MDTS4214_728_5

Oindrila Chakraborty

2026-02-20

PREDICTIVE ANALYTICS_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) 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: 1. Fit a multiple linear regression equation of y on x1 and x2. 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 : yi = 1/(−2 + 1.4x1i − 2.6x2i + 2.9x21i) + 3.1 sin(x2i) − 1.5x1ix22i + εi. Work out the problems in (1) and (2). Compare and comment.

```
set.seed(123)
x1i=rnorm(1000,0,2)
x2i=rpois(1000,1.5)
ei=rnorm(1000,0,1)
yi= -2+1.4*x1i-2.6*x2i+ei
df=data.frame(yi,x1i,x2i,ei)
dim(df)
```

```
## [1] 1000    4
```

```
train=df[c(1:800),]
test=df[c(801:1000),]
head(train)
```

```
##           yi        x1i x2i         ei
## 1 -4.3903185 -1.1209513   0 -0.8209867
## 2 -2.9517542 -0.4603550   0 -0.3072572
## 3  1.4622853  3.1174166   0 -0.9020980
## 4 -3.7755078  0.1410168   1  0.6270687
## 5 -3.1176393  0.2585755   1  1.1203550
## 6 -0.2706045  3.4301300   2  2.1272136
```

```
head(test)
```

```
##             yi        x1i x2i          ei
## 801 -0.7945918  0.7125667   0  0.20781483
## 802 -6.6277609 -1.3160204   1 -0.18533229
```

```
## 803 -2.1739932  1.7104044   1  0.03144067
## 804  1.6395734  2.3058725   0  0.41135193
## 805 -7.2026151  0.5525491   2 -0.77618389
## 806 -5.6568293  0.2882093   2  1.13967766
```

```
#MLR model
model=lm(yi~x1i+x2i,data=train)
pred=predict(model,newdata=test)
test_mse=mean((test$yi-pred)^2)
test_mse
```

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

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

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: ggplot2
```

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

```
#KNN Regression
knn1=knnreg(yi~x1i+x2i,data=train,k=1)
knn1
```

```
## 1-nearest neighbor regression model
```

```
Y_knn_1=predict(knn1,test)
head(Y_knn_1)
```

```
## [1] -1.072575 -5.132580 -2.504298  2.772551 -7.489844 -8.128553
```

```
mse_knn_1=mean((test$yi-Y_knn_1)^2)
mse_knn_1
```

```
## [1] 2.219793
```

```
knn2=knnreg(yi~x1i+x2i,data=train,k=2)
knn2
```

```
## 2-nearest neighbor regression model
```

```
Y_knn_2=predict(knn2,test)
head(Y_knn_2)
```

```
## [1] -1.347804 -5.771267 -1.951418  2.058042 -7.001174 -8.016417
```

```
mse_knn_2=mean((test$yi-Y_knn_2)^2)
mse_knn_2
```

```
## [1] 1.729587
```

```r
knn3=knnreg(yi~x1i+x2i,data=train,k=5)
knn3

## 5-nearest neighbor regression model

Y_knn_3=predict(knn3,test)
head(Y_knn_3)

## [1] -1.329243 -5.839882 -2.036713  1.268136 -6.419926 -7.099145

mse_knn_3=mean((test$yi-Y_knn_3)^2)
mse_knn_3

## [1] 1.303978

knn4=knnreg(yi~x1i+x2i,data=train,k=9)
knn4

## 9-nearest neighbor regression model

Y_knn_4=predict(knn4,test)
head(Y_knn_4)

## [1] -1.5236806 -6.4672730 -2.3198437  0.7931524 -6.2226527 -6.8144305

mse_knn_4=mean((test$yi-Y_knn_4)^2)
mse_knn_4

## [1] 1.205371

knn5=knnreg(yi~x1i+x2i,data=train,k=15)
knn5

## 15-nearest neighbor regression model

Y_knn_5=predict(knn5,test)
head(Y_knn_5)

## [1] -1.2897422 -6.3552745 -2.2964106  0.7384823 -6.1288842 -7.0430497

mse_knn_5=mean((test$yi-Y_knn_5)^2)
mse_knn_5

## [1] 1.23273
```

Comments

1. The Linear Scenario Winner: Linear Regression (OLS). Reasoning: Since the true model is $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$, OLS is the Best Linear Unbiased Estimator(BLUE). KNN will struggle because it is “local.” It uses nearby points to estimate the value, which introduces noise compared to the global approach of OLS that perfectly captures the underlying slope.For MLR the MSE is 0.9989 but for KNN Regression all MSE are above 1.So lm model is more suitable for this model.

```r
#Non-Linear response
library(caret) # Required for knnreg

set.seed(123)
x1i=rnorm(1000, 0, 2)
x2i=rpois(1000, 1.5)
ei=rnorm(1000, 0, 1)
y_nonlin=1/(-2 + 1.4*x1i - 2.6*x2i + 2.9*x1i^2) + 3.1*sin(x2i) -
1.5*x1i*x2i^2 + ei
data_nonlin=data.frame(yi2 = y_nonlin, x1i = x1i, x2i = x2i)
train_nl=data_nonlin[1:800, ]
test_nl=data_nonlin[801:1000, ]
lm_mod_nl=lm(yi2 ~ x1i + x2i, data = train_nl)
mse_lm_nl=mean((test_nl$yi2 - predict(lm_mod_nl, test_nl))^2)
ks=c(1, 2, 5, 9, 15)
knn_mses=numeric(length(ks))

for(i in 1:length(ks)) {
  knn_fit <- knnreg(yi2 ~ x1i + x2i, data = train_nl, k = ks[i])
  predictions <- predict(knn_fit, test_nl)
  knn_mses[i] <- mean((test_nl$yi2 - predictions)^2)
}

results <- data.frame(K = ks, MSE = knn_mses)
print(paste("Linear Regression MSE:", round(mse_lm_nl, 4)))

## [1] "Linear Regression MSE: 205.1776"

print(results)

##    K      MSE
## 1  1 47.52490
## 2  2 54.48942
## 3  5 59.77963
## 4  9 62.10913
## 5 15 63.72303
```

The Non-Linear Scenario

Winner: KNN (likely at a moderate k like 5 or 9). Reasoning: The complex function involving $sin(x)$ and x^2 violates the assumptions of linear regression. OLS has high bias here because it tries to fit a straight plane to a curved surface. KNN is flexible; it doesn't care about the shape of the function, only about the proximity of data points, allowing it to "bend" to the non-linear patterns.

The Role of kLow k (e.g., $k = 1$): Low bias but extremely high variance. The model captures the noise (ϵ) in the training set, leading to poor test MSE (Overfitting).High k (e.g., $k = 15$): Low variance but higher bias. The model"oversmooths" the data, potentially missing local trends.

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.
(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.

```
install.packages("ggplot2")
```

```
## Warning: package 'ggplot2' is in use and will not be installed
```

```
library(ggplot2)
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
```

```
data(diamonds)
head(diamonds)
```

```
## # A tibble: 6 × 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

```
dim(diamonds)
```

```
## [1] 53940    10
```

```r
str(diamonds)
```

```
## tibble [53,940 × 10] (S3: tbl_df/tbl/data.frame)
##  $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22
0.23 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3
...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5
...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4
5 ...
##  $ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1
59.4 ...
##  $ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4
...
##  $ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78
4.05 ...
##  $ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49
2.39 ...
```

```r
variable_info=data.frame(
  Variable = names(diamonds),
  Type = c("Quantitative", "Qualitative", "Qualitative",
"Qualitative","Quantitative","Quantitative","Quantitative","Quantitative","Qu
antitative","Quantitative"),
  Scale = c("Continuous", "Ordinal(non-numeric)","Ordinal(non-
numeric)","Ordinal(non-numeric)",
"Continuous","Continuous","Continuous","Continuous","Continuous","Continuous"
))
print(variable_info)
```

```
##    Variable         Type                Scale
## 1     carat Quantitative           Continuous
## 2       cut  Qualitative Ordinal(non-numeric)
## 3     color  Qualitative Ordinal(non-numeric)
## 4   clarity  Qualitative Ordinal(non-numeric)
## 5     depth Quantitative           Continuous
## 6     table Quantitative           Continuous
## 7     price Quantitative           Continuous
## 8         x Quantitative           Continuous
## 9         y Quantitative           Continuous
## 10        z Quantitative           Continuous
```

```r
#a)
diamonds$cut=factor(diamonds$cut,ordered=FALSE)
diamonds$cut=relevel(diamonds$cut,ref="Ideal")
fit111=lm(price~cut,data=diamonds)
summary(fit111)
```

```
##
## 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)   3457.54      27.00 128.051  < 2e-16 ***
## cutFair        901.22     102.41   8.800  < 2e-16 ***
## cutGood        471.32      62.70   7.517  5.7e-14 ***
## cutVery Good   524.22      45.05  11.636  < 2e-16 ***
## cutPremium    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
```

```
coef111=coef(fit111)
coef111
```

```
##  (Intercept)      cutFair      cutGood cutVery Good   cutPremium
##    3457.5420     901.2158     471.3225     524.2179    1126.7157
```

```
#contrasts(diamonds$cut)=contr.sdif(5)
ord_mod=lm(price~cut,data=diamonds)
summary(ord_mod)
```

```
##
## 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)   3457.54      27.00 128.051  < 2e-16 ***
## cutFair        901.22     102.41   8.800  < 2e-16 ***
## cutGood        471.32      62.70   7.517  5.7e-14 ***
## cutVery Good   524.22      45.05  11.636  < 2e-16 ***
## cutPremium    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
```

```
#aggregate(ord_mod)
aggregate(price ~ cut, data = diamonds, mean)
```

```
##         cut    price
## 1     Ideal 3457.542
## 2      Fair 4358.758
## 3      Good 3928.864
## 4 Very Good 3981.760
## 5   Premium 4584.258
```

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

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

```
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))
   }
}

coded_pred=t(sapply(lvl,code_fun))
#str(coded_pred)
colnames(coded_pred)=c("Premium","Very_Good","Good","Fair")
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)  4584.26      33.75 135.816  < 2e-16 ***
```

```
## Premium      -602.50      49.39 -12.198  < 2e-16 ***
## Very_Good     -52.90      67.10  -0.788  0.43055
## Good          429.89     113.85   3.776  0.00016 ***
## Fair         -901.22     102.41  -8.800  < 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
```

```
#model:base=Ideal(0,0,0,0)"estimate:3457.54
#Premium(1,0,0,0) estimate:3457.54+1126.72=4584.26
aggregate(diamonds$price,list(diamonds$cut),mean)
```

```
##     Group.1        x
## 1     Ideal 3457.542
## 2      Fair 4358.758
## 3      Good 3928.864
## 4 Very Good 3981.760
## 5   Premium 4584.258
```

```
#lm(price~relevel(factor(as.character(cut)),ref="Ideal"),data=diamonds)
##fit333=lm(price~clarity,data=diamonds)
#summary(fit333)

#fit222=lm(price~color,data=diamonds)
#summary(fit222)

#stargazer(fit111,fit222,fit333,type="text",out="f21.txt")

#b) Test whether the expected price of diamond with premium cut is
significantly different from that of the ideal cut.
#install.packages("car")
#library(car)
#linearHypothesis(fit111, "cutPremium = cutIdeal")
```

Fitted Regression Model (a): The model is expressed using dummy variables where "Ideal" is the reference point (intercept):

$$\widehat{\text{Price}} = \hat{\beta}_0 + \hat{\beta}_1 \text{cutPremium} + \hat{\beta}_2 \text{cutVery Good} + \hat{\beta}_3 \text{cutGood} + \hat{\beta}_4 \text{cutFair}$$

$\hat{\beta}_0$ is the intercept (Price for Ideal).$\hat{\beta}_{1\ldots4}$ represent the difference in price between that cut and the Ideal cut.

(b) Test: Premium Cut vs. Ideal CutLook at the summary(fit111) output for the row cutPremium.The Null Hypothesis (H_0): There is no difference in price between Premium and Ideal ($\beta_1 = 0$). Result: Check the p-value ($Pr(>|t|)$). If $p < 0.05$, the price of a Premium cut is significantly different from an Ideal cut. Interestingly, in this dataset,

Premium diamonds are often more expensive on average because they tend to be larger (carat), even though the cut quality is lower than Ideal.

(c) Expected Price of Ideal CutThe expected price for the reference category is simply the Intercept value from fit111. Based on the data, this is approximately $3,457.50.

(d) Incorporating the "Table" PredictorNow we add a quantitative variable (table) to the categorical model.

```
fit6=lm(price ~ cut + table, data = diamonds)
summary(fit6)

##
## 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)  -6563.672    517.450 -12.685  < 2e-16 ***
## cutFair        345.611    106.002   3.260 0.001113 **
## cutGood        -19.957     67.426  -0.296 0.767246
## cutVery Good   165.206     48.562   3.402 0.000669 ***
## cutPremium     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
```

Fitted Regression Model (d):

$$\widehat{\text{Price}} = \hat{\beta}_0 + \hat{\beta}_1 \text{cutPrem} + \hat{\beta}_2 \text{cutV.Good} + \hat{\beta}_3 \text{cutGood} + \hat{\beta}_4 \text{cutFair} + \hat{\beta}_5 \text{table}$$

(e) Significance of "Table"In summary(fit6), check the row for table.If the p-value is less than 0.05, the table width is a significant predictor of price when holding the cut quality constant.

```
avg_table <- mean(diamonds$table, na.rm = TRUE)
new_data <- data.frame(cut = "Fair", table = avg_table)
predicted_price <- predict(fit6, newdata = new_data)
cat("The average estimated price for a Fair cut diamond with a table of",
    round(avg_table, 2), "is: $", round(predicted_price, 2))

## The average estimated price for a Fair cut diamond with a table of 57.46
is: $ 4072.8
```

Explanations of the LogicCategorical Encoding:

When R sees a factor like cut, it creates $k - 1$ "dummy" columns. If a diamond is "Fair", the Fair column is 1 and all others are 0.Intercept Interpretation: The Intercept is the expected value when all dummy variables are 0 (the reference group) and the quantitative variable (table) is 0.Additive Effect: By adding table to the model, we are essentially creating a model with several parallel lines (or planes)—one for each cut—where the slope of table is assumed to be the same for all cuts.

The analysis of the diamonds dataset demonstrates how qualitative (ordinal) variables function as "intercept shifters" in a regression model. By converting the cut quality into numerical dummy variables, we move beyond simple correlations to a model that quantifies the specific "premium" or "discount" associated with each level of craftsmanship.

Findings:

The Baseline Effect: Using "Ideal" as the reference category, we found that the Intercept (3457.54) represents the expected price for that specific group. Every other coefficient tells us how much the price deviates from that "Ideal" baseline.

The Power of Controls:

By incorporating the "table" variable (a quantitative predictor), the model becomes more robust. It allows us to isolate the effect of a diamond's physical dimensions from its categorical quality. We concluded that the table percentage is a significant predictor, meaning that even within the same cut category, the specific proportions of the stone significantly impact its market value.

Predictive Versatility:

This regression framework allows for precise estimation. For instance, we can calculate that a Fair cut diamond, even when adjusted for an average table value, carries a distinct price point that differs significantly from its higher-graded counterparts.Essentially, this problem illustrates that in predictive analytics, the "best" model isn't just about the numbers—it's about how effectively you encode human-defined categories (like quality and beauty) into a mathematical structure that R can interpret.