multiple-regression-analysis.R

Sharad Deshmukh

2022-03-01

```
D=read.table("C:\\Users\\Sharad Deshmukh\\Desktop\\MSC=SEM-
II\\practical\\NationalFootballLeague.txt",header=TRUE,sep="\t",dec=".")
##Data preproccing
summary(D)  ##No missing value

##        X               y                 x1             x2             x3
##  Min.   : 1.00   Min.   : 0.000   Min.   :1416   Min.   :1414   Min.   
:35.10
##  1st Qu.: 7.75   1st Qu.: 4.000   1st Qu.:1896   1st Qu.:1714   1st 
Qu.:37.38
##  Median :14.50   Median : 6.500   Median :2111   Median :2106   Median 
:38.85
##  Mean   :14.50   Mean   : 6.964   Mean   :2110   Mean   :2127   Mean   
:38.64
##  3rd Qu.:21.25   3rd Qu.:10.000   3rd Qu.:2303   3rd Qu.:2474   3rd 
Qu.:39.70
##  Max.   :28.00   Max.   :13.000   Max.   :2971   Max.   :2929   Max.   
:42.30
##        x4              x5                x6                x7
##  Min.   :38.10   Min.   :-22.00   Min.   : 576.0   Min.   :43.80
##  1st Qu.:52.42   1st Qu.: -5.75   1st Qu.: 710.5   1st Qu.:54.77
##  Median :57.70   Median :  1.00   Median : 787.5   Median :58.65
##  Mean   :59.40   Mean   :  0.00   Mean   : 789.9   Mean   :58.16
##  3rd Qu.:68.80   3rd Qu.:  6.25   3rd Qu.: 869.8   3rd Qu.:61.10
##  Max.   :78.30   Max.   : 19.00   Max.   :1037.0   Max.   :67.50
##        x8             x9
##  Min.   :1457   Min.   :1575
##  1st Qu.:1848   1st Qu.:1913
##  Median :2050   Median :2101
##  Mean   :2110   Mean   :2128
##  3rd Qu.:2320   3rd Qu.:2328
##  Max.   :2876   Max.   :2670

##reducing the dimension
dim(D)

## [1] 28 11

names(D)

##  [1] "X"  "y"  "x1" "x2" "x3" "x4" "x5" "x6" "x7" "x8" "x9"

str(D)
```

```
## 'data.frame':    28 obs. of  11 variables:
##  $ X : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ y : int  10 11 11 13 10 11 10 11 4 2 ...
##  $ x1: int  2113 2003 2957 2285 2971 2309 2528 2147 1689 2566 ...
##  $ x2: int  1985 2855 1737 2905 1666 2927 2341 2737 1414 1838 ...
##  $ x3: num  38.9 38.8 40.1 41.6 39.2 39.7 38.1 37 42.1 42.3 ...
##  $ x4: num  64.7 61.3 60 45.3 53.8 74.1 65.4 78.3 47.6 54.2 ...
##  $ x5: int  4 3 14 -4 15 8 12 -1 -3 -1 ...
##  $ x6: int  868 615 914 957 836 786 754 761 714 797 ...
##  $ x7: num  59.7 55 65.6 61.4 66.1 61 66.1 58 57 58.9 ...
##  $ x8: int  2205 2096 1847 1903 1457 1848 1564 1821 2577 2476 ...
##  $ x9: int  1917 1575 2175 2476 1866 2339 2092 1909 2001 2254 ...
```

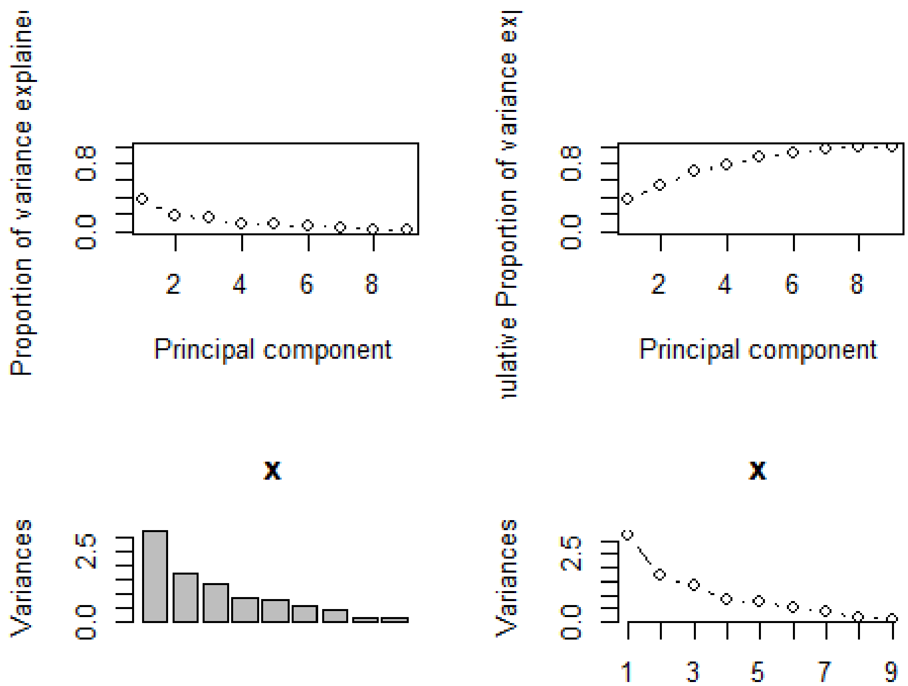
```
D=D[,-1]
wineClasses <- factor(D[,1])

#plot(main="Three Different
Cultivars",D[,2],D[,3],D[,4],D[,5],D[,6],D[,7],D[,8],D[,9],D[,10], col =
wineClasses)

##PCA
dimPCA <- prcomp(scale(D[,-1]))
#Step 3: Choose the principal components with highest variances
#Now the 13 features has reduced to only 2 new Principal Components These are
not 2 of those 13, but 2 new components
summary(dimPCA)
```

```
## Importance of components:
##                           PC1    PC2   PC3     PC4     PC5    PC6     PC7
## Standard deviation     1.7954 1.3035 1.146 0.90734 0.85870 0.7336 0.64352
## Proportion of Variance 0.3582 0.1888 0.146 0.09147 0.08193 0.0598 0.04601
## Cumulative Proportion  0.3582 0.5469 0.693 0.78445 0.86638 0.9262 0.97219
##                           PC8     PC9
## Standard deviation     0.3612 0.34612
## Proportion of Variance 0.0145 0.01331
## Cumulative Proportion  0.9867 1.00000
```

```
pcaCharts <- function(x) {
  x.var <- x$sdev ^ 2
  x.pvar <- x.var/sum(x.var)
  print("proportions of variance:")
  print(x.pvar)

  par(mfrow=c(2,2))
  plot(x.pvar,xlab="Principal component", ylab="Proportion of variance
explained", ylim=c(0,1), type='b')
  plot(cumsum(x.pvar),xlab="Principal component", ylab="Cumulative Proportion
of variance explained", ylim=c(0,1), type='b')
  screeplot(x)
  screeplot(x,type="l")
```

```
  par(mfrow=c(1,1))
}
pcaCharts(dimPCA)
```

```
## [1] "proportions of variance:"
## [1] 0.35815489 0.18877760 0.14604473 0.09147301 0.08192947 0.05979687
0.04601363
## [8] 0.01449860 0.01331121
```




```
###Check VIF factor(Multicollinearity)
library(car)
```

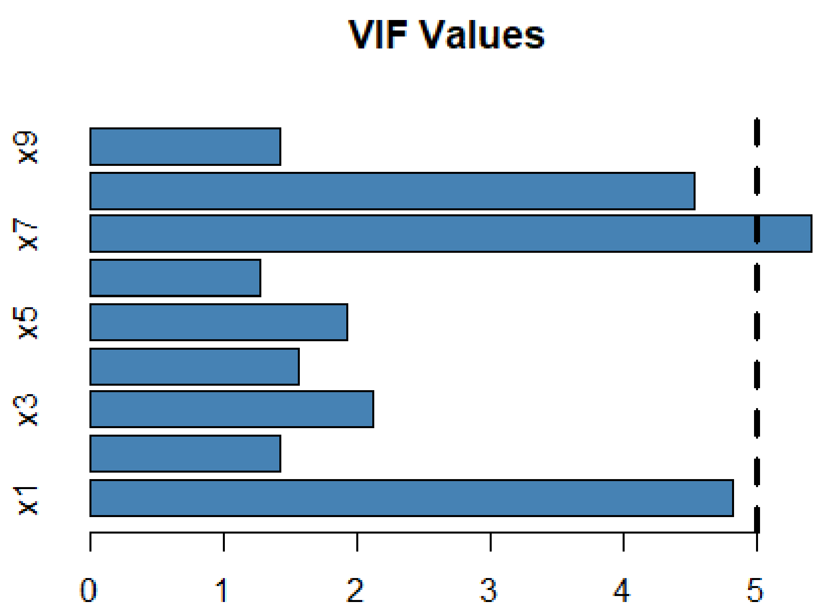
```
## Warning: package 'car' was built under R version 4.1.2
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.1.2
```

```
#create vector of VIF values
model=lm(y~.,D)
vif_values <- vif(model)

#create horizontal bar chart to display each VIF value
barplot(vif_values, main = "VIF Values", horiz = TRUE, col = "steelblue")
#add vertical line at 5
abline(v = 5, lwd = 3, lty = 2)
##
```

```
library(tidyverse) #tidyverse for easy data manipulation and visualization

## Warning: package 'tidyverse' was built under R version 4.1.1

## -- Attaching packages --------------------------------------- tidyverse
1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.1.1

## Warning: package 'stringr' was built under R version 4.1.1

## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::recode() masks car::recode()
## x purrr::some()   masks car::some()
```




```
library(caret)      #caret for easy machine learning workflow

## Warning: package 'caret' was built under R version 4.1.2
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

library(leaps)     #leaps, for computing stepwise regression

## Warning: package 'leaps' was built under R version 4.1.2

model <- lm(y~.,data=D)
library(MASS)

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

# Fit the full model
full.model <- lm(y ~., data = D)
# Stepwise regression model
step.model <- stepAIC(full.model, direction = "both",
                      trace = FALSE)
summary(step.model)

##
## Call:
## lm(formula = y ~ x2 + x7 + x8 + x9, data = D)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3519 -0.5612 -0.0856  0.6972  3.2802
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.8217034  7.7847061  -0.234  0.81705
## x2           0.0038186  0.0007051   5.416 1.67e-05 ***
## x7           0.2168941  0.0886759   2.446  0.02252 *
## x8          -0.0040149  0.0013983  -2.871  0.00863 **
## x9          -0.0016349  0.0012460  -1.312  0.20244
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.681 on 23 degrees of freedom
## Multiple R-squared:  0.8012, Adjusted R-squared:  0.7666
## F-statistic: 23.17 on 4 and 23 DF,  p-value: 8.735e-08
```

```r
#lm(formula = y ~ x2 + x7 + x8 + x9, data = D) ##Best model using stepwise regression



smp_siz = floor(0.75*nrow(D))
set.seed(123)
train_ind = sample(seq_len(nrow(D)),size = smp_siz)
train =D[train_ind,]
test=D[-train_ind,]

##Fitting the important model
model1=lm(lm(formula = y ~ x2 + x7 + x8 + x9, data = train))
summary(model1)

##
## Call:
## lm(formula = lm(formula = y ~ x2 + x7 + x8 + x9, data = train))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.41776 -0.42020 -0.03068  0.98249  2.57813
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.129e+00  7.507e+00  -0.550  0.58992
## x2           3.234e-03  8.151e-04   3.968  0.00110 **
## x7           2.298e-01  9.089e-02   2.529  0.02235 *
## x8          -4.440e-03  1.448e-03  -3.066  0.00739 **
## x9           2.662e-05  1.607e-03   0.017  0.98698
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.496 on 16 degrees of freedom
## Multiple R-squared:  0.8419, Adjusted R-squared:  0.8024
## F-statistic:  21.3 on 4 and 16 DF,  p-value: 3.02e-06

##x2,x7,x8 are three significant variables
pre=predict(model1,test)
z=data.frame(test[,1],floor(pre))
z

##    test...1. floor.pre.
## 1         10          6
## 2         11          8
## 6         11         11
## 12        10          7
## 21         3          6
```

```
## 23          4           4
## 27          2           1
```

```r
sigma(model1)/mean(D[,1])
```

```
## [1] 0.2148519
```

```r
##21% mean prediction error
```