HW5

Your Name Goes Here

Details

Due Date

Please commit your submission for this assignment by 5:00 PM Wednesday Oct 30.

Grading

20% of your grade on this assignment is for completion. A quick pass will be made to ensure that you've made a reasonable attempt at all problems.

Some of the problems will be graded more carefully for correctness. In grading these problems, an emphasis will be placed on full explanations of your thought process. You usually won't need to write more than a few sentences for any given problem, but you should write complete sentences! Understanding and explaining the reasons behind your decisions is more important than making the "correct" decision.

Solutions to all problems will be provided.

Collaboration

You are allowed to work with others on this assignment, but you must complete and submit your own write up. You should not copy large blocks of code or written text from another student.

Sources

You may refer to class notes, our textbook, Wikipedia, etc.. All sources you refer to must be cited in the space I have provided at the end of this problem set.

In particular, you may find the following resources to be valuable:

- Courses assigned on DataCamp
- Example R code from class
- Cheat sheets and resources linked from [http://www.evanlray.com/stat340_f2019/resources.html]

Load Packages

The following R code loads packages needed in this assignment.

```
library(readr)
library(dplyr)
library(ggplot2)
library(caret)
```

Conceptual Problems

If you prefer, you can write out your answers to the conceptual problems by hand and then either turn in a physical copy or scan and commit them to GitHub.

Problem 1: Decision boundaries in logistic regression.

Suppose I fit the following variation on a logistic regression model with two quantitative explanatory variables:

$$\begin{aligned} Y_i | x_{i1}, x_{i2} &\sim \text{Bernoulli}(f(x_{i1}, x_{i2})) \\ f(x_{i1}, x_{i2}) &= \frac{e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i2}}}{1 + e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i2}}} \end{aligned}$$

Each Y_i is assumed independent of the others.

(a) Find an expression for the decision boundary if you will predict $\hat{Y}_i = 1$ whenever $\hat{f}(x_{i1}, x_{i2}) > 0.5$.

We need to solve for values of x_{i1} and x_{i2} such that $f(x_{i1}, x_{i2}) = 0.5$.

$$\begin{aligned} 0.5 &= f(x_{i1}, x_{i2}) \\ &= \frac{e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i2}}}{1 + e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i2}}} \\ \Rightarrow 0.5 + 0.5 e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i2}} &= e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i2}} \\ \Rightarrow \quad 1 &= e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i2}} \\ \Rightarrow \quad 0 &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i2} \\ \Rightarrow \quad x_{i2} &= \frac{-\beta_0}{\beta_3} + \frac{-\beta_1}{\beta_3} x_{i1} + \frac{-\beta_2}{\beta_3} x_{i1}^2 \end{aligned}$$

(b) Find an expression for the decision boundary if you will predict $\hat{Y}_i = 1$ whenever $\hat{f}(x_{i1}, x_{i2}) > 0.1$.

We need to solve for values of x_{i1} and x_{i2} such that $f(x_{i1}, x_{i2}) = 0.1$.

$$
\begin{aligned}
0.1 &= f(x_{i1}, x_{i2}) \\
&= \frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i1}^2+\beta_3 x_{i2}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i1}^2+\beta_3 x_{i2}}} \\
\Rightarrow 0.1 + 0.1e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i1}^2+\beta_3 x_{i2}} &= e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i1}^2+\beta_3 x_{i2}} \\
\Rightarrow \quad \frac{1}{9} &= e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i1}^2+\beta_3 x_{i2}} \\
\Rightarrow \quad \log\left(\frac{1}{9}\right) &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i2} \\
\Rightarrow \quad x_{i2} &= \frac{-\beta_0 + \log(1/9)}{\beta_3} + \frac{-\beta_1}{\beta_3} x_{i1} + \frac{-\beta_2}{\beta_3} x_{i1}^2
\end{aligned}
$$

Problem 2: More decision boundaries

(a) Suppose you are doing a classification task. Your sample size is large, you don't have many explanatory variables, and the true decision boundary is highly non-linear. Would you prefer a KNN classification model or a logistic regression model? Why? Your justification should reference the sample size, number of explanatory variables, and shape of the decision boundary.

We would prefer a KNN model in this case. If the sample size was very small or the number of explanatory variable was very large we would not prefer KNN because it suffers more in those cases (in the second case, because of the curse of dimensionality). In this case, we don't need to worry about that so KNN and logistic regression are both possibilities. Since the decision boundary is highly non-linear, KNN is more of a natural choice. To get highly non-linear decision boundaries out of logistic regression, we have to include many polynomial terms and experiment to get the correct specification. It is easier to just use KNN which will automatically capture non-linear decision boundaries.

(b) Suppose you are doing a classification task. Your sample size is relatively small, you have 100 explanatory variables all of which may be relevant to the classification task, and based on a pairs plot it looks like the decision boundary is approximately linear. Would you prefer a KNN classification model or a logistic regression model? Why? Your justification should reference the sample size, number of explanatory variables, and shape of the decision boundary.

Since our sample size is small and there are 100 explanatory variables, we should not use KNN since its performance degrades quickly as the number of explanatory variables increases because of the curse of dimensionality. Additionally, since the decision boundary is approximately linear logistic regression is a good choice since it will give a linear decision boundary.

Problem 3: Adapted from ISLR Example 4.6.

Suppose we collect data for a group of students in a statistics class with variables

- X_1 = hours studied
- X_2 = undergrad GPA
- Y = receive an A in this class ("Yes" or "No")

We fit a logistic regression model and produce estimated coefficients, $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, and $\hat{\beta}_2 = 1$.

(a) What is the interpretation of the coefficient estimate $\hat{\beta}_1 = 0.05$, in terms of the odds of getting an A?

For each increase of one hour in time spent studying, the estimated odds of getting an A increases by a multiplicative factor of $e^{0.05} \approx 1.051$.

(b) Estimate the *probability* that a student who studies for 40 hours and has an undergrad GPA of 3.5 gets an A in the class.

$\hat{P}(\text{Student gets an A}) = \frac{e^{(-6+0.05*40+1*3.5)}}{1+e^{(-6+0.05*40+1*3.5)}} = 0.378$

(c) Estimate the *probability* that a student who studies for 41 hours and has an undergrad GPA of 3.5 gets an A in the class.

$\hat{P}(\text{Student gets an A}) = \frac{e^{(-6+0.05*41+1*3.5)}}{1+e^{(-6+0.05*41+1*3.5)}} = 0.389$

(d) By using your answer to part (b) and the definition of odds, estimate the *odds* that a student who studies for the class for 40 hours and has an undergrad GPA of 3.5 gets an A in the class. Do this again for the *odds* that a student who studies for the class for 41 hours and has an undergrad GPA of 3.5 gets an A in the class, using your answer to part (c). Verify that the interpretation you gave in part (a) holds in this example.

For student studying 40 hours:

$$\begin{aligned}
\widehat{\text{Odds}}(\text{Student gets an A}) &= \frac{\hat{P}(\text{Student gets an A})}{\hat{P}(\text{Student doesn't get an A})} \\
&= \frac{\hat{P}(\text{Student gets an A})}{1 - \hat{P}(\text{Student gets an A})} \\
&= \frac{\hat{P}(0.378)}{1 - 0.378)} \\
&= 0.608
\end{aligned}$$

For student studying 41 hours:

$$\begin{aligned}
\widehat{\text{Odds}}(\text{Student gets an A}) &= \frac{\hat{P}(\text{Student gets an A})}{\hat{P}(\text{Student doesn't get an A})} \\
&= \frac{\hat{P}(\text{Student gets an A})}{1 - \hat{P}(\text{Student gets an A})} \\
&= \frac{\hat{P}(0.389)}{1 - 0.389)} \\
&= 0.637
\end{aligned}$$

Verifying interpretation:

0.608 * 1.051 = 0.639. This is not quite the result of 0.637, but the difference is rounding error.

(e) Suppose a student has an undergrad GPA of 3.5. How many hours would they need to study for us to estimate that there is a probability of 0.5 that they will get an A in the class?

We solve for x in the following equation:

$$0.5 = \frac{e^{(-6+0.05*x+1*3.5)}}{1+e^{(-6+0.05*x+1*3.5)}}$$

Rearranging, we obtain

$$1 = e^{(-6+0.05*x+1*3.5)}$$

Taking logs yields

$$0 = -6 + 0.05 * x + 1 * 3.5$$

Now solving for x gives

$$x = 50$$

The student would have to study for 50 hours for us to estimate the probability that they will get an A is 0.5.

Applied Problems

Problem 4: Breast cancer diagnosis

This problem is adapted from an example in the book "Extending the Linear Model with R" by Julian J. Faraway.

The data set `wbca`, read in below, comes from a study of breast cancer in Wisconsin. There are 681 cases of potentially concerous tumors of which 238 are actually malignant. Determining whether a tumor is really malignant is traditionally determined by an invasive surgical procedure. The purpose of this study was to determine whether a new procedure called fine needle aspiration, which draws only a small sample of tissue, could be effective in determining tumor status.

```
wbca <- faraway::wbca %>%
  mutate(
    Class = ifelse(Class == 1, "Benign", "Malignant"),
    Class01 = ifelse(Class == "Benign", 0, 1)
  )
set.seed(38355)
```

(a) Obtain a train/test split of the data, and further obtain a partition of your training set into 10 folds for cross-validation.

```
train_inds <- caret::createDataPartition(wbca$Class, p = 0.7)
train_wbca <- wbca %>% slice(train_inds[[1]])
test_wbca <- wbca %>% slice(-train_inds[[1]])

xval_folds <- caret::createFolds(train_wbca$Class, k = 10)
```

(b) There are 9 predictor variables, which are measures of different characteristics of the tissue sample drawn from fine needle aspiration. Use a backwards stepwise variable selection procedure to pick a model that uses a subset of these variables. The following sub-parts of this problem guide you through this process.

i. Find a cross-validated classification error rate from a model that uses all 9 predictor variables. Also fit a model with all 9 predictor variables to the full training data set and print out a model summary. This is your starting model.

```
fit <- train(
  form = Class ~ Adhes + BNucl + Chrom + Epith + Mitos + NNucl + Thick + UShap + USize,
  data = train_wbca,
  family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
  method = "glm", # method for fit; "generalized linear model"
  trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
)
summary(fit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.71547  -0.10989  -0.06268   0.02149   2.57792
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.3005     1.4568  -7.071 1.54e-12 ***
## Adhes         0.1420     0.1714   0.829 0.407261
## BNucl         0.4298     0.1262   3.406 0.000658 ***
## Chrom         0.5637     0.2077   2.714 0.006641 **
## Epith         0.1829     0.2063   0.887 0.375304
## Mitos         0.3464     0.3490   0.993 0.320925
## NNucl         0.3474     0.1519   2.287 0.022206 *
## Thick         0.5613     0.1813   3.095 0.001968 **
## UShap         0.3346     0.2784   1.202 0.229495
## USize        -0.1447     0.2626  -0.551 0.581706
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.587  on 477  degrees of freedom
## Residual deviance:  68.761  on 468  degrees of freedom
## AIC: 88.761
##
## Number of Fisher Scoring iterations: 8
```

```
error_rate <- rep(NA, 10)
for(i in 1:10) {
  traintrain <- train_wbca %>% slice(-xval_folds[[i]])
  valval <- train_wbca %>% slice(xval_folds[[i]])

  fit <- train(
```

```
    form = Class ~ Adhes + BNucl + Chrom + Epith + Mitos + NNucl + Thick + UShap + USize,
    data = traintrain,
    family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
    method = "glm", # method for fit; "generalized linear model"
    trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
  )

  preds <- predict(fit, newdata = valval)

  error_rate[i] <- mean(preds != valval$Class)
}
error_rate
```

```
##  [1] 0.02083333 0.06250000 0.04166667 0.04166667 0.02083333 0.08333333
##  [7] 0.02040816 0.02127660 0.00000000 0.02127660
```

```
mean(error_rate)
```

```
## [1] 0.03337947
```

ii. Find a cross-validated classification error rate from a model that uses the 8 predictor variables with the smallest individual p-values in your model from part i. Also fit this model with 8 predictor variables to the full training data set and print out a model summary.

```
error_rate <- rep(NA, 10)
for(i in 1:10) {
  traintrain <- train_wbca %>% slice(-xval_folds[[i]])
  valval <- train_wbca %>% slice(xval_folds[[i]])

  fit <- train(
    form = Class ~ Adhes + BNucl + Chrom + Epith + Mitos + NNucl + Thick + UShap,
    data = traintrain,
    family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
    method = "glm", # method for fit; "generalized linear model"
    trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
  )

  preds <- predict(fit, newdata = valval)

  error_rate[i] <- mean(preds != valval$Class)
}
error_rate
```

```
##  [1] 0.02083333 0.06250000 0.02083333 0.04166667 0.02083333 0.08333333
##  [7] 0.02040816 0.02127660 0.00000000 0.02127660
```

```
mean(error_rate)
```

```
## [1] 0.03129614
```

```
fit <- train(
  form = Class ~ Adhes + BNucl + Chrom + Epith + Mitos + NNucl + Thick + UShap,
  data = train_wbca,
  family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
  method = "glm", # method for fit; "generalized linear model"
  trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
```

```
)
summary(fit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.70847  -0.11329  -0.06472   0.02169   2.55247
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.1145     1.3911  -7.271 3.58e-13 ***
## Adhes         0.1361     0.1718   0.792 0.428200
## BNucl         0.4308     0.1258   3.425 0.000615 ***
## Chrom         0.5297     0.1987   2.666 0.007676 **
## Epith         0.1684     0.2016   0.835 0.403648
## Mitos         0.3433     0.3470   0.989 0.322439
## NNucl         0.3296     0.1498   2.201 0.027725 *
## Thick         0.5506     0.1809   3.043 0.002341 **
## UShap         0.2306     0.2146   1.074 0.282606
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.587  on 477  degrees of freedom
## Residual deviance:  69.052  on 469  degrees of freedom
## AIC: 87.052
##
## Number of Fisher Scoring iterations: 8
```

iii. Find a cross-validated classification error rate from a model that uses the 7 predictor variables with the smallest individual p-values in your model from part ii. Also fit this model with 7 predictor variables to the full training data set and print out a model summary.

```
error_rate <- rep(NA, 10)
for(i in 1:10) {
  traintrain <- train_wbca %>% slice(-xval_folds[[i]])
  valval <- train_wbca %>% slice(xval_folds[[i]])

  fit <- train(
    form = Class ~ BNucl + Chrom + Epith + Mitos + NNucl + Thick + UShap,
    data = traintrain,
    family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
    method = "glm", # method for fit; "generalized linear model"
    trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
  )

  preds <- predict(fit, newdata = valval)

  error_rate[i] <- mean(preds != valval$Class)
```

```
}
error_rate
```

```
##  [1] 0.02083333 0.06250000 0.04166667 0.04166667 0.02083333 0.08333333
##  [7] 0.02040816 0.02127660 0.00000000 0.00000000
```

```
mean(error_rate)
```

```
## [1] 0.03125181
```

```
fit <- train(
  form = Class ~ BNucl + Chrom + Epith + Mitos + NNucl + Thick + UShap,
  data = train_wbca,
  family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
  method = "glm", # method for fit; "generalized linear model"
  trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
)
summary(fit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.66118  -0.11732  -0.06898   0.02696   2.58887
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -9.9478     1.3421  -7.412 1.24e-13 ***
## BNucl         0.4615     0.1205   3.830 0.000128 ***
## Chrom         0.5307     0.1972   2.691 0.007125 **
## Epith         0.1860     0.2055   0.905 0.365385
## Mitos         0.3557     0.3413   1.042 0.297368
## NNucl         0.3262     0.1493   2.185 0.028907 *
## Thick         0.5318     0.1770   3.005 0.002658 **
## UShap         0.2686     0.2109   1.273 0.202898
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.587  on 477  degrees of freedom
## Residual deviance:  69.671  on 470  degrees of freedom
## AIC: 85.671
##
## Number of Fisher Scoring iterations: 8
```

iv. Find a cross-validated classification error rate from a model that uses the 6 predictor variables with the smallest individual p-values in your model from part iii. Also fit this model with 6 predictor variables to the full training data set and print out a model summary.

```
error_rate <- rep(NA, 10)
for(i in 1:10) {
  traintrain <- train_wbca %>% slice(-xval_folds[[i]])
```

```
  valval <- train_wbca %>% slice(xval_folds[[i]])

  fit <- train(
    form = Class ~ BNucl + Chrom + Mitos + NNucl + Thick + UShap,
    data = traintrain,
    family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
    method = "glm", # method for fit; "generalized linear model"
    trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
  )

  preds <- predict(fit, newdata = valval)

  error_rate[i] <- mean(preds != valval$Class)
}
error_rate
```

```
##  [1] 0.02083333 0.04166667 0.02083333 0.04166667 0.02083333 0.08333333
##  [7] 0.02040816 0.02127660 0.00000000 0.00000000
```

```
mean(error_rate)
```

```
## [1] 0.02708514
```

```
fit <- train(
  form = Class ~ BNucl + Chrom + Mitos + NNucl + Thick + UShap,
  data = train_wbca,
  family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
  method = "glm", # method for fit; "generalized linear model"
  trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
)
summary(fit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.75021  -0.11746  -0.06896   0.02683   2.47972
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -9.6858     1.2648  -7.658 1.89e-14 ***
## BNucl         0.4865     0.1188   4.093 4.25e-05 ***
## Chrom         0.5423     0.1927   2.814  0.00490 **
## Mitos         0.3641     0.3356   1.085  0.27795
## NNucl         0.3391     0.1463   2.318  0.02047 *
## Thick         0.5250     0.1743   3.012  0.00259 **
## UShap         0.3385     0.1983   1.707  0.08788 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.587  on 477  degrees of freedom
```

```
## Residual deviance:  70.446  on 471  degrees of freedom
## AIC: 84.446
##
## Number of Fisher Scoring iterations: 8
```

v. Find a cross-validated classification error rate from a model that uses the 5 predictor variables with the smallest individual p-values in your model from part iv. Also fit this model with 5 predictor variables to the full training data set and print out a model summary.

```
error_rate <- rep(NA, 10)
for(i in 1:10) {
  traintrain <- train_wbca %>% slice(-xval_folds[[i]])
  valval <- train_wbca %>% slice(xval_folds[[i]])

  fit <- train(
    form = Class ~ BNucl + Chrom + NNucl + Thick + UShap,
    data = traintrain,
    family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
    method = "glm", # method for fit; "generalized linear model"
    trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
  )

  preds <- predict(fit, newdata = valval)

  error_rate[i] <- mean(preds != valval$Class)
}
error_rate
```

```
##  [1] 0.02083333 0.04166667 0.02083333 0.04166667 0.02083333 0.08333333
##  [7] 0.02040816 0.04255319 0.00000000 0.00000000
```

```
mean(error_rate)
```

```
## [1] 0.0292128
```

```
fit <- train(
  form = Class ~ BNucl + Chrom + NNucl + Thick + UShap,
  data = train_wbca,
  family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
  method = "glm", # method for fit; "generalized linear model"
  trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
)
summary(fit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.59128  -0.11885  -0.06334   0.02559   2.46357
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -9.6606     1.2624  -7.653 1.97e-14 ***
## BNucl         0.4910     0.1201   4.090 4.32e-05 ***
```

```
## Chrom         0.5594     0.1898   2.948 0.003200 **
## NNucl         0.3655     0.1443   2.533 0.011311 *
## Thick         0.6068     0.1650   3.678 0.000235 ***
## UShap         0.3085     0.1912   1.613 0.106641
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.59  on 477  degrees of freedom
## Residual deviance:  72.21  on 472  degrees of freedom
## AIC: 84.21
##
## Number of Fisher Scoring iterations: 8
```

vi. Find a cross-validated classification error rate from a model that uses the 4 predictor variables with the smallest individual p-values in your model from part v. Also fit this model with 4 predictor variables to the full training data set and print out a model summary.

```
error_rate <- rep(NA, 10)
for(i in 1:10) {
  traintrain <- train_wbca %>% slice(-xval_folds[[i]])
  valval <- train_wbca %>% slice(xval_folds[[i]])

  fit <- train(
    form = Class ~ BNucl + Chrom + NNucl + Thick,
    data = traintrain,
    family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
    method = "glm", # method for fit; "generalized linear model"
    trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
  )

  preds <- predict(fit, newdata = valval)

  error_rate[i] <- mean(preds != valval$Class)
}
error_rate
```

```
##  [1] 0.02083333 0.04166667 0.04166667 0.04166667 0.02083333 0.08333333
##  [7] 0.02040816 0.04255319 0.02127660 0.00000000
```

```
mean(error_rate)
```

```
## [1] 0.0334238
```

```
fit <- train(
  form = Class ~ BNucl + Chrom + NNucl + Thick,
  data = train_wbca,
  family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
  method = "glm", # method for fit; "generalized linear model"
  trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
)
summary(fit)
```

```
##
## Call:
```

```
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.88308  -0.12685  -0.06105   0.02418   2.63713
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -9.9944     1.2887  -7.755 8.80e-15 ***
## BNucl         0.5665     0.1145   4.946 7.56e-07 ***
## Chrom         0.6631     0.1728   3.837 0.000124 ***
## NNucl         0.4445     0.1315   3.380 0.000726 ***
## Thick         0.7096     0.1572   4.513 6.39e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.587  on 477  degrees of freedom
## Residual deviance:  75.167  on 473  degrees of freedom
## AIC: 85.167
##
## Number of Fisher Scoring iterations: 8
```

vii. Find a cross-validated classification error rate from a model that uses the 3 predictor variables with the smallest individual p-values in your model from part vi. Also fit this model with 3 predictor variables to the full training data set and print out a model summary.

```
error_rate <- rep(NA, 10)
for(i in 1:10) {
  traintrain <- train_wbca %>% slice(-xval_folds[[i]])
  valval <- train_wbca %>% slice(xval_folds[[i]])

  fit <- train(
    form = Class ~ BNucl + Chrom + Thick,
    data = traintrain,
    family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
    method = "glm", # method for fit; "generalized linear model"
    trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
  )

  preds <- predict(fit, newdata = valval)

  error_rate[i] <- mean(preds != valval$Class)
}
error_rate
```

```
##  [1] 0.06250000 0.04166667 0.04166667 0.04166667 0.00000000 0.04166667
##  [7] 0.02040816 0.08510638 0.02127660 0.00000000
```

```
mean(error_rate)
```

```
## [1] 0.03559578
```

```r
fit <- train(
  form = Class ~ BNucl + Chrom + Thick,
  data = train_wbca,
  family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
  method = "glm", # method for fit; "generalized linear model"
  trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
)
summary(fit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.81267  -0.15653  -0.06927   0.02584   2.61956
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -9.8945     1.1698  -8.459  < 2e-16 ***
## BNucl         0.6532     0.1102   5.927 3.09e-09 ***
## Chrom         0.7973     0.1602   4.976 6.48e-07 ***
## Thick         0.8177     0.1447   5.651 1.59e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.587  on 477  degrees of freedom
## Residual deviance:  90.015  on 474  degrees of freedom
## AIC: 98.015
##
## Number of Fisher Scoring iterations: 8
```

viii. Find a cross-validated classification error rate from a model that uses the 2 predictor variables with the smallest individual p-values in your model from part vii. Also fit this model with 2 predictor variables to the full training data set and print out a model summary.

```r
error_rate <- rep(NA, 10)
for(i in 1:10) {
  traintrain <- train_wbca %>% slice(-xval_folds[[i]])
  valval <- train_wbca %>% slice(xval_folds[[i]])

  fit <- train(
    form = Class ~ BNucl + Thick,
    data = traintrain,
    family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
    method = "glm", # method for fit; "generalized linear model"
    trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
  )

  preds <- predict(fit, newdata = valval)
```

```
  error_rate[i] <- mean(preds != valval$Class)
}
error_rate
```

```
##  [1] 0.06250000 0.08333333 0.06250000 0.04166667 0.04166667 0.02083333
##  [7] 0.10204082 0.06382979 0.00000000 0.00000000
```

```
mean(error_rate)
```

```
## [1] 0.04783706
```

```
fit <- train(
  form = Class ~ BNucl + Thick,
  data = train_wbca,
  family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
  method = "glm", # method for fit; "generalized linear model"
  trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
)
summary(fit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5136  -0.2447  -0.0619   0.0385   2.6547
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -8.0465     0.8567  -9.392  < 2e-16 ***
## BNucl         0.8692     0.1065   8.165 3.23e-16 ***
## Thick         0.9209     0.1292   7.128 1.02e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.59  on 477  degrees of freedom
## Residual deviance: 127.77  on 475  degrees of freedom
## AIC: 133.77
##
## Number of Fisher Scoring iterations: 7
```

ix. Find a cross-validated classification error rate from a model that uses the 1 predictor variable with the smallest individual p-value in your model from part viii. Also fit this model with 1 predictor variable to the full training data set and print out a model summary.

```
error_rate <- rep(NA, 10)
for(i in 1:10) {
  traintrain <- train_wbca %>% slice(-xval_folds[[i]])
  valval <- train_wbca %>% slice(xval_folds[[i]])

  fit <- train(
    form = Class ~ BNucl,
```

```
    data = traintrain,
    family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
    method = "glm", # method for fit; "generalized linear model"
    trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
  )

  preds <- predict(fit, newdata = valval)

  error_rate[i] <- mean(preds != valval$Class)
}
error_rate
```

```
##  [1] 0.06250000 0.10416667 0.10416667 0.06250000 0.10416667 0.06250000
##  [7] 0.14285714 0.10638298 0.02127660 0.08510638
```

```
mean(error_rate)
```

```
## [1] 0.08556231
```

```
fit <- train(
  form = Class ~ BNucl,
  data = train_wbca,
  family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
  method = "glm", # method for fit; "generalized linear model"
  trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
)
summary(fit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3153  -0.3659  -0.3659   0.0907   2.3398
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.57715    0.28326  -12.63   <2e-16 ***
## BNucl        0.90686    0.09048   10.02   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.59  on 477  degrees of freedom
## Residual deviance: 233.16  on 476  degrees of freedom
## AIC: 237.16
##
## Number of Fisher Scoring iterations: 6
```

x. Looking back at your results from parts i. through ix, which model would you choose? If two different models have the same classification error rate, choose the simpler of those two models (that is, the one with fewer explanatory variables). This is a common approach, based on the idea that a simpler model is less likely to have overfit the training or validation data.

I would use the model with the lowest cross-validated classification error rate. In my analysis above, this was the model that included the following explanatory variables: BNucl, Chrom, Mitos, NNucl, Thick, and UShap.

(c) Using your selected model, construct a confusion matrix for the model's classifications on the test set. Find the classification accuracy, classification error rate, false positive rate, and true positive rate for classifications on the test set.

```
fit <- train(
  form = Class ~ BNucl + Chrom + Mitos + NNucl + Thick + UShap,
  data = train_wbca,
  family = "binomial", # this is an argument to glm; response is 0 or 1, binomial
  method = "glm", # method for fit; "generalized linear model"
  trControl = trainControl(method = "none", classProbs = TRUE, savePredictions = TRUE)
)

test_pred <- predict(fit, newdata = test_wbca)

table(test_wbca$Class, test_pred)
```

```
##            test_pred
##             Benign Malignant
##   Benign       130         2
##   Malignant      3        68
```

Classification Accuracy: $(130 + 68)/(130 + 2 + 3 + 68) = 0.9753695$

Classification Error Rate: $(2 + 3)/(130 + 2 + 3 + 68) = 0.02463054$

False Positive Rate: $2/(130 + 2) = 0.01515152$

True Positive Rate: $68/(3 + 68) = 0.9577465$

(d) Suppose we change the cut off to 0.1, so that we will classify a tumor as malignant if the estimated probability that it is malignant is at least 0.1.

i. Using your selected model, construct a confusion matrix for the model's classifications on the test set. Find the classification accuracy, classification error rate, false positive rate, and true positive rate for classifications on the test set.

```
test_pred_prob <- predict(fit, newdata = test_wbca, type = "prob")
test_pred <- ifelse(test_pred_prob[["Malignant"]] > 0.1, "Malignant", "Benign")

table(test_wbca$Class, test_pred)
```

```
##            test_pred
##             Benign Malignant
##   Benign       126         6
##   Malignant      1        70
```

Classification Accuracy: $(126 + 70)/(126 + 6 + 1 + 70) = 0.9655172$

Classification Error Rate: $(6 + 1)/(126 + 6 + 1 + 70) = 0.03448276$

False Positive Rate: $6/(126 + 6) = 0.04545455$

True Positive Rate: $70/(1 + 70) = 0.9859155$

ii. Explain why someone might realistically use a cut off of 0.1 in the context of medical screening using this procedure.

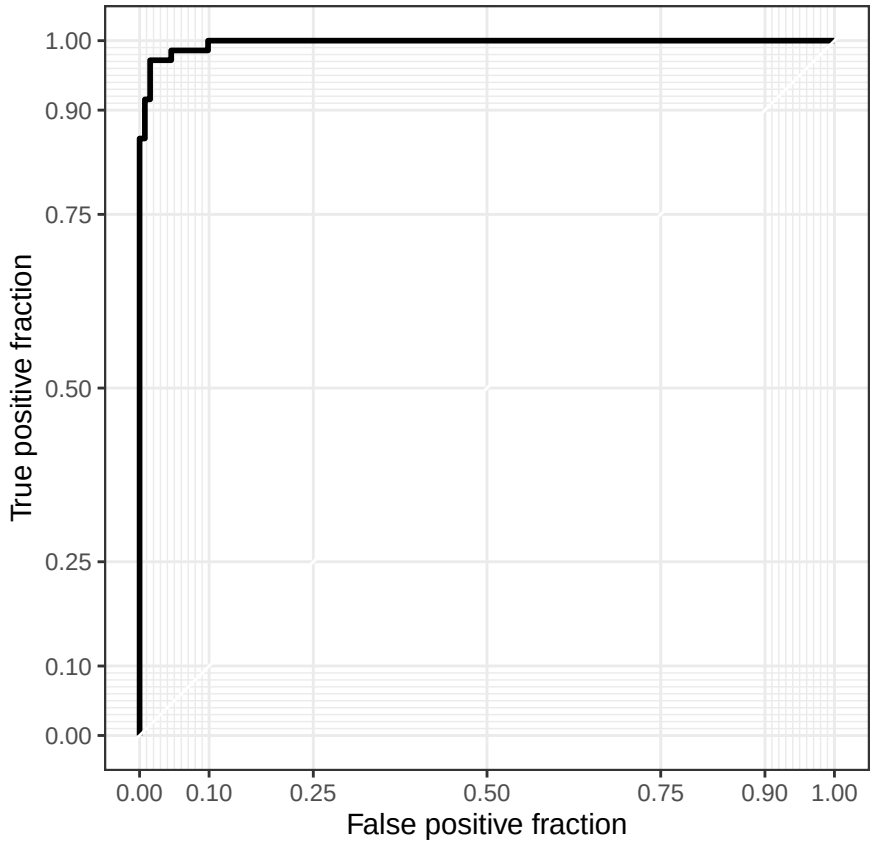
Since we are doing a non-invasive screening with a new medical procedure, I might like to follow up with the standard screening procedure if there is any reasonable chance that the person has a malignant tumor.

(e) Produce a ROC plot and find the area under the curve (AUC) based on the test set predictions. How do your answers to parts (c) and (d) i. relate to the ROC plot?

```
library(plotROC)

test_wbca <- test_wbca %>%
  mutate(
    prob_hat = predict(fit, newdata = test_wbca, type = "prob")[["Malignant"]]
  )

p <- ggplot(data = test_wbca, mapping = aes(m = prob_hat, d = Class01)) +
  geom_roc(n.cuts=0) +
  coord_equal() +
  style_roc()
p
```




```
calc_auc(p)
```

```
##   PANEL group       AUC
## 1     1    -1 0.9966923
```

The false positive and true positive rates from parts (c) and (d) i. give two of the points that make up the ROC plot.

(f) Does the AUC you found in part (e) indicate a good predictive model? A "Yes" or "No" answer is good enough.

Yes (we want high AUC).

Collaboration and Sources

If you worked with any other students on this assignment, please list their names here.

If you referred to any sources (including our text book), please list them here. No need to get into formal citation formats, just list the name of the book(s) you used or provide a link to any online resources you used.