STATS415hw5

Yunguo Cai Section:003 uniqname:cyunguo

1.The R code for preparing data is the same as Hw4.

```
fit = glm(mpg01~acceleration+displacement+horsepower+weight, data=auto_train, family = binomial)
summary(fit)
```

```
##
## Call:
## glm(formula = mpg01 ~ acceleration + displacement + horsepower +
##     weight, family = binomial, data = auto_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4486  -0.2362   0.0622   0.3898   3.4025
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  13.2877094  2.8844742   4.607 4.09e-06 ***
## acceleration -0.1340170  0.1357340  -0.987  0.32347
## displacement -0.0153865  0.0062065  -2.479  0.01317 *
## horsepower   -0.0579451  0.0217346  -2.666  0.00768 **
## weight       -0.0010545  0.0009373  -1.125  0.26056
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 432.52  on 311  degrees of freedom
## Residual deviance: 170.89  on 307  degrees of freedom
## AIC: 180.89
##
## Number of Fisher Scoring iterations: 7
```

The coefficients of displacement and horsepower are significant (with p-value smaller than 0.05).

2. If the predicted probability is greater than 0.5, we classify the observation as 1; otherwise, we classify it as 0.

```
expit <- function(x){
return (exp(x)/(1 + exp(x)))
}
# train error
pred_train = predict(fit,auto_train)
predProb_train = expit(pred_train)
trainPrediction = rep(0,length(auto_train$mpg01))
trainPrediction[predProb_train>.5]=1
mean(trainPrediction!=auto_train$mpg01)
```

```
## [1] 0.1089744
```

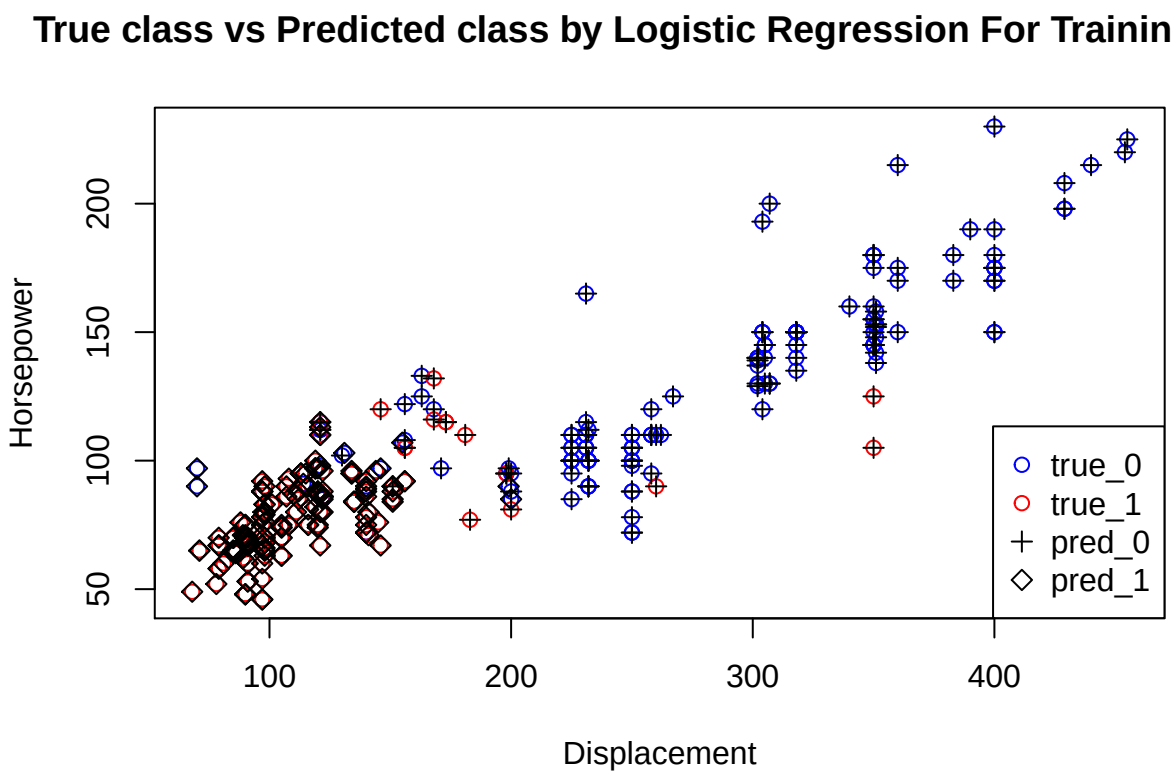
```
# test error
pred_test = predict(fit,auto_test)
predProb_test = expit(pred_test)
```

```
testPrediction = rep(0,length(auto_test$mpg01))
testPrediction[predProb_test>.5]=1
mean(testPrediction!=auto_test$mpg01)
```

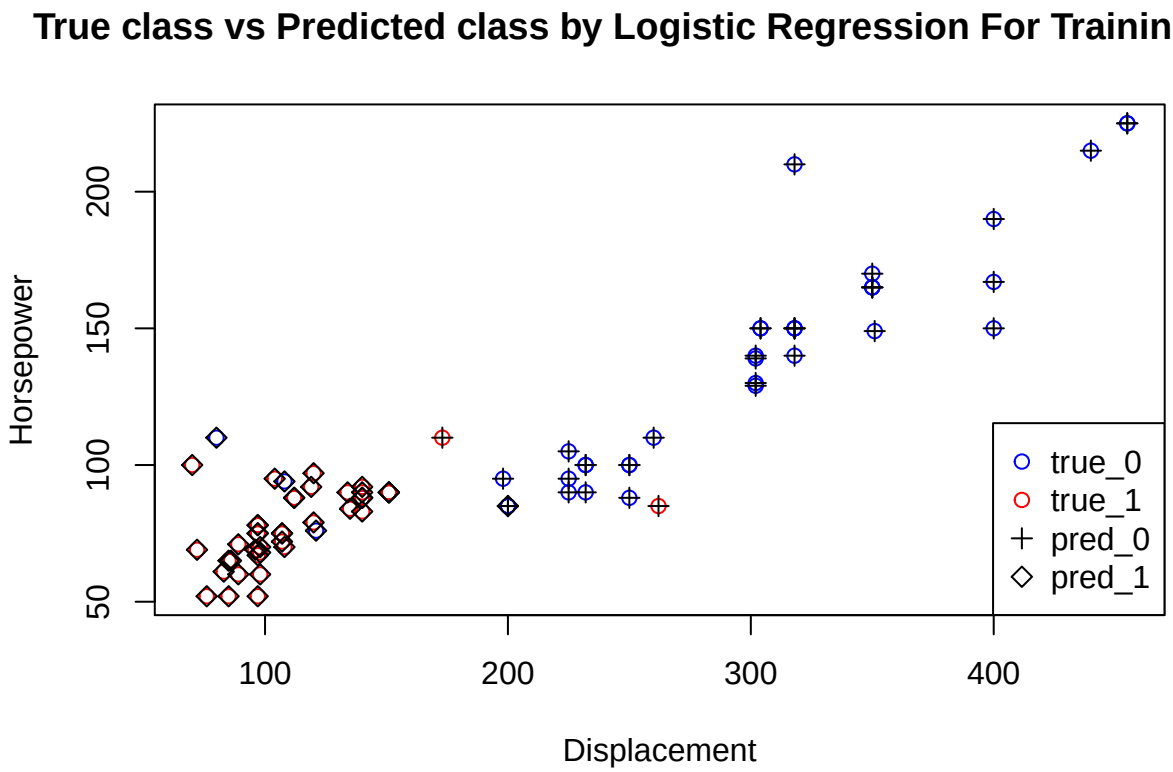
```
## [1] 0.0875
```

The training error for logistic regression is 0.109, and the test error rate is 0.0875.

```
plot(auto_train$displacement,auto_train$horsepower,
     col = c("blue", "red")[auto_train$mpg01],
     xlab = "Displacement", ylab = "Horsepower",
     main = "True class vs Predicted class by Logistic Regression For Training Data")
points(auto_train$displacement,auto_train$horsepower,
       pch = c(3,5)[trainPrediction+1])
legend("bottomright", c("true_0","true_1","pred_0","pred_1"),
col=c("blue", "red", "black", "black"),
pch=c(1,1,3,5))
```




```
plot(auto_test$displacement,auto_test$horsepower, col = c("blue", "red")[auto_test$mpg01],
xlab = "Displacement", ylab = "Horsepower",
main = "True class vs Predicted class by Logistic Regression For Training Data")
points(auto_test$displacement,auto_test$horsepower,
       pch = c(3,5)[testPrediction+1])
legend("bottomright", c("true_0","true_1","pred_0","pred_1"), col=c("blue", "red", "black", "black"),
       pch=c(1,1,3,5))
```



3.

```
pred_median = predict(fit,data.frame(displacement=median(auto_train$displacement),
weight=median(auto_train$weight),
acceleration=median(auto_train$acceleration)))
expit(pred_median)
```

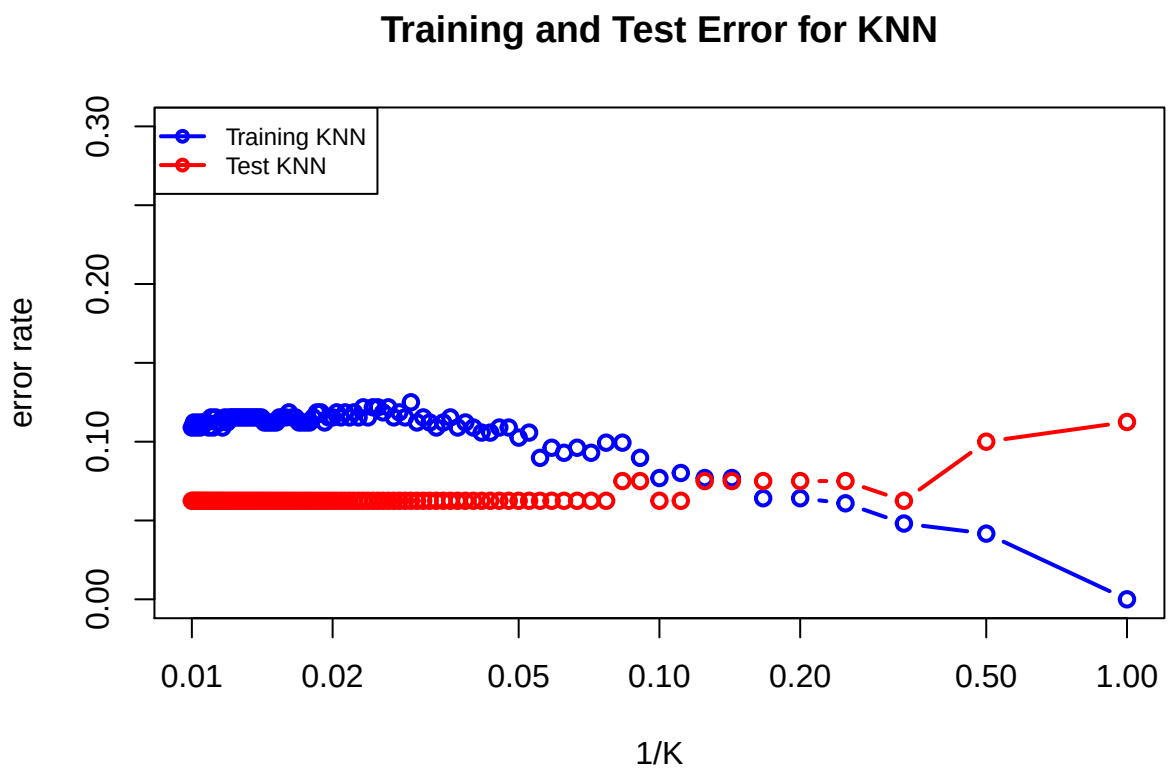
```
##         1
## 0.6187167
```

The estimated probability of a car having above-median mpg is 0.619 if its displacement, horsepower, weight and accerlation are all at the median values for the training dataset.

4.

```
library("FNN")
n=100
k_range = seq(1,n)
knn_train_error = rep(0,n)
knn_test_error = rep(0,n)
for (i in k_range){
knn_train_result = knn(train = scale(auto_train[c("displacement","horsepower","weight","acceleration")]),
                       test = scale(auto_train[c("displacement","horsepower","weight","acceleration")]),
                       cl = auto_train$mpg01, k = k_range[i])
knn_train_error[i] = mean( (knn_train_result!=auto_train$mpg01) )
knn_test_result= knn(train = scale(auto_train[c("displacement","horsepower","weight","acceleration")]),
                     test = scale(auto_test[c("displacement","horsepower","weight","acceleration")]),
                     cl = auto_train$mpg01, k = k_range[i])
knn_test_error[i] = mean( (knn_test_result!=auto_test$mpg01) )
}
k_inverse = 1/k_range
plot(knn_train_error ~ k_inverse, log = "x", type = "b", lwd = 2, col = "blue",
```

```
     pch = 1, lty = 1, main = "Training and Test Error for KNN",xlab = "1/K",
     ylab = "error rate", ylim = c(0,0.3))
lines(knn_test_error ~ k_inverse, type = "b", lwd = 2, col = "red",pch = 1, lty = 1)
legend("topleft", legend = c("Training KNN", "Test KNN"), cex = .75, lwd = c(2, 2),
col = c("blue", "red"), pch = c(1, 1), lty = c(1, 1))
```




```
k_range[which.min(knn_train_error)]
```

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

```
k_range[which.min(knn_test_error)]
```

```
## [1] 3
```

K = 1 gives the best performance on the training data. K = 3 gives the best performance on the test data.

5.

```
knn_train_error[which.min(knn_test_error)]
```

```
## [1] 0.04807692
```

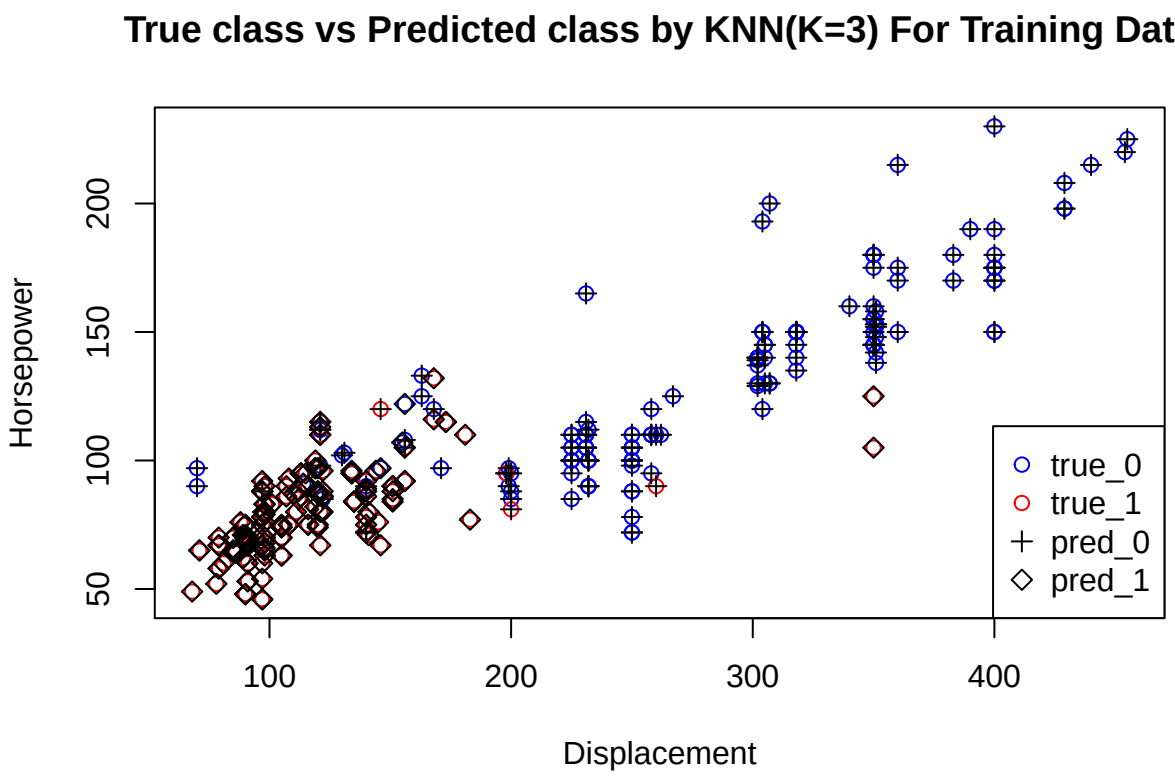
```
knn_test_error[which.min(knn_test_error)]
```

```
## [1] 0.0625
```

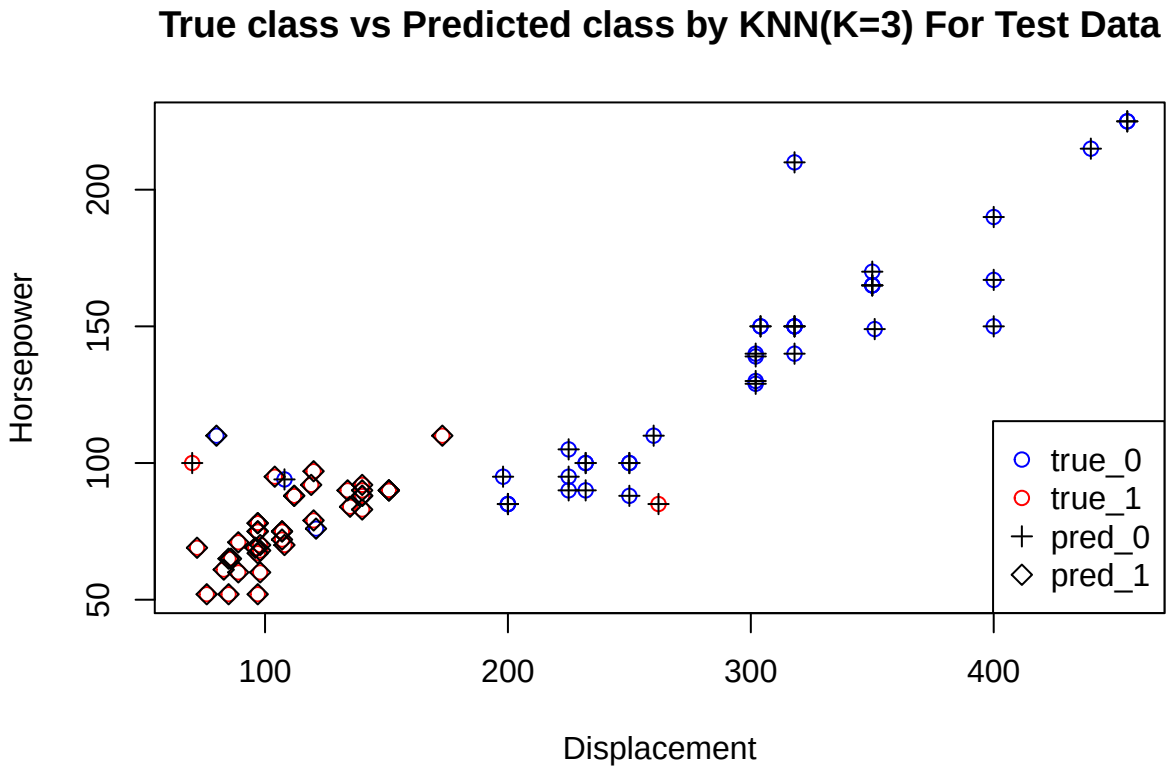
My choice of K is 3, which gives the best performance on the test data. When K = 3, the training error is 0.048, and the test error is 0.0625.

```
knn_train_result = knn(train = scale(auto_train[c("displacement","horsepower","weight","acceleration")]),
                       test = scale(auto_train[c("displacement","horsepower","weight","acceleration")]),
cl = auto_train$mpg01, k = 3)
plot(auto_train$displacement,auto_train$horsepower, col = c("blue", "red")[auto_train$mpg01],
xlab = "Displacement", ylab = "Horsepower",
```

```
main = "True class vs Predicted class by KNN(K=3) For Training Data")
points(auto_train$displacement,auto_train$horsepower,
       pch = c(3,5)[knn_train_result])
legend("bottomright", c("true_0","true_1","pred_0","pred_1"), col=c("blue", "red", "black", "black"),
       pch=c(1,1,3,5))
```




```
knn_test_result = knn(train = scale(auto_train[c("displacement","horsepower","weight","acceleration")])
                      test = scale(auto_test[c("displacement","horsepower","weight","acceleration")]),
cl = auto_train$mpg01, k = 3)
plot(auto_test$displacement,auto_test$horsepower, col = c("blue", "red")[auto_test$mpg01],
xlab = "Displacement", ylab = "Horsepower",
main = "True class vs Predicted class by KNN(K=3) For Test Data")
points(auto_test$displacement,auto_test$horsepower,
       pch = c(3,5)[knn_test_result])
legend("bottomright", c("true_0","true_1","pred_0","pred_1"), col=c("blue", "red", "black", "black"),
       pch=c(1,1,3,5))
```



6.

```
horsepower = median(auto_train$horsepower)
displacement = median(auto_train$displacement)
weight = median(auto_train$weight)
acceleration = median(auto_train$acceleration)
a = data.frame(horsepower,displacement,weight,acceleration)
Test=auto_test[c("displacement","horsepower","weight","acceleration")]
b=rbind(Test,a)
Train=auto_train[c("displacement","horsepower","weight","acceleration")]
knnf=knn(train=scale(Train),test=scale(b),cl=auto_train$mpg01,k=3)
```

No, I can't answer question 3 with KNN regression because we can't know the probability. KNN predict the class of data by determining which class most of its K nearest neighbors belong to. From the code above, we get 1. So we can only predict whether a car would have mpg above median or not with the four predictors are all at the median values for the training dataset.

7. The test errors for LDA, QDA, KNN and Logistic Regression are 0.075, 0.0625, 0.0625 and 0.0875 respectively. QDA and KNN have the lowest test errors and perform the best. LDA is a bit worse, about 1%. The logistic regression preforms the worst. LDA performs better than logistic regression shows that dataset is normally distributed. QDA and KNN works best, better than LDA, shows that the boundary between classes are not linear.