DA5020.P3.LUO

Xinxin Luo

4/15/2020

##Problem 1 (95 Points) 1.1:(0 Pts) Load the data set on franchise sales. The variables in the data set are: NetSales = net sales in $1000s for a franchise; StoreSize = size of store in 1000s square-feet; InvValue = inventory value in $1000s; AdvBudget = advertising budget in $1000s; DistrictSize = number of households in sales district in 1000s ; NumComp = number of competing stores in sales district. Do you detect any multi-collinearity that would affect the construction of a multiple regression model?

```r
sales <- read.csv("franchisesales.csv")
summary(sales)
```

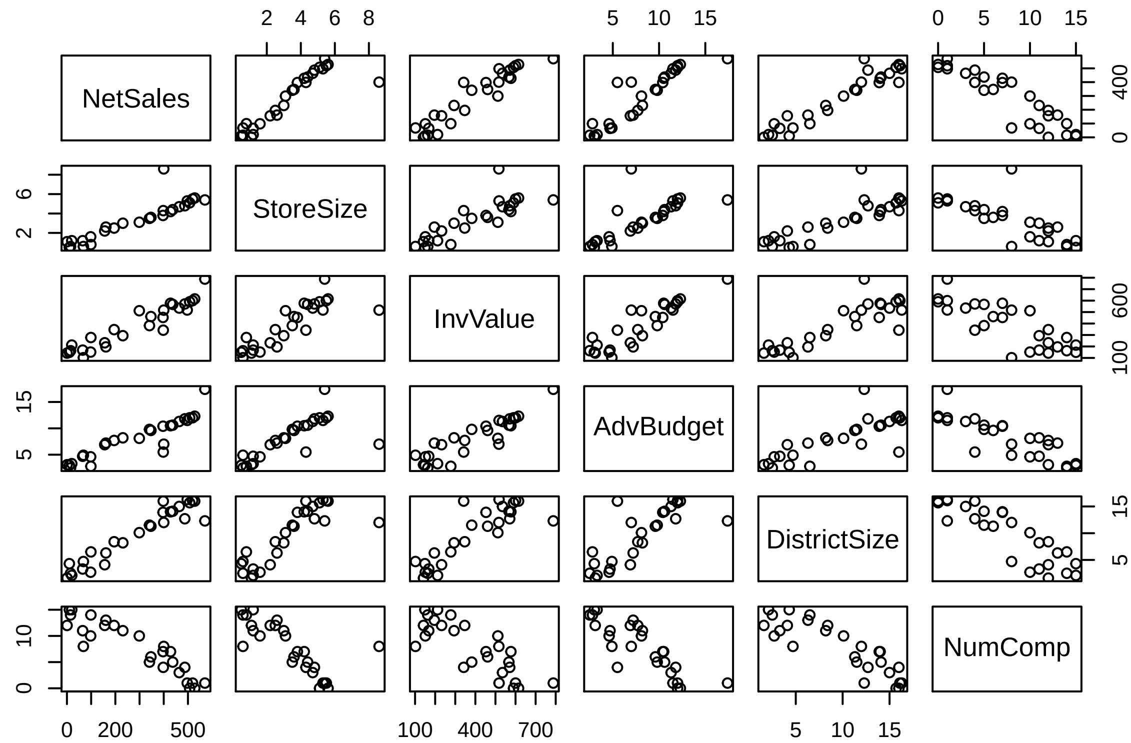
```
##     NetSales       StoreSize        InvValue       AdvBudget    
##  Min.   :  0.5   Min.   :0.500   Min.   :102.0   Min.   : 2.50  
##  1st Qu.: 98.5   1st Qu.:1.400   1st Qu.:204.0   1st Qu.: 4.80  
##  Median :341.0   Median :3.500   Median :382.0   Median : 8.10  
##  Mean   :286.6   Mean   :3.326   Mean   :387.5   Mean   : 8.10  
##  3rd Qu.:450.5   3rd Qu.:4.750   3rd Qu.:551.0   3rd Qu.:10.95  
##  Max.   :570.0   Max.   :8.600   Max.   :788.0   Max.   :17.40  
##   DistrictSize       NumComp      
##  Min.   : 1.600   Min.   : 0.000  
##  1st Qu.: 4.500   1st Qu.: 4.000  
##  Median :11.300   Median : 8.000  
##  Mean   : 9.693   Mean   : 7.741  
##  3rd Qu.:14.050   3rd Qu.:12.000  
##  Max.   :16.300   Max.   :15.000
```

```r
plot(sales)
cor(sales)
```

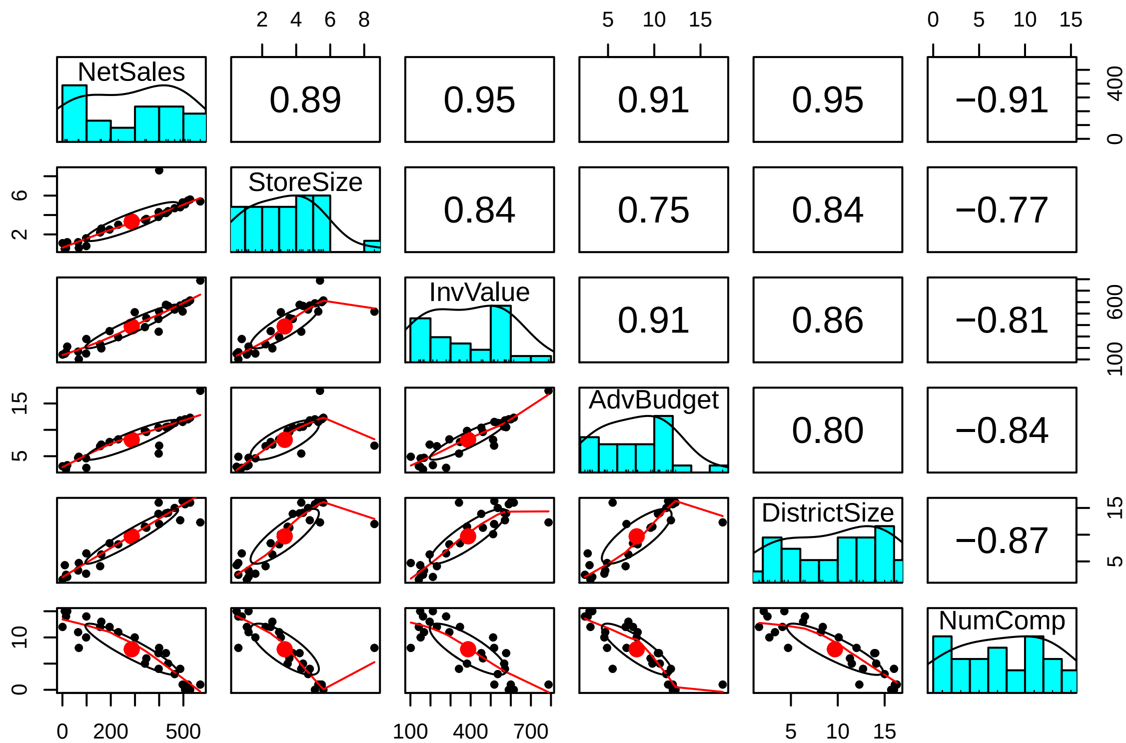
```
##                NetSales  StoreSize   InvValue  AdvBudget DistrictSize
## NetSales      1.0000000  0.8940921  0.9455036  0.9140241    0.9536831
## StoreSize     0.8940921  1.0000000  0.8436158  0.7485872    0.8380229
## InvValue      0.9455036  0.8436158  1.0000000  0.9062306    0.8639169
## AdvBudget     0.9140241  0.7485872  0.9062306  1.0000000    0.7954345
## DistrictSize  0.9536831  0.8380229  0.8639169  0.7954345    1.0000000
## NumComp      -0.9122364 -0.7657378 -0.8073804 -0.8412800   -0.8695896
##                 NumComp
## NetSales     -0.9122364
## StoreSize    -0.7657378
## InvValue     -0.8073804
## AdvBudget    -0.8412800
## DistrictSize -0.8695896
## NumComp       1.0000000
```

```r
#load library psych

library(psych)
```




```r
pairs.panels(sales)
```




```
# cor() function returns a full correlation matrix between all variables from the data frame.
# comment: When x variables in the correlation matrix show linear relationships, there is an indication
```

1.2: (20 Pts) Normalize all columns, except NetSales, using z-score standardization.

```r
# Z score normalization for every feature except first column NetSales
sales_z<-scale(sales[,2:6], center = TRUE, scale = TRUE)

# check standardization
summary(sales_z)
```

```
##    StoreSize          InvValue          AdvBudget         DistrictSize
##  Min.   :-1.40520   Min.   :-1.49336   Min.   :-1.4836   Min.   :-1.5744
##  1st Qu.:-0.95767   1st Qu.:-0.95979   1st Qu.:-0.8743   1st Qu.:-1.0102
##  Median : 0.08656   Median :-0.02867   Median : 0.0000   Median : 0.3127
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.70813   3rd Qu.: 0.85537   3rd Qu.: 0.7551   3rd Qu.: 0.8477
##  Max.   : 2.62255   Max.   : 2.09512   Max.   : 2.4639   Max.   : 1.2855
##     NumComp
##  Min.   :-1.58110
##  1st Qu.:-0.76407
##  Median : 0.05296
##  Mean   : 0.00000
##  3rd Qu.: 0.86999
##  Max.   : 1.48276
```

1.3: (50 Pts) Implement the k-NN algorithm in R (do not use an implementation of k-NN from a package); write a function called kNN-predict(data,y,x,k) that takes a data set of predictor variables, a set of NetSales values, a new set of values for the variables, and a k and returns a prediction. To predict a continuous variable you need to calculate the distances of x to all observations in data, then take the k closest cases and average the NetSales values for those cases. That average is your prediction.

```
# write a function called kNN-predicat, whereas data=set of predictor variables, y=Netsales values, x =
kNN_predict <- function(data,y,x,k){
  # 1. Find neighbors
  n <- nrow(data)
  ds <- numeric(n)
  for (i in 1:n) {
    ds[i] <- sqrt(sum(data[i,]-x)^2) # or can write as dist(ds, method="euclidean")

  # 2. Order the k neighbors
  ordered.neighbors <- order(ds)
  k.closest <- ordered.neighbors[1:k]

  # 3. Find the average of the NetSales Values
  return(mean(y[k.closest]))
  }
}
```

1.4: (10 Pts) Use your algorithm with a k=3 to predict net sales of a store with the following values for the variables in order: (4.2, 601, 7.8, 14.2, 6). Compare that prediction to the one you obtained in Assignment 10.

```
# create a new item
new <- data.frame(StoreSize=4.2, InvValue=601, AdvBudget=7.8, DistrictSize=14.2, NumComp=6)

# Z-score standardization function
normalize <- function(x, mean, sd) {
return ((x - mean) / sd) }

# load necessary library
library(tibble)
mean <- as_tibble(lapply(sales[,-1], mean))
sd <- as_tibble(lapply(sales[,-1], sd))

# Normalize each new case using z-score and whole data's mean and sd, as new cases contains 0 values.
new.n <- mapply(normalize,new,mean,sd)

# Make Prediction
new_predict <- kNN_predict(sales_z,sales$NetSales, new.n, 3)

new_predict
```

```
## [1] 228.3333
```

```
# Comment: the prediction use kNN_predict is 228.33, compared to 405.02 from Assignment 10.
```

1.5: (15 Pts) Calculate the mean square error (MSE) for the kNN by predicting each actual value in the data set and comparing it to the actual observation. Compare the MSE to the MSE you calculated in Assignment 10 and comment on the difference. Which model is better?

```r
# find out number of rows in sales dataset
n <- nrow(sales)

# predict each NetSales value using kNN by creating a new column in the dataset
sales$predicted <- numeric(n)
for (i in 1:n) {
  sales$predicted[i] <- kNN_predict(sales_z,sales$NetSales, sales_z[i,], 3)
}

# write MSE function
mse <- function(x,y){
  return(mean((x-y)^2))
}

# MSE for using the kNN_predict function formulated.
mse(sales$predicted,sales$NetSales)
```

```
## [1] 39273.77
```

```r
# Comment: the MSE is 39274, which is quite different from Assignment 10's MSE = 242.27.
```

##Problem 2 (15 Points) 2.1: (10 Pts) Determine an optimal k by trying all values from 1 through 7 for your own k-NN algorithm implementation against the cases in the entire data set (if we had a larger data set, we would split it into training and validation data to avoid overfitting). What is the optimal k, i.e., the k that results in the best accuracy as measured by smallest MSE?

```r
# create 7 predicted columns for k values from 1 through 7
predict_index <- c("k_1","k_2","k_3","k_4","k_5","k_6","k_7")

for (s in predict_index) {
  sales[,s] <- 0}

# create separate dataframe to store all these values
sales2 <- sales[,c(8:14)]

# predicted NetSales values for k from 1 through 7
for (i in 1:7){
    for (j in 1:n) {
  sales2[j,i] <- kNN_predict(sales_z,sales$NetSales, sales_z[j,], i)

  }
}
```

```r
# calculate MSE for NetSales with k value varies from 1 through 7
mse_values <- numeric(7)

for (i in 1:7){
  mse_values[i]<- mse(sales2[,i],sales$NetSales)
}

# find out the optiomal value for k

cat("The optimal k value for my own k-NN algorithm implementation is", which(mse_values==min(mse_values)
```

```
## The optimal k value for my own k-NN algorithm implementation is 4 , which is 35467.79 .
```

2.2: (5 Pts) Create a plot of k (x-axis) versus MSE using ggplot.

```r
# load library
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```r
# create k vector
k <- c(1:7)

# create a ggplot
ggplot() + geom_bar(aes(k, mse_values), stat="identity", position = "dodge",fill="blue")
```