A Decision Tree for Predicting Diabetes

October 11, 2017

The Data and Prediction Challenge

We will build a decision tree to predict diabetes for subjects in the Pima Indians dataset based on predictor variables such as age, blood pressure, and bmi. A subset of the Pima Indians data from the UCIrvine Machine Learning Repository is a built-in dataset in the MASS library. The Pima data in MASS contains 532 complete records from the original dataset. These 532 records have been broken down into two dataframes: Pima.tr (200 subjects) and Pima.te (332 subjects). All records with zeros that don't make sense have been cleaned out of these datasets.

```
> library(MASS)  #Pima indians data is in this package
> #Lets familiarize ourselves with the data
> ?Pima.tr
> dim(Pima.tr)

[1] 200   8

> head(Pima.tr)

  npreg glu bp skin  bmi   ped age type
1     5  86 68   28 30.2 0.364  24   No
2     7 195 70   33 25.1 0.163  55  Yes
3     5  77 82   41 35.8 0.156  35   No
4     0 165 76   43 47.9 0.259  26   No
5     0 107 60   25 26.4 0.133  23   No
6     5  97 76   27 35.6 0.378  52  Yes

> str(Pima.tr)  #gives the structure of the dataframe, easy way get types of data in each column

'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 ...

> summary(Pima.tr)

     npreg            glu              bp              skin
 Min.   : 0.00   Min.   : 56.0   Min.   : 38.00   Min.   : 7.00
 1st Qu.: 1.00   1st Qu.:100.0   1st Qu.: 64.00   1st Qu.:20.75
 Median : 2.00   Median :120.5   Median : 70.00   Median :29.00
 Mean   : 3.57   Mean   :124.0   Mean   : 71.26   Mean   :29.21
 3rd Qu.: 6.00   3rd Qu.:144.0   3rd Qu.: 78.00   3rd Qu.:36.00
 Max.   :14.00   Max.   :199.0   Max.   :110.00   Max.   :99.00
      bmi             ped              age         type
 Min.   :18.20   Min.   :0.0850   Min.   :21.00   No :132
 1st Qu.:27.57   1st Qu.:0.2535   1st Qu.:23.00   Yes: 68
 Median :32.80   Median :0.3725   Median :28.00
```

```
 Mean   :32.31   Mean   :0.4608   Mean   :32.11
 3rd Qu.:36.50   3rd Qu.:0.6160   3rd Qu.:39.25
 Max.   :47.90   Max.   :2.2880   Max.   :63.00
```

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

```
[1] 332   8
```

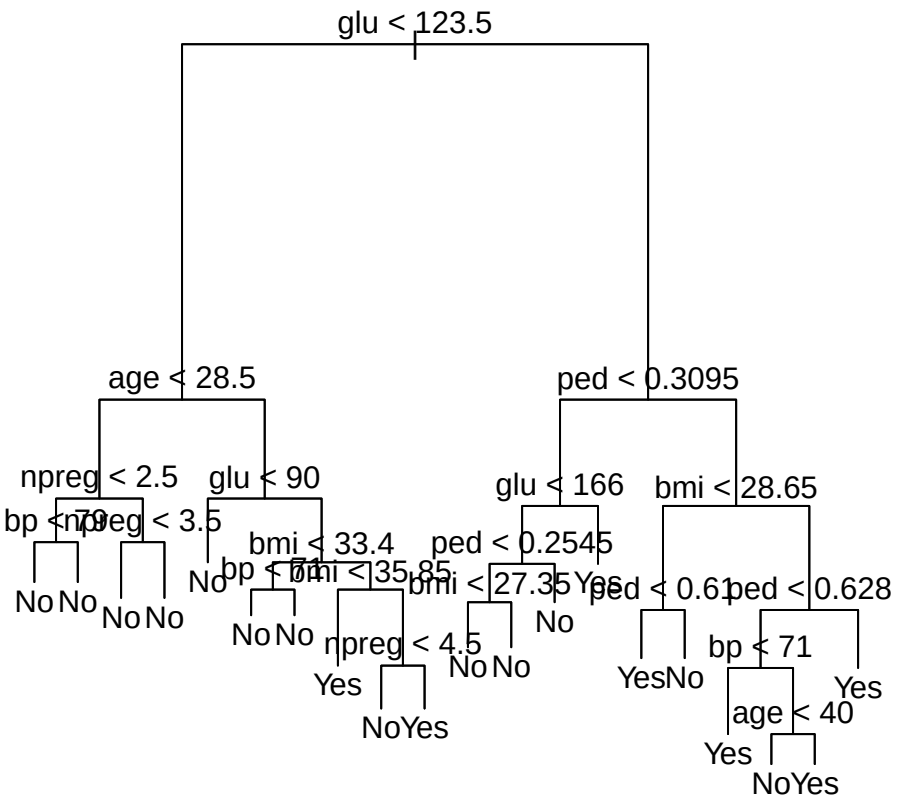
The First Tree

We Want to predict the variable "type" which indicates whether or not the subject has diabetes using all the other variables as predictors.

```
> library(tree)   #Loads the tree package
> pimatree <- tree(type~.,data=Pima.tr)  #Contructs the tree model based on all training data
> plot(pimatree)
> text(pimatree)
> summary(pimatree)  #compact summary of tree
```

```
Classification tree:
tree(formula = type ~ ., data = Pima.tr)
Variables actually used in tree construction:
[1] "glu"   "age"   "npreg" "bp"    "bmi"   "ped"
Number of terminal nodes:  20
Residual mean deviance:  0.4425 = 79.66 / 180
Misclassification error rate: 0.115 = 23 / 200
```

```
> #pimatree   #detailed verbal description of tree
```



We want to see how well the tree classifies subjects into diabetes or no diabetes categories so we use the tree constructed based on `Pima.tr`, the training data, to make predictions for the subjects in `Pima.te`, the test data. This is a more fair test of the tree model since a model usually predicts much better on the dataset used to construct it than on new data.

```
> head(Pima.te)  #Take a look at the test data
```

```
  npreg glu bp skin  bmi   ped age type
1     6 148 72   35 33.6 0.627  50  Yes
2     1  85 66   29 26.6 0.351  31   No
3     1  89 66   23 28.1 0.167  21   No
4     3  78 50   32 31.0 0.248  26  Yes
5     2 197 70   45 30.5 0.158  53  Yes
6     5 166 72   19 25.8 0.587  51  Yes
```

```
> set.seed(223)
> pima.pred <- predict(pimatree,newdata=Pima.te,type="class")  #Use the tree we just made to predict for the
> #pima.pred  #look at predictions
> #rbind(pima.pred,Pima.te$type)  #compare predictions to actual diabetes status
> table(pima.pred,Pima.te$type)   #diagonal are correct predictions, off-diagonal are incorrect
```

```
pima.pred  No Yes
      No  173  44
      Yes  50  65
```

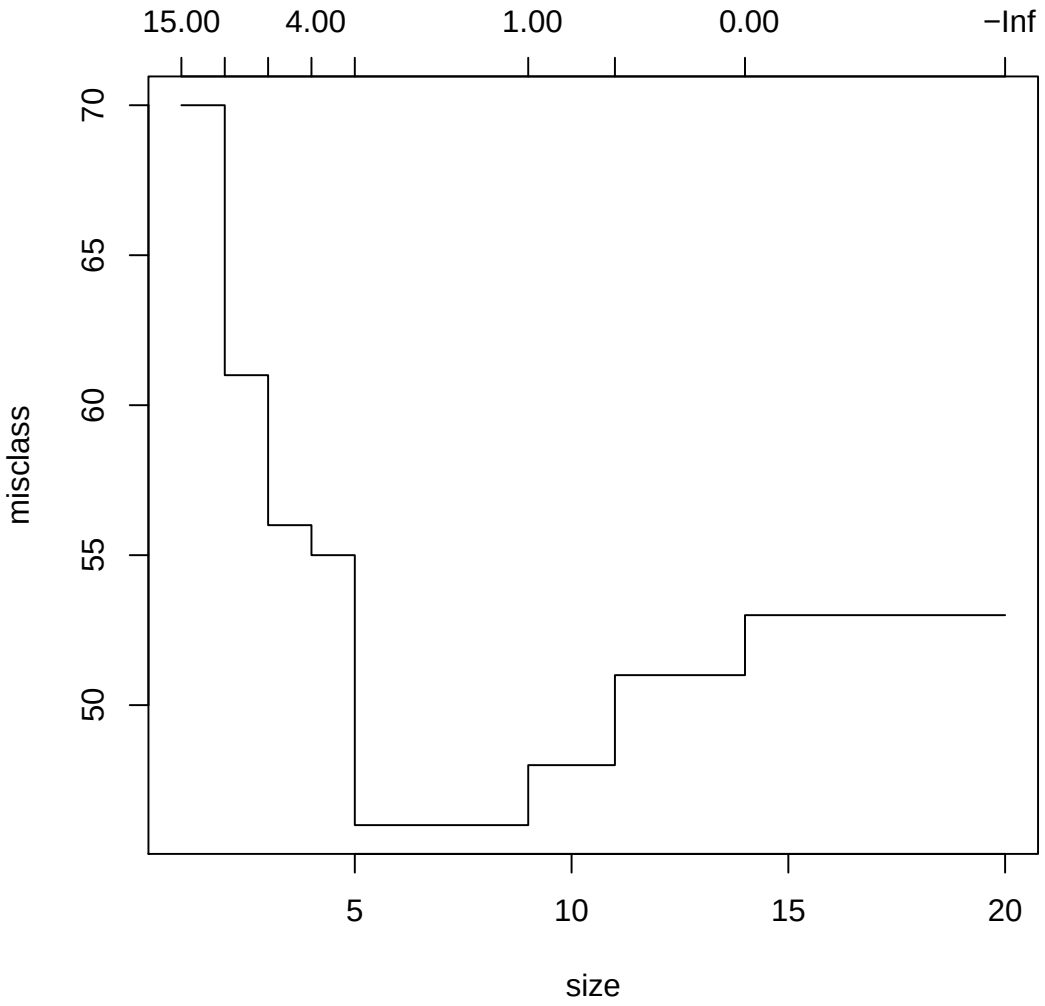
```
> (44+50)/332  #proportion misclassified
```

```
[1] 0.2831325
```

Cross-validation to determine optimal size of the tree

The tree has lots of terminal nodes and is quite bushy. It is likely that it overfits to the training data. We prune the tree down to make it easier to interpret and better at predicting outside the training data without losing much predictive accuracy. We use cross-validation to choose the amount of pruning, i.e. how much we cut down the size of the tree.

```
> set.seed(561)  #saves the randomization used in cv.tree, you need not do this
> pima.cv <- cv.tree(pimatree,,FUN=prune.misclass,K=10)  #K=10 is the default value
> pima.cv

$size
[1] 20 14 11  9  5  4  3  2  1

$dev
[1] 53 53 51 48 46 55 56 61 70

$k
[1]       -Inf  0.0000000  0.6666667  1.0000000  1.5000000  4.0000000  5.0000000
[8] 11.0000000 15.0000000

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"

> plot(pima.cv)  #Plot of $size vs $dev - any choice of $size from 5 to 9 inclusive seems to make $dev the sm
```



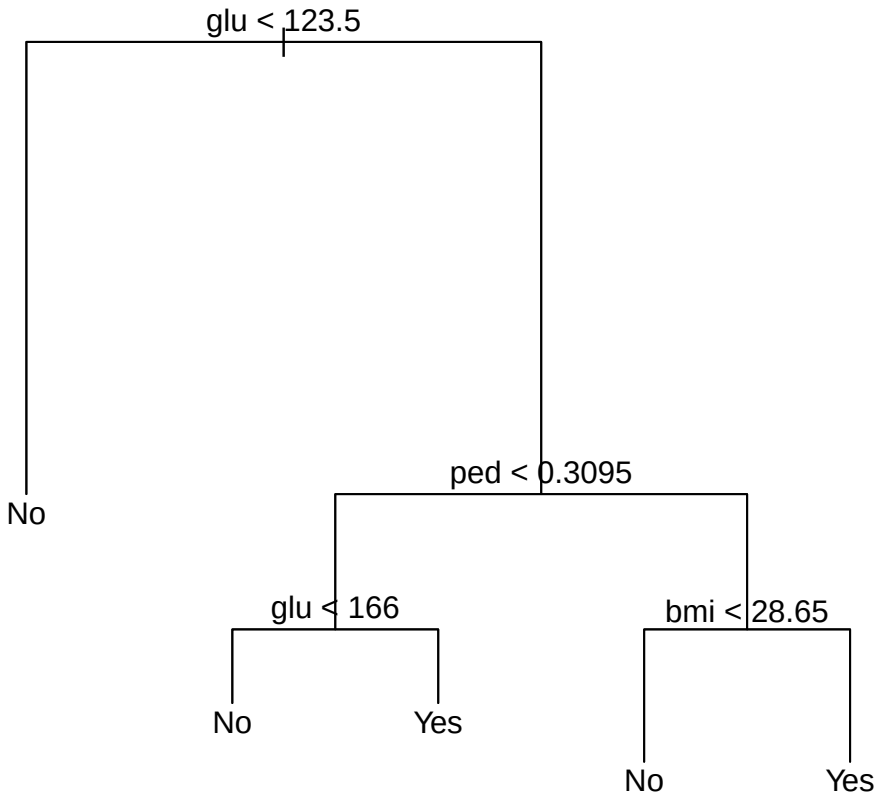
Note the object produced by `cv.tree()` is pima.cv. pima.cv holds a summary of cross-validation results. $size is the number of terminal nodes or leaves in the tree. $dev is a measure of how much you lose by reducing the original tree to the corresponding value in $size. Large $dev (deviance) indicates a large loss of information when the original tree is reduced to the corresponding $size. Choose $size to correspond to the smallest value of $dev.

The Improved, Pruned Tree

Cross-validation is used to help you find the *size* of the best pruned tree. Now we need to construct the best pruned tree using the training data in Pima.tr.

```
> pima.prune <- prune.misclass(pimatree,best=5)  #here best is the number of terminal nodes you want in the p
> summary(pima.prune)

Classification tree:
snip.tree(tree = pimatree, nodes = c(12L, 15L, 14L, 2L))
Variables actually used in tree construction:
[1] "glu" "ped" "bmi"
Number of terminal nodes:  5
Residual mean deviance:  0.9063 = 176.7 / 195
Misclassification error rate: 0.165 = 33 / 200

> plot(pima.prune)
> text(pima.prune,pretty=T)
> pima.prune.pred <- predict(pima.prune,newdata=Pima.te,type="class")  #Predict test data using pruned tree
> table(pima.prune.pred,Pima.te$type)
```

```
pima.prune.pred  No Yes
            No  193  51
            Yes  30  58

> (30+51)/332  #Proportion misclassified with pruned tree, compare to (44+50)/332 with full tree

[1] 0.2439759
```



So the pruned tree misclassifies 24% of the test dataset, while the first, unpruned tree misclassified about 28%.

Random Forest and Bagging

Now to try to improve our predictions by growing a forest of bushy trees.

```
> library(randomForest)
> set.seed(387)
> pimaForest <- randomForest(type ~.,data=Pima.tr)  #Number of variables tried at each split was 2
> pimaForest

Call:
 randomForest(formula = type ~ ., data = Pima.tr)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 28.5%
Confusion matrix:
     No Yes class.error
No  108  24   0.1818182
Yes  33  35   0.4852941
```

One of the arguments you might like to change in `randomForest()` is mtry. At each split, a subset of size mtry is chosen from all predictors. The split is determined based only on the predictors in this subset. We'll use the error from the test data to find the best value of mtry.

```
> error.by.mtry  <- numeric(7)
> oob.error <- numeric(7)
> for (i in 1:7)
+ {pimaForest2 <- randomForest(type~.,data=Pima.tr,mtry=i,ntree=400)
+ oob.error[i] <- pimaForest2$err.rate[400]
+
+  pred.mtry <- predict(pimaForest2,newdata=Pima.te,type="class")
+  num.misclassified <- sum(Pima.te$type!=pred.mtry)  #number misclassified in test set
+   error.by.mtry[i] <- num.misclassified
+   print(i)}

[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7

> error.by.mtry/332

[1] 0.2319277 0.2349398 0.2379518 0.2409639 0.2469880 0.2439759 0.2469880

> oob.error

[1] 0.260 0.275 0.260 0.280 0.280 0.275 0.270

> plot(1:7,oob.error,col="blue",pch=19,type="b",ylim=c(0,.3))
> points(1:7,error.by.mtry/332,col="red",pch=19,type="b")
```

Random forests doesn't improve performance on test data very much.

Importance of Predictors

Roughly, the importance of each variable or predictor is determined by determining how much splits in this variables reduces the Gini Index for classification trees. The larger the Gini Index, the more "mixed" the observation in each terminal node are across the classes. The Gini Index is small when all the observations in each terminal node tend to be from the same class.

```
> importance(pimaForest)

      MeanDecreaseGini
npreg         8.999907
glu          21.581238
bp            7.908906
skin          9.820643
bmi          12.637090
ped          13.544807
age          14.637802

> varImpPlot(pimaForest)
```