7003 ml

2024-10-21

LIU SHENG

3036424496

Q1

When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

(a) Suppose that we have a set of observations, each with measurements on p = 1 feature, X. We assume that X is uniformly (evenly) distributed on [0, 1]. Associated with each observation is a response value.Suppose that we wish to predict a test observation's response using only observations that are within 10 % of the range of X closest to that test observation. For instance, to predict the response for a test observation with X = 0.6, we will use observations in the range [0.55, 0.65].On average, what fraction of the available observations will we use to make the prediction?

Answer:

- **We will use 10% of the observations on average to make the prediction.**
- **fraction=0.1**

(b) Now suppose that we have a set of observations, each with measurements on p = 2 features, 1and 2. We assume that (1, 2) are uniformly distributed on [0, 1] × [0, 1]. We wish to predict a test observation's response using only observations that are within 10 % of the range of 1 and within 10 % of the range of 2 closest to that test observation. For instance, to predict the response for a test observation with 1= 0.6 and 2 = 0.35, we will use observations in the range [0.55, 0.65] for 1 and in the range [0.3, 0.4] for 2. On average, what fraction of the available observations will we use to make the prediction?

Answer:

- **fraction=**0.1^2

(c) Now suppose that we have a set of observations on p = 100 features. Again, the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10 % of each feature's range that is closest to that test observation.What fraction of the available observations will we use to make the prediction?

Answer:

- **fraction=**0.1^{100}

Note: A hypercube is a generalization of a cube to an arbitrary number of dimensions. When p = 1, a hypercube is simply a line segment, when p = 2 it is a square, and when p = 100 it is a 100-dimensional cube.

Q2

Suppose we collect data for a group of students in a statistics class with variables $X1 = hoursstudied, X2 = undergradGPA, andY = receiveanA$ We fit a logistic regression and produce estimated coefficient, $\hat{\beta}_0 = -6, \quad \hat{\beta}_1 = 0.05, \quad \hat{\beta}_2 = 1$

(a) Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.

Answer:

- **the probability that a student who get A is about 38%**
- as we know the logistic regression coefficients

$$\hat{\beta}_0 = -6, \quad \hat{\beta}_1 = 0.05, \quad \hat{\beta}_2 = 1$$

- and the expression is

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2)}}$$

- so:

$$-(\beta_0 + \beta_1 X_1 + \beta_2 X_2) = -(-6 + 40 * 0.05 + 3.5 * 1) = -(-0.5)$$

$$P(Y = 1|X) = \frac{1}{1 + e^{-(-0.5)}} \approx 0.38$$

(b) How many hours would the student in part (a) need to study to have a 50 % chance of getting an A in the class?

Answer:

- **the student in part (a) need to study 50 hours**
- as we know p=0.5 and $\hat{\beta}_0 = -6, \quad \hat{\beta}_1 = 0.05, \quad \hat{\beta}_2 = 1$
- so:

$$0.5 = \frac{1}{1 + e^{-(-6 + 0.05 * X_1 + 1 * 3.5)}}$$

- $X_1 = 50$

Q3

In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the*Auto* data set

(a)

Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median.

- You can compute the median using the median () function.
- Hint: Note you may find it helpful to use the data. frame () function to create a single data set containing both mpg01 and the other Auto variables.

code

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
Auto<- read.csv("https://raw.githubusercontent.com/kwan-MSDA/MSDA7003/refs/heads/main/Auto.csv")
#str(Auto)
Auto$horsepower <- as.numeric(as.character(Auto$horsepower))
```

```
## Warning: NAs introduced by coercion
```

```r
#Auto$name <- as.numeric(as.character(Auto$name))
str(Auto)
```

```
## 'data.frame':    397 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : int  3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : int  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : chr  "chevrolet chevelle malibu" "buick skylark 320" "plymouth satellite" "amc rebel
```

```r
Auto <- na.omit(Auto)
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
```

```
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : int  3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : int  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : chr  "chevrolet chevelle malibu" "buick skylark 320" "plymouth satellite" "amc rebel
##  - attr(*, "na.action")= 'omit' Named int [1:5] 33 127 331 337 355
##   ..- attr(*, "names")= chr [1:5] "33" "127" "331" "337" ...
```

```r
mpgmedian<-median(Auto$mpg)
mpgmedian
```

```
## [1] 22.75
```

```r
Auto$mpg01 <- ifelse(Auto$mpg > mpgmedian, 1, 0)
table(Auto$mpg01)
```

```
##
##   0   1
## 196 196
```

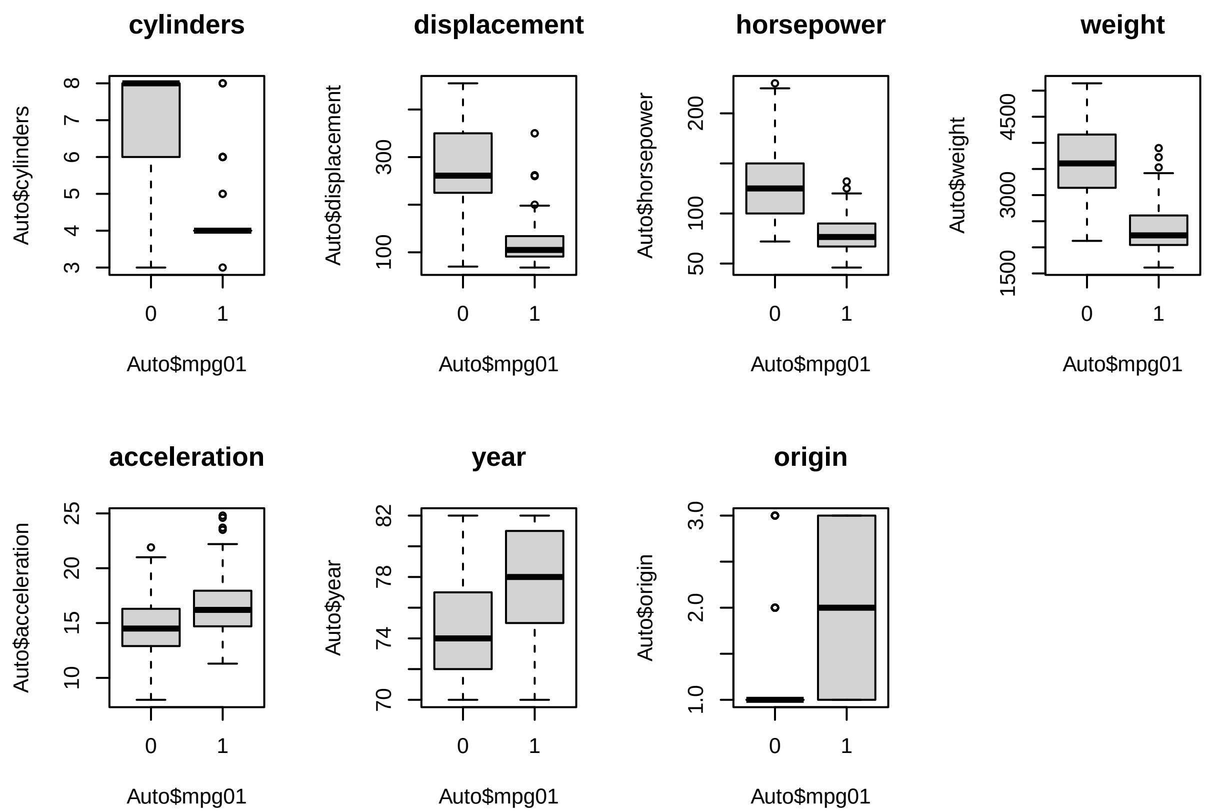
(b)

Explore the data graphically to investigate the association between mpg01 and the other features.

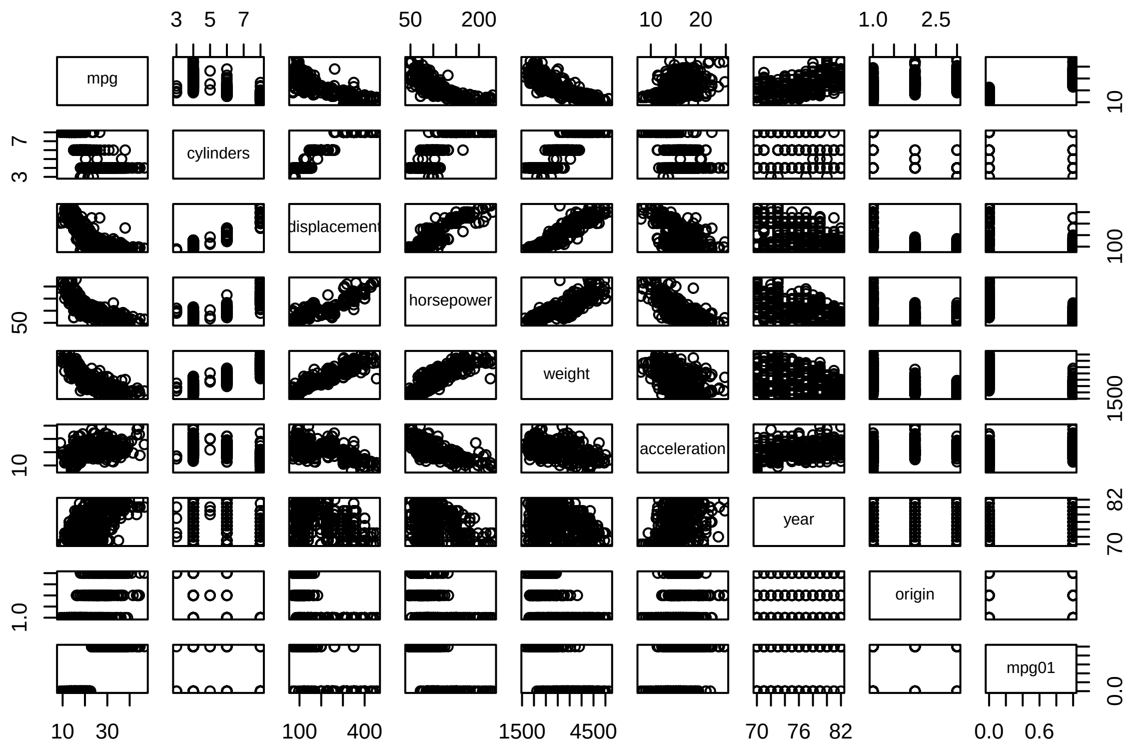
- Which of the other features seem most likely to be useful in predicting mpg01?
- Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

code

```r
par(mfrow=c(2,4))
#1.cylinders
boxplot(Auto$cylinders~Auto$mpg01,main='cylinders')
#2.displacement
boxplot(Auto$displacement~Auto$mpg01,main='displacement')
#3.horsepower
boxplot(Auto$horsepower~Auto$mpg01,main='horsepower')
#4.weight
boxplot(Auto$weight~Auto$mpg01,main='weight')
#5.acceleration
boxplot(Auto$acceleration~Auto$mpg01,main='acceleration')
#6.year
boxplot(Auto$year~Auto$mpg01,main='year')
#7.origin
boxplot(Auto$origin~Auto$mpg01,main='origin')
```




```r
pairs(Auto[-9])
```



Describe: ***The features cylinders, displacement, horsepower, weight, and year are likely to be the most useful in predicting mpg01.***

(c)

Split the data into a training set and a test set.

code

```
# Set seed for reproducibility
set.seed(233)   #

# Determine the index for the training set (80% of the rows)
train_index <- sample(seq_len(nrow(Auto)), size = 0.8 * nrow(Auto))

# Split the data into training and test sets
train_data <- Auto[train_index, ]
test_data <- Auto[-train_index, ]

nrow(train_data)/nrow(Auto)
```

```
## [1] 0.7984694
```

```r
nrow(test_data)/nrow(Auto)
```

```
## [1] 0.2015306
```

(d)

Perform logistic regression on the training data to predict mpg01 using the variables that seemed most associated with mpg01 in (b).

- What is the test error of the model obtained?

code

```r
cor(Auto[-9])
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
## mpg01         0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
##              acceleration       year     origin      mpg01
## mpg             0.4233285  0.5805410  0.5652088  0.8369392
## cylinders      -0.5046834 -0.3456474 -0.5689316 -0.7591939
## displacement   -0.5438005 -0.3698552 -0.6145351 -0.7534766
## horsepower     -0.6891955 -0.4163615 -0.4551715 -0.6670526
## weight         -0.4168392 -0.3091199 -0.5850054 -0.7577566
## acceleration    1.0000000  0.2903161  0.2127458  0.3468215
## year            0.2903161  1.0000000  0.1815277  0.4299042
## origin          0.2127458  0.1815277  1.0000000  0.5136984
## mpg01           0.3468215  0.4299042  0.5136984  1.0000000
```

```r
glm_fit <- glm(
    mpg01 ~ cylinders+displacement + horsepower + weight + year,
    data = train_data, family = binomial
  )

glm_summary<- summary(glm_fit)

glm_summary
```

```
##
## Call:
## glm(formula = mpg01 ~ cylinders + displacement + horsepower +
##     weight + year, family = binomial, data = train_data)
##
```

```
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -17.869449   5.679882  -3.146 0.001655 **
## cylinders     -0.183221   0.474041  -0.387 0.699120
## displacement  -0.007015   0.011526  -0.609 0.542785
## horsepower    -0.042258   0.018635  -2.268 0.023353 *
## weight        -0.004087   0.001055  -3.873 0.000107 ***
## year           0.464487   0.089044   5.216 1.82e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 433.75  on 312  degrees of freedom
## Residual deviance: 113.04  on 307  degrees of freedom
## AIC: 125.04
##
## Number of Fisher Scoring iterations: 8
```

```r
# Predict on the test set
logistic_probs <- predict(glm_fit, test_data,type = "response")
#logistic_probs[1:10]
logistic_pred <- ifelse(logistic_probs > 0.5, 1, 0)

# Calculate the test error
test_error <- mean(logistic_pred != test_data$mpg01)

# Print the test error
test_error
```

```
## [1] 0.1772152
```

(e)

Using linear regression to do the same task as logistic regression in (d)

code

```r
# Fit a linear regression model to predict mpg01 using selected variables
linear_model <- lm(mpg01 ~ cylinders + displacement + horsepower + weight + year,
                   data = train_data)

# Summarize the linear regression model
summary(linear_model)
```

```
##
## Call:
## lm(formula = mpg01 ~ cylinders + displacement + horsepower +
##     weight + year, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.91939 -0.16281  0.07726  0.18855  0.94329
```

```
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.738e-01  4.142e-01  -1.385 0.166997
## cylinders    -1.211e-01  3.219e-02  -3.762 0.000202 ***
## displacement -2.287e-04  7.049e-04  -0.324 0.745865
## horsepower    2.463e-03  1.038e-03   2.374 0.018227 *
## weight       -2.591e-04  5.466e-05  -4.741 3.26e-06 ***
## year          3.015e-02  5.089e-03   5.925 8.37e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2965 on 307 degrees of freedom
## Multiple R-squared:  0.6549, Adjusted R-squared:  0.6492
## F-statistic: 116.5 on 5 and 307 DF,  p-value: < 2.2e-16
```

```r
# Predict on the test data using the linear model
linear_pred <- predict(linear_model, test_data)

# Convert continuous predictions to binary class (0 or 1)
linear_pred_class <- ifelse(linear_pred > 0.5, 1, 0)

# Calculate the test error for the linear regression model
test_error_linear <- mean(linear_pred_class != test_data$mpg01)

# Print the test error
test_error_linear
```

```
## [1] 0.1139241
```

(f)

Using naive Bayes to do the same task as logistic regression in (d)

code

```r
#install.packages('e1071')
library(e1071)
# Train the Naive Bayes model
naive_bayes_model <- naiveBayes(mpg01 ~ cylinders + displacement + horsepower + weight + year,  data = t
naive_bayes_model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.5111821 0.4888179
##
```

```
## Conditional probabilities:
##    cylinders
## Y       [,1]     [,2]
##   0 6.756250 1.381581
##   1 4.163399 0.643362
## 
##    displacement
## Y       [,1]     [,2]
##   0 272.6875 87.79917
##   1 115.4608 36.30099
## 
##    horsepower
## Y        [,1]     [,2]
##   0 130.18750 36.88793
##   1  78.81699 15.75697
## 
##    weight
## Y       [,1]     [,2]
##   0 3627.675 678.7644
##   1 2339.059 395.4262
## 
##    year
## Y       [,1]     [,2]
##   0 74.36875 2.977107
##   1 77.60131 3.578701
```

```r
naive_bayes_pred <- predict(naive_bayes_model, test_data)
test_error_nb <- mean(naive_bayes_pred != test_data$mpg01)
test_error_nb
```

```
## [1] 0.1265823
```

(g)

Using K-NN to do the same task as logistic regression in (d). Does the choice of K matter here?

code

k=5

```r
#install.packages("class")
# Load the necessary package
library(class)

train_data_scaled <- scale(train_data[, c("cylinders", "displacement", "horsepower", "weight", "year")]
test_data_scaled <- scale(test_data[, c("cylinders", "displacement", "horsepower", "weight", "year")])

knn_pred <- knn(train = train_data_scaled, test = test_data_scaled, cl = train_data$mpg01, k = 5)

test_error_knn <- mean(knn_pred != test_data$mpg01)

test_error_knn
```

```
## [1] 0.07594937
```

k=3

```
knn_pred <- knn(train = train_data_scaled, test = test_data_scaled, cl = train_data$mpg01, k = 3)

test_error_knn <- mean(knn_pred != test_data$mpg01)

test_error_knn
```

```
## [1] 0.08860759
```

the choice of K is important