OIS 8.6

(a)

df = 31-2-1 = 28, t^*_{df} = 2.05 (95% confidence interval)

CI: $0.34 \pm t^*_{df} * SE = 0.34 \pm 2.05 * 0.13$, $(0.0735, 0.6065)$

We are 95% confident that additional one feet added to the height, the volume of a tree could be increased by 0.0735 to 0.6065 cubic feet when controlling for the other variables in the model.

(b)

$$y = 0.34 height + 4.71 diameter$$

$$y_i = 0.34 * 79 + 4.71 * 11.3 = 80.083$$

$$y_i = 24.2$$

$$y_i - y_i = -55.883$$

The model overestimates the volume of this tree by 55.883 cubic feet.

8.8

The learner status variable should be removed first, since the adjusted R^2 (0.0723) becomes larger than the full model one (0.0701) when removing it.

8.10

For forward selection method, we need to select the variable which has the corresponding largest R^2_{adj}, or the variable which has the corresponding smallest p-value. During the first step of selection, the original R^2_{adj} = 0. From this table, we can find that ethnicity has the largest R^2_{adj} and the smallest p-value. So ethnicity should be added to the model first.

8.16

(a)

In general, as the temperature increases, the damaged O-rings become less. Just at temperature 53°, the number of damaged O-rings is a large number with high proportion of all the O-rings. When temperature rises up to 57°, the number of damaged O-rings rapidly drop down to 1. Then as the temperature continue goes up, the number of damaged O-rings tends to more likely to be 0. Even not 0, the number can only be 1.

(b)

There is only one explanatory variable for this model, which is the temperature. The coefficient of the temperature variable is -0.2162, which implies a negative correlation. That is to say, as the temperature rises, the probability of the code being 0 increases, i.e. it is more likely to produce an undamaged O-ring. The standard error represents the variation and uncertainty, and it can contribute to the calculation of confidence interval. The p-value for temperature coefficient is nearly to 0, which means the null hypothesis can definitely be rejected, i.e. there is super likely a relationship between the temperature and whether it is a damaged or undamaged O-ring.

(c)

The probability of Yi being 1 (for a damaged O-ring) is Pi.

$$p_i = \frac{e^{11.6630-0.2162temperature}}{1+e^{11.6630-0.2162temperature}}$$

(d)

When temperature equals 53, the $p_i = 0.551$. When temperature equals 81, the $p_i = 0.00287$. It is a monotonic decreasing model as temperature increases. There is a significant decrease of the probability to be a damaged O-rings. The concerns regarding O-rings are justified since there exists a statistical significantly relationship between the temperature and the damaged or undamaged O-rings.

8.18

(a)

$$p_i = \frac{e^{11.6630-0.2162temperature}}{1+e^{11.6630-0.2162temperature}}$$

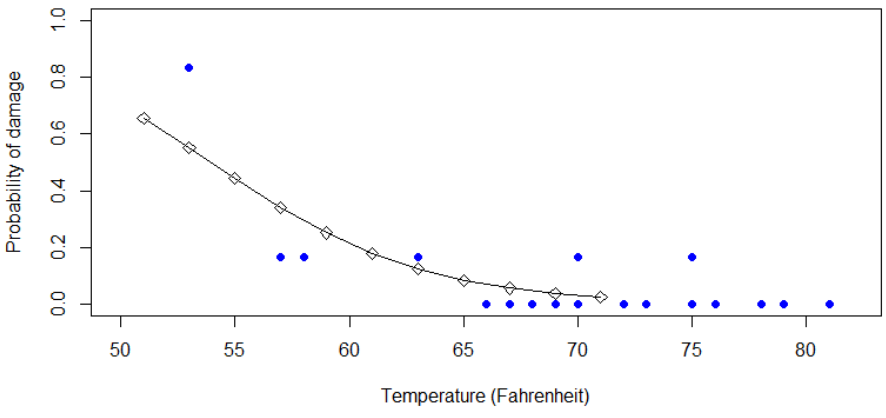
$$p_{51} = \frac{e^{11.6630-0.2162*51}}{1+e^{11.6630-0.2162*51}} = 0.654$$

$$p_{53} = \frac{e^{11.6630-0.2162*53}}{1+e^{11.6630-0.2162*53}} = 0.551$$

$$p_{55} = \frac{e^{11.6630-0.2162*55}}{1+e^{11.6630-0.2162*55}} = 0.443$$

(b) R script:

```
# 8.18

temperature = c(53,57,58,63,66,67,67,67,68,69,70,70,70,70,72,73,75,75,76,76,78,79,81)

probability = c(5/6, 1/6, 1/6,1/6,0,0,0,0,0,0,1/6,0,1/6,0,0,0,0,1/6,0,0,0,0,0)

plot(temperature,probability, xlab = "Temperature (Fahrenheit)", ylab = "Probability of damage",col="blue", ylim = c(0,1), xlim = c(50,82), pch = 19)

estimate.temperature = seq(51,71,by=2)

estimate.probability = c(0.654,0.551,0.443,0.341,0.251,0.179,0.124,0.084,0.056,0.037,0.024)

lines(estimate.temperature, estimate.probability, type = "o", pch = 5)
```



(c)

It is appropriate to use logistic regression model for this circumstance since it satisfies the necessary requirements. First, there is only one predictor in the model, and it is linearly related to logit(pi) without influence by any other predictor. Second, each y_i is independent to each other, which is an essential assumption for the validity of the logistic regression model.

From the plot including true values and model-estimated possibilities, we can see that there are some true points located at both sides of the estimated fitting curve. There is no apparent outlier either. So it seems a good model.

ISL chapter 4 #13 (for logistic and KNN only, not LDA)

R script:

```
### ISL 4.13

install.packages("MASS")

library("MASS", lib.loc="C:/Program Files for operation/R-3.3.1/library")

dim(Boston)

View(Boston)


Boston$crim.level = as.numeric(Boston$crim > median(Boston$crim))

# above the median is 1, below the median is 0


table(Boston$crim.level)

# 0   1

# 253 253

# 253 above, 253 below


# training set: 75% of the dataset

# test set: 25% of the dataset

sample.size = floor(0.75 * nrow(Boston))

set.seed(1)

sample.index = sample(1:nrow(Boston), size = sample.size, replace = FALSE)

Boston.train = Boston[sample.index, ]

Boston.test = Boston[-sample.index, ]
```

```
########################

# Logistic Regression

# Backward elimination


glm.fit = glm(crim.level ~ lstat+rm+zn+nox+dis+rad+ptratio+black+medv+age+chas+indus+tax, data=Boston.train, family = binomial)

summary(glm.fit)

# lstat has the highest p-value, so remove it from the model.


glm.fit = glm(crim.level ~ rm+zn+nox+dis+rad+ptratio+black+medv+age+chas+indus+tax, data=Boston.train, family = binomial)

summary(glm.fit)

# rm has the highest p-value, so remove it from the model.


glm.fit = glm(crim.level ~ zn+nox+dis+rad+ptratio+black+medv+age+chas+indus+tax, data=Boston.train, family = binomial)

summary(glm.fit)

# chas has the highest p-value, so remove it from the model.


glm.fit = glm(crim.level ~ zn+nox+dis+rad+ptratio+black+medv+age+indus+tax, data=Boston.train, family = binomial)

summary(glm.fit)

# black has the highest p-value, so remove it from the model.


glm.fit = glm(crim.level ~ zn+nox+dis+rad+ptratio+medv+age+indus+tax, data=Boston.train, family = binomial)

summary(glm.fit)

# indus has the highest p-value, so remove it from the model.


glm.fit = glm(crim.level ~ zn+nox+dis+rad+ptratio+medv+age+tax, data=Boston.train, family = binomial)

summary(glm.fit)

# ptratio has the highest p-value, so remove it from the model.
```

```
glm.fit = glm(crim.level ~ zn+nox+dis+rad+medv+age+tax, data=Boston.train, family = binomial)
summary(glm.fit)
# All p-values are less than 0.05.

# The final model through logistic regression using backward elimination based on p-value:

# Call:
#   glm(formula = crim.level ~ zn + nox + dis + rad + medv + age +
#         tax, family = binomial, data = Boston.train)
#
# Deviance Residuals:
#   Min       1Q   Median       3Q      Max
# -1.9177  -0.2099   0.0002   0.0085   3.4612
#
# Coefficients:
#   Estimate Std. Error z value Pr(>|z|)
# (Intercept) -28.975814   5.096740  -5.685 1.31e-08 ***
#   zn          -0.099758   0.036843  -2.708 0.006776 **
#   nox          39.414339   7.034725   5.603 2.11e-08 ***
#   dis          0.757036   0.253424   2.987 0.002815 **
#   rad          0.585181   0.154083   3.798 0.000146 ***
#   medv         0.072261   0.035329   2.045 0.040817 *
#   age          0.039316   0.011929   3.296 0.000982 ***
#   tax         -0.007144   0.002840  -2.516 0.011878 *
#   ---
#   Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#
# (Dispersion parameter for binomial family taken to be 1)
#
# Null deviance: 525.38  on 378  degrees of freedom
# Residual deviance: 165.87  on 371  degrees of freedom
# AIC: 181.87
```

```
#
# Number of Fisher Scoring iterations: 9


# test model
glm.probs=predict(glm.fit, Boston.test, type="response")
glm.pred=rep(0,nrow(Boston.test))
glm.pred[glm.probs > 0.50]=1
table(glm.pred,Boston.test$crim.level)
# glm.pred  0  1
#        0 58  9
#        1  7 53
mean(glm.pred==Boston.test$crim.level)
# The predicted accuracy rate is 0.8740157.



########################
# KNN
install.packages("class")
library("class", lib.loc="~/R/win-library/3.3")

set.seed(1)

# use the predictors selected from logistic regreesion
train.Boston = Boston.train[,c("nox","rad","medv","age","tax","dis","zn")]
test.Boston =  Boston.test[,c("nox","rad","medv","age","tax","dis","zn")]

knn.pred=knn(train.Boston,test.Boston,Boston.train$crim.level,k=1)

# test the model
table(knn.pred,Boston.test$crim.level)
# knn.pred  0  1
```

```
#       0 63  8
#       1  2 54
mean(knn.pred==Boston.test$crim.level)
# The predicted accuracy rate is 0.9212598

knn.pred=knn(train.Boston,test.Boston,Boston.train$crim.level,k=2)

# test the model
table(knn.pred,Boston.test$crim.level)
# knn.pred  0  1
#       0 60 10
#       1  5 52
mean(knn.pred==Boston.test$crim.level)
# The predicted accuracy rate is 0.8818898

knn.pred=knn(train.Boston,test.Boston,Boston.train$crim.level,k=3)

# test the model
table(knn.pred,Boston.test$crim.level)
# knn.pred  0  1
#       0 63 10
#       1  2 52
mean(knn.pred==Boston.test$crim.level)
# The predicted accuracy rate is 0.9055118

knn.pred=knn(train.Boston,test.Boston,Boston.train$crim.level,k=4)

# test the model
table(knn.pred,Boston.test$crim.level)
# knn.pred  0  1
#       0 61 10
#       1  4 52
```

```
mean(knn.pred==Boston.test$crim.level)
# The predicted accuracy rate is 0.8897638

knn.pred=knn(train.Boston,test.Boston,Boston.train$crim.level,k=5)

# test the model
table(knn.pred,Boston.test$crim.level)
# knn.pred  0  1
# 0 60 10
# 1  5 52
mean(knn.pred==Boston.test$crim.level)
# The predicted accuracy rate is 0.8818898

knn.pred=knn(train.Boston,test.Boston,Boston.train$crim.level,k=6)

# test the model
table(knn.pred,Boston.test$crim.level)
# knn.pred  0  1
# 0 60 10
# 1  5 52
mean(knn.pred==Boston.test$crim.level)
# The predicted accuracy rate is 0.8818898

knn.pred=knn(train.Boston,test.Boston,Boston.train$crim.level,k=10)

# test the model
table(knn.pred,Boston.test$crim.level)
# knn.pred  0  1
#       0 58 10
#       1  7 52
mean(knn.pred==Boston.test$crim.level)
# The predicted accuracy rate is 0.8661417
```

```
# So, for KNN, when k = 1, the predicted accuracy rate is the highest of 92.13%.
# k = 3 is also good with a 90.55% predicted accuracy rate. Afterwards, as k becomes higher,
# the accuracy rate tends to decline slowly.
# And the higher k might get more robust classification.

# The above KNN classifiers all adopt the predictors selected by the previous logistic regression model.

# For logistic regression model the predicted accuracy rate is 87.4%. So KNN can improve the
# accuracy rate to some extent.


# use all the predictors instead
train.Boston = Boston.train
test.Boston =  Boston.test

knn.pred=knn(train.Boston,test.Boston,Boston.train$crim.level,k=1)

# test the model
table(knn.pred,Boston.test$crim.level)
# knn.pred  0  1
#          0 62  8
#          1  3 54
mean(knn.pred==Boston.test$crim.level)
# The predicted accuracy rate is 0.9133858, which is smaller than that of trimmed predictors model.
```