ML hw3

Runze Wang

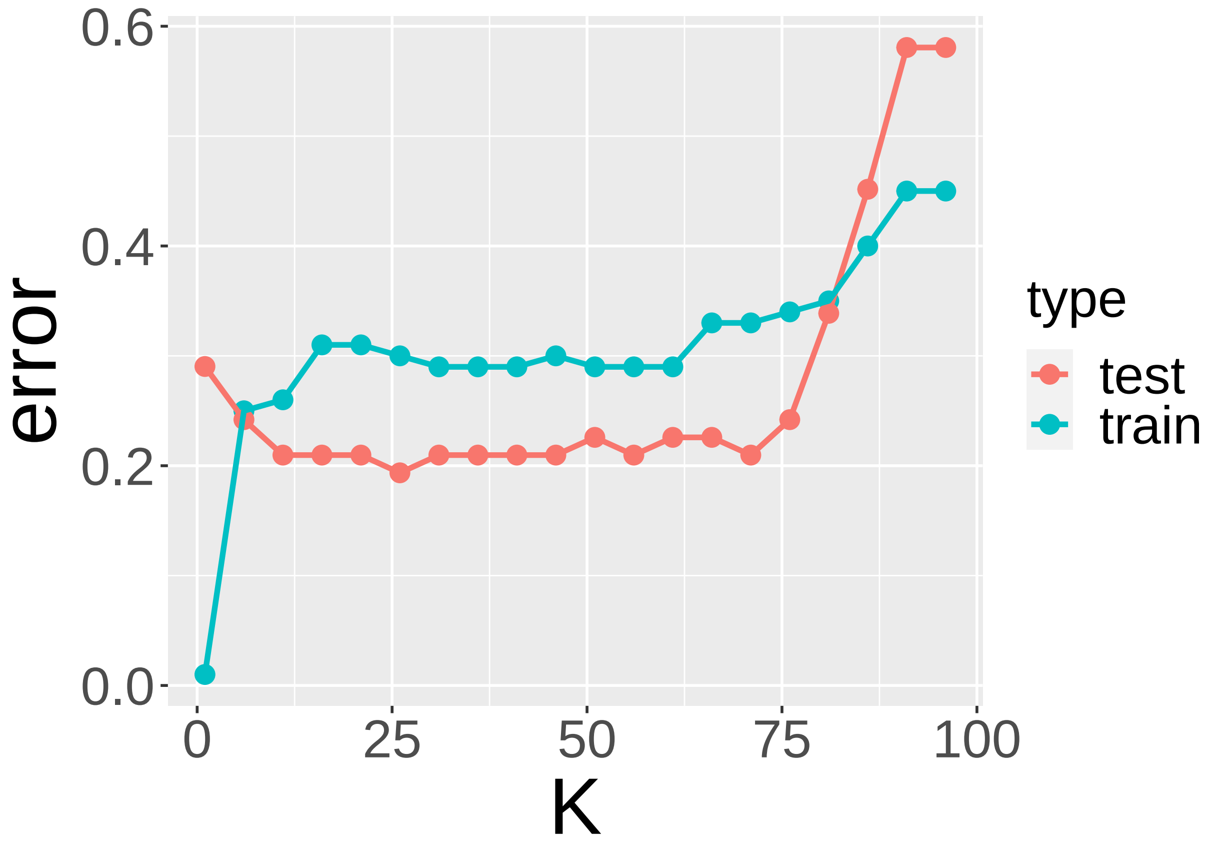
2024-02-22

```r
library(r02pro)     #INSTALL IF NECESSARY
library(tidyverse)  #INSTALL IF NECESSARY
library(MASS)
library(pROC)
library(caret)
my_sahp <- sahp %>%
  na.omit() %>%
  mutate(expensive = sale_price > median(sale_price)) %>%
  dplyr::select(gar_car, liv_area, oa_qual, expensive)
my_sahp$expensive <- as.factor(my_sahp$expensive)
my_sahp_train <- my_sahp[1:100, ]
my_sahp_test <- my_sahp[-(1:100), ]
```

Q1 1. Use the training data `my_sahp_train` to fit a KNN model of `expensive` on variables `gar_car` and `liv_area`. a. Vary the nearest number K from 1 to 100 with increment 5. For each K, fit the KNN classification model on the training data, and predict on the test data. Visualize the training and test error trend as a function of K. Discuss your findings.

```r
k_seq <- seq(from = 1, to = 100, by = 5)
train_error_seq <- test_error_seq <- NULL
for(k_ind in seq_along(k_seq)){
 k <- k_seq[k_ind]
 fit_knn <- knn3(expensive ~ gar_car + liv_area , data = my_sahp_train, k = k)
 pred_knn <- predict(fit_knn, newdata = my_sahp_train, type = "class")
 train_error_seq[k_ind] <- mean(pred_knn != my_sahp_train$expensive)
 pred_knn <- predict(fit_knn, newdata = my_sahp_test, type = "class")
 test_error_seq[k_ind] <- mean(pred_knn != my_sahp_test$expensive)
}
knn_re <- rbind(data.frame(K = k_seq, error = train_error_seq, type = "train"),
                data.frame(K = k_seq, error = test_error_seq, type = "test"))
mytheme <- theme(axis.title = element_text(size = 30),
        axis.text = element_text(size = 20),
        legend.text = element_text(size = 20),
        legend.title = element_text(size = 20))
ggplot(knn_re, mapping = aes(x = K, y = error, color = type)) +
  geom_point(size = 3) +
  geom_line(size = 1) +
  mytheme
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

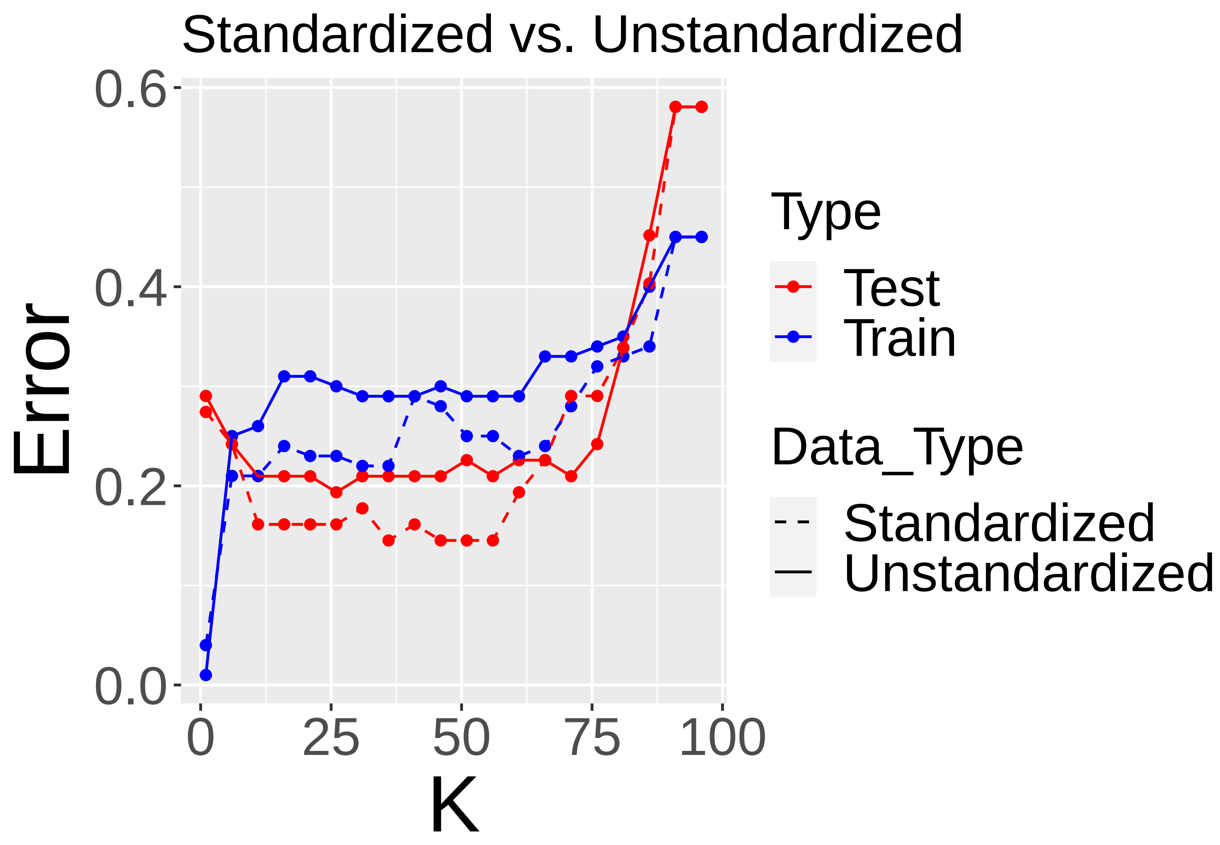


When k=1, the training error is the lowest. The training and test error increases, as the k increases, then becomes stable around between k=25 and k=75. The smallest test error is when k=25. The model is overfitting when k is greater than 75.

Q1 b. First, standardize `gar_car` and `liv_area`. Then, repeat the task in a, and visualize the training and test error together with the unstandarized version as a function of K.

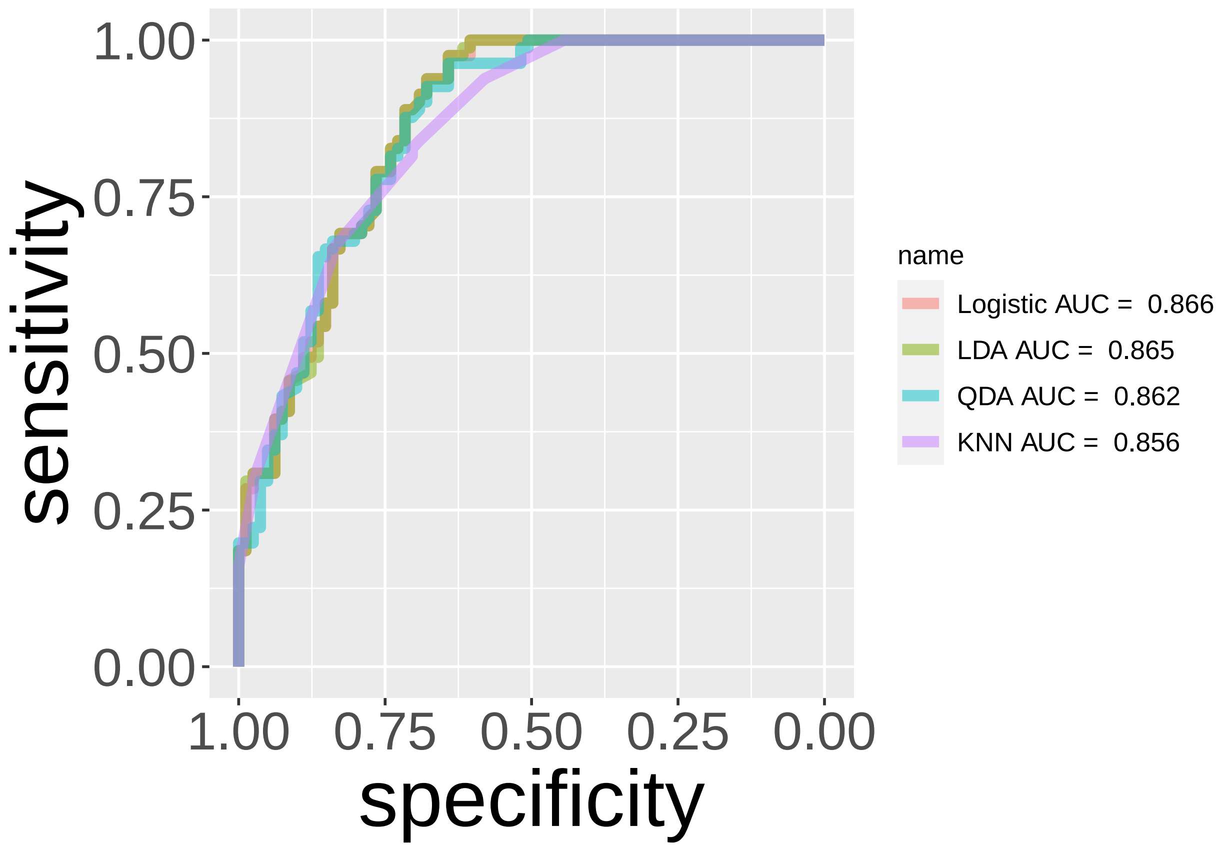
```r
my_sahp_train$gar_car_sd <- scale(my_sahp_train$gar_car)
my_sahp_train$liv_area_sd <- scale(my_sahp_train$liv_area)

my_sahp_test$gar_car_sd <- scale(my_sahp_test$gar_car)
my_sahp_test$liv_area_sd <- scale(my_sahp_test$liv_area)
```

```r
k_seq <- seq(from = 1, to = 100, by = 5)
train_error_seq_sd<- test_error_seq_sd<- NULL
for(k_ind in seq_along(k_seq)){
 k <- k_seq[k_ind]
 fit_knn_sd <- knn3(expensive ~ gar_car_sd + liv_area_sd , data = my_sahp_train, k = k)
 pred_knn_sd<- predict(fit_knn_sd, newdata = my_sahp_train, type = "class")
 train_error_seq_sd[k_ind] <- mean(pred_knn_sd != my_sahp_train$expensive)
 pred_knn_sd<- predict(fit_knn_sd, newdata = my_sahp_test, type = "class")
 test_error_seq_sd[k_ind] <- mean(pred_knn_sd != my_sahp_test$expensive)
}

knn_re_sd <- rbind(data.frame(K = k_seq, error = train_error_seq_sd, type = "train"),
                data.frame(K = k_seq, error = test_error_seq_sd, type = "test"))

knn_re_combined <- rbind(
  data.frame(K = k_seq, Error = train_error_seq, Type = "Train", Data_Type = "Unstandardized"),
  data.frame(K = k_seq, Error = test_error_seq, Type = "Test", Data_Type = "Unstandardized"),
  data.frame(K = k_seq, Error = train_error_seq_sd, Type = "Train", Data_Type = "Standardized"),
  data.frame(K = k_seq, Error = test_error_seq_sd, Type = "Test", Data_Type = "Standardized"))

ggplot(knn_re_combined, aes(x = K, y = Error, color = Type, linetype = Data_Type)) +
  geom_point() +
  geom_line() +
  labs(title = "Standardized vs. Unstandardized ",
       x = "K", y = "Error") +
  scale_color_manual(values = c("Train" = "blue", "Test" = "red")) +
  scale_linetype_manual(values = c("Unstandardized" = "solid", "Standardized" = "dashed")) +
  theme(axis.title = element_text(size = 20),
        axis.text = element_text(size = 20),
        legend.title = element_text(size = 20),
        legend.text = element_text(size = 20),
        plot.title = element_text(size = 20),
        legend.position = "right") +
  mytheme
```


Standardized vs. Unstandardized
Error
0.6
0.4
0.2
0.0
0
25
50
75
100
K
Type
Test
Train
Data_Type
Standardized
Unstandardized

Q2 Use the data `my_sahp` (without standardization) to fit four models of `expensive` on variables `gar_car` and `liv_area`, using Logistic regression, LDA, QDA, and KNN (with $K = 7$). Visualize the ROC curves for them and add the AUC values to the legend. Discuss your findings.

```r
##logistic
logistic <- glm(expensive ~ gar_car + liv_area, data = my_sahp,  family='binomial')
logistic.pred <- predict(logistic, my_sahp, type = "response")
rocobj_logistic <- roc(my_sahp$expensive, logistic.pred)
auc_logistic <- auc(rocobj_logistic)

##lda
lda.fit <- lda(expensive ~ gar_car + liv_area, data = my_sahp)
lda.pred <- predict(lda.fit)$posterior[, 2]
rocobj_lda <- roc(my_sahp$expensive, lda.pred)
auc_lda <- auc(rocobj_lda)

##qda
qda.fit <- qda(expensive ~ gar_car + liv_area, data = my_sahp)
qda.pred <- predict(qda.fit)$posterior[, 2]
rocobj_qda <- roc(my_sahp$expensive, qda.pred)
auc_qda <- auc(rocobj_qda)

##knn
knn.fit <- knn3(expensive ~ gar_car + liv_area, data = my_sahp, k = 7,prob = TRUE)
knn.pred <- predict(knn.fit, newdata = my_sahp, type = "prob")
rocobj_knn <- roc(my_sahp$expensive, knn.pred[ ,2])
auc_knn <- auc(rocobj_knn)

rocobjs <- list(Logistic = rocobj_logistic, LDA = rocobj_lda, QDA = rocobj_qda,
                KNN = rocobj_knn)
methods_auc <- paste(c("Logistic", "LDA", "QDA","KNN"),
                     "AUC = ",
                     round(c(auc_logistic, auc_lda, auc_qda, auc_knn),3))
ggroc(rocobjs, size = 2, alpha = 0.5) +
  scale_color_discrete(labels = methods_auc) +
  mytheme +
  theme(
    plot.title = element_text(size = 15),
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 10)
  )
```



The logistic model has the largest AUC, so it has a good measure of separability compared to the other three models. The ROC curve for LDA is the closest to the upper left corner, so it has the highest accuracy of the test compared to the other three models.

Q3 (ISLRv2 Chapter 4 Q4) 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 curse of dimensionality 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, in order 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?

Due to 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, so the interval length is 0.1.

$$f(x) = \begin{cases} X + 0.05 & \text{if } 0 \le X < 0.05, \\ 0.1 & \text{if } 0.05 \le X \le 0.95, \\ 1.05 - X & \text{if } 0.95 < X \le 1, \\ 0 & \text{otherwise.} \end{cases}$$

As we know, the X is uniformly (evenly) distributed on [0, 1], so we can write

$$\int_0^{0.05} x + 0.05\,dx + \int_{0.05}^{0.95} 0.1\,dx + \int_{0.95}^{1} 1.05 - x\,dx = 0.0975.$$

On average, 9.75% of the available observations will we use to make the prediction.

b Now suppose that we have a set of observations, each with measurements on p = 2 features, X1 and X2. We assume that (X1, X2) 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 X1 and within 10 % of the range of X2 closest to that test observation. For instance, in order to predict the response for a test observation with X1 = 0.6 and X2 = 0.35, we will use observations in the range [0.55, 0.65] for X1 and in the range [0.3, 0.4] for X2. On average, what fraction of the available observations will we use to make the prediction?

Assuming x1 and x2 are independent and they are uniformly distributed on [0, 1] * [0, 1]. we can get 0.0975*0.0975 = 0.0095 by multiplication rule for independent random variables. On average, 0.95 % of the available observations will we use to make the prediction.

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?

$$(0.0975)^{100} \approx 0$$

There are no available observations will we use to make the prediction.

d Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations “near” any given test observation.

$$\lim_{p\to\infty}(0.975)^p = 0$$

When the p increases the fraction of the available observations will we use to make the prediction tends to close zero, which means there are very few training observations “near” any given test observation.

e Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10 % of the training observations. For p = 1, 2, and 100, what is the length of each side of the hypercube? Comment on your answer.

From the note below this question, we can get p=1, length of each side of the hypercube is 0.1. p=2,

$$0.1^{1/2} \approx 0.32$$

length of each side of the hypercube is around 0.32. p=100

$$0.1^{1/100} \approx 0.98$$

length of each side of the hypercube is around 0.98.

when p increases, the length of each side of the hypercube increases. As we know, the volume of the hypercube is s^p. For p =100, in order to use 10% of the data to make our decision, we must cover 98% of the range of each dimension. we have to use almost the entire space to find the KNN. The points in the data set are close to the data points in the training dataset, so the kNN assumption breaks down.