Control 5
Investigación de Operaciones

Martín Salinas
201773557-0

July 27, 2020

Diabetes

Los resultados de este código, se pueden ver en el siguiente link dentro del Notebook diabetes.ipynb:
`https://github.com/xMuramasa/IO-STUFF-WITH-R/tree/master/Control%205%20en%20R`.

1. Generar un árbol, sin utilizar poda y considerando el 70 % de los datos entregados para Training. Conteste lo siguiente:

   - Describa el árbol obtenido: cantidad de niveles obtenido, cantidad de hojas del árbol obtenido, entre otros.

     Se utiliza Jupiter Notebook para la ejecución del código.

     Para la obtención de los datos, primero se debe usar la siguiente seccińn de opciones de configuracińn visual:

```
#configuraciones
library(repr)
options(repr.plot.width=30,repr.plot.height=10)
IRdisplay::display_html("<style>.container{width:95%!important;}</style>")
options(jupyter.plot_mimetypes=c("text/plain","image/png"))
```

     Luego, se carga la librería a utilizar y el dataset:

```
library(tree)
data <- read.table("DatosControl.csv",header=TRUE,sep=',')
```

     Además, se deben pasar a factor las columnas siguientes, esto se hace porque para el árbol es necesario discretizar algunos valores.

```
data$Nembarazada <- factor(data$Nembarazada)
data$Diabetes <- factor(data$Diabetes)
```

Para el árbol se utiliza la siguiente función, específica para este problema.

```
createData<-function(X, d, show){
    #Seteamos una semilla para replicar resultados
    set.seed(42)
    # En este caso utilizaremos un conjunto de
    # entrenamiento con el X% de los datos
    train_size <- floor(X * nrow(d))
    #El size sera el X% de la cantidad de filas
    train_mask <- sample(seq_len(nrow(d)),size= train_size)
    #Mascara aleatoria
    # Separamos los conjuntos
    train <- data[train_mask, ]
    test <- data[-train_mask, ]
    # se crea el arbol
    t = tree(Diabetes ~ Nembarazada + Glucosa + Presion + Triceps +
                Insulina + IMC + Pedigree + Edad, data = train)
    if(show){
        # plot del arbol
        plot(t)
        text(t,pretty=1)
        # se muestra datos del arbol
        print(summary(t))
    }
    # variable retorno
    dat <- list("train" = train, "test" = test, "tree" = t)
    return(dat)
    }
```

Para uzar esta función se hace de la siguiente manera:

```
newDat <- createData(X=0.7, d=data, show=T)
```

Lo que retorna lo siguiente:

```
Classification tree:
tree(formula = Diabetes ~ Nembarazada + Glucosa + Presion + Triceps
+ Insulina + IMC + Pedigree + Edad, data = train)
Variables actually used in tree construction:
[1] "Glucosa"     "Edad"        "IMC"         "Triceps"     "Nembarazada"
[6] "Presion"     "Pedigree"
Number of terminal nodes:  19
Residual mean deviance:  0.7274 = 376.8 / 518
Misclassification error rate: 0.1695 = 91 / 537
```

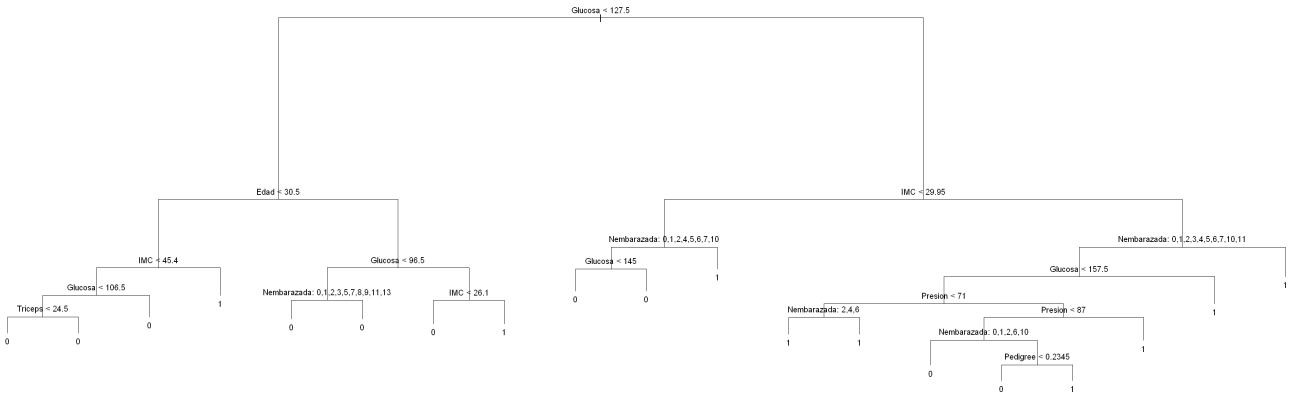

Figure 1: Árbol Resultante

En base a lo anterior, se tiene que la cantidad de hojas es 19. La cantidad de niveles es 9, el ratio de error de missclassification es de 0.1695.

- ¿Qué variables no generan particiones?, explique porqué éstas variables no participan en el árbol generado.

  Las variables utilizadas en la construcción del árbol fueron: "Glucosa", "Edad", "IMC", "Triceps", "Nembarazada", "Precision" y "Pedigree". Esto se debe a que al usar la variable "Insulina" en la construcción del árbol presenta una bondad menor que realizarlo incluyendo dicha variable.

2. Nuevamente, sin utilizar poda y considerando el 70 % de los datos entregados para Training, ¿Qué sucede si las variables Glucosa e IMC no son consideradas en el modelo? Describa detalladamente:

   - Compare con el árbol obtenido en la pregunta (1) considerando cantidad de niveles del árbol, cantidad de hojas del árbol obtenido, entre otros atributos.

     Para la obtención del nuevo árbol, se debe re diseñar la funcion anterior y ejecutar las siguientes líneas de código.

```
createData<-function(X, d, show){
    #Seteamos una semilla para replicar resultados
    set.seed(42)
    # En este caso utilizaremos un conjunto de
    entrenamiento con el X% de los datos
    train_size <- floor(X * nrow(d))
    #El tamano sera el X% de la cantidad de filas
    train_mask <- sample(seq_len(nrow(d)),size= train_size)
    #Mascara aleatoria
    # Separamos los conjuntos
    train <- data[train_mask, ]
    test <- data[-train_mask, ]
    # se crea el arbol
    t = tree(Diabetes ~ Nembarazada + Presion + Triceps
            + Insulina + Pedigree + Edad, data = train)

    if(show){
        # plot del arbol
        plot(t)
        text(t,pretty=1)
        # se muestra datos del arbol
        print(summary(t))
    }
    # variable retorno
    dat <- list("train" = train, "test" = test, "tree" = t)
    return(dat)
    }

 newDat2 <- createData(X=0.7, d=data, show=T)
```

     Esto retorna lo siguiente:

```
Classification tree:
tree(formula = Diabetes ~ Nembarazada + Presion + Triceps + Insulina +
    Pedigree + Edad, data = train)
Variables actually used in tree construction:
[1] "Edad"        "Triceps"     "Nembarazada" "Pedigree"    "Insulina"
Number of terminal nodes:  11
Residual mean deviance:  0.973 = 511.8 / 526
Misclassification error rate: 0.2626 = 141 / 537
```

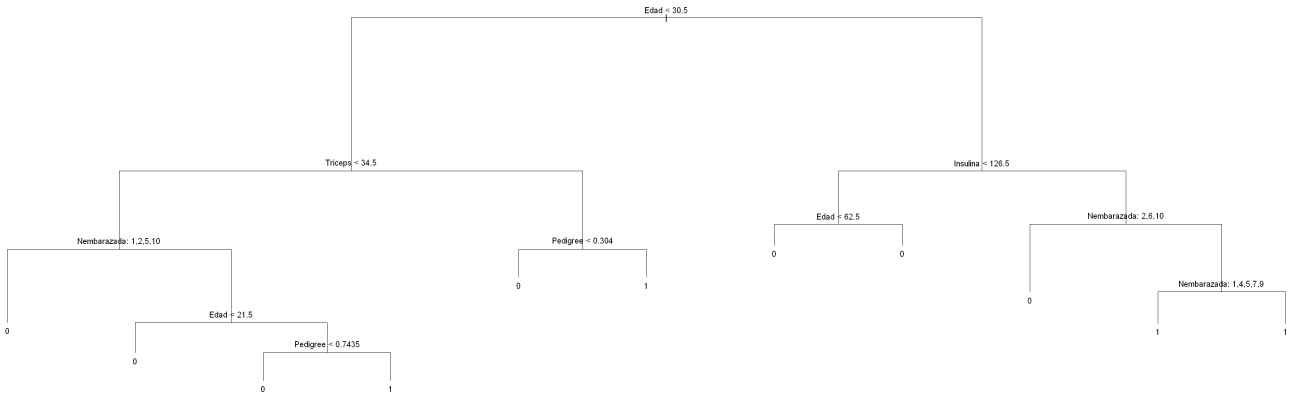

Figure 2: Árbol Resultante de Remover Glucosa e IMC

En este caso, se puede apreciar que el árbol es más pequeño y además, la cantidad de variables usadas en su construcción es menor, obviando la variable "Presión". Resultando en un mayor ratio de error de missclasification: 0.2626.

- Considerando el Testing set, ¿Cuál es la precisión del árbol?
  Para presentar la precisión del árbol, se debe ejectuar lo siguiente:

```
# Predecimos cada tier para los datos de pruebas
test <- newDat2$test
pred <- predict(newDat2$tree,test,type='class')
# Creamos la matriz de confusion
conf_matrix2 <- with(test,table(pred,test$Diabetes))
conf_matrix2
```

  Esto retorna la matriz de confusión al ejecutar el árbol con los datos de test:

```
pred   0   1
   0 135  60
   1  21  15
```

  Para el error:

```
# precision del modelo
acc <- sum(diag(conf_matrix2))/nrow(test)
miss_class_error2 <- 1 - acc
miss_class_error2
```

  Obteniendo la siguiente precisión:

```
0.350649350649351
```

3. Considerando el árbol de la pregunta (2), ¿ Cual sería el diagnóstico asignado al siguiente paciente?: (2,30,71,26,5.3, 27, 0.33, 40).

   Con el fin de diagnosticar, se recorre el arbol de la siguiente manera:

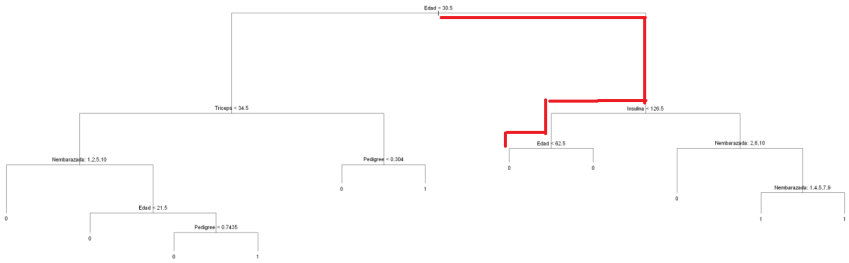

Figure 3: Árbol Recorrido Para Diagnóstico

Esto permite concluir que el/la paciente tiene diagnóstico negativo.