Project One for Statistical Data Mining

Raymond Anthony Ford[*]

Due: 24 September 2018

Contents

1 Exploratory data analysis

We first begin the project by bringing the data in; the data is available in the `mlbench` package.

```
suppressMessages(require("mlbench"))
data(BreastCancer, package="mlbench")
dat <- BreastCancer
```

We next need to inspect the data. In particular, variable types, missing values, and the number of unique values.

```
n <- nrow(dat)
out <- NULL
for (k in 1:ncol(dat)) {
    vname <- colnames(dat)[k]
    x <- as.vector(dat[,k])
    n1 <- sum(is.na(x), na.rm=TRUE)
    n2 <- sum(x=="NA", na.rm=TRUE)
    n3 <- sum(x=='', na.rm=TRUE)
    n4 <- sum(x=="?", na.rm=TRUE)
    n5 <- sum(x=="*", na.rm=TRUE)
    n6 <- sum(x==".", na.rm=TRUE)
    nmiss <- n1 + n2 + n3 + n4 + n5 + n6
    ncomplete <- n - nmiss
    var.type <- typeof(x)
    if (var.type == "integer") {
```

[*]raford2@miners.utep.edu

```
        if (length(unique(x)) == 2) {
            out <- rbind(out, c(col.number=k, vname=vname, mode="binary",
                                n.levels=length(unique(x)), ncomplete=ncomplete,
                                miss.prop=round(nmiss/n, digits=4)))
        } else {
            out <- rbind(out, c(col.number=k, vname=vname, mode=typeof(x),
                                n.levels=length(unique(x)), ncomplete=ncomplete,
                                miss.prop=round(nmiss/n, digits=4)))
        }
    } else {
        out <- rbind(out, c(col.number=k, vname=vname, mode=typeof(x),
                            n.levels=length(unique(x)), ncomplete=ncomplete,
                            miss.prop=round(nmiss/n, digits=4)))
    }
}
out <- as.data.frame(out)
row.names(out) <- NULL
out
```

```
##    col.number           vname      mode n.levels ncomplete miss.prop
## 1           1              Id character      645       699         0
## 2           2    Cl.thickness character       10       699         0
## 3           3       Cell.size character       10       699         0
## 4           4      Cell.shape character       10       699         0
## 5           5   Marg.adhesion character       10       699         0
## 6           6    Epith.c.size character       10       699         0
## 7           7     Bare.nuclei character       11       683    0.0229
## 8           8     Bl.cromatin character       10       699         0
## 9           9 Normal.nucleoli character       10       699         0
## 10         10         Mitoses character        9       699         0
## 11         11           Class character        2       699         0
```

From the above outout, we notice a few problems with our data. We correct these issues in the next few steps and then perform some exploratory data analysis.

We note that one of the variables is the patient's ID. Since this is not relevant to the task at hand, we will remove it.

```
dat <- dat[, -1]
```

Next we need to handle our missing values. We will perform missing value imputation using MICE.

```
suppressMessages(library(mice))
fit.mice <- mice(dat, m=1, maxit=50, method='pmm', seed=5474, printFlag=FALSE)
dat <- complete(fit.mice, 1)
```

Since most of our variables are coded as characters—while they should be numeric—we recode all except the target variable as numeric.

```
for (i in 1:9) {
    dat[, i] <- as.numeric(as.character(dat[, i]))
}
```

Finally, we recode our target variable `class` to be either zero or one.

```
dat$Class <- ifelse(dat$Class == "malignant", 1, 0)
```

Since we are performing classification, we need to know if the class memberships are severely imbalanced. Will consider severely imbalanced to be any proportion of the minority class to be less than 0.07.

```
mean(dat$Class)
```

```
## [1] 0.3447783
```

```
dat$Class <- factor(dat$Class)
```

We see from the above output that class membership amongst the response variable is not severely imbalaced, thus remedial measures do not need to be applied to the data.

We can also test the association between `Class` and a few predictors, as shown below.

```
# Perform and print results for a chi-squared test on each variable
sig.vars <- c(8, 9)
for (k in sig.vars) {
    vname <- colnames(dat)[k]
    print(k)
    print(vname)
    x <- as.vector(dat[,k])
    tab <- table(dat$Class, x, useNA='no')
    print(tab)
    print(chisq.test(tab))
}
```

```
## [1] 8
## [1] "Normal.nucleoli"
##    x
##       1   2   3   4   5   6   7   8   9  10
##   0 402  30  12   1   2   4   2   4   1   0
##   1  41   6  32  17  17  18  14  20  15  61
##
##  Pearson's Chi-squared test
##
## data:  tab
## X-squared = 420.3, df = 9, p-value < 2.2e-16
```

```
##
## [1] 9
## [1] "Mitoses"
##    x
##       1   2   3   4   5   6   7   8  10
##   0 445   8   2   0   1   0   1   1   0
##   1 134  27  31  12   5   3   8   7  14
```

```
## Warning in chisq.test(tab): Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab
## X-squared = 195.98, df = 8, p-value < 2.2e-16
```

From the above output we see that the association between `Class` and `Normal.nucleoli`, and `Class` and `Mitoses` are significant at any resonable level of significance.

2 Logistic regression

We next split our data into a training, validation, and testing datasets with a ratio of 2:1:1, respectively.

```
set.seed(5494)
n <- NROW(dat)
id.split <- sample(x=1:3, size=n, replace=TRUE, prob=c(0.5, 0.25, 0.25))
dat.train <- dat[id.split==1, ]
dat.valid <- dat[id.split==2, ]
dat.test <- dat[id.split==3, ]
```

Next we build a logistic regression model using the training data and prepare our validation data set to assist us in identifying the best lambda.

```
suppressMessages(require("glmnet"))
formula0 <- Class ~ Cl.thickness + Cell.size + Cell.shape + Marg.adhesion +
    Epith.c.size + Bare.nuclei + Bl.cromatin + Normal.nucleoli + Mitoses
X <- model.matrix(as.formula(formula0), data=dat.train)
y <- dat.train$Class
X.valid <- model.matrix(as.formula(formula0), data=dat.valid)
y.valid <- dat.valid$Class
```

Now we use the validation data help us find the best lambda for a LASSO logistic regression model.

```
Lambda <- seq(0.0001, 0.5, length.out = 500)
L <- length(Lambda)
OUT <- matrix(0, L, 2)
for (i in 1:L) {
    fit <- glmnet(x=X, y=y, family="binomial", alpha=1, lambda=Lambda[i],
                  standardize=TRUE, thresh=1e-07, maxit=1000)
    pred <- predict(fit, newx=X.valid, s=Lambda[i], type="response")
    mse <- mean((as.numeric(y.valid)-pred)**2)
    OUT[i, ] <- c(Lambda[i], mse)
}
```

We use the following code to help us identify which value of lambda yields the smallest mean square error.

```
lam <- which.min(OUT[ ,2])
lambda.best <- OUT[lam, 1]
lambda.best
```

```
## [1] 0.04618297
```

We next make a plot showing the relationship between the many values of lambda available and their effect on mean square error.

```
plot(OUT[,1], OUT[,2], type="l", col="orange", lwd=4, xlab="lambda",
     ylab="Mean Square Error",
     main="Plot of MSE vs lambda from validation data")
abline(v=OUT[lam,1], col="blue", lty=2, lwd=2)
```

Next, we will use the best value of lambda to construct a LASSO logistic regression model both our training and validation datasets.

```
X.d1d2 <- rbind(dat.train, dat.valid)
X <- model.matrix(as.formula(formula0), data=X.d1d2)
y <- X.d1d2$Class
fit.best <- glmnet(x=X, y=y, family="binomial", alpha=1, lambda=lambda.best,
                   standardize=TRUE, thresh=1e-07, maxit=1000)
coef(fit.best)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)     -4.47945108
## (Intercept)      .
## Cl.thickness     0.20613065
## Cell.size        0.12217438
## Cell.shape       0.12190784
## Marg.adhesion    0.03985213
## Epith.c.size     .
```

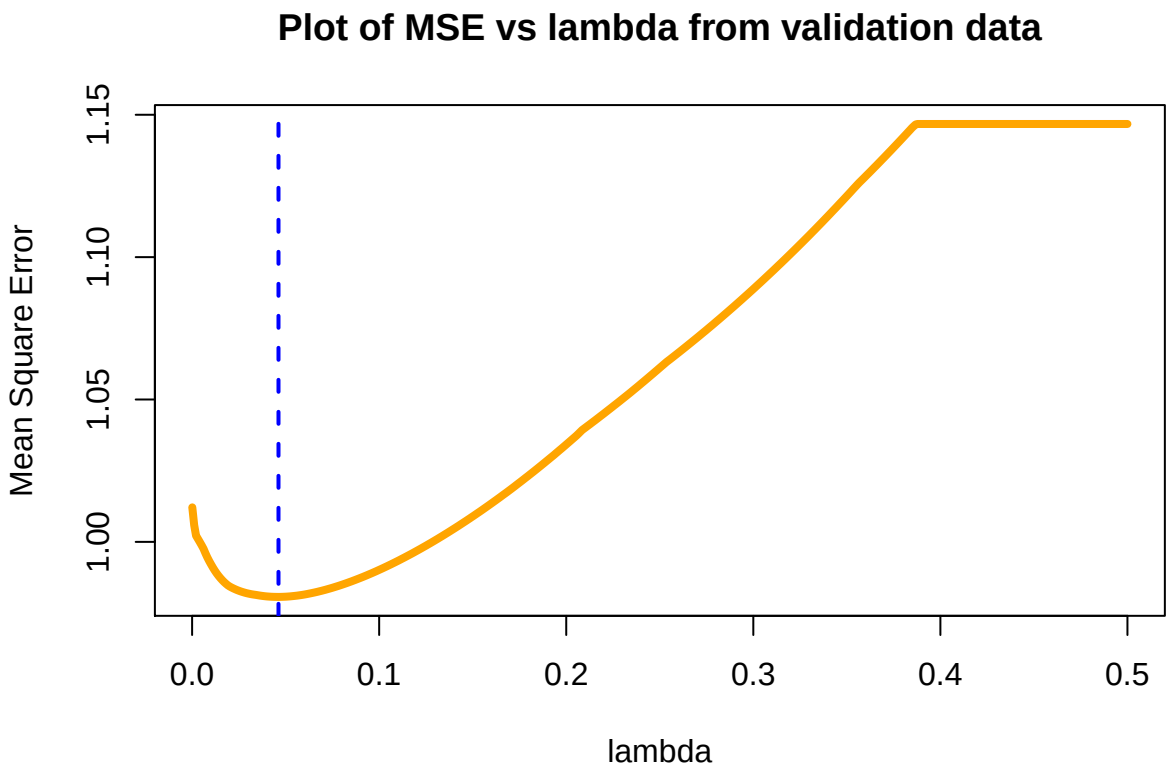

Figure 1: The best lambda for the lowest MSE

```
## Bare.nuclei      0.24421674
## Bl.cromatin      0.14872798
## Normal.nucleoli  0.12126497
## Mitoses          .
```

From above, we see that all pretictors except `Epith.c.size` and `Mitoses` are important predictors.

Finall we apply the final logistic model to the test data and present the ROC curve.

```
suppressMessages(suppressWarnings(library(pROC, quietly=TRUE)))
X.test <- model.matrix(as.formula(formula0), data=dat.test)
pred.fit <- predict(fit.best, newx=X.test, s=lambda.best, type="response")
par(pty="s")
plot.roc(dat.test$Class, pred.fit, main="ROC curve for the best model", percent=TRUE, pr
```

```
## Warning in roc.default(x, predictor, plot = TRUE, ...): Deprecated use
## a matrix as predictor. Unexpected results may be produced, please pass a
## numeric vector.
```

In Figure 2, we see that we obtained an AUC of 99.1%. This high of an AUC came as a surprise for me.

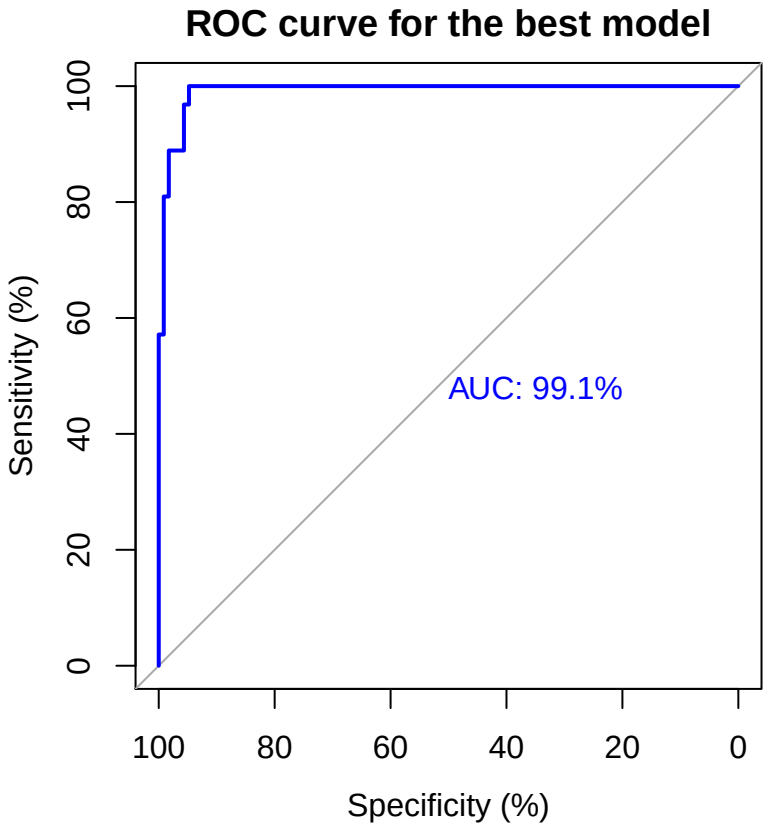

Figure 2: ROC curve for the model using the testing data