

INSTITUTO TECNOLOGICO DE TIJUANA

CARRERA

INGENIERÍA EN SISTEMAS COMPUTACIONALES

MATERIA
MINERÍA DE DATOS
TAREA

PRÁCTICA #4, UNIDAD #3

FECHA ENTREGA
23/05/2022

ALUMNO(A)

HOWARD HERRERA ERWIN #18210716
PÉREZ LÓPEZ ALICIA GUADALUPE #18210514

DOCENTE

JOSE CHRISTIAN ROMERO HERNANDEZ

Import of the dataset and we will specify the range of columns that we will use (3 to 5)

```
# Importing the dataset
dataset = read.csv('Social_Network_Ads.csv')
dataset = dataset[3:5]
```

We change the value of the “Purchased” column as a factor type to be able to use it later

```
# Encoding the target feature as factor
dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))
```

.
We divide the dataset into 2 parts, training and testing with a ratio of 0.75, and the seed is specified to have randomness.

```
# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

We will scale the training and test values to model them for use in the KNN algorithm.

```
# Feature Scaling
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])
```

. We send the test and training data to the KNN algorithm, in addition, the variable k is set as 5 to seek to obtain a better classification

```
# Fitting K-NN to the Training set and Predicting the Test set
results
library(class)
y_pred = knn(train = training_set[, -3],
```

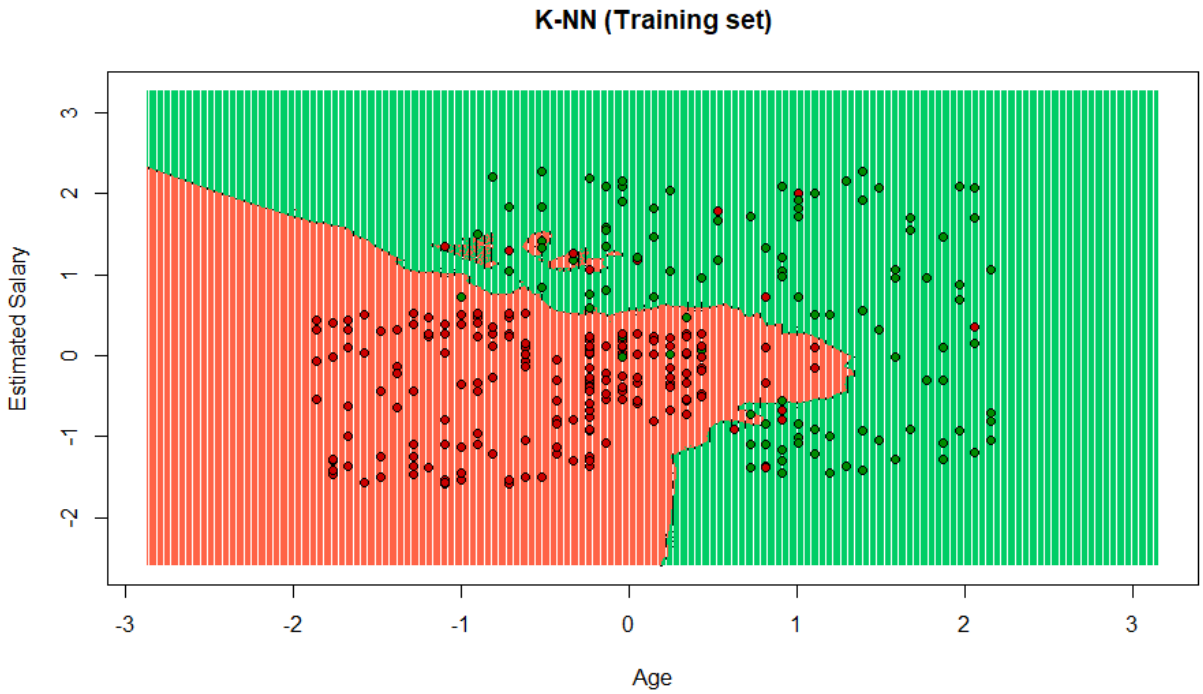
```
             test = test_set[, -3],
             cl = training_set[, 3],
             k = 5,
             prob = TRUE)
```

We make the confusion matrix using the predictor variable created from the knn algorithm and sending the test data.

```
# Making the Confusion Matrix
cm = table(test_set[, 3], y_pred)
```

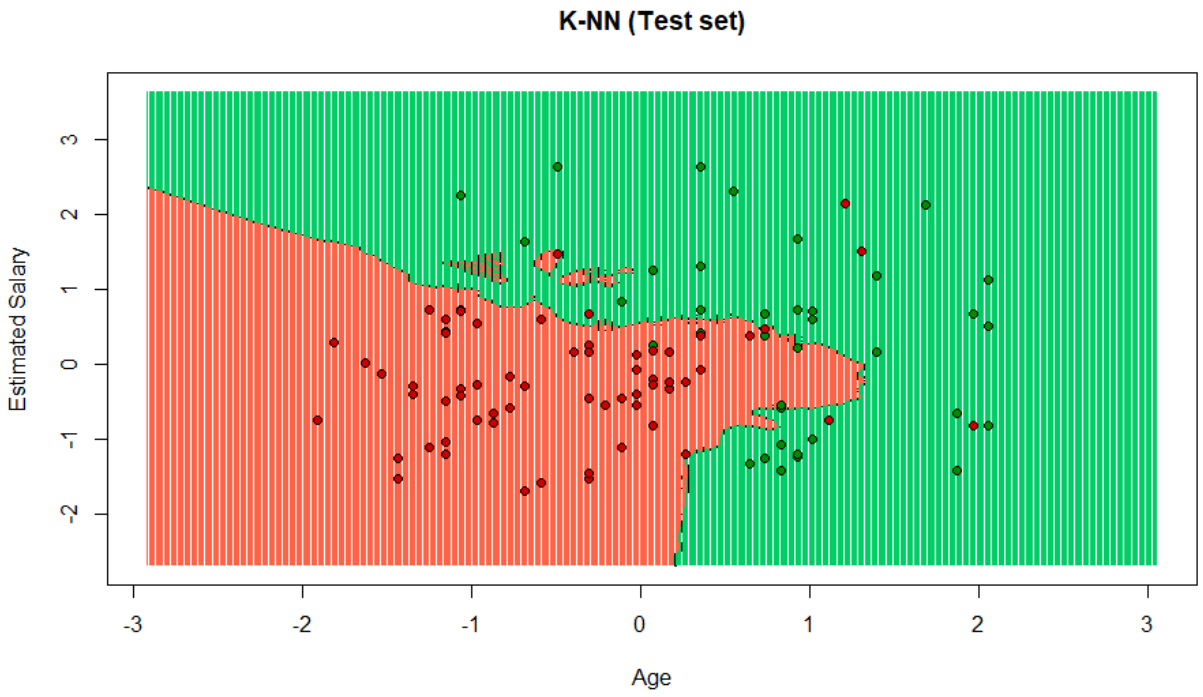
Visualization of the results

```
# Visualising the Training set results
install.packages('ElemStatLearn')
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 = knn(train = training_set[, -3], test = grid_set, cl =
training_set[, 3], k = 5)
plot(set[, -3],
     main = 'K-NN (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'))
```



Visualization of the results using the test values of the algorithm.

```
# 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 = knn(train = training_set[, -3], test = grid_set, cl =
training_set[, 3], k = 5)
plot(set[, -3],
     main = 'K-NN (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'))
```


K-NN (Test set)
Estimated Salary
Age
-3
-2
-1
0
1
2
3
-2
-1
0
1
2
3