

“Supervised learning final project”

Diego Cubides,Larry castro, Tomas Mendez

2023-05-29

Abstract

The present report is based on the implementation of machine learning to predict three types of environments using the KNN algorithm, decision tree, random forest, and logistic regression. Data was collected from a DHT11 sensor, which measures humidity and temperature, an RGB color sensor TCS3200D TCS230, and a PT100 probe. RStudio was used to train the models with a dataset, and then the prediction was implemented for new data.

For data acquisition and communication, a robot controlled by an Arduino was used, communicating via serial communication with the HC-05 Bluetooth module. This allowed the robot to move easily between different environments and gather data characterizing each one.

Keywords: machine learning, predict, KNN algorithm, decision tree, random forest, logistic regression, DHT11 sensor, humidity, temperature, RGB color sensor TCS3200D TCS230, PT100 probe, RStudio, model training, prediction of new data, data acquisition, serial communication, Arduino, HC-05 Bluetooth module, robot, environments.

Resumen

El presente informe se centra en la aplicación del aprendizaje automático para predecir tres tipos de ambientes utilizando los algoritmos KNN, árbol de decisiones, bosque aleatorio y regresión logística. Se recopilaron datos de un sensor DHT11 para medir la humedad y la temperatura, un sensor de color RGB TCS3200D TCS230 y una sonda PT100. Se utilizó RStudio para entrenar los modelos utilizando un conjunto de datos y luego se implementó la predicción para nuevos datos.

Para la adquisición de datos y la comunicación, se utilizó un robot controlado por un Arduino, que se comunicaba a través de la comunicación serial con el módulo de Bluetooth HC-05. Esto permitió que el robot se moviera fácilmente entre diferentes ambientes y recopilara los datos que caracterizaban cada uno de ellos.

Palabras clave: aprendizaje de máquina, predecir, algoritmo Knn, árbol de decisiones, bosque aleatorio, regresión logística, sensor DHT11, humedad, temperatura, sensor de color RGB TCS3200D TCS230, sonda PT100, RStudio, entrenamiento de modelos, predicción de nuevos datos, adquisición de datos, comunicación serial, Arduino, módulo de Bluetooth HC-05, robot, ambientes.

Introducción

El aprendizaje de máquina y sus modelos, como KNN, árbol de decisiones, bosque aleatorio y regresión logística, son vitales en la actualidad. Estos modelos permiten resolver problemas complejos y tomar decisiones precisas. KNN clasifica ejemplos según su proximidad, mientras que el árbol de decisiones utiliza reglas jerárquicas. El bosque aleatorio combina múltiples árboles para mayor precisión, y la regresión logística predice probabilidades binarias. Estos modelos analizan grandes volúmenes de datos, revelando patrones ocultos y haciendo predicciones en diversos campos, desde medicina hasta finanzas. Su importancia radica en mejorar la toma de decisiones y generar conocimientos valiosos.

Materiales

- Sensor de color RGB TCS3200D TCS230
- Sensor DHT11
- Sonda PT100
- Modulo puente h l298n
- Carro a control remoto
- Bluetooth hc-05
- Arduino UNO
- Dispositivo Android

Software

- RStudio
- Arduino
- App inventor
- Excel
- PLX-DAQ

Procedimiento

adquisision de datos

Se desarrolló un sistema de control remoto para un carro mediante la tecnología Bluetooth, el cual posee la capacidad de moverse a diferentes velocidades y en múltiples direcciones. Para lograr esto, se creó una aplicación utilizando App Inventor, que permite el control preciso del movimiento del carro y la transmisión de datos en tiempo real. La app facilita la interacción con el carro y la recepción de los datos capturados en cada instante.



Figure 1: App

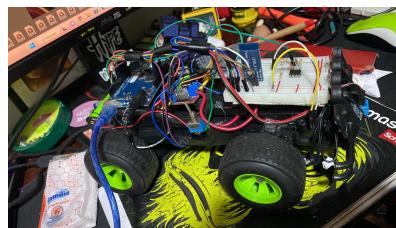


Figure 2: Robot

Para la adquisición de datos en tiempo real, se empleó el programa PLX-DAQ, el cual facilita la comunicación entre los datos provenientes del Arduino y Microsoft Excel. Esta integración permitió abrir los datos en RStudio como un archivo CSV, lo que posibilitó su posterior análisis y procesamiento.

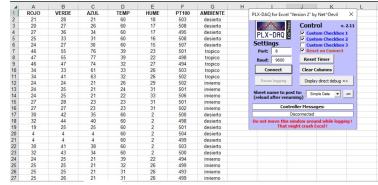


Figure 3: PLX-DQ

Entrenamiento e implementacion de los modelos

Se lleva a cabo el análisis exploratorio de datos para cada una de las variables con el objetivo de examinar su comportamiento. Para este propósito, se utiliza la función hist para generar el histograma correspondiente. El análisis exploratorio de datos nos brinda una comprensión inicial de la distribución y características de las variables, lo que nos permite identificar posibles patrones, tendencias o anomalías en los datos. Este proceso es fundamental para obtener información relevante y realizar futuros análisis y modelos basados en los datos disponibles.

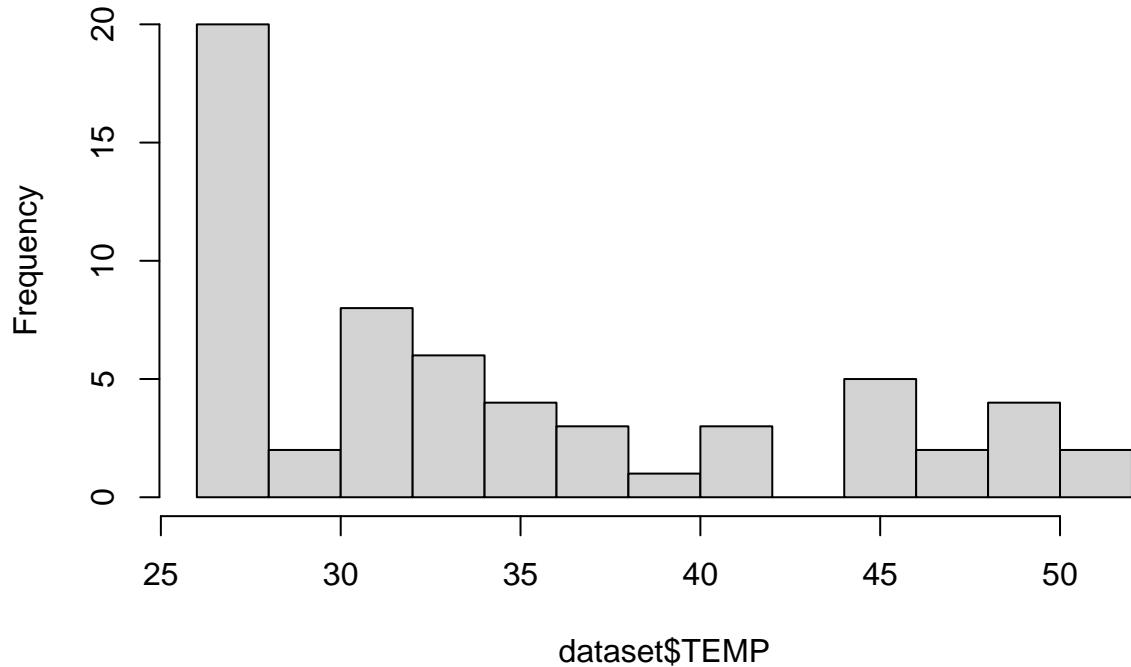
```
head(dataset)
hist(dataset$ROJO,breaks= 10)
hist(dataset$VERDE,breaks= 10)
hist(dataset$AZUL,breaks= 10)
hist(dataset$TEMP,breaks= 10)
hist(dataset$HUME,breaks= 10)
hist(dataset$PT100,breaks= 10)
```

```
kable(head(dataset))
```

| ROJO | VERDE | AZUL | TEMP | HUME | PT100 | AMBIENTE |
|------|-------|------|------|------|-------|----------|
| 17 | 22 | 21 | 41 | 33 | 501 | desierto |
| 17 | 22 | 21 | 46 | 27 | 497 | desierto |
| 16 | 21 | 14 | 46 | 28 | 504 | desierto |
| 17 | 23 | 21 | 48 | 27 | 504 | desierto |
| 17 | 22 | 22 | 48 | 27 | 496 | desierto |
| 15 | 19 | 13 | 49 | 25 | 504 | desierto |

```
hist(dataset$TEMP,breaks= 10)
```

Histogram of dataset\$TEMP



Se genera un resumen estadístico del conjunto de datos dataset utilizando la función summary(). Luego, el resultado se formatea en una tabla utilizando la función kable() del paquete knitr para una presentación más legible

```
kable(summary(dataset))
```

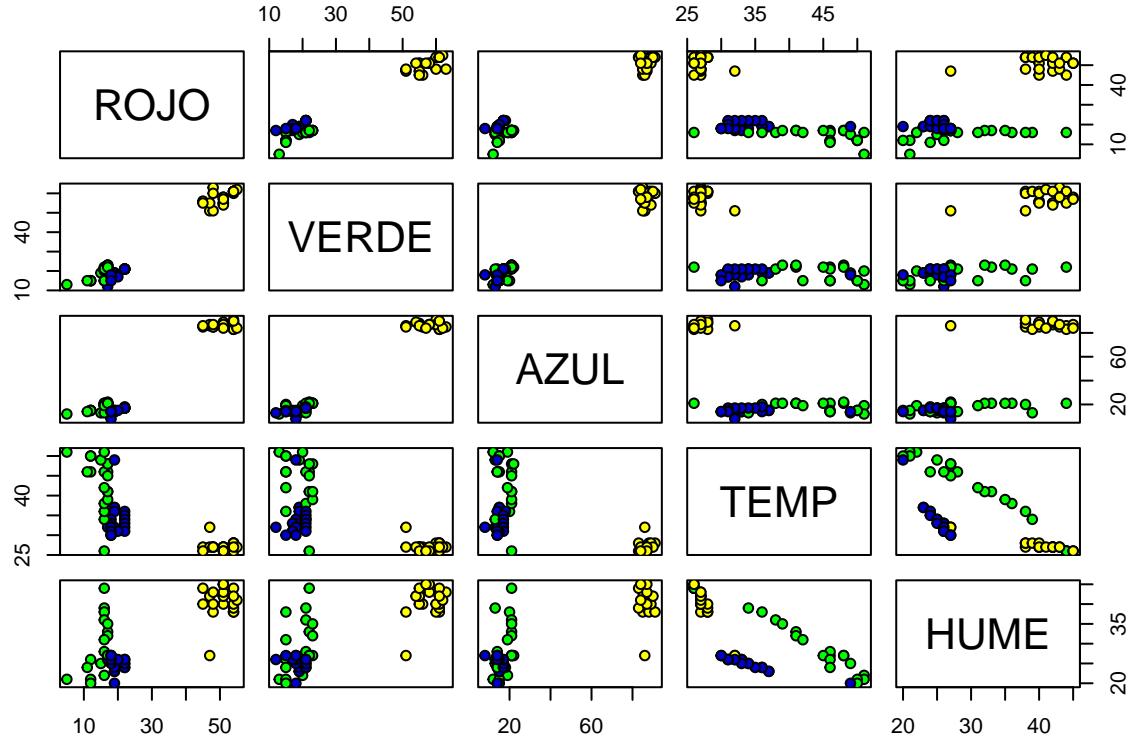
| ROJO | VERDE | AZUL | TEMP | HUME | PT100 | AMBIENTE |
|-------------|------------|-------------|-------------|-------------|-------------|-------------|
| Min. : 5.00 | Min. :12.0 | Min. : 8.00 | Min. :26.00 | Min. :20.00 | Min. :487.0 | desierto:20 |
| 1st | 1st | 1st | 1st | 1st | 1st | invierno:20 |
| Qu.:17.00 | Qu.:18.0 | Qu.:14.75 | Qu.:27.00 | Qu.:25.75 | Qu.:495.8 | |
| Median | Median | Median | Median | Median | Median | tropico :20 |
| :19.00 | :21.0 | :19.50 | :32.50 | :27.00 | :500.0 | |
| Mean :28.33 | Mean :31.8 | Mean :39.77 | Mean :34.93 | Mean :31.63 | Mean :498.9 | NA |
| 3rd | 3rd | 3rd | 3rd | 3rd | 3rd | NA |
| Qu.:48.00 | Qu.:55.0 | Qu.:85.00 | Qu.:41.00 | Qu.:40.00 | Qu.:502.0 | |
| Max. :55.00 | Max. :63.0 | Max. :91.00 | Max. :51.00 | Max. :45.00 | Max. :506.0 | NA |

Se convierten las columnas AMBIENTE en factores utilizando la función as.factor(). Esto es útil cuando se trabaja con variables categóricas en modelos de aprendizaje automático.

```
dataset$AMBIENTE <- as.factor(dataset$AMBIENTE)  
datasetcopia$AMBIENTE <- as.factor(datasetcopia$AMBIENTE)
```

Se genera un diagrama de dispersión utilizando las primeras 5 columnas del conjunto de datos dataset. Cada punto en el diagrama de dispersión está marcado con un símbolo (pch) y un color de fondo (bg) determinados por la variable categórica AMBIENTE.

```
plot(dataset[1:5]
      ,pch=21, bg=c("green","blue3","yellow") [unclass(dataset$AMBIENTE)])
```



- Knn method

Se utiliza Cross-Validation para dividir los datos en un 70% de entrenamiento y un 30% de prueba. Las variables predictoras incluyen los datos del sensor de color (ROJO, VERDE, AZUL) y los sensores de humedad y temperatura. Esto permite evaluar el rendimiento del modelo en datos no vistos y predecir diferentes ambientes.

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

predictors <- c("ROJO", "VERDE", "AZUL", "TEMP", "HUME", "PT100")
train.data  <- dataset[sample.index
                        ,c(predictors, "AMBIENTE")]
                        ,drop=F]
test.data   <- dataset[-sample.index
                        ,c(predictors, "AMBIENTE")]
                        ,drop=F]
```

Para entrenar el modelo, se utiliza la función train con los parámetros de la variable a predecir y los sensores. Se emplea el método KNN y se realiza una normalización min-max como preprocesso de datos. Esto permite obtener un modelo entrenado y listo para hacer predicciones precisas.

```

ctrl <- trainControl(method = "cv", p=0.7)
Knnfit <- train(AMBIENTE ~ ROJO+VERDE+AZUL+TEMP+HUME+PT100
  ,data = train.data
  ,method = "knn", trControl = ctrl
  ,preProcess= c("range")
  ,tuneLength=20)

```

Una vez que se tiene el modelo, se evalúa su rendimiento utilizando los valores de prueba. Para esto, se utiliza la función predict, a la cual se le pasa el modelo (Knnfit) y los datos de prueba (test.data). Esta función realiza predicciones basadas en el modelo entrenado, utilizando los datos de prueba para evaluar su desempeño y precisión.

```
Knnpredict <- predict(Knnfit,newdata = test.data)
```

Para visualizar los resultados, se genera la matriz de confusión. Esta matriz proporciona una representación visual de la precisión de las predicciones del modelo. Permite identificar y analizar la cantidad de clasificaciones correctas e incorrectas realizadas por el modelo en cada clase o categoría. La matriz de confusión es una herramienta útil para evaluar el rendimiento del modelo y determinar posibles áreas de mejora.

```

Knnpredict <- predict(Knnfit,newdata = test.data)
confusionMatrix(Knnpredict,test.data$AMBIENTE)

```

```

## Confusion Matrix and Statistics
##
##             Reference
## Prediction desierto invierno tropico
##   desierto      5       1       0
##   invierno     1       4       0
##   tropico      0       0       7
##
## Overall Statistics
##
##                  Accuracy : 0.8889
##                  95% CI : (0.6529, 0.9862)
##      No Information Rate : 0.3889
##      P-Value [Acc > NIR] : 1.685e-05
##
##                  Kappa : 0.8318
##
## Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                               Class: desierto Class: invierno Class: tropico
## Sensitivity                  0.8333        0.8000       1.0000
## Specificity                  0.9167        0.9231       1.0000
## Pos Pred Value                0.8333        0.8000       1.0000
## Neg Pred Value                0.9167        0.9231       1.0000
## Prevalence                   0.3333        0.2778       0.3889
## Detection Rate                 0.2778        0.2222       0.3889
## Detection Prevalence          0.3333        0.2778       0.3889
## Balanced Accuracy              0.8750        0.8615       1.0000

```