stat415-homework5

1. Perform logistic regression on the training data in order to predict mpg01

```
median_mpg = median(Auto$mpg)
mpg01 = rep("0", nrow(Auto))
for (i in 1:nrow(Auto)){
  if (Auto$mpg[i] > median_mpg){
    mpg01[i] = "1"
  }
}
data_new = cbind(Auto, mpg01)

mpg01_0 = which(data_new$mpg01 == 0)
mpg01_1 = which(data_new$mpg01 == 1)
train_index = c(sample(mpg01_0, size = trunc(0.80 * length(mpg01_0))),
                sample(mpg01_1, size = trunc(0.80 * length(mpg01_1))))
Auto_train = data_new[train_index, ]
Auto_test = data_new[-train_index, ]

# The most associated quantitative variables are horsepower, weight, di
splacement and acceleration
library(MASS)
# Logistic Regression
mpg01 = as.numeric(mpg01)
data_new2 = cbind(Auto, mpg01) # numeric mpg01
Auto_train2 = data_new2[train_index, ]
Auto_test2 = data_new2[-train_index, ]
glm.fit = glm(mpg01 ~ horsepower + weight + acceleration + displacement,
 data=Auto_train2, family = binomial)
summary(glm.fit)
## Call:
## glm(formula = mpg01 ~ horsepower + weight + acceleration + displacem
ent,
##     family = binomial, data = Auto_train2)
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  13.2877094  2.8844742   4.607 4.09e-06 ***
## horsepower   -0.0579451  0.0217346  -2.666  0.00768 **
## weight       -0.0010545  0.0009373  -1.125  0.26056
## acceleration -0.1340170  0.1357340  -0.987  0.32347
## displacement -0.0153865  0.0062065  -2.479  0.01317 *
```

Comment: The Logistic model has been performed above. According to the p-value, the coefficients of Horsepower and displacement are quite significant since they are less than a=0.05. The coefficients of weight and acceleration are not significant here.

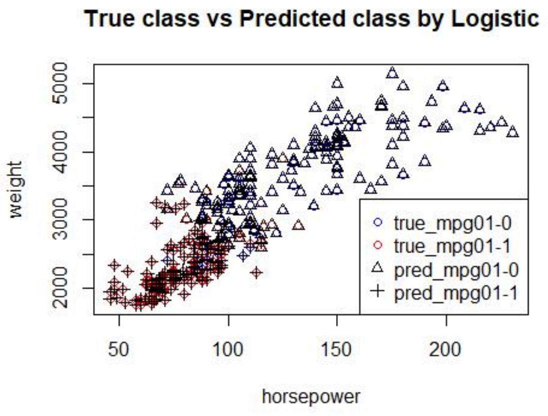
2. Report the training and the test errors for logistic regression.

```
# turn prediction into proper prob
expit = function(x) exp(x) / (1 + exp(x))
glm_train_pred = predict(glm.fit, Auto_train2)
```

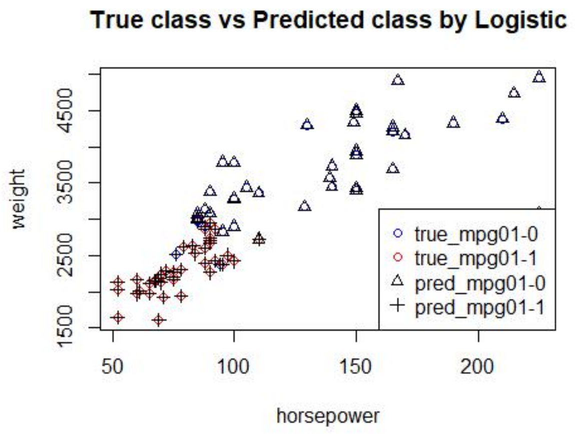
```r
glm_test_pred = predict(glm.fit, Auto_test2)
# decide 0 or 1 based on prob estimation for training data set
trainPrediction = rep("0", nrow(Auto_train2))
trainPrediction[expit(glm_train_pred) > 0.5] = "1"
table(trainPrediction, Auto_train2$mpg01, dnn = c("Predicted", "Actual
"))

trainPrediction = as.numeric(trainPrediction)
glm_train_pred_data = cbind(Auto_train2[,-1],trainPrediction)
# decide 0 or 1 based on prob estimation for testing data set
testPrediction = rep("0", nrow(Auto_test2))
testPrediction[expit(glm_test_pred) > 0.5] = "1"
table(testPrediction, Auto_test2$mpg01, dnn = c("Predicted", "Actual"))

testPrediction = as.numeric(testPrediction)
glm_test_pred_data = cbind(Auto_test2[,-1],testPrediction)
# define the error function
calc_class_err = function(actual, predicted){
  mean(actual != predicted)
}
calc_class_err(predicted = trainPrediction, actual = Auto_train2$mpg01)

## [1] 0.1089744
calc_class_err(predicted = testPrediction, actual = Auto_test2$mpg01)

## [1] 0.0875
```

Comment: According to the result, the training error for Logistic Regression is 0.1089744 and the test error for Logistic Regression is 0.0875.

```r
Auto_train2$mpg01 = as.factor(Auto_train2$mpg01)
plot(Auto_train2$horsepower,Auto_train2$weight, col =
c("blue","red")[Auto_train2$mpg01], xlab = "horsepower", ylab =
"weight", main = "True class vs Predicted class by Logistic")
trainPrediction = as.factor(trainPrediction)
points(glm_train_pred_data$horsepower,glm_train_pred_data$weight, pch =
c(2,3)[trainPrediction])
legend("bottomright", c("true_mpg01-0","true_mpg01-1","pred_mpg01-
0","pred_mpg01-1"), col=c("blue", "red", "black",
"black"),pch=c(1,1,2,3))
```



```
Auto_test2$mpg01 = as.factor(Auto_test2$mpg01)
plot(Auto_test2$horsepower,Auto_test2$weight, col = c("blue","red")[Aut
o_test2$mpg01], xlab = "horsepower", ylab = "weight", main = "True clas
s vs Predicted class by Logistic")
# Plot the predicted test data
testPrediction = as.factor(testPrediction)
points(glm_test_pred_data$horsepower,glm_test_pred_data$weight, pch = c
(2,3)[testPrediction])
legend("bottomright", c("true_mpg01-0","true_mpg01-1","pred_mpg01-0","p
red_mpg01-1"), col=c("blue", "red", "black", "black"),pch=c(1,1,2,3))
```



3. Using your fitted model, estimate the probability of a car having above median mpg if its four predictors you used are all at the median values for the training data set.

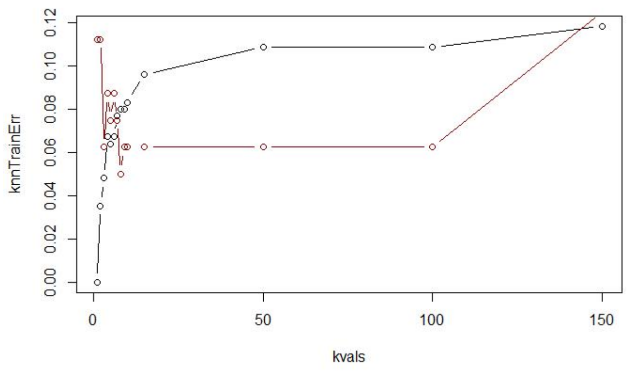
```
median_displacement = median(Auto_train2$displacement)
median_horspower = median(Auto_train2$horsepower)
median_weight = median(Auto_train2$weight)
median_acceleration = median(Auto_train2$acceleration)
result = 13.2877094 -0.0579451*median_horspower -0.0010545*median_weigh
t -0.1340170*median_acceleration -0.0153865*median_displacement
prob = expit(result)
prob

## [1] 0.6187401
```

Comment: According to the result,the prob of a car having above median mpg if its four predictors are at the median values is 0.6187401.

4. Perform KNN classification on the training data.

```
library(class)
trainX = as.matrix(Auto_train2[c("horsepower", "acceleration","displace
ment","weight")])
testX = as.matrix(Auto_test2[c("horsepower", "acceleration","displaceme
nt","weight")])
trainX = scale(trainX)
testX = scale(testX)
set.seed(1)
kvals = c(1:10, 15, 50, 100, 150)
```

```
knnTrainErr = rep(0,length(kvals))
knnTestErr = rep(0,length(kvals))
for (i in 1:length(kvals)) {
knn.pred1 = knn(train = trainX, test = trainX, cl = Auto_train2$mpg01,
k=kvals[i])
knn.pred2 = knn(train = trainX, test = testX, cl = Auto_train2$mpg01, k
=kvals[i])
knnTrainErr[i] <- mean(knn.pred1 != Auto_train2$mpg01)
knnTestErr[i] <- mean(knn.pred2 != Auto_test2$mpg01)
}
par(mfrow=c(1,2))
plot(knnTrainErr ~ kvals, type = "b",col = "black")
points(knnTestErr ~ kvals, type = "b", col = "dark red")
kval_train = kvals[which(knnTrainErr == min(knnTrainErr))]
kval_train

## [1] 1

kval_test = kvals[which(knnTestErr == min(knnTestErr))]
kval_test

## [1] 8
```

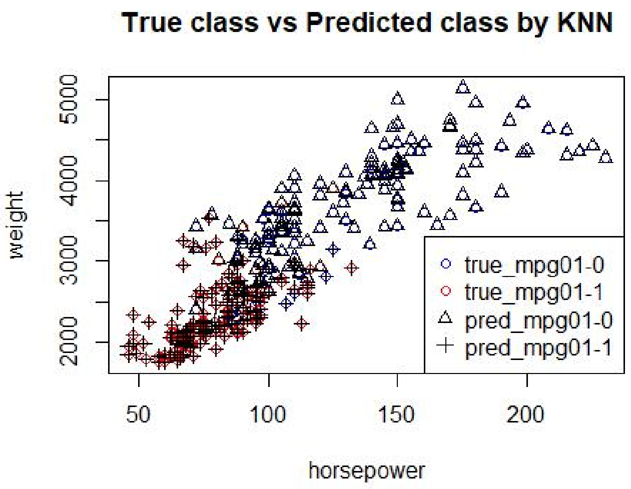


Comment: when k=1, it gives the best performance on the training data. When k=8, it gives the best performance on the test data.

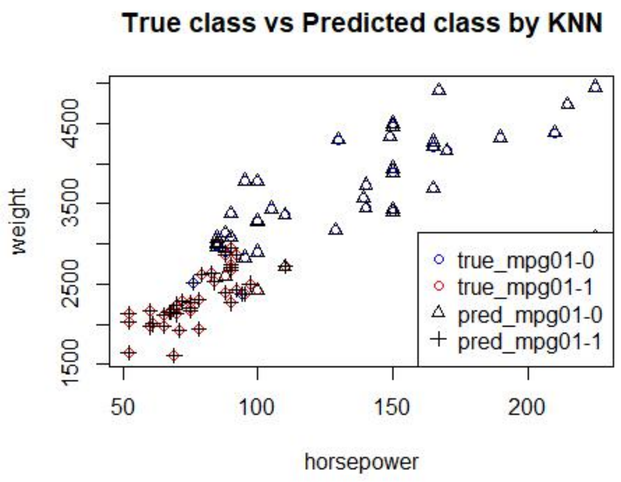
5. Report the training and the test errors for KNN with your choice of K.

```
# choose k=5 for KNN model
# Training data
(knnTrainErr[5])

## [1] 0.06410256

# Test data
(knnTestErr[5])

## [1] 0.075

# plot for training data
knn.pred1 = knn(train = trainX, test = trainX, cl = Auto_train2$mpg01,
k=5)
```

```
knn_train_pred_data = cbind(Auto_train2[,-1],knn.pred1)
Auto_train2$mpg01 = as.factor(Auto_train2$mpg01)
plot(Auto_train2$horsepower,Auto_train2$weight, col = c("blue","red")[A
uto_train2$mpg01], xlab = "horsepower", ylab = "weight", main = "True c
lass vs Predicted class by KNN")
# Plot the predicted train data
knn.pred1 = as.factor(knn.pred1)
points(knn_train_pred_data$horsepower,knn_train_pred_data$weight, pch =
 c(2,3)[knn.pred1])
legend("bottomright", c("true_mpg01-0","true_mpg01-1","pred_mpg01-0","p
red_mpg01-1"), col=c("blue", "red", "black", "black"),pch=c(1,1,2,3))
```



```
# plot for test data
knn.pred2 = knn(train = trainX, test = testX, cl = Auto_train2$mpg01, k
=5)
knn_test_pred_data = cbind(Auto_test2[,-1],knn.pred2)
knn.pred2 = as.factor(knn.pred2)
plot(Auto_test2$horsepower,Auto_test2$weight, col = c("blue","red")[Aut
o_test2$mpg01], xlab = "horsepower", ylab = "weight", main = "True clas
s vs Predicted class by KNN")
points(knn_test_pred_data$horsepower,knn_test_pred_data$weight, pch = c
(2,3)[knn.pred2])
legend("bottomright", c("true_mpg01-0","true_mpg01-1","pred_mpg01-0","p
red_mpg01-1"), col=c("blue", "red", "black", "black"),pch=c(1,1,2,3))
```



Comment: the training error for KNN is 0.06410256 and the test error for KNN is 0.075. Corresponding plots have been shown above and k is 5.

6. Can you answer question 3 with KNN regression? If yes, give the answer. If not, explain why not and what you can report.

```
test_point = c(median_horspower, median_acceleration,median_displacemen
t,median_weight)
data = as.matrix(Auto_test2[c("horsepower", "acceleration","displacemen
t","weight")])
data_append = t(data.frame(t(data),test_point))
data_append = scale(data_append)
point_predict = knn(train = trainX, test = data_append, cl = Auto_train
2$mpg01, k=kvals[3])
point_predict[length(point_predict)]

## [1] 1
```

Comment: we can not give the answer for question 3 since the KNN method is a non-parametric approach. It fails to tell the importance of predictors and can not give coefficients or p-values.But we can use these four median value to deduce which class this point should be in. According to the prediction result, this point's mpg01 should be 1, which indicates that the probability of its mpg being higher than median is larger than 0.5.

7. Compare and contrast the performance of LDA, QDA (take from HW 4), logistic regression, and KNN on this data set.

Comment: the training error and test error of LDA are 0.1185897 and 0.075. The training error and test error of QDA are 0.1025641 and 0.0625. The training error and test error of Logistic regression are 0.1089744 and 0.0875. The training error and test error of KNN are 0.06410256 and 0.075. Thus, KNN is more suitable for this data and QDA also works better than the other two methods. Since QDA works well in fitting, it is very likely that samples for different classes have have different variance. Also, LDA and logistic assume linear boundary while KNN and QDA assume non-linear boundary. According to the error estimation, it is very likely that the decision boundary is non-linear,especially for quadratic boundary.