IST438-W6-Applications

2023-04-17

Decision trees - II

In this application, we will train decision trees for regression and classification tasks by tuning the hyperparameters in `rpart()`.

Dataset

The `PimaIndiansDiabetes` data set as it relates to predicting whether someone has diabetes. This data is provided by the mlbench package.

```
#install.packages("mlbench")
library(mlbench)
data("PimaIndiansDiabetes")
str(PimaIndiansDiabetes)
```

```
'data.frame':   768 obs. of  9 variables:
 $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
 $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
 $ pressure: num  72 66 64 66 40 74 50 0 70 96 ...
 $ triceps : num  35 29 0 23 35 0 32 0 45 0 ...
 $ insulin : num  0 0 0 94 168 0 88 0 543 0 ...
 $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
 $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
 $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
```

The relevant variables are:

`pregnant` - Number of times pregnant

`glucose` - Plasma glucose concentration (glucose tolerance test)

`pressure` - Diastolic blood pressure (mm Hg)

`triceps` - Triceps skin fold thickness (mm)

`insulin` - 2-Hour serum insulin (mu U/ml)

`mass` - Body mass index (weight in kg/(height in m)^2)

`pedigree` - Diabetes pedigree function

`age` - Age (years)

`diabetes` - Class variable (test for diabetes)

Splitting

```r
library(rsample)
```

```
Warning: package 'rsample' was built under R version 4.3.1
```

```r
diabetes_split <- initial_split(data = PimaIndiansDiabetes, # dataset to split
                                  prop = 0.80)    # proportion of train set

diabetes_train <- diabetes_split |> training()
diabetes_test  <- diabetes_split |> testing()
```

Hyperparameters

`rpart()` consists many hyperparameters, but we focus on the most commonly used ones as follows:

- `minsplit`: the minimum number of observations that must exist in a node in order for a split to be attempted.
- `minbucket`: the minimum number of observations in any terminal node.
- `cp`: complexity parameter.
- `maxdepth`: set the maximum depth of any node of the final tree, with the root node counted as depth 0.

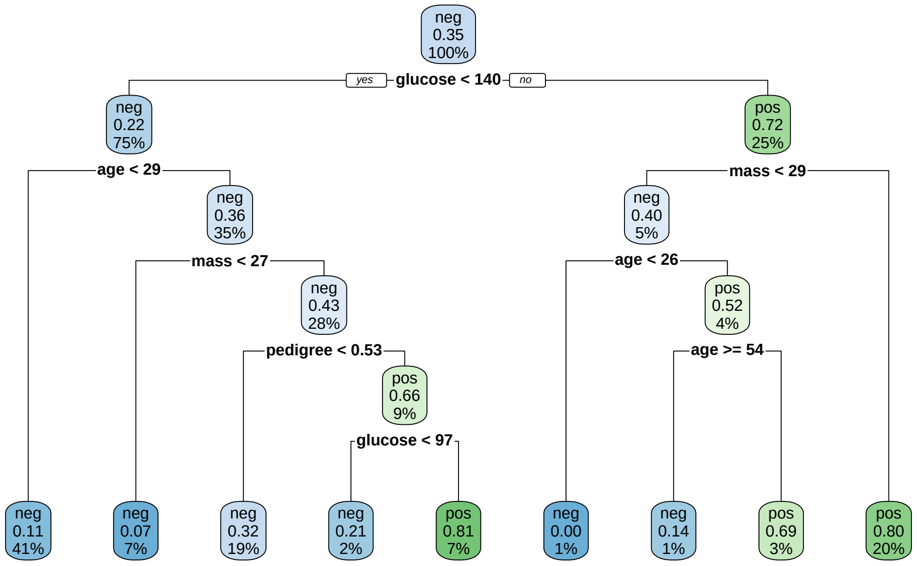
The default values of hyperparameters in `rpart()`: `minsplit = 20`, `minbucket = round(minsplit/3)`, `cp = 0.01`, and `maxdepth = 30`.

Training a vanilla decision tree

```r
library(rpart)
```

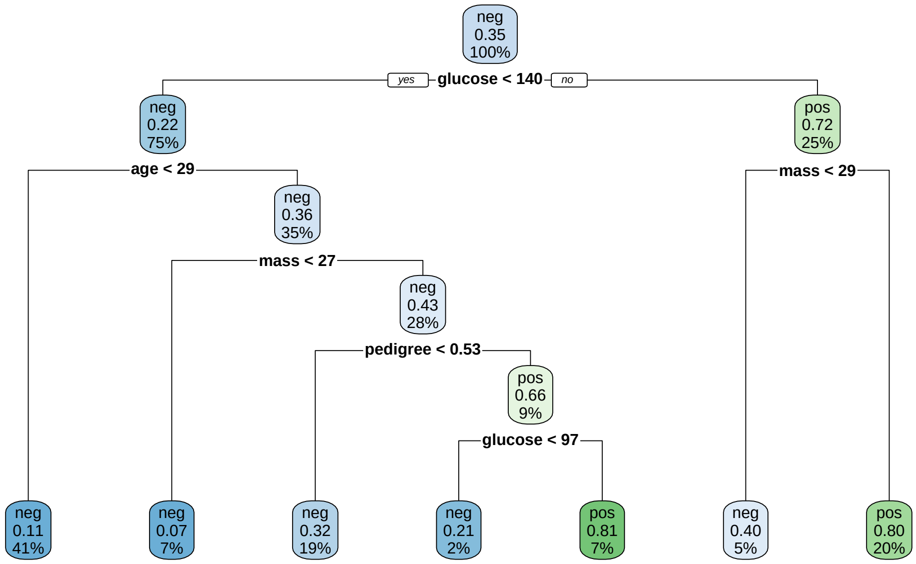
```
Warning: package 'rpart' was built under R version 4.3.1
```

```r
library(rpart.plot)
vanilla_dt <- rpart(diabetes ~ .,
                    data = diabetes_train,
                    method = "class")

rpart.plot(vanilla_dt)
```



Training a less deeper decision tree by tuning `cp`

```r
less_dt1 <- rpart(diabetes ~ .,
                  data = diabetes_train,
                  method = "class",
                  cp = 0.015)

rpart.plot(less_dt1)
```



Compare the performance of the vanilla dt and less deeper dt1

```r
library(caret)
```

```
Loading required package: ggplot2
```

```
Warning: package 'ggplot2' was built under R version 4.3.1
```

```
Loading required package: lattice
```

```
Warning: package 'lattice' was built under R version 4.3.1
```

```
# performance metrics of the vanilla dt
vanilla_preds <- predict(vanilla_dt, diabetes_test, type = "class")

confusionMatrix(vanilla_preds,
                diabetes_test$diabetes,
                positive = "pos")
```

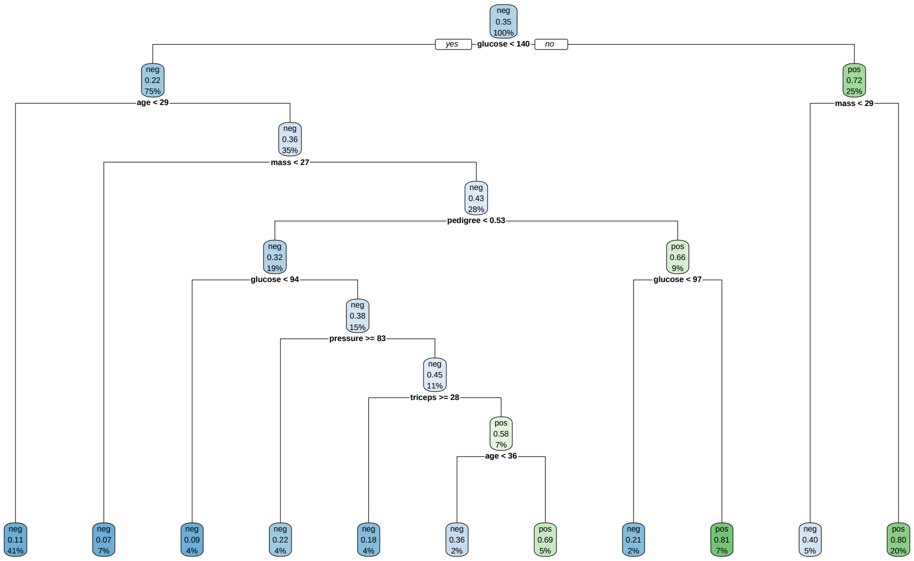
```
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  74  24
       pos  25  31

               Accuracy : 0.6818
                 95% CI : (0.602, 0.7545)
    No Information Rate : 0.6429
    P-Value [Acc > NIR] : 0.1778

                  Kappa : 0.3099

 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.5636
            Specificity : 0.7475
         Pos Pred Value : 0.5536
         Neg Pred Value : 0.7551
             Prevalence : 0.3571
         Detection Rate : 0.2013
   Detection Prevalence : 0.3636
      Balanced Accuracy : 0.6556

       'Positive' Class : pos
```

```
# performance metrics of the less deeper dt
less_preds1 <- predict(less_dt1, diabetes_test, type = "class")

confusionMatrix(less_preds1,
                diabetes_test$diabetes,
                positive = "pos")
```

```
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  75  26
       pos  24  29

               Accuracy : 0.6753
                 95% CI : (0.5953, 0.7485)
    No Information Rate : 0.6429
    P-Value [Acc > NIR] : 0.2257

                  Kappa : 0.2872

 Mcnemar's Test P-Value : 0.8875

            Sensitivity : 0.5273
            Specificity : 0.7576
         Pos Pred Value : 0.5472
         Neg Pred Value : 0.7426
             Prevalence : 0.3571
         Detection Rate : 0.1883
   Detection Prevalence : 0.3442
      Balanced Accuracy : 0.6424

       'Positive' Class : pos
```

Training a less deeper decision tree by tuning `minsplit`

```
less_dt2 <- rpart(diabetes ~ .,
                  data = diabetes_train,
                  method = "class",
                  minsplit = 30)

rpart.plot(less_dt2)
```



Compare the performance of the vanilla dt and less deeper dt2

```r
# performance metrics of the vanilla dt
confusionMatrix(vanilla_preds,
                diabetes_test$diabetes,
                positive = "pos")
```

```
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  74  24
       pos  25  31

               Accuracy : 0.6818
                 95% CI : (0.602, 0.7545)
    No Information Rate : 0.6429
    P-Value [Acc > NIR] : 0.1778

                  Kappa : 0.3099
```

```
 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.5636
            Specificity : 0.7475
         Pos Pred Value : 0.5536
         Neg Pred Value : 0.7551
             Prevalence : 0.3571
         Detection Rate : 0.2013
   Detection Prevalence : 0.3636
      Balanced Accuracy : 0.6556

       'Positive' Class : pos
```

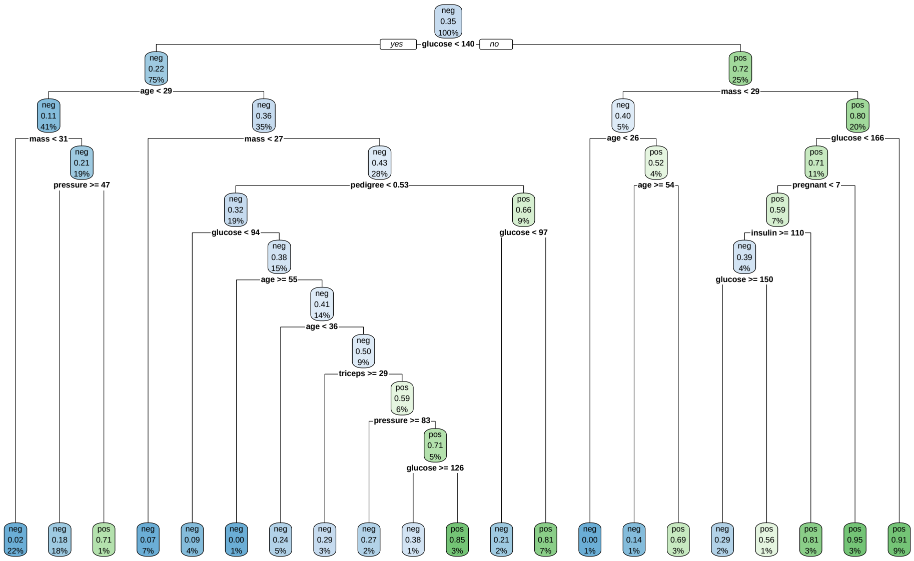
```
# performance metrics of the less deeper dt2
less_preds2 <- predict(less_dt2, diabetes_test, type = "class")

confusionMatrix(less_preds2,
                diabetes_test$diabetes,
                positive = "pos")
```

```
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  73  24
       pos  26  31

               Accuracy : 0.6753
                 95% CI : (0.5953, 0.7485)
    No Information Rate : 0.6429
    P-Value [Acc > NIR] : 0.2257

                  Kappa : 0.2986

 Mcnemar's Test P-Value : 0.8875

            Sensitivity : 0.5636
            Specificity : 0.7374
         Pos Pred Value : 0.5439
         Neg Pred Value : 0.7526
```

```
             Prevalence : 0.3571
         Detection Rate : 0.2013
   Detection Prevalence : 0.3701
      Balanced Accuracy : 0.6505

       'Positive' Class : pos
```

Training a deeper decision tree by tuning `cp`

```
deeper_dt <- rpart(diabetes ~ .,
                   data = diabetes_train,
                   method = "class",
                   cp = 0.001)

rpart.plot(deeper_dt)
```



Compare the performance of the vanilla dt, less deeper dt2, deeper tree

```
# performance metrics of the vanilla dt
confusionMatrix(vanilla_preds,
                diabetes_test$diabetes,
                positive = "pos")
```

```
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  74  24
       pos  25  31

               Accuracy : 0.6818
                 95% CI : (0.602, 0.7545)
    No Information Rate : 0.6429
    P-Value [Acc > NIR] : 0.1778

                  Kappa : 0.3099

 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.5636
            Specificity : 0.7475
         Pos Pred Value : 0.5536
         Neg Pred Value : 0.7551
             Prevalence : 0.3571
         Detection Rate : 0.2013
   Detection Prevalence : 0.3636
      Balanced Accuracy : 0.6556

       'Positive' Class : pos
```

```
# performance metrics of the less deeper dt2
confusionMatrix(less_preds2,
                diabetes_test$diabetes,
                positive = "pos")
```

```
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  73  24
       pos  26  31

               Accuracy : 0.6753
                 95% CI : (0.5953, 0.7485)
    No Information Rate : 0.6429
    P-Value [Acc > NIR] : 0.2257

                  Kappa : 0.2986

 Mcnemar's Test P-Value : 0.8875

            Sensitivity : 0.5636
            Specificity : 0.7374
         Pos Pred Value : 0.5439
         Neg Pred Value : 0.7526
             Prevalence : 0.3571
         Detection Rate : 0.2013
   Detection Prevalence : 0.3701
      Balanced Accuracy : 0.6505

       'Positive' Class : pos
```

```
# performance metrics of the deeper tree
deeper_preds <- predict(deeper_dt, diabetes_test, type = "class")

confusionMatrix(deeper_preds,
                diabetes_test$diabetes,
                positive = "pos")
```

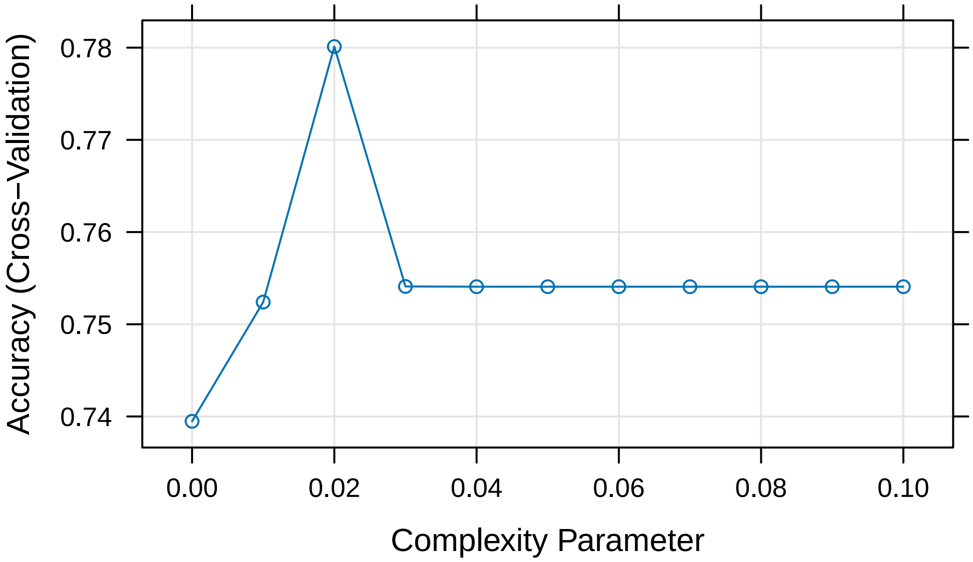
```
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  74  22
       pos  25  33
```

```
               Accuracy : 0.6948
                 95% CI : (0.6156, 0.7664)
    No Information Rate : 0.6429
    P-Value [Acc > NIR] : 0.1026

                  Kappa : 0.3433

 Mcnemar's Test P-Value : 0.7705

            Sensitivity : 0.6000
            Specificity : 0.7475
         Pos Pred Value : 0.5690
         Neg Pred Value : 0.7708
             Prevalence : 0.3571
         Detection Rate : 0.2143
   Detection Prevalence : 0.3766
      Balanced Accuracy : 0.6737

       'Positive' Class : pos
```

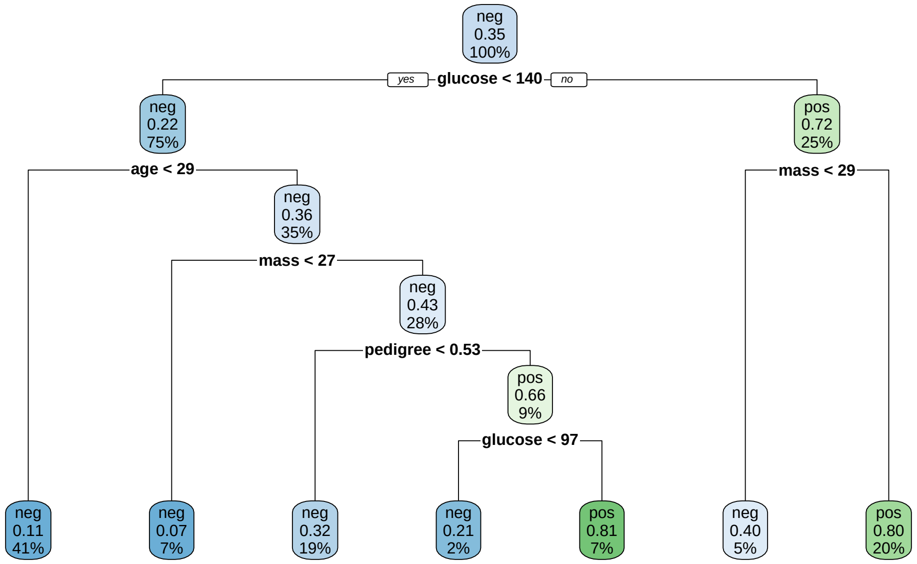
Grid search in caret

caret provides many models with the list of hyperparameters: https://topepo.github.io/caret/available-models.html

```
library(caret)
fit_control <- trainControl(method = "cv", number = 10)

dt_model <- train(diabetes ~ .,
                  data = diabetes_train,
                  method = "rpart",
                  trControl = fit_control,
                  tuneGrid = expand.grid(cp = seq(0, 0.1, 0.01)))

plot(dt_model)
```

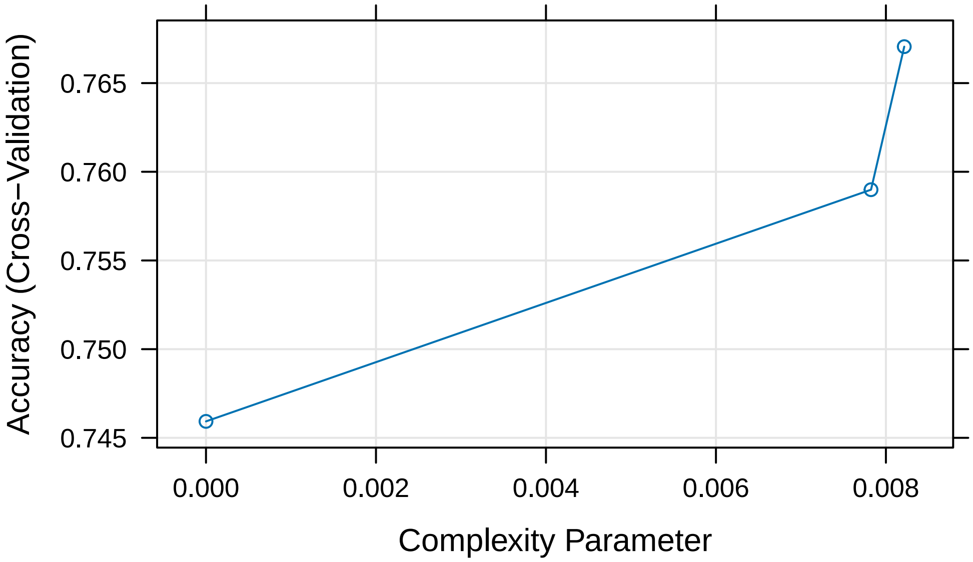


```r
rpart.plot(dt_model$finalModel)
```

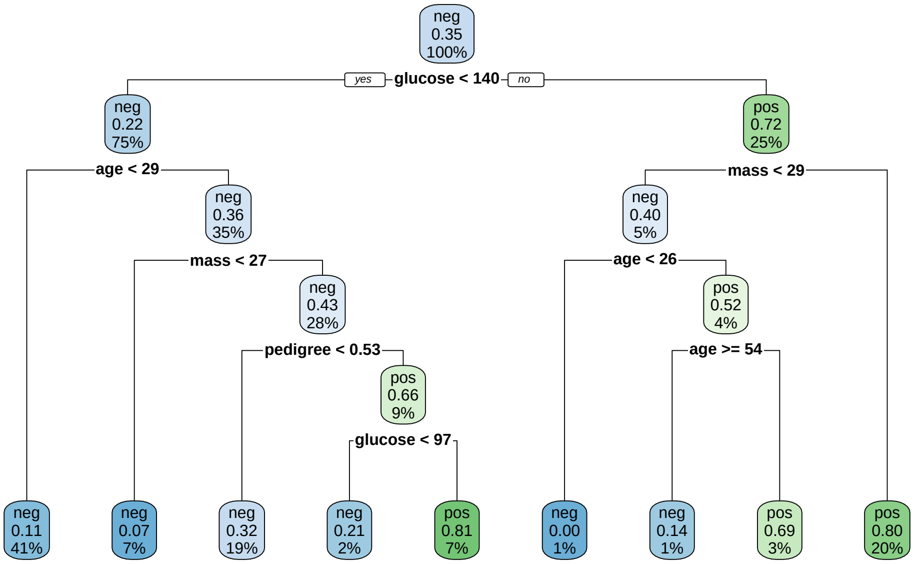


Random search in caret

```
library(caret)
fit_control <- trainControl(method = "cv", number = 10,
                            search = "random")

dt_model <- train(diabetes ~ .,
                  data = diabetes_train,
                  method = "rpart",
                  trControl = fit_control)

plot(dt_model)
```



```
rpart.plot(dt_model$finalModel)
```



neg
0.35
100%
yes
glucose < 140
no
neg
0.22
75%
pos
0.72
25%
age < 29
mass < 29
neg
0.36
35%
neg
0.40
5%
mass < 27
age < 26
neg
0.43
28%
pos
0.52
4%
pedigree < 0.53
age >= 54
pos
0.66
9%
glucose < 97
neg
0.11
41%
neg
0.07
7%
neg
0.32
19%
neg
0.21
2%
pos
0.81
7%
neg
0.00
1%
neg
0.14
1%
pos
0.69
3%
pos
0.80
20%