Assignment_5_710

PROTYAY CHATTERJEE

2026-02-26

PROBLEM 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.

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

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

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

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

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

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

```
library(ggplot2)
```

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

```
library(MASS)
```

```
lvl = as.numeric(diamonds$cut)

code_fun=function(x){
  if(x-5==0){
    return(c(0,0,0,0))
  }else if(x-5==-1){
    return(c(1,0,0,0))
  }else if(x-5==-2){
    return(c(1,1,0,0))
  }else if(x-5==-3){
    return(c(1,1,1,0))
  }else if(x-5==-4){
    return(c(1,1,1,1))
  }
}

coded_pred = t(sapply(lvl,code_fun))
colnames(coded_pred)=c("Premium","Very_Good","Good","Fair")
```

```r
diamonds = cbind(diamonds,coded_pred)

ord_mod = lm(price ~ Premium + Very_Good + Good + Fair, data=diamonds)
summary(ord_mod)
```

```
##
## Call:
## lm(formula = price ~ Premium + Very_Good + Good + Fair, data = diamonds)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -4258  -2741  -1494   1360  15348
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3457.54      27.00 128.051  < 2e-16 ***
## Premium      1126.72      43.22  26.067  < 2e-16 ***
## Very_Good    -602.50      49.39 -12.198  < 2e-16 ***
## Good          -52.90      67.10  -0.788  0.43055
## Fair          429.89     113.85   3.776  0.00016 ***
## ---
## 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
```

(a) The fitted regression model is:

$$\widehat{price} = \hat{\beta}_0 + \hat{\beta}_1(Premium) + \hat{\beta}_2(Very\_Good) + \hat{\beta}_3(Good) + \hat{\beta}_4(Fair)$$

Where:

Ideal cut is the reference category.

Each coefficient represents the incremental change moving down the cut hierarchy.

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

Hypotheses:

$H_0 : \mu_{Premium} = \mu_{Ideal}$

against

$H_1 : \mu_{Premium} \neq \mu_{Ideal}$

From the output, we have:

$$p < 2 \times 10^{-16}$$

Since,p <0.05,reject H_0.

The expected price of a Premium cut diamond is significantly different from that of an Ideal cut diamond.

Also, $\mu_{Ideal} - \mu_{Premium} = -1126.72$*, it implies:*

$\mu_{Premium} - \mu_{Ideal} = 1126.72$

So Premium diamonds are estimated to cost about 1126.72 more than Ideal on average.

(c) The expected price of a diamond with Ideal cut is approximately 3457.542

(d) The fitted regression model after incorporating the predictor "table" is:

$$\widehat{price} = \hat{\beta}_0 + \hat{\beta}_1(\text{Premium}) + \hat{\beta}_2(\text{Very\_Good}) + \hat{\beta}_3(\text{Good}) + \hat{\beta}_4(\text{Fair}) + \hat{\beta}_5(\text{table})$$

```
m2=lm(price ~ cut + table, data = diamonds)
summary(m2)
```

```
##
## 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
```

```
coef(m2)
```

```
## (Intercept)       cut.L       cut.Q       cut.C       cut^4       table
## -6340.25613   -14.24440   -65.59998  -517.97015  -130.06587   179.10483
```

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

Hypotheses:

H_0 *:β{table}=0*

against

H_1 *:β{table}≠0*

From the output, we have:

$$p < 2 \times 10^{-16}$$

Since,p<0.05,reject H_0.

So, table is highly significant for predicting diamond price (after accounting for cut).

```
mean_table=mean(diamonds$table)
mean_table
```

```
## [1] 57.45718
```

```
predict(m2, newdata = data.frame(
  cut = factor("Fair", levels = levels(diamonds$cut)),
  table = mean_table
))
```

```
##        1
## 4072.798
```

(f) The average estimated price for a Fair cut diamond at table(mean) = 57.46 is: 4072.8

PROBLEM SET 5

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

Consider a setting with n = 1000 observations. Generate

(i) x_{1i} from $N(0, 2^2)$ and x_{2i} from Poisson(λ = 1.5).

(ii) ε_i from $N(0,1)$.

(iii) $y_i = -2 + 1.4x_{1i} - 2.6x_{2i} + \varepsilon_i$.

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

1. Fit a multiple linear regression equation of y on x_1 and x_2. Calculate test MSE.

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

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

$$y_i = \frac{1}{(-2+1.4x_{1i}-2.6x_{2i}+2.9x_{1i}^2)} + 3.1\sin(x_{2i}) - 1.5x_{1i}x_{2i}^2 + \varepsilon_i.$$

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

PART A:

```
# Generate data

set.seed(123)
# Generate predictors
x1 <- rnorm(1000, mean = 0, sd = 2)
x2 <- rpois(1000, lambda = 1.5)

# Generate error
eps <- rnorm(1000, mean = 0, sd = 1)

# Generate response (linear model)
y <- -2 + 1.4*x1 - 2.6*x2 + eps

# Create data frame
df <- data.frame(y, x1, x2)

# Split data
train_data <- df[1:800, ]
test_data  <- df[801:1000, ]
```

1. Multiple Linear Regression

```
lm_fit <- lm(y ~ x1 + x2, data = train_data)
summary(lm_fit)

##
## 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
```

```
# Predictions on test set
y_pred_lm <- predict(lm_fit, newdata = test_data)

# Test MSE
mse_lm <- mean((test_data$y - y_pred_lm)^2)
mse_lm
```

```
## [1] 0.998901
```

Conclusion (Linear Regression – Linear DGP)

The estimated coefficients should be close to the true values:

Intercept ≈ -2

Coefficient of x_1 ≈ 1.4

Coefficient of x_2 ≈ -2.6

The test MSE is expected to be small.

Since the true model is linear, the linear regression model is correctly specified.

Therefore, linear regression performs very well in this case.

2. KNN Regression

```
##install.packages("caret")

library(caret)
```

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

```
## Loading required package: lattice
```

```
k_values <- c(1, 2, 5, 9, 15)
mse_knn_linear <- numeric(length(k_values))
```

```r
for(i in seq_along(k_values)){
  knn_fit <- knnreg(y ~ x1 + x2, data = train_data, k = k_values[i])
  y_pred_knn <- predict(knn_fit, test_data)
  mse_knn_linear[i] <- mean((test_data$y - y_pred_knn)^2)
}

data.frame(k = k_values, Test_MSE = mse_knn_linear)
```

```
##    k Test_MSE
## 1  1 2.219793
## 2  2 1.729587
## 3  5 1.303978
## 4  9 1.205371
## 5 15 1.232730
```

Conclusion (KNN Regression – Linear DGP)

KNN with k = 1 may have a very low training MSE but can overfit, leading to higher test MSE. As k increases, the model becomes smoother, reducing variance but increasing bias. Conclusion (KNN – Linear DGP)

For small k (e.g., 1), test MSE may be higher due to high variance (overfitting). As k increases, MSE may decrease initially and then increase again.

However, overall KNN does not outperform linear regression here.

Since the true relationship is linear, the parametric linear model is more efficient.

Part B :

Now, we will generate the response using the non-linear model and repeat the analysis.

Nonlinear Data Generating Process

```r
set.seed(123)
# Generate nonlinear response
y_nl <- 1/(-2 + 1.4*x1 - 2.6*x2 + 2.9*x1^2) +
        3.1*sin(x2) -
        1.5*x1*(x2^2) + eps

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

# Split data
train_data_nl <- df_nl[1:800, ]
test_data_nl  <- df_nl[801:1000, ]
```

1. Multiple Linear Regression on Nonlinear Data

```r
lm_fit_nl <- lm(y ~ x1 + x2, data = train_data_nl)
summary(lm_fit_nl)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = train_data_nl)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -241.819   -5.150   -0.051    5.566  155.662
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.2369     1.0272  -0.231  0.81764
## x1           -6.3747     0.3373 -18.901  < 2e-16 ***
## x2            1.3849     0.5283   2.621  0.00892 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.7 on 797 degrees of freedom
## Multiple R-squared:  0.3146, Adjusted R-squared:  0.3129
## F-statistic: 182.9 on 2 and 797 DF,  p-value: < 2.2e-16
```

```
y_pred_lm_nl <- predict(lm_fit_nl, newdata = test_data_nl)

mse_lm_nl <- mean((test_data_nl$y - y_pred_lm_nl)^2)
mse_lm_nl
```

```
## [1] 205.1776
```

Conclusion (Linear Regression – Nonlinear DGP)

The linear model is misspecified.

It cannot capture:

- The quadratic term x_{1i}^2
- The sine function $\sin(x_{2i})$
- The interaction term $x_{1i}x_{2i}^2$

As a result, test MSE is relatively large.

Linear regression under fits the data. 2. KNN Regression on Nonlinear Data

```
mse_knn_nonlinear <- numeric(length(k_values))

for(i in seq_along(k_values)){
  knn_fit_nl <- knnreg(y ~ x1 + x2, data = train_data_nl, k = k_values[i])
  y_pred_knn_nl <- predict(knn_fit_nl, test_data_nl)
  mse_knn_nonlinear[i] <- mean((test_data_nl$y - y_pred_knn_nl)^2)
}

data.frame(k = k_values, Test_MSE = mse_knn_nonlinear)
```

```
##    k Test_MSE
## 1  1 47.52490
## 2  2 54.48942
## 3  5 59.77963
## 4  9 62.10913
## 5 15 63.72303
```

Conclusion (KNN – Nonlinear DGP)

KNN adapts to nonlinear patterns automatically.

Moderate values of k (such as 5 or 9) typically give the lowest test MSE.

Very small k leads to high variance.

Very large k leads to high bias.

KNN significantly outperforms linear regression in this case.

Final Comparison and Discussion

Linear Case:

Linear regression has the lowest test MSE.

KNN with small k (e.g., k = 1) may overfit.

Larger k reduces variance but increases bias.

Nonlinear Case:

Linear regression is misspecified and produces larger test MSE.

KNN adapts to nonlinear structure.

Moderate values of k (e.g., 5 or 9) often give the best performance.

Very small k → high variance.

Very large k → high bias.