1.

A.

① (a)

Simple Linear equation:.

$y \in R^n$.

$X \rightarrow$ matrix $n \times p$.

$$y = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots \quad \beta_p x_{ip} + \epsilon_i$$

mth principal component:-

$$Z_{im} = \phi_{1m} x_{i1} + \phi_{2m} x_{i2} + \cdots \quad + \phi_{pm} x_{ip} \quad - (1)$$

Let the first M principal score vectors be $Z_1, Z_2 \ldots Z_M$.

Hence

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 Z_{i1} + \hat{\beta}_2 Z_{i2} + \hat{\beta}_3 Z_{i3} + \cdots \quad \hat{\beta}_M Z_{iM} + \epsilon_i$$

Where $Z_{i1}, Z_{i2} \ldots Z_{iM}$ are principal score vectors.

$\hat{\beta}_0, \hat{\beta}_1 \ldots \hat{\beta}_p$ are coefficients estimated using least square.

B. Answer below

C. Answer below

(b) plugging equation (1) in equation from (a)

$$y_i = \hat{\beta}_0 + \hat{\beta}_1(\phi_{11} x_{i1} + \phi_{21} x_{i2} + \cdots \phi_{p1} x_{ip}) + \cdots$$

$$\hat{\beta}_M(\phi_{1M} x_{i1} + \phi_{2M} x_{i2} + \cdots \phi_{PM} x_{iM}) + \epsilon_i$$

$$= \hat{\beta}_0 + (\hat{\beta}_1 \phi_{11} + \hat{\beta}_2 \phi_{12} + \cdots \hat{\beta}_M \phi_{1M}) x_{i1} + \cdots$$

$$+ (\hat{\beta}_1 \phi_{p1} + \hat{\beta}_2 \phi_{p2} + \cdots \hat{\beta}_M \phi_{pM}) x_{ip} + \epsilon_i \quad -(2)$$

(c) The above equation can be expressed as.

$$y_i = \bar{\beta}_0 + \bar{\beta}_1 x_{i1} + \bar{\beta}_2 x_{i2} + \cdots \bar{\beta}_p x_{ip} + \epsilon_i \quad -(3)$$

where,

$$\bar{\beta}_0 = \hat{\beta}_0$$

$$\bar{\beta}_1 = (\hat{\beta}_1 \phi_{11} + \hat{\beta}_2 \phi_{12} + \cdots \hat{\beta}_M \phi_{1M})$$

$$\bar{\beta}_2 = (\hat{\beta}_1 \phi_{21} + \hat{\beta}_2 \phi_{22} + \cdots \hat{\beta}_M \phi_{2M})$$

$$\vdots$$

$$\bar{\beta}_p = (\hat{\beta}_1 \phi_{p1} + \hat{\beta}_2 \phi_{p2} + \cdots \hat{\beta}_M \phi_{pM}).$$

Since the above equation is a linear combination of columns of X and the degree of X-variables is 1, we can sayh that the equation is linear in columns of X.

D. In the light of the above equation, we can say that the claim in this question is false.
Since, in linear least square model, the coefficients are estimated using least square method. However, in above equation(C), the coefficients have Phi terms which are derived using PCA and the estimator used in PCA do not rely on the dependent variable.

So, the coefficients in both linear least square model and PCA regressor would be different and hence they would give different results/predictions.
Also, since M is less than p, the above equation which is derived from M principal score vectors wouldn't account for all the variance in the original data. Hence the predictions would be different.

2.
A. Data generation:

Four clusters were created by shifting means of the random data generated.

```
library(cluster)
library(rdist)
set.seed(100)
x <- matrix(rnorm(100*2),ncol=2)
x[1:25, 1] <- x[1:25, 1] + 3
x[1:25, 2] <- x[1:25, 2] - 4

x[26:50, 1] <- x[26:50, 1] + 6
x[26:50, 1] <- x[26:50, 1] + 7

x[51:75, 1] <- x[51:75, 1] + 9
x[51:75, 1] <- x[51:75, 1] + 10

plot(x)
```

The function to calculate the LHS of the equation:

```
LHS.WSS <- function(x){
    n.ck <- nrow(x)
    sum <- 0
    for (i in 1:n.ck)

      for(j in 1:n.ck){
        #if(j<=n.ck)
        {
      for(k in 1:ncol(x)){

        sum<- sum + (x[i,k]-x[j,k])^2

      }
      }
      }

    sum <- sum / n.ck
    return(sum)
}
```

The function to calculate the RHS of the equation:

```
RHS.centroid.dist <- function(x){
  n.ck <- nrow(x)
  sum <- 0
  for (i in 1:n.ck){
      for(k in 1:ncol(x)){

        sum<- sum + (x[i,k]-mean(x[,k]))^2
    }
  }
  return(sum)
}
```

The value after computing LHS: **327.7264**

```
> LHS<- LHS.WSS(x[1:25,]) + LHS.WSS(x[26:50,]) + LHS.WSS(x[51:75,]) + LHS.WSS(x[76:100,])
> LHS
[1] 327.7264
```

The value after computing RHS:**327.7264**

```
> RHS<- 2*(RHS.centroid.dist(x[1:25,]) + RHS.centroid.dist(x[26:50, ]) + RHS.centroid.dist(x[51:75, ]) + RHS.centroid.dist(x
[76:100, ]))
> RHS
[1] 327.7264
```

Hence, LHS=RHS.

3.

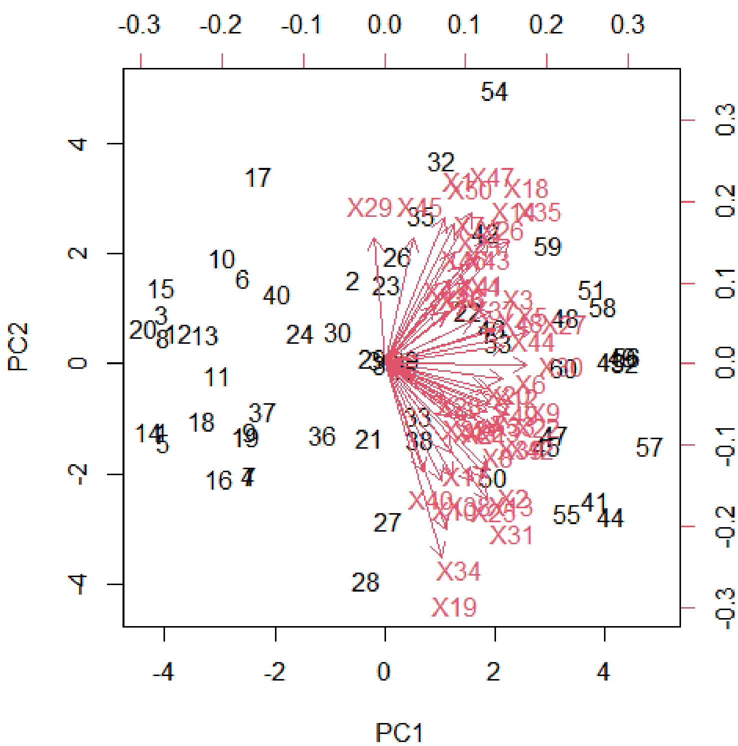
A. Data is generated using rnorm and 3 well separated classed were formed by shifting means.

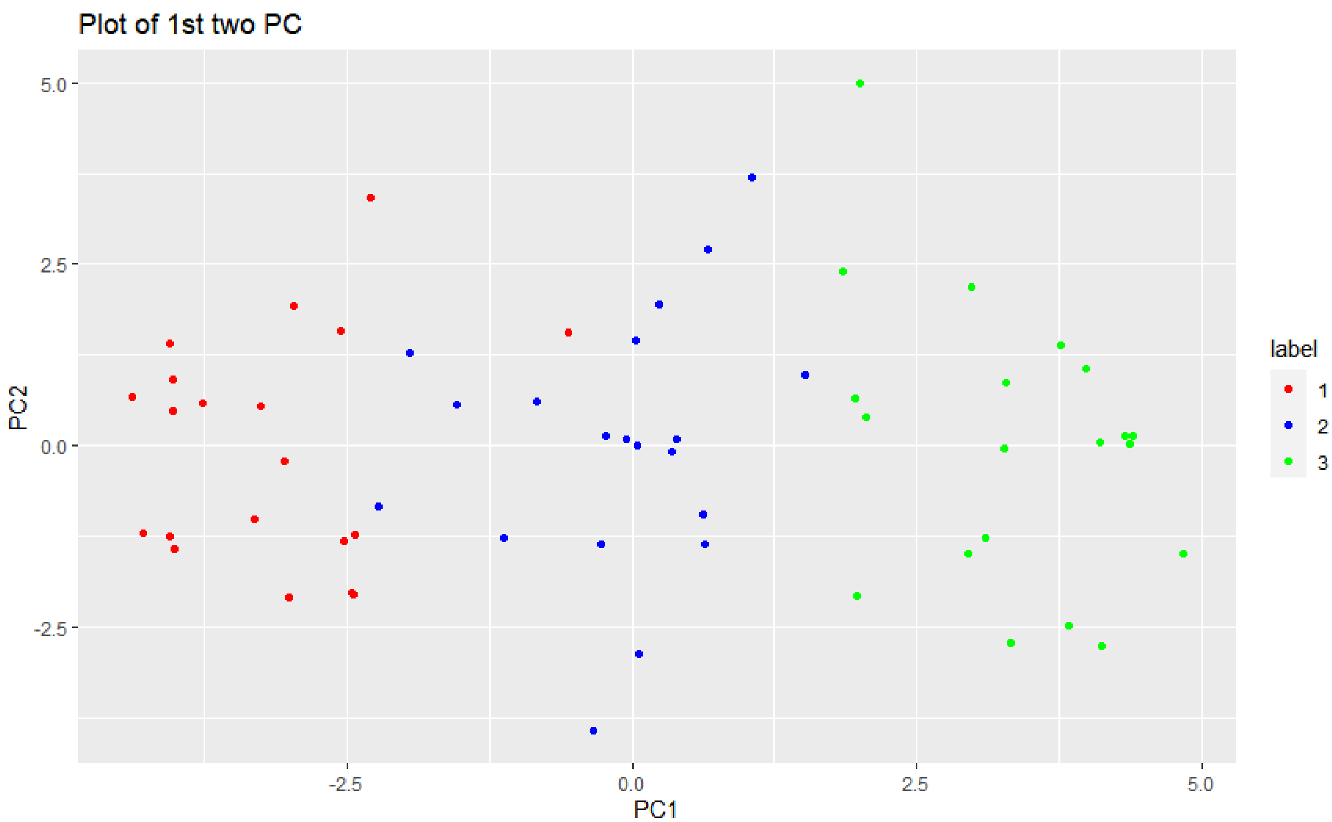
```
library(cluster)
library(rdist)
set.seed(100)
x <- matrix(rnorm(60*50),ncol=50)
x[1:20, ] <- x[1:20, ] + 2


x[21:40, ] <- x[21:40, ] + 2.5


x[41:60, ] <- x[41:60, ] + 3


plot(x)


df_k1 <- data.frame(x[1:20,])
df_k2 <- data.frame(x[21:40, ])
df_k3 <- data.frame(x[41:60, ])

df_k1$label <- as.factor('1')
df_k2$label <- as.factor('2')
df_k3$label <- as.factor('3')

data <- rbind(df_k1,df_k2,df_k3)
```

B. PCA was performed on the data generated above and first two principal components were plotted:

```
pr.out <- prcomp (data[,c(1:50)] , scale = TRUE)
dim (pr.out$x)
biplot (pr.out , scale = 0)
#pr.out$rotation = -pr.out$rotation
#pr.out$x = -pr.out$x
#biplot (pr.out , scale = 0)

library(ggplot2)

pc_data <- data.frame(pr.out$x)
pc_data$label <- 1
pc_data[1:20,]$label <- 1
pc_data[21:40,]$label <- 2
pc_data[41:60,]$label <- 3
pc_data$label <- as.factor(pc_data$label)
plot1= ggplot(data=pc_data,aes(PC1,PC2,color=label)) +
  geom_point()+scale_color_manual(values = c("1" = "red", "2" = "blue","3"="green"))
plot1 + ggtitle('Plot of 1st two PC')
```

Plots of first 2 PCs using biplot and ggplot:





C. K-means clustering was performed with K=3

```
set.seed(2)
km3.out <- kmeans (data[,c(1:50)], 3, nstart = 20)
km3.out$cluster
#ss <- function(x) sum(scale(x, scale = FALSE)^2)

data$clustersK3 <- km3.out$cluster

table(data$label,data$clustersK3)
```

```
> table(data[,c('label','clustersK3')])
     clustersK3
label  1  2  3
    1  5  0 15
    2 15  5  0
    3  0 20  0
```

```
> km3.out
K-means clustering with 3 clusters of sizes 20, 25, 15

Cluster means:
        X1       X2       X3       X4       X5       X6       X7       X8       X9      X10      X11      X12      X13
1 2.331565 2.530876 2.372954 2.168852 2.081110 2.000602 1.660542 2.267599 1.981415 2.801649 2.936514 2.615973 2.274527
2 2.935644 2.883431 2.936434 2.676046 3.287673 3.161084 2.882901 2.846916 3.236739 2.850374 2.579173 3.043244 3.232627
3 2.241383 1.582230 1.958486 2.286427 2.264596 1.801284 2.472017 1.937828 1.993071 1.944517 1.423343 1.840574 2.025201
       X14      X15      X16      X17      X18      X19      X20      X21      X22      X23      X24      X25      X26
1 2.135488 2.827273 1.962098 2.371138 2.387868 2.713675 2.685533 2.671322 2.368514 2.176716 2.210389 2.890503 2.655498
2 3.220972 3.089877 2.595543 2.891902 3.272183 2.310671 2.905403 2.704763 3.157697 2.786845 2.987832 2.778519 3.159138
3 2.255519 2.383899 2.156872 2.357217 2.016831 1.436105 1.728154 1.972123 1.884869 2.121141 1.996686 1.336482 1.936462
       X27      X28      X29      X30      X31      X32      X33      X34      X35      X36      X37      X38      X39
1 2.091000 2.591783 2.085884 1.996294 2.449726 2.614508 2.485467 2.325916 1.616258 2.683804 2.349101 2.455352 2.504440
2 3.082758 2.750614 2.448419 3.161368 2.948498 3.062550 2.806091 2.510543 3.197359 2.910636 3.095304 2.889456 3.015790
3 1.424176 2.109335 2.847039 1.781234 1.845420 2.223910 1.823142 2.186268 2.136790 1.933280 2.218194 2.051296 1.832209
       X40      X41      X42      X43      X44      X45      X46      X47      X48      X49      X50
1 2.689023 2.407693 2.323501 2.237646 2.251501 2.638271 2.417715 2.284633 2.296229 2.142292 2.283320
2 2.441210 3.090178 2.893204 2.674024 3.096274 2.544043 3.143495 2.922919 2.889477 3.078388 2.730875
3 1.982553 1.849219 1.878480 1.919045 1.968658 2.193510 2.133408 1.767956 1.884769 2.243903 2.085141

Clustering vector:
 [1] 3 3 3 1 3 3 1 3 1 3 3 3 3 3 3 1 3 3 1 3 1 2 1 1 1 2 1 1 1 1 1 2 2 1 2 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[59] 2 2

Within cluster sum of squares by cluster:
[1]  951.9317 1319.9501  692.1522
 (between_SS / total_SS =  16.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
> |
```

As we can see above, each true label had 20 observations, and an ideal clustering would have yielded exactly 20 observations in each of the 3 estimated clusters with all the observations in belonging to exactly one of the true labels.

But in the above table, cluster 1 has 20 observations but with 2 true labels. Cluster 2 has 25 observations with 2 true labels and cluster 3 has 15 observations.

Since, the classes were well separated, we can assume the maximum number of observations in the true classes in each of the clusters are the corrected classified points.

With the above logic, **the total misidentified points in the three clusters are 5+5=10.** We can make this interpretation since number of clusters here is equal to number of true labels.

Another interpretation is that the observations of true label 3 are all identified in one cluster(2) but true labels-1 & 2 are identified in 2 different clusters.

Based on the results, we can say that K-mean clustering did a decent job in creating clusters close to true labels.

D.

```r
set.seed(2)
km2.out <- kmeans (data[,c(1:50)], 2, nstart = 20)
data$clustersK2 <- km2.out$cluster

data$clustersK2 <- as.factor(data$clustersK2)

km2.out
table(data[,c('label','clustersK2')])
```

```
> km2.out
K-means clustering with 2 clusters of sizes 28, 32

Cluster means:
        X1       X2       X3       X4       X5       X6       X7       X8       X9      X10      X11      X12      X13
1 2.884143 2.834412 2.856677 2.689452 3.196847 3.081781 2.734508 2.943468 3.105110 2.833202 2.600995 2.941873 3.170584
2 2.277723 2.096038 2.195633 2.164686 2.133476 1.867767 2.056168 1.974225 1.984367 2.410326 2.241622 2.301148 2.122121
       X14      X15      X16      X17      X18      X19      X20      X21      X22      X23      X24      X25      X26
1 3.212906 3.015123 2.589738 2.953677 3.212184 2.366540 2.898116 2.779001 3.012325 2.730101 2.873123 2.816676 3.018200
2 2.097046 2.660232 1.999092 2.261737 2.183540 2.103711 2.222524 2.275479 2.195020 2.143117 2.137700 2.139166 2.394555
       X27      X28      X29      X30      X31      X32      X33      X34      X35      X36      X37      X38      X39
1 3.025807 2.788419 2.479954 3.052623 2.826741 2.960093 2.875645 2.451462 3.032433 2.858497 3.123872 2.788752 2.957337
2 1.735281 2.317666 2.381095 1.881410 2.226235 2.479061 2.084084 2.294843 1.856340 2.356352 2.192785 2.313370 2.192538
       X40      X41      X42      X43      X44      X45      X46      X47      X48      X49      X50
1 2.458114 3.132837 2.826426 2.650177 3.098155 2.654400 3.089404 3.028126 2.860039 3.048693 2.662298
2 2.366307 2.044598 2.119919 2.068257 2.038075 2.342061 2.263734 1.890546 2.073498 2.128146 2.208470

Clustering vector:
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1 1 2 2 2 2 2 1 1 2 1 2 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[59] 1 1

Within cluster sum of squares by cluster:
[1] 1479.892 1634.430
 (between_SS / total_SS =  12.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
> table(data[,c('label','clustersK2')])
     clustersK2
label  1  2
    1  0 20
    2  8 12
    3 20  0
```

With k=2, two clusters were created, one with size 28 and other with size 32.
The BSS/TSS ratio is 12.5%.
If we compare the clusters with true labels using 'table', we can see that all the observations in true labels -1&3 are identified in a single cluster but the observations in true label are identified in 2 different clusters and mostly in the second cluster.

E.

```
set.seed(2)
km4.out <- kmeans (data[,c(1:50)], 4, nstart = 20)
km4.out$cluster
data$clustersK4 <- km4.out$cluster

data$clustersK4 <- as.factor(data$clustersK4)

km4.out
table(data[,c('label','clustersK4')])
```

```
> km4.out
K-means clustering with 4 clusters of sizes 18, 9, 14, 19

Cluster means:
        X1       X2       X3       X4       X5       X6       X7       X8       X9      X10      X11      X12      X13
1 2.363442 2.444117 2.328108 2.281417 1.980797 1.915602 1.535406 2.227671 1.933215 2.802707 2.929896 2.624399 2.286062
2 2.990502 2.797618 2.553254 1.864586 3.258771 2.748473 3.150965 2.049399 2.714464 2.079771 2.494091 2.258554 2.492710
3 2.249229 1.557307 1.925235 2.264753 2.096830 1.859845 2.269722 1.875042 1.912583 1.992181 1.383053 1.898365 1.974266
4 2.773551 2.919040 3.074143 2.895857 3.339161 3.200184 2.873237 3.199950 3.391507 3.126466 2.632215 3.256099 3.445316
       X14      X15      X16      X17      X18      X19      X20      X21      X22      X23      X24      X25      X26
1 2.219851 2.853916 1.889233 2.413117 2.328915 2.773379 2.629894 2.570844 2.189000 2.238311 2.053626 2.950491 2.578404
2 2.998958 3.357969 2.452873 2.392231 3.165748 1.550558 2.744538 2.449573 2.971024 1.875673 3.499508 2.167623 3.354435
3 2.217097 2.268069 2.204057 2.269501 2.013563 1.374153 1.728827 2.007324 1.846955 2.223246 1.859117 1.344338 1.898769
4 3.109449 2.958194 2.607618 3.070493 3.221701 2.656204 2.948710 2.852815 3.294060 2.985604 2.861336 2.941162 3.050073
       X27      X28      X29      X30      X31      X32      X33      X34      X35      X36      X37      X38      X39
1 2.110820 2.746293 2.099361 1.991911 2.536096 2.577900 2.568768 2.359446 1.513776 2.561787 2.393776 2.502181 2.652323
2 2.975130 1.713762 2.321215 2.731765 2.056421 3.023415 2.302081 1.567459 3.151695 3.046977 2.940068 2.066993 1.911496
3 1.285859 2.134083 2.832507 1.700359 1.855086 2.166704 1.777087 2.218539 2.094054 1.890486 2.208163 2.105456 1.747933
4 3.025192 3.026670 2.489432 3.233331 3.171556 3.066620 2.914368 2.865221 3.125317 2.917863 3.009193 3.104964 3.344754
       X40      X41      X42      X43      X44      X45      X46      X47      X48      X49      X50
1 2.722009 2.547681 2.336731 2.246566 2.320440 2.655298 2.355916 2.240175 2.206865 2.191992 2.241040
2 1.874618 2.400931 2.151347 2.938111 2.907711 2.906644 2.497531 3.076334 2.812575 2.111896 2.767256
3 2.053825 2.026159 1.835763 1.821458 1.903696 2.202962 2.198865 1.724419 1.821157 2.112270 2.088638
4 2.627775 3.016512 3.150177 2.526716 3.019877 2.340659 3.330232 2.796471 2.942110 3.443652 2.670023

Clustering vector:
 [1] 3 2 3 1 3 3 1 3 1 3 3 3 3 3 3 1 3 3 1 3 1 2 1 1 1 2 1 1 2 2 1 2 4 1 2 1 1 1 1 1 4 2 4 4 4 4 4 4 4 4 4 4 4 2 4 4 4 4
[59] 4 4

Within cluster sum of squares by cluster:
[1] 860.0720 420.8250 632.5946 918.7308
 (between_SS / total_SS =  20.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
> table(data[,c('label','clustersK4')])
     clustersK4
label  1  2  3  4
    1  5  1 14  0
    2 13  6  0  1
    3  0  2  0 18
```

With k=4, four clusters were created of sizes-18,9,14,19.

The BSS/TSS ratio is 20.5%.

If we compare the clusters with true labels using ‘table’, we can see that all the observations in true labels -1 are identified in 3 different clusters with most observations in cluster-3.

All the observations in true labels -2 are identified in 3 different clusters with most observations in cluster-1.

All the observations in true labels -3 are identified in 2 different clusters with most observations in cluster-4.

F.

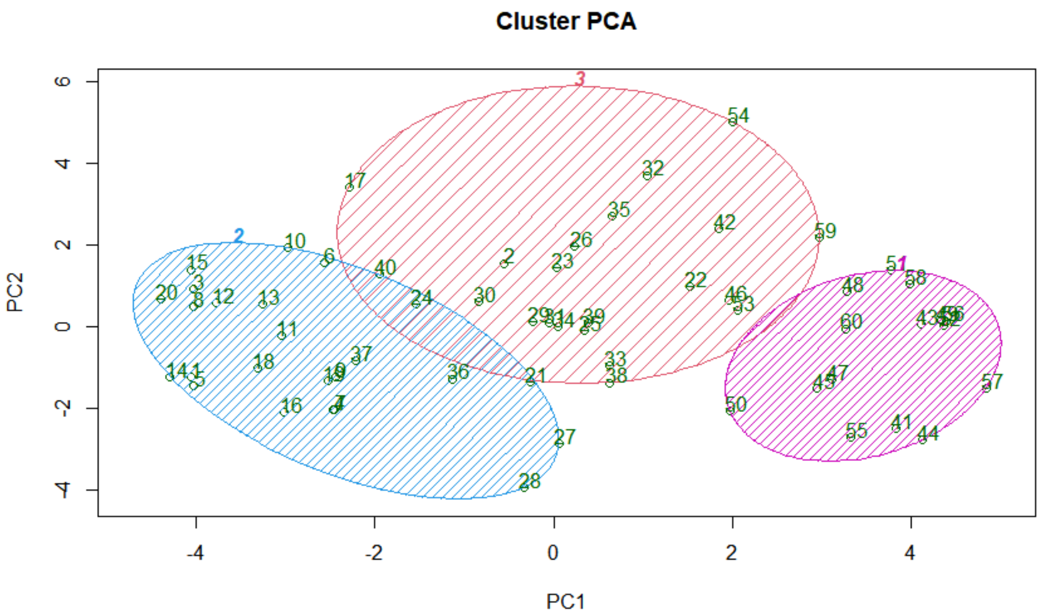
```
set.seed(2)
km3PCA.out <- kmeans (pc_data[,c(1:2)], 3, nstart = 20)
km3PCA.out$cluster
pc_data$clustersK3 <- km3PCA.out$cluster

pc_data$clustersK3 <- as.factor(pc_data$clustersK3)

km3PCA.out
table(pc_data$label,pc_data$clustersK3)

par(mfrow=c(1,1))

## Visualizing clusters
y_kmeans <- km3PCA.out$cluster
clusplot(pc_data[,c(1:2)],
         y_kmeans,
         lines = 0,
         shade = TRUE,
         color = TRUE,
         labels = 2,
         plotchar = FALSE,
         span = TRUE,
         main = paste("Cluster PCA"),
         xlab = 'PC1',
         ylab = 'PC2')


pc_data$clustersK3 <- ifelse(pc_data$clustersK3== 1,3,ifelse(pc_data$clustersK3== 2,1,2))
pc_data$clustersK3 <- as.factor(pc_data$clustersK3)

plot2= ggplot(data=pc_data,aes(PC1,PC2,color=label,shape=clustersK3)) +
  geom_point()+scale_color_manual(values = c("1" = "red", "2" = "blue","3"="green"))
plot2 + ggtitle('Cluster with PCA')
```

```
> km3PCA.out
K-means clustering with 3 clusters of sizes 15, 25, 20

Cluster means:
        PC1        PC2
1  3.7099772 -0.7141582
2 -2.7223692 -0.5661390
3  0.6204786  1.2432924

Clustering vector:
 [1] 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 3 3 2 3 3 2 2 3 3 3 3 3 3 3 2 2 3 3 2 1 3 1 1 1 3 1 1 1 1 1 1 3 3 1 1 1 1
[59] 3 1

Within cluster sum of squares by cluster:
[1] 35.26039 94.59465 78.21616
 (between_SS / total_SS =  68.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
> table(pc_data$label,pc_data$clustersK3)

     1  2  3
  1  0 18  2
  2  0  7 13
  3 15  0  5
```

With k=3, three clusters were created of sizes-15,25,20
The BSS/TSS ratio is 68.2%.

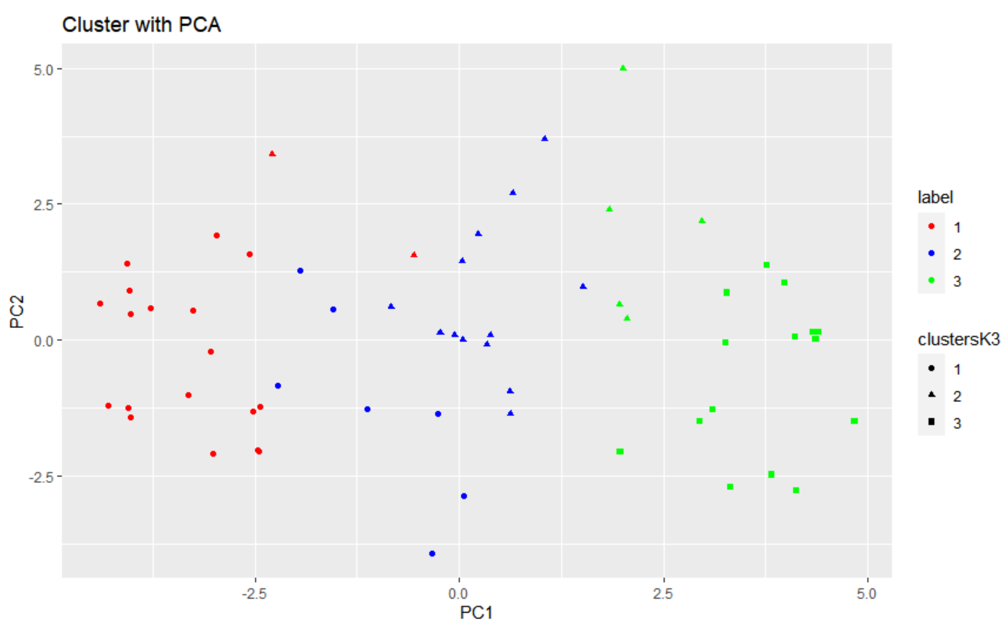
If we compare the clusters with true labels using 'table', we can see that all the observations in true labels -1 are identified in 2 different clusters with most observations in cluster-2. All the observations in true labels -2 are identified in 2 different clusters with most observations in cluster-3. all the observations in true labels -3 are identified in 2 different clusters with most observations in cluster-1.

The clusters are visualized below:



These two components explain 100 % of the point variability.

The true classes vs clusters plot:



G.

```
set.seed(2)
km3Scale.out <- kmeans (scale(data[,c(1:50)],center = TRUE,scale=TRUE), 3, nstart = 20)
km3Scale.out$cluster
data$clustersK3_scale <- km3Scale.out$cluster
data$clustersK3_scale <- as.factor(data$clustersK3_scale)
km3Scale.out
table(data$label,data$clustersK3_scale)
table(data$label,data$clustersK3)
```

```
> km3Scale.out
K-means clustering with 3 clusters of sizes 15, 23, 22

Cluster means:
           x1          x2         x3          x4         x5         x6         x7         x8          x9        x10
1  0.002760176  0.56674936  0.6512990  0.60635389  0.5985546  0.6194736  0.2361103  0.6444779  0.79601527  0.3355996
2 -0.400795678 -0.29041146 -0.5769587 -0.33621939 -0.5872598 -0.4244672 -0.3818092 -0.2908613 -0.46691718 -0.3517774
3  0.417131726 -0.08280803  0.1591166 -0.06192102  0.2058480  0.0213928  0.2381799 -0.1353345 -0.05459699  0.1389493
         x11         x12         x13        x14        x15          x16         x17        x18        x19        x20
1  0.1575825  0.60504570  0.80317613  0.5979171  0.1147389  0.467109673  0.41833965  0.3736523  0.4791019  0.5097969
2 -0.4825197 -0.37188242 -0.45334535 -0.5255383 -0.1752450 -0.303229583 -0.31326837 -0.6604825 -0.1561872 -0.6163679
3  0.3970098 -0.02374499 -0.07366814  0.1417557  0.1049796 -0.001471122  0.04227626  0.4357414 -0.1633738  0.2967958
         x21         x22        x23        x24         x25        x26         x27        x28        x29         x30
1  0.4738649  0.71693675  0.7240943  0.3581384  0.34671394  0.3641512  0.77518813  0.1302019 -0.1955930  0.91096529
2 -0.4201019 -0.48810545 -0.2458052 -0.5760297 -0.26569056 -0.4235345 -0.58972813 -0.2391397 -0.1431571 -0.68915588
3  0.1161078  0.02147155 -0.2367225  0.3580275  0.04137154  0.1945011  0.08799659  0.1612357  0.2830231  0.09936844
         x31         x32        x33        x34         x35        x36        x37        x38         x39          x40
1  0.7792327  0.54343956  0.5534013  0.7117921  0.75681103  0.1932422  0.3878593  0.6375109  0.61008173  0.286680008
2 -0.4086500 -0.38924543 -0.5512227 -0.1242425 -0.42562706 -0.5564908 -0.5010470 -0.0681524 -0.43252447 -0.004912799
3 -0.1040701  0.03641143  0.1989592 -0.3554229 -0.07103377  0.4500298  0.2593723 -0.3634163  0.03621986 -0.190327534
         x41        x42        x43          x44        x45        x46        x47         x48         x49        x50
1  0.4124800  0.9782307  0.4462153  0.680257392 -0.1785155  0.3496015  0.2267948  0.64452514  0.79889188  0.2339528
2 -0.4029284 -0.3982175 -0.4724584 -0.441685640 -0.2586383 -0.3445611 -0.5316501 -0.47471470 -0.43798012 -0.4716900
3  0.1400070 -0.2506572  0.1896960 -0.002049598  0.3921098  0.1218583  0.4011832  0.05684369 -0.08681071  0.3336171

Clustering vector:
 [1] 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 2 2 3 3 3 1 3 1 1 1 3 1 1 1 1 1 1 3 3 1 1 1 1
[59] 3 1

Within cluster sum of squares by cluster:
[1] 599.1670 934.3637 928.6906
 (between_SS / total_SS =  16.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
> table(data$label,data$clustersK3_scale)

     1  2  3
  1  0 19  1
  2  0  4 16
  3 15  0  5
> table(data$label,data$clustersK3)

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

The BSS/TSS ratio after scaling is 16.5% as compared to 16.7% before scaling. Also, if we look at the confusion matrix between clusters and true labels in both pre-scaling and post-scaling versions, we can clearly see that the distribution of observations in the true labels across different clusters are different.
This makes sense as K-means clustering is not scale-invariant.

4.

A. Training and test datasets were created

```
library(ISLR2)
df_OJ <- data.frame(OJ)
train <- sample(nrow(df_OJ),800)
df_OJ_training <- df_OJ[train,]
df_OJ_test <- df_OJ[-train,]
```

B. SVM model with cost =0.01 and linear kernel was built and summary statistics were observed.

```
set.seed(1)
modelsvm = svm(Purchase~.,df_OJ_training,cost=0.01,kernel='linear')
summary(modelsvm)
plot(modelsvm , df_OJ_training,PriceCH ~PriceMM)
```

```
> summary(modelsvm)

Call:
svm(formula = Purchase ~ ., data = df_OJ_training, cost = 0.01, kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01

Number of Support Vectors:  423

 ( 211 212 )


Number of Classes:  2

Levels:
 CH MM
```

Total number of support vectors used here were 423- 211 support vectors for one class and 212 for the other. The kernel used is linear and cost is 0.01.

C.

```
predYsvm.train = predict(modelsvm, df_OJ_training )
cnf.train <- table (predict = predYsvm.train , truth = df_OJ_training$Purchase)
train.error <- (cnf.train[1,2]+cnf.train[2,1])/sum(cnf.train)*100
train.error
predYsvm.test = predict(modelsvm, df_OJ_test )
cnf.test <- table (predict = predYsvm.test , truth = df_OJ_test$Purchase)
test.error <- (cnf.test[1,2]+cnf.test[2,1])/sum(cnf.test)*100
test.error
```

The training error rate is 15.375%.
The test error rate is 21.111%.

D.

```
set.seed(1)
OptModelsvm=tune(svm, Purchase~., data=df_OJ_training,kernel='linear', ranges=list(cost=seq(0.01,10,0.1)))
print(OptModelsvm)
#Find out the best model
BstModel=OptModelsvm$best.model
BstModel
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
 2.61

- best performance: 0.15625

> #Find out the best model
> BstModel=OptModelsvm$best.model
> BstModel

Call:
best.tune(method = svm, train.x = Purchase ~ ., data = df_OJ_training, ranges = list(cost = seq(0.01,
    10, 0.1)), kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  2.61

Number of Support Vectors:  311
```

The best parameter obtained from tune() are:
Cost- 2.61
Cv error- 15%

E.

```
set.seed(1)
modelsvm = svm(Purchase~.,df_OJ_training,cost=BstModel$cost,kernel='linear')
summary(modelsvm)
predYsvm.train = predict(modelsvm, df_OJ_training )
cnf.train <- table (predict = predYsvm.train , truth = df_OJ_training$Purchase)
train.error <- (cnf.train[1,2]+cnf.train[2,1])/sum(cnf.train)*100
train.error
predYsvm.test = predict(modelsvm, df_OJ_test )
cnf.test <- table (predict = predYsvm.test , truth = df_OJ_test$Purchase)
test.error <- (cnf.test[1,2]+cnf.test[2,1])/sum(cnf.test)*100
test.error
```

```
> train.error
[1] 14.75
> predYsvm.test = predict(modelsvm, df_OJ_test )
> cnf.test <- table (predict = predYsvm.test , truth = df_OJ_test$Purchase)
> test.error <- (cnf.test[1,2]+cnf.test[2,1])/sum(cnf.test)*100
> test.error
[1] 20.37037
```

The training error rate is 14.75%.
The test error rate is 20.3703%.

Using the best parameters from tune gave us lower training error and test error.

F.

```
#----------------------------------------------------
set.seed(1)
modelsvm = svm(Purchase~.,df_OJ_training,cost=0.01,kernel='radial')
summary(modelsvm)
plot(modelsvm , df_OJ_training,PriceCH ~PriceMM)


#Predict using SVM regression
predYsvm.train = predict(modelsvm, df_OJ_training )
cnf.train <- table (predict = predYsvm.train , truth = df_OJ_training$Purchase)
train.error <- (cnf.train[1,2]+cnf.train[2,1])/sum(cnf.train)*100
train.error
predYsvm.test = predict(modelsvm, df_OJ_test )
cnf.test <- table (predict = predYsvm.test , truth = df_OJ_test$Purchase)
test.error <- (cnf.test[1,2]+cnf.test[2,1])/sum(cnf.test)*100
test.error


#Tune the SVM model
#,ranges=list(elsilon=seq(0,1,0.1)
set.seed(1)
OptModelsvm=tune(svm, Purchase~., data=df_OJ_training,kernel='radial', ranges=list(cost=seq(0.01,10,0.1)))
print(OptModelsvm)
#Find out the best model
BstModel=OptModelsvm$best.model
BstModel

set.seed(1)
modelsvm = svm(Purchase~.,df_OJ_training,cost=BstModel$cost,kernel='radial')
summary(modelsvm)
predYsvm.train = predict(modelsvm, df_OJ_training )
cnf.train <- table (predict = predYsvm.train , truth = df_OJ_training$Purchase)
train.error <- (cnf.train[1,2]+cnf.train[2,1])/sum(cnf.train)*100
train.error
predYsvm.test = predict(modelsvm, df_OJ_test )
cnf.test <- table (predict = predYsvm.test , truth = df_OJ_test$Purchase)
test.error <- (cnf.test[1,2]+cnf.test[2,1])/sum(cnf.test)*100
test.error
```

```
> summary(modelsvm)

Call:
svm(formula = Purchase ~ ., data = df_OJ_training, cost = 0.01, kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  0.01

Number of Support Vectors:  618

 ( 311 307 )


Number of Classes:  2

Levels:
 CH MM



> plot(modelsvm , df_OJ_training,PriceCH ~PriceMM)
>
>
> #Predict using SVM regression
> predYsvm.train = predict(modelsvm, df_OJ_training )
> cnf.train <- table (predict = predYsvm.train , truth = df_OJ_training$Purchase)
> train.error <- (cnf.train[1,2]+cnf.train[2,1])/sum(cnf.train)*100
> train.error
[1] 38.375
> predYsvm.test = predict(modelsvm, df_OJ_test )
> cnf.test <- table (predict = predYsvm.test , truth = df_OJ_test$Purchase)
> test.error <- (cnf.test[1,2]+cnf.test[2,1])/sum(cnf.test)*100
> test.error
[1] 40.74074
```

A support vector classifier was trained using a ‘radial’ kernel and cost=0.01.
Total number of support vectors used here were 618: 311 support vectors for one class and 307 for the other.

The training error was 38.375%
The test error was 40.74%.

```
> #Tune the SVM model
> #,ranges=list(elsilon=seq(0,1,0.1)
> set.seed(1)
> OptModelsvm=tune(svm, Purchase~., data=df_OJ_training,kernel='radial', ranges=list(cost=seq(0.01,10,0.1)))
> print(OptModelsvm)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
 0.61

- best performance: 0.15125

> #Find out the best model
> BstModel=OptModelsvm$best.model
> BstModel

Call:
best.tune(method = svm, train.x = Purchase ~ ., data = df_OJ_training, ranges = list(cost = seq(0.01,
    10, 0.1)), kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  0.61

Number of Support Vectors:  378
```

The svm classifier was retrained for a range of costs. The parameters of the best performing model was:
cost=0.61
Cv error=15.1%

```
> set.seed(1)
> modelsvm = svm(Purchase~.,df_OJ_training,cost=BstModel$cost,kernel='radial')
> summary(modelsvm)

Call:
svm(formula = Purchase ~ ., data = df_OJ_training, cost = BstModel$cost, kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  0.61

Number of Support Vectors:  378

 ( 190 188 )


Number of Classes:  2

Levels:
 CH MM


> predYsvm.train = predict(modelsvm, df_OJ_training )
> cnf.train <- table (predict = predYsvm.train , truth = df_OJ_training$Purchase)
> train.error <- (cnf.train[1,2]+cnf.train[2,1])/sum(cnf.train)*100
> train.error
[1] 13.375
> predYsvm.test = predict(modelsvm, df_OJ_test )
> cnf.test <- table (predict = predYsvm.test , truth = df_OJ_test$Purchase)
> test.error <- (cnf.test[1,2]+cnf.test[2,1])/sum(cnf.test)*100
> test.error
[1] 20.37037
```

The training error rate is 13.375%.
The test error rate is 20.3703%.

Using the best parameters from tune() gave us a lower training and test error. However the test error is same as obtained using 'linear' kernel.

G.

```
#g
set.seed(1)
modelsvm = svm(Purchase~.,df_OJ_training,cost=0.01,kernel='polynomial',degree=2)
summary(modelsvm)
plot(modelsvm , df_OJ_training,PriceCH ~PriceMM)


#Predict using SVM regression
predYsvm.train = predict(modelsvm, df_OJ_training )
cnf.train <- table (predict = predYsvm.train , truth = df_OJ_training$Purchase)
train.error <- (cnf.train[1,2]+cnf.train[2,1])/sum(cnf.train)*100
train.error
predYsvm.test = predict(modelsvm, df_OJ_test )
cnf.test <- table (predict = predYsvm.test , truth = df_OJ_test$Purchase)
test.error <- (cnf.test[1,2]+cnf.test[2,1])/sum(cnf.test)*100
test.error


#Tune the SVM model
#,ranges=list(elsilon=seq(0,1,0.1)
set.seed(1)
OptModelsvm=tune(svm, Purchase~., data=df_OJ_training,kernel='polynomial',degree=2, ranges=list(cost=seq(0.01,10,0.1)))
print(OptModelsvm)
#Find out the best model
BstModel=OptModelsvm$best.model
BstModel

set.seed(1)
modelsvm = svm(Purchase~.,df_OJ_training,cost=BstModel$cost,kernel='polynomial',degree=2)
summary(modelsvm)
predYsvm.train = predict(modelsvm, df_OJ_training )
cnf.train <- table (predict = predYsvm.train , truth = df_OJ_training$Purchase)
train.error <- (cnf.train[1,2]+cnf.train[2,1])/sum(cnf.train)*100
train.error
predYsvm.test = predict(modelsvm, df_OJ_test )
cnf.test <- table (predict = predYsvm.test , truth = df_OJ_test$Purchase)
test.error <- (cnf.test[1,2]+cnf.test[2,1])/sum(cnf.test)*100
test.error
```

```
> summary(modelsvm)

Call:
svm(formula = Purchase ~ ., data = df_OJ_training, cost = 0.01, kernel = "polynomial", degree = 2)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  0.01
     degree:  2
     coef.0:  0

Number of Support Vectors:  620

 ( 313 307 )


Number of Classes:  2

Levels:
 CH MM



> plot(modelsvm , df_OJ_training,PriceCH ~PriceMM)
>
>
> #Predict using SVM regression
> predYsvm.train = predict(modelsvm, df_OJ_training )
> cnf.train <- table (predict = predYsvm.train , truth = df_OJ_training$Purchase)
> train.error <- (cnf.train[1,2]+cnf.train[2,1])/sum(cnf.train)*100
> train.error
[1] 37.25
> predYsvm.test = predict(modelsvm, df_OJ_test )
> cnf.test <- table (predict = predYsvm.test , truth = df_OJ_test$Purchase)
> test.error <- (cnf.test[1,2]+cnf.test[2,1])/sum(cnf.test)*100
> test.error
[1] 40.37037
```

A support vector classifier was trained using a ‘polynomial’ kernel and cost=0.01.
Total number of support vectors used here were 620: 313 support vectors for one class and 307 for the other.

The training error was 37.25%
The test error was 40.37%.

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
 6.51

- best performance: 0.1625

> #Find out the best model
> BstModel=OptModelsvm$best.model
> BstModel

Call:
best.tune(method = svm, train.x = Purchase ~ ., data = df_OJ_training, ranges = list(cost = seq(0.01, 
    10, 0.1)), kernel = "polynomial", degree = 2)


Parameters:
   SVM-Type:  C-classification 
 SVM-Kernel:  polynomial 
       cost:  6.51 
     degree:  2 
     coef.0:  0 

Number of Support Vectors:  338

> 
> set.seed(1)
> modelsvm = svm(Purchase~.,df_OJ_training,cost=BstModel$cost,kernel='polynomial',degree=2)
> summary(modelsvm)

Call:
svm(formula = Purchase ~ ., data = df_OJ_training, cost = BstModel$cost, kernel = "polynomial", degree = 2)


Parameters:
   SVM-Type:  C-classification 
 SVM-Kernel:  polynomial 
       cost:  6.51 
     degree:  2 
     coef.0:  0 

Number of Support Vectors:  338

 ( 172 166 )


Number of Classes:  2 

Levels: 
 CH MM
```

The svm classifier was retrained for a range of costs and degree=2 with a polynomial kernel. The parameters of the best performing model was:
cost=6.51
Cv error=16.25%

```
> predYsvm.train = predict(modelsvm, df_OJ_training )
> cnf.train <- table (predict = predYsvm.train , truth = df_OJ_training$Purchase)
> train.error <- (cnf.train[1,2]+cnf.train[2,1])/sum(cnf.train)*100
> train.error
[1] 13.5
> predYsvm.test = predict(modelsvm, df_OJ_test )
> cnf.test <- table (predict = predYsvm.test , truth = df_OJ_test$Purchase)
> test.error <- (cnf.test[1,2]+cnf.test[2,1])/sum(cnf.test)*100
> test.error
[1] 22.22222
> 
```

The training error rate is 13.5%.
The test error rate is 22.22%.

Using the best parameters from tune() gave us a lower training and test error. However, the test error is slightly higher than what obtained using ‘linear’ and ‘radial’ kernel.

H. The svm model using ‘linear’ and ‘radial’ kernel with the best parameters yielded same and lower test error than svm model using ‘polynomial’ kernel.

But the training error rate and number of support vectors using ‘linear’ kernel were 14.75% and 311. Whereas, using ‘radial’ kernel,the training error rate and number of support vectors were 13.375% and 378. Since, **linear kernel** yielded the same test error using less support vectors and higher training error, we can say that it overfits less and is less complex, and hence, it is slightly **better than the model using ‘radial’ kernel.**