Predictive Analytics for Qualitative Health Data

04/23/2021

Classification problem for health data

Classification is a supersived learning method for which the true class lables of a qualitative/categorical response variable are given in the training data. A number of supervised algorithms can be applied to a classification problem including: Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K- Nearest Neighbor (KNN) and logistic regression model.

Pima Indians Diabetes Example (source: kaggle.com)

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the study is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. All patients here are females at least 21 years old of Pima Indian heritage.

The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Goal:

Classify based on the diagnostic featurs whether or not the patients in the dataset have diabetes or not.

```
# Read the dataset
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
setwd("~/Box/MyDocs/Teaching/Fall/2020/MATH 624/LectureNotes/Week 4/Lecture")
pima = read_csv("PI_diabetes.csv")
```

```
## Parsed with column specification:
## cols(
##   Pregnancies = col_double(),
##   Glucose = col_double(),
##   BloodPressure = col_double(),
##   SkinThickness = col_double(),
```

```
##   Insulin = col_double(),
##   BMI = col_double(),
##   DiabetesPedigreeFunction = col_double(),
##   Age = col_double(),
##   Outcome = col_double()
## )
```

```
head(pima)
```

```
## # A tibble: 6 x 9
##   Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI DiabetesPedigre~
##         <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>            <dbl>
## 1           6     148            72            35       0  33.6            0.627
## 2           1      85            66            29       0  26.6            0.351
## 3           8     183            64             0       0  23.3            0.672
## 4           1      89            66            23      94  28.1            0.167
## 5           0     137            40            35     168  43.1            2.29
## 6           5     116            74             0       0  25.6            0.201
## # ... with 2 more variables: Age <dbl>, Outcome <dbl>
```

```
train=sample(768,500) # Randomly sample 500 items from 768 items
train.dat=pima[train,] # Select observations from the original dataset to #create a training dataset

test.dat = pima[-train, ]
```

Logistic Regression for classification problem

```
fit = glm(Outcome ~ Pregnancies+Glucose+BloodPressure+SkinThickness+Insulin+BMI+DiabetesPedigreeFunctio
summary(fit)
```

```
##
## Call:
## glm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
##     SkinThickness + Insulin + BMI + DiabetesPedigreeFunction +
##     Age, family = binomial, data = train.dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4754  -0.7482  -0.4416   0.7647   2.8372
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -7.793198   0.851278  -9.155  < 2e-16 ***
## Pregnancies               0.155168   0.042237   3.674 0.000239 ***
## Glucose                   0.033379   0.004411   7.567 3.81e-14 ***
## BloodPressure            -0.014472   0.006284  -2.303 0.021275 *
## SkinThickness            -0.005760   0.008742  -0.659 0.509945
## Insulin                  -0.001107   0.001132  -0.978 0.327966
## BMI                       0.098051   0.018453   5.313 1.08e-07 ***
## DiabetesPedigreeFunction  0.881434   0.371276   2.374 0.017593 *
## Age                       0.000745   0.011336   0.066 0.947602
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 652.26  on 499  degrees of freedom
## Residual deviance: 482.17  on 491  degrees of freedom
## AIC: 500.17
##
## Number of Fisher Scoring iterations: 5
```

You may drop SkinThickness, Insulin, and Age from the model, since these are not statistically significant.

```
fit1 = glm(Outcome ~ Pregnancies+Glucose+BloodPressure  +BMI+DiabetesPedigreeFunction, data=train.dat, f
summary(fit1)
```

```
##
## Call:
## glm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
##     BMI + DiabetesPedigreeFunction, family = binomial, data = train.dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6797  -0.7528  -0.4428   0.7494   2.8031
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -7.518306   0.804332  -9.347  < 2e-16 ***
## Pregnancies               0.165029   0.036454   4.527 5.98e-06 ***
## Glucose                   0.032070   0.003982   8.054 8.01e-16 ***
## BloodPressure            -0.015364   0.006054  -2.538   0.0112 *
## BMI                       0.091264   0.017331   5.266 1.40e-07 ***
## DiabetesPedigreeFunction  0.785216   0.364189   2.156   0.0311 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 652.26  on 499  degrees of freedom
## Residual deviance: 484.75  on 494  degrees of freedom
## AIC: 496.75
##
## Number of Fisher Scoring iterations: 5
```

All the features are statistically significant and should be regarded as important features for classifying whether a patient has diabetes or not.

```
fit1.probs =predict(fit1, test.dat, type ="response")
glm.pred=rep(0,268)
glm.pred[fit1.probs >.5]=1

# Create the confusion matrix
table(glm.pred, test.dat$Outcome)
```

```
##
## glm.pred   0   1
##        0 152  34
##        1  27  55
```

```
# Rate of misclassification
1-mean(glm.pred==test.dat$Outcome)
```

```
## [1] 0.2276119
```

LDA for classification problem

Discriminant analysis models the distribution of the predictors X separately in each of the response classes and then uses Bayes' theorem to flip these around into estimates for the probability of the response category given the value of X.

LDA computes `discriminant scores` for each observation to classify what response variable class it is in. These scores are obtained by finding linear combinations of the independent variables.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
lda.fit=lda(Outcome ~., data=train.dat)
Outcome.test = test.dat$Outcome

# Prediction in LDA
lda.pred=predict(lda.fit,test.dat)

# Get the predicted class labels
lda.class = lda.pred$class

# Construct the confusion matrix
table(lda.class, Outcome.test)
```

```
##          Outcome.test
## lda.class   0   1
##         0 156  36
##         1  23  53
```

```
# Rate of misclassification
1-mean(lda.class==Outcome.test)
```

```
## [1] 0.2201493
```

Classification trees

As regression trees, classification trees are also based on recursive partitioning of the feature space.

```
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
tree.pima = tree(as.factor(Outcome)~.,train.dat)
tree.pred = predict(tree.pima,test.dat,type = "class")
table(tree.pred,as.factor(test.dat$Outcome))
```

```
##
## tree.pred   0   1
##         0 121  13
##         1  58  76
```

```
1-mean(tree.pred==as.factor(test.dat$Outcome))
```

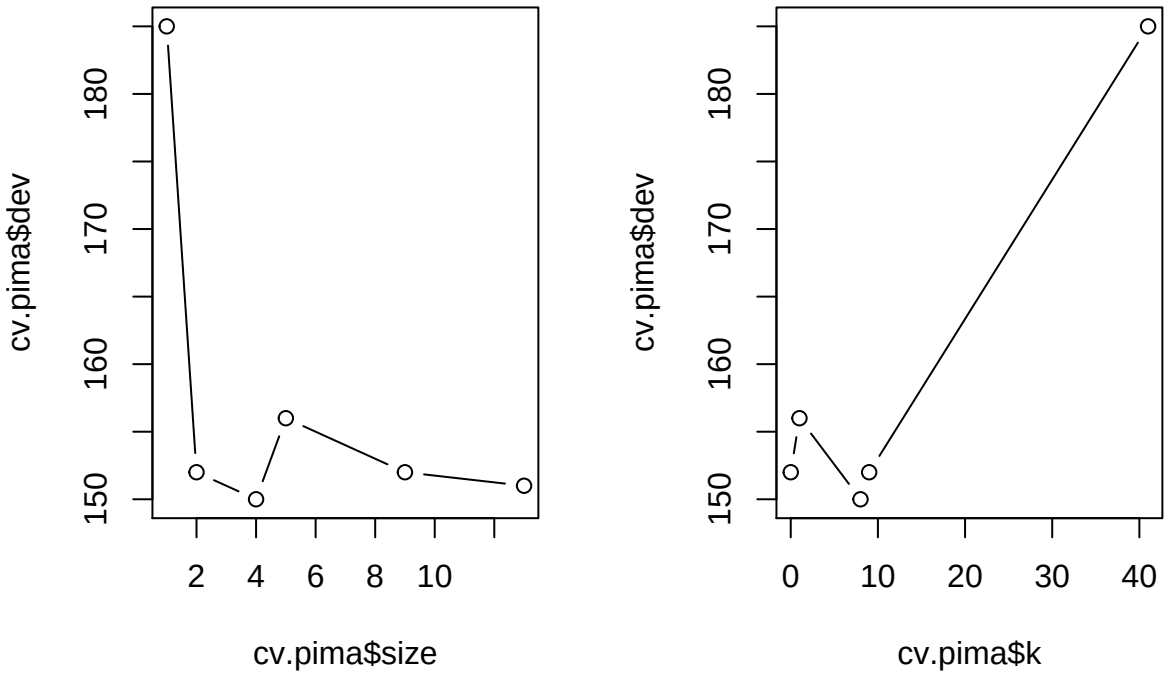
```
## [1] 0.2649254
```

One approach to improve the performance of a classification tree is to grow a large tree and then prune it.

```
set.seed(3)
cv.pima = cv.tree(tree.pima,FUN=prune.misclass)
names(cv.pima)
```

```
## [1] "size"   "dev"    "k"      "method"
```

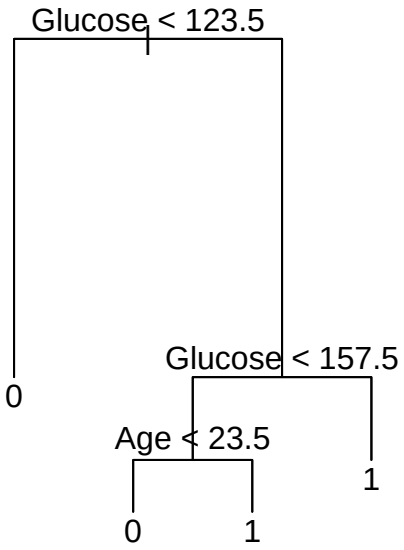
```
par(mfrow = c(1,2))
plot(cv.pima$size,cv.pima$dev,type="b")
plot(cv.pima$k,cv.pima$dev,type="b")
```




```
prune.pima = prune.misclass(tree.pima,best = 4)

plot(prune.pima)
```

```
text(prune.pima, pretty = 0)
```




```
pruned.tree.pred = predict(prune.pima, test.dat,type="class")
table(pruned.tree.pred,as.factor(test.dat$Outcome))
```

```
##
## pruned.tree.pred   0   1
##                0 136  27
##                1  43  62
```

```
1-mean(pruned.tree.pred==as.factor(test.dat$Outcome))
```

```
## [1] 0.261194
```

With pruning misclassification error is supposed to reduce.

Random Forest for Classfiction

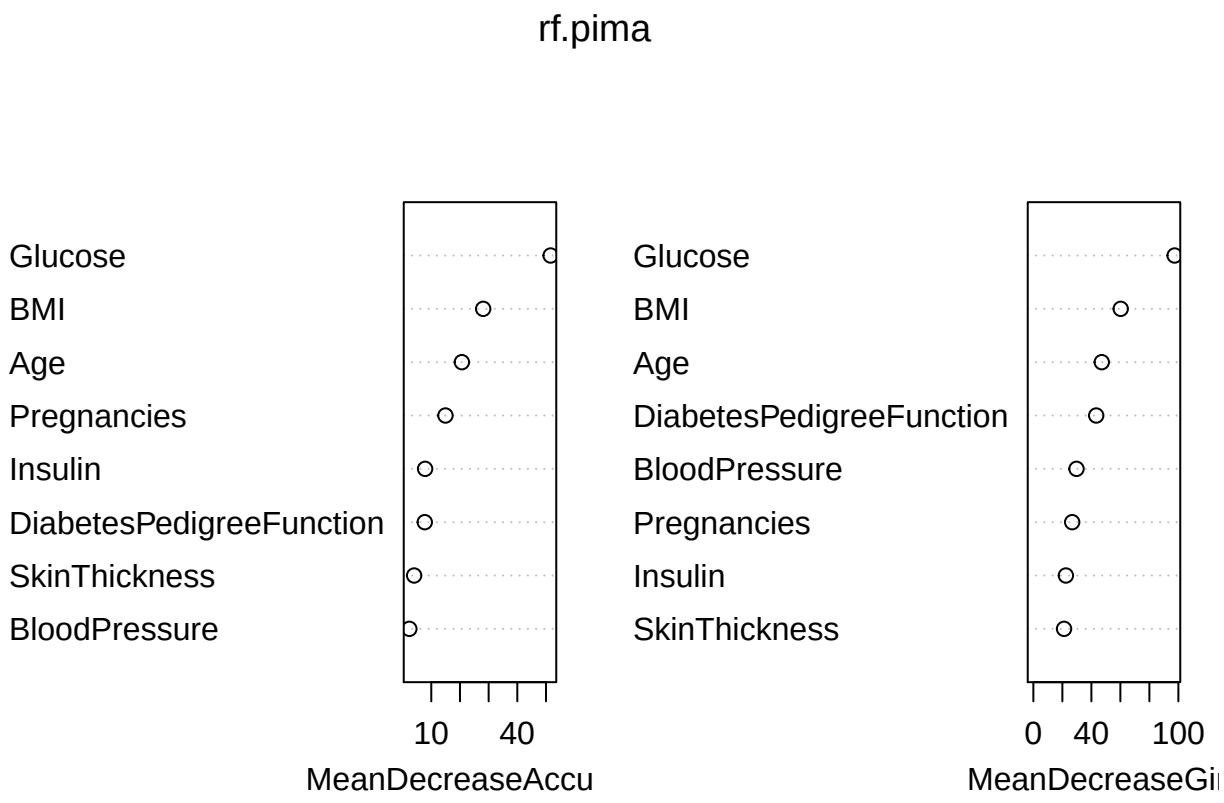
```
library(randomForest)
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
set.seed (31)
pima["out_f"] = as.factor(pima$Outcome)
pima = pima[,-9]
rf.pima = randomForest(out_f~.,data=pima, mtry=3, importance=TRUE)
rf.pima
```

```
##
## Call:
##  randomForest(formula = out_f ~ ., data = pima, mtry = 3, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 23.57%
## Confusion matrix:
##     0   1 class.error
## 0 422  78   0.1560000
## 1 103 165   0.3843284
```

```r
varImpPlot(rf.pima)
```



Random Forest for classification typically improves the misclassification error rate compared to the other methods.