Project Six for Data Mining

Raymond Anthony Ford
raford2@miners.utep.edu

Due: 01 May 2018

Contents

1 Data cleaning

We begin this project by bringing the data into R.

```
dat <- read.csv("~/Dropbox/Spring2018/STAT5474/ILPD.csv", header=FALSE,
                sep=",", col.names=c("age", "gender", "TB", "DB", "alkphos",
                "sgpt", "sgot","TP", "alb", "AGratio", "liver"))
```

A quick inspection of the data shows that our target variable `liver` is coded as either a one or two, and that `gender` is coded as either male or female. In order to proceed, we recode these variables so that one means that the person has liver disease (zero means that they do not) and one means that the person is male (zero means that the person is female). We then print the mean for `dat$liver` to determine the proportion of subjects diagnosed with liver disease.

```
dat$liver <- as.integer(ifelse(dat$liver==1, 1, 0))
dat$gender <- as.integer(ifelse(dat$gender=="Male", 1, 0))
mean(dat$liver)
```

```
## [1] 0.7135506
```

From the above output we that 71% of the subjects were diagnosed with liver disease. We do not believe that this is anywhere near the real prevalence rate of liver disease in the general population, as it is fairly high.

Next we need to check for missing values in our data.

```
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)
    nmiss <- n1 + n2 + n3
    ncomplete <- n - nmiss
    var.type <- typeof(x)
    if (var.type == "integer") {
        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     age integer       72       583         0
## 2           2  gender  binary        2       583         0
## 3           3      TB  double      113       583         0
## 4           4      DB  double       80       583         0
## 5           5 alkphos integer      263       583         0
## 6           6    sgpt integer      152       583         0
## 7           7    sgot integer      177       583         0
## 8           8      TP  double       58       583         0
```

```
## 9           9     alb  double       40       583         0
## 10         10 AGratio  double       70       579    0.0069
## 11         11   liver  binary        2       583         0
```

From the above output, we see that `AGratio` has some missing values. We will impute these missing the mice function from mice.

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

2 EDA and variable screening

From the output in the previous section, we know that `age`, `alkphos`, `sgpt`, and `sgot` are integers, `gender` is categorical, `TB`, `DB`, `TP`, `alb`, and `AGratio` are continuous.

Now, we will perform some variable screening. For each categorical predictor we use a χ^2 test of independence to assess its association with `liver`, and for all other predictors we will use a two sample t-test. We use a threshold significance level of $\alpha = 0.20$ and remove all predictors having a p-value greater than that.

```
library(car, quietly=TRUE)
vars.nominal <- c("gender")
cols.x <- 1:(NCOL(dat)-1)
xnames <- names(dat)[cols.x]
y <- dat$liver
OUT <- NULL
for (j in 1:length(cols.x)){
    x <- dat[, cols.x[j]]
    xname <- xnames[j]
    if (is.element(xname, vars.nominal)){
        tbl <- table(x, y)
        pvalue <- chisq.test(tbl)$p.value
    } else {
        # TWO-SAMPLE t-test
        pvalue.equal.var <- (leveneTest(x~factor(y))$"Pr(>F)")[1]
        equal.var <- ifelse(pvalue.equal.var <= 0.05, FALSE, TRUE)
        pvalue <- t.test(x~y, alternative="two.sided",
                         var.equal=equal.var)$p.value
    }
    OUT <- rbind(OUT, cbind(xname=xname, pvalue=pvalue))
}
OUT <- as.data.frame(OUT)
colnames(OUT) <- c("name", "pvalue")
OUT
```

```
##       name                pvalue
## 1      age 0.000884063155626139
## 2   gender   0.0596658468577747
## 3       TB 4.91200919556184e-16
## 4       DB 2.26995324945029e-19
## 5  alkphos 1.08124966726932e-08
## 6     sgpt 1.18047797924202e-09
## 7     sgot 1.40944977692876e-08
## 8       TP    0.398819127523851
## 9      alb 9.07436084295548e-05
## 10 AGratio 6.39088236401563e-05
```

From the above output, we see that only one predictor needs to be removed, namely `TP`, so we remove it with the following code.

```
OUT$pvalue <- as.character(OUT$pvalue)
OUT$pvalue <- as.numeric(OUT$pvalue)
non.sig <- which(OUT$pvalue > 0.2)
dat <- dat[,-c(non.sig)]
```

3 Model building

We will fit three different logistic regression models and compare their performance using the area under the curve (AUC) on a receiver operating characteristic (ROC) curve.

3.1 Full model

The first model that we fit is nothing more than a logistic regression model containing all of the predictors.

```
fit.full <- suppressWarnings(glm(liver~., family=binomial, data=dat))
```

The following output shows all the predictors in the model, and their slope parameter estimates.

```
coef(fit.full)
```

```
##  (Intercept)          age       gender           TB           DB
## -0.861893318  0.018587980  0.038548650  0.008389071  0.506690771
##      alkphos         sgpt         sgot          alb      AGratio
##  0.001277385  0.009934171  0.003333270  0.029102509 -0.537766290
```

3.2 Stepwise variable selection

The second model that we fit has it's predictors chosen via stepwise variable selection starting with the full model. The best model is determined based on its BIC.

```
fit.step <- suppressWarnings(step(fit.full, direction = "both",
                                  k=log(nrow(dat)), trace=FALSE))
```

The following output shows all the predictors selected from the stepwise variable selection to be used in the model, and their slope parameter estimates.

```
coef(fit.step)
```

```
## (Intercept)         age          DB        sgpt
## -1.12264221  0.01993645  0.65782404  0.01509633
```

3.3 LASSO

Finally, the third model that we fit is a LASSO model. Since LASSO requires us to find a λ, we accomplish this task via crossvalidation. We select the smallest λ and use this for our model.

```
suppressMessages(library(glmnet, quietly=TRUE))
X <- model.matrix(~., data=dat)
X <- X[,-c(1,11)]
y <- dat$liver
CV <- cv.glmnet(x=X, y=y, family="binomial", alpha = 1,
    lambda.min = 1e-4, nlambda = 200, standardize = T, thresh = 1e-07,
    maxit=1000)
b.lambda <- CV$lambda.min
fit.pen <- glmnet(x=X, y=y, family="binomial", alpha = 1,
    lambda=b.lambda, standardize = T, thresh = 1e-07,
    maxit=1000)
```

The following output shows all the predictors selected from the regularization to be used in the model, and their slope parameter estimates.

```
t(as.matrix(coef(fit.pen)))
```

```
##    (Intercept)         age      gender         TB         DB      alkphos
## s0  -0.6188795 0.01704721 0.04740395 0.0029395 0.4483932 0.001308496
##          sgpt        sgot alb    AGratio
## s0 0.00780689 0.002113851   0 -0.4624839
```

4 Model comparison

Next, we compute the jackknife residuals for each of the models from the previous section.

```
suppressMessages(library(pROC, quietly=TRUE))
```

```
## Warning: package 'pROC' was built under R version 3.4.4
```

```
# Compute jackknife values for the full model
n <- NROW(dat)
p.jk <- rep(0, n)
for (i in 1:n){
    fit.i <- suppressWarnings(glm(formula(fit.full), data=dat[-i,],
                                  family = "binomial"))
    p.jk[i] <- predict(fit.i, newdata=dat[i,], type="response")
}
y <- dat$liver
yhat.full <- p.jk

# Compute jackknife values for the step model
n <- NROW(dat)
p.jk <- rep(0, n)
for (i in 1:n){
    fit.i <- suppressWarnings(glm(formula(fit.step), data=dat[-i,],
                                  family = "binomial"))
    p.jk[i] <- predict(fit.i, newdata=dat[i,], type="response")
}
y <- dat$liver
yhat.step <- p.jk

# Compute jackknife values for LASSO
terms <- colnames(X)[as.vector(coef(fit.pen)) != "."]
formula.pen <- as.formula(paste(c("liver ~ 1", terms[1:length(terms)-1]),
                                collapse=" + "))
n <-  nrow(dat)
p.jk <- rep(0, n)
for (i in 1:n) {
    fit.i <- suppressWarnings(glm(formula.pen, data=dat[-i,], family = "binomial"))
    p.jk[i] <- predict(fit.i, newdata=dat[i,], type="response")
}
y <- dat$liver
yhat.pen <- p.jk
```

We next plot the ROC curves for each of the models.

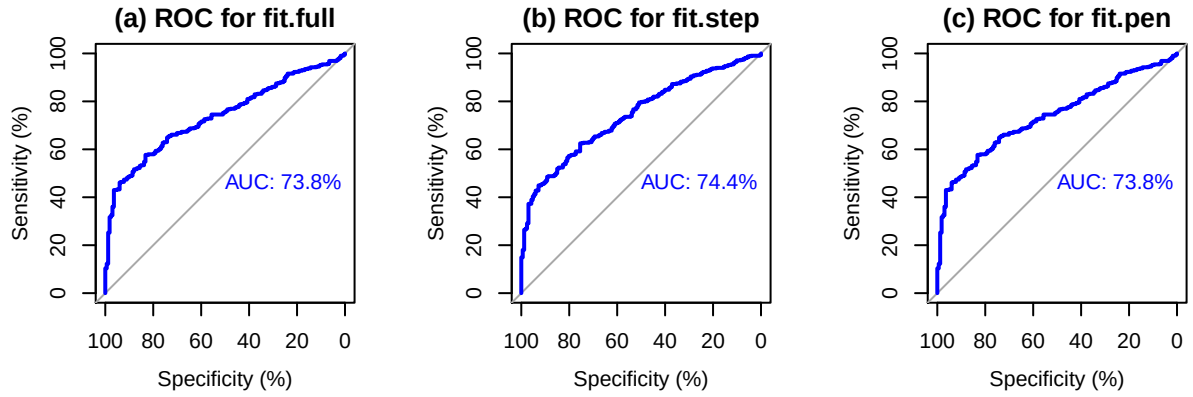

Figure 1: ROC curves for each of the logistic regression models fit.

```
par(mfrow=c(1, 3), mar=rep(4,4), pty="s")
plot.roc(y, yhat.full,  main="(a) ROC for fit.full", percent=TRUE,
    print.auc=TRUE, print.auc.cex=1.0, col="blue", xlim=c(100, 0))
plot.roc(y, yhat.step,  main="(b) ROC for fit.step", percent=TRUE,
    print.auc=TRUE, print.auc.cex=1.0, col="blue")
plot.roc(y, yhat.pen,  main="(c) ROC for fit.pen", percent=TRUE,
    print.auc=TRUE, print.auc.cex=1.0, col="blue")
```

In Figure 1, we see that model fitted using stepwise selection (b) has the highest AUC compared to the other methods with its AUC being 74.4%, compared to the AUC of 73.8% obtained with both the full model and LASSO.

5 Final model

Since the model fitted using stepwise selection has the highest AUC, we will use this as our final model. We next compute, and output, the 95% confidence intervals for the odds ratios from this final model.

```
ci <- suppressWarnings(suppressMessages(confint(fit.step, level = 0.95)))
exp(ci)  # CI FOR OR
```

```
##                 2.5 %    97.5 %
## (Intercept) 0.1698137 0.6127237
## age         1.0080930 1.0326225
## DB          1.4296011 2.8357334
## sgpt        1.0082985 1.0234853
```

The first thing that we notice is that none of these intervals contain any values less than one. This is important in this context because odds ratios less than one would imply that the predictor is a protective factor, but since we do not see this with our predictors—they all contain values greater than one—we conclude that they are all risk factors.

Looking specifically at `DB` (direct bilirubin), we see that it appears to be a significant risk factor relative to `age` and `sgp` because it's odds ratio is somewhere in the interval (1.43, 2.84), as oppossed to being relatively close to one. On the lower end of the interval there is a 1.43 relative increase in the odds of disease, and on the upper end there is 2.84 relative increase in the odds of disease. Checking the measurements of `DB` in data set the confirms that there are some individuals that have high `DB` levels, and many have measurements that are well above normal.