Predictive_PS3(Q3) & PS5_735

Srinjana Sen

2026-02-26

PREDICTIVE ANALYTICS

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.

```
library(ggplot2)
```

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

```
head(diamonds$cut)
```

```
## [1] Ideal     Premium   Good      Premium   Good      Very Good
## Levels: Fair < Good < Very Good < Premium < Ideal
```

```
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
  {
    return(c(1,1,1,1))
```

```
  }
}

pred = t(sapply(lvl,code_fun))
colnames(pred)=c("Premium","Very_Good","Good","Fair")
diamonds=cbind(diamonds,pred)
model = lm(price ~ Premium+Very_Good+Good+Fair, data = diamonds)
summary(model)

##
## 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
```

The fitted model is: Price^hat = 3457.54 + 1126.72 * Premium - 602.50 * Very Good - 52.90 * Good + 429.89 * Fair.

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

Yes it is, since BetaIdeal = 3457.54, and Beta Premium = 3457.54 + 1126.72. Expected Price of a Premium cut diamond increases by 1126.72 units than that of an ideal cut diamond.

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

3457.54

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

```
m2 = lm(price ~ Premium+Very_Good+Good+Fair+table, data = diamonds)
library(stargazer)

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

```
##
## Please cite as:

##  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary
Statistics Tables.

##  R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```
stargazer(m2,type = "text")
```

```
##
## ================================================
##                         Dependent variable:
##                     ---------------------------
##                                price
## ------------------------------------------------
## Premium                      626.220***
##                               (50.215)
##
## Very_Good                    -461.015***
##                               (49.761)
##
## Good                         -185.162***
##                               (67.220)
##
## Fair                         365.568***
##                               (113.504)
##
## table                        179.105***
##                                (9.236)
##
## Constant                    -6,563.672***
##                               (517.450)
##
## ------------------------------------------------
## Observations                   53,940
## R2                              0.020
## Adjusted R2                     0.020
## Residual Std. Error   3,950.136 (df = 53934)
## F Statistic         216.744*** (df = 5; 53934)
## ================================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

Fitted regression model is: Price^hat = -6,563.672 + 626.220 * Premium - 461.015 * Very Good - 185.162 * Good + 365.568 * Fair + 179.105 * table

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

p value for the predictor table is less than 0.01. This shows that table is an important predictor and significantly influences the expected price of diamonds.

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

Price^hat = -6,563.672 + 626.220 * Premium - 461.015 * Very Good - 185.162 * Good + 365.568 * Fair + 179.105 * table

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.

```
set.seed(123)
n=1000
```

Generate
(i) x1i from N(0, 2^2) and x2i from Poisson(λ = 1.5).

```
x1i=rnorm(n, mean = 0, sd = 2)
x2i=rpois(n, lambda = 1.5)
```

(ii) εi from N(0, 1).

```
ei=rnorm(n, mean = 0, sd = 1)
```

(iii) yi = −2 + 1.4x1i − 2.6x2i + εi.

```
yi=-2 + 1.4*x1i - 2.6*x2i + ei
```

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

```
data=data.frame(y = yi, x1 = x1i, x2 = x2i)
head(data)
```

```
##            y         x1 x2
## 1 -4.3903185 -1.1209513  0
## 2 -2.9517542 -0.4603550  0
## 3  1.4622853  3.1174166  0
## 4 -3.7755078  0.1410168  1
## 5 -3.1176393  0.2585755  1
## 6 -0.2706045  3.4301300  2
```

```
train_data=data[1:800, ]
test_data=data[801:1000, ]
train_x=data.frame(train_data$x1,train_data$x2)
test_x=data.frame(test_data$x1,test_data$x2)
```

```
train_y=train_data$y
test_y=test_data$y
```

Work out the following:
1. Fit a multiple linear regression equation of y on x1 and x2. Calculate test MSE.

```
model=lm(y ~ x1 + x2, data = train_data)
summary(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

y_pred=predict(model, newdata = test_data)
test_mse=mean((test_data$y - y_pred)^2)
test_mse

## [1] 0.998901
```

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

```
library(FNN)

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

k_values=c(1, 2, 5, 9, 15)
test_mse_knn=numeric(length(k_values))
for (i in seq_along(k_values)) {
  k=k_values[i]
  pred=knn.reg(
    train = train_x,
    test  = test_x,
    y     = train_y,
    k     = k
```

```
  )$pred
  test_mse_knn[i]=mean((test_y - pred)^2)
}
data.frame(k = k_values, Test_MSE = test_mse_knn)
```

```
##    k Test_MSE
## 1  1 2.219793
## 2  2 1.729587
## 3  5 1.303978
## 4  9 1.205371
## 5 15 1.235776
```

Suppose the data in Step (iii) is generated as:
yi = 1/(−2 + 1.4x1i − 2.6x2i + 2.9(x1i)^2) + 3.1 sin(x2i) − 1.5x1i(x2i)^2 + εi.
Work out the problems in (1) and (2). Compare and comment on the results.

```
yi=(1/(-2+1.4*x1i-2.6*x2i+2.9*(x1i^2)))+3.1*sin(x2i)-1.5*x1i*(x2i^2)+ei
data=data.frame(y = yi, x1 = x1i, x2 = x2i)
head(data)
```

```
##            y         x1 x2
## 1  12.581968 -1.1209513  0
## 2  -0.799890 -0.4603550  0
## 3  -0.869362  3.1174166  0
## 4   2.793949  0.1410168  1
## 5   3.093778  0.2585755  1
## 6 -15.603221  3.4301300  2
```

```
train_data=data[1:800, ]
test_data=data[801:1000, ]
train_x=data.frame(train_data$x1,train_data$x2)
test_x=data.frame(test_data$x1,test_data$x2)
train_y=train_data$y
test_y=test_data$y

model=lm(y ~ x1 + x2, data = train_data)
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = train_data)
##
## 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=predict(model, newdata = test_data)
test_mse=mean((test_data$y - y_pred)^2)
test_mse
```

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

```
library(FNN)
k_values=c(1, 2, 5, 9, 15)
test_mse_knn=numeric(length(k_values))
for (i in seq_along(k_values)) {
  k=k_values[i]
  pred=knn.reg(
    train = train_x,
    test  = test_x,
    y     = train_y,
    k     = k
  )$pred
  test_mse_knn[i]=mean((test_y - pred)^2)
}
data.frame(k = k_values, Test_MSE = test_mse_knn)
```

```
##    k Test_MSE
## 1  1 47.52490
## 2  2 54.48942
## 3  5 59.77963
## 4  9 62.10913
## 5 15 63.72010
```

In the first set of results, the test MSE decreases from k = 1 to k = 9 and then increases slightly at k = 15. This is the expected behavior of KNN due to the bias–variance tradeoff. When k is very small, the model has high variance and tends to overfit, leading to larger test error. As k increases, variance reduces and the error decreases, reaching a minimum at a moderate value of k. If k becomes too large, the model becomes overly smooth, bias increases, and the test error rises slightly. The magnitude of the MSE (around 1–2) is reasonable given that the true error variance is 1, indicating correct implementation, most likely with proper standardization of predictors.

In the second set, the test MSE values are extremely large (around 50–60) and increase steadily as k increases. This pattern is not theoretically expected. Such inflated errors strongly suggest that the predictors were not standardized before applying KNN. Since KNN relies on Euclidean distance, differences in scale across variables can distort

neighborhood formation and severely worsen prediction performance. Therefore, the first set reflects proper modeling, while the second set likely results from incorrect preprocessing.