Assignment 5

Sreshtha Chatterjee 0724

2026-02-26

PRACTICE SET 3

3. Problem to demonstrate the role of qualitative (ordinal) predictors in addition to quantitative predictors in multiple linear regression.

Consider “diamonds” data set in R. It is in the ggplot2 package. Make a list of all the ordinal categorical variables. Identify the response.

```
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.5.2

data("diamonds")
View(diamonds)
```

ANSWER: The ordinal categorical variables in the diamonds dataset are “cut”, “color”, and “clarity”. The response variable is “price”.

(a) Run a linear regression of the response on the quality of cut. Write the fitted regression model.

```
model_cut <- lm(price ~ cut, data = diamonds)
```

(b) Test whether the expected price of diamond with premium cut is significantly different from that of the ideal cut.

```
summary(model_cut)

##
## Call:
## lm(formula = price ~ cut, data = diamonds)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -4258  -2741  -1494   1360  15348
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4062.24      25.40 159.923  < 2e-16 ***
## cut.L        -362.73      68.04  -5.331  9.8e-08 ***
## cut.Q        -225.58      60.65  -3.719    2e-04 ***
## cut.C        -699.50      52.78 -13.253  < 2e-16 ***
## cut^4        -280.36      42.56  -6.588  4.5e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 3964 on 53935 degrees of freedom
## Multiple R-squared:  0.01286,    Adjusted R-squared:  0.01279
## F-statistic: 175.7 on 4 and 53935 DF,  p-value: < 2.2e-16
```

(c) What is the expected price of a diamond of ideal cut?

```
ideal_price <- predict(model_cut, newdata = data.frame(cut = "Ideal"))
ideal_price

##        1
## 3457.542
```

(d) Modify the regression model in (a) by incorporating the predictor “table”. Write the fitted regression model.

```
model_cut_table <- lm(price ~ cut + table, data = diamonds)
model_cut_table

##
## Call:
## lm(formula = price ~ cut + table, data = diamonds)
##
## Coefficients:
## (Intercept)        cut.L        cut.Q        cut.C        cut^4
table
##    -6340.26       -14.24       -65.60      -517.97      -130.07
179.10
```

(e) Test for the significance of “table” in predicting the price of diamond.

```
summary(model_cut_table)

##
## Call:
## lm(formula = price ~ cut + table, data = diamonds)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -5630  -2694  -1458   1346  15690
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6340.256    537.007 -11.807  < 2e-16 ***
## cut.L         -14.244     70.145  -0.203  0.83908
## cut.Q         -65.600     61.000  -1.075  0.28219
## cut.C        -517.970     53.423  -9.696  < 2e-16 ***
## cut^4        -130.066     43.112  -3.017  0.00255 **
## table         179.105      9.236  19.393  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3950 on 53934 degrees of freedom
## Multiple R-squared:  0.0197, Adjusted R-squared:  0.01961
## F-statistic: 216.7 on 5 and 53934 DF,  p-value: < 2.2e-16
```

(f) Find the average estimated price of a diamond with an average table value and which is of fair cut.

```
average_table <- mean(diamonds$table)
average_table

## [1] 57.45718

pred= predict(model_cut_table,newdata =
data.frame(cut="Fair",table=average_table));pred

##        1
## 4072.798
```

Problem Set 5: K nearest neighbors regression

```
set.seed(123)
library(class)
library(caret)

## Warning: package 'caret' was built under R version 4.5.2

## Loading required package: lattice

library(FNN)

## Warning: package 'FNN' was built under R version 4.5.2

##
## Attaching package: 'FNN'

## The following objects are masked from 'package:class':
##
##     knn, knn.cv
```

Q1. Problem to demonstrate the utility of K nearest neighbour regression over least squares regression.

Consider a setting with n = 1000 observations. Generate

(i) x1i from N(0,22) and x2i from Poisson(λ = 1.5).

(ii) ϵi from N(0,1).

(iii) yi = −2 +1.4x1i −2.6x2i +ϵi.

```
set.seed(123)

n <- 1000

x1 <- rnorm(n, mean = 0, sd = 2)
x2 <- rpois(n, lambda = 1.5)
```

```r
epsilon <- rnorm(n, mean = 0, sd = 1)

y <- -2 + 1.4*x1 - 2.6*x2 + epsilon

data <- data.frame(x1, x2, y)
```

Split the data into train and test sets. Keep the first 800 observations as training data and the remaining as test data.

```r
train_data <- data[1:800, ]
test_data  <- data[801:1000, ]
```

Work out the following:

1. Fit a multiple linear regression equation of y on x1 and x2. Calculate test MSE.

```r
lm_model <- lm(y ~ x1 + x2, data = train_data)

summary(lm_model)

## 
## Call:
## lm(formula = y ~ x1 + x2, data = train_data)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -3.0727 -0.6573 -0.0125  0.6921  3.2412 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.07300    0.05382  -38.52   <2e-16 ***
## x1           1.38207    0.01767   78.21   <2e-16 ***
## x2          -2.55584    0.02768  -92.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.98 on 797 degrees of freedom
## Multiple R-squared:  0.9492, Adjusted R-squared:  0.9491 
## F-statistic:  7445 on 2 and 797 DF,  p-value: < 2.2e-16

pred_lm <- predict(lm_model, newdata = test_data)

mse_lm <- mean((test_data$y - pred_lm)^2)

mse_lm

## [1] 0.998901
```

2. Fit a KNN model with k = 1,2,5,9,15. Calculate test MSE for each choice of k.

```r
library(FNN)

train_x <- train_data[, c("x1", "x2")]
```

```r
test_x  <- test_data[, c("x1", "x2")]
train_y <- train_data$y

pred_k1 <- knn.reg(train = train_x, test = test_x, y = train_y, k = 1)$pred

mse_k1 <- mean((test_data$y - pred_k1)^2)
mse_k1

## [1] 2.219793

pred_k2 <- knn.reg(train = train_x, test = test_x, y = train_y, k = 2)$pred

mse_k2 <- mean((test_data$y - pred_k2)^2)
mse_k2

## [1] 1.729587

pred_k5 <- knn.reg(train = train_x, test = test_x, y = train_y, k = 5)$pred

mse_k5 <- mean((test_data$y - pred_k5)^2)
mse_k5

## [1] 1.303978

pred_k9 <- knn.reg(train = train_x, test = test_x, y = train_y, k = 9)$pred

mse_k9 <- mean((test_data$y - pred_k9)^2)
mse_k9

## [1] 1.205371

pred_k15 <- knn.reg(train = train_x, test = test_x, y = train_y, k = 15)$pred

mse_k15 <- mean((test_data$y - pred_k15)^2)
mse_k15

## [1] 1.235776
```

Suppose the data in Step (iii) is generated as :

yi = 1/(−2+1.4x1i −2.6x2i +2.9x21i) + 3.1sin(x2i) − 1.5x1ix22i + ϵi.

Work out the problems in (1) and (2). Compare and comment on the results.

```r
epsilon2 <- rnorm(n, mean = 0, sd = 1)

y_nl <- 1/(-2 + 1.4*x1 - 2.6*x2 + 2.9*(x1^2)) + 3.1*sin(x2) - 1.5*x1*(x2^2) +
epsilon2

data_nl <- data.frame(x1, x2, y = y_nl)

train_nl <- data_nl[1:800, ]
test_nl  <- data_nl[801:1000, ]
```

```r
lm_model_nl <- lm(y ~ x1 + x2, data = train_nl)

summary(lm_model_nl)

##
## Call:
## lm(formula = y ~ x1 + x2, data = train_nl)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -242.908   -4.964    0.106    5.261  157.805
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.1933     1.0257  -0.188   0.8506
## x1           -6.3755     0.3368 -18.930   <2e-16 ***
## x2            1.3565     0.5275   2.571   0.0103 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.68 on 797 degrees of freedom
## Multiple R-squared:  0.3151, Adjusted R-squared:  0.3134
## F-statistic: 183.4 on 2 and 797 DF,  p-value: < 2.2e-16

pred_lm_nl <- predict(lm_model_nl, newdata = test_nl)

mse_lm_nl <- mean((test_nl$y - pred_lm_nl)^2)

mse_lm_nl

## [1] 211.4074

train_x_nl <- train_nl[, c("x1", "x2")]
test_x_nl  <- test_nl[, c("x1", "x2")]
train_y_nl <- train_nl$y

pred_k1_nl <- knn.reg(train = train_x_nl, test = test_x_nl, y = train_y_nl, k
= 1)$pred

mse_k1_nl <- mean((test_nl$y - pred_k1_nl)^2)
mse_k1_nl

## [1] 50.06576

pred_k2_nl <- knn.reg(train = train_x_nl, test = test_x_nl,
                      y = train_y_nl, k = 2)$pred

mse_k2_nl <- mean((test_nl$y - pred_k2_nl)^2)
mse_k2_nl

## [1] 57.34949
```

```r
pred_k5_nl <- knn.reg(train = train_x_nl, test = test_x_nl,
                      y = train_y_nl, k = 5)$pred

mse_k5_nl <- mean((test_nl$y - pred_k5_nl)^2)
mse_k5_nl

## [1] 63.71091

pred_k9_nl <- knn.reg(train = train_x_nl, test = test_x_nl,
                      y = train_y_nl, k = 9)$pred

mse_k9_nl <- mean((test_nl$y - pred_k9_nl)^2)
mse_k9_nl

## [1] 66.16254

pred_k15_nl <- knn.reg(train = train_x_nl, test = test_x_nl,
                       y = train_y_nl, k = 15)$pred

mse_k15_nl <- mean((test_nl$y - pred_k15_nl)^2)
mse_k15_nl

## [1] 68.46044
```

Interpretation:

Linear True Model:

1.Linear regression gives lowest Test MSE. 2.KNN slightly worse. 3.Because the true relationship is linear.

Nonlinear True Model

1.Linear regression performs poorly (misspecified). 2.KNN performs better for moderate k (usually 5 or 9). 3.Small k → high variance. 4.Large k → high bias.