“kNN, Linear regression, and multilinear regression”

Chavez Miguel Angel 80811

Ducuara Quesada Daniela 85742

Garcia Castillo Fernando 61865

Mora Andres Acevedo 55305

Morales Javier Sebastian 73322

Palacios Diego Alejandro 46026

2023-04-21

-**Enlace de GitHub:** https://github.com/Khallavan/sensorsTraining

Machine Learning

Machine Learning es una forma de la inteligencia artificial que permite crear modelos estadisticos y algoritmos, en donde la máquina aprende de los datos, mas no mediante la programación predeterminada. Estos modelos pueden procesar grandes cantidades de datos y por medio del entrenamiento identificar patrones de los datos, lo que permite crear modelos mas precisos en la predicción de la información de salida.

¿Que es KNN?

KNN,en ingles K Nearest Neighbor, es un algoritmo que utiliza la “similitud de caracteristicas” para predecir los valores de los nuevos puntos de datos, lo que significa que al nuevo punto de datos se le asignara un valor en función de que tan cerca conicide los puntos en el conjunto de entrenamiento. algunas de las caracteristicas que tiene el KNN son:

- utiliza conjuntos de datos de entrada etiquetados para asi poder predecir la salida de los datos. -Es un algoritmo de aprendizaje simple y puede ser implementado facilmente para un conjunto variado de problemas.
- Se basa principalmemte en la similitud de caracteristicas,es decir, verifica cuán similares es un punto de datos con relacion a su vecino y clasifica el punto de datos en la cl…

Regresion lineal y multilineal

Un analisis de regresión lineal se utiliza para predecir el valor de una variable según el valor de la otra, por lo que estima los coeficientes de la ecuacion lineal,la regresion lineal se ajusta a una linea recta lo que minimiza discrepancias entre los valores de salida previsto y los reales. El modelo de regresion lineal se considera relativamente

sensillo ya que propporciona una formula matematica facil de interpretar que puede generar predicciones.

El modelo de regresión multilineal,es un modelo estadistico que permite generar un modelo lineal en el que el valor de la variable dependiente, se determina a partir de un conjunto de variables independientes llamadas “predictores”. ## Resumen:

En el presente informe se realiza una adquisión de datos de dos sensores diferentes que se adaptan a un carrito robot, esto con el fin de tomar los datos que arroja cada sensor a diferentes distancias sobre una superficie (pared) plana, concava y convexa. Los dos sensores utilizados fueron los siguientes: Ultrasonido HCR-04 Y Laser VL53L0X. Se realiza los siguientes pasos para poder tomar los datos:

##1. Adquisición de datos Desarrollar un sistema de adquisición (hardware y software) para capturar datos de distancia.

-Se acondicionaron dos sensores, un ultrasonido y uno láser. El ultrasonido entrega el valor que tarda la onda en emitir y recibirla. Para el láser se tomaron mediciones directas en longitud, esto para poder generar los modelos de entrenamiento y predicción sin usar librerías que suministren la curva o ecuación característica del comportamiento de los sensores, ya que se verían afectados los valores con los modelos definidos.

- La primera distancia se toma desde 10 cm desde la pared, y se fue aumentando la distancia cada 10 cm, para llegar a un máximo de 50 cm de distancia. En cada distancia se tomaron 40 observaciones.
2. Se tomaron los datos y se registran en un archivo .csv. A continuación se carga el archivo para poder ver los datos para cada sensor y con cada superficie.

```
load_datasets <- function(name_dataset) {
    if(!require(tidyverse))
        install.packages("tidyverse")
    library(tidyverse)

    folder <- dirname(rstudioapi::getSourceEditorContext()$path)
    parentFolder <- dirname(folder)
    sensors <- read.csv2(paste0(parentFolder,"/datasets/",name_dataset))
    return (sensors)
}
```

Esta función load_datasets se encarga de leer los datasets que estan cargados en el repositorio. Lee los archivos .csv que estan separados por punto y coma y por ello se emplea la función (read.csv2)

1.1 **Carga de datasets**

```
if(!require(tidyverse))
  install.packages("tidyverse")
```

```
## Loading required package: tidyverse
```

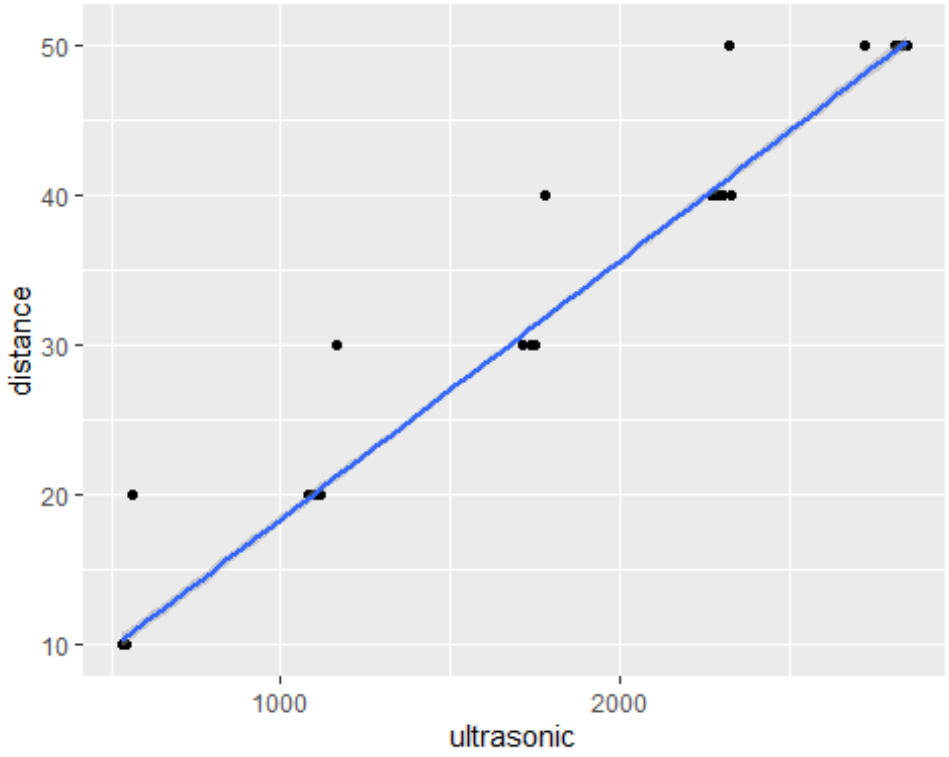
```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.1     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors

library(tidyverse)
sensors.df <- load_datasets("sensors_dataset_A.csv")
ultrasonic.data <- sensors.df %>% select(ultrasonic, distance)
laser.data <- sensors.df %>% select(laser, distance)
multi.data <- sensors.df %>% select(ultrasonic, laser, distance)
```

Aquí se carga el dataset con el nombre de sensors.df. A continuación, se muestra una pequeña parte de los datos:

```
head(multi.data)

##   ultrasonic laser distance
## 1        542   108       10
## 2        541   106       10
## 3        541   106       10
## 4        535   108       10
## 5        535   108       10
## 6        536   107       10
```

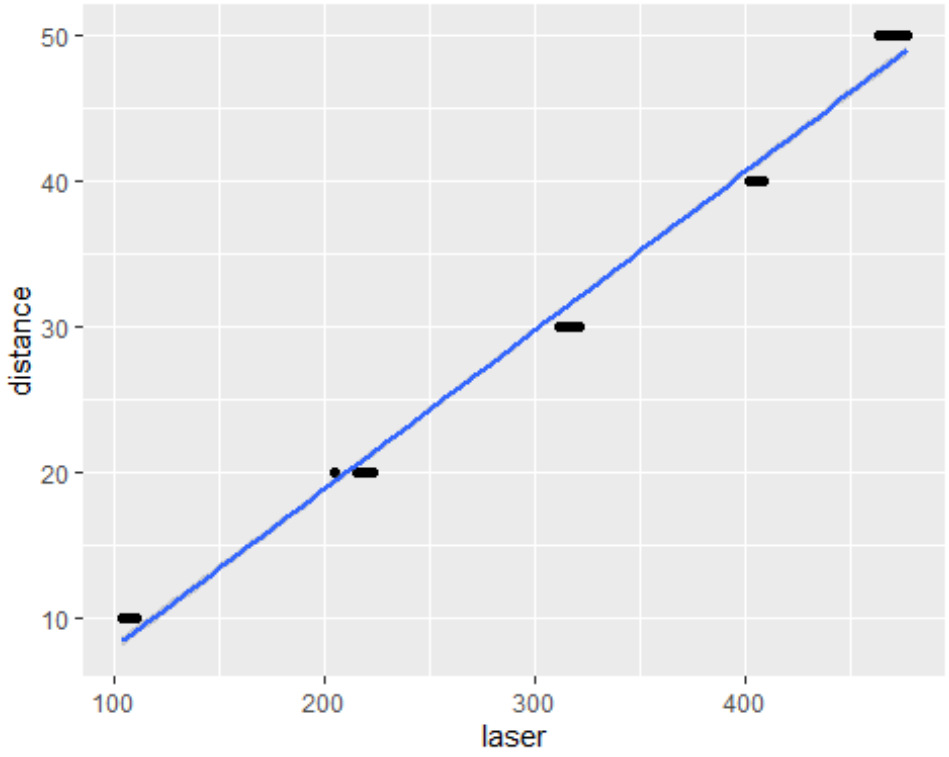
1.2 **visualización de los datos en gráfica de dispersión**

```
if(!require(ggplot2))
  install.packages("ggplot2")
library(ggplot2)
ultrasonic.data %>%
    ggplot(aes(x = ultrasonic, y = distance)) +
    geom_point() +
    geom_smooth(method = "lm")

## `geom_smooth()` using formula = 'y ~ x'
```



Gráfica de los datos del sensor ultrasonico.

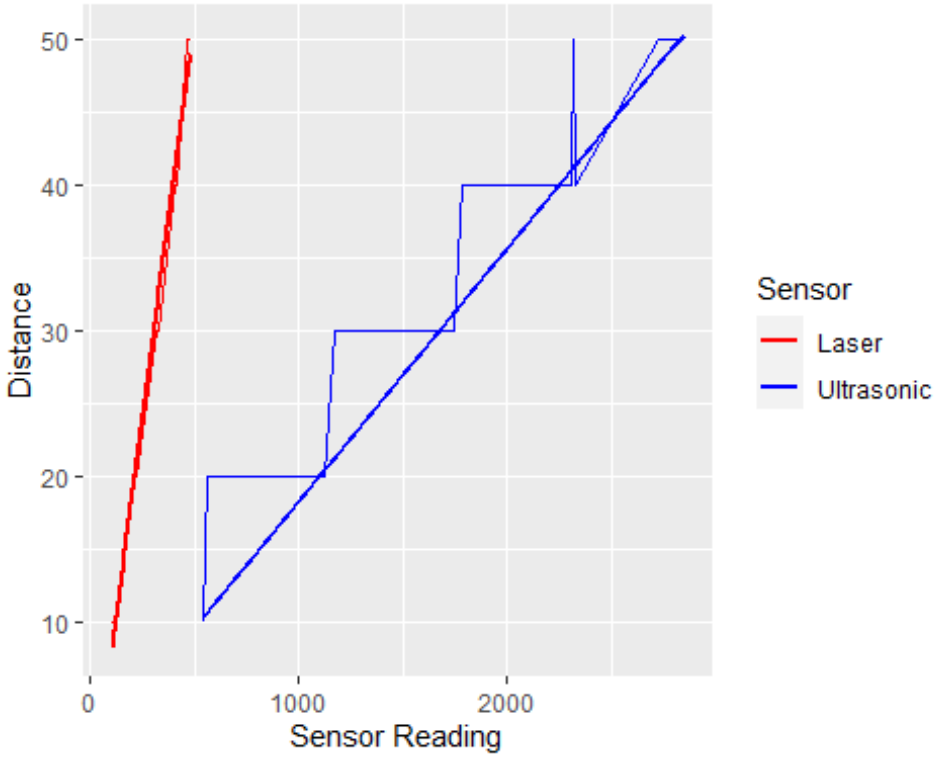
```
if(!require(ggplot2))
  install.packages("ggplot2")
library(ggplot2)
laser.data %>%
    ggplot(aes(x = laser, y = distance)) +
    geom_point() +
    geom_smooth(method = "lm")

## `geom_smooth()` using formula = 'y ~ x'
```



Gráfica de los datos del sensor Laser.

La siguiente gráfica se mostraran la dispersión de los datos de los dos sensores.

```
if(!require(ggplot2))
  install.packages("ggplot2")
library(ggplot2)
multi.data %>%
    ggplot(aes(y = distance)) +
    geom_line(aes(x = ultrasonic, color = "Ultrasonic")) +
    geom_line(aes(x = laser, color = "Laser")) +
    geom_smooth(aes(x = ultrasonic, color = "Ultrasonic"), method = "lm",
se = FALSE) +
    geom_smooth(aes(x = laser, color = "Laser"), method = "lm", se =
FALSE) +
    scale_color_manual(name = "Sensor", values = c("Ultrasonic" = "blue",
"Laser" = "red")) +
    labs(y = "Distance", x = "Sensor Reading")

## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



1.2 **Preparación de datos para cada modelo¨**

Se usa la libreria caret para realizar el entrenamiento de los modelos predictivos mediante la automatización de tareas comunes, como la normalización de datos, la creación de modelos, la validación cruzada y la evaluación de modelos.

Para una mejor claridad de los datos se emplea y para su posterior replicación en caso de que se requiera, se utiliza una semilla predeterminada.

```
if(!require(caret))
  install.packages("caret")

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

library(caret)
set.seed(1)
ultrasonic.train.samples <- ultrasonic.data$distance %>%
    createDataPartition(p = 0.8, list = FALSE)
laser.train.samples <- laser.data$distance %>%
```

```
    createDataPartition(p = 0.8, list = FALSE)
multi.train.samples <- multi.data$distance %>%
    createDataPartition(p = 0.8, list = FALSE)


ultrasonic.train.data <- ultrasonic.data[ultrasonic.train.samples,
                                         c("ultrasonic", "distance")]
ultrasonic.test.data <- ultrasonic.data[-ultrasonic.train.samples,
                                        c("ultrasonic", "distance")]

laser.train.data <- laser.data[laser.train.samples, c("laser",
"distance")]
laser.test.data <- laser.data[-laser.train.samples, c("laser",
"distance")]
multi.train.data <- multi.data[multi.train.samples, c("ultrasonic",
"laser", "distance")]
multi.test.data <- multi.data[-multi.train.samples, c("ultrasonic",
"laser", "distance")]
```

Implementando la función de caret, entrenamos un modelo de regresión lineal para cada sensor y un modelo multilineal empleando ambos sensores. La ventaja que nos porporciona la libreria Caret es que simplifica los valores de predicción R^2, MAE y RMSE.

1.3 **Entrenamiento de los modelos**

```
model.distance.ultrasonic <- train(
        distance ~ ultrasonic,
        data = ultrasonic.train.data,
        method = "lm",
        trControl = trainControl("cv", number = 10)
        )
model.distance.laser <- train(
        distance ~ laser,
        data = laser.train.data,
        method = "lm",
        trControl = trainControl("cv", number = 10)
    )
model.multi.distance <- train(
        distance ~ ultrasonic + laser,
        data = multi.train.data,
        method= "lm",
        trControl = trainControl("cv", number = 10)
    )
```

-En esta parte usamos la otra parte del 20% de los datos separados.

```
predict.ultrasonic <- model.distance.ultrasonic %>%
predict(ultrasonic.test.data)
predict.laser <- model.distance.laser %>% predict(laser.test.data)
predict.multi <- model.multi.distance %>% predict(multi.test.data)
```

```
postResample(predict.ultrasonic, ultrasonic.test.data$distance)

##      RMSE  Rsquared       MAE
## 1.5587459 0.9884411 0.6708602

postResample(predict.laser, laser.test.data$distance)

##     RMSE Rsquared      MAE
## 1.430263 0.989803 1.351133

postResample(predict.multi, multi.test.data$distance)

##      RMSE  Rsquared       MAE
## 1.2553884 0.9930187 1.0816251
```

-Prueba de los modelos en base a un dataset con los datos de los sensores capturados de forma aleatoria.

```
new_df_test <- load_datasets("random_dataset_sensors_A.csv")
df_test_cleared <- new_df_test %>% select(ultrasonic, laser)
df_test_ultrasonic <- df_test_cleared %>% select(ultrasonic)
df_test_laser <- df_test_cleared %>% select(laser)

predict(model.multi.distance, df_test_cleared)

##         1         2         3         4         5         6         7
8
##  7.284619  7.284619  7.035488  7.035488  7.213829  7.213829  7.426199
7.426199
##         9        10        11        12        13        14        15
16
##  7.318648  7.318648  7.355409  7.355409  7.426199  7.426199  7.318648
7.318648
##        17        18        19        20        21        22        23
24
##  7.496989  7.496989  7.426199  7.426199  7.318648  7.318648  7.213829
7.213829
##        25        26        27        28        29        30        31
32
##  7.349282  7.349282  7.312521  7.312521  7.496989  7.496989 11.178064
11.178064
##        33        34        35        36        37        38        39
40
## 16.064234 16.064234 16.875714 16.875714 16.946504 16.946504 17.105803
17.105803
##        41        42        43        44        45        46        47
48
## 16.964885 16.964885 16.970350 16.970350 16.930856 16.930856 16.964885
16.964885
##        49        50        51        52        53        54        55
56
## 16.817178 16.817178 16.894095 16.894095 16.894095 16.894095 17.100338
```

```
17.100338
##        57        58        59        60        61        62        63
64
## 16.946504 16.946504 16.894095 16.894095 17.029548 17.029548 16.964885
16.964885
##        65        66        67        68        69        70        71
72
## 16.964885 16.964885 16.964885 16.964885 17.177255 17.177255 17.143226
17.143226
##        73        74        75        76        77        78        79
80
## 16.964885 16.964885 16.823305 16.823305 16.964885 16.964885 17.111930
17.111930
##        81        82        83        84        85        86        87
88
## 16.752515 16.752515 17.248044 17.248044 20.855597 20.855597 22.835901
22.835901
##        89        90        91        92        93        94        95
96
## 23.124526 23.124526 23.087764 23.087764 23.148372 23.148372 22.989073
22.989073
##        97        98        99       100       101       102       103
104
## 22.982946 22.982946 23.012919 23.012919 23.225288 23.225288 23.148372
23.148372
##       105       106       107       108       109       110       111
112
## 23.093891 23.093891 22.918283 22.918283 22.952312 22.952312 22.989073
22.989073
##       113       114       115       116       117       118       119
120
## 23.195316 23.195316 23.059863 23.059863 23.124526 23.124526 22.989073
22.989073
##       121       122       123       124       125       126       127
128
## 23.266106 23.266106 25.359168 25.359168 35.510370 35.510370 38.792966
38.792966
##       129       130       131       132       133       134       135
136
## 38.515271 38.515271 38.657513 38.657513 38.509806 38.509806 38.509806
38.509806
##       137       138       139       140       141       142       143
144
## 38.580596 38.580596 38.620090 38.620090 38.586723 38.586723 38.651386
38.651386
##       145       146       147       148       149       150       151
152
## 38.651386 38.651386 38.368226 38.368226 38.580596 38.580596 38.792966
38.792966
##       153       154       155       156       157       158       159
```

```
160
## 38.232773 38.232773 38.722176 38.722176 38.368226 38.368226 38.432889
38.432889
##       161       162       163       164       165       166       167
168
## 38.509806 38.509806 38.543835 38.543835 38.580596 38.580596 38.473045
38.473045
##       169       170       171       172       173       174       175
176
## 39.577120 39.577120 43.749754 43.749754 46.034331 46.034331 46.534002
46.534002
##       177       178       179       180       181       182       183
184
## 46.470001 46.470001 46.362450 46.362450 46.291660 46.291660 46.504029
46.504029
##       185       186       187       188       189       190       191
192
## 46.574819 46.574819 46.079290 46.079290 46.150080 46.150080 46.014627
46.014627
##       193       194       195       196       197       198       199
200
## 46.186841 46.186841 46.540791 46.540791 46.470001 46.470001 46.328421
46.328421
##       201       202       203       204       205       206       207
208
## 46.433239 46.433239 45.725340 45.725340 46.362450 46.362450 46.798781
46.798781
##       209       210       211       212       213       214       215
216
## 50.558182 50.558182 54.100178 54.100178 53.828610 53.828610 54.170968
54.170968
##       217       218       219       220       221       222       223
224
## 53.964725 53.964725 54.094051 54.094051 53.533858 53.533858 53.715594
53.715594
##       225       226       227       228       229       230       231
232
## 53.958598 53.958598 54.204996 54.204996 53.893935 53.893935 53.681565
53.681565
##       233       234       235       236       237       238       239
240
## 54.177094 54.177094 54.029388 54.029388 54.170968 54.170968 53.899400
53.899400
##       241       242       243       244       245       246       247
248
## 54.170968 54.170968 54.111770 54.111770 54.075670 54.075670 53.893935
53.893935
##       249       250       251       252       253       254       255
256
## 53.786384 53.786384 54.106305 54.106305 53.361644 53.361644 55.682149
```

```
55.682149
##       257       258       259       260       261       262       263
264
## 58.765205 58.765205 56.919202 56.919202 56.913737 56.913737 56.919864
56.919864
##       265       266       267       268       269       270       271
272
## 56.533957 56.533957 56.386912 56.386912 56.670071 56.670071 56.457702
56.457702
##       273       274       275       276       277       278       279
280
## 56.452236 56.452236 56.546872 56.546872 56.181330 56.181330 56.824566
56.824566
##       281       282       283       284       285       286       287
288
## 57.401816 57.401816 56.457702 56.457702 57.094811 57.094811 57.113191
57.113191
##       289       290       291       292       293       294       295
296
## 57.802710 57.802710 56.670071 56.670071 52.918206 52.918206 53.802805
53.802805
##       297       298       299       300       301       302       303
304
## 54.158912 54.158912 54.190956 54.190956 56.563036 56.563036 60.200561
60.200561
##       305       306       307       308       309       310       311
312
## 57.720586 57.720586 59.346021 59.346021 60.583478 60.583478 57.945871
57.945871
##       313       314
## 57.945871 57.945871

predict(model.distance.ultrasonic, df_test_ultrasonic)

##         1         2         3         4         5         6         7
8
##  8.641835  8.641835  8.537758  8.537758  8.641835  8.641835  8.641835
8.641835
##         9        10        11        12        13        14        15
16
##  8.537758  8.537758  8.641835  8.641835  8.641835  8.641835  8.537758
8.537758
##        17        18        19        20        21        22        23
24
##  8.641835  8.641835  8.641835  8.641835  8.537758  8.537758  8.641835
8.641835
##        25        26        27        28        29        30        31
32
##  8.624489  8.624489  8.520411  8.520411  8.641835  8.641835  8.641835
8.641835
```

```
##        33        34        35        36        37        38        39
40
## 15.059964 15.059964 17.557830 17.557830 17.557830 17.557830 18.008833
18.008833
##        41        42        43        44        45        46        47
48
## 17.609869 17.609869 18.026180 18.026180 17.713946 17.713946 17.609869
17.609869
##        49        50        51        52        53        54        55
56
## 17.592522 17.592522 17.609869 17.609869 17.609869 17.609869 17.592522
17.592522
##        57        58        59        60        61        62        63
64
## 17.557830 17.557830 17.609869 17.609869 17.592522 17.592522 17.609869
17.609869
##        65        66        67        68        69        70        71
72
## 17.609869 17.609869 17.609869 17.609869 17.609869 17.609869 17.713946
17.713946
##        73        74        75        76        77        78        79
80
## 17.609869 17.609869 17.609869 17.609869 17.609869 17.609869 18.026180
18.026180
##        81        82        83        84        85        86        87
88
## 17.609869 17.609869 17.609869 17.609869 17.401713 17.401713 22.206636
22.206636
##        89        90        91        92        93        94        95
96
## 22.622947 22.622947 22.518869 22.518869 23.091297 23.091297 22.640293
22.640293
##        97        98        99       100       101       102       103
104
## 22.622947 22.622947 23.108643 23.108643 23.108643 23.108643 23.091297
23.091297
##       105       106       107       108       109       110       111
112
## 22.536216 22.536216 22.640293 22.640293 22.536216 22.536216 22.640293
22.640293
##       113       114       115       116       117       118       119
120
## 22.622947 22.622947 22.640293 22.640293 22.622947 22.622947 22.640293
22.640293
##       121       122       123       124       125       126       127
128
## 22.622947 22.622947 22.536216 22.536216 29.630849 29.630849 37.922377
37.922377
##       129       130       131       132       133       134       135
136
```

```
## 38.338688 38.338688 37.939723 37.939723 37.922377 37.922377 37.922377
37.922377
##       137       138       139       140       141       142       143
144
## 37.922377 37.922377 38.234610 38.234610 37.939723 37.939723 37.922377
37.922377
##       145       146       147       148       149       150       151
152
## 37.922377 37.922377 37.922377 37.922377 37.922377 37.922377 37.922377
37.922377
##       153       154       155       156       157       158       159
160
## 37.939723 37.939723 37.922377 37.922377 37.922377 37.922377 37.905031
37.905031
##       161       162       163       164       165       166       167
168
## 37.922377 37.922377 37.818299 37.818299 37.922377 37.922377 37.818299
37.818299
##       169       170       171       172       173       174       175
176
## 38.338688 38.338688 40.732476 40.732476 45.797594 45.797594 47.011834
47.011834
##       177       178       179       180       181       182       183
184
## 46.630216 46.630216 46.526138 46.526138 46.526138 46.526138 46.526138
46.526138
##       185       186       187       188       189       190       191
192
## 46.526138 46.526138 46.526138 46.526138 46.526138 46.526138 46.543484
46.543484
##       193       194       195       196       197       198       199
200
## 46.630216 46.630216 46.630216 46.630216 46.630216 46.630216 46.630216
46.630216
##       201       202       203       204       205       206       207
208
## 46.526138 46.526138 46.526138 46.526138 46.526138 46.526138 46.959795
46.959795
##       209       210       211       212       213       214       215
216
## 47.983227 47.983227 57.610419 57.610419 58.044076 58.044076 57.610419
57.610419
##       217       218       219       220       221       222       223
224
## 57.627765 57.627765 57.593073 57.593073 57.610419 57.610419 57.523687
57.523687
##       225       226       227       228       229       230       231
232
## 57.610419 57.610419 57.506341 57.506341 57.627765 57.627765 57.627765
57.627765
```

```
##       233       234       235       236       237       238       239
240
## 57.627765 57.627765 57.610419 57.610419 57.610419 57.610419 58.044076
58.044076
##       241       242       243       244       245       246       247
248
## 57.610419 57.610419 58.044076 58.044076 57.541034 57.541034 57.627765
57.627765
##       249       250       251       252       253       254       255
256
## 57.523687 57.523687 57.627765 57.627765 57.523687 57.523687 59.483818
59.483818
##       257       258       259       260       261       262       263
264
## 65.607060 65.607060 65.190749 65.190749 64.774438 64.774438 64.791784
64.791784
##       265       266       267       268       269       270       271
272
## 65.502982 65.502982 65.086671 65.086671 65.086671 65.086671 65.086671
65.086671
##       273       274       275       276       277       278       279
280
## 64.670360 64.670360 65.138710 65.138710 64.705052 64.705052 64.722399
64.722399
##       281       282       283       284       285       286       287
288
## 65.555021 65.555021 65.086671 65.086671 65.086671 65.086671 65.138710
65.138710
##       289       290       291       292       293       294       295
296
## 65.086671 65.086671 65.086671 65.086671 65.086671 65.086671 73.603700
73.603700
##       297       298       299       300       301       302       303
304
## 76.014835 76.014835 77.107651 77.107651 75.806680 75.806680 76.084220
76.084220
##       305       306       307       308       309       310       311
312
## 70.065057 70.065057 69.856901 69.856901 69.752823 69.752823 69.700785
69.700785
##       313       314
## 69.700785 69.700785

predict(model.distance.laser, df_test_laser)

##         1         2         3         4         5         6         7
8
##  6.765219  6.765219  6.439088  6.439088  6.656509  6.656509  6.982639
6.982639
##         9        10        11        12        13        14        15
```

```
16
##  6.873929  6.873929  6.873929  6.873929  6.982639  6.982639  6.873929
6.873929
##        17        18        19        20        21        22        23
24
##  7.091350  7.091350  6.982639  6.982639  6.873929  6.873929  6.656509
6.656509
##        25        26        27        28        29        30        31
32
##  6.873929  6.873929  6.873929  6.873929  7.091350  7.091350 12.744283
12.744283
##        33        34        35        36        37        38        39
40
## 16.766563 16.766563 16.657853 16.657853 16.766563 16.766563 16.766563
16.766563
##        41        42        43        44        45        46        47
48
## 16.766563 16.766563 16.549143 16.549143 16.657853 16.657853 16.766563
16.766563
##        49        50        51        52        53        54        55
56
## 16.549143 16.549143 16.657853 16.657853 16.657853 16.657853 16.983984
16.983984
##        57        58        59        60        61        62        63
64
## 16.766563 16.766563 16.657853 16.657853 16.875274 16.875274 16.766563
16.766563
##        65        66        67        68        69        70        71
72
## 16.766563 16.766563 16.766563 16.766563 17.092694 17.092694 16.983984
16.983984
##        73        74        75        76        77        78        79
80
## 16.766563 16.766563 16.549143 16.549143 16.766563 16.766563 16.766563
16.766563
##        81        82        83        84        85        86        87
88
## 16.440433 16.440433 17.201404 17.201404 22.854338 22.854338 23.289179
23.289179
##        89        90        91        92        93        94        95
96
## 23.506600 23.506600 23.506600 23.506600 23.289179 23.289179 23.289179
23.289179
##        97        98        99       100       101       102       103
104
## 23.289179 23.289179 23.071759 23.071759 23.397890 23.397890 23.289179
23.289179
##       105       106       107       108       109       110       111
112
## 23.506600 23.506600 23.180469 23.180469 23.289179 23.289179 23.289179
```

```
23.289179
##       113       114       115       116       117       118       119
120
## 23.615310 23.615310 23.397890 23.397890 23.506600 23.506600 23.289179
23.289179
##       121       122       123       124       125       126       127
128
## 23.724020 23.724020 26.985328 26.985328 38.726037 38.726037 39.269589
39.269589
##       129       130       131       132       133       134       135
136
## 38.617327 38.617327 39.052168 39.052168 38.834747 38.834747 38.834747
38.834747
##       137       138       139       140       141       142       143
144
## 38.943458 38.943458 38.834747 38.834747 38.943458 38.943458 39.052168
39.052168
##       145       146       147       148       149       150       151
152
## 39.052168 39.052168 38.617327 38.617327 38.943458 38.943458 39.269589
39.269589
##       153       154       155       156       157       158       159
160
## 38.399906 38.399906 39.160878 39.160878 38.617327 38.617327 38.726037
38.726037
##       161       162       163       164       165       166       167
168
## 38.834747 38.834747 38.943458 38.943458 38.943458 38.943458 38.834747
38.834747
##       169       170       171       172       173       174       175
176
## 40.247981 40.247981 45.357364 45.357364 46.118335 46.118335 46.227046
46.227046
##       177       178       179       180       181       182       183
184
## 46.335756 46.335756 46.227046 46.227046 46.118335 46.118335 46.444466
46.444466
##       185       186       187       188       189       190       191
192
## 46.553176 46.553176 45.792205 45.792205 45.900915 45.900915 45.683494
45.683494
##       193       194       195       196       197       198       199
200
## 45.900915 45.900915 46.444466 46.444466 46.335756 46.335756 46.118335
46.118335
##       201       202       203       204       205       206       207
208
## 46.335756 46.335756 45.248653 45.248653 46.227046 46.227046 46.661887
46.661887
##       209       210       211       212       213       214       215
```

```
216
## 51.879980 51.879980 52.097400 52.097400 51.445138 51.445138 52.206110
52.206110
##       217       218       219       220       221       222       223
224
## 51.879980 51.879980 52.097400 52.097400 51.227718 51.227718 51.553849
51.553849
##       225       226       227       228       229       230       231
232
## 51.879980 51.879980 52.314821 52.314821 51.771269 51.771269 51.445138
51.445138
##       233       234       235       236       237       238       239
240
## 52.206110 52.206110 51.988690 51.988690 52.206110 52.206110 51.553849
51.553849
##       241       242       243       244       245       246       247
248
## 52.206110 52.206110 51.879980 51.879980 52.097400 52.097400 51.771269
51.771269
##       249       250       251       252       253       254       255
256
## 51.662559 51.662559 52.097400 52.097400 51.010297 51.010297 53.510634
53.510634
##       257       258       259       260       261       262       263
264
## 54.923867 54.923867 52.314821 52.314821 52.532241 52.532241 52.532241
52.532241
##       265       266       267       268       269       270       271
272
## 51.553849 51.553849 51.553849 51.553849 51.988690 51.988690 51.662559
51.662559
##       273       274       275       276       277       278       279
280
## 51.879980 51.879980 51.771269 51.771269 51.445138 51.445138 52.423531
52.423531
##       281       282       283       284       285       286       287
288
## 52.858372 52.858372 51.662559 51.662559 52.640951 52.640951 52.640951
52.640951
##       289       290       291       292       293       294       295
296
## 53.728054 53.728054 51.988690 51.988690 46.227046 46.227046 42.965738
42.965738
##       297       298       299       300       301       302       303
304
## 42.204766 42.204766 41.661214 41.661214 46.009625 46.009625 51.445138
51.445138
##       305       306       307       308       309       310       311
312
## 50.901587 50.901587 53.510634 53.510634 55.467418 55.467418 51.445138
```

```
51.445138
##       313       314
## 51.445138 51.445138
```

-Estas líneas de código lo que realizan es que guardan los modelos lineales entrenados en formato RDS, en la carpeta "models" del paquete.

```
folder <- dirname(rstudioapi::getSourceEditorContext()$path)
parentFolder <- dirname(folder)
saveRDS(model.distance.laser,
paste0(parentFolder,"/models/model_laser_distance.rds"))
saveRDS(model.distance.ultrasonic,
paste0(parentFolder,"/models/model_ultrasonic.rds"))
saveRDS(model.multi.distance,
paste0(parentFolder,"/models/model_laser_ultrasonic_distance.rds"))
```

2. Predicción de una variable categorica

En esta sección vamos a realizar la predicción de los datos obtenidos por cada sensor para categorizar las diferentes superficies, plana, concava o convexa.

```
if(!require(tidyverse))
  install.packages("tidyverse")
library(tidyverse)
if(!require(caret))
  install.packages("caret")
library(caret)

shape_df <- load_datasets("sensors_shapes_dataset.csv")
shape_df$Shape <- as.factor(shape_df$Shape)
head(shape_df)

##   count ultrasonic laser distance  Shape
## 1     1        804   142       10 convex
## 2     2        804   145       10 convex
## 3     3        804   145       10 convex
## 4     4        804   146       10 convex
## 5     5        804   146       10 convex
## 6     6        780   142       10 convex

set.seed(1)
data.train.idx <- createDataPartition(shape_df$Shape, p = 0.7, list =
FALSE)
train.data = shape_df[data.train.idx,]
test.data = shape_df[-data.train.idx,]
```

-Entrenamiento de los datos con hold out cross-validation (70-30).

```
ctrl <- trainControl(method="LGOCV", p = 0.3)
    shape.model.classificator <- train(
        Shape ~ ultrasonic + laser,
```

```
        data = train.data,
        method = "knn",
        tuneGrid = data.frame(k = 3),
        trControl = ctrl,
    )
```

-predición de los datos

```
predictions <- predict(shape.model.classificator, test.data)
confusionMatrix(predictions, test.data$Shape)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction concave convex plane
##    concave      58      0     0
##    convex        0     58     0
##    plane         2      2    60
##
## Overall Statistics
##
##                Accuracy : 0.9778
##                  95% CI : (0.9441, 0.9939)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9667
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: concave Class: convex Class: plane
## Sensitivity                  0.9667        0.9667       1.0000
## Specificity                  1.0000        1.0000       0.9667
## Pos Pred Value               1.0000        1.0000       0.9375
## Neg Pred Value               0.9836        0.9836       1.0000
## Prevalence                   0.3333        0.3333       0.3333
## Detection Rate               0.3222        0.3222       0.3333
## Detection Prevalence         0.3222        0.3222       0.3556
## Balanced Accuracy            0.9833        0.9833       0.9833
```

```
confusionMatrix(predictions, test.data$Shape)$overall["Accuracy"]
```

```
##  Accuracy
## 0.9777778
```

-Prueba con los datos aleatorios

```
random_df <- load_datasets("random_shape_dataset.csv")
random_df <- random_df %>% select("ultrasonic", "laser")
predict(shape.model.classificator, random_df)
```

```
##   [1] plane   plane   plane   plane   plane   plane   plane   plane
plane
##  [10] plane   plane   plane   plane   plane   plane   plane   plane
plane
##  [19] plane   plane   plane   plane   plane   plane   plane   plane
plane
##  [28] plane   plane   plane   plane   plane   plane   plane   plane
plane
##  [37] plane   plane   plane   plane   plane   plane   concave concave
concave
##  [46] concave concave concave concave concave concave concave concave
concave
##  [55] concave concave concave concave concave concave concave concave
concave
##  [64] concave concave concave concave concave concave concave concave
concave
##  [73] concave concave concave concave concave concave concave concave
plane
##  [82] plane   convex  convex  convex  convex  convex  convex  convex
convex
##  [91] convex  convex  convex  convex  convex  convex  convex  convex
convex
## [100] convex  convex  convex  convex  convex  convex  convex  convex
convex
## [109] convex  convex  plane   plane   convex  convex  convex  convex
convex
## [118] convex  convex  convex  concave concave concave concave concave
concave
## [127] concave concave concave concave concave concave concave concave
concave
## [136] concave concave concave concave concave concave concave concave
concave
## [145] concave concave concave concave concave concave concave concave
concave
## [154] concave concave concave concave concave concave concave convex
convex
## [163] convex  convex  convex  convex  convex  convex  convex  convex
convex
## [172] convex  convex  convex  convex  convex  convex  convex  convex
convex
## [181] convex  convex  convex  convex  convex  convex  convex  convex
convex
## [190] convex  convex  convex  convex  convex  convex  convex  convex
convex
## [199] convex  convex  convex  plane   plane   convex  convex  plane
plane
## [208] convex  convex  plane   plane   convex  convex  plane   plane
convex
## [217] convex  plane   plane   convex  convex  plane   plane   convex
convex
```

```
## [226] plane   plane   convex  convex  plane   plane   convex  convex
convex
## [235] convex  plane   plane   convex  convex  plane   plane   plane
plane
## [244] plane   plane   plane   plane   plane   plane   plane   plane
plane
## [253] plane   plane   plane   plane   plane   plane   plane   plane
plane
## [262] plane   plane   plane   plane   plane   plane   plane   plane
plane
## [271] plane   plane   plane   plane   plane   plane   plane   plane
plane
## [280] plane   plane   plane   plane   concave concave concave concave
concave
## [289] concave convex  convex  concave concave concave concave concave
concave
## [298] concave concave concave concave concave concave concave concave
concave
## [307] concave concave concave concave concave convex  convex  concave
concave
## [316] concave concave concave concave plane   convex  concave concave
concave
## [325] concave convex  convex  concave concave concave concave concave
concave
## [334] concave concave concave concave concave concave concave concave
concave
## [343] concave concave concave concave concave concave concave concave
concave
## [352] concave concave concave concave concave concave concave concave
concave
## [361] concave plane   plane   plane   plane   plane   plane   plane
plane
## [370] plane   plane   plane   plane   plane   plane   plane   plane
plane
## [379] plane   plane   plane   plane   plane   plane   plane   plane
plane
## [388] plane   plane   plane   plane   plane   plane   plane   plane
plane
## [397] plane   plane   plane   plane   plane   plane   plane   plane
plane
## [406] plane   plane   plane   plane   plane   plane   plane   plane
plane
## [415] plane   plane   plane   plane   plane   plane   plane   plane
plane
## [424] plane   plane   plane   plane   plane   plane   plane   plane
plane
## [433] convex  convex  concave concave concave concave concave concave
concave
## [442] concave concave concave concave concave concave concave concave
concave
```

```
## [451] concave concave concave concave concave concave concave concave
concave
## [460] concave concave concave concave concave concave concave concave
concave
## [469] concave concave concave concave concave concave concave concave
concave
## [478] concave concave concave concave concave concave concave concave
concave
## [487] plane   plane   concave concave concave concave concave concave
concave
## [496] concave concave concave concave concave concave concave concave
concave
## [505] concave concave concave concave concave concave concave concave
concave
## [514] concave concave concave concave concave concave concave concave
concave
## [523] concave concave plane   plane   concave concave concave concave
concave
## [532] concave concave concave concave concave concave concave concave
concave
## [541] concave concave concave concave concave concave concave concave
concave
## [550] concave concave concave concave concave concave concave concave
concave
## [559] concave concave concave concave concave concave concave concave
concave
## [568] concave concave concave plane   plane   convex  convex  convex
convex
## [577] convex  convex  convex  convex  convex  convex  convex  convex
convex
## [586] convex  convex  convex  convex  convex  convex  convex  convex
convex
## [595] convex  convex  convex  convex  convex  convex
## Levels: concave convex plane
```

-Almacenar el modelo

```
folder <- dirname(rstudioapi::getSourceEditorContext()$path)
parentFolder <- dirname(folder)
saveRDS(shape.model.classificator,
paste0(parentFolder,"/models/shape_model_classificator.rds"))
```

##Conclusiones de los modelos -Con la libreria Caret se pudo realizar el entrenamiento de los datos de una forma mas eficiente y con una interfaz fácil si se comprenden las bases de su funcionamiento. Esto también nos ahorra tiempo ya que se automatiza el proceso de construcción de modelos de regresión lineal y modelos de predicción de Knn.

-Se garantiza que los datos tengan una mejor limpieza, lo que minimiza el ruido al momento de la captura de los datos.

-Ayuda al preprocesamiento de datos de clasificación categorica, si estos datos seran empleados al momento de clasificar la forma de las superficies.

-Cuando se realizaron la toma de datos y se procesaron, y por ultimo se realiza el modelo de predicción la distancia no es un dato predictor para poder clasificar la forma de la superficie.

-El modelo Knn es un modelo que se utiliza para calsificar objetos de diferentes categorías, por lo que se basa en la similitud entre los datos y los objetos del entrenamiento..