INFORME - ELECTIVA AREA ELECTRONICA

Santiago Avila, Andres Hernandez, Andres Montenegro, Brayan Zambrano, Nikolas Riapira

2023-05-28

Supervised Methods in Machine Learning

3rd report guideline: "Supervised learning final project"

El link al repositorio público de Github es el siguiente: https://github.com/AndresFHernandezJ/AssignmentThree

Definicion del problema

Los robots móviles son una solución para la exploración de ambientes hostiles, como lo pueden ser, por ejemplo, ambientes radiactivos o tóxicos, donde la intervención humana directa no es posible. De esta manera se construyó un robot móvil y 3 ambientes diferentes caracterizados por condiciones, las cuales puedan ser sensadas por el robot según los sensores planteados para el mismo.

Para la adquisición de datos se hizo uso de un robot móvil que contó con tres sensores, el sensor de distancia ultrasónico HC-SR04, el sensor de color (RGB)TCS3200 (el cual generaba 3 datos, uno por cada color) y una fotorresistencia para la medición de la intensidad de la luz incidente.

Figure 1: HC-SR04 (Imagen tomada de https://ja-bots.com/producto/sensor-hc-sr04-ultrasonico/)

De esta manera se obtuvo un dataset de 510 observaciones y 6 variable, la primera de ellas fue el valor obtenido por el sensor ultrasónico, la segunda el valor de la fotorresistencia, la tercera el valor correspondiente a la intensidad del color rojo, la cuarta con respecto a la intensidad del color verde, la cuarta relacionada a la intensidad del color azul y por ultimo la variable objetivo, el ambiente.

A continuación se muestra el robot móvil con los sensores el cual mediante programación en el software de Arduino hizo la toma de los datos.

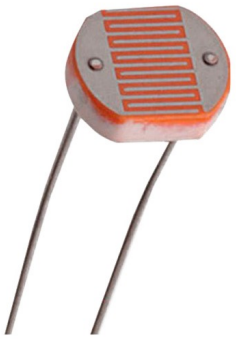


Figure 2: Fotorresistencia (Imagen tomada de https://laelectronica.com.gt/extras/que-es-un-ldr-y-como-funciona

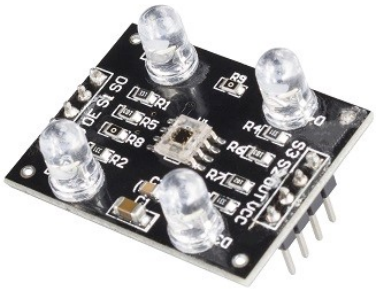
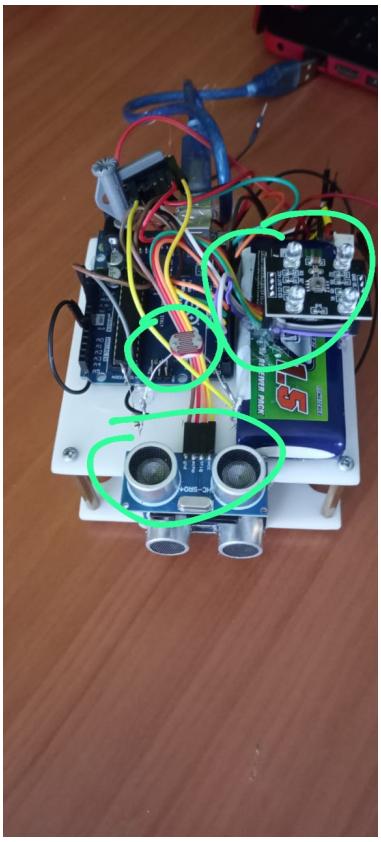


Figure 3: (RGB)TCS3200 (Imagen tomada de http://wiki.sunfounder.cc/index.php?title=TCS3200_Color_Sensor_Module



En la siguiente imagen se presentan los 3 ambientes junto con el robot móvil.

Figure 4: Robot movil

Analisis exploratorio de datos EDA

No existió un pre procesamiento de los datos con respecto al dataset inicial debido a su estructura, el análisis de datos exploratorio se realizó de la siguiente manera.

El dataset de manera resumida se presenta a continuación.

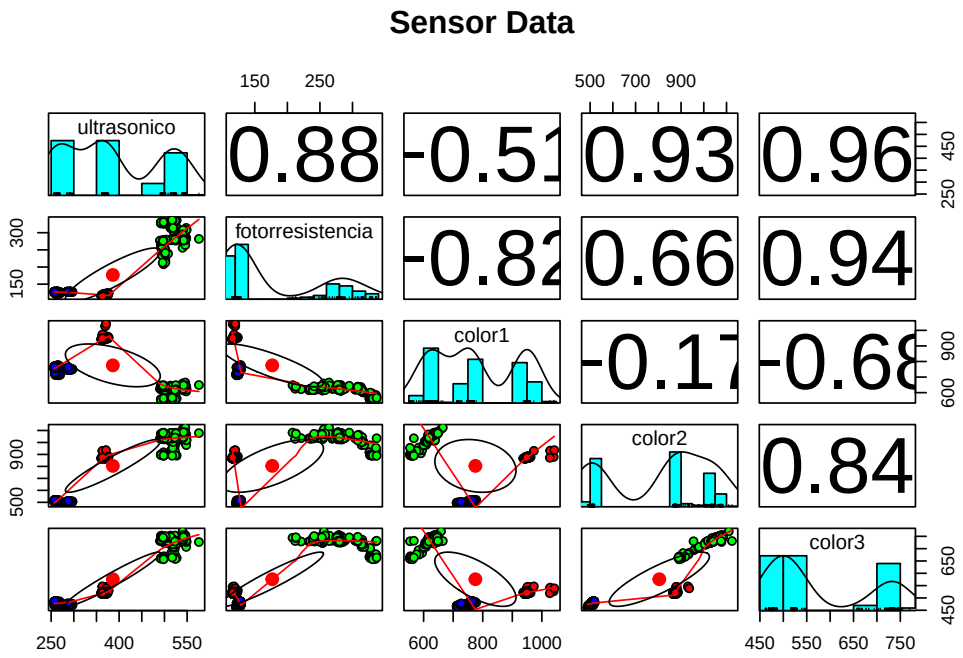
```
##   ultrasonico fotorresistencia color1 color2 color3 ambiente
## 1         286              128    750    502    469      UNO
## 2         292              128    748    508    470      UNO
## 3         292              128    747    502    476      UNO
## 4         292              128    748    502    476      UNO
## 5         292              128    753    507    470      UNO
## 6         291              128    749    507    469      UNO
```

Del cual se puede identificar que cada una de las variables es de tipo int, a excepción de la última la cual es de tipo character y mas o menos se ve la estructura de sus datos, también es posible obtener datos estadísticos como los siguientes.

```
##   ultrasonico    fotorresistencia     color1          color2
##  Min.   :257.0   Min.   :115.0    Min.   : 553.0   Min.   : 488.0
##  1st Qu.:269.2   1st Qu.:119.0    1st Qu.: 634.0   1st Qu.: 513.2
##  Median :369.0   Median :127.0    Median : 760.0   Median : 873.0
##  Mean   :386.1   Mean   :176.8    Mean   : 775.2   Mean   : 804.3
##  3rd Qu.:502.0   3rd Qu.:278.0    3rd Qu.: 946.0   3rd Qu.:1026.8
##  Max.   :576.0   Max.   :337.0    Max.   :1042.0   Max.   :1124.0
##      color3        ambiente
##  Min.   :461.0   Length:510
##  1st Qu.:482.0   Class :character
##  Median :522.0   Mode  :character
##  Mean   :575.9
##  3rd Qu.:727.0
##  Max.   :770.0
```

De esta manera se observa la estructura del dataset, los datos que se obtuvieron con cada uno de los 3 sensores, se observan los valores mínimos, máximos y sus respectivos valores del 25%, 50%, y 75% de los datos.

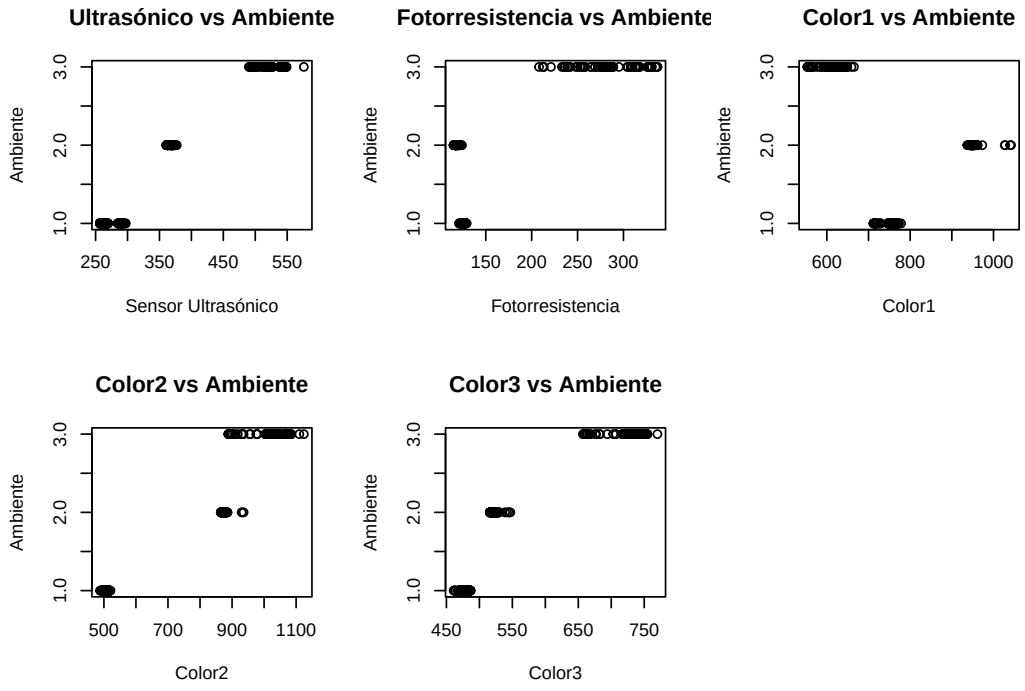
A continuación se muestra la gráfica relacionando cada una de las variables.



El gráfico anterior es una matriz de dispersión que muestra en su diagonal un histograma de cada variable individual y los gráficos de dispersión entre pares de variables así como la correlación que hay entre ellas.

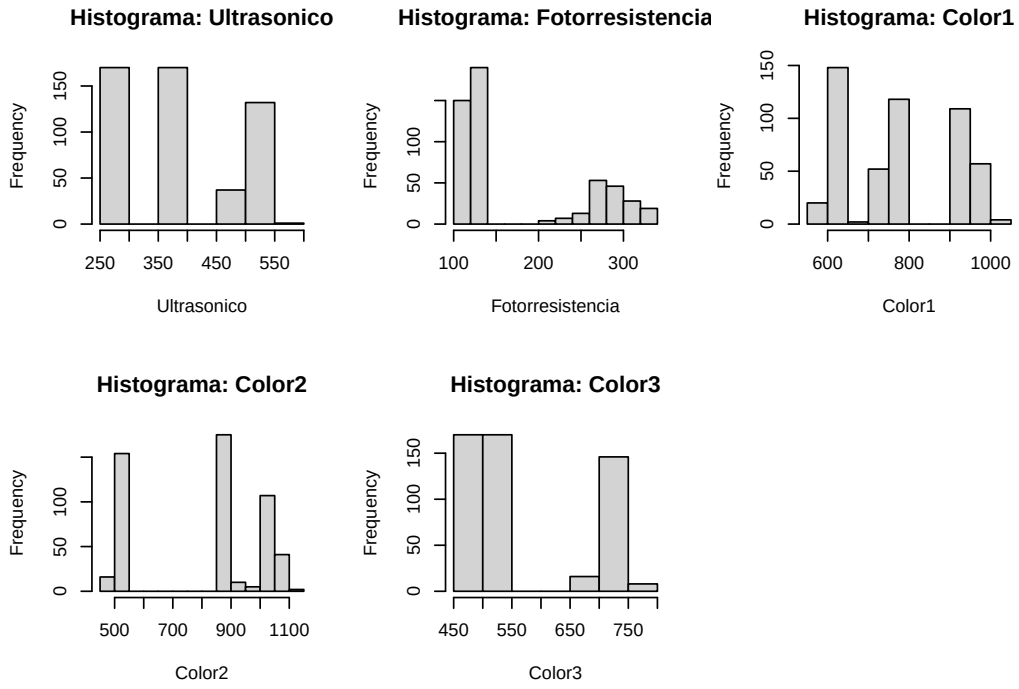
A continuación se muestran los gráficos de dispersión entre cada una de las variables predictoras vs la variable

objetivo.



Se observa la relación de dispersión que hay entre cada uno de los sensores con respecto al ambiente en el que fueron tomados los datos, se alcanza a distinguir in tipo de dispersión similar entre los mismos.

A continuación se presentan los histogramas con respecto a la misma relación variable predictora vs variable objetivo.



Métodos de clasificación

A continuación se presentan los métodos de clasificación, algo a tener en cuenta es que inicialmente se genera el siguiente código donde se asigna el dataset a al dataframe Dataset1, seguid a esto se asigna dicho dataframe a uno nuevo donde la variable ambiente se convierte a factor con 3 niveles correspondientes a cada ambiente, "UNO", "DOS" y "TRES".

```
Dataset1 <- read.csv("D:/Usr/Desktop/Andres/8vo Semestre/DataScience/AssignmentThree/Datasets/DATA_SET_
Dataset1_df <- Dataset1
Dataset1_df$ambiente <- factor(Dataset1_df$ambiente, levels = c("uno", "dos", "tres"))
Dataset1_norm <- Dataset1_df
variables <- c("ultrasonico", "fotorresistencia", "color1", "color2", "color3")
#normalizacion z-score
Dataset1_norm[, variables] <- scale(Dataset1_norm[, variables])
```

Como se observa, se genera un nuevo dataframe llamado Dataset1_norm el cual toma los datos del dataset con la variable factor y los normaliza por el método de Z-Score, finalmente se establecen las variables correspondientes a cada sensor.

Resultando de la siguiente manera:

```
##   ultrasonico fotorresistencia     color1    color2     color3 ambiente
## 1  -0.9552447       -0.6206016 -0.1837503 -1.370725 -0.9709900     <NA>
## 2  -0.8979828       -0.6206016 -0.1983110 -1.343519 -0.9619082     <NA>
## 3  -0.8979828       -0.6206016 -0.2055913 -1.370725 -0.9074172     <NA>
## 4  -0.8979828       -0.6206016 -0.1983110 -1.370725 -0.9074172     <NA>
## 5  -0.8979828       -0.6206016 -0.1619093 -1.348053 -0.9619082     <NA>
## 6  -0.9075264       -0.6206016 -0.1910306 -1.348053 -0.9709900     <NA>
```

KNN K-Nearest Neighbors (Vecinos más cercanos)

El método KNN es un método de aprendizaje automático supervisado que se utiliza para la clasificación de datos. La idea inicial de KNN es que los puntos de datos similares tienden a agruparse juntos en el espacio. Por lo tanto, clasifica nuevos puntos de datos con respecto a la clase de los puntos de datos cercanos.

```
sample.indexA <- sample(1:nrow(Dataset1_norm)
                        ,nrow(Dataset1_norm)*0.7
                        ,replace = F)

predictorsA <- c("ultrasonico", "fotorresistencia", "color1", "color2", "color3")

train.dataA <- Dataset1_norm[sample.indexA
                          ,c(predictorsA,"ambiente")
                          ,drop=F]

test.dataA <- Dataset1_norm[-sample.indexA
                          ,c(predictorsA,"ambiente")
                          ,drop=F]
```

El código anterior es un fragmento de código que se mostrará en cada uno de los modelos debido a que sigue la metodología explicada a continuación que se repite en cada modelo.

1. Inicialmente se crea un sample.index el cual es una muestra de 70% de los datos del dataset de manera aleatoria

2. Se establecen las variables predictoras, es decir, los sensores y sus valores.

3. Se genera un dataset de entrenamiento compuesto por dicho 70% de los datos aleatorios obtenidos en el primer punto, este se basa en la variable objetivo con respecto a los predictores.

4. Igualmente se genera un segundo dataset de prueba compuesto por el 30% restante de los datos aleatorios, tambien basado en la variable objetivo con respecto a los predictores. A continuación se muestra el código de Knn donde se explica cada linea relevante con un comentario.

KNN CODIGO

```
ctrl <- trainControl(method="cv", p=0.7) #Configuración cross-validation 70-30
knnFitAmbiente <- #Modelo
  train(ambiente ~ ultrasonico+fotorresistencia+color1+color2+color3 #Entrenamiento ambiente en función
                , data = train.dataA #A partir de los datos de entrenamiento (70%)
                , method = "knn" # Método Knn
                , trControl = ctrl #Configuración establecida
                , preProcess = c("center","scale")#Normalización Z-Score
                , tuneLength = 20)

#Salida del modelo
knnFitAmbiente

#Predicciones a partir de los datos de prueba (30%)
knnPredict <- predict(knnFitAmbiente, newdata = test.dataA)
```

```r
#Matriz de confusión basada en la predicción
confusionMatrix(knnPredict, test.dataA$ambiente)
#Cross table similar a la matriz
CrossTable(test.dataA$ambiente, knnPredict, prop.chisq = FALSE)


saveRDS(knnFitAmbiente, "knn_model.rds")#Codigo para guardar el modelo entrenado.
```

El cual genera la siguiente información donde es posible observar los datos correspondientes al sample, los 5 predictores, las 3 clases, el pre procesamiento (Z-Score) y las cross validación correspondiente a 10 fold.

Tambien se observa la alta precisión obtenida, practicamente perfecta, con un valor de 1.

```
## k-Nearest Neighbors
##
## 357 samples
##   5 predictor
##   3 classes: 'UNO', 'DOS', 'TRES'
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 321, 321, 321, 322, 321, 322, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy  Kappa
##    5  1         1
##    7  1         1
##    9  1         1
##   11  1         1
##   13  1         1
##   15  1         1
##   17  1         1
##   19  1         1
##   21  1         1
##   23  1         1
##   25  1         1
##   27  1         1
##   29  1         1
##   31  1         1
##   33  1         1
##   35  1         1
##   37  1         1
##   39  1         1
##   41  1         1
##   43  1         1
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 43.
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction UNO DOS TRES
##       UNO   45   0    0
##       DOS    0  59    0
##       TRES   0   0   49
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9762, 1)
##     No Information Rate : 0.3856
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: UNO Class: DOS Class: TRES
## Sensitivity              1.0000     1.0000      1.0000
## Specificity              1.0000     1.0000      1.0000
## Pos Pred Value           1.0000     1.0000      1.0000
## Neg Pred Value           1.0000     1.0000      1.0000
## Prevalence               0.2941     0.3856      0.3203
## Detection Rate           0.2941     0.3856      0.3203
## Detection Prevalence     0.2941     0.3856      0.3203
## Balanced Accuracy        1.0000     1.0000      1.0000
```

Finalmente se muestra la matriz de confusión y sus estadísticas donde también se presenta una predicción perfecta.

Al finalizar y haber guardado el modelo en el archivo .rds, se generaron (para cada uno de los modelos) un nuevo Script para tratar nuevos datos, el cual carga el modelo guardado, obtiene los nuevos, los normaliza y finalmente realiza las predicciones y genera los resultados obtenidos con los nuevos datos.

A continuación se muestra el código correspondiente.

```
#Se carga el modelo guardado en la variable NewKnnFit
NewknnFitAmbiente <- readRDS("knn_model.rds")
#Se asignan los nuevos datos a la variable nuevos_datos1_norm
nuevos_datos1_norm <- PruebaNuevos
#Se normalizan los datos a través de Z-Score
nuevos_datos1_norm[, variables] <- scale(nuevos_datos1_norm[, variables])
#Se establece la variable ambiente como factor y se organizan los niveles
nuevos_datos1_norm$ambiente <- factor(nuevos_datos1_norm$ambiente, levels = c("UNO", "DOS", "TRES"))
#Se realizan las nuevas predicciones
predicciones1 <- predict(NewknnFitAmbiente, newdata = nuevos_datos1_norm)
#se verifican los resultados
CrossTable(nuevos_datos1_norm$ambiente, predicciones1, prop.chisq = FALSE)
```

Logistic Regression

La Regresión Logística es un método utilizado para predecir una variable categórica binaria (como "Sí/No") en función de un conjunto de variables predictoras. Se basa en el modelo logit, que relaciona las probabilidades de la variable dependiente con las variables independientes.

Como se explicó en el modelo de KNN el código inicial de la configuración de los datos entre entrenamiento y prueba es el mismo y maneja la misma metodología anteriormente explicada.

A continuación se muestra el código de logistic regression donde se explica cada linea relevante con un comentario, se debe tener en cuenta que, debido a que la variable ambiente cuenta con mas de 2 niveles, se establece un metodo multinomial y no binomial.

Logistic Regression Codigo

```
#Igualmente se toma las muestras aleatorias y se divide en datos de entrenamiento (70%) y datos de prue
#####################################################################
Sample.indexA2 <- sample(1:nrow(Dataset1_norm)
                         ,nrow(Dataset1_norm)*0.7
                         ,replace = F)

predictorsA2 <- c("ultrasonico", "fotorresistencia", "color1", "color2", "color3")

train.dataA2 <- Dataset1_norm[sample.indexA2
                              ,c(predictorsA,"ambiente")
                              ,drop=F]

test.dataA2 <- Dataset1_norm[-sample.indexA2
                             ,c(predictorsA,"ambiente")
                             ,drop=F]

#####################################################################
#Se crea la configuración para el modelo, Repeated cross-validation con 20 repeticiones, 10 folds
fit.control <- trainControl(method = "repeatedcv"
                            ,number = 10, repeats = 20)
#Se entrena el modelo, ambiente en funcion de los valoes de los sensores
LRfit <- train(ambiente ~ ultrasonico + fotorresistencia + color1 + color2 + color3
                       ,data = train.dataA2 #Datos de entrenamiento
                       ,method = "multinom" #Metodo multinomial debido a los 3 niveles de la variable obj
                       ,trControl = fit.control #Asigna la configuracion establecida
                        , trace = FALSE)
#Se llama el modelo
LRfit
#Se realizan las predicciones con los datos de prueba de tipo probabilistico
LRPredict <- predict(LRfit, test.dataA2, type = "prob")
#Se llama a las predicciones
LRPredict
#Resumen de los resultados
summary(LRfit)

saveRDS(LRfit, "LR_model.rds")#Se guarda el modelo entrenado
```

El modelo LRfit mostro la siguiente información:

```
## Penalized Multinomial Regression
##
## 357 samples
##   5 predictor
##   3 classes: 'UNO', 'DOS', 'TRES'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 20 times)
## Summary of sample sizes: 322, 321, 321, 322, 320, 320, ...
## Resampling results across tuning parameters:
##
##   decay  Accuracy  Kappa
##   0e+00  1         1
##   1e-04  1         1
##   1e-01  1         1
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was decay = 0.1.
```

Y las predicciones de tipo probabilistico arrojaron la siguiente tabla.

```
##          UNO         DOS        TRES
## 5  0.9944026 0.003808483 0.001788869
## 8  0.9944590 0.003439714 0.002101311
## 12 0.9949293 0.003197271 0.001873422
## 13 0.9948135 0.003323160 0.001863378
## 14 0.9948795 0.003169016 0.001951505
## 17 0.9948877 0.003386451 0.001725827
```

Donde se puede observar el 30% de los datos, correspondientes a los datos de prueba y la probabilidad asignada según la clase, ya sea del ambiente UNO, DOS o TRES, las predicciones fueron perfectas de igual manera, con datos de 99% para cada ambiente y su dato correspondiente.

Igualmente se creó un script para los nuevos datos muy similar al respecto al modelo de Knn.

Decision Tree

Un Árbol de Decisión (Decision Tree) es un modelo de aprendizaje automático que utiliza una estructura de árbol para tomar decisiones o realizar predicciones. Se basa en el principio de dividir el conjunto de datos en subconjuntos más pequeños y homogéneos mediante la selección de variables y puntos de corte óptimos.

Como se ha venido explicando, en el modelo de KNN y logistic regression el código inicial de la configuración de los datos entre entrenamiento y prueba es el mismo y maneja la misma metodología anteriormente explicada.

A continuación se muestra el código de Decision Tree donde se explica cada linea relevante con un comentario.

Decision Tree Codigo

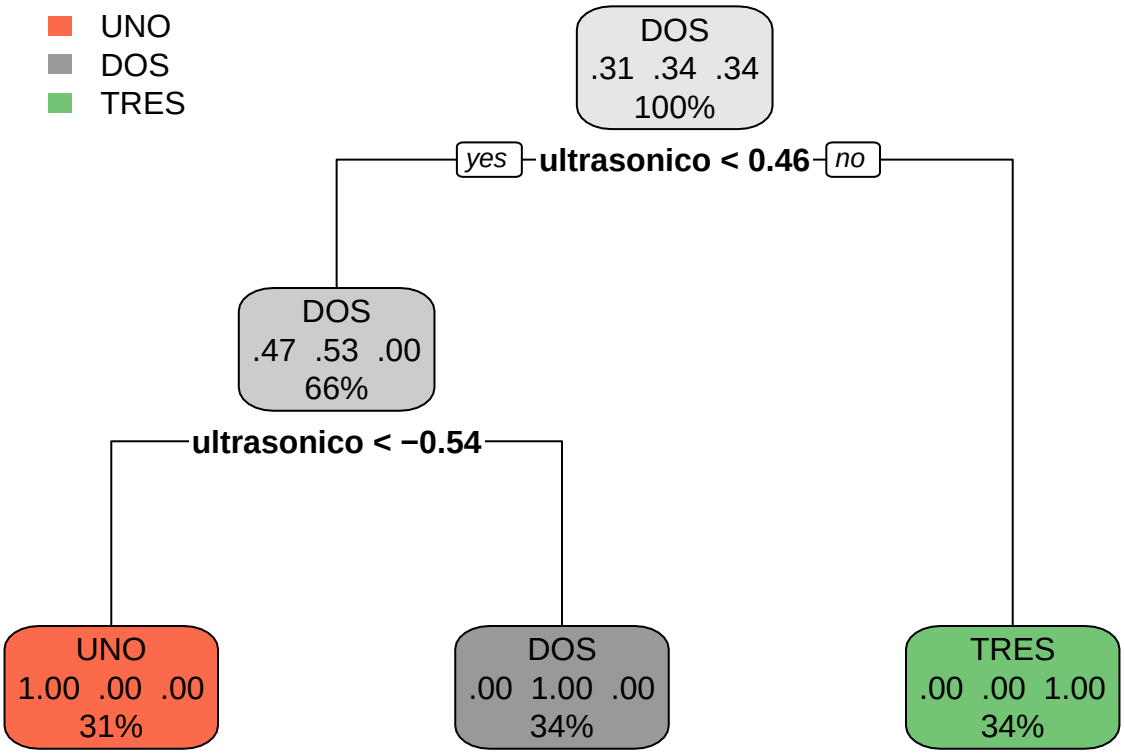
```
library(rpart)
library(rpart.plot)
#Igualmente se toma las muestras aleatorias y se divide en datos de entrenamiento (70%) y datos de prue
```

```
#############################################################################
sample.indexA3 <- sample(1:nrow(Dataset1_norm)
                         ,nrow(Dataset1_norm)*0.7
                         ,replace = F)

train.dataA3 <- Dataset1_norm[sample.indexA3
                              ,c(predictorsA3,"ambiente")
                               ,drop=F]

test.dataA3 <- Dataset1_norm[-sample.indexA3
                             ,c(predictorsA3,"ambiente")
                             ,drop=F]
#############################################################################
#El modelo arbol_ambiente se crea a partir de la funcion rpart() de la librería rpart la cual funciona
arbol_ambiente <- rpart(ambiente ~ ultrasonico+fotorresistencia+color1+color2+color3
                        ,data = train.dataA3)
#Se genera una gráfica de árbol
rpart.plot(arbol_ambiente, type = 2, extra = 104)
#Se guarda el modelo
saveRDS(arbol_ambiente, "DT_model.rds")
```

La grafica del arbol generada es la siguiente:



Donde se puede observar que partió desde el ambiente DOS con el 100$ de los datos y generó la condición de color1 < 0.58, donde de no ser menor a dicho valor, se clasifica como ambiente DOS con un 34% de los datos, el otro 66% de los datos paso a un segundo nodo donde se condicionó esta vez respecto al sensor de ultrasonido con un valor menor a 0.071, en donde de ser cierto se clasifica como del ambiente UNO y por el contrario, de no ser cierto se clasifica automáticamente en el ambiente TRES.

En los últimos 3 nodos se puede observar el porcentaje de datos en cada ambiente.

Random Forest

Random Forest es un algoritmo de aprendizaje automático que se basa en la construcción de múltiples árboles de decisión y combina sus predicciones para obtener un resultado final. Se considera un método de ensamble, ya que combina las predicciones de varios modelos individuales para mejorar la precisión y la generalización.

A continuación se muestra el código de Random Forest donde se explica cada linea relevante con un comentario.

Random Forest Codigo

```
library(randomForest)
library(rpart.plot)
library(caret)
#Igualmente se toma las muestras aleatorias y se divide en datos de entrenamiento (70%) y datos de prue
#############################################################################
data.samples <- sample(1:nrow(Dataset1_norm),
                       nrow(Dataset1_norm) * 0.7,
                       replace = FALSE)

training.dataA4 <- Dataset1_norm[data.samples, ]

test.dataA4 <- Dataset1_norm[-data.samples, ]
#############################################################################
#Modelo a partir de la funcion randomFores de la libreria randomFores, ambiente en funcion de los senso
RFfit.rf <- randomForest(ambiente ~ ultrasonico + fotorresistencia + color1 + color2 + color3,
                         data = training.dataA4)#Datos de entrenamiento
#Predicciones a partir de los datos de prueba
Aprediction.rf <- predict(RFfit.rf, test.dataA4)
# se genera una tabla para verificar los resultados de las predicciones
table(test.dataA4$ambiente, Aprediction.rf)

#Se guarda el modelo
saveRDS(RFfit.rf, "RF_model.rds")
```

La tabla generada es la siguiente.

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:psych':
##
##     outlier
```

```
##       Aprediction.rf
##        UNO DOS TRES
##   UNO   53   0    0
##   DOS    0  48    0
##   TRES   0   0   52
```

De igual manera se observa la predicción perfecta, de la cual se va a tratar mas adelante.

Para random forest es entendido en la manera en que internamente se generan varios arboles como en el modelo de Decision Tree, donde por ejemplo de 4 arboles generados, 3 de ellos generan la respuesta al ambiente UNO y el cuarto al ambiente TRES, el resultado se basa en la mayoria, donde la credibilidad de 3 sobre 1 es superior y se toma dicho valor como la respuesta final.

Conclusiones

Cabe resaltar que la precisión obtenida fue de 1, es decir perfecta, igualmente las predicciones en la matriz de confusión fueron perfectas, gracias a esto internamente se realizaron diferentes pruebas como la eliminación de algunas variables, dos nuevos datasets, variación de la configuración de los modelos, entre muchas otras variaciones mas, de esta manera se concluyo en que la construcción de los ambientes permitieron tener características muy distinguibles entre si, lo que permitía como tal predecir de manera correcta con cualquiera de los sensores aquí utilizados.

El color en cada ambiente fue claro y distinto, la altura que también hacia variar la intensidad de luz incidente en la fotorresistencia fueron variables que permitieron una distinción muy grande entre los ambientes.

Recomendaciones

Es recomendable generar aspectos poco distintivos entre los ambientes para una predicción mas real, por ejemplo que en el techo de los ambientes se creen hoyos de distintas formas lo que variará la forma en la que la luz incide sobre la fotorresistencia, de igual manera utilizar colores compuestos y no primarios, y para el sensor ultrasónico, la posibilidad de generar superficies distintas a sensar, ya sea corrugado, ondulado, en zig-zag, entre otros.