2nd report guideline: “kNN, Linear regression, and multilinear regression”

Danny Rodriguez

2022-10-09

## 2nd report guideline: “kNN, Linear regression, and multilinear regression”

# Linear regression and andmultilinear regression

# Objetivos

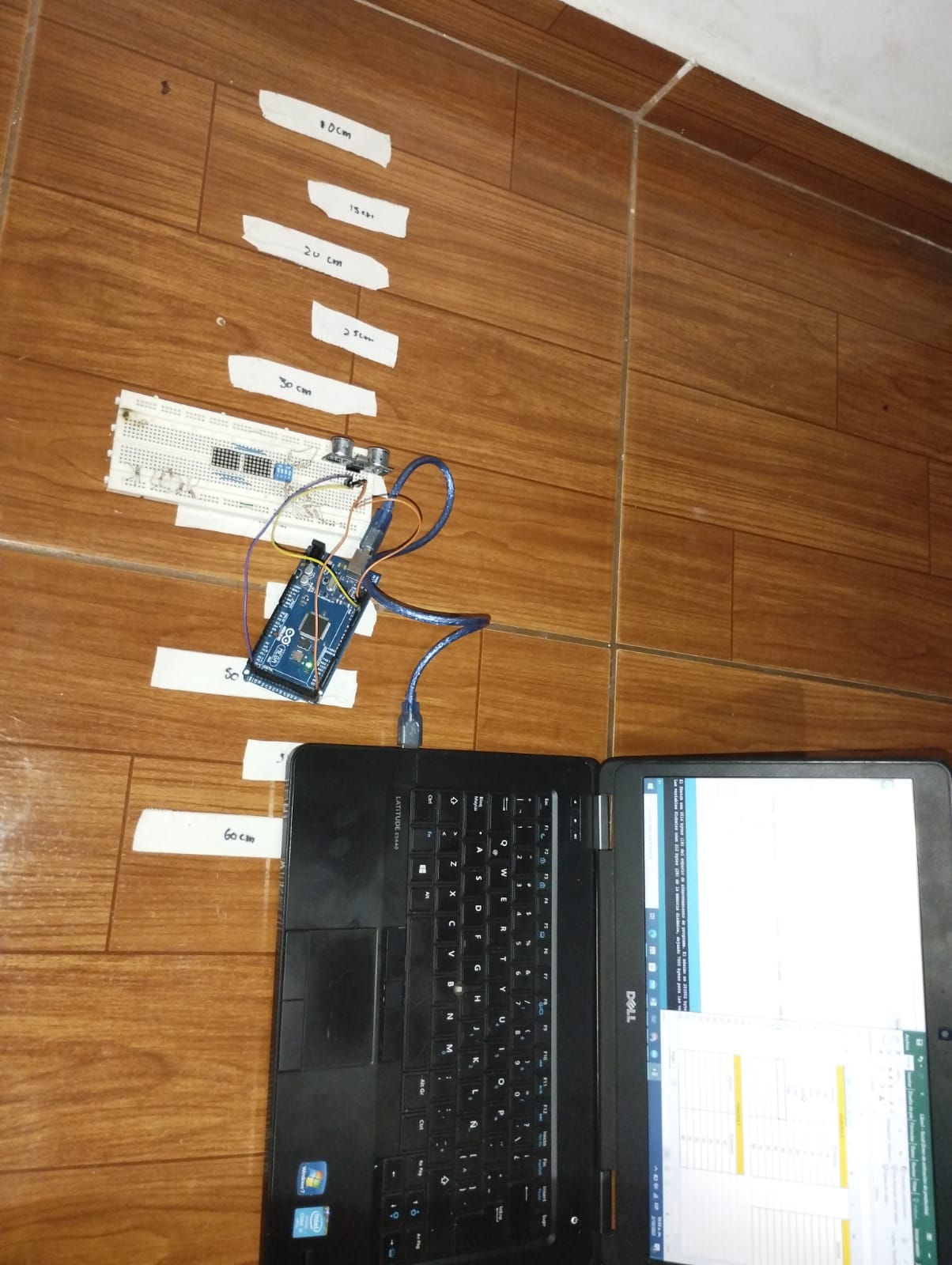
Desarrollar un modelo de adquisición de datos mediante dos sensores utilizando un microcontrolador, por otra parte, con estos datos realizar dos métodos de predicción

## Resumen

Con dos sensores de ultrasonido se realizo la adquisición de datos para medir distancias a una pared plana, con un delta de 5 cm se toman 11 datos con 5 muestras diferentes luego de esto se crea un data frame para realizar diferentes modelos de prediccion y luego de esto se realiza una validacion mediante “cross-validation”.

## Procedimiento para modelos predictivos

Mediante un software llamado arduino se crea un programa para realizar la obtencion de datos como se ve en la siguientes imagenes



En la imagen se ve la obtencion que se hizo colocando unos puntos marcados por una cinta para evitar la variacion.



Codigo\_Arduino

Se tomo muestras desde los 10 cm hasta los 60 cm con un delta de 5 cm para un total de datos de 11 los resultados se pueden ver a continuacion en las siguientes tablas.

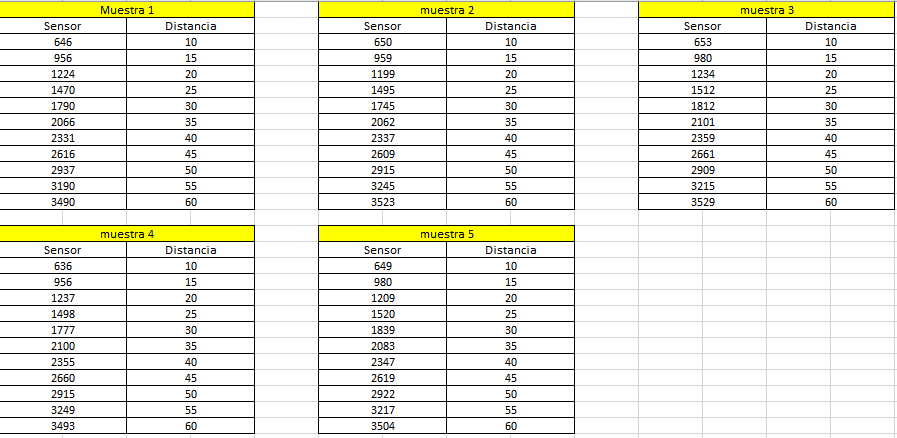


Tabla Sensor 1

La Tabla Sensor 1 son los valores que se obtuvieron mediante 5 muestras tomadas aleatoriamente para el primer sensor.

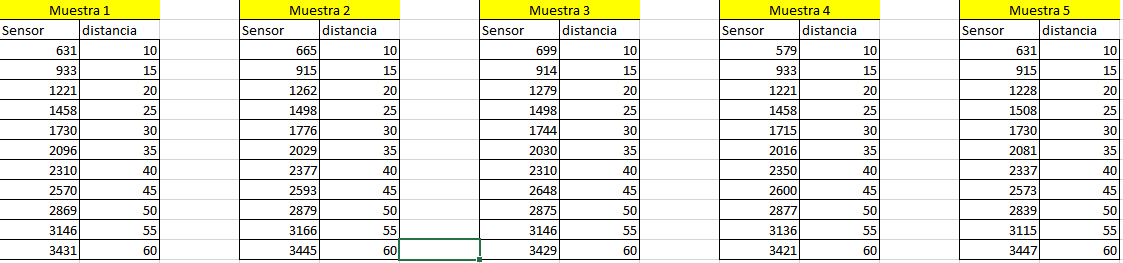


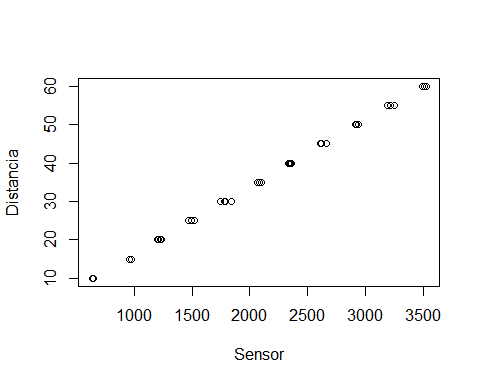
Tabla Sensor 2

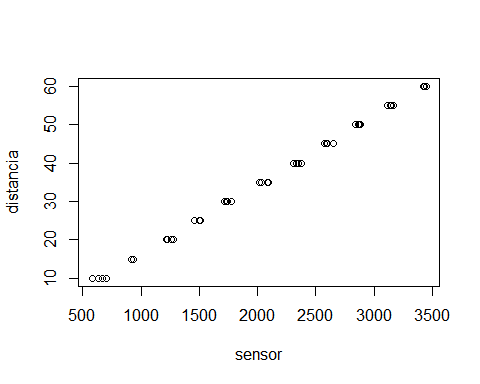
En la Tabla Sensor 2 podemos ver los valores obtenidos para el sensor 2.

Luego de tener los data-frames creados vamos a realizar el entrenamiento para un modelo lineal, un modelo lineal con regularizado y para un modelo, multilineal.

library(tidyverse)  
library(caret)  
library(readr)  
library(caret)  
library(readr)  
library(class)  
library(gmodels)  
  
folder <- dirname(rstudioapi::getSourceEditorContext()$path)  
parentFolder <- dirname(folder)  
   
  
data <- read\_delim("C:/Users/admin/Documents/Universidad/electiva/Nueva carpeta/data.csv",   
 delim = ";", escape\_double = FALSE, trim\_ws = TRUE)  
  
data2 <- read\_delim("C:/Users/admin/Documents/Universidad/electiva/Nueva carpeta/data2.csv",   
 delim = ";", escape\_double = FALSE, trim\_ws = TRUE)  
data3 <- read\_delim("C:/Users/admin/Documents/Universidad/electiva/Nueva carpeta/data3.csv",   
 delim = ";", escape\_double = FALSE, trim\_ws = TRUE)

En estas graficas podemos que podemos realizar un modelo lineal ya que la varianza no estan grande.





ademas podemos ver algunas de las caracteristicas de nuestros predictores y nuestro valor a predecir por ejemplo el valor minimo, maximo la media y el promedio.

Aca podemos ver algunas caracteristicas principales del data frame del sensor 1

#Data 1  
summary(data)

## Sensor Distancia   
## Min. : 636 Min. :10   
## 1st Qu.:1234 1st Qu.:20   
## Median :2074 Median :35   
## Mean :2073 Mean :35   
## 3rd Qu.:2915 3rd Qu.:50   
## Max. :3523 Max. :60

Aca podemos ver algunas caracteristicas principales del data frame del sensor 2

#Data 2  
summary(data2)

## sensor distancia   
## Min. : 579 Min. :10   
## 1st Qu.:1270 1st Qu.:20   
## Median :2030 Median :35   
## Mean :2041 Mean :35   
## 3rd Qu.:2854 3rd Qu.:50   
## Max. :3447 Max. :60

Aca podemos ver algunas caracteristicas principales del data frame cuando tenemos los dos sensores

#Data 2  
summary(data3)

## sensor1 sensor2 distancia   
## Min. : 636.0 Min. : 579 Min. :10   
## 1st Qu.: 658.5 1st Qu.:1270 1st Qu.:20   
## Median :1745.0 Median :2030 Median :35   
## Mean :1789.3 Mean :2041 Mean :35   
## 3rd Qu.:2617.5 3rd Qu.:2854 3rd Qu.:50   
## Max. :3523.0 Max. :3447 Max. :60

luego de realizar toda la parte de exploracion de datos y ver el comportamiento de los sensores vamos a entrenar los modelos lineales y un modelo multilineal para ver cual modelo predictivo es mas acertado

#-------------------------------------------#  
  
#---Creo el modelo y hago la validacion del data1---#  
#hold-out cross-validation  
sample.index <- sample(1:nrow(data)  
   
 , nrow(data) \* 0.75, replace = FALSE)  
  
training.data <-data[sample.index,, drop=F]  
test.data <- data[-sample.index,, drop=F]  
#creating a model with 4 predictors  
model <- lm(Distancia~Sensor,training.data)  
  
# Make predictions and compute the R2, RMSE and MAE using CARET  
predictions <- model %>% predict(test.data)  
data.frame( R2 = R2(predictions, test.data$Distancia),  
 RMSE = RMSE(predictions, test.data$Distancia),  
 MAE = MAE(predictions, test.data$Distancia))

## R2 RMSE MAE  
## 1 0.9995861 0.3462548 0.2458084

#--------------------------------------------------------------------#  
# Define training control  
set.seed(1)  
train.control <- trainControl(method = "cv", number = 10)  
# Train the model  
model <- train(Distancia ~Sensor , data = data, method = "lm",  
 trControl = train.control)  
  
# Summarize the results  
print(model)

## Linear Regression   
##   
## 44 samples  
## 1 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 40, 39, 41, 39, 39, 41, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 0.344901 0.9996199 0.3048464  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

#--------------------------------------------------------------------#  
  
#----Ahora entreno el segundo data----#  
#hold-out cross-validation  
sample.index <- sample(1:nrow(data2)  
   
 , nrow(data2) \* 0.75, replace = FALSE)  
  
training.data2 <-data2[sample.index,, drop=F]  
test.data2 <- data2[-sample.index,, drop=F]  
#creating a model with 4 predictors  
model2 <- lm(distancia~sensor,training.data2)  
  
# Make predictions and compute the R2, RMSE and MAE using CARET  
predictions <- model2 %>% predict(test.data2)  
data.frame( R2 = R2(predictions, test.data2$distancia),  
 RMSE = RMSE(predictions, test.data2$distancia),  
 MAE = MAE(predictions, test.data2$distancia))

## R2 RMSE MAE  
## 1 0.9990916 0.3776732 0.2747193

#--------------------------------------------------------------------#  
# Define training control  
set.seed(1)  
train.control <- trainControl(method = "cv", number = 10)  
# Train the model  
model2 <- train(distancia ~sensor , data = data2, method = "lm",  
 trControl = train.control)  
  
# Summarize the results  
print(model2)

## Linear Regression   
##   
## 55 samples  
## 1 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 49, 48, 49, 51, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 0.5065019 0.9992616 0.408878  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

#--------------------------------------------#  
#--------entreno ahora el tercero------------#  
#-----------------------------------------#  
#hold-out cross-validation  
sample.index <- sample(1:nrow(data3)  
   
 , nrow(data3) \* 0.75, replace = FALSE)  
  
training.data3 <-data3[sample.index,, drop=F]  
test.data3 <- data3[-sample.index,, drop=F]  
#creating a model with 4 predictors  
model3 <- lm(distancia~sensor1+sensor2,training.data3)  
  
# Make predictions and compute the R2, RMSE and MAE using CARET  
predictions <- model3 %>% predict(test.data3)  
data.frame( R2 = R2(predictions, test.data3$distancia),  
 RMSE = RMSE(predictions, test.data3$distancia),  
 MAE = MAE(predictions, test.data3$distancia))

## R2 RMSE MAE  
## 1 0.9991026 0.3752907 0.2733262

#--------------------------------------------------------------------#  
# Define training control  
set.seed(1)  
train.control <- trainControl(method = "cv", number = 10)  
# Train the model  
model3 <- train(distancia~sensor1+sensor2 , data = data3, method = "lm",  
 trControl = train.control)  
  
# Summarize the results  
print(model3)

## Linear Regression   
##   
## 55 samples  
## 2 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 49, 48, 49, 51, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 0.5210459 0.9992354 0.4239296  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

## Resultados

Una vez obtenidos los resultados se ve que el modelo 1 es el que tiene menos error a la hora de predecir cuando le enviamos un valor nuevo con nuestro sensor.

Aca una vista de nuestro modelo tomamos una muestra a 50 cm y el nos regresa un valor de 3000 lo pasamos por nuestro modelo y la respuesta es la siguiente

test\_data1<-data.frame(Sensor=3000)  
predict(model,test\_data1)

## 1   
## 51.2943

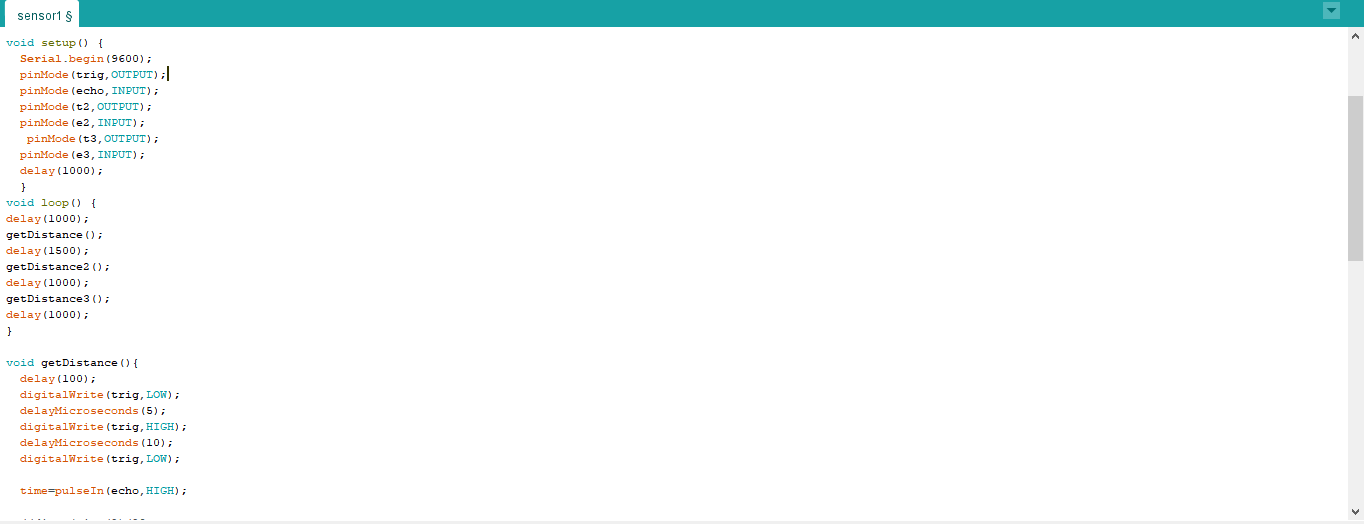
## kNN

## Resumen

Con tres sensores de ultrasonido se realizo la adquisición de datos para medir distancias a una pared plana, concava y convexa con distancias desde 24 cm hasta los 74 cm con variaciones de 10 cm, para cada clase obtuvimos 50 datos que se tomaron aleatoriamente.

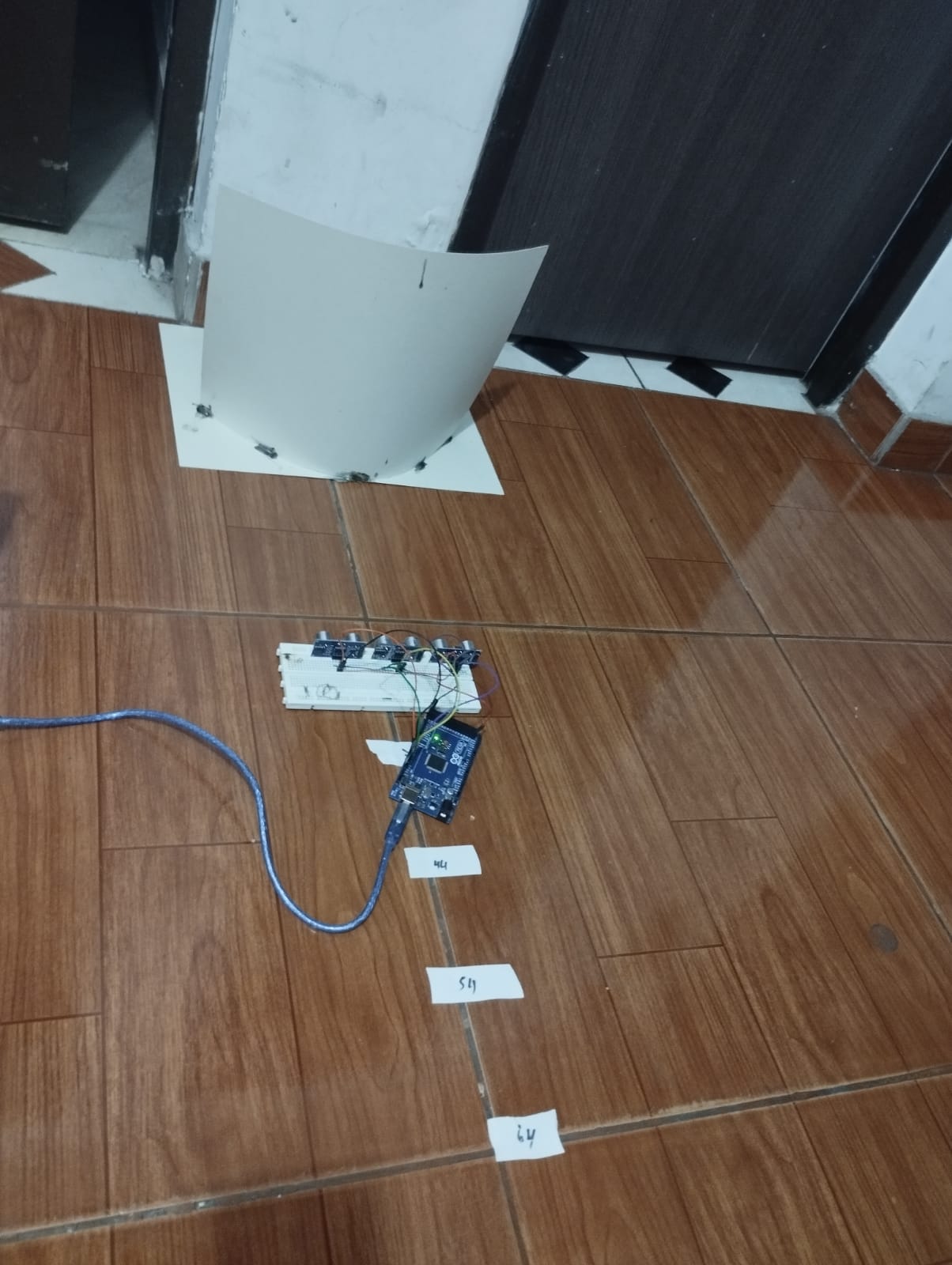
## Procedimiento para modelos predictivos

Lo primero que se hace es hacer un codigo en Arduino con tres sensores de proximidad como se ve en la siguiente imagen.



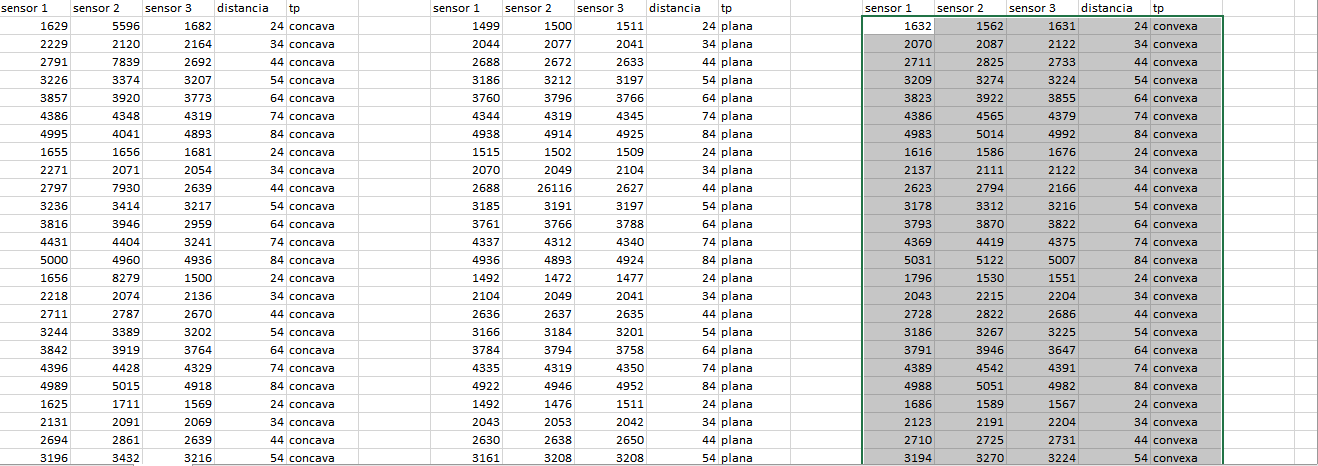
Codigo Arduino 2

Luego pasamos a la obtencion de datos que se hizo con una pared concava primero, luego una pared convexa y al final una pared plana como muestra la siguiente imagen.



Obtencion de datos para los tipos de pared

Una vez obtenidos los datos los guardamos en un data-frame.



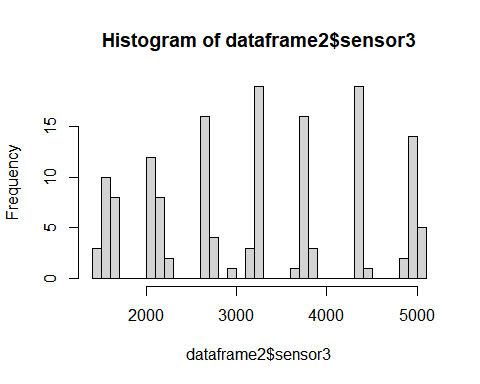
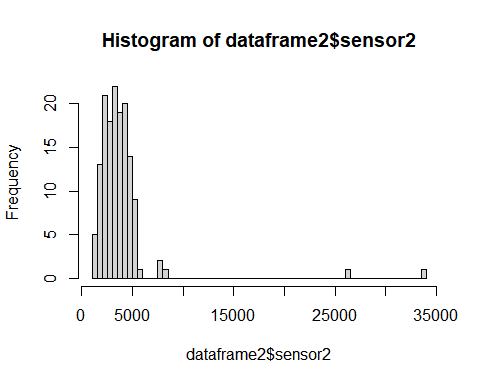
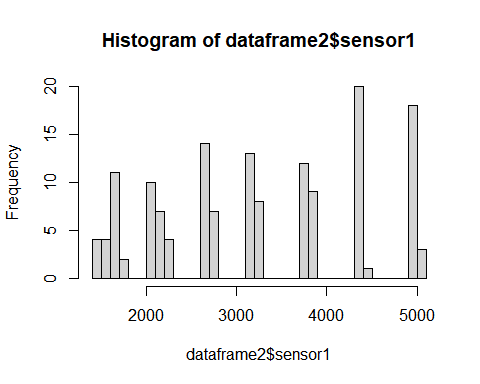
Datos obtenidos por los sensores

Luego de esto cargamos nuestro data-frame en Rstudio y comenzamos a crear nuestro modelo de Knn

Una vez cargado nuestro data-frame pasamos a crear una funcion de max-min para mas adelante crear un modelo basado en este practica.

Luego de esto pasamos nuestro valor a predecir como factor y creamos dos nuevos datas uno normalizado y el otro con Z-score.

##   
## concava convexa plana   
## 0.3333333 0.3333333 0.3333333



Luego de esto voy a realizar mi entrenamiento de Knn

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 45   
##   
##   
## | prediction   
## test.data$tp | concava | convexa | plana | Row Total |   
## -------------|-----------|-----------|-----------|-----------|  
## concava | 8 | 3 | 2 | 13 |   
## | 0.615 | 0.231 | 0.154 | 0.289 |   
## | 0.727 | 0.214 | 0.100 | |   
## | 0.178 | 0.067 | 0.044 | |   
## -------------|-----------|-----------|-----------|-----------|  
## convexa | 3 | 10 | 0 | 13 |   
## | 0.231 | 0.769 | 0.000 | 0.289 |   
## | 0.273 | 0.714 | 0.000 | |   
## | 0.067 | 0.222 | 0.000 | |   
## -------------|-----------|-----------|-----------|-----------|  
## plana | 0 | 1 | 18 | 19 |   
## | 0.000 | 0.053 | 0.947 | 0.422 |   
## | 0.000 | 0.071 | 0.900 | |   
## | 0.000 | 0.022 | 0.400 | |   
## -------------|-----------|-----------|-----------|-----------|  
## Column Total | 11 | 14 | 20 | 45 |   
## | 0.244 | 0.311 | 0.444 | |   
## -------------|-----------|-----------|-----------|-----------|  
##   
##

Este Knn es de los datos normales sin ningun procesamiento luego de esto vamos a realizar el entrenamiento con los datos normalizados

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 45   
##   
##   
## | prediction   
## test.data$tp | concava | convexa | plana | Row Total |   
## -------------|-----------|-----------|-----------|-----------|  
## concava | 8 | 3 | 2 | 13 |   
## | 0.615 | 0.231 | 0.154 | 0.289 |   
## | 0.727 | 0.214 | 0.100 | |   
## | 0.178 | 0.067 | 0.044 | |   
## -------------|-----------|-----------|-----------|-----------|  
## convexa | 3 | 10 | 0 | 13 |   
## | 0.231 | 0.769 | 0.000 | 0.289 |   
## | 0.273 | 0.714 | 0.000 | |   
## | 0.067 | 0.222 | 0.000 | |   
## -------------|-----------|-----------|-----------|-----------|  
## plana | 0 | 1 | 18 | 19 |   
## | 0.000 | 0.053 | 0.947 | 0.422 |   
## | 0.000 | 0.071 | 0.900 | |   
## | 0.000 | 0.022 | 0.400 | |   
## -------------|-----------|-----------|-----------|-----------|  
## Column Total | 11 | 14 | 20 | 45 |   
## | 0.244 | 0.311 | 0.444 | |   
## -------------|-----------|-----------|-----------|-----------|  
##   
##

Estos son nuestros entrenamiento con el procesamiento normalizado y al final hacemos el del Z-Core

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 45   
##   
##   
## | prediction   
## test.data$tp | concava | convexa | plana | Row Total |   
## -------------|-----------|-----------|-----------|-----------|  
## concava | 7 | 4 | 2 | 13 |   
## | 0.538 | 0.308 | 0.154 | 0.289 |   
## | 0.700 | 0.235 | 0.111 | |   
## | 0.156 | 0.089 | 0.044 | |   
## -------------|-----------|-----------|-----------|-----------|  
## convexa | 3 | 10 | 0 | 13 |   
## | 0.231 | 0.769 | 0.000 | 0.289 |   
## | 0.300 | 0.588 | 0.000 | |   
## | 0.067 | 0.222 | 0.000 | |   
## -------------|-----------|-----------|-----------|-----------|  
## plana | 0 | 3 | 16 | 19 |   
## | 0.000 | 0.158 | 0.842 | 0.422 |   
## | 0.000 | 0.176 | 0.889 | |   
## | 0.000 | 0.067 | 0.356 | |   
## -------------|-----------|-----------|-----------|-----------|  
## Column Total | 10 | 17 | 18 | 45 |   
## | 0.222 | 0.378 | 0.400 | |   
## -------------|-----------|-----------|-----------|-----------|  
##   
##