Programming for Data Science
Digital Assignment – 3

Feature Engineering & Model Fitting

Lab Slot: L33 & L34
Name: Tejas Rahul Rokade
Reg. No.: 20BDS0033

Questions:

1. Perform Label encoding on IRIS Dataset
2. Perform One-hot encoding on IRIS Dataset
3. Feature scaling or standardization
   a. Normalization
   b. Z-scale
4. Find the principal components of IRIS dataset
5. House rent prediction using linear regression
6. Medical diagnosis for disease spread pattern Using SVM

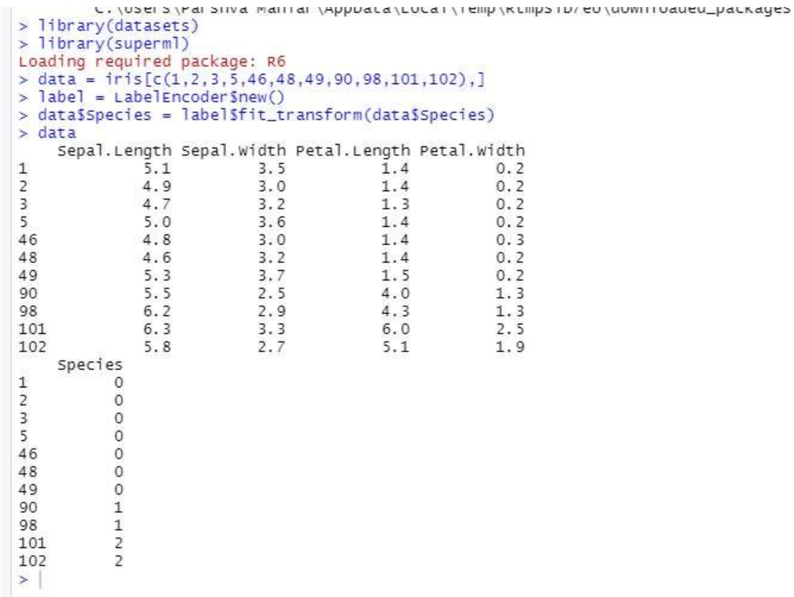
Q1 Perform Label Encoding on IRIS dataset.

Ans 1

Code:

```
library(datasets)

library(superml)

data  =  iris[c(1,2,3,5,46,48,49,90,98,101,102),]

label = LabelEncoder$new()

data$Species = label$fit_transform(data$Species)

data
```

Output:



```
        C:\Users\Parshva Manhar\AppData\Local\Temp\RtmpsTb7e0\downloaded_packages
> library(datasets)
> library(superml)
Loading required package: R6
> data = iris[c(1,2,3,5,46,48,49,90,98,101,102),]
> label = LabelEncoder$new()
> data$Species = label$fit_transform(data$Species)
> data
    Sepal.Length Sepal.Width Petal.Length Petal.Width
1            5.1         3.5          1.4         0.2
2            4.9         3.0          1.4         0.2
3            4.7         3.2          1.3         0.2
5            5.0         3.6          1.4         0.2
46           4.8         3.0          1.4         0.3
48           4.6         3.2          1.4         0.2
49           5.3         3.7          1.5         0.2
90           5.5         2.5          4.0         1.3
98           6.2         2.9          4.3         1.3
101          6.3         3.3          6.0         2.5
102          5.8         2.7          5.1         1.9
    Species
1         0
2         0
3         0
5         0
46        0
48        0
49        0
90        1
98        1
101       2
102       2
> 
```

Q2. Perform One-hot encoding on IRIS Dataset

Ans 2.

Code:

library(mltools)

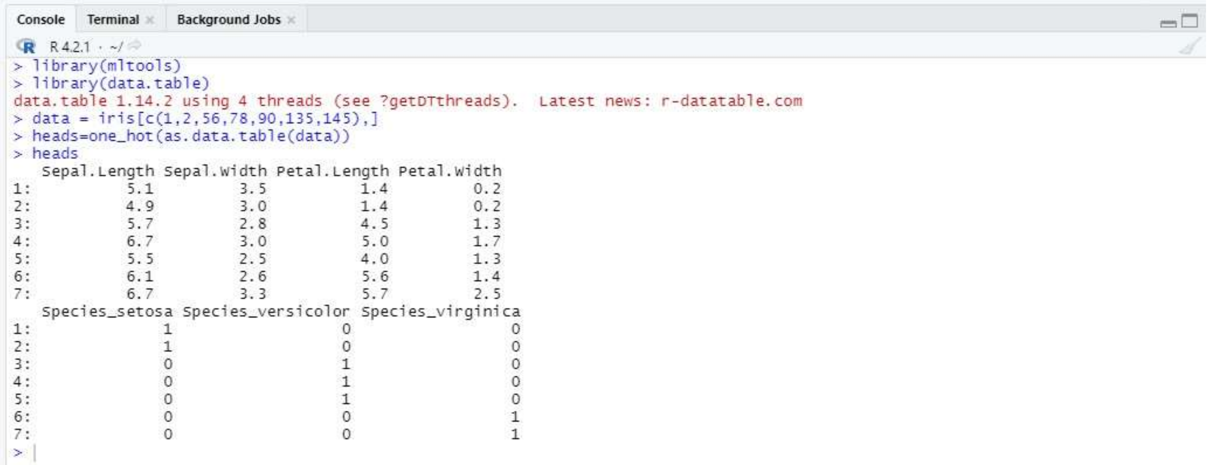
library(data.table)

data = iris[c(1,2,56,78,90,135,145),]

heads=one_hot(as.data.table(data))

heads

Output:



```
> library(mltools)
> library(data.table)
data.table 1.14.2 using 4 threads (see ?getDTthreads).  Latest news: r-datatable.com
> data = iris[c(1,2,56,78,90,135,145),]
> heads=one_hot(as.data.table(data))
> heads
   Sepal.Length Sepal.Width Petal.Length Petal.Width
1:          5.1         3.5          1.4         0.2
2:          4.9         3.0          1.4         0.2
3:          5.7         2.8          4.5         1.3
4:          6.7         3.0          5.0         1.7
5:          5.5         2.5          4.0         1.3
6:          6.1         2.6          5.6         1.4
7:          6.7         3.3          5.7         2.5
   Species_setosa Species_versicolor Species_virginica
1:              1                  0                 0
2:              1                  0                 0
3:              0                  1                 0
4:              0                  1                 0
5:              0                  1                 0
6:              0                  0                 1
7:              0                  0                 1
> 
```

Q3. Feature scaling or standardization

a) **Normalization**

Code:

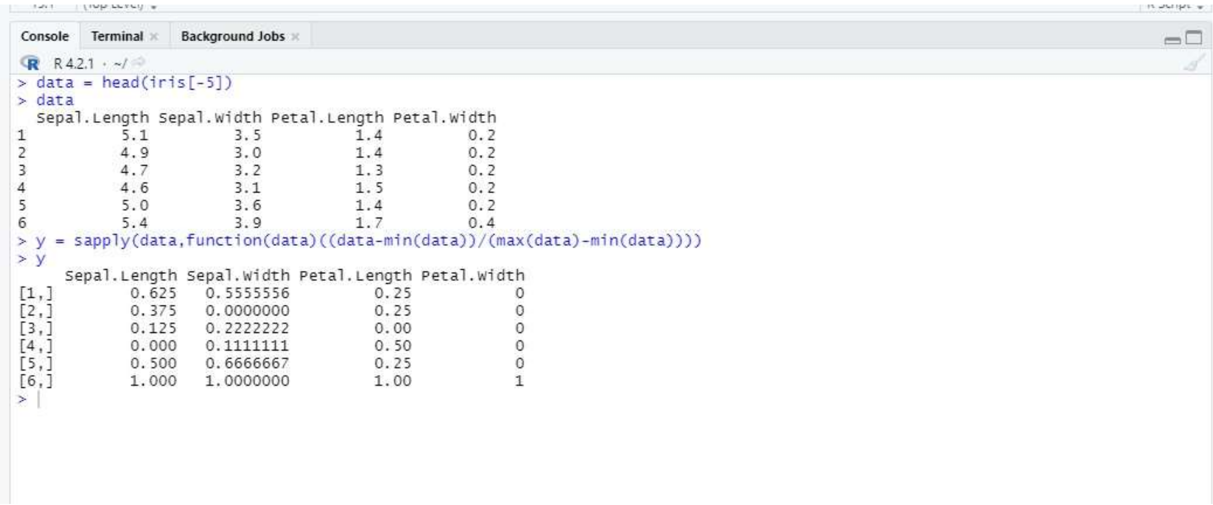
data = head(iris[-5])

data

y = sapply(data,function(data)((data-min(data))/(max(data)-min(data))))

y

Output:



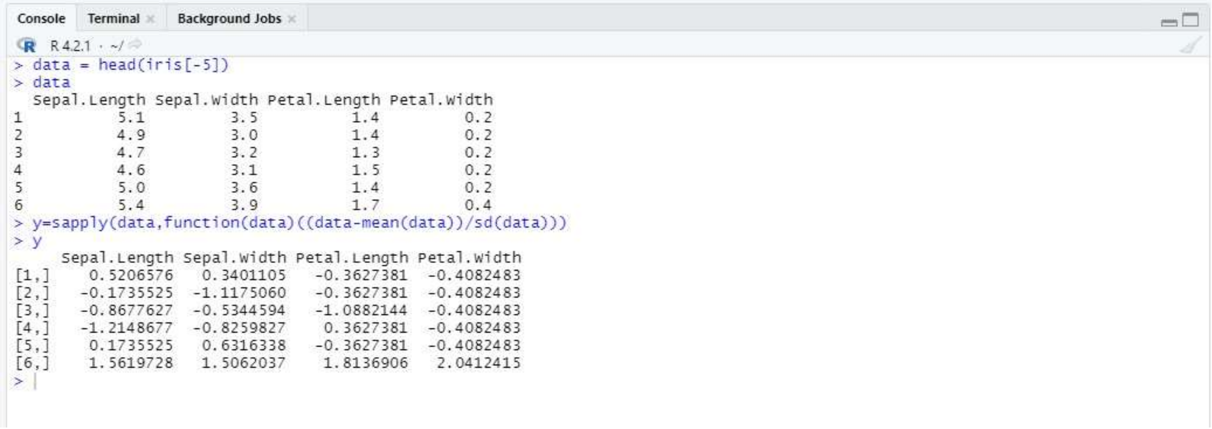
```
> data = head(iris[-5])
> data
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4
> y = sapply(data,function(data)((data-min(data))/(max(data)-min(data))))
> y
     Sepal.Length Sepal.Width Petal.Length Petal.Width
[1,]        0.625   0.5555556         0.25           0
[2,]        0.375   0.0000000         0.25           0
[3,]        0.125   0.2222222         0.00           0
[4,]        0.000   0.1111111         0.50           0
[5,]        0.500   0.6666667         0.25           0
[6,]        1.000   1.0000000         1.00           1
> 
```

b) Z-Scale

Code:

```
data = head(iris[-5])

data

y=sapply(data,function(data)((data-mean(data))/sd(data)))

y
```

Output:



```
> data = head(iris[-5])
> data
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4
> y=sapply(data,function(data)((data-mean(data))/sd(data)))
> y
     Sepal.Length Sepal.Width Petal.Length Petal.Width
[1,]    0.5206576   0.3401105   -0.3627381  -0.4082483
[2,]   -0.1735525  -1.1175060   -0.3627381  -0.4082483
[3,]   -0.8677627  -0.5344594   -1.0882144  -0.4082483
[4,]   -1.2148677  -0.8259827    0.3627381  -0.4082483
[5,]    0.1735525   0.6316338   -0.3627381  -0.4082483
[6,]    1.5619728   1.5062037    1.8136906   2.0412415
> 
```

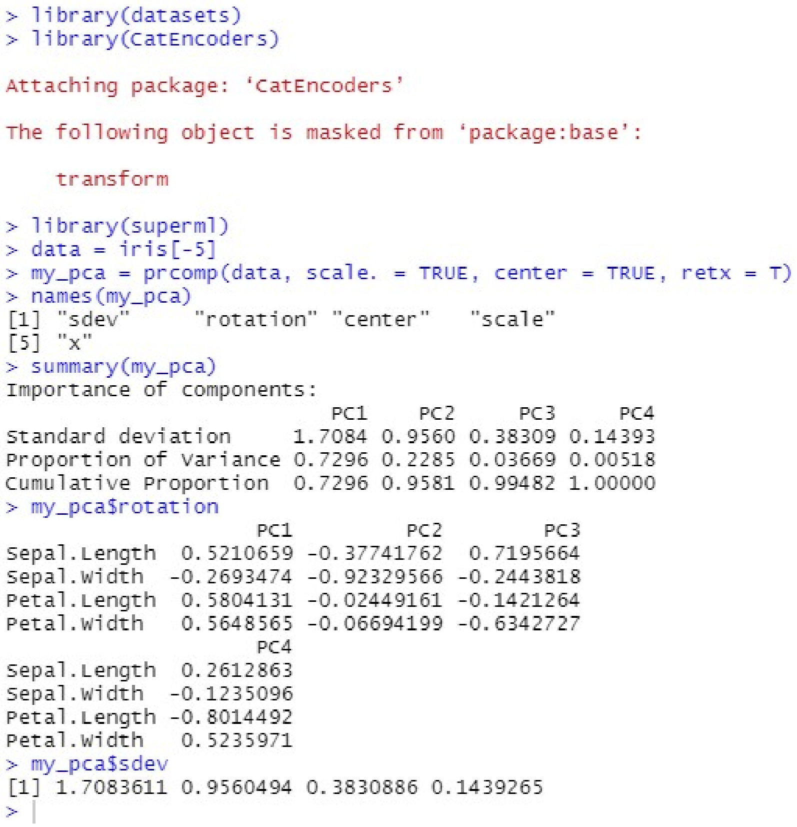
Q4. Find the principal components of IRIS dataset.

Ans 4

Code:

```
library(dplyr)

library(datasets)

library(CatEncoders)

library(superml)

data = iris[-5]

my_pca = prcomp(data, scale. = TRUE, center = TRUE, retx = T)

names(my_pca)

summary(my_pca)

my_pca$rotation

my_pca$sdev
```

Output:



```
> library(datasets)
> library(CatEncoders)

Attaching package: ‘CatEncoders’

The following object is masked from ‘package:base’:

    transform

> library(superml)
> data = iris[-5]
> my_pca = prcomp(data, scale. = TRUE, center = TRUE, retx = T)
> names(my_pca)
[1] "sdev"     "rotation" "center"   "scale"
[5] "x"
> summary(my_pca)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
> my_pca$rotation
                    PC1         PC2        PC3
Sepal.Length  0.5210659 -0.37741762  0.7195664
Sepal.Width  -0.2693474 -0.92329566 -0.2443818
Petal.Length  0.5804131 -0.02449161 -0.1421264
Petal.Width   0.5648565 -0.06694199 -0.6342727
                    PC4
Sepal.Length  0.2612863
Sepal.Width  -0.1235096
Petal.Length -0.8014492
Petal.Width   0.5235971
> my_pca$sdev
[1] 1.7083611 0.9560494 0.3830886 0.1439265
> 
```

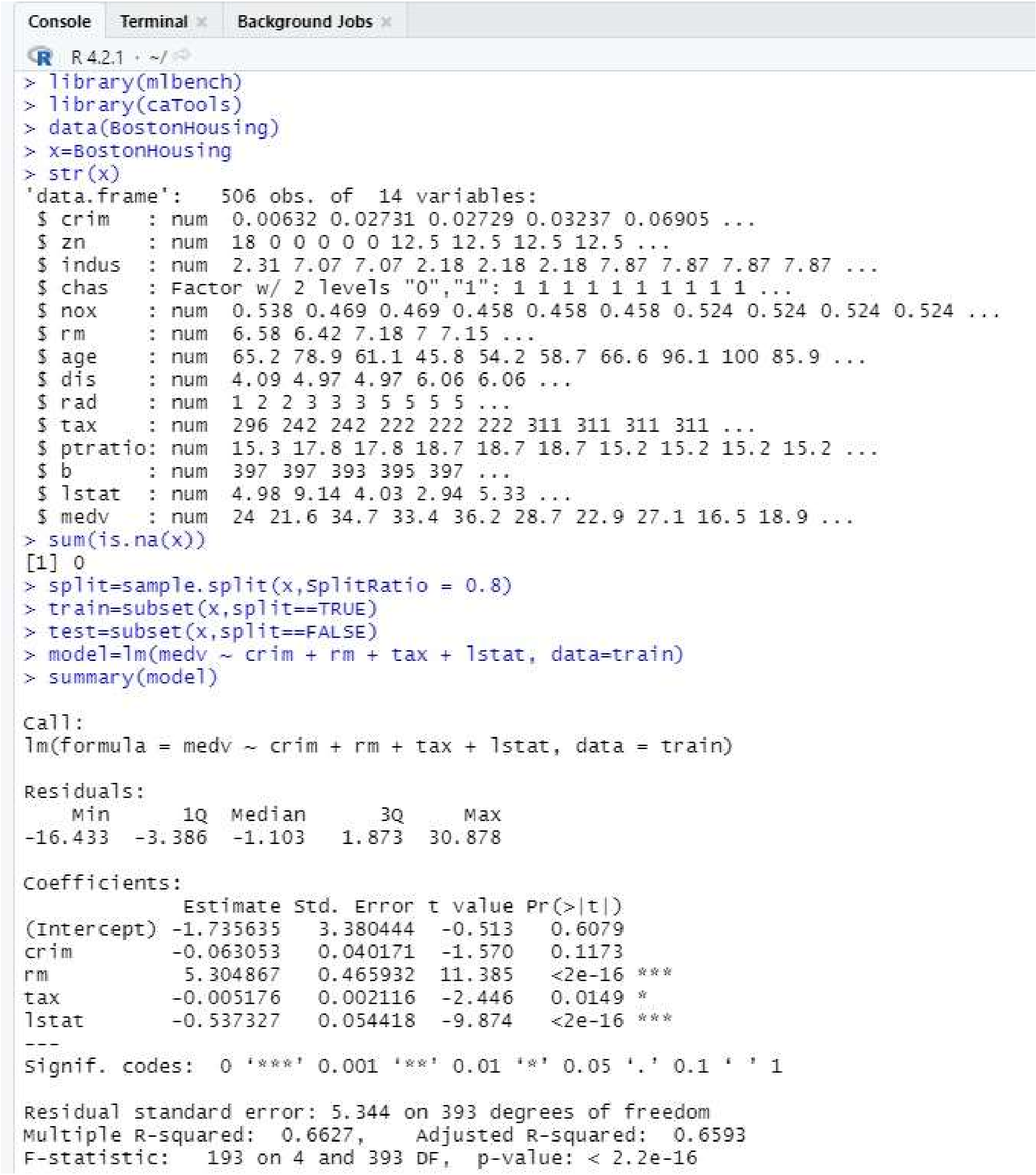
Q5 House rent prediction using linear regression

Ans 5

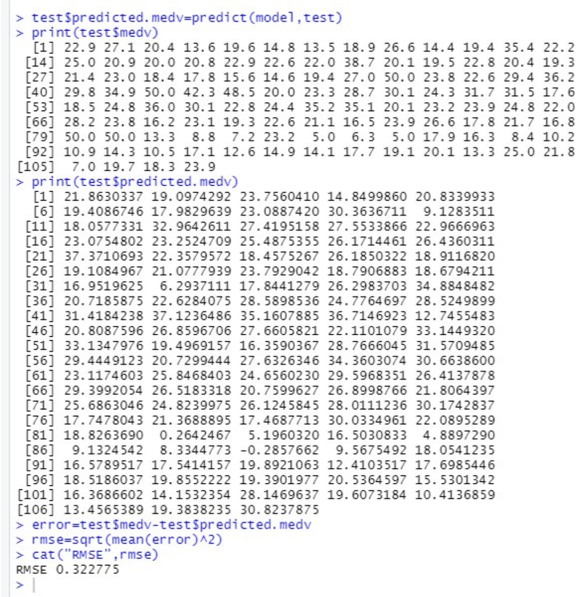
Code:

```
library(mlbench)

library(caTools)

data(BostonHousing)

x=BostonHousing

str(x)

sum(is.na(x))

split=sample.split(x,SplitRatio = 0.8)

train=subset(x,split==TRUE)

test=subset(x,split==FALSE)

model=lm(medv ~ crim + rm + tax + lstat, data=train)

summary(model)

test$predicted.medv=predict(model,test)

print(test$medv)

print(test$predicted.medv)

error=test$medv-test$predicted.medv

rmse=sqrt(mean(error)^2)

cat("RMSE",rmse)
```

Output:



```
> library(mlbench)
> library(caTools)
> data(BostonHousing)
> x=BostonHousing
> str(x)
'data.frame':   506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : num  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ b      : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
> sum(is.na(x))
[1] 0
> split=sample.split(x,SplitRatio = 0.8)
> train=subset(x,split==TRUE)
> test=subset(x,split==FALSE)
> model=lm(medv ~ crim + rm + tax + lstat, data=train)
> summary(model)

Call:
lm(formula = medv ~ crim + rm + tax + lstat, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-16.433  -3.386  -1.103   1.873  30.878 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.735635   3.380444  -0.513   0.6079    
crim        -0.063053   0.040171  -1.570   0.1173    
rm           5.304867   0.465932  11.385   <2e-16 ***
tax         -0.005176   0.002116  -2.446   0.0149 *  
lstat       -0.537327   0.054418  -9.874   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.344 on 393 degrees of freedom
Multiple R-squared:  0.6627,	Adjusted R-squared:  0.6593 
F-statistic:   193 on 4 and 393 DF,  p-value: < 2.2e-16
```



```
> test$predicted.medv=predict(model,test)
> print(test$medv)
  [1] 22.9 27.1 20.4 13.6 19.6 14.8 13.5 18.9 26.6 14.4 19.4 35.4 22.2
 [14] 25.0 20.9 20.0 20.8 22.9 22.6 22.0 38.7 20.1 19.5 22.8 20.4 19.3
 [27] 21.4 23.0 18.4 17.8 15.6 14.6 19.4 27.0 50.0 23.8 22.6 29.4 36.2
 [40] 29.8 34.9 50.0 42.3 48.5 20.0 23.3 28.7 30.1 24.3 31.7 31.5 17.6
 [53] 18.5 24.8 36.0 30.1 22.8 24.4 35.2 35.1 20.1 23.2 23.9 24.8 22.0
 [66] 28.2 23.8 16.2 23.1 19.3 22.6 21.1 16.5 23.9 26.6 17.8 21.7 16.8
 [79] 50.0 50.0 13.3  8.8  7.2 23.2  5.0  6.3  5.0 17.9 16.3  8.4 10.2
 [92] 10.9 14.3 10.5 17.1 12.6 14.9 14.1 17.7 19.1 20.1 13.3 25.0 21.8
[105]  7.0 19.7 18.3 23.9
> print(test$predicted.medv)
  [1] 21.8630337 19.0974292 23.7560410 14.8499860 20.8339933
  [6] 19.4086746 17.9829639 23.0887420 30.3636711  9.1283511
 [11] 18.0577331 32.9642611 27.4195158 27.5533866 22.9666963
 [16] 23.0754802 23.2524709 25.4875355 26.1714461 26.4360311
 [21] 37.3710693 22.3579572 18.4575267 26.1850322 18.9116820
 [26] 19.1084967 21.0777939 23.7929042 18.7906883 18.6794211
 [31] 16.9519625  6.2937111 17.8441279 26.2983703 34.8848482
 [36] 20.7185875 22.6284075 28.5898536 24.7764697 28.5249899
 [41] 31.4184238 37.1236486 35.1607885 36.7146923 12.7455483
 [46] 20.8087596 26.8596706 27.6605821 22.1101079 33.1449320
 [51] 33.1347976 19.4969157 16.3590367 28.7666045 31.5709485
 [56] 29.4449123 20.7299444 27.6326346 34.3603074 30.6638600
 [61] 23.1174603 25.8468403 24.6560230 29.5968351 26.4137878
 [66] 29.3992054 26.5183318 20.7599627 26.8998766 21.8064397
 [71] 25.6863046 24.8239975 26.1245845 28.0111236 30.1742837
 [76] 17.7478043 21.3688895 17.4687713 30.0334961 22.0895289
 [81] 18.8263690  0.2642467  5.1960320 16.5030833  4.8897290
 [86]  9.1324542  8.3344773 -0.2857662  9.5675492 18.0541235
 [91] 16.5789517 17.5414157 19.8921063 12.4103517 17.6985446
 [96] 18.5186037 19.8552222 19.3901977 20.5364597 15.5301342
[101] 16.3686602 14.1532354 28.1469637 19.6073184 10.4136859
[106] 13.4565389 19.3838235 30.8237875
> error=test$medv-test$predicted.medv
> rmse=sqrt(mean(error)^2)
> cat("RMSE",rmse)
RMSE 0.322775
> |
```

Q6 Medical diagnosis for disease spread pattern Using SVM

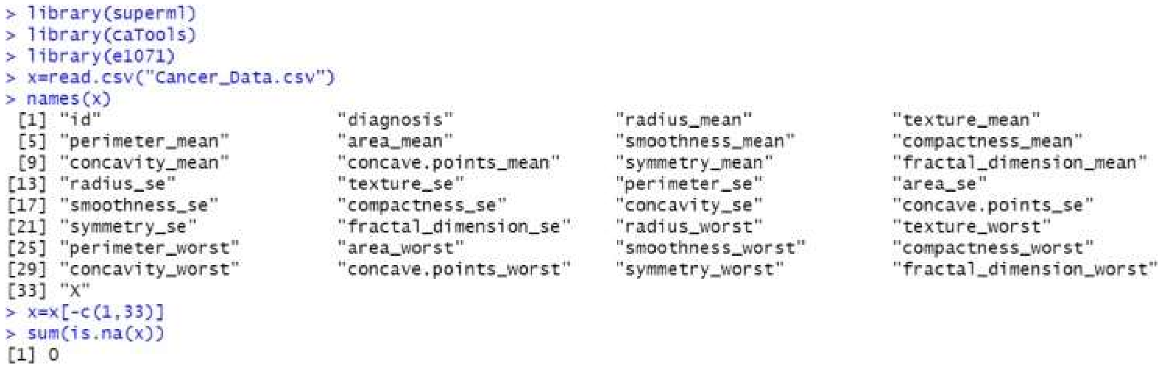
Ans 6

Code:

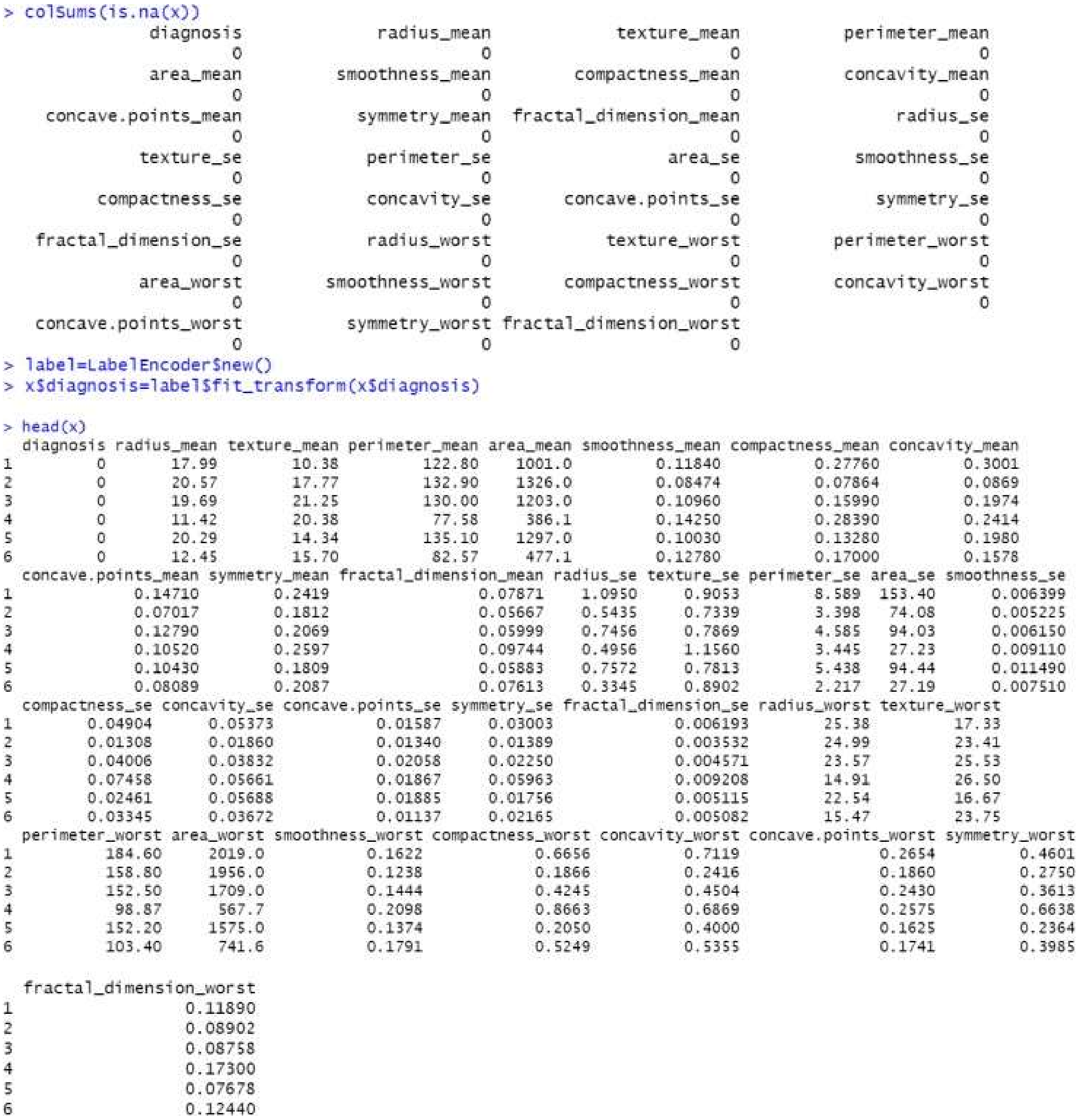
```
library(superml)

library(caTools)

library(e1071)

library(tidyverse)

x=read.csv("Cancer_Data.CSV")

names(x)

x=x[-c(1,33)]

sum(is.na(x))

colSums(is.na(x))

label=LabelEncoder$new()

x$diagnosis=label$fit_transform(x$diagnosis)
```

```
head(x)

split=sample.split(x$diagnosis,SplitRatio = 0.8)

train=subset(x,split==TRUE)

test=subset(x,split==FALSE)

train[-1]=scale(train[-1])

test[-1]=scale(test[-1])

names(train)

classifier=svm(formula=diagnosis~.,

            data = train,

            type = 'C-Classification',

            kernel = 'linear')

Diag_pred=predict(classifier, newdata = test[-1])

cm = table(test[,1], Daig_pred)

print(cm)
```

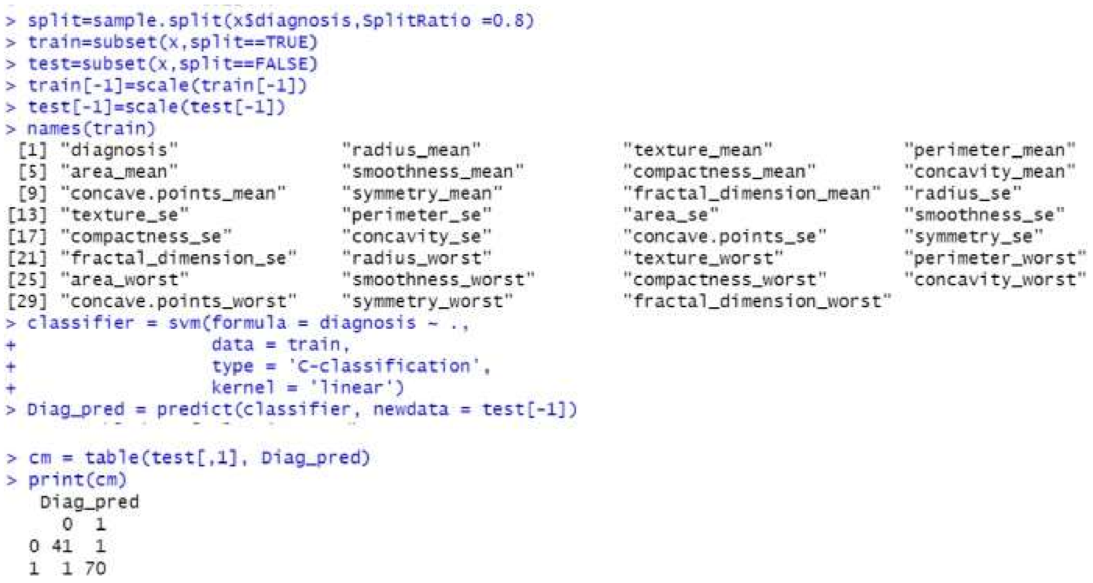
Output:



```
> library(superml)
> library(caTools)
> library(e1071)
> x=read.csv("Cancer_Data.csv")
> names(x)
 [1] "id"                      "diagnosis"               "radius_mean"             "texture_mean"
 [5] "perimeter_mean"          "area_mean"               "smoothness_mean"         "compactness_mean"
 [9] "concavity_mean"          "concave.points_mean"     "symmetry_mean"           "fractal_dimension_mean"
[13] "radius_se"               "texture_se"              "perimeter_se"            "area_se"
[17] "smoothness_se"           "compactness_se"          "concavity_se"            "concave.points_se"
[21] "symmetry_se"             "fractal_dimension_se"    "radius_worst"            "texture_worst"
[25] "perimeter_worst"         "area_worst"              "smoothness_worst"        "compactness_worst"
[29] "concavity_worst"         "concave.points_worst"    "symmetry_worst"          "fractal_dimension_worst"
[33] "X"
> x=x[-c(1,33)]
> sum(is.na(x))
[1] 0
```



```
> colSums(is.na(x))
              diagnosis             radius_mean            texture_mean          perimeter_mean 
                      0                       0                       0                       0 
              area_mean         smoothness_mean        compactness_mean          concavity_mean 
                      0                       0                       0                       0 
    concave.points_mean           symmetry_mean  fractal_dimension_mean               radius_se 
                      0                       0                       0                       0 
             texture_se            perimeter_se                 area_se           smoothness_se 
                      0                       0                       0                       0 
         compactness_se            concavity_se       concave.points_se             symmetry_se 
                      0                       0                       0                       0 
   fractal_dimension_se            radius_worst           texture_worst         perimeter_worst 
                      0                       0                       0                       0 
             area_worst        smoothness_worst       compactness_worst         concavity_worst 
                      0                       0                       0                       0 
   concave.points_worst          symmetry_worst fractal_dimension_worst 
                      0                       0                       0 
> label=LabelEncoder$new()
> x$diagnosis=label$fit_transform(x$diagnosis)

> head(x)
  diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean
1         0       17.99        10.38         122.80    1001.0         0.11840          0.27760         0.3001
2         0       20.57        17.77         132.90    1326.0         0.08474          0.07864         0.0869
3         0       19.69        21.25         130.00    1203.0         0.10960          0.15990         0.1974
4         0       11.42        20.38          77.58     386.1         0.14250          0.28390         0.2414
5         0       20.29        14.34         135.10    1297.0         0.10030          0.13280         0.1980
6         0       12.45        15.70          82.57     477.1         0.12780          0.17000         0.1578
  concave.points_mean symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se area_se smoothness_se
1             0.14710        0.2419                0.07871    1.0950     0.9053        8.589  153.40      0.006399
2             0.07017        0.1812                0.05667    0.5435     0.7339        3.398   74.08      0.005225
3             0.12790        0.2069                0.05999    0.7456     0.7869        4.585   94.03      0.006150
4             0.10520        0.2597                0.09744    0.4956     1.1560        3.445   27.23      0.009110
5             0.10430        0.1809                0.05883    0.7572     0.7813        5.438   94.44      0.011490
6             0.08089        0.2087                0.07613    0.3345     0.8902        2.217   27.19      0.007510
  compactness_se concavity_se concave.points_se symmetry_se fractal_dimension_se radius_worst texture_worst
1        0.04904      0.05373           0.01587     0.03003             0.006193        25.38         17.33
2        0.01308      0.01860           0.01340     0.01389             0.003532        24.99         23.41
3        0.04006      0.03832           0.02058     0.02250             0.004571        23.57         25.53
4        0.07458      0.05661           0.01867     0.05963             0.009208        14.91         26.50
5        0.02461      0.05688           0.01885     0.01756             0.005115        22.54         16.67
6        0.03345      0.03672           0.01137     0.02165             0.005082        15.47         23.75
  perimeter_worst area_worst smoothness_worst compactness_worst concavity_worst concave.points_worst symmetry_worst
1          184.60     2019.0           0.1622            0.6656          0.7119               0.2654         0.4601
2          158.80     1956.0           0.1238            0.1866          0.2416               0.1860         0.2750
3          152.50     1709.0           0.1444            0.4245          0.4504               0.2430         0.3613
4           98.87      567.7           0.2098            0.8663          0.6869               0.2575         0.6638
5          152.20     1575.0           0.1374            0.2050          0.4000               0.1625         0.2364
6          103.40      741.6           0.1791            0.5249          0.5355               0.1741         0.3985

  fractal_dimension_worst
1                 0.11890
2                 0.08902
3                 0.08758
4                 0.17300
5                 0.07678
6                 0.12440
```



```
> split=sample.split(x$diagnosis,SplitRatio =0.8)
> train=subset(x,split==TRUE)
> test=subset(x,split==FALSE)
> train[-1]=scale(train[-1])
> test[-1]=scale(test[-1])
> names(train)
 [1] "diagnosis"               "radius_mean"             "texture_mean"            "perimeter_mean"
 [5] "area_mean"               "smoothness_mean"         "compactness_mean"        "concavity_mean"
 [9] "concave.points_mean"     "symmetry_mean"           "fractal_dimension_mean"  "radius_se"
[13] "texture_se"              "perimeter_se"            "area_se"                 "smoothness_se"
[17] "compactness_se"          "concavity_se"            "concave.points_se"       "symmetry_se"
[21] "fractal_dimension_se"    "radius_worst"            "texture_worst"           "perimeter_worst"
[25] "area_worst"              "smoothness_worst"        "compactness_worst"       "concavity_worst"
[29] "concave.points_worst"    "symmetry_worst"          "fractal_dimension_worst"
> classifier = svm(formula = diagnosis ~ .,
+                  data = train,
+                  type = 'C-classification',
+                  kernel = 'linear')
> Diag_pred = predict(classifier, newdata = test[-1])

> cm = table(test[,1], Diag_pred)
> print(cm)
   Diag_pred
     0  1
  0 41  1
  1  1 70
```