```r
install.packages('glmnet')
install.packages('leaps')
library(glmnet)
library(leaps)
# Q3 --------

## Part A --------
# KNN function for one X-variable

knn_1x = function (training_x,training_y,testing_x,k){
  predicted_y = list()
  for (i in testing_x){
    # distance from each training x to target x
    distance = abs(training_x - i)
    # pick out the nearest k neighbors
    indices = order(distance)[1:k]
    # take the mean of the nearest y values
    predicted_y = append(predicted_y,mean(training_y[indices]))
  }
  return (predicted_y)
}

# ============================
## Part B --------

# Generate Data
set.seed(111)
Data_a_i = rnorm(100,mean = 1,sd = 1)
Data_a_ii = rnorm(100,mean = 2.5,sd = 1)
Data_b_i = rnorm(100,mean = 1,sd = 1)
Data_b_ii = rnorm(100,mean = 4, sd = 1)
y1 = rep(1,100)
y2 = rep(0,100)
Data_a_all = c(Data_a_i,Data_a_ii)
Data_b_all = c(Data_b_i,Data_b_ii)
y_all = c(y1,y2)

# loocv with knn
loocv_knn = function(x,y,k){
  errors = list()
  for (i in 1:length(x)){
    training_x = x[-i]
    training_y = y[-i]
    testing_x = x[i]
    true_y = y[i]
    output = knn_1x(training_x,training_y,testing_x,k)[[1]]
    errors = append(errors,round(output)-true_y)
  }
  return(errors)
}

#loocv with ls
loocv_ls = function(x,y){
  errors = list()
  for (i in 1:length(x)){
    training_x = x[-i]
    training_y = y[-i]
    df = data.frame(x = training_x,y = training_y)
    testing_x = x[i]
    true_y = y[i]
    lm1 = lm(y ~ x, data = df)
    output = predict(lm1, newdata =  data.frame(x = testing_x))
    errors = append(errors,round(output) - true_y)
  }
  return(errors)
}

# error of prediction
```

```
knn5_a = loocv_knn(Data_a_all,y_all,5)
knn10_a = loocv_knn(Data_a_all,y_all,10)
knn15_a = loocv_knn(Data_a_all,y_all,15)
ls_a = loocv_ls(Data_a_all,y_all)

knn5_b = loocv_knn(Data_b_all,y_all,5)
knn10_b = loocv_knn(Data_b_all,y_all,10)
knn15_b = loocv_knn(Data_b_all,y_all,15)
ls_b = loocv_ls(Data_b_all,y_all)

# calculate accuracy by count number of 0s, more 0s more accurate
accuracy_a = c(sum(unlist(knn5_a) == 0),sum(unlist(knn10_a) == 0),sum(unlist(knn15_a)
== 0), sum(unlist(ls_a)==0))
accuracy_b = c(sum(unlist(knn5_b) == 0),sum(unlist(knn10_b) == 0),sum(unlist(knn15_b)
== 0), sum(unlist(ls_b)==0))
accuracy_a
accuracy_b

# We can tell that the accuracy is higher for Data2 as the gap of mean is bigger
# Ls performs better than Knn in Data2 while Knn performs better than Ls in Data1
# for Knn, when difference of mean is small, K=10 performs best and K=15 performs
worst, while in second case, K=15 performs best and K=5 performs worst


#================
# Q5 --------
diabetes = read.csv("/Users/vanris/Downloads/diabetes_F24.txt",sep = "", header = TRUE)
head(diabetes)

#subset_selection
x = as.matrix(diabetes[,c("age","sex","BMI","bp","S1","S2","S3","S4","S5","S6")])
y = diabetes$y

best_subset = leaps(x, y, nbest = 1)
best_subset
model2 = lm(y ~ diabetes$sex + diabetes$BMI + diabetes$bp + diabetes$S1 + diabetes$S2 +
diabetes$S5)
plot(x = c(1:10), y = best_subset$Cp)

# Ridge model use CV to find best parameter lambda
cv_ridge = cv.glmnet(x, y, alpha = 0)
best_lambda_ridge = cv_ridge$lambda.min
final_ridge_model <- glmnet(x, y, alpha = 0, lambda = best_lambda_ridge)
ridge_coefficients <- coef(ridge_model, s = best_lambda_ridge)


lasso_model = glmnet(x, y, alpha = 1)
cv_lasso <- cv.glmnet(x, y, alpha = 1)
best_lambda_lasso <- cv_lasso$lambda.min
final_lasso_model <- glmnet(x, y, alpha = 1, lambda = best_lambda_lasso)
lasso_coefficients <- coef(final_lasso_model, s = best_lambda_lasso)
```