Decision Tree Example: Statlog Heart

We will use statlog heart dataset again for decision tree model practice. Statlog (Heart) Data Set is downloaded from UCI machine learning repository. It has 13 different attributes and 1 class variable. The detailed feature attributes are presented in the below. You could also find it here: https://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29

The attributes include: 1. Age; 2. Sex (female =0, male =1); 3. chest pain type (4 values); 4. resting blood pressure; 5. serum cholestoral in mg/dl; 6. fasting blood sugar > 120 mg/dl (have =1, don’t have =0); 7. resting electrocardiographic results (values 0,1,2); 8. maximum heart rate achieved; 9. exercise induced angina (don’t have =0, have =1); 10. oldpeak = ST depression induced by exercise relative to rest; 11. the slope of the peak exercise ST segment; 12. number of major vessels (0-3) colored by flourosopy; 13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect;

The class variable is statlogheart: 1=Absence of heart disease; 2= presence of heart disease;

Among all attributes, we have real attributes: 1,4,5,8,10,12; ordered attributes: 11; binary attributes: 2,6,9 and nominal attributes: 3, 7, 13;

# 1. Import dataset to R using stringsAsFactors = FALSE.

Run the following transformations:

dt[c(2,3,6,7,9,13,14)]<-data.frame(lapply(dt[c(2,3,6,7,9,13,14)],factor))

dt <- dt[c(2,3,9,10,11,12,14)]

**Code:**

dt<- read.csv(file.choose(),stringsAsFactors = FALSE)

# Run the following transformations:

dt[c(2,3,6,7,9,13,14)]<-data.frame(lapply(dt[c(2,3,6,7,9,13,14)],factor))

dt <- dt[c(2,3,9,10,11,12,14)]

str(dt)

**Ans :**

data.frame': 270 obs. of 7 variables:

$ sex : Factor w/ 2 levels "0","1": 2 1 2 2 1 2 2 2 2 1 ...

$ chestpain : Factor w/ 4 levels "1","2","3","4": 4 3 2 4 2 4 3 4 4 4 ...

$ angina : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 2 2 1 1 ...

$ oldpeak : num 2.4 1.6 0.3 0.2 0.2 0.4 0.6 1.2 1.2 4 ...

$ st.segment : int 2 2 1 2 1 1 2 2 2 2 ...

$ majorvessels: int 3 0 0 1 1 0 1 1 2 3 ...

$ statlogheart: Factor w/ 2 levels "1","2": 2 1 2 1 1 1 2 2 2 2 ...

# 2. Pick 70% records randomly as training dataset.

**Code**:

nrows <- nrow(dt)

nrows

train.size<- floor(0.7 \* nrows)

train.size

set.seed(1234)

train.index <- sample(1:nrows,train.size,replace = F)

dt.train<- dt[train.index,] # 189 training observations

dt.test<- dt[-train.index,] # 81 testing observations

str(dt.train)

str(dt.test)

# 3. Build a logit regression model where statlogheart is the target variable.

**Code**:

model <- glm(statlogheart ~ . , data = dt.train, family='binomial')

summary(model)

# 4. What is the McFadden R^2 for this model? Evaluate the model by its fitness.

**Code**:

R2logit<- function(model){

R2<- 1-(model$deviance/model$null.deviance)

return(R2)

}

R2logit(model)

**Ans**:

0.5170106

# 5. Evaluate the performance of training dataset by using cross table. What percentage of people who do not have heart disease are wrongly predicted?

# Hint for predict: predicted<-predict(model, dt.train, type= "response")

# Hint for converting logs ratio: predicted<- ifelse(predicted>=0.5,2,1)

**Code:**

predicted\_train<-predict(model, dt.train, type= "response")

predicted\_train<- ifelse(predicted\_train>=0.5,2,1)

summary(predicted\_train)

install.packages('gmodels')

library(gmodels)

CrossTable(dt.train$statlogheart, predicted\_train)

**Ans**:

# 25 / 0.25%

# 6. Evaluate the performance of testing dataset by using cross table. What percentage of people who have heart disease are wrongly predicted?

# Hint for predict: predicted<-predict(model, dt.test, type= "response")

**Code:**

str(dt.train)

predicted\_test <-predict(model, dt.test, type= "response")

summary(predicted\_test)

#install.packages('gmodels')

library(gmodels)

predicted\_test<- ifelse(predicted\_test>=0.5,2,1)

CrossTable(dt.test$statlogheart, predicted\_test)

**Ans:**

16 / 0.16%

# 7. Predict using naiveBayes

# a. Predict classification for the training dataset and create cross table.

# Note for predict: parameter type=response is needed only for logit.

# Predict classification for the testing dataset and create cross table.

**Code**:

install.packages('e1071')

library(e1071)

model <- naiveBayes(x=dt.train[-7], y=dt.train$statlogheart, laplace = 1)

predicted.train.naive <- predict(model, dt.train[-7])

CrossTable(dt.train[[7]], predicted.train.naive)

predicted.test.naive <- predict(model, dt.test[-7])

CrossTable(dt.test[[7]], predicted.test.naive)

# 8. Predict using decision tree

# a. Predict classification for the training dataset and create cross table.

# Predict classification for the testing dataset and create cross table.

**Code**:

install.packages("C50")

library(C50)

model <- C5.0(dt.train[-7],dt.train$statlogheart)

model

statlog\_train\_pred <- predict(model,dt.train)

install.packages('gmodels')

library(gmodels)

CrossTable(dt.train$statlogheart, statlog\_train\_pred, dnn=c("Actual", "Predicted"))

statlog\_test\_pred <- predict(model,dt.test)

install.packages('gmodels')

library(gmodels)

CrossTable(dt.train$statlogheart, statlog\_test\_pred, dnn=c("Actual", "Predicted"))

# 9. Predict using knn at k=3

# a. Predict classification for the training dataset and create cross table.

# Hint: train=dt.train, test=dt.train

**Code**:

library(class)

library(gmodels)

predicted.train <- knn(train=dt.train[-7], test=dt.train[-7],

cl=dt.train$statlogheart, k=3)

CrossTable(dt.train$statlogheart, predicted.train, dnn=c("Actual", "Predicted"))

# b. Predict classification for the testing dataset and create cross table.

# Hint: train=dt.train, test=dt.test

**Code**:

predicted.test <- knn(train=dt.train[-7], test=dt.test[-7],

cl=dt.train$statlogheart, k=3)

CrossTable(dt.test$statlogheart, predicted.test, dnn=c("Actual", "Predicted"))

# 10. Compare performance of models using training dataset

**Ans**:

#naiveBayes <- FN = 9 FP = 17 , total = 26

#decision tree <- FN = 13 FP = 9 , total =22

#knn <- FN = 11 FP = 10 , total = 21

#logit <- FN = 13 FP = 12 , total = 25

# 11. Compare performance of models using testing dataset

**Ans**:

#naiveBayes <- FN = 1 FP = 11 , total = 12

#decision tree <- FN = 1 FP = 11 , total = 12

#knn <- FN = 2 FP = 15 , total = 17

#logit <- FN = 1 FP = 15 , total = 16

# 12. Which model would you recommend overall?

# Hint: Typically you want to give preference to a model with good test dataset performance. If 2 models perform equally well over test dataset then look for model that performs well for training dataset.

# If you are looking for attribution and not prediction then you try to get the performance of regression model as close to the best performing other model, and then use regression model.

**Ans**:

#Decision tree performs better