Logistic Regression Homework

Load the faraway package and then type “wbca” at the prompt… but without the quotes. Read about this dataset here: https://cran.r-project.org/web/packages/faraway/faraway.pdf. There are 681 cases of potentially cancerous 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.

(a) Fit a binomial regression with Class as the response and the other nine variables as predictors. Just do

```
model1 <- glm(Class ~ ., data = wbca, family=binomial)
summary(model1)
```

Report the residual deviance and associated degrees of freedom. Can this information be used to determine if this model fits the data well (think in terms of sample sizes)? If so, what does the chi-squared test for the residual deviance indicate?

Here’s the output:

```
Call:
glm(formula = Class ~ ., family = binomial, data = wbca)

Deviance Residuals:
     Min       1Q    Median       3Q       Max
-2.48282  -0.01179   0.04739   0.09678   3.06425

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 11.16678    1.41491   7.892 2.97e-15 ***
Adhes       -0.39681    0.13384  -2.965  0.00303 **
BNucl       -0.41478    0.10230  -4.055 5.02e-05 ***
Chrom       -0.56456    0.18728  -3.014  0.00257 **
Epith       -0.06440    0.16595  -0.388  0.69795
Mitos       -0.65713    0.36764  -1.787  0.07387 .
NNucl       -0.28659    0.12620  -2.271  0.02315 *
Thick       -0.62675    0.15890  -3.944 8.01e-05 ***
UShap       -0.28011    0.25235  -1.110  0.26699
USize        0.05718    0.23271   0.246  0.80589
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 881.388  on 680  degrees of freedom
Residual deviance:  89.464  on 671  degrees of freedom
AIC: 109.46

Number of Fisher Scoring iterations: 8
```

In order that the Chi-square approximation to work well, we need at least several observations of each treatment… for example, we would need 5 or ten observations where Adhes = 1, BNucl=1, Chrom =3, Epth=2, … and 5 or ten observations where Adhes=2, BNucl = 1, Chrom = 3, Epith = 2, etc. Probably there are not enough. Even so, I will get the p-value for a chi-squared test:

```
> pchisq(deviance(model1), df.residual(model1), lower=FALSE)
[1] 1
```

This indicates the model fits sufficiently well. This is good, since we're still using all the variables at this point. This is probably one of the better fitting models we could get!

(b) Use AIC as the criterion in best-subsets method to determine the best subset of variables. This is easy- just do

```
reduced <- step(model1)
summary(reduced)
```

Here's my output:

```
Call:
glm(formula = Class ~ Adhes + BNucl + Chrom + Mitos + NNucl +
    Thick + UShap, family = binomial, data = wbca)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.44161  -0.01119   0.04962   0.09741   3.08205

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  11.0333     1.3632   8.094 5.79e-16 ***
Adhes        -0.3984     0.1294  -3.080  0.00207 **
BNucl        -0.4192     0.1020  -4.111 3.93e-05 ***
Chrom        -0.5679     0.1840  -3.085  0.00203 **
Mitos        -0.6456     0.3634  -1.777  0.07561 .
NNucl        -0.2915     0.1236  -2.358  0.01837 *
Thick        -0.6216     0.1579  -3.937 8.27e-05 ***
UShap        -0.2541     0.1785  -1.423  0.15461
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 881.388  on 680  degrees of freedom
Residual deviance:  89.662  on 673  degrees of freedom
AIC: 105.66

Number of Fisher Scoring iterations: 8
```

Again using the Chi-square test:

```
> pchisq(deviance(reduced), df.residual(reduced), lower=FALSE)
[1] 1
```

Checking the null model (model with no predictors, just the intercept term):

```
> pchisq(881, 680, lower=FALSE)
[1] 2.74918e-07
```

(c) Use the reduced model (the one from part (b)) to predict the outcome (malignant or not) for a patient with

Adhes = 1, BNucl = 1, Chrom = 3, Epith = 2, Mitos = 1,
NNucl = 1, Thick = 4, UShap = 1, USize = 1.

Also, give a 95% confidence interval for your prediction.

```
> nd <- data.frame(Adhes=1, BNucl=1, Chrom = 3, Epith = 2, Mitos = 1, NNucl=1,
Thick=4, UShap=1, USize=1)
> predict(model1, nd, type="response")
        1
0.9923019
```

Be careful now… when you go to the reduced model, you no longer have the Epith and Usize… so

```
> nd2 <- data.frame(Adhes=1, BNucl=1, Chrom = 3, Mitos = 1, NNucl=1, Thick=4,
UShap=1)
> predict(reduced, nd2, type="response")
            1
0.9921115
```

Both the full model and the reduced model predict “Benign” for this person. In case you’re interested, you can also get the predicted values like this:

```
> x0<-c(1,1,1,3,1,1,4,1)
> ilogit1<-sum(x0*coef(reduced))
> ilogit1
```

```
[1] 4.834428
> ilogit(ilogit1)
[1] 0.9921115
```

We can also get a 95% prediction interval for the fit like this:

```
> predict(reduced, newdata=data.frame(Adhes=1, BNucl=1, Chrom=3, Mitos=1,
NNucl=1, Thick=4, UShap=1), se=T)
$fit
        1
4.834428

$se.fit
[1] 0.5815185

$residual.scale
[1] 1
```

These numbers refer to the linear prediction… we can use them to make a 95% CI for the logistic regression p() function at this set of predictor values like this:

```
>  ilogit(c(4.834428-1.96*.5815185,4.834428+1.96*.5815185))
[1] 0.9757467 0.9974629
```

That is, we can be about 95% confident that, provided all the model assumptions are met, that the true p() value for Adhes=1, BNucl=1, etc. is between 0.9757467 and 0.9974629.

(d) Suppose a tumor is classified as benign if $p > 0.5$ and classified as malignant if $p < 0.5$ (remember “1” means benign and “0” indicates malignant for the Class variable). Compute the number of errors of both types that will be made if this method is applied to the current data with the reduced model. Also, compute the percentage of classifications that result in each kind of error (that is, compute the error rates- false positive and false negative).

```
> table1<-predict(reduced, type="response")
> table(wbca$Class, 1*table1 > .5)

   FALSE TRUE
0   227   11
1     9  434
```

That is, the number of malignant tumors that were falsely categorized as benign was 9. The number of benign tumors falsely categorized as malignant was 11.

Accuracy = (227 + 434) / 681 = .97063
False positive rate = 11 / 238 = 0.0462 (I’m assuming “positive” means malignant)
False negative rate = 9 / 443 = 0.0203

(e) Suppose we move the cut-off point to 0.9 so that p < 0.9 indicates malignant and p > 0.9 indicates benign. What are the new error counts and rates?

```
> table(wbca$Class, 1*table1 > .9)

    FALSE TRUE
  0   237    1
  1    16  427
```

Accuracy = (237 + 427) / 681 = 0.97503
False positive rate = 1 / 238 = 0.004202 (I'm assuming "positive" means malignant)
False negative rate = 16 / 443 = 0.036117

The false positive rate has decreased, but the false negative rate has increased. I would prefer to see the false negative rate go down.

(f) It can be misleading to use the same data to fit a model and test its predictive ability. So split the original dataset into two parts: assign every third record to the test set and all the others to the training set. Use the training set to determine a good model (repeat parts (a) and (b)). Then use the test set to assess predictive performance (repeat parts (d) and (e)).

```
> Obs3rd<-which((1:681)%%3 ==0)
> test<-wbca[Obs3rd,]
> training <- wbca[-Obs3rd,]
> modelTrain<-glm(Class~.,data=training, family=binomial(logit))
> summary(modelTrain)

Call:
glm(formula = Class ~ ., family = binomial(logit), data = training)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.98138  -0.00954   0.03310   0.07084   3.07275

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  12.0244     2.0462   5.876 4.19e-09 ***
Adhes        -0.4859     0.1555  -3.126  0.00177 **
BNucl        -0.3732     0.1292  -2.888  0.00388 **
Chrom        -0.6655     0.2536  -2.625  0.00868 **
Epith         0.1779     0.2148   0.828  0.40744
Mitos        -0.6075     0.5103  -1.190  0.23388
NNucl        -0.5168     0.1828  -2.828  0.00469 **
Thick        -0.6533     0.2044  -3.197  0.00139 **
UShap        -0.5291     0.2612  -2.026  0.04280 *
```

```
USize        0.2672     0.2320   1.152  0.24947
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 592.796  on 453  degrees of freedom
Residual deviance:  57.651  on 444  degrees of freedom
AIC: 77.651

Number of Fisher Scoring iterations: 9


> pchisq(deviance(modelTrain), df.residual(modelTrain), lower=FALSE)
[1] 1

> redTrain<-step(modelTrain)
Start:  AIC=77.65
Class ~ Adhes + BNucl + Chrom + Epith + Mitos + NNucl + Thick +
    UShap + USize

        Df Deviance    AIC
- Epith  1   58.340 76.340
- USize  1   58.880 76.880
<none>       57.651 77.651
- Mitos  1   60.712 78.712
- UShap  1   61.450 79.450
- Chrom  1   65.983 83.983
- BNucl  1   67.373 85.373
- NNucl  1   67.538 85.538
- Adhes  1   68.073 86.073
- Thick  1   71.162 89.162

Step:  AIC=76.34
Class ~ Adhes + BNucl + Chrom + Mitos + NNucl + Thick + UShap +
    USize

        Df Deviance    AIC
- USize  1   59.536 75.536
<none>       58.340 76.340
- Mitos  1   61.264 77.264
- UShap  1   61.702 77.702
- Chrom  1   66.515 82.515
- BNucl  1   67.402 83.402
- NNucl  1   67.556 83.556
- Adhes  1   68.310 84.310
- Thick  1   72.311 88.311
```

```
Step:  AIC=75.54
Class ~ Adhes + BNucl + Chrom + Mitos + NNucl + Thick + UShap

       Df Deviance    AIC
<none>       59.536 75.536
- UShap  1   61.894 75.894
- Mitos  1   62.329 76.329
- Chrom  1   66.762 80.762
- NNucl  1   67.576 81.576
- BNucl  1   68.332 82.332
- Adhes  1   68.359 82.359
- Thick  1   72.363 86.363
>

> pchisq(deviance(redTrain), df.residual(redTrain), lower=FALSE)
[1] 1

> summary(redTrain)

Call:
glm(formula = Class ~ Adhes + BNucl + Chrom + Mitos + NNucl +
    Thick + UShap, family = binomial(logit), data = training)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.03312  -0.01224   0.04042   0.08373   2.85056

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  11.5571     1.8285   6.321  2.6e-10 ***
Adhes        -0.4249     0.1441  -2.949  0.00318 **
BNucl        -0.3341     0.1187  -2.815  0.00487 **
Chrom        -0.5963     0.2422  -2.462  0.01382 *
Mitos        -0.5822     0.4872  -1.195  0.23207
NNucl        -0.4192     0.1604  -2.614  0.00895 **
Thick        -0.6037     0.1924  -3.138  0.00170 **
UShap        -0.2943     0.2034  -1.447  0.14795
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 592.796  on 453  degrees of freedom
Residual deviance:  59.536  on 446  degrees of freedom
AIC: 75.536
```

```
Number of Fisher Scoring iterations: 9


> table2 <- predict(redTrain, newdata = test, type = "response", se = T)
> table(test$Class, 1*(table2$fit > .9))

     0   1
  0  73   2
  1   3 149
```

Accuracy = (73 + 149) / (73+2+3+149) = 0.977974
False positive rate = 2/75=0.0266667
False negative rate = 3/152 = 0.01974

(g) Discuss how you could search for the "best" cut-off point to use for classifying tumors. Then write an R program to carry it out. What is the "best" cut-off value? What do you mean by "best"?

First, you would essentially want to determine which of the classification rates is most important. In the case of cancer, I think the false negative rate is. So what one could do is build a bunch of models for varying values of the cut-off (say, 0.01, 0.02, 0.03, …, 0.98, 0.99) on the training set. Then for each one, evaluate the false negative rate. Choose the value of the cut-off that minimizes the false negative rate.

Of course, the hospital administrators and insurance providers might be more concerned with the false positive rate, because if a test is positive, then maybe further testing (which is expensive) is required… They might arrive at a different model… or perhaps some weighting of the two error rates would be appropriate….