STA567 Homework3

Lina Lee

9/26/2019

```
library(tidyverse)

setwd("C:\\Users\\linal\\Desktop\\Miami2019\\STA567\\Homework\\Homework3")
# Load in the sample using the load function
load(file="June2019Cabs.Rdata")

# Standardize trip distance and duration,
# and convert numeric labels on payments to "Credit"/"Cash"


my_standardize <- function(x, train_x){
  z= (x-mean(train_x))/sd(train_x)
  return(z)
}

cabs_sample <- cabs_sample %>%
  mutate(duration = as.numeric(tpep_dropoff_datetime - tpep_pickup_datetime,
units="mins"),
         payment_type = factor(payment_type, levels=c(1,2),
                               labels=c("Credit","Cash")),
         DO_service_zone= factor(DO_service_zone,
                                 levels=c("Yellow Zone","Boro Zone"),
                                 labels=c("Yellow Zone","Boro Zone")))
cabs_sample$trip_distance<-
my_standardize(cabs_sample$trip_distance,cabs_sample$trip_distance)
cabs_sample$duration<-my_standardize(cabs_sample$duration,cabs_sample$duration)
```

Problem 1

```
# use this function to add K grouping indeces
add_cv_folds <- function(dat,cv_K){
  if(nrow(dat) %% cv_K == 0){ # if perfectly divisible
    dat$fold <- sample(rep(1:cv_K, each=(nrow(dat)%/%cv_K)))
  } else { # if not perfectly divisible
    dat$fold <- sample(c(rep(1:(nrow(dat) %% cv_K), each=(nrow(dat)%/%cv_K + 1)),
                         rep((nrow(dat) %% cv_K + 1):cv_K,each=(nrow(dat)%/%cv_K)) )
)
  }
  return(dat)
}

# add 10-fold CV labels to cabs_sample data
library(caret)
```

```r
set.seed(12345)
cabs_sample_cv <- add_cv_folds(cabs_sample,10)
head(cabs_sample_cv)
str(cabs_sample_cv)
table(cabs_sample_cv$fold)

which(cabs_sample_cv$fold ==1 )

# initalize a vector for storing the predicted prices
kfold_preds1 <- rep(NA, 1000)
kfold_preds2 <- rep(NA, 1000)
kfold_preds15nn <- rep(NA, 1000)
kfold_preds30nn <- rep(NA, 1000)
for (i in 1:10){
  fold_index <- which(cabs_sample_cv$fold ==i)
  # separate data into train (all but 1 row) and test (1 row)
  train_data <- cabs_sample[-fold_index , ]
  test_data <- cabs_sample[ fold_index , ]

  # fit the model using training dat
  mod1 <- lm(fare_amount ~ trip_distance + duration , data=train_data)
  mod2 <- lm(fare_amount ~trip_distance+ duration+ trip_distance^2+ duration^2,
data=train_data)
  mod_15nn <- knnreg(fare_amount ~ trip_distance + duration, data=train_data, k = 15)
  mod_30nn <- knnreg(fare_amount ~ trip_distance + duration, data=train_data, k = 30)
  # predict the responses for the testing data
  kfold_preds1[fold_index] <- predict(mod1, newdata=test_data)
  kfold_preds2[fold_index] <- predict(mod2, newdata=test_data)
  kfold_preds15nn[fold_index] <- predict(mod_15nn, newdata=test_data)
  kfold_preds30nn[fold_index] <- predict(mod_30nn, newdata=test_data)
}
head(kfold_preds1)
head(kfold_preds2)
head(kfold_preds15nn)
head(kfold_preds30nn)
```

For each model provide the 10-fold CV MSE. Which models perform best, which perform worst. Briefly describe your results.

```r
# 10-Fold CV-MSE for price prediction mean absolute error
mean((cabs_sample$fare_amount - kfold_preds1)^2)

## [1] 15.68829

mean((cabs_sample$fare_amount - kfold_preds2)^2)

## [1] 15.68829

mean((cabs_sample$fare_amount - kfold_preds15nn)^2)

## [1] 13.70171

mean((cabs_sample$fare_amount - kfold_preds30nn)^2)

## [1] 15.27172

sqrt(mean((cabs_sample$fare_amount - kfold_preds1)^2))
```

```
## [1] 3.960844

sqrt(mean((cabs_sample$fare_amount - kfold_preds2)^2))

## [1] 3.960844

sqrt(mean((cabs_sample$fare_amount - kfold_preds15nn)^2))

## [1] 3.701582

sqrt(mean((cabs_sample$fare_amount - kfold_preds30nn)^2))

## [1] 3.907905
```

Answers: MSE of 15-nearest neighbors regression is the smallest among those of four models.The 15-nearest neighbors regression performs best.

	MSE	RMSE
Regression	15.688	3.961
Regression with polynomials	15.688	3.961
15-nearest neighbors	13.701	3.702
30-nearest neighbors	15.271	3.908

```
library(MASS)        # lda and qda functions in MASS package

library(caret)
```

Problem 2

Leave One Out Cross Validation: for each model calculate the LOOCV misclassification rate.

LDA, QDA

```
loo_preds_lda <- rep(NA, 1000)
loo_preds_qda <- rep(NA, 1000)

for (i in 1:1000){
  # separate data into train (all but 1 row) and test (1 row)
  train_data <- cabs_sample[-i, ]
  test_data <- cabs_sample[i, ]
```

```r
  # fit the model using training dat
  mod_lda <- train(DO_service_zone ~ trip_distance + duration, data=train_data,
                   method="lda")
  mod_qda <- train(DO_service_zone ~ trip_distance + duration, data=train_data,
                   method="qda")


  # predict the responses for the testing data
  loo_preds_lda[i] <-predict(mod_lda, newdata = test_data)
  loo_preds_qda[i] <-predict(mod_qda, newdata = test_data)

}

# test misclassification error rate for LDA = 0.099
mean(ifelse(loo_preds_lda==1,"Yellow Zone","Boro Zone") !=
cabs_sample$DO_service_zone)

## [1] 0.099

# test misclassification error rate for QDA = 0.123
mean(ifelse(loo_preds_qda==1,"Yellow Zone","Boro Zone") !=
cabs_sample$DO_service_zone)

## [1] 0.128
```

Answer: The test misclassification error rate for LDA is 0.099, and test misclassification error rate for QDA = 0.128

knn

```r
loo_preds_20nn <- rep(NA, 1000)

for (i in 1:1000){
  # separate data into train (all but 1 row) and test (1 row)
  train_data <- cabs_sample[-i, ]
  test_data <- cabs_sample[i, ]

  # fit the model using training dat

  mod_20nn <- knn3(DO_service_zone ~ trip_distance + duration, data=train_data,
                    k = 20)


  # predict the responses for the testing data

  loo_preds_20nn[i] <-predict(mod_20nn, test_data, type="class")

    # save as column in test data
}
# test misclassification error rate for KNN

mean(ifelse(loo_preds_20nn==1,"Yellow Zone","Boro Zone") !=
cabs_sample$DO_service_zone)
```

```
## [1] 0.111
```

Answer: The test misclassification error rate for KNN is 0.111

logit

```
cabs_sample_logit <-cabs_sample %>%
mutate(binary = ifelse(cabs_sample$DO_service_zone=="Yellow Zone", 0, 1))

loo_preds_logit <- rep(NA, 1000)

for (i in 1:1000){
  # separate data into train (all but 1 row) and test (1 row)
  train_data <- cabs_sample_logit[-i, ]
  test_data <- cabs_sample_logit[i, ]

  # fit the model using training dat
  mod_logit <- glm(binary ~ trip_distance + duration, family=binomial(link=logit),
                 data=train_data)

  # predict the responses for the testing data

  loo_preds_logit[i] <-predict(mod_logit, newdata = test_data,type="response")
}

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# test misclassification error rate for logistic = 0.106
mean(ifelse(loo_preds_logit > 0.5 , "Boro Zone", "Yellow Zone") !=
cabs_sample_logit$DO_service_zone)

## [1] 0.106
```

The test misclassification error rate for logistic = 0.106

Problem3

Using LOOCV, find the decision threshold required so that the sensitivity for correctly predicting observations dropping off in the Boro Zone is at least 0.8.

```
threshold=0.096
predicted_values<-ifelse(loo_preds_logit > threshold, 1, 0)
actual_values<-cabs_sample_logit$binary
conf_matrix<-table(predicted_values,actual_values)
conf_matrix

##                  actual_values
## predicted_values   0   1
##                0 710  39
##                1 183  68

sensitivity(conf_matrix)
```

```
## [1] 0.7950728
```

```
threshold=0.097
predicted_values<-ifelse(loo_preds_logit > threshold, 1, 0)
actual_values<-cabs_sample_logit$binary
conf_matrix<-table(predicted_values,actual_values)
conf_matrix
```

```
##                 actual_values
## predicted_values   0   1
##                0 715  39
##                1 178  68
```

```
sensitivity(conf_matrix)
```

```
## [1] 0.8006719
```

Answer: When the threshold is 0.096, the sensitivity is 0.7950. When the thresold is 0.097, the sensitivity is 0.8006. Therefore, the decision threshold required so that the sensitivity is at least 0.8 is 0.097

Probelm4

```
library(leaps)
n <- length(cabs_sample$fare_amount)
subset <- regsubsets(fare_amount ~.,
                     method="exhaustive", nbest=1, data=cabs_sample)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : nvmax reduced to 6
```

```
cbind(summary(subset)$outmat, round(summary(subset)$rsq, 3),
      round(summary(subset)$adjr2, 3), round(summary(subset)$cp, 1),
round(sqrt(summary(subset)$rss/(n-c(rep(1:7,rep(2,7)),8)-1)), 4))
```

```
## Warning in summary(subset)$rss/(n - c(rep(1:7, rep(2, 7)), 8) - 1): longer
## object length is not a multiple of shorter object length

## Warning in cbind(summary(subset)$outmat, round(summary(subset)$rsq, 3), :
## number of rows of result is not a multiple of vector length (arg 5)
```

```
##          payment_typeCash tpep_pickup_datetime tpep_dropoff_datetime
## 1  ( 1 ) " "              " "                  " "
## 2  ( 1 ) " "              " "                  " "
## 3  ( 1 ) " "              " "                  " "
## 4  ( 1 ) " "              "*"                  " "
## 5  ( 1 ) " "              " "                  "*"
## 6  ( 1 ) "*"              "*"                  "*"
##          passenger_count trip_distance DO_service_zoneBoro Zone duration
## 1  ( 1 ) " "             "*"           " "                      " "
## 2  ( 1 ) " "             "*"           " "                      "*"
## 3  ( 1 ) " "             "*"           "*"                      "*"
```

```
## 4  ( 1 ) " "              "*"           "*"                      "*"
## 5  ( 1 ) "*"              "*"           "*"                      "*"
## 6  ( 1 ) "*"              "*"           "*"                      " "
##
## 1  ( 1 ) "0.696" "0.695" "-0.9" "3.949"
## 2  ( 1 ) "0.696" "0.696" "-0.8" "3.9452"
## 3  ( 1 ) "0.696" "0.695" "0.7"  "3.9463"
## 4  ( 1 ) "0.697" "0.695" "2.3"  "3.9454"
## 5  ( 1 ) "0.697" "0.695" "4.1"  "3.947"
## 6  ( 1 ) "0.697" "0.695" "6"    "3.9469"
```

Explanation: In the subsets of regression approach, it gives us 6 best subsets. adj-R^2 are around 0.695, there are not much different. all the Cp are below p,RSS are around 3.94. All of them are not much different although model 2 has the highest adj-R^2,the lowest RSS. Therefore, we will see the performance for prediction for all the six models.

```
fit1<-lm(fare_amount~trip_distance,data=cabs_sample)
fit2<-lm(fare_amount~trip_distance+duration,data=cabs_sample)
fit3<-lm(fare_amount~trip_distance+DO_service_zone+duration,data=cabs_sample)
fit4<-
lm(fare_amount~tpep_pickup_datetime+trip_distance+DO_service_zone+duration,data=cabs_
sample)
fit5<-
lm(fare_amount~tpep_dropoff_datetime+passenger_count+trip_distance+DO_service_zone+du
ration,data=cabs_sample)
fit6<-
lm(fare_amount~payment_type+tpep_dropoff_datetime+passenger_count+trip_distance+DO_se
rvice_zone,data=cabs_sample)

AIC(fit1)

## [1] 5588.823

AIC(fit2)

## [1] 5588.874

AIC(fit3)

## [1] 5590.424

AIC(fit4)

## [1] 5591.965

AIC(fit5)

## [1] 5593.805

AIC(fit6)

## [1] 5595.618
```

Explanation: AIC of first model and second model are same as 5588.8. AIC of third model is 5590.4,and AIC of fourth model is 5591.9. The fifth and sixth models have quite different AICs from the others.

```
# use this function to add K grouping indeces
add_cv_folds <- function(dat,cv_K){
  if(nrow(dat) %% cv_K == 0){ # if perfectly divisible
    dat$fold <- sample(rep(1:cv_K, each=(nrow(dat)%/%cv_K)))
  } else { # if not perfectly divisible
    dat$fold <- sample(c(rep(1:(nrow(dat) %% cv_K), each=(nrow(dat)%/%cv_K + 1)),
                         rep((nrow(dat) %% cv_K + 1):cv_K,each=(nrow(dat)%/%cv_K)) )
)
  }
  return(dat)
}



# add 10-fold CV labels to cabs_sample data
library(caret)
set.seed(12345)
cabs_sample_cv <- add_cv_folds(cabs_sample,10)
head(cabs_sample_cv)
str(cabs_sample_cv)
table(cabs_sample_cv$fold)

which(cabs_sample_cv$fold ==1 )

# initalize a vector for storing the predicted prices
kfold_preds_pb4_1 <- rep(NA, 1000)
kfold_preds_pb4_2 <- rep(NA, 1000)
kfold_preds_pb4_final <- rep(NA, 1000)
kfold_preds_pb4_4 <- rep(NA, 1000)
kfold_preds_pb4_5 <- rep(NA, 1000)
kfold_preds_pb4_6 <- rep(NA, 1000)
for (i in 1:10){
  fold_index <- which(cabs_sample_cv$fold ==i)
  # separate data into train (all but 1 row) and test (1 row)
  train_data <- cabs_sample[-fold_index , ]
  test_data <- cabs_sample[ fold_index , ]

  # fit the model using training dat
  mod_pb4_1 <- lm(fare_amount ~ trip_distance , data=train_data)
  mod_pb4_2 <- lm(fare_amount ~ trip_distance+duration, data=train_data)
  my_final_mod <- lm(fare_amount ~ trip_distance+DO_service_zone+duration ,
data=train_data)
  mod_pb4_4 <- lm(fare_amount ~
tpep_pickup_datetime+trip_distance+DO_service_zone+duration,
                  data=train_data)
  mod_pb4_5 <-
lm(fare_amount~tpep_dropoff_datetime+passenger_count+trip_distance+DO_service_zone+du
ration,
                  data=train_data)
  mod_pb4_6 <-
```

```r
lm(fare_amount~payment_type+tpep_dropoff_datetime+passenger_count+trip_distance+DO_se
rvice_zone,
                  data=train_data)

  # predict the responses for the testing data
  kfold_preds_pb4_1[fold_index] <- predict(mod_pb4_1, newdata=test_data)
  kfold_preds_pb4_2[fold_index] <- predict(mod_pb4_2, newdata=test_data)
  kfold_preds_pb4_final[fold_index] <- predict(my_final_mod, newdata=test_data)
  kfold_preds_pb4_4[fold_index] <- predict(mod_pb4_4, newdata=test_data)
  kfold_preds_pb4_5[fold_index] <- predict(mod_pb4_5, newdata=test_data)
  kfold_preds_pb4_6[fold_index] <- predict(mod_pb4_6, newdata=test_data)
}


mean(abs(cabs_sample_cv$fare_amount - kfold_preds_pb4_1))

## [1] 1.91838

mean(abs(cabs_sample_cv$fare_amount - kfold_preds_pb4_2))

## [1] 1.917215

mean(abs(cabs_sample_cv$fare_amount - kfold_preds_pb4_final))

## [1] 1.911148

mean(abs(cabs_sample_cv$fare_amount - kfold_preds_pb4_4))

## [1] 1.913362

mean(abs(cabs_sample_cv$fare_amount - kfold_preds_pb4_5))

## [1] 1.912378

mean(abs(cabs_sample_cv$fare_amount - kfold_preds_pb4_6))

## [1] 1.912157
```

Answer: The third model, fare_amount ~ trip_distance+DO_service_zone+duration, has the lowest mean absolute error as 1.911. Therefore, this model is a strong model for predicting cab fare.