**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 #5**

Documentación del programa Support Vector Machine(SVM) en R

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

Rodriguez Medrano Marco Antonio 17210635

**NOMBRE DEL DOCENTE:**

José Christian Romero Hernández

**Tijuana, Baja California jueves 03 de Junio de 2021**

**Instrucciones:** Se le pidió al alumno que documente la parte del código que trabaja la visualización del programa svm y muestre los resultados

Una vez ya instanciado la ubicación de la carpeta de trabajo(repositorio), cargado y elegido los datos con los que trabajaremos. Utilizamos la library(caTools) para dividir los datos cargados(400) en dos set’s uno de entrenamiento(300) y el otro de prueba(100) a los cuales les creamos unas escalas para poder darles un mejor sentido a los datos y ayudarnos en la creación del método de clasificación svm que esta misma antes debemos instalar el paquete e1071 para poder crear el método.

|  |
| --- |
| 1. install.packages("e1071", dep = TRUE) 2) library(e1071)   classifier = svm(formula = Purchased ~ .,  data = training\_set,  type = 'C-classification',  kernel = 'linear') |

podemos observar primero que instanciamos el id classifier y le asignamos la función svm que esta se utiliza para crear las funciones de regresión y clasificador, pero nosotros nos enfocaremos en el clasificador, lo siguiente sería instanciar la palabra fórmula asignándole Purchased que es el nombre de una de nuestras columnas y le diremos ~ ., que vamos a tomar toda la columna y todos los renglones con data le indicaremos que utilizaremos el set de entrenamiento para la fórmula y por último en la fórmula nos pide un type y el kernel, como vamos a crear un clasificador ocupamos que el type sea de tipo clasificador y el kernel lineal. Una vez creado el clasificador podemos ver la fórmula con el comando:

|  |
| --- |
| y\_pred = predict(classifier, newdata = test\_set[-3]) > y\_pred  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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   86 87 89 103 104 107 108 109 117 124 126 127 131 134 139 148 154 156 159 162 163 170 175 176 193   0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  199 200 208 213 224 226 228 229 230 234 236 237 239 241 255 264 265 266 273 274 281 286 292 299 302   0 0 1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1  305 307 310 316 324 326 332 339 341 343 347 353 363 364 367 368 369 372 373 380 383 389 392 395 400   0 1 0 0 0 0 1 0 1 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 0 |

El siguiente comando lo utilizamos para la creación de la matriz de confusión la cual nos da el porcentaje de eficacia y de error de nuestra predicción.

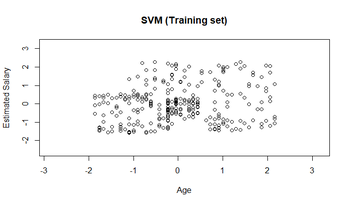
|  |
| --- |
| cm = table(test\_set[, 3], y\_pred) > cm  y\_pred   0 1  0 57 7  1 13 23 |

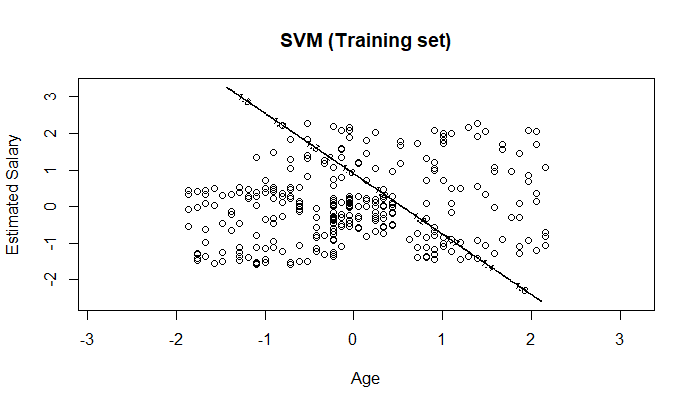
Si sumamos el 57 y 23 nos daría una eficacia del 80% lo otro por lógica es el 20% de error.

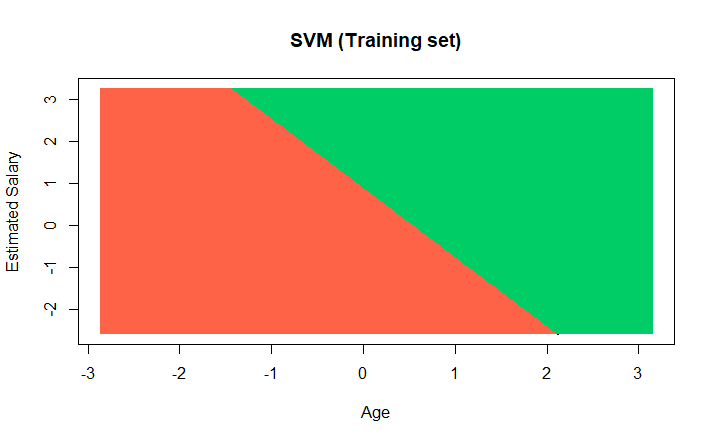
Una vez ya hecho todo esto ya estamos listos para crear la visualización de los datos, para ella utilizamos la librería ElemStatLearn que esta fue creada por medio de un libro el cual lo pueden googlear con el nombre de la librería, una vez instanciado la librería cargamos los datos que vamos a visualizar en este caso serán los de entrenamiento y posteriormente creamos dos variables X1 y X2 las cuales nos ayudarán a limitar el rango de la gráfica en la coordenada x, después las expandimos con el comando expand.grid(X1, X2) y a continuación agregamos los nombres de las columnas de la gráfica, agregamos la predicción en una variable nueva y proseguimos a crear el plot utilizando del set todas la columnas pero solo -3 renglones, agregamos el nombre del plot así como label’s de las coordenadas y el límite en la coordenada x , pero la grafica no nos muestra un resultado congruente por lo que con el comando contour agregamos línea que nos dividirá el espacio de la gráfica para ver mejor los resultados.

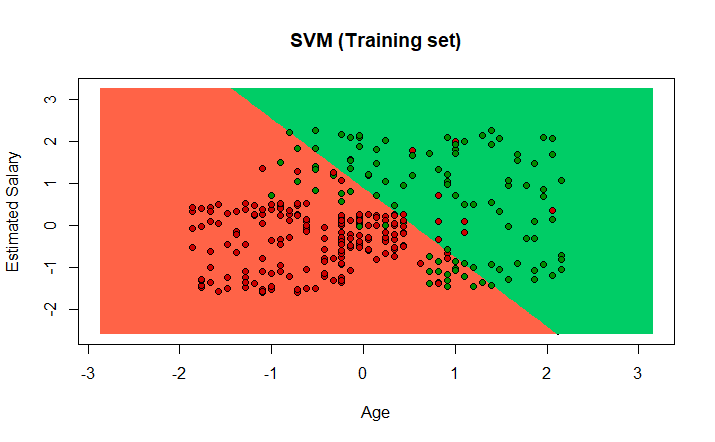
Y por último utilizaremos dos veces el comando points el cual nos ayudará a darle color primero al espacio de la gráfica y el segundo para darle color a los puntos así veremos mejor los datos asi podremos observar su comportamiento de mejor manera.

|  |
| --- |
| *# 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) plot(set[, -3],  main = 'SVM (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')) |









Repetimos los mismos pasos pero ahora con el set de prueba.

*# 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)  
plot(set[, -3], main = 'SVM (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'))

