**Tecnológico Nacional de México**

**Instituto Tecnológico de Tijuana**

**Subdirección Académica**

**Departamento de Sistemas y Computación**

**SEMESTRE:**

Febrero-Julio 2021

**CARRERA:**

Ingeniería en Sistemas Computacionales e Ingeniería en Tecnologías de la Información y Comunicaciones

**MATERIA Y SERIE:**

Minería de Datos

BDD-1703 TI9A

**UNIDAD A EVALUAR:**

Unidad III

**NOMBRE DEL TRABAJO:**

**Práctica #3**

Documentación del programa Decision tree en R

**NOMBRE Y NÚMERO DE CONTROL DE LOS INTEGRANTES:**

Rodriguez Medrano Marco Antonio 17210635

**NOMBRE DEL DOCENTE:**

José Christian Romero Hernández

31 de mayo de 2021

Instrucciones: Se le pidió al alumno que documente la visualización de los datos ya previamente trabajados del programa decision tree en R y muestre los resultados obtenidos.

Antes de la visualización, se inició la sesión del repositorio del programa con el comando getwd (“ruta del repositorio del trabajo”) y se cargó el archivo .csv en la variable dataset posteriormente se seleccionó las columnas con las que trabajamos utilizando el comando dataset[3:5], con la cual seleccionara las columnas del 3 al 5.

Despues con el comando dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1)) convertimos a factor de la columna purchased a factores de 0 y 1, con ello pasaremos a cargar la librería caTools la cual nos ayudará a separar los datos en dos set’s los cuales son el set de entrenamiento que contrenda 300 datos de los 400 en total y el set de prueba el cual por consiguiente contendrá 100 datos esta separación tendrá un radio del 75%. Tras esto crearemos las escalas de los datos de los dos set’s para que posteriormente sean utilizados para hacer la clasificación decision tree y para hacerla utilizamos la librería rpart la cual nos ayudará a hacer la regresión tree y la partición recursiva.

Una vez hecho el clasificador ahora haremos la predicción con el siguiente comando y\_pred = predict(classifier, newdata = test\_set[-3]), pero hay un detalle si vemos los datos del comando anterior obtendremos lo siguiente:

0 1

2 0.96703297 0.03296703

4 0.96703297 0.03296703

5 0.96703297 0.03296703

9 0.96703297 0.03296703

12 0.96703297 0.03296703

… …………………………..

pero si sumamos ambos números nos daría 1 lo cual es un error porque nosotros queremos obtener como resultado una predicción de 0 o 1 osea si o no en binario, para lograrlo al comando le agregaremos el atributo , type = 'class' el cual nos permitirá obtener dicha predicción binaria:

2 4 5 9 12 18 19 20 22 29 32 34 35 38 45 46 48 52 66 69 74 75 82 84 85

0 0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0

86 87 89 103 104 107 108 109 117 124 126 127 131 134 139 148 154 156 159

…………………………………………………………………………………………………..

NOTA: Los números no binarios corresponden a la columna donde se ubica el valor binario

Ya por último para hacer la visualización de los resultados crearemos la matriz de confusión con el siguiente comando cm = table(test\_set[, 3], y\_pred), este mismo nos dará como resultado el porcentaje de eficacia y de error.

Lo marcado con verde es el porcentaje de eficacia el cual es del 83% y lo marcado con rojo es el porcentaje de fallo que es del 17%.

Ahora para la visualización volveremos a utilizar la librería ElemStatLearn que viene de un libro el cual lo pueden buscar en google, después señalaremos el set con el cual trabajaremos el cual en este caso es el set de entrenamiento posteriormente creamos dos variables X1 y X2 las cuales nos ayudaran mas adelante hacer los rangos de los límites en x, con el comando expand.grid expandimos las variables para después asignarles un nombre a las columnas, a continuación utilizando la predicción nos preparamos para plottear(grafica) asignando las etiquetas correspondientes

|  |
| --- |
| *# Visualising the Training set results* library(ElemStatLearn) set = training\_set X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01) X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01) grid\_set = expand.grid(X1, X2) colnames(grid\_set) = c('Age', 'EstimatedSalary') y\_grid = predict(classifier, newdata = grid\_set, type = 'class') plot(set[, -3],  main = 'Decision Tree Classification (Training set)',  xlab = 'Age', ylab = 'Estimated Salary',  xlim = range(X1), ylim = range(X2))  con el comando contour, agregaremos a la gráfica un contorno el cual será la matriz de confusión que calculamos anteriormente.  contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)    Pero como podemos ver con todo esto no podemos saber qué puntos pertenecen a qué región y mucho menos donde pertenece cada región, para ello aplicamos los siguientes comandos.  points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'springgreen3', 'tomato'))    este es para la región  points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))  y este para saber la clasificación de los puntos, para la visualización de los resultados del set de prueba se siguen los mismos pasos.  *# Visualising the Test set results* library(ElemStatLearn) set = test\_set X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01) X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01) grid\_set = expand.grid(X1, X2) colnames(grid\_set) = c('Age', 'EstimatedSalary') y\_grid = predict(classifier, newdata = grid\_set, type = 'class') plot(set[, -3], main = 'Decision Tree Classification (Test set)',  xlab = 'Age', ylab = 'Estimated Salary',  xlim = range(X1), ylim = range(X2)) contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE) points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'springgreen3', 'tomato')) points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))  Resultados          Pero solo plotteamos el clasificador, nos falta el arbol(tree) el cual se crea con el siguiente comando. |