# ASSIGNMENT_5_MDTS4214_0709

Saurabh Mishra

2026-02-26

## Problem Set 3: Multiple Linear Regression

**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

The ordinal (ordered) categorical variables are:

cut: Fair < Good < Very Good < Premium < Ideal

color: D < E < F < G < H < I < J

clarity: I1 < SI2 < SI1 < VS2 < VS1 < VVS2 < VVS1 < IF

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

```
library(ggplot2)

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

library(MASS)

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

contrasts(diamonds$cut)=contr.sdif(5)

m1=lm(price ~ cut, data = diamonds)
summary(m1)

##
## 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 ***
## cut2-1       -429.89     113.85  -3.776  0.00016 ***
## cut3-2         52.90      67.10   0.788  0.43055
## cut4-3        602.50      49.39  12.198  < 2e-16 ***
## cut5-4      -1126.72      43.22 -26.067  < 2e-16 ***
## ---
## 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
```

```r
coef(m1)
```

```
## (Intercept)      cut2-1      cut3-2      cut4-3      cut5-4
##  4062.23636  -429.89331    52.89544   602.49781 -1126.71573
```

**price^ =4062.24+-429.89(Cut2-1)+52.89(Cut3-2)+602.50(Cut4-3)-1126.715(Cut5-4)**

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

Hypotheses:

$H0$:$\mu$(Premium)=$\mu$(Ideal) against $H1$:$\mu$(Premium)≠$\mu$(Ideal)

From the output, we have:

$p$<2×10^(−16)

Since,$p$<0.05,reject $H0$.

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

```r
predict(m1,
        newdata = data.frame(
          cut = factor("Ideal", levels=levels(diamonds$cut))
        ))
```

```
##        1
## 3457.542
```

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

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

```r
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 ***
## cut2-1       -365.568    113.504  -3.221  0.00128 ** 
## cut3-2        185.162     67.220   2.755  0.00588 ** 
## cut4-3        461.015     49.761   9.265  < 2e-16 ***
## cut5-4       -626.220     50.215 -12.471  < 2e-16 ***
## 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)      cut2-1      cut3-2      cut4-3      cut5-4       table 
##  -6340.2561   -365.5679    185.1625    461.0147   -626.2203    179.1048
```

**price^ = −6340.256 −365.568(cut2-1) +185.163(cut3-2) +461.015(cut4-3) −626.220(cut5-4) +179.105(table)**

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

Hypotheses:

$H0$:β(table)=0 against $H1$:β(table)≠0

From the output, we have:

$p$<2×10^(−16)

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

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

```r
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
```

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

## Problem Set 5: K nearest neighbours regression

### 1 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, 2^2) and x2i from Poisson(λ = 1.5).

(ii) εi from N(0, 1).

(iii) yi = −2 + 1.4x1i − 2.6x2i + εi.

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:

```
set.seed(123)
n= 1000
x1=rnorm(n, mean=0, sd=2)
x2=rpois(n, lambda=1.5)
eps=rnorm(n, 0, 1)

y=-2 +1.4*x1 -2.6*x2 +eps

data=data.frame(x1, x2, y)

train=data[1:800, ]
test=data[801:1000, ]
```

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

```
lm_fit=lm(y ~ x1 + x2, data=train)
summary(lm_fit)

##
## Call:
## lm(formula = y ~ x1 + x2, data = train)
##
## 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_fit, newdata=test)

mse_lm=mean((test$y - pred_lm)^2)
mse_lm

## [1] 0.998901
```

Test MSE ≈ 1

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

```
#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))

for(i in seq_along(k_values)){
  knn_fit=knnreg(y ~ x1 + x2, data = train, k = k_values[i])
  y_pred_knn=predict(knn_fit, test)
  mse_knn_linear[i]=mean((test$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 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.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.

Suppose the data in Step (iii) is generated as : yi = 1/(−2 + 1.4x1i − 2.6x2i + 2.9x1i^2) + 3.1 sin(x2i) −1.5x1ix2i^2 + εi.

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

```
y2=1/(-2 + 1.4*x1 - 2.6*x2 + 2.9*x1^2) +
      3.1*sin(x2) -
      1.5*x1*(x2^2) +
      eps

data2=data.frame(x1,x2,y=y2)

train2=data2[1:800,]
test2=data2[801:1000,]

lm_fit2 <- lm(y ~ x1 + x2, data=train2)
summary(lm_fit2)

##
## Call:
## lm(formula = y ~ x1 + x2, data = train2)
##
## 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

pred_lm2 <- predict(lm_fit2, newdata=test2)

mse_lm2 <- mean((test2$y - pred_lm2)^2)
mse_lm2

## [1] 205.1776
```

Since it is a Misspecified Model, it has a large MSE.

```r
mse_knn_nonlinear=numeric(length(k_values))

for(i in seq_along(k_values)){
  knn_fit_nl=knnreg(y ~ x1 + x2, data = train2, k = k_values[i])
  y_pred_knn_nl=predict(knn_fit_nl, test2)
  mse_knn_nonlinear[i]=mean((test2$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: * In the linear model, the best KNN is at moderate k (9) and gives low test MSE (~1.21).

- In the nonlinear model, the best KNN is at very small k (1), because the relationship is complex and needs high flexibility.

This clearly demonstrates the bias–variance tradeoff,i.e; * If k is small then we get, low bias & high variance * If k is large then we get, higher bias & lower variance