



**National Technology of Mexico
Technological Institute of Tijuana**

ACADEMIC SUBDIRECTION
Systems and Computing Department

SEMESTER
February – June 2021

CAREER
Information and Communication Technologies Engineer

SUBJECT AND KEY:
Data Mining BDD-1703TI9A

STUDENT'S NAME AND REGISTRATION:
Camacho Manabe Juan Daniel 17210534
Velázquez Farrera César Alejandro 17212937

NAME OF THE JOB:
Practice 4

UNIT TO BE EVALUATED
Unit III

TEACHER'S NAME:
Mc. José Christian Romero Hernández

Práctica #4 Random Forest

Instrucciones

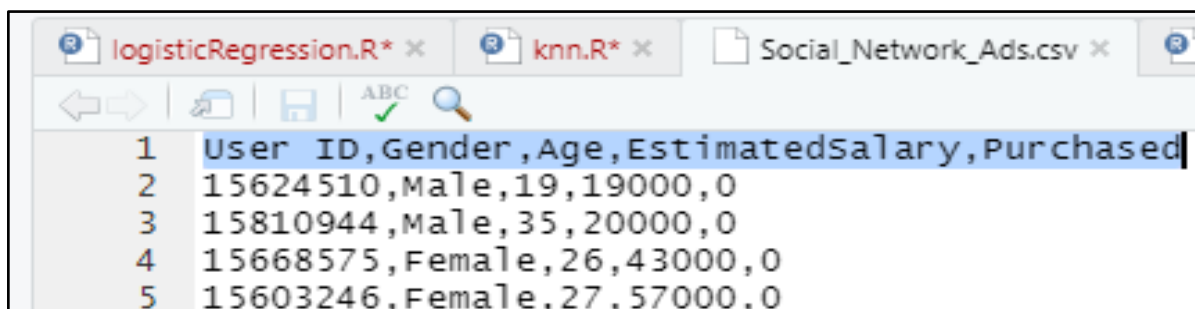
Hacer el análisis correspondiente al script R de bosque aleatorio, el cual debe ser documentado en su repositorio poniendo en este su resultados visuales y su descripción detallada de sus observaciones así como la fuente de código.

Podemos definir la ruta en la que queremos trabajar.

```
getwd()  
setwd("/home/chris/Documents/itt/Enero_Junio_2020/Mineria_de_datos/DataMining/MachineLearning/RandomForest")  
getwd()
```

Buscamos nuestro archivo de datos en la computadora con ayuda del explorador de archivos, si estamos trabajando en la carpeta donde se encuentra el archivo, simplemente lo cargamos con el nombre del archivo.

```
dataset = read.csv(file.choose())  
dataset = read.csv('Social_Network_Ads.csv')
```



	User ID	Gender	Age	EstimatedSalary	Purchased
1	15624510	Male	19	19000	0
2	15810944	Male	35	20000	0
3	15668575	Female	26	43000	0
4	15603246	Female	27	57000	0

Como el archivo de datos tiene columnas que no serán útiles, solo seleccionamos las columnas con las que queremos trabajar.

```
dataset = dataset[3:5]
```

\$ Age	: int	19	35	26	27	19	27	27	32	25	35	...
\$ EstimatedSalary:	int	19000	20000	43000	57000	76000	58000	84000	150000	330...		
\$ Purchased	: int	0	0	0	0	0	0	1	0	0	...	

Cargamos la librería “*caTools*” que cuenta con varias funciones para estadística. Con la función “*set.seed(n)*” sembramos una semilla de aleatoriedad donde “*n*” es el punto de partida utilizado en la generación de la secuencia de números aleatorios.

```
library(caTools)
set.seed(123)
```

A continuación dividimos los datos de la columna “*Purchased*”, con un ratio del 0.75, es decir, el 75% de los datos se tomará como verdadero y el 25% restante como falso. Posteriormente tomamos los valores de la división de los datos que fueron verdaderos y los empleamos para entrenamiento, el resto de los datos se etiquetaron como falsos serán empleados para probar la efectividad del modelo entrenado con los datos etiquetados como verdaderos.

```
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

Para el entrenamiento y la prueba es necesario normalizar los datos, entonces empleamos la función “*scale*”. La escala de características es un método que se utiliza para normalizar el rango de variables independientes o características de los datos. En el procesamiento de datos, también se conoce como normalización de datos y generalmente se realiza durante el paso de procesamiento previo de datos.

```
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])
```

Instalamos y cargamos la librería (“*randomForest*”), la cual contiene funciones de estadística.

```
install.packages('randomForest')
```

```
library(randomForest)
```

Adoptamos la máquina de soporte de vectores al conjunto de datos de entrenamiento, es decir, la clasificación con el método “svm” el cual se utiliza para ajustar el modelo. Un dato curioso es que hace uso de un kernel.

```
classifier = randomForest(x = training_set[-3], y = training_set$Purchased,  
ntree = 10)
```

Obteniendo la probabilidad de predecir clasificaciones correctas con los datos de prueba. Imprimimos la predicción. Evaluamos si la probabilidad de la predicción está entre 0.5, 1 y 0 e imprimimos la probabilidad.

```
y_pred = predict(classifier, newdata = test_set[-3])  
y_pred
```

Guardamos los datos resultantes en una tabla con los datos de prueba y las predicciones correspondientes, es decir, creamos una matriz de confusión. Luego la imprimimos.

```
cm = table(test_set[, 3], y_pred)  
cm
```

Para continuar necesitaremos un paquete especial que contiene la librería “ElemStatLearn”, dado que el paquete está obsoleto para las nuevas versiones de r y no viene por defecto en las herramientas de rStudio, es necesario descargarlo e instalarlo manualmente. El archivo se encontrara en la siguiente liga: <https://cran.r-project.org/src/contrib/Archive/ElemStatLearn/> , la instalacion se realizara con el siguiente comando, sustituyendo la dirección del archivo que sea conveniente.

```
install.packages("~/home/daniel-  
camacho/Descargas/ElemStatLearn_2015.6.26.tar.gz", repos=NULL,  
type="source")
```

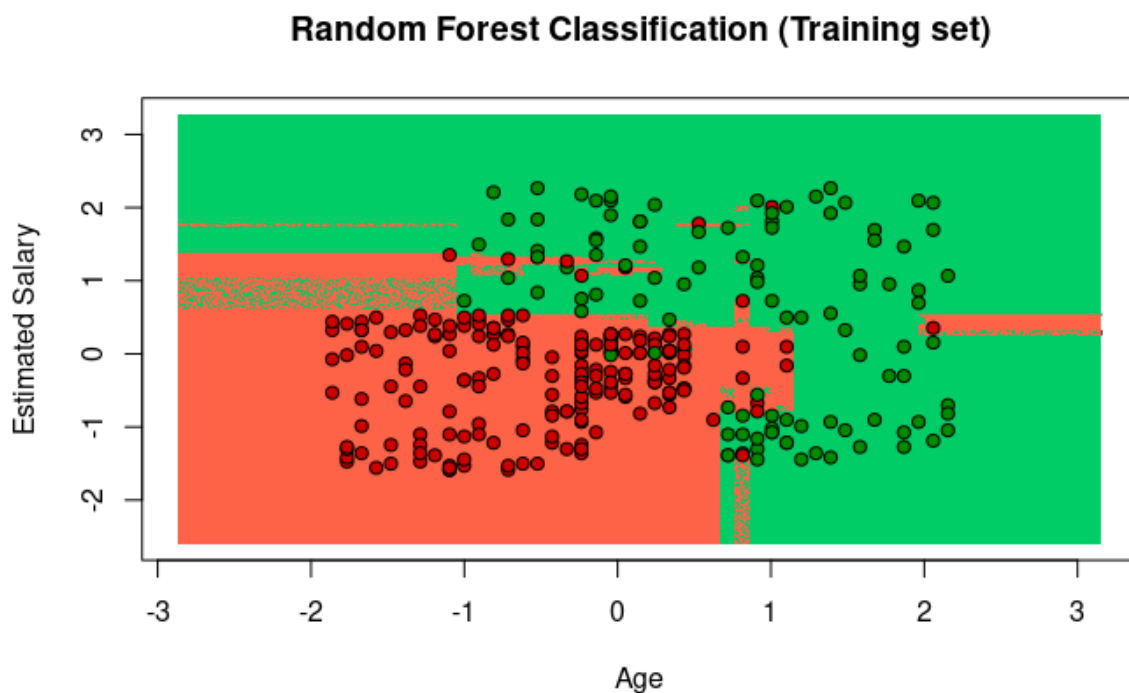
1. Una vez instalado el paquete, cargamos la librería “*ElemStatLearn*”.
2. Con el comando “*set*” tomamos los datos de entrenamiento, después creamos dos variables “*x1*”, “*x2*” en donde empleamos la función “*seq*” (secuencia), definiremos un rango mínimo y máximo, le sumaremos un entero positivo y lo multiplicaremos por la longitud de “0.01”
3. A continuación definimos una cuadrícula que expandimos usando las dimensiones definidas en “*x1*” y “*x2*”.
4. Definimos la columnas de la cuadrícula mediante un vector, las cuales serán año y salario estimado.
5. A continuación probamos el modelo de regresión logística pero con la cuadrícula que contiene las columnas que definimos.
6. Graficamos los datos con un máximo de -1, colocamos título a la gráfica que está dentro de (cuadrícula), colocamos el nombre a los ejes, asignamos los límites para los ejes manteniendo los datos dentro de las dimensiones de la cuadrícula.
7. Colocamos un contorno y añadimos un rango de números condicionado al rango numérico mínimo y máximo de la cuadrícula.
8. Agregamos puntos a la gráfica comparando los datos de año y salarios estimado con las predicciones del modelo de regresión logística, en donde si son diferentes se le asignará un color rojo y si son verdaderos uno verde, en el caso de los errores se colocarán puntos rojos dentro del área verde que se ha definido como conjuntos verdaderos y por el contrario un punto verde en el área definida como falso.

```
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, grid_set)
plot(set[, -3],
      main = 'Random Forest Classification (Training 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'))

```



De la misma forma que en código anterior se hace la evaluación y clasificación de los datos, solo que esta vez, para la gráfica no serán los datos de entrenamiento sino los de prueba, por lo cual en el comando “set” se asignan los datos de la variable “test_set” obtenida en la división de los datos correspondiente al 25% de la totalidad del dataset que contiene las tres columnas iniciales.

```

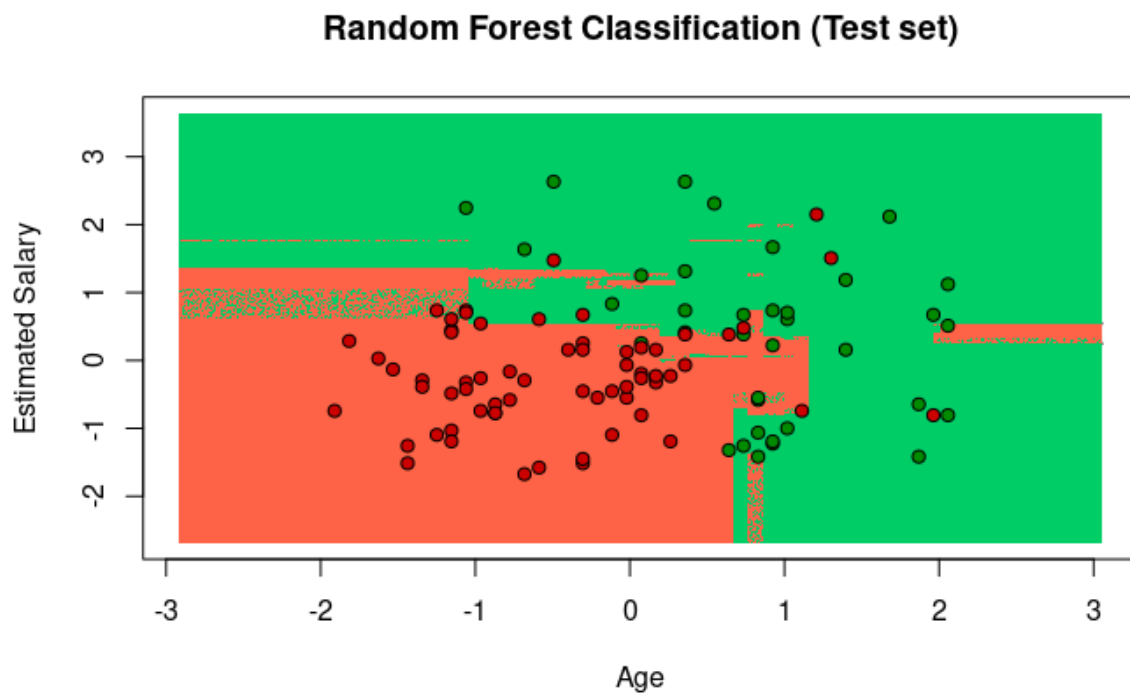
library(ElemStatLearn)
set = test_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

```

```

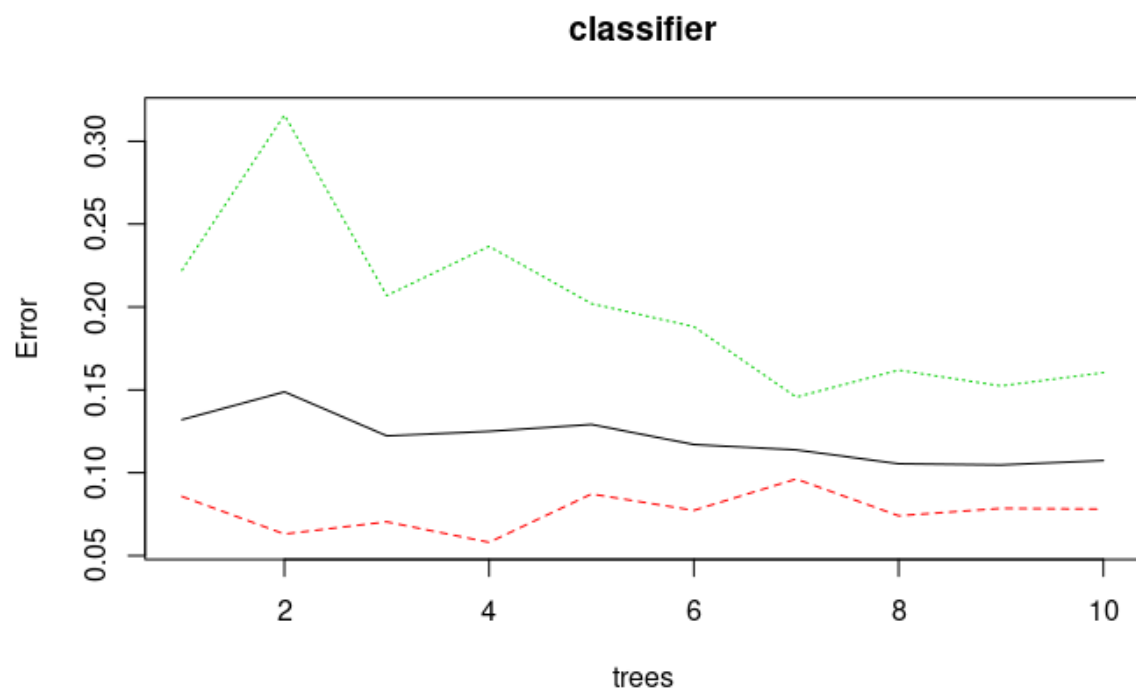
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, grid_set)
plot(set[, -3], main = 'Random Forest Classification (Test set)',
      xlab = 'Age', ylab = 'Estimated Salary',
      xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), 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'))

```



Graficamos el número de árboles y el error que le corresponde de acuerdo al ajuste de los datos.

```
plot(classifier)
```



Conclusiones y observaciones

En base a las observaciones, el primer método es el más sencillo de implementar en términos de sintaxis y comprensión del lenguaje R. Sin embargo, al momento de representar los resultados de las predicción (utilizando la matriz de confusión y), no es fácil de entender a primera vista para los usuarios que desconocen el tema.

Con los resultados obtenidos con las predicciones, se puede concluir que las columnas de edad ("age") tiene mayor correlación que las columnas de salario estimado ("EstimatedSalary") y Compras Realizadas ("Purchased"). Esto es evidente en las gráficas de puntos con el método "randomForest" por la línea, que representa el ajuste, aunque en dos dimensiones no se aprecia de la manera que se vería en tercera dimensión.

El segundo método, con el uso de la librería "ElemStatLearn" puede generar una gráfica mucho más fácil de comprender, a coste de la sintaxis compleja. Se puede observar en la segunda gráfica del segundo método, los puntos rojos representa a las personas que no compraron el producto y de verde las que sí.

Podemos concluir, que entre más edad tenga el cliente potencial, más probabilidades de vender el producto tiene. A diferencia de que el cliente sea más joven, existen menos posibilidades de que adquiera el producto.

Bibliografía

1. jcromerohdz. (2020). LogisticRegression. 29/05/30, de GitHub Sitio web:
<https://github.com/jcromerohdz/DataMining/tree/master/MachineLearning/RandomForest>
2. https://cran.archive.r-project.org/web/checks/2020/2020-01-28_check_results_ElemStatLearn.html
3. <https://cran.r-project.org/src/contrib/Archive/ElemStatLearn/>
4. <https://riptutorial.com/r/example/5556/install-package-from-local-source>
5. Cánovas-García, F., Alonso-Sarría, F., & Gomariz-Castillo, F. (2016). Modificación del algoritmo Random Forest para su empleo en clasificación de imágenes de Teledetección. In *Aplicaciones de las Tecnologías de la Información Geográfica (TIG) para el desarrollo económico sostenible XVII Congreso Nacional de Tecnologías de Información Geográfica, Málaga* (Vol. 29, No. 30, pp. 359-368).
6. Flores, A., Maldonado, S., & Weber, R. (2015). Selección de atributos y Support Vector Machine adaptado al problema de fuga de clientes. *Revista Ingeniería de Sistemas*, 29.
7. Saltos, V. A. A., & Flores, P. (2018). Comparativa entre classification trees, random forest y gradient boosting; en la predicción de la satisfacción laboral en Ecuador. *Ciencia Digital*, 2(4.1.), 42-54.
8. Dfuf, I. A. (2018). *Análisis de Sensibilidad Mediante Random Forest* (Doctoral dissertation, Ph. D. thesis, Universidad Politécnica de Madrid).
9. Flores, J. B. A., & Aruner, M. E. P. (2018). Random Forest para identificar los factores sociodemográficos asociados al uso de Internet en el Perú.