MDTS4214 Assignment 5 Roll 703

Priyanshu Dey

2026-02-25

Predictive Analysis

Problem set 4

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)
diamond_df = diamonds
```

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

```
head(diamond_df$cut)
```

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

```
cut_numeric = as.numeric(diamond_df$cut)

create_dummies = function(val)
{
  if (val - 5 == 0)
  {
    return(c(0,0,0,0))
  }
  else if (val - 5 == -1)
  {
    return(c(1,0,0,0))
  }else if (val - 5 == -2)
  {
    return(c(1,1,0,0))
  }else if (val - 5 == -3)
  {
    return(c(1,1,1,0))
  }
  else
  {
    return(c(1,1,1,1))
  }
}
```

```r
dummy_vars = t(sapply(cut_numeric, create_dummies))
colnames(dummy_vars) = c("Premium", "Very_Good", "Good", "Fair")
diamond_df = cbind(diamond_df, dummy_vars)

model_base = lm(price ~ Premium + Very_Good + Good + Fair, data = diamond_df)
summary(model_base)

##
## Call:
## lm(formula = price ~ Premium + Very_Good + Good + Fair, data = diamond_df)
##
## 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: $ = 3457.54 + 1126.72 - 602.50 - 52.90 + 429.89 $

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

Yes it is, since $\beta_{Ideal} = 3457.54$, and $\beta_{Premium} = 3457.54 + 1126.72$. The expected price of a Premium cut diamond increases by 1126.72 units compared to 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.

```r
model_table = lm(price ~ Premium + Very_Good + Good + Fair + table, data =
diamond_df)
library(stargazer)
```

```
##
## 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(model_table, 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: $ = -6563.672 + 626.220 - 461.015 - 185.162 + 365.568 + 179.105 $

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

The 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.

$ = -6563.672 + 626.220 - 461.015 - 185.162 + 365.568 + 179.105 $

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).**

```
set.seed(123)
obs_count = 1000
var_x1 = rnorm(obs_count, 0, 2)
var_x2 = rpois(obs_count, 1.5)
```

(ii) ε_i **from N(0, 1).**

```
err_term = rnorm(obs_count, 0, 1)
```

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

```
y_lin = -2 + 1.4 * var_x1 - 2.6 * var_x2 + err_term

sim_data_linear = data.frame(var_x1, var_x2, err_term, y_lin)
head(sim_data_linear)

##       var_x1 var_x2   err_term      y_lin
## 1 -1.1209513      0 -0.8209867 -4.3903185
## 2 -0.4603550      0 -0.3072572 -2.9517542
## 3  3.1174166      0 -0.9020980  1.4622853
## 4  0.1410168      1  0.6270687 -3.7755078
## 5  0.2585755      1  1.1203550 -3.1176393
## 6  3.4301300      2  2.1272136 -0.2706045
```

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:

```
train_lin = sim_data_linear[1:800, ]
test_lin = sim_data_linear[801:1000, ]
```

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

```
mlr_linear_model = lm(y_lin ~ var_x1 + var_x2, data = train_lin)
summary(mlr_linear_model)

##
## Call:
## lm(formula = y_lin ~ var_x1 + var_x2, data = train_lin)
```

```
##
## 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 ***
## var_x1       1.38207    0.01767   78.21   <2e-16 ***
## var_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_mlr_lin = predict(mlr_linear_model, newdata = test_lin)
mse_mlr_lin = mean((test_lin$y_lin - pred_mlr_lin)^2, na.rm = TRUE)
mse_mlr_lin
```

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

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

```
library(caret)
```

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

```
train_feat_lin <- train_lin[, c("var_x1", "var_x2")]
test_feat_lin  <- test_lin[, c("var_x1", "var_x2")]

knn_lin_1 = knnregTrain(train_feat_lin, test_feat_lin, k = 1,
train_lin$y_lin)
mse_knn_1 = mean((test_lin$y_lin - knn_lin_1)^2, na.rm = TRUE)

knn_lin_2 = knnregTrain(train_feat_lin, test_feat_lin, k = 2,
train_lin$y_lin)
mse_knn_2 = mean((test_lin$y_lin - knn_lin_2)^2, na.rm = TRUE)

knn_lin_5 = knnregTrain(train_feat_lin, test_feat_lin, k = 5,
train_lin$y_lin)
mse_knn_5 = mean((test_lin$y_lin - knn_lin_5)^2, na.rm = TRUE)

knn_lin_9 = knnregTrain(train_feat_lin, test_feat_lin, k = 9,
train_lin$y_lin)
mse_knn_9 = mean((test_lin$y_lin - knn_lin_9)^2, na.rm = TRUE)

knn_lin_15 = knnregTrain(train_feat_lin, test_feat_lin, k = 15,
train_lin$y_lin)
```

```
mse_knn_15 = mean((test_lin$y_lin - knn_lin_15)^2, na.rm = TRUE)

mse_results_linear = data.frame(mse_knn_1, mse_knn_2, mse_knn_5, mse_knn_9,
mse_knn_15)
mse_results_linear

##   mse_knn_1 mse_knn_2 mse_knn_5 mse_knn_9 mse_knn_15
## 1  2.219793  1.729587  1.303978  1.205371    1.23273
```

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

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

```
y_non_lin = (1 / (-2 + 1.4 * var_x1 - 2.6 * var_x2 + 2.9 * (var_x1^2))) + 3.1
* sin(var_x2) - 1.5 * (var_x1 * var_x2^2) + err_term

sim_data_nonlin = data.frame(var_x1, var_x2, y_non_lin)

train_nonlin = sim_data_nonlin[1:800, ]
test_nonlin = sim_data_nonlin[801:1000, ]
```

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

Multiple Linear Regression

```
mlr_nonlin_model = lm(y_non_lin ~ var_x1 + var_x2, data = train_nonlin)
pred_mlr_nonlin = predict(mlr_nonlin_model, newdata = test_nonlin)
mse_mlr_nonlin = mean((test_nonlin$y_non_lin - pred_mlr_nonlin)^2)
mse_mlr_nonlin

## [1] 205.1776
```

KNN Regression

```
train_feat_nonlin <- train_nonlin[, c("var_x1", "var_x2")]
test_feat_nonlin  <- test_nonlin[, c("var_x1", "var_x2")]

knn_nl_1 = knnregTrain(train_feat_nonlin, test_feat_nonlin, k = 1,
train_nonlin$y_non_lin)
mse_knn_nl_1 = mean((test_nonlin$y_non_lin - knn_nl_1)^2, na.rm = TRUE)

knn_nl_2 = knnregTrain(train_feat_nonlin, test_feat_nonlin, k = 2,
train_nonlin$y_non_lin)
mse_knn_nl_2 = mean((test_nonlin$y_non_lin - knn_nl_2)^2, na.rm = TRUE)

knn_nl_5 = knnregTrain(train_feat_nonlin, test_feat_nonlin, k = 5,
train_nonlin$y_non_lin)
mse_knn_nl_5 = mean((test_nonlin$y_non_lin - knn_nl_5)^2, na.rm = TRUE)

knn_nl_9 = knnregTrain(train_feat_nonlin, test_feat_nonlin, k = 9,
train_nonlin$y_non_lin)
mse_knn_nl_9 = mean((test_nonlin$y_non_lin - knn_nl_9)^2, na.rm = TRUE)

knn_nl_15 = knnregTrain(train_feat_nonlin, test_feat_nonlin, k = 15,
train_nonlin$y_non_lin)
mse_knn_nl_15 = mean((test_nonlin$y_non_lin - knn_nl_15)^2, na.rm = TRUE)

mse_results_nonlin = data.frame(mse_knn_nl_1, mse_knn_nl_2, mse_knn_nl_5,
mse_knn_nl_9, mse_knn_nl_15)
mse_results_nonlin

##   mse_knn_nl_1 mse_knn_nl_2 mse_knn_nl_5 mse_knn_nl_9 mse_knn_nl_15
## 1      47.5249     54.48942     59.77963     62.10913      63.72303
```