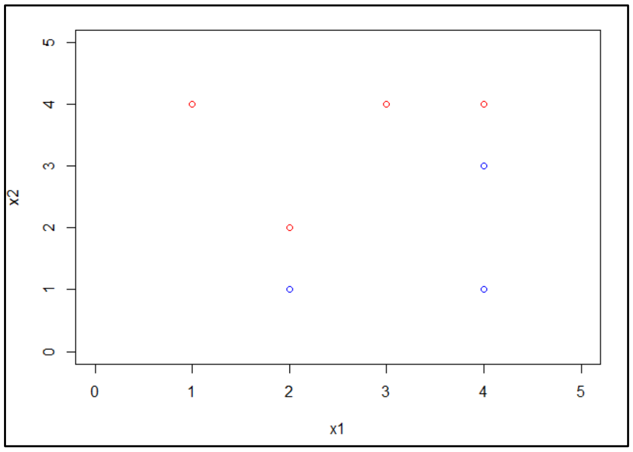
Ghorpade, Mrunal Mahesh
Mghorp2@uic.edu

1) Here we explore the maximal margin classifier on a toy data set.

(a) We are given *n* = 7 observations in *p* = 2 dimensions. For each observation, there is an associated class label. Sketch the observations.

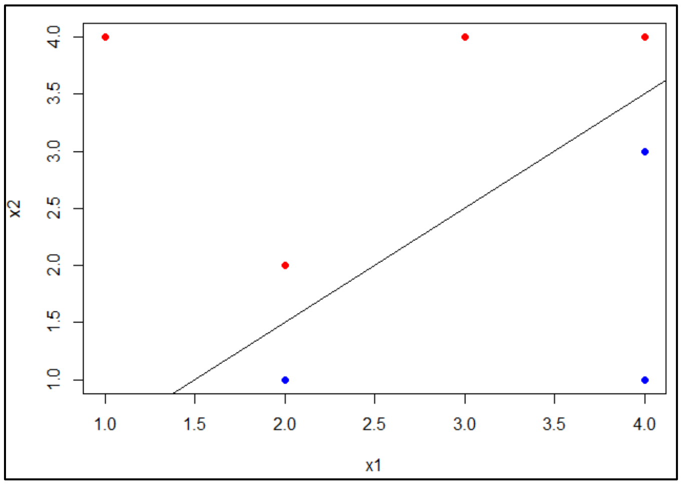
Obs.	X1	X2	Y
1	3	4	Red
2	2	2	Red
3	4	4	Red
4	1	4	Red
5	2	1	Blue
6	4	3	Blue
7	4	1	Blue

Ans: Plot for the above observation is as follows:

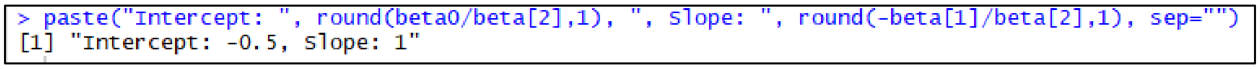


(b) Sketch the optimal separating hyperplane, and provide the equation for this hyperplane (of the form (9.1)).

Ans: Below plot shows the optimal separating:



Following are the intercepts and slope of the hyper plane:



```
> paste("Intercept: ", round(beta0/beta[2],1), ", Slope: ", round(-beta[1]/beta[2],1), sep="")
[1] "Intercept: -0.5, Slope: 1"
```

There fore the equation of the line can be given as: -0.5 + X1 - X2 = 0

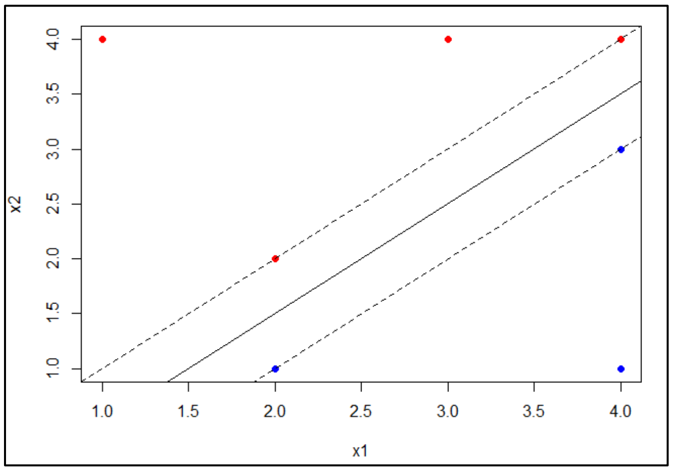
(c) Describe the classification rule for the maximal margin classifier. It should be something along the lines of "Classify to Red if$\beta 0$ +$\beta 1X1$ +$\beta 2X2$ > 0, and classify to Blue otherwise." Provide the values for $\beta 0$, $\beta 1$, and $\beta 2$.

Ans: Classify to Red if -0.5 + X1 - X2 > 0 else classify to Blue

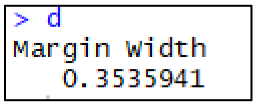
Where $\beta 0$= -0.5, $\beta 1$ = 1, and $\beta 2$. = -1

(d) On your sketch, indicate the margin for the maximal margin hyperplane.

Ans: The maximal margin hyperplane is the separating hyperplane for which the margin is largest i.e., it is the hyperplane that has the farthest minimum distance to the training observations.



The solid line shown in above plot is the maximal margin hyperplane. The distance from the dashed line to the solid line will give us the margin width which in this case is as given below:



```
> d
Margin Width
   0.3535941
```

(e) Indicate the support vectors for the maximal margin classifier.

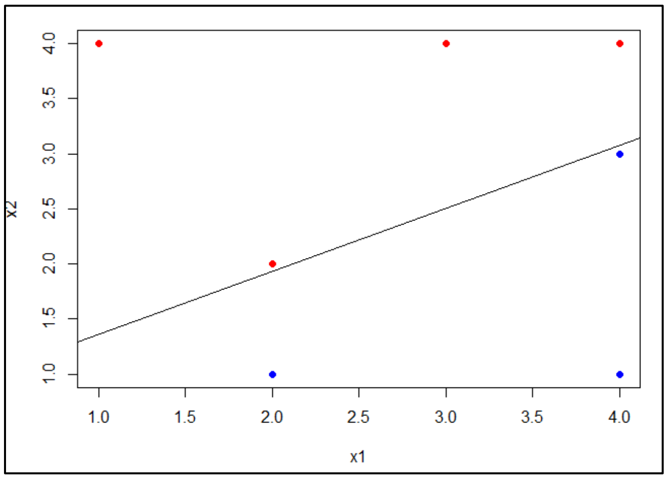
Observation: The support vectors for the maximal margin classifier are points represented as Blue: (2,1) and (4,3) and Red: (2,2) and (4,4)

(f) Argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane.

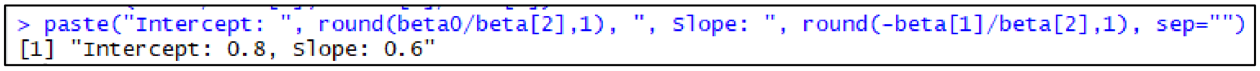
Observation: The maximal margin hyperplane will only be affected if any of the support vectors are changed. As seventh observation, (4,1) is not an support vector so even if we change the observation it will not affect the maximal margin hyperplane.

(g) Sketch a hyperplane that is *not* the optimal separating hyperplane, and provide the equation for this hyperplane.

Ans: Following is the separating hyperplane which is not the optimal separating hyperplane:



Following are the intercepts and slope of the hyper plane:

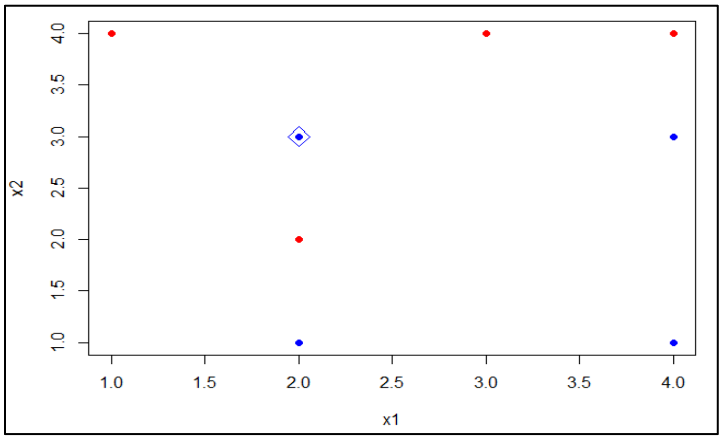


```
> paste("Intercept: ", round(beta0/beta[2],1), ", Slope: ", round(-beta[1]/beta[2],1), sep="")
[1] "Intercept: 0.8, Slope: 0.6"
```

Equation for this is: $0.8 + 0.6\,X1 - X2 = 0$

(h) Draw an additional observation on the plot so that the two classes are no longer separable by a hyperplane.

Ans: When a blue point (2,3) as shown below, is added to the observations, the two classes are no longer separated by a hyperplane

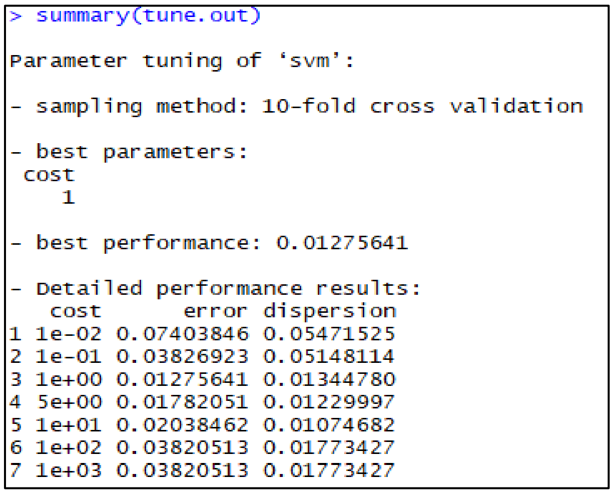


2) In this problem, you will use support vector approaches in order to predict whether a given car gets high or low gas mileage based on the Auto data set.

(a) Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median. (Please refer to R code- Problem 7)

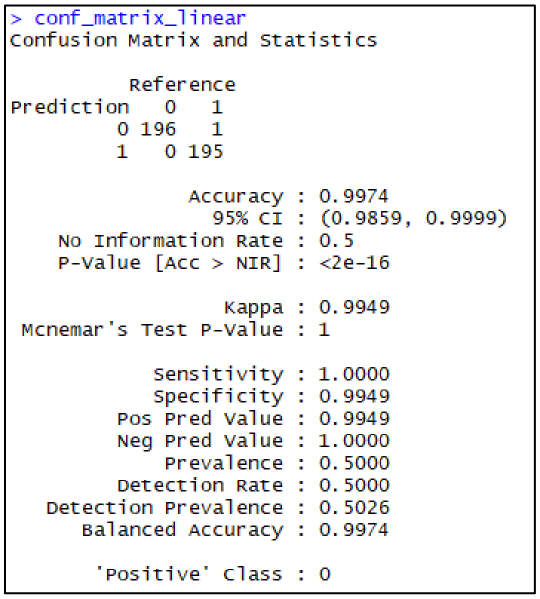
(b) Fit a support vector classifier to the data with various values of cost, in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results.

Ans: Below are the results for different values of cost and their corresponding errors:



```
> summary(tune.out)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
    1

- best performance: 0.01275641

- Detailed performance results:
   cost      error dispersion
1 1e-02 0.07403846 0.05471525
2 1e-01 0.03826923 0.05148114
3 1e+00 0.01275641 0.01344780
4 5e+00 0.01782051 0.01229997
5 1e+01 0.02038462 0.01074682
6 1e+02 0.03820513 0.01773427
7 1e+03 0.03820513 0.01773427
```

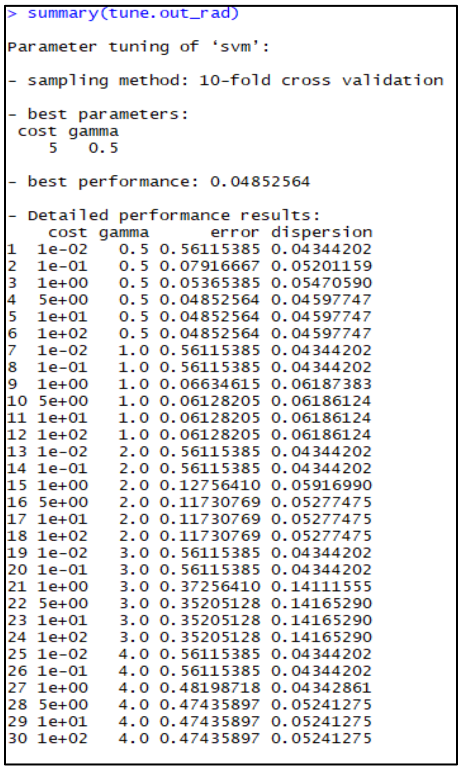
Observation: The best choice of parameters involves Cost =1 as it has the least cross validation error i.e. 0.0127.
Below is the confusion matrix and accuracy by using best parameters and the accuracy = 99.74%;



```
> conf_matrix_linear
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 196   1
         1   0 195

               Accuracy : 0.9974
                 95% CI : (0.9859, 0.9999)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9949
 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9949
         Pos Pred Value : 0.9949
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5026
      Balanced Accuracy : 0.9974

       'Positive' Class : 0
```

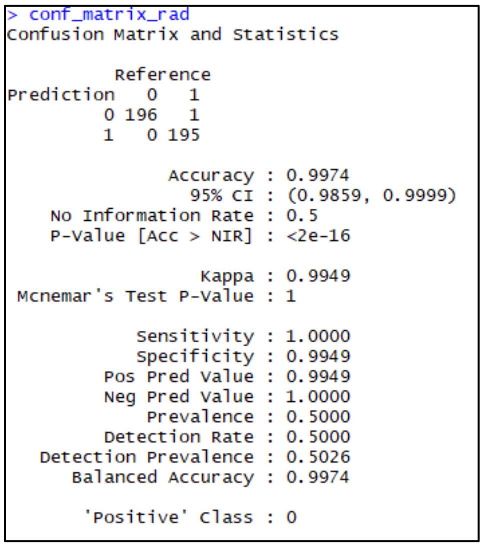
(c) Now repeat (b), this time using SVMs with radial and polynomial basis kernels, with different values of gamma and degree and cost. Comment on your results.

Ans: Following are the results of SVM with radial basis kernel:



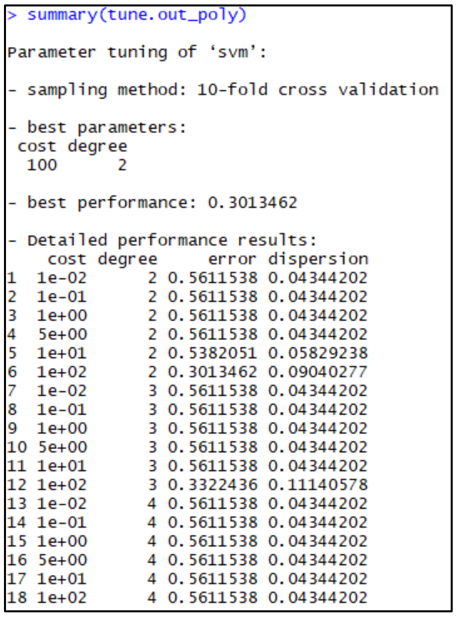
```
> summary(tune.out_rad)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    5   0.5

- best performance: 0.04852564

- Detailed performance results:
    cost gamma      error dispersion
1  1e-02   0.5 0.56115385 0.04344202
2  1e-01   0.5 0.07916667 0.05201159
3  1e+00   0.5 0.05365385 0.05470590
4  5e+00   0.5 0.04852564 0.04597747
5  1e+01   0.5 0.04852564 0.04597747
6  1e+02   0.5 0.04852564 0.04597747
7  1e-02   1.0 0.56115385 0.04344202
8  1e-01   1.0 0.56115385 0.04344202
9  1e+00   1.0 0.06634615 0.06187383
10 5e+00   1.0 0.06128205 0.06186124
11 1e+01   1.0 0.06128205 0.06186124
12 1e+02   1.0 0.06128205 0.06186124
13 1e-02   2.0 0.56115385 0.04344202
14 1e-01   2.0 0.56115385 0.04344202
15 1e+00   2.0 0.12756410 0.05916990
16 5e+00   2.0 0.11730769 0.05277475
17 1e+01   2.0 0.11730769 0.05277475
18 1e+02   2.0 0.11730769 0.05277475
19 1e-02   3.0 0.56115385 0.04344202
20 1e-01   3.0 0.56115385 0.04344202
21 1e+00   3.0 0.37256410 0.14111555
22 5e+00   3.0 0.35205128 0.14165290
23 1e+01   3.0 0.35205128 0.14165290
24 1e+02   3.0 0.35205128 0.14165290
25 1e-02   4.0 0.56115385 0.04344202
26 1e-01   4.0 0.56115385 0.04344202
27 1e+00   4.0 0.48198718 0.04342861
28 5e+00   4.0 0.47435897 0.05241275
29 1e+01   4.0 0.47435897 0.05241275
30 1e+02   4.0 0.47435897 0.05241275
```

For a radial kernel, the lowest cross-validation error i.e. 0.048 is obtained for a gamma of 0.5 and a cost of 5.
Below is the confusion matrix and accuracy by using best parameters and the accuracy = 99.74%;



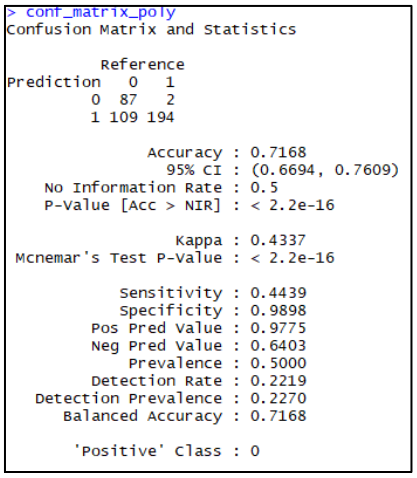
```
> conf_matrix_rad
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 196   1
         1   0 195

               Accuracy : 0.9974
                 95% CI : (0.9859, 0.9999)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9949
 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9949
         Pos Pred Value : 0.9949
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5026
      Balanced Accuracy : 0.9974

       'Positive' Class : 0
```

Following are the results of SVM with Polynomial basis kernel:



```
> summary(tune.out_poly)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost degree
  100      2

- best performance: 0.3013462

- Detailed performance results:
    cost degree     error dispersion
1  1e-02      2 0.5611538 0.04344202
2  1e-01      2 0.5611538 0.04344202
3  1e+00      2 0.5611538 0.04344202
4  5e+00      2 0.5611538 0.04344202
5  1e+01      2 0.5382051 0.05829238
6  1e+02      2 0.3013462 0.09040277
7  1e-02      3 0.5611538 0.04344202
8  1e-01      3 0.5611538 0.04344202
9  1e+00      3 0.5611538 0.04344202
10 5e+00      3 0.5611538 0.04344202
11 1e+01      3 0.5611538 0.04344202
12 1e+02      3 0.3322436 0.11140578
13 1e-02      4 0.5611538 0.04344202
14 1e-01      4 0.5611538 0.04344202
15 1e+00      4 0.5611538 0.04344202
16 5e+00      4 0.5611538 0.04344202
17 1e+01      4 0.5611538 0.04344202
18 1e+02      4 0.5611538 0.04344202
```

For a polynomial kernel, the lowest cross-validation error i.e. 0.301 is obtained for a degree of 2 and a cost of 100.
Below is the confusion matrix and accuracy by using best parameters and the accuracy = 71.68%;

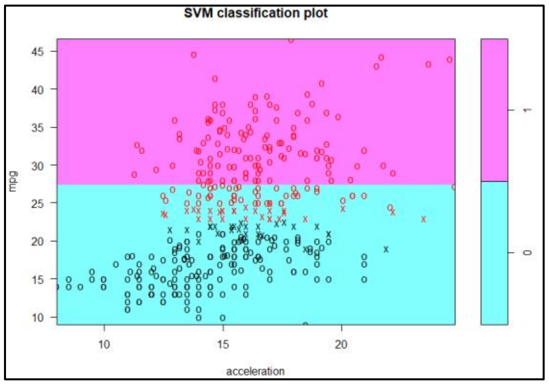


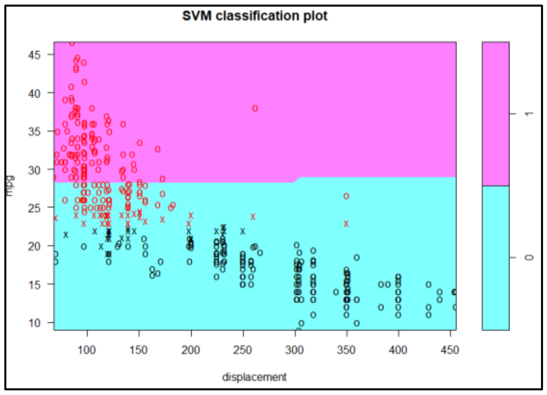
```
> conf_matrix_poly
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  87   2
         1 109 194

               Accuracy : 0.7168
                 95% CI : (0.6694, 0.7609)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4337
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.4439
            Specificity : 0.9898
         Pos Pred Value : 0.9775
         Neg Pred Value : 0.6403
             Prevalence : 0.5000
         Detection Rate : 0.2219
   Detection Prevalence : 0.2270
      Balanced Accuracy : 0.7168

       'Positive' Class : 0
```

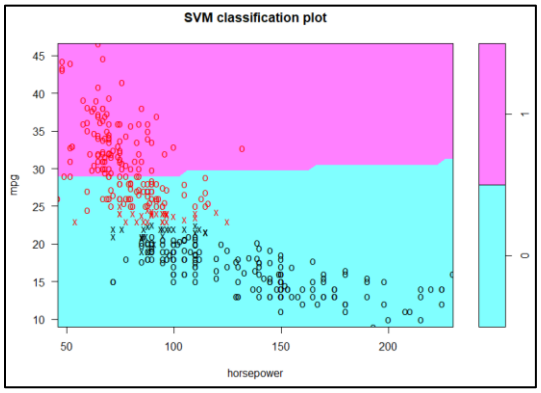
Observation: We get maximum accuracy with SVM with radial kernal

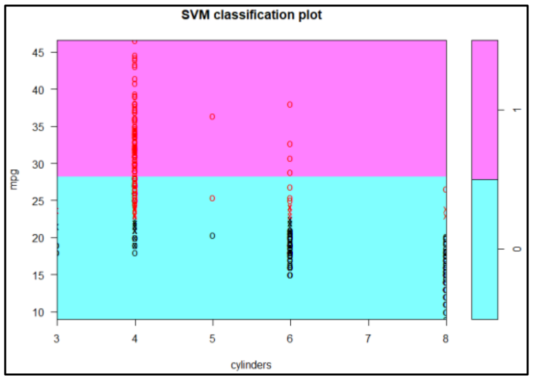
(d) Make some plots to back up your assertions in (b) and (c).
(Hint: In the lab, we used the plot() function for svm objects only in cases with p = 2. When p > 2, you can use the plot() function to create plots displaying pairs of variables at a time.
Essentially, instead of typing > plot(svmfit , dat)where svmfit contains your fitted model and dat is a data frame containing your data, you can type
> plot(svmfit , dat , x1~x4) in order to plot just the first and fourth variables. However, you must replace x1 and x4 with the correct variable names. To find out more, type ?plot.svm.)

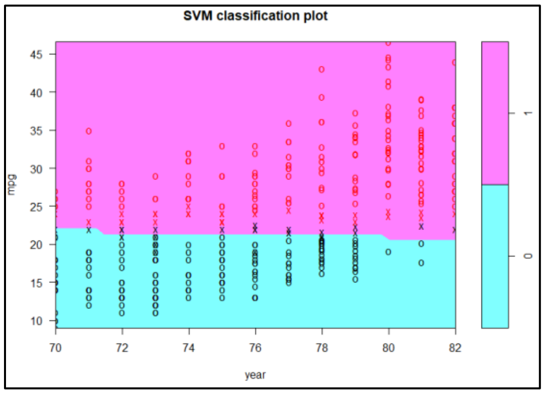
Plots of SVM with linear kernel:

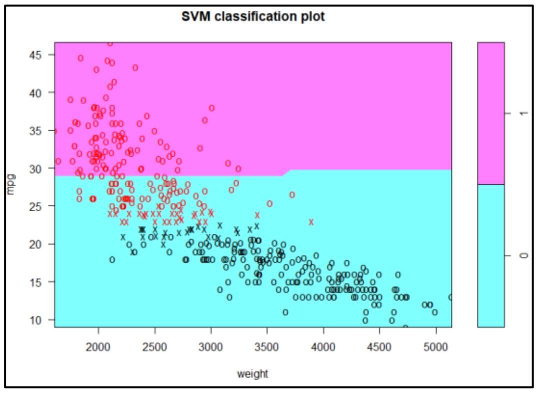




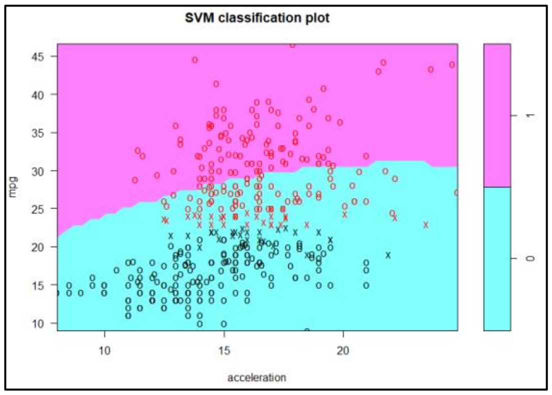


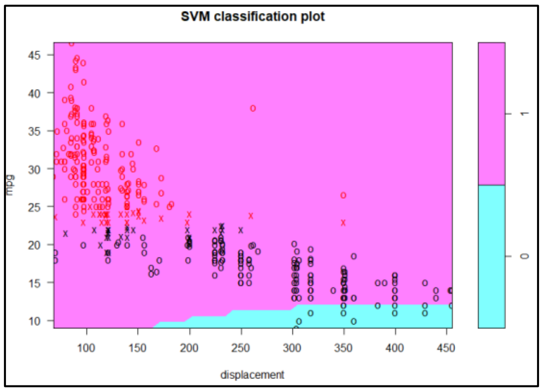


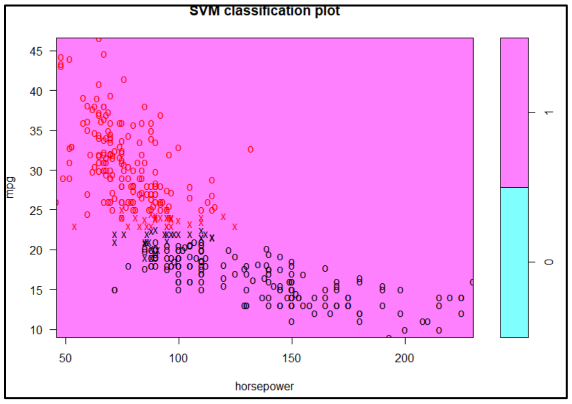


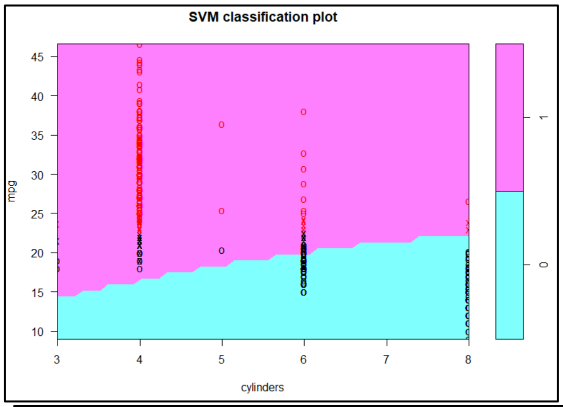


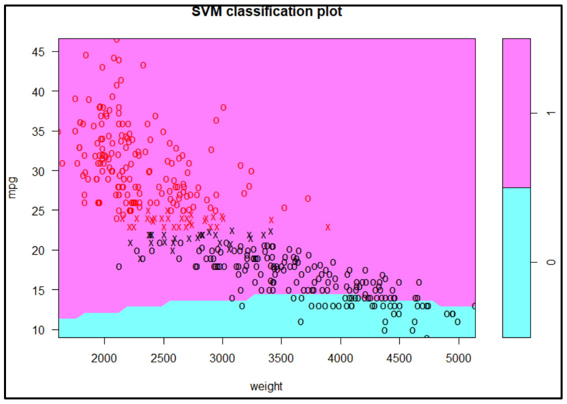
Plots of SVM with radial kernel:

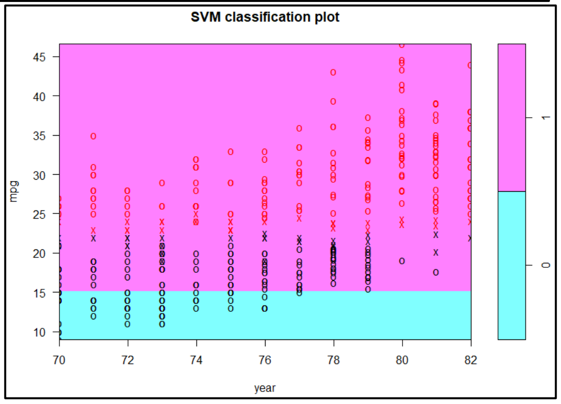




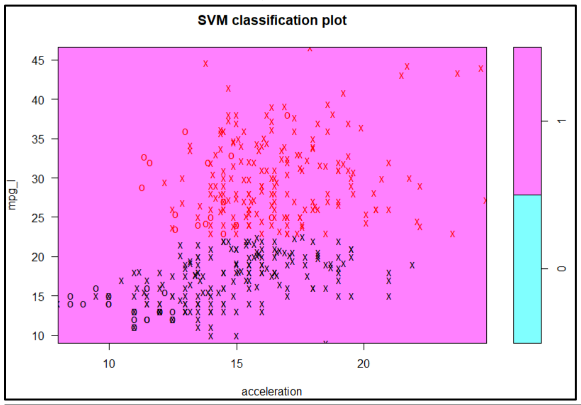


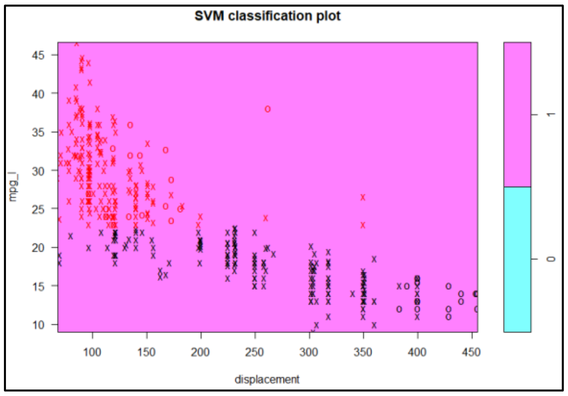


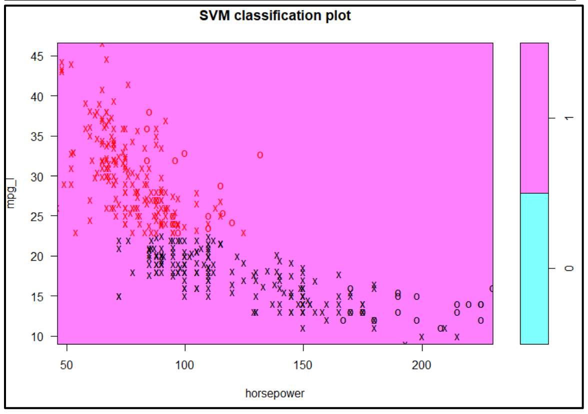


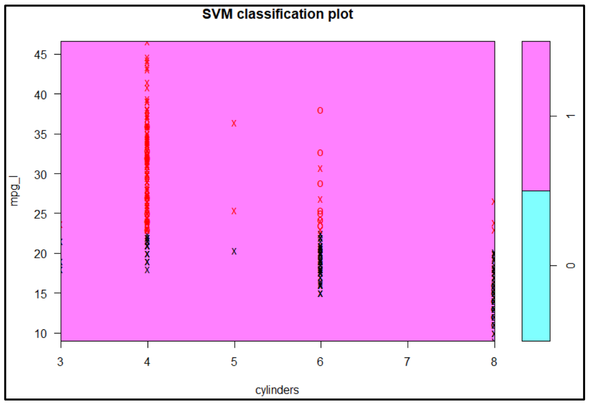


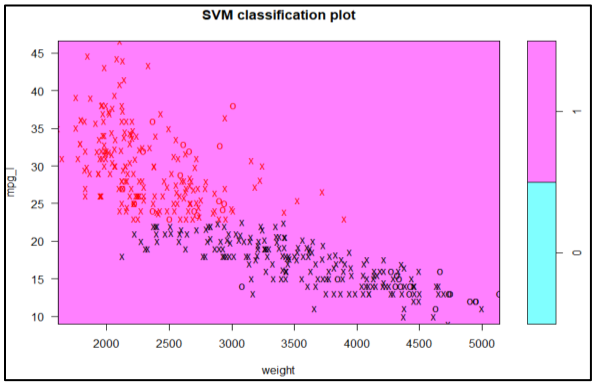
Plots of SVM with Polynomial kernel

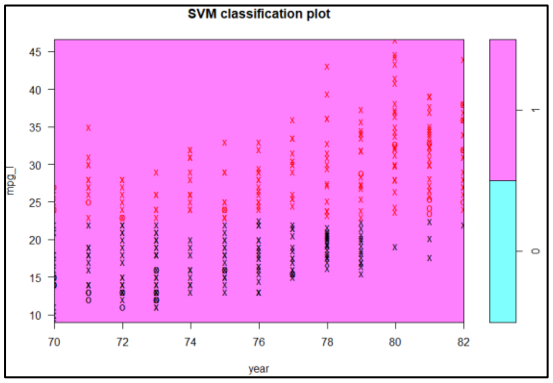










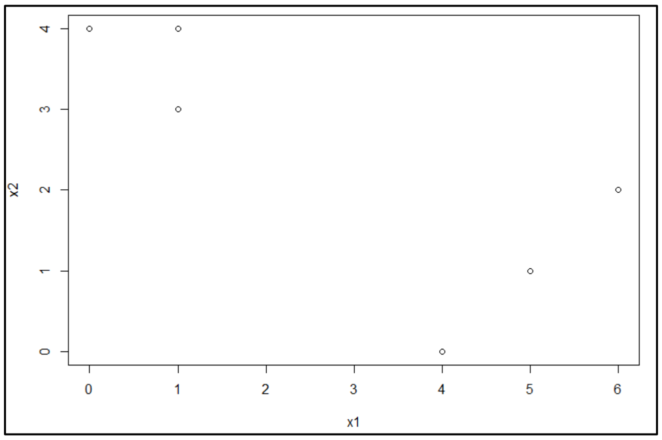


<u>Observations:</u> From the plots it can be infered that SVM with linear kernal gives is most correct classification. SVM with polynomial kernal classifies most of them as 1. This is consistant with our findings in question (b) and (c).

3) In this problem, you will perform *K*-means clustering manually, with *K* = 2, on a small example with *n* = 6 observations and *p* = 2 features. The observations are as follows.

Obs.	X1	X2
1	1	4
2	1	3
3	0	4
4	5	1
5	6	2
6	4	0

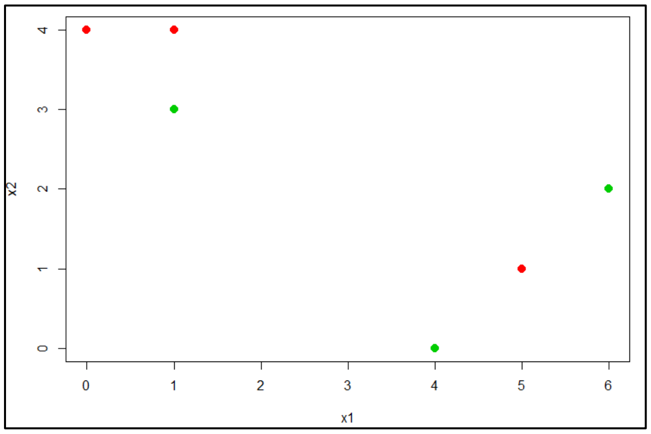
(a) Plot the observations.



(b) Randomly assign a cluster label to each observation. You can use the sample() command in R to do this. Report the cluster labels for each observation.

Ans: Randomly assigned the labels for each observation and plotted as below;

Obs.	X1	X2	Labels
1	1	4	Red
2	1	3	Green
3	0	4	Red
4	5	1	Red
5	6	2	Green
6	4	0	Green



(c) Compute the centroid for each cluster.

Ans: Centroids are calculated as bellows:

For red cluster:

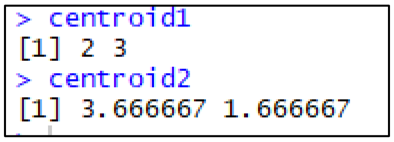
$$X_{11}=1/3*(0+1+5) = 2$$

And $$X_{12}=1/3*(4+4+1) = 3$$

For Green cluster:

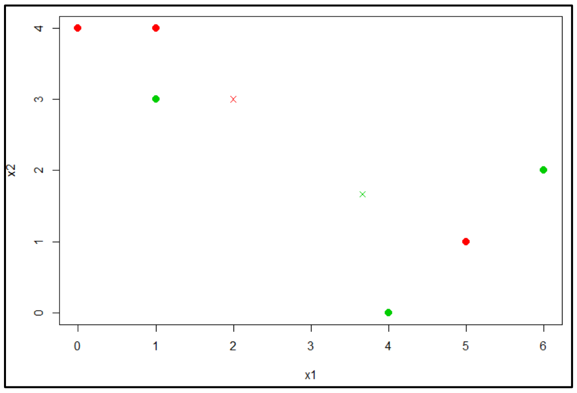
$$X_{21}=1/3*(1+3+6) = 10/3 = 3.666$$

And $$X_{22}=1/3*(0+2+3)=5/3 = 1.6667$$



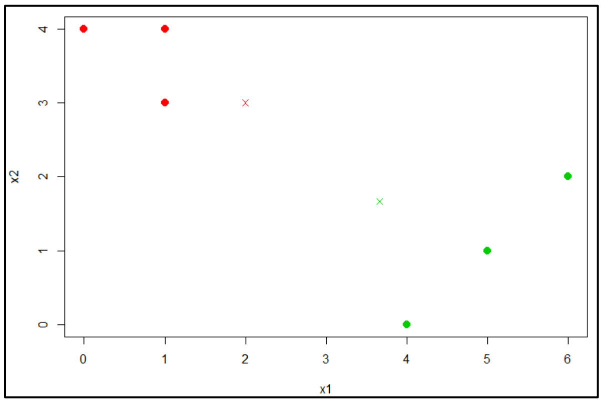
```
> centroid1
[1] 2 3
> centroid2
[1] 3.666667 1.666667
```

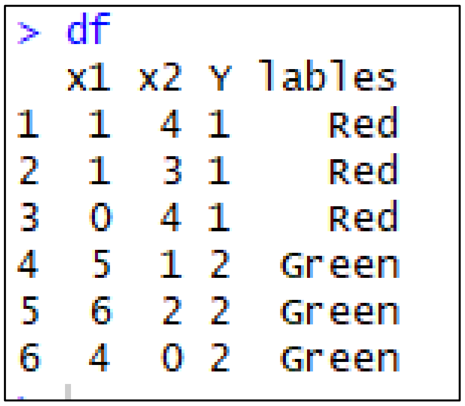
The above centroids are ploted below;



(d) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

Ans: After assigning each observation to its closest centroid the observations are plotted below with their new cluster labels:





```
> df
  x1 x2 Y lables
1  1  4 1    Red
2  1  3 1    Red
3  0  4 1    Red
4  5  1 2  Green
5  6  2 2  Green
6  4  0 2  Green
```

(e) Repeat (c) and (d) until the answers obtained stop changing.

Ans: Centroids are calculated as bellows:
For red cluster:

$$X_{11}=1/3*(0+1+1) = 2/3 = 0.666$$

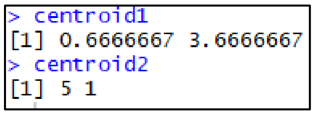
And $$X_{12}=1/3*(4+4+3) = 11/3=3.666$$

For Green cluster:

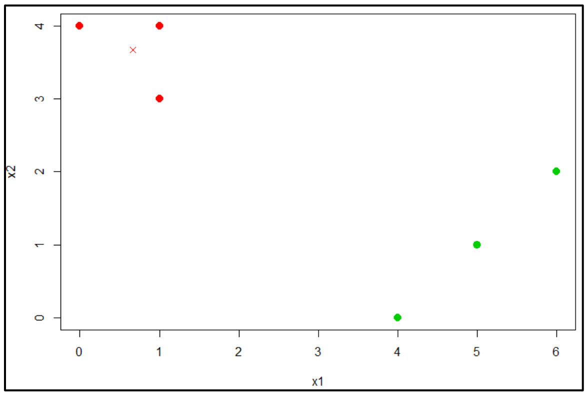
$$X_{21}=1/3*(4+5+6) = 15/3 = 5$$

And $$X_{22}=1/3*(0+1+2)=3/3 = 1$$

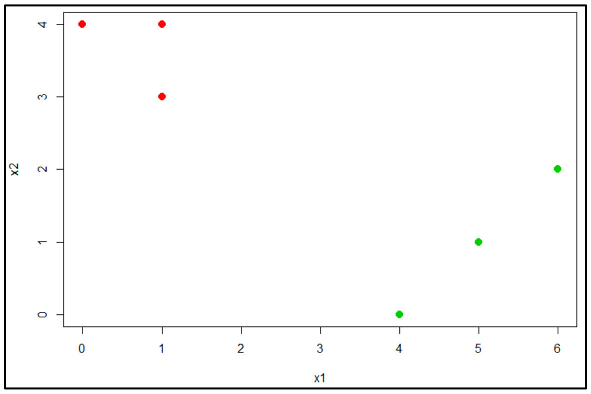
R output:



```
> centroid1
[1] 0.6666667 3.6666667
> centroid2
[1] 5 1
```



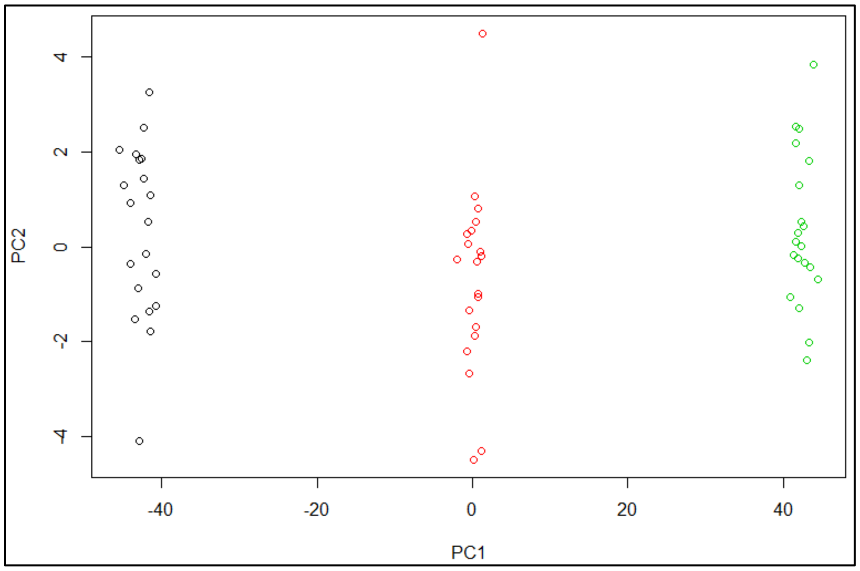
(f) In your plot from (a), color the observations according to the cluster labels obtained.



4) In this problem, you will generate simulated data, and then perform PCA and *K*-means clustering on the data.

(a) Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables.
Hint: There are a number of functions in R *that you can use to generate data. One example is the* rnorm() *function;* runif() *is another option. Be sure to add a mean shift to the observations in each class so that there are three distinct classes.*
(Please refer to the R code- Ch10 Problem 10)

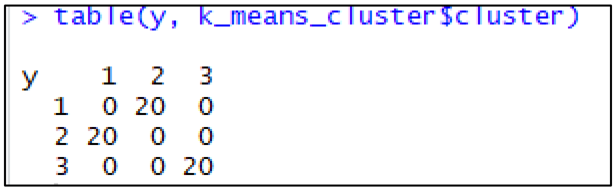
(b) Perform PCA on the 60 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes n appear separated in this plot, then continue on to part(c) If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes. Do not continue to part (c) until the three classes show at least some separation in the first two principal component score vectors.



(c) Perform *K*-means clustering of the observations with *K* = 3. How well do the clusters that you obtained in *K*-means clustering compare to the true class labels?

(Hint: You can use the table() function in R to compare the true class labels to the class labels obtained by clustering. Be careful how you interpret the results: K-means clustering will arbitrarily number the clusters, so you cannot simply check whether the true class labels and clustering labels are the same.)

Ans: Confusion matrix for k-means clustering with k=3

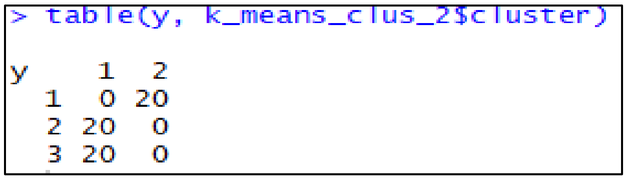


```
> table(y, k_means_cluster$cluster)

y     1  2  3
  1   0 20  0
  2  20  0  0
  3   0  0 20
```

Observation: All the classes are separated perfectly but actual class 1 is predicted as class 2 and visa versa

(d) Perform *K*-means clustering with *K* = 2. Describe your results.

Ans: Confusion matrix for k-means clustering with k=2

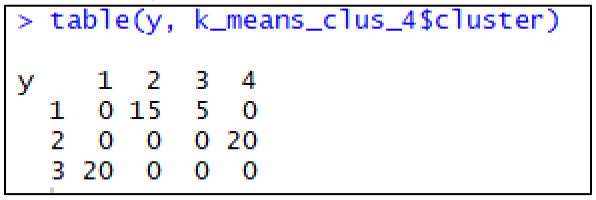


```
> table(y, k_means_clus_2$cluster)

y     1  2
  1   0 20
  2  20  0
  3  20  0
```

Observations: One of the clusters amongst the actual 3 is predicted as a separate cluster however the remaining 2 clusters are i.e. 2 and 3 are being predicted as one cluster

(e) Now perform *K*-means clustering with *K* = 4, and describe your results.

Ans: Confusion matrix for k-means clustering with k=2

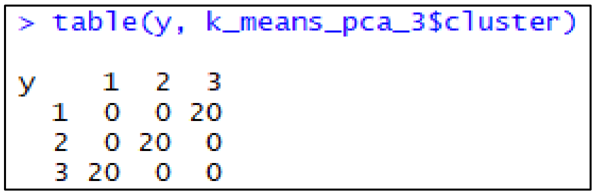


```
> table(y, k_means_clus_4$cluster)

y    1  2  3  4
  1  0 15  5  0
  2  0  0  0 20
  3 20  0  0  0
```

Observations: Two of the clusters amongst the actual 3 clusters are being predicted as 2 separate clusters defines as cluster 1 and 4 from the 4 predicted clusters however the remaining 1 cluster is i.e. actual 1st cluster is being predicted as two different clusters

(f) Now perform *K*-means clustering with K = 3 on the first two principal component score vectors, rather than on the raw data. That is, perform K-means clustering on the 60 × 2 matrix of which the first column is the first principal component score vector, and the second column is the second principal component score vector. Comment on the results.

Ans:

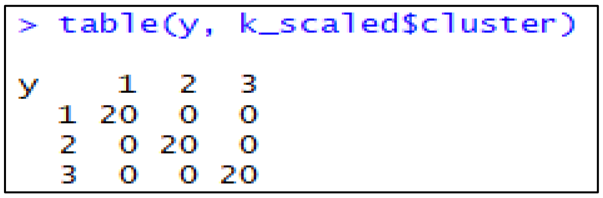


```
> table(y, k_means_pca_3$cluster)

y    1  2  3
  1  0  0 20
  2  0 20  0
  3 20  0  0
```

Observation: All the classes are separated perfectly but actual class 1 is predicted as class 3 and visa-versa. This means that the first 2 PCAs are sufficient in defining the difference between all the clusters.

(g) Using the scale() function, perform *K*-means clustering with *K* = 3 on the data *after scaling each variable to have standard deviation one*. How do these results compare to those obtained in (b)? Explain.

Ans:



```
> table(y, k_scaled$cluster)

y    1  2  3
  1 20  0  0
  2  0 20  0
  3  0  0 20
```

Observation: All the classes are separated perfectly. Scaling X did not affect anything in this case.