Feature Selection

With the Caret Package

and the Pima data set

Karen Mazidi

The Caret (classification and regression training) Package contains various functions for training and plotting classification and regression models. The documentation for this package is about 200 pages but we're just going to look at some funtions to help identify important predictors.

Pima

The Pima.tr (train) data set contains 200 observations about women of Pima Indian heritage. Column *type* is a Yes/No factor indicating a diabetes diagnosis. The other 7 columns are numeric predictors: number of pregnancies, glucose, blood pressure, skin thickness on arm, bmi, family history (ped), and age.

The test set has 332 observations.

```
library(MASS)
str(Pima.tr)
```

```
## 'data.frame':    200 obs. of  8 variables:
##  $ npreg: int  5 7 5 0 0 5 3 1 3 2 ...
##  $ glu  : int  86 195 77 165 107 97 83 193 142 128 ...
##  $ bp   : int  68 70 82 76 60 76 58 50 80 78 ...
##  $ skin : int  28 33 41 43 25 27 31 16 15 37 ...
##  $ bmi  : num  30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...
##  $ ped  : num  0.364 0.163 0.156 0.259 0.133 ...
##  $ age  : int  24 55 35 26 23 52 25 24 63 31 ...
##  $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
```

```
dim(Pima.te)
```

```
## [1] 332   8
```

Naive Bayes

First we try a naive bayes model on the data. We get 75.6% accuracy which is not bad.

```
library(e1071)
nb1 <- naiveBayes(Pima.tr[,-8], Pima.tr[,8], data=Pima.tr)
summary(nb1)
```

```
##         Length Class  Mode
## apriori 2      table  numeric
## tables  7      -none- list
## levels  2      -none- character
## call    4      -none- call
```

```
pred <- predict(nb1, newdata=Pima.te[,-8], type="class")
table(pred, Pima.te$type)
```

```
##
## pred   No Yes
##   No  185  43
##   Yes  38  66
```

```
acc1 <- mean(pred==Pima.te$type)
acc1
```

```
## [1] 0.7560241
```

Look for highly correlated predictors

Next we use the caret package to find highly correlated predictors. Column 4 (skin) was highly correlated wtih column 5 (bmi), which makes sense: if you are obese you probably can pinch more under the arms. The correlation there was about 0.66. It also flagged column 7 (age) as being highly correlated with column 1 (number of pregnancies). It does stand to reason that as time goes by, women have more pregnancies. The correlation was about 0.6.

```
set.seed(1234)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
corMatrix <- cor(Pima.tr[,1:7])
corMatrix
```

```
##             npreg        glu          bp      skin        bmi         ped
## npreg  1.00000000 0.17052466  0.25206108 0.1090493 0.05833606 -0.11947275
## glu    0.17052466 1.00000000  0.26938132 0.2175965 0.21678987  0.06071011
## bp     0.25206108 0.26938132  1.00000000 0.2649635 0.23882134 -0.04739958
## skin   0.10904927 0.21759652  0.26496347 1.0000000 0.65903563  0.09540270
## bmi    0.05833606 0.21678987  0.23882134 0.6590356 1.00000000  0.19055065
## ped   -0.11947275 0.06071011 -0.04739958 0.0954027 0.19055065  1.00000000
## age    0.59892236 0.34340695  0.39107336 0.2519257 0.13191956 -0.07140965
##               age
## npreg  0.59892236
## glu    0.34340695
## bp     0.39107336
## skin   0.25192574
## bmi    0.13191956
## ped   -0.07140965
## age    1.00000000
```

```
findCorrelation(corMatrix, cutoff=0.5, verbose=TRUE)
```

```
## Compare row 7  and column  1 with corr  0.599
##   Means:  0.298 vs 0.228 so flagging column 7
## Compare row 4  and column  5 with corr  0.659
##   Means:  0.269 vs 0.183 so flagging column 4
## All correlations <= 0.5
```

```
## [1] 7 4
```

Naive Bayes model 2

Next we build another naive bayes model, this time omitting the highly correlated predictors.

It seems that this improved our accuracy from .756 to .771

```
nb2 <- naiveBayes(Pima.tr[,-c(1,3,8)], Pima.tr[,8], data=Pima.tr) # omit npreg, skin
pred <- predict(nb2, newdata=Pima.te[,-c(1,3,8)], type="class")
table(pred, Pima.te$type)
```

```
##
## pred   No Yes
##   No  194  47
##   Yes  29  62
```

```
mean(pred==Pima.te$type)
```

```
## [1] 0.7710843
```

Rank features by importance

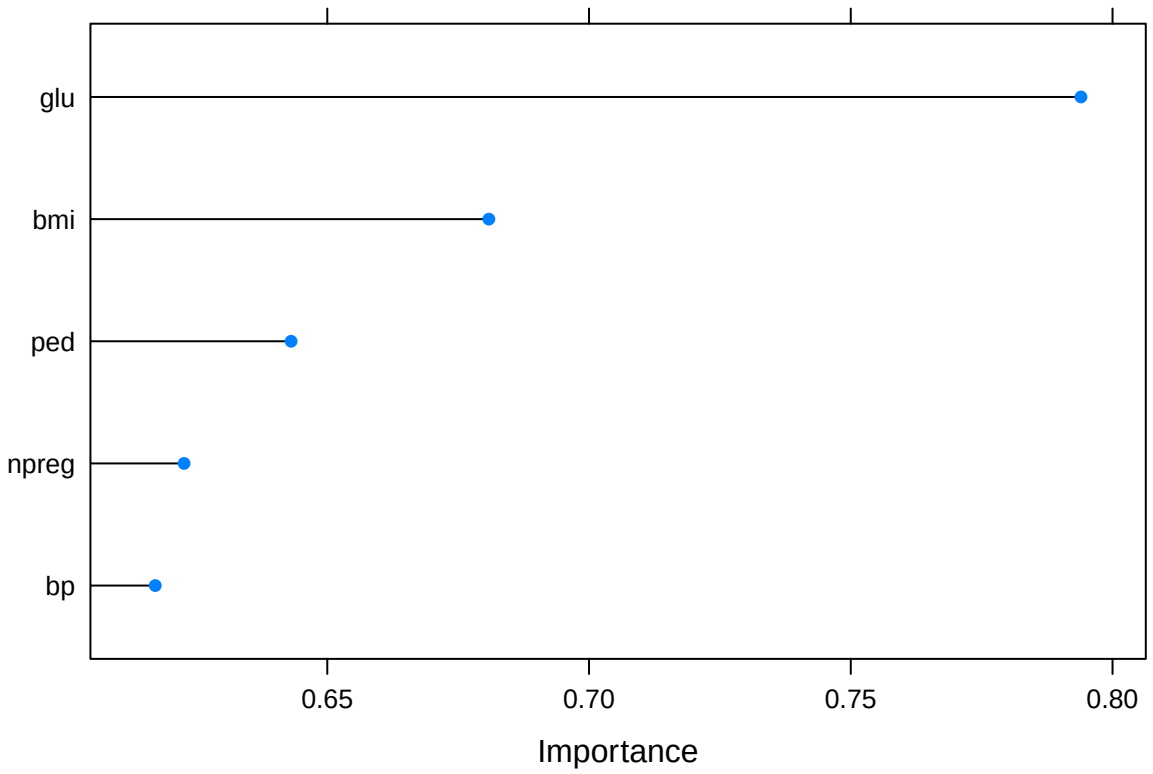
Caret also has functions to rank features by importance.

First, some control parameters are stored in variable ctrl. Next we combined the train and test sets with rbind to give the function more data. We also omitted the highly correlated predictors identified in the previous step. The varImp function gives us the variables, ranked by importance, which we can also plot. The train() method below used knn to learn feature importance.

```
ctrl <- trainControl(method="repeatedcv", repeats=5)
Pima <- rbind(Pima.tr, Pima.te) # get all the data
Pima <- Pima[,-c(4,7)]  # omit highly correlated predictors
model <- train(type~., data=Pima, method="knn", preProcess="scale", trControl=ctrl)
importance <- varImp(model, scale=FALSE)
importance
```

```
## ROC curve variable importance
##
##       Importance
## glu       0.7940
## bmi       0.6809
## ped       0.6431
## npreg     0.6226
## bp        0.6171
```

```
plot(importance)
```



Recursive Feature Selection

Another Caret function is rfe - recursive feature selection, which selects predictors based on importance ranking. It can be used to find a subset of features that are good predictors. The rfe() function below used cross validation.

```
ctrl <- rfeControl(functions=rfFuncs, method="cv", number=10)
Pima <- rbind(Pima.tr, Pima.te)
rfe_out <- rfe(Pima[,1:7], Pima[,8], sizes=c(1:7), rfeControl=ctrl)
rfe_out
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.7407 0.3831    0.08333  0.2020
##          2   0.7459 0.4138    0.05321  0.1234
##          3   0.7702 0.4651    0.06828  0.1659
##          4   0.7818 0.4884    0.05599  0.1411
##          5   0.7819 0.4903    0.05958  0.1500        *
##          6   0.7744 0.4714    0.05543  0.1393
##          7   0.7631 0.4426    0.05923  0.1426
##
## The top 5 variables (out of 5):
##    glu, age, bmi, npreg, ped
```

```
predictors(rfe_out)
```

```
## [1] "glu"   "age"   "bmi"   "npreg" "ped"
```

Naive Bayes model 3

Next we build another naive bayes model, this time using the predictors

It seems that this improved our accuracy to 0.783, the highest of the 3 models. This represents a nearly 4% improvement over model 1.

```
nb3 <- naiveBayes(Pima.tr[,-c(3, 4, 8)], Pima.tr[,8], data=Pima.tr) # omit bp, skin
pred <- predict(nb3, newdata=Pima.te[,-c(3,4,8)], type="class")
table(pred, Pima.te$type)
```

```
##
## pred   No Yes
##   No  193  42
##   Yes  30  67
```

```
acc3 <- mean(pred==Pima.te$type)
acc3
```

```
## [1] 0.7831325
```

```
(acc3 - acc1)/acc1  # improvement
```

```
## [1] 0.03585657
```

Logistic Regression

The caret package proved helpful for selecting important features for the Naive Bayes model, but other algorithms in R include feature ranking as part of the function. As an example, we next build a logistic regression model.

The logistic regression model got a higher accuracy than the naive bayes model 3. Further, it indicates that glucose was the most important feature, followed by ped, and then age and bmi.

```
glm1 <- glm(type~., data=Pima.tr, family="binomial")
summary(glm1)
```

```
##
## Call:
## glm(formula = type ~ ., family = "binomial", data = Pima.tr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9830  -0.6773  -0.3681   0.6439   2.3154
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.773062   1.770386  -5.520 3.38e-08 ***
## npreg        0.103183   0.064694   1.595  0.11073
## glu          0.032117   0.006787   4.732 2.22e-06 ***
## bp          -0.004768   0.018541  -0.257  0.79707
## skin        -0.001917   0.022500  -0.085  0.93211
```

```
## bmi          0.083624   0.042827   1.953  0.05087 .
## ped          1.820410   0.665514   2.735  0.00623 **
## age          0.041184   0.022091   1.864  0.06228 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 178.39  on 192  degrees of freedom
## AIC: 194.39
##
## Number of Fisher Scoring iterations: 5
```

```
probs <- predict(glm1, newdata=Pima.te, type='response')
pred <- ifelse(probs>0.5, "Yes", "No")
table(predicted=pred, actual=Pima.te$type)
```

```
##          actual
## predicted  No Yes
##       No  200  43
##       Yes  23  66
```

```
mean(pred==Pima.te$type)
```

```
## [1] 0.8012048
```

Logistic Regression Model 2

Let's build another logistic regression model, using the same predictors as for naive bayes model 3.

That didn't improve the model. For logistic regression, the built-in feature selection works fine.

```
glm2 <- glm(type~npreg+glu+bmi+ped+age, data=Pima.tr, family="binomial")
summary(glm2)
```

```
##
## Call:
## glm(formula = type ~ npreg + glu + bmi + ped + age, family = "binomial",
##     data = Pima.tr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0009  -0.6816  -0.3664   0.6467   2.2898
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.938059   1.541571  -6.447 1.14e-10 ***
## npreg        0.103142   0.064517   1.599  0.10989
## glu          0.031809   0.006667   4.771 1.83e-06 ***
## bmi          0.079672   0.032649   2.440  0.01468 *
## ped          1.811417   0.661048   2.740  0.00614 **
## age          0.039286   0.020967   1.874  0.06097 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 178.47  on 194  degrees of freedom
## AIC: 190.47
##
## Number of Fisher Scoring iterations: 5
```

```
probs <- predict(glm2, newdata=Pima.te, type='response')
pred <- ifelse(probs>0.5, "Yes", "No")
table(predicted=pred, actual=Pima.te$type)
```

```
##          actual
## predicted  No Yes
##       No  199  42
##       Yes  24  67
```

```
mean(pred==Pima.te$type)
```

```
## [1] 0.8012048
```

The take-away is that if your R algorithm includes feature ranking, then let it select the features it finds most helpful. If your R algorithm does not rank featues, then it may be helpful to use Caret to identify the best predictors. This could be particularly important if there are large numbers of columns in the data set.