stats 415 hw 5

Li Hsuan Lin

3/10/2020

Question 1

Question 1a

Using (4.2) equation in textbook,

$$Pr(get\ an\ A|x_1\ = 5, x_2\ = 3.5) = \frac{e^{-4+0.05*(5)+1*(3.5)}}{1 + e^{-4+0.05*(5)+1*(3.5)}}$$

$$= 0.4378235$$

```
exp(-4 + 0.05 * 5 + 3.5) /(1 +exp(-4 + 0.05 * 5 + 3.5) )
```

```
## [1] 0.4378235
```

Question 1b

Odds of getting an A = $\frac{p(get\ and\ A\ |x_1=5,x_2=3.5)}{1-p(get\ and\ A\ |x_1=5,x_2=3.5)}$
Thus,

$$desired\ odds = \frac{0.4378235}{(1 - 0.4378235)} = 0.7788008$$

```
0.4378235 / (1-0.4378235)
```

```
## [1] 0.7788008
```

Question 1c

$p(get\ an\ A\ |x_1, x_2) = 0.5$means that odds $\frac{p(get\ and\ A\ |x_1,x_2)}{1-p(get\ and\ A\ |x_1,x_2)} = 1$, which also means the logodds is 0.

$$logodds = \beta_0 + \beta_1 * x_1 + \beta_2 * (3.5) == 0$$

$$-4 + 0.05 * x_1 + 3.5 = 0$$

$$x = 10\ (hrs)$$

Question 2

```
library(ISLR)
library(tidyverse)
```

```
## ─ Attaching packages ──────────────────────────────────────── tidyver
se 1.2.1 ─
```

```
## ✔ ggplot2 3.2.1     ✔ purrr   0.3.3
## ✔ tibble  2.1.3     ✔ dplyr   0.8.3
## ✔ tidyr   1.0.0     ✔ stringr 1.4.0
## ✔ readr   1.3.1     ✔ forcats 0.4.0
```

```
## ─ Conflicts ─────────────────────────────────────────── tidyverse_conf
licts() ─
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(FNN)
```

Reuse the code from my HW4 and the result that cylinders, displacement, horsepower, and weight are good quantative variables.

```
dat = Auto
#create a new binary variable
dat_mpg = dat %>% mutate(mpg01 = ifelse(mpg>25,1,0))
mpg01 = as.factor(dat_mpg$mpg01)
dat_mpg = dat_mpg[,-10]%>% cbind(mpg01)
```

```
RNGkind(sample.kind = "Rejection")
set.seed(123)
value_0 = which(dat_mpg$mpg01 == "0")
value_1 = which(dat_mpg$mpg01 == "1")
train_id = c(sample(value_0, size = trunc(0.8 * length(value_0))),sample(value_1,
size = trunc(0.8 * length(value_1))))
dat_train = dat_mpg[train_id,]
dat_test = dat_mpg[-train_id,]
dat_train%>% head(4)
```

```
##      mpg cylinders displacement horsepower weight acceleration year origin
## 199 18.0         6          250         78   3574         21.0   76      1
## 273 20.3         5          131        103   2830         15.9   78      2
## 228 16.0         8          400        180   4220         11.1   77      1
## 14  14.0         8          455        225   3086         10.0   70      1
##                          name mpg01
## 199       ford granada ghia     0
## 273               audi 5000     0
## 228   pontiac grand prix lj     0
## 14  buick estate wagon (sw)     0
```

Question 2a

As we can see from the regression output, only “horsepower” isj significant at $\alpha = 0.05$ level

```
set.seed(123)
auto_logreg = glm(mpg01 ~ cylinders + displacement + horsepower + weight,data = da
t_train,family = binomial)
summary(auto_logreg)
```

```
## 
## Call:
## glm(formula = mpg01 ~ cylinders + displacement + horsepower + 
##     weight, family = binomial, data = dat_train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.4509  -0.3227  -0.0096   0.4011   3.1862  
## 
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)    
## (Intercept)  11.4334433  2.1397949   5.343 9.13e-08 ***
## cylinders    -0.2384585  0.4551988  -0.524    0.600    
## displacement -0.0083310  0.0112652  -0.740    0.460    
## horsepower   -0.0746357  0.0178597  -4.179 2.93e-05 ***
## weight       -0.0010316  0.0008111  -1.272    0.203    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 419.30  on 311  degrees of freedom
## Residual deviance: 184.06  on 307  degrees of freedom
## AIC: 194.06
## 
## Number of Fisher Scoring iterations: 7
```

Question 2b

The train error rate is 0.1410256, and the test error rate is 0.15.

In hw4, I use weight and displacement.

```
# predicted logodd
train_pred_logodd = predict(auto_logreg,newdata = dat_train)
test_pred_logodd = predict(auto_logreg,newdata = dat_test)

# predicted Probabilites
train_pred_pr = binomial()$linkinv(train_pred_logodd)
test_pred_pr = binomial()$linkinv(test_pred_logodd)

# predicted label y based on probabilities
train_pred_label = ifelse(train_pred_pr > 0.5,1,0)
mean(train_pred_label != dat_train$mpg01)
```

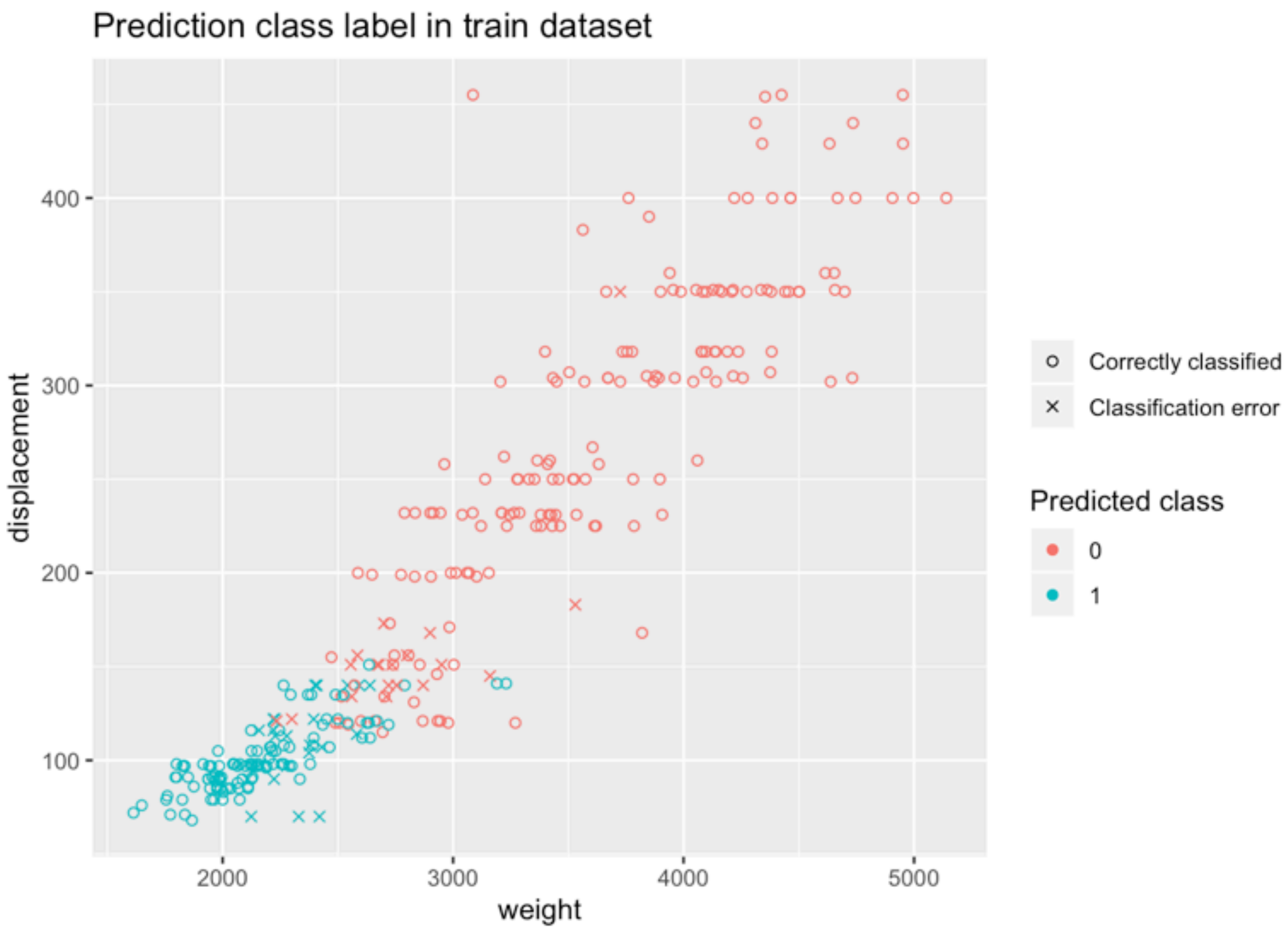
```
## [1] 0.1410256
```

```
test_pred_label = ifelse(test_pred_pr > 0.5,1,0)
mean(test_pred_label != dat_test$mpg01)
```

```
## [1] 0.15
```

```
#drop the attribute
yhat_test = as.vector(test_pred_label)

#factorize the vector
yhat_test = as.factor(test_pred_label)


ggplot(dat_test,
       aes(x=weight, y=displacement, color = yhat_test, shape = (mpg01 == yhat_tes
t))) +
  geom_point(size=1.5) +
  scale_shape_manual(values = c(4, 1),
                     breaks = c(TRUE, FALSE),
                     labels=c('Correctly classified',
                              'Classification error'),
                     name='') +
  scale_color_discrete(name='Predicted class') + labs(title = "Prediction class la
bel in test dataset")
```


```
#drop the attribute
yhat_train = as.vector(train_pred_label)

#factorize the vector
yhat_train = as.factor(train_pred_label)


ggplot(dat_train,
       aes(x=weight, y=displacement, color = yhat_train, shape = (mpg01 == yhat_tr
ain))) +
  geom_point(size=1.5) +
  scale_shape_manual(values = c(4, 1),
                     breaks = c(TRUE, FALSE),
                     labels=c('Correctly classified',
                              'Classification error'),
                     name='') +
  scale_color_discrete(name='Predicted class') + labs(title = "Prediction class la
bel in train dataset")
```



2c:

Using the model in 2b, the desired probability is 0.3722611.

```
four_pred = dat_train[,c("cylinders", "displacement", "horsepower", "weight")]

#median of 4 quantative predictors
map_dbl(four_pred,median)
```

```
##    cylinders displacement   horsepower       weight
##          4.0        151.0         92.0       2789.5
```

```
temp_4 = data.frame(cylinders = 4,displacement = 151,horsepower = 92,weight = 2789
.5)
```

```
#convert to probability
pr_median = binomial()$linkinv(predict(auto_logreg,temp_4))
#silence the attribute
names(pr_median) = NULL
pr_median
```

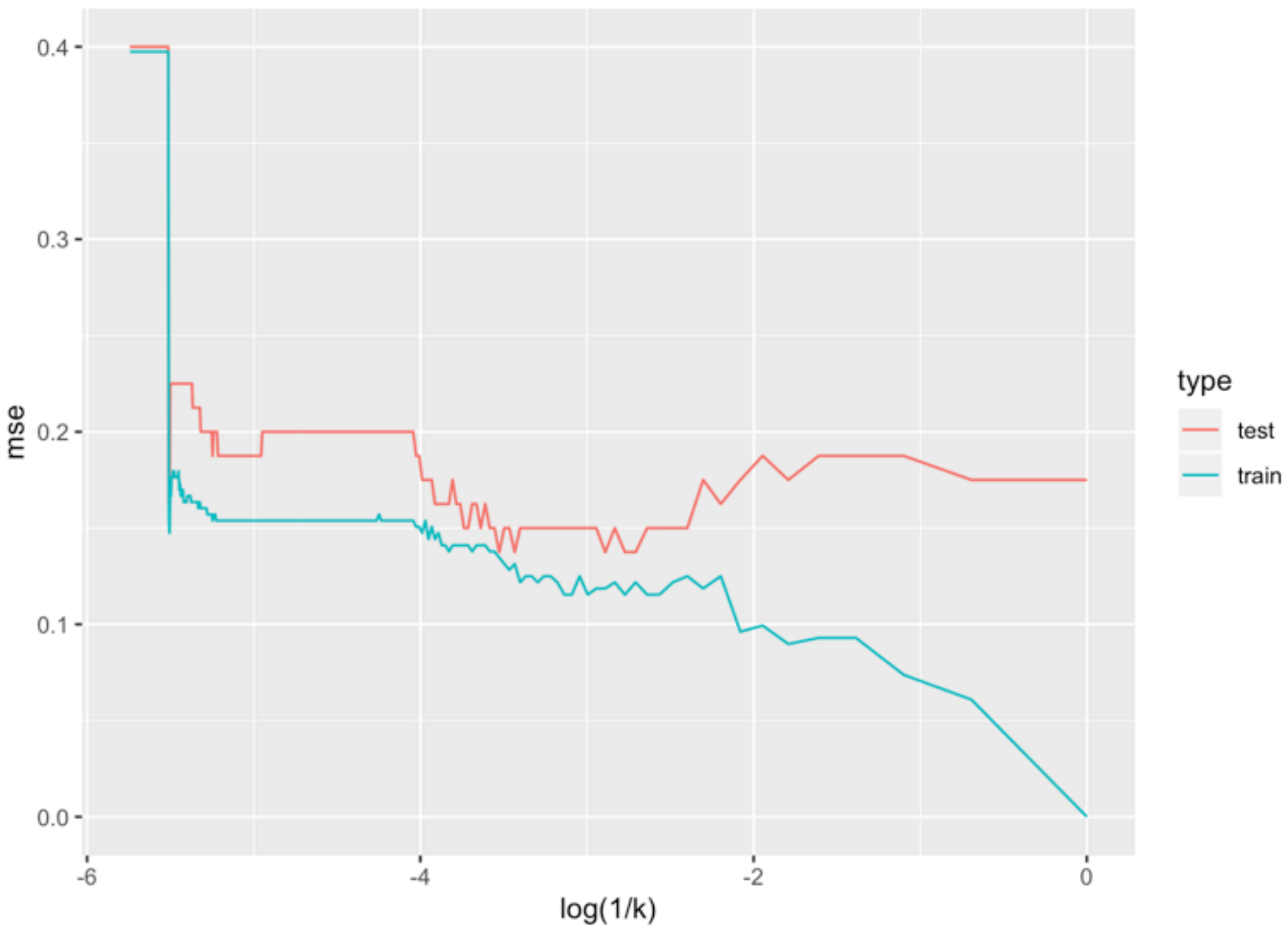
```
## [1] 0.3722611
```

2d:

Based on the result below, the smallest train error rate is 0 when k = 1, and the smallest test error rate is 0.1375 when k is 15,16,18,31,34.

```
k_reg = c(1:nrow(dat_train))

train_knn_mse = rep(NA,length(k_reg))
test_knn_mse = rep(NA,length(k_reg))

#train, standardize
knn_train = dat_train[,c("cylinders", "displacement", "horsepower", "weight")]
knn_test = dat_test[,c("cylinders", "displacement", "horsepower", "weight")]

#mean, std of train set
mean_train = colMeans(knn_train)
std_train = sqrt(diag(var(knn_train)))

#rescale the train and test set
knn_train_scale =  scale(knn_train, center = mean_train , scale = std_train)
knn_test_scale = scale(knn_test, center = mean_train, scale = std_train)

for(i in 1:312){
   mpg_train_pred = knn(train = knn_train_scale,cl = dat_train$mpg01,test = knn_tr
ain_scale,k = k_reg[i])
   mpg_train_pred = factor(mpg_train_pred,levels = levels( dat_train$mpg01))
   train_knn_mse[i] = mean(mpg_train_pred !=dat_train$mpg01 )

   mpg_test_pred = knn(train = knn_train_scale,cl = dat_train$mpg01,test = knn_tes
t_scale,k = k_reg[i])
   mpg_test_pred = factor(mpg_test_pred,levels = levels( dat_test$mpg01))
   test_knn_mse[i] = mean(mpg_test_pred !=dat_test$mpg01 )

}
```

```
knn_dat = data.frame(k = rep(1:312,times = 2),
                    mse = c(train_knn_mse,test_knn_mse),
                    type = c(rep(c("train"),312),rep(c("test"),312)))
ggplot(knn_dat) + geom_line(aes(log(1/k),mse,color =type))
```




```
knn_dat %>%  group_by(type) %>% summarise(mse_min = min(mse))
```

```
## # A tibble: 2 x 2
##   type  mse_min
##   <fct>   <dbl>
## 1 test    0.138
## 2 train   0
```

```
#the value of k which has smallest testing error
knn_dat %>% filter(mse == 0.1375) %>% filter(type =="test")
```

```
##    k    mse type
## 1 15 0.1375 test
## 2 16 0.1375 test
## 3 18 0.1375 test
## 4 31 0.1375 test
## 5 34 0.1375 test
```

```
#the value of k which has smallet training error
knn_dat %>% filter(mse == 0) %>% filter(type =="train")
```

```
##   k mse  type
## 1 1   0 train
```

2e:

As we can see, although when k = 1, it has the smallest training. However, since we care more about the smallest testing error and the fact that simpler model is preferred, I picked k = 34 as it yields the same smallerst testing error and it is simplest model among 5 different k's.

Thus in this case, when k = 34, the training error is 0.1346154 and testing error is 0.1375.

```
knn_dat %>% filter(k==34)
```

```
##    k       mse  type
## 1 34 0.1346154 train
## 2 34 0.1375000  test
```

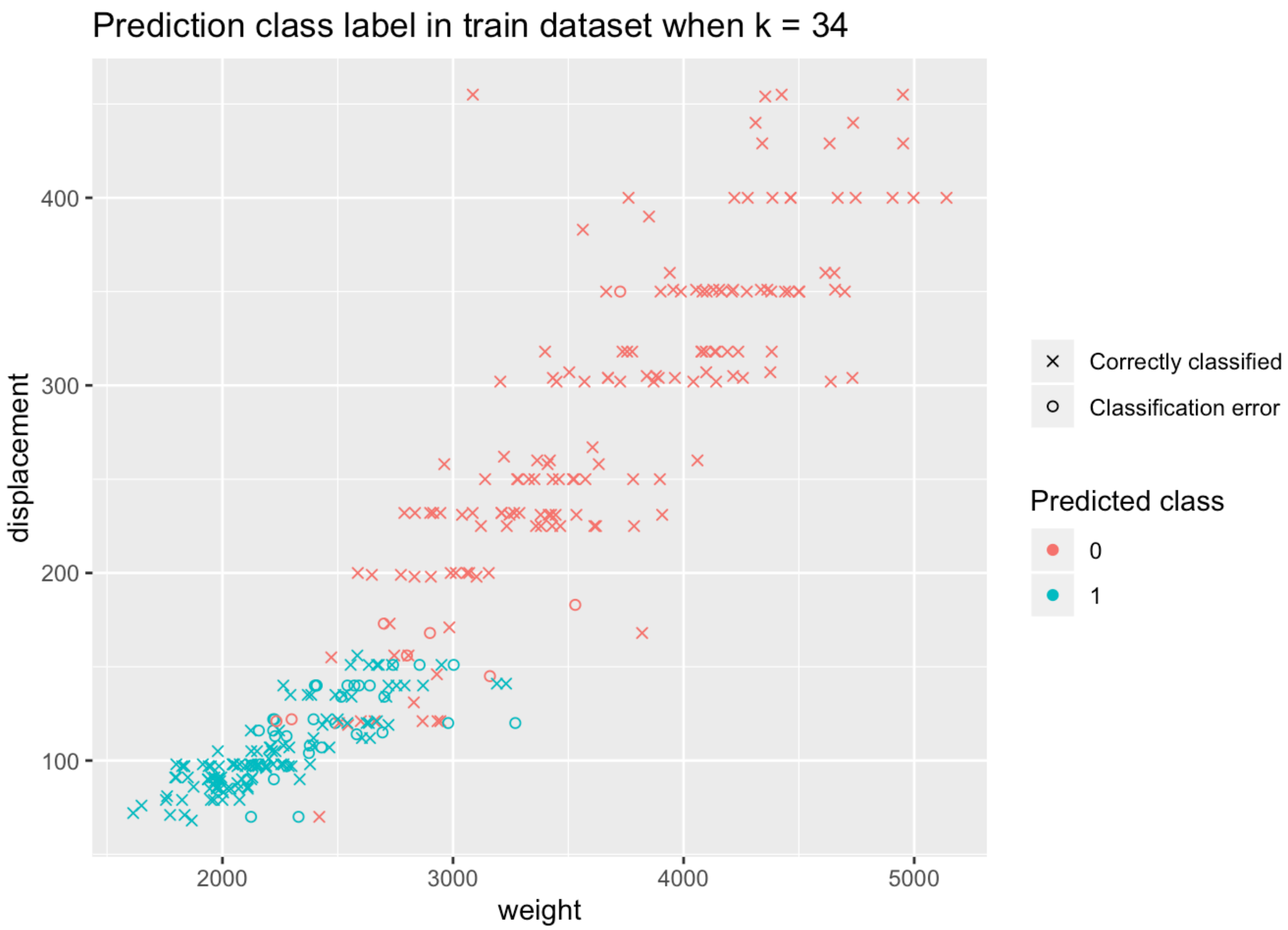
```
knn_34_test = knn(train = knn_train_scale,cl = dat_train$mpg01,test = knn_test_sca
le,k = 34)
```

testing dataset

```
yhat_test_knn = knn_34_test

ggplot(dat_test,
       aes(x=weight, y=displacement, color = yhat_test_knn, shape = (mpg01 == yhat
_test_knn))) +
  geom_point(size=1.5) +
  scale_shape_manual(values = c(4, 1),
                     breaks = c(TRUE, FALSE),
                     labels=c('Correctly classified',
                              'Classification error'),
                     name='') +
  scale_color_discrete(name='Predicted class') + labs(title = "Prediction class la
bel in test dataset when k = 34")
```

training dataset

```
knn_34_train = knn(train = knn_train_scale,cl = dat_train$mpg01,test = knn_train_s
cale,k = 34)
knn_34_train= knn_34_train %>% as.vector()

yhat_train_knn = knn_34_train


ggplot(dat_train,
       aes(x=weight, y=displacement, color = yhat_train_knn, shape = (mpg01 == yha
t_train_knn))) +
  geom_point(size=1.5) +
  scale_shape_manual(values = c(1, 4),
                     breaks = c(TRUE, FALSE),
                     labels=c('Correctly classified',
                              'Classification error'),
                     name='') +
  scale_color_discrete(name='Predicted class') + labs(title = "Prediction class la
bel in train dataset when k = 34")
```



2f:

No we can not estimate the probability since KNN classification only assign label to X based on which label has the most "vote" in given K's nearest neighbors.
We can report the label KNN classification assigns (0 or 1) when four predictors are all at the median values.

2g:

Using data from hw4 and previous part

```
dat_comp = data.frame(type = c("LDA","QDA","logistic","knn"),train = c(0.1474,0.13
78,0.1410,0.1346),test = c(0.2125,0.175,0.15,0.1375))
dat_comp
```

```
##       type  train   test
## 1      LDA 0.1474 0.2125
## 2      QDA 0.1378 0.1750
## 3 logistic 0.1410 0.1500
## 4      knn 0.1346 0.1375
```

Remark:
Based on the result, we can see that all 4 methods has around the same performance on training dataset, while KNN performed the best on testing dataset. Particularly, the testing error of LDA and QDA is quite large compared to KNN. It suggest that the normality assumption of using LDA and QDA is not met. In other words, the distribution of “auto” dataset does not follow multivariate normal distribution. It also suggests the boundary between classes is complicated (not linear nor quadractic) since KNN performed the best, according to the lecture slide (Logistic regression, 22)