Reporte 2 - kNN, Linear regression, and multilinear regression

Fabian Torres 109486, Emily Moreno 78108,Luis Sanchez 72717, Daniel Sierra 99932

2022-10-13

primera parte:prediccion de una variable numerica

1.1 Adquisición de datos

-Se desarrollo un sistema de adquisición (hardware y software) para capturar datos de distancia.

El sistema permite registrar datos de dos sensores los cuales pueden ser registrados en un archivo .csv elcual se puede utilizar en R,se tomo un rango de 10 a 60 cm.

##You need to capture at least 50 observations (instances). Se hizo la captura de 55 datos por sensor , y de igual manera se agregaron las respectivas distancias medidas con el flexometro.

##You should take 3 or more observations at each distance point -se tomaron 5 observaciones por cada punto de distancia.

1.2 modelo predictivo

Se utilizaron los datos adquiridos en el punto 1.1

-se hizo un pre proceso a los datos y un analisis exploratorio de datos -se entrego un modelo lineal por cada uno de los sensores -cada uno de esos modelos intenta detectar la distancia desde los sensores. -Se entrenaron dos modelos lineales regularizados, uno para cada uno de los sensores. -Se entreno un modelo de regresion multilineal para predecir la distancia del dispositivo a una pared plana.

multilineal porque las entradas son x1(lecturasenonsor 1) y x2(lectura sensor 2) y y(distancia a esa pared) -para los 5 modelos que creemos (o los que creemeos)verifiquen el funcionmiento o rendimiento usando cross validation nosotros seleccionamos criterios para organizar o rankiar nuestros modelos y seleccionar un modelo por sensor .

para los modelos que creados se verifico el rendimiento usando cross validation ,se clasificaron los modelos y se selecciono uno por sensor .

Se utilizan librerias que necesitamos para poder usar las funciones como es la libreria caret que me ayuda a poder usar los metodos

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

Se agregan las base de datos tanto como la de entrenamiento como la que deseamos predecir

```
folder <- dirname(rstudioapi::getSourceEditorContext()$path)
parentFolder <- dirname(folder)

punto1 <- read.csv(paste0(folder

                          ,"/base-de-datos-1punto-_1_ (1).csv"))

punto1.1 <- read.csv(paste0(folder

                          ,"/base-de-datos-muestreo.csv")

                     , stringsAsFactors = TRUE)
```

Visualizamos las base de datos viendo que la base de datos 1.1 no tiene la columna de distancia

```
punto1
```

```
##    SENSOR1 SENSOR2 DISTANCIA
## 1      704     752        10
## 2      700     748        10
## 3      724     728        10
## 4      696     752        10
## 5      696     752        10
## 6     1088    1156        15
## 7     1036    1052        15
## 8     1032    1052        15
## 9     1032    1076        15
## 10    1036    1156        15
## 11    1324    1380        20
## 12    1316    1700        20
## 13    1292    1380        20
## 14    1296    1332        20
## 15    1296    1332        20
## 16    1556    1580        25
## 17    1560    1528        25
## 18    1584    1556        25
## 19    1588    1552        25
## 20    1560    1528        25
## 21    1720    1688        30
```

```
## 22    1712    1632        30
## 23    1736    1632        30
## 24    1712    1640        30
## 25    1708    1636        30
## 26    2108    2048        35
## 27    2156    2064        35
## 28    2116    2088        35
## 29    2128    2088        35
## 30    2124    2032        35
## 31    2468    2400        40
## 32    2440    2396        40
## 33    2464    2368        40
## 34    5328    2416        40
## 35    2516    2396        40
## 36    2748    2648        45
## 37    2728    2696        45
## 38    2752    2724        45
## 39    2844    2848        45
## 40    2768    2720        45
## 41    3064    2992        50
## 42    3060    3016        50
## 43    3064    3016        50
## 44    3040    3144        50
## 45    3036    3064        50
## 46    3372    3372        55
## 47    3324    3364        55
## 48    3320    3336        55
## 49    3344    3360        55
## 50    3340    3340        55
## 51    3560    3532        60
## 52    3584    3552        60
## 53    3572    3528        60
## 54    3648    3552        60
## 55    3584    3552        60
```

```
punto1.1
```

```
##   SENSOR1 SENSOR2
## 1    4050    4056
## 2    4500    4560
## 3    4200    4272
## 4    3584    3520
```

```
head(punto1)
```

```
##   SENSOR1 SENSOR2 DISTANCIA
## 1     704     752        10
## 2     700     748        10
## 3     724     728        10
## 4     696     752        10
## 5     696     752        10
## 6    1088    1156        15
```

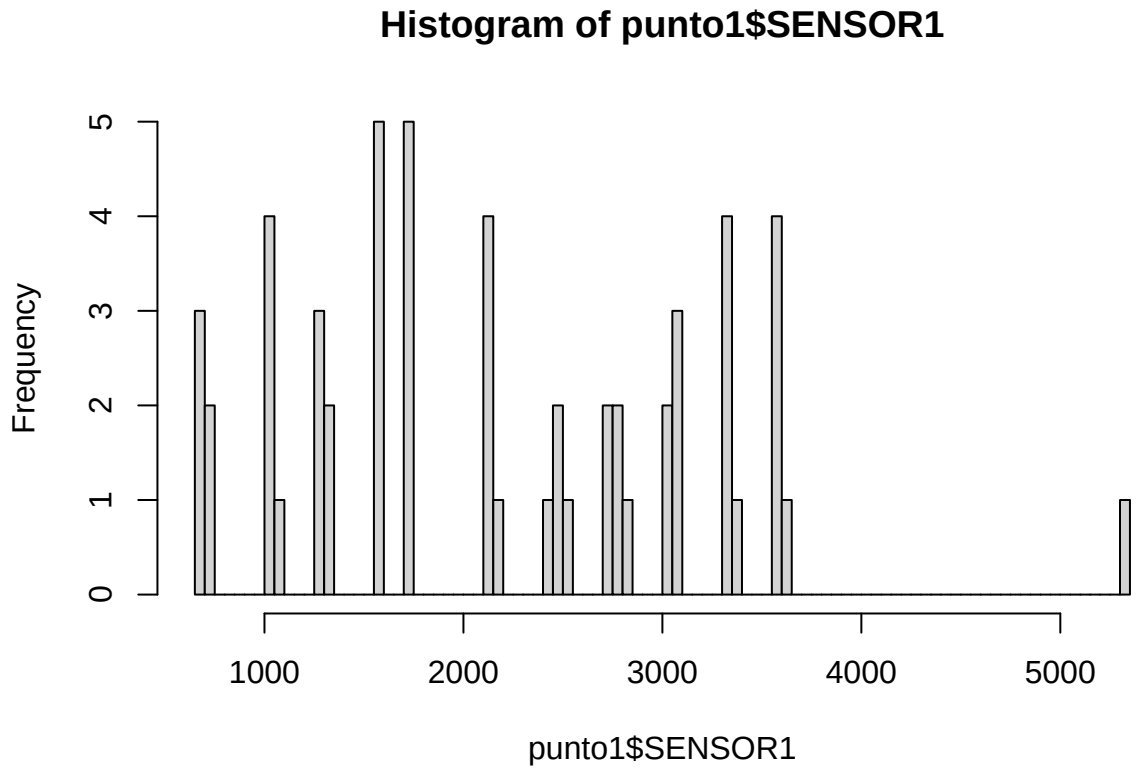
Muestra datos del sensor 1 y 2

```
summary(punto1)
```

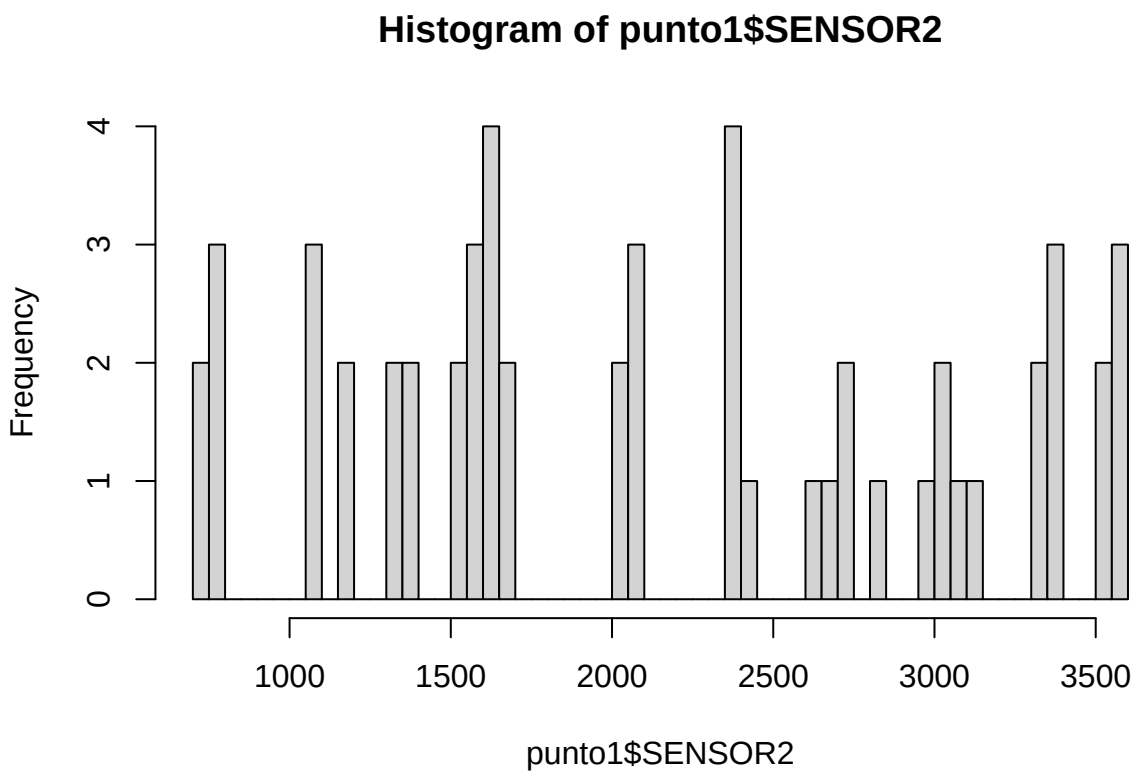
```
##     SENSOR1        SENSOR2       DISTANCIA
##  Min.   : 696   Min.   : 728   Min.   :10
##  1st Qu.:1320   1st Qu.:1454   1st Qu.:20
##  Median :2124   Median :2064   Median :35
##  Mean   :2206   Mean   :2145   Mean   :35
##  3rd Qu.:3050   3rd Qu.:3004   3rd Qu.:50
##  Max.   :5328   Max.   :3552   Max.   :60
```

Realizamos histogramas para ver la diferencia que hay en los datos de nuestros predictores ademas de ver una grafica que se comparara con nuestra variable que sera la distancia

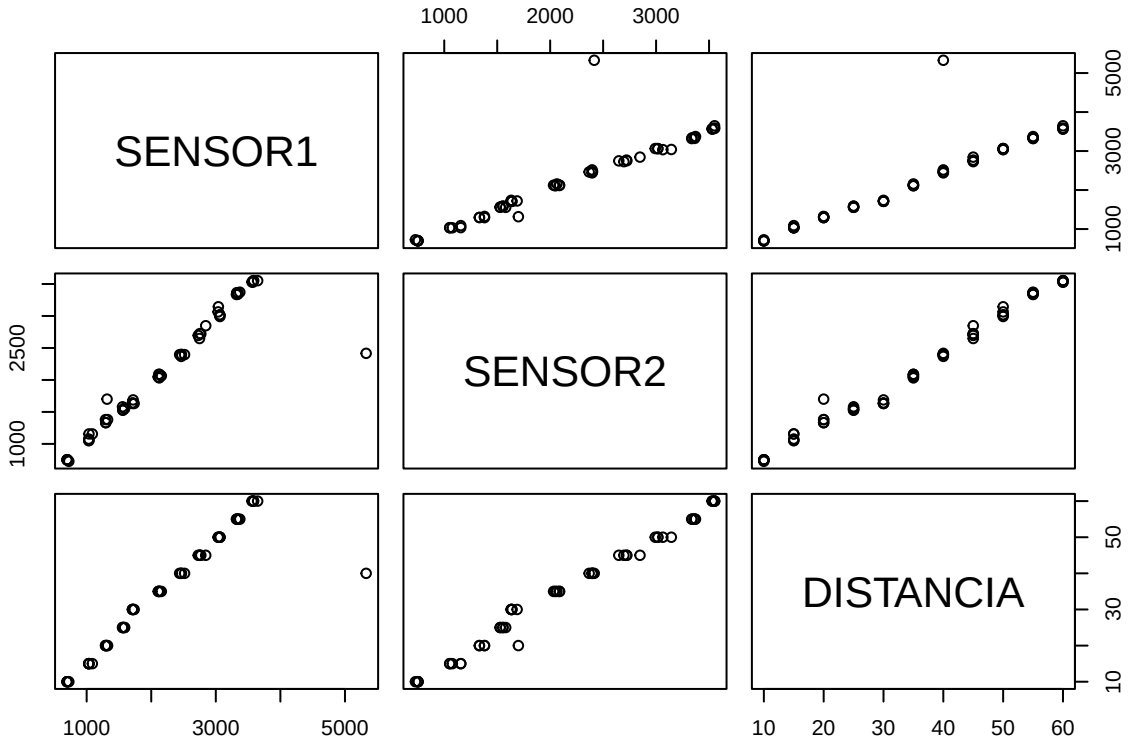
```
hist(punto1$SENSOR1, breaks = 100)
```



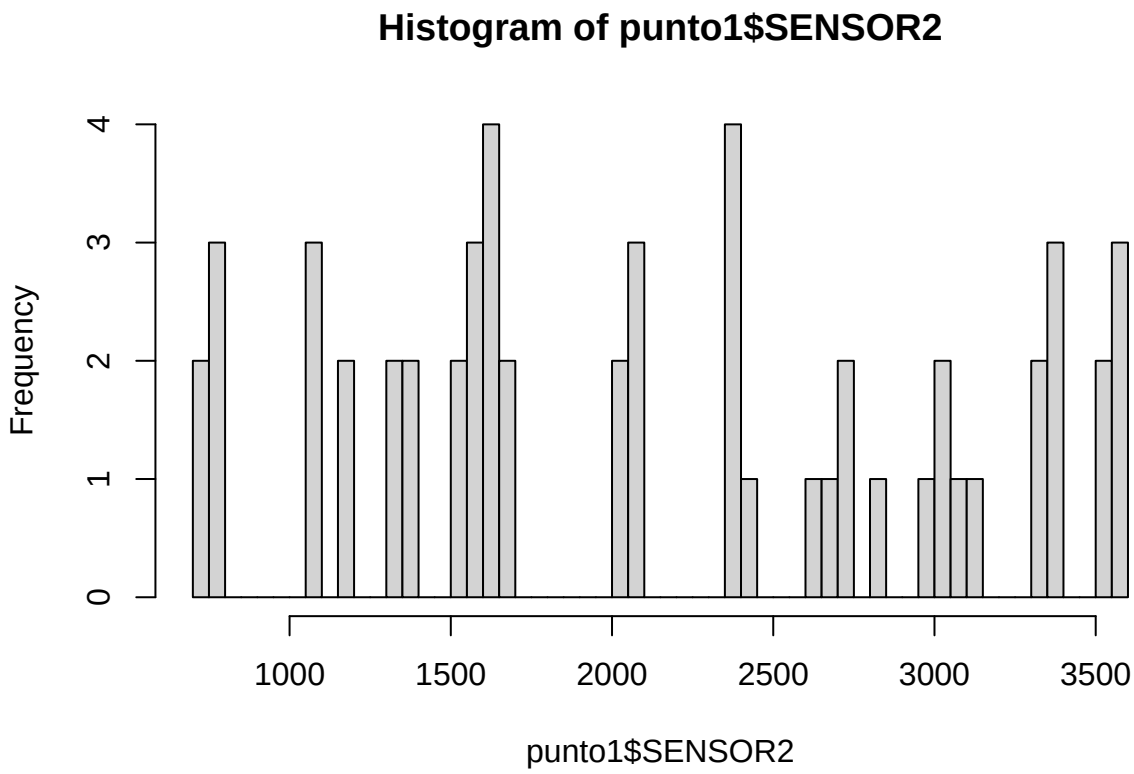

```
hist(punto1$SENSOR2, breaks = 100)
```



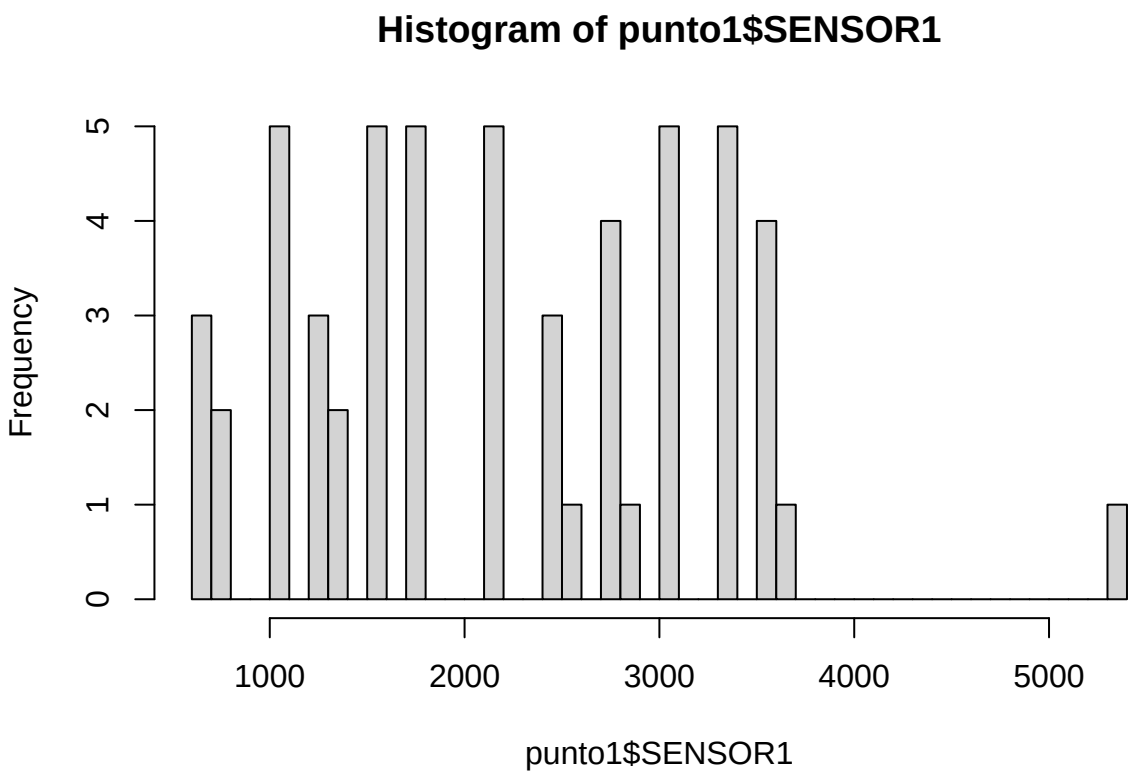

```
pairs(punto1[-c(7,8)], pch = 21

      , bg = c("red", "green3", "blue")[unclass(punto1$DISTANCIA)])
```




```
hist(punto1$SENSOR2, breaks = 50)
```




```
hist(punto1$SENSOR1, breaks = 50)
```

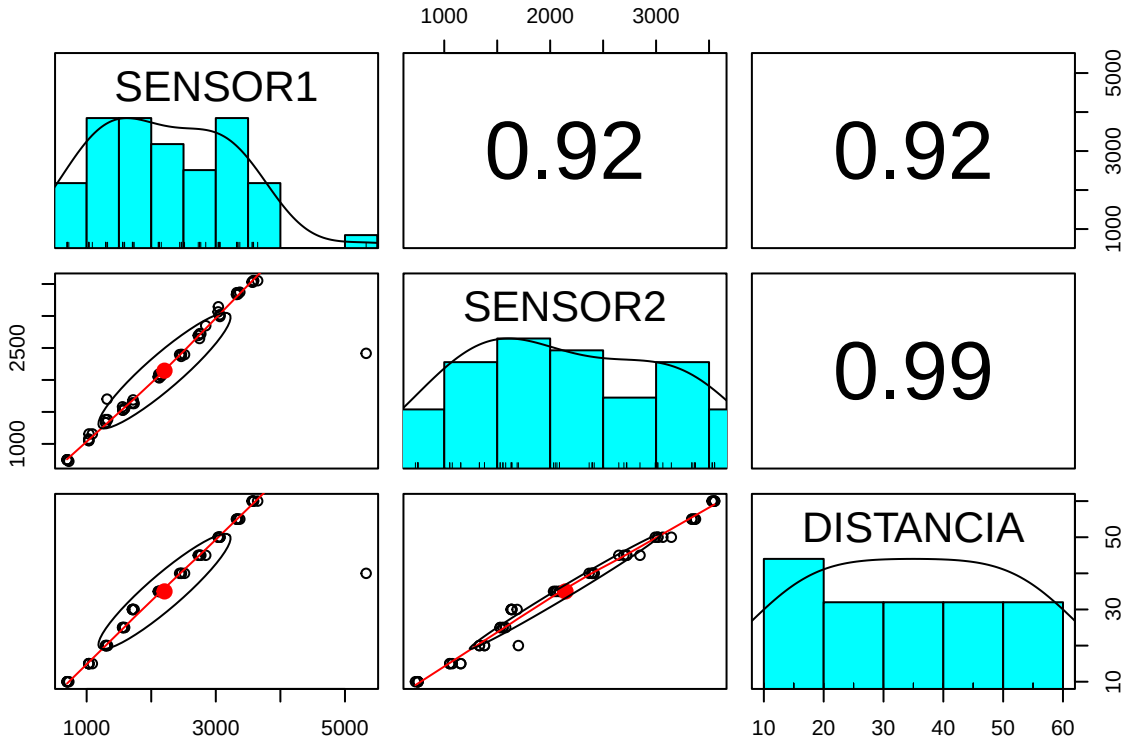


Vemos el comportamiento que tiene el sensor 1 y el sensor dos con nuestra variable siendo el sensor 2 mejor para predecir nuestra distancia

```
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
pairs.panels(punto1[c("SENSOR1",

                      "SENSOR2",

                       "DISTANCIA")]

            ,pch=21, bg=c("red","green3","blue", "orange")[unclass(punto1$DISTACIA)])
```




```
predictors <- colnames(punto1)[-3]

sample.index <- sample(1:nrow(punto1)
                       ,nrow(punto1)*0.3
                       ,replace = F)

train.data <- punto1[sample.index,c(predictors,"DISTANCIA"),drop=F]
test.data <- punto1[-sample.index,c(predictors,"DISTANCIA"),drop=F]
```

Realizamos nuestro primer modelo en donde tendremos los dos sensores en cuenta sera nuestra regrecion multilineal

```
ctrl <-trainControl(method="cv",number=5)
modelo1 <- train(DISTANCIA~.,data = punto1,method="lm",trControl=ctrl)
```

Realizamos nuestro segundo modelo donde sera una regresion lineal donde tendremos en cuenta el sensor 1

```
modelo2 <- train(DISTANCIA~SENSOR1,data = punto1,method="lm",trControl=ctrl)
```

Realizamos nuestro segundo modelo donde sera una regresion lineal donde tendremos en cuenta el sensor 2

```
modelo3 <- train(DISTANCIA~SENSOR2,data = punto1,method="lm",trControl=ctrl)
```

#1.3 validacion de los modelos

Se seleccionaron dos modelos uno para cada sensor y se tiene un modelo de regresion multilineal

```
modelo1
```

```
## Linear Regression
##
## 55 samples
##  2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 44, 44, 44, 44, 44
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   5.388213  0.8895346  2.367603
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
modelo2
```

```
## Linear Regression
##
## 55 samples
##  1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 44, 44, 44, 44, 44
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   5.576914  0.9172628  3.311104
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
modelo3
```

```
## Linear Regression
##
## 55 samples
##  1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 44, 44, 44, 44, 44
## Resampling results:
##
```

```
##   RMSE      Rsquared   MAE
##   1.826866  0.9874981  1.302714
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Se verifican los rms y los rsquared donde el rms debe ser bajo y el rsquaed debe ser mas cercano a uno como vemos el que mejor cumple esto es el modelo numero 3

Se realizan la implrementacion de los modelos a la base de datos que deseamos aplicarle los modelos usando el modelo 3 para agregarlo a la base de datos

```
p1 <- predict(modelo1,newdata=punto1.1)
p1
```

```
##        1        2        3        4
## 68.34409 77.10670 72.06282 59.03588
```

```
p2 <- predict(modelo2,newdata=punto1.1)
p2
```

```
##        1        2        3        4
## 61.56107 68.04123 63.72113 54.85050
```

```
p3 <- predict(modelo3,newdata=punto1.1)
p3
```

```
##        1        2        3        4
## 68.46991 77.29688 72.25290 59.08249
```

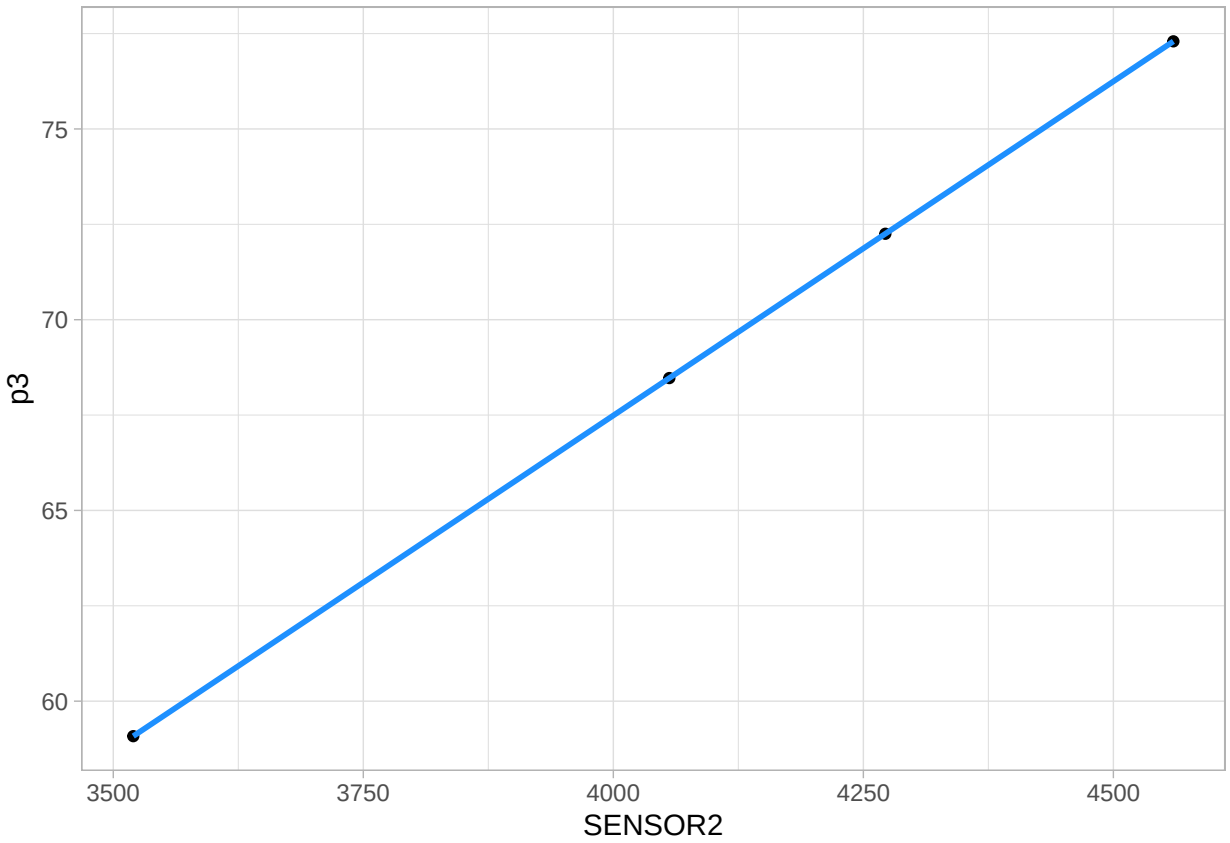
```
prediction<-punto1.1$p3 <- c(p3)
```

El rankind de los 3 modelos sera

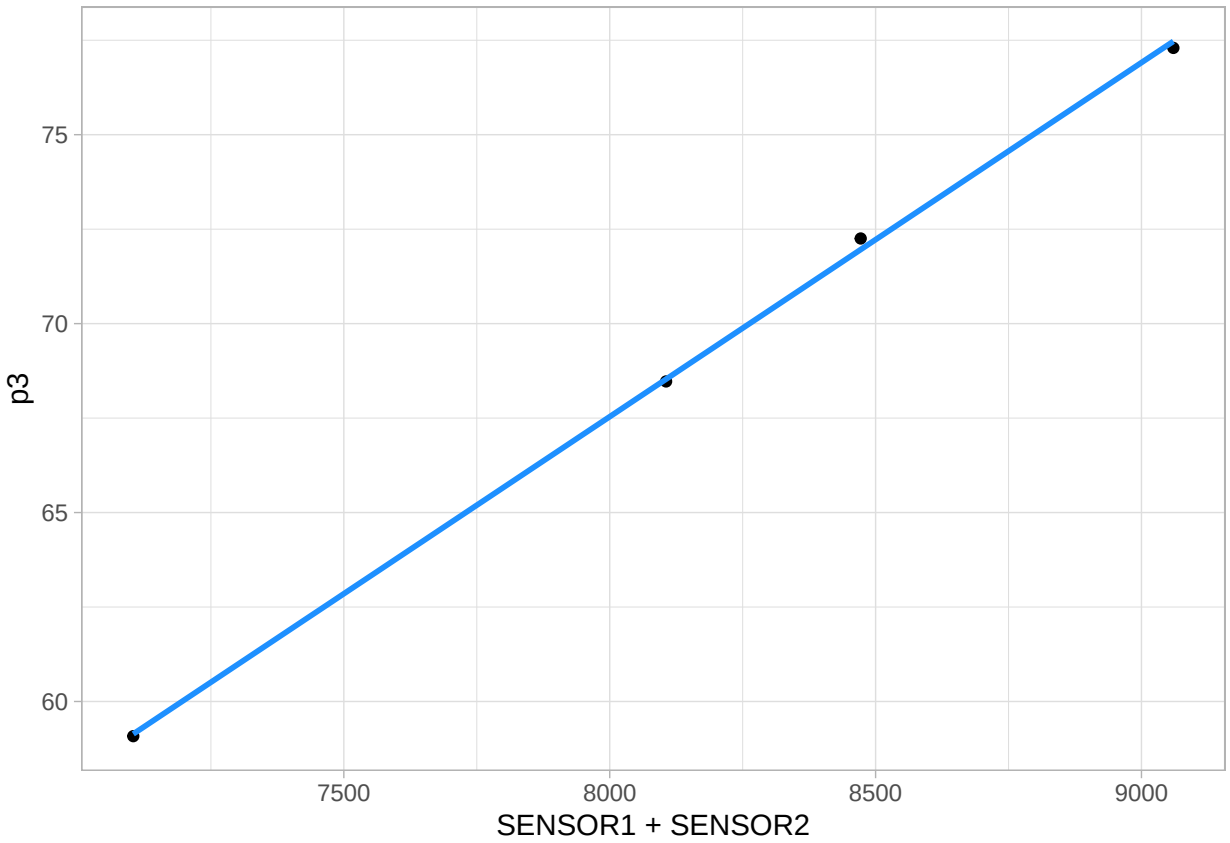
1. **modelo 3 regresion lineal con el sensor 2**
2. **modelo 2 regresion lineal con el sensor 1**
3. **modelo 1 regresion multilineal**

se grafica la relacion de los sensores de la base de datos a la cual le emos aplicado el modelo

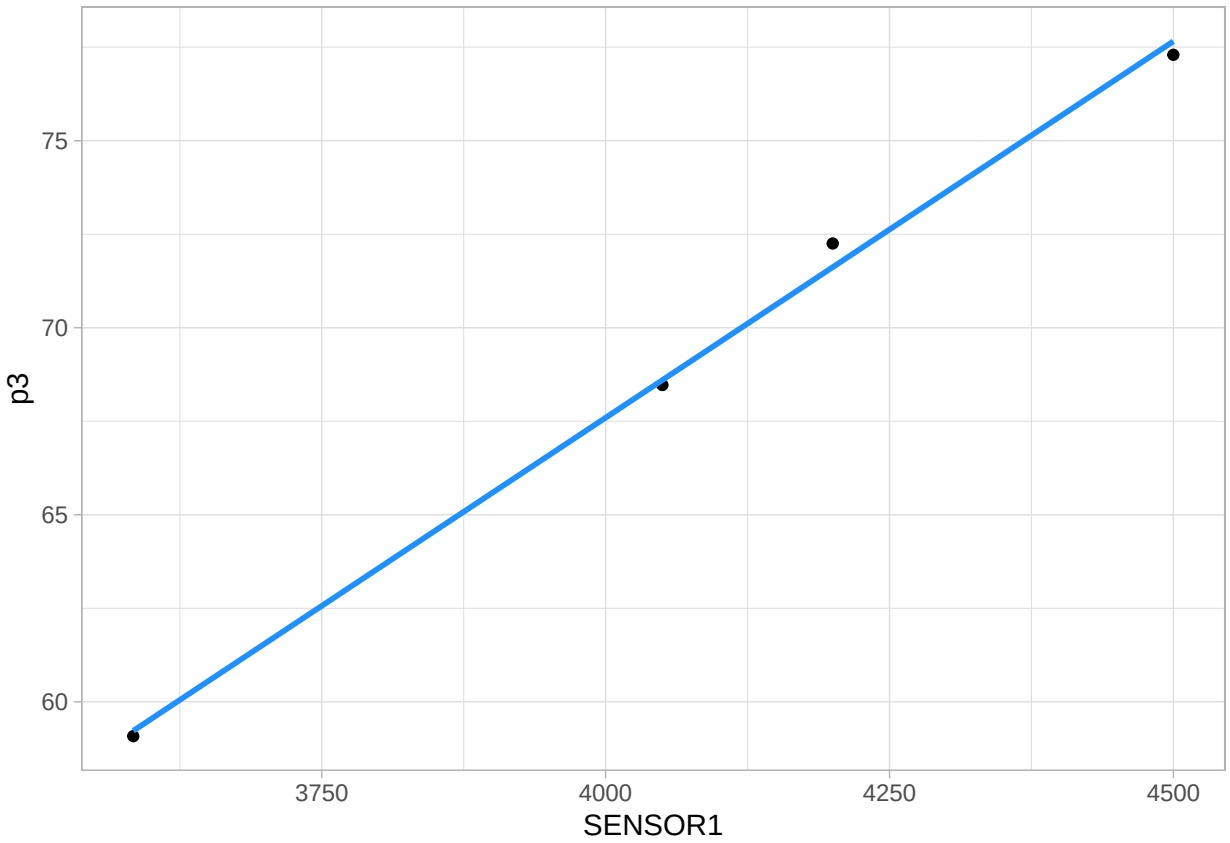
```
ggplot(punto1.1, aes(x=SENSOR2, y=p3)) +
  geom_point() +
  geom_smooth(method='lm', formula=y~x, se=FALSE, col='dodgerblue1') +
  theme_light()
```




```
ggplot(punto1.1, aes(x=SENSOR1+SENSOR2, y=p3)) +
  geom_point() +
  geom_smooth(method='lm', formula=y~x, se=FALSE, col='dodgerblue1') +
  theme_light()
```




```
ggplot(punto1.1, aes(x=SENSOR1, y=p3)) +
  geom_point() +
  geom_smooth(method='lm', formula=y~x, se=FALSE, col='dodgerblue1') +
  theme_light()
```



2.1 adquisicion de datos

se tienen almenos 150 observaciones tenemos 50 observaciones por clase en un archivo separado por comas

2.2

Utilizamos los datos adquiridos en 2.1 para: • Preprocesar sus datos y realizar un Análisis Exploratorio de Datos. • Entrenar un modelo de clasificación kNN para predecir la forma de la pared. • Probar el rendimiento del modelo utilizando cross validation de (70-30). • Probar el rendimiento del modelo mediante el uso de cross validation de 5 veces (validación cruzada de k veces) • Probar el rendimiento del modelo utilizando 3 cross validation repetidas de 10 veces. • Clasificar los modelos y se selecciono uno paraser el modelo final.

2.3 Validación del modelo

La validación se realizará utilizando el sistema desarrollado en una defensa en vivo del informe. • se Prepara un entorno adecuado para realizar la defensa

- se Capturan algunas muestras de datos de cada clase posible.

se Utiliza R y los nuevos datos recopilados para demostrar el funcionamiento de los modelos

```
library(tidyverse)
library(dplyr)
library(caret)
```

Se agregan la base de datos de entreno y la que vamos a predecir

```
folder <- dirname(rstudioapi::getSourceEditorContext()$path)
parentFolder <- dirname(folder)

punto2 <- read.csv(paste0(folder

                          ,"/BASE_DATOS_TRAIN.csv"))

punto2.1 <- read.csv(paste0(folder

                            ,"/DATASET_PRUEBA.csv")

                     , stringsAsFactors = TRUE)




head(punto2)
```

```
##   SENSOR1 SENSOR2 SENSOR3  TIPO
## 1    1016     888    1003 PLANA
## 2    1099     895    1012 PLANA
## 3    1099     895    1012 PLANA
## 4    1099     871    1011 PLANA
## 5    1099     894    1012 PLANA
## 6    1099     903    1011 PLANA
```

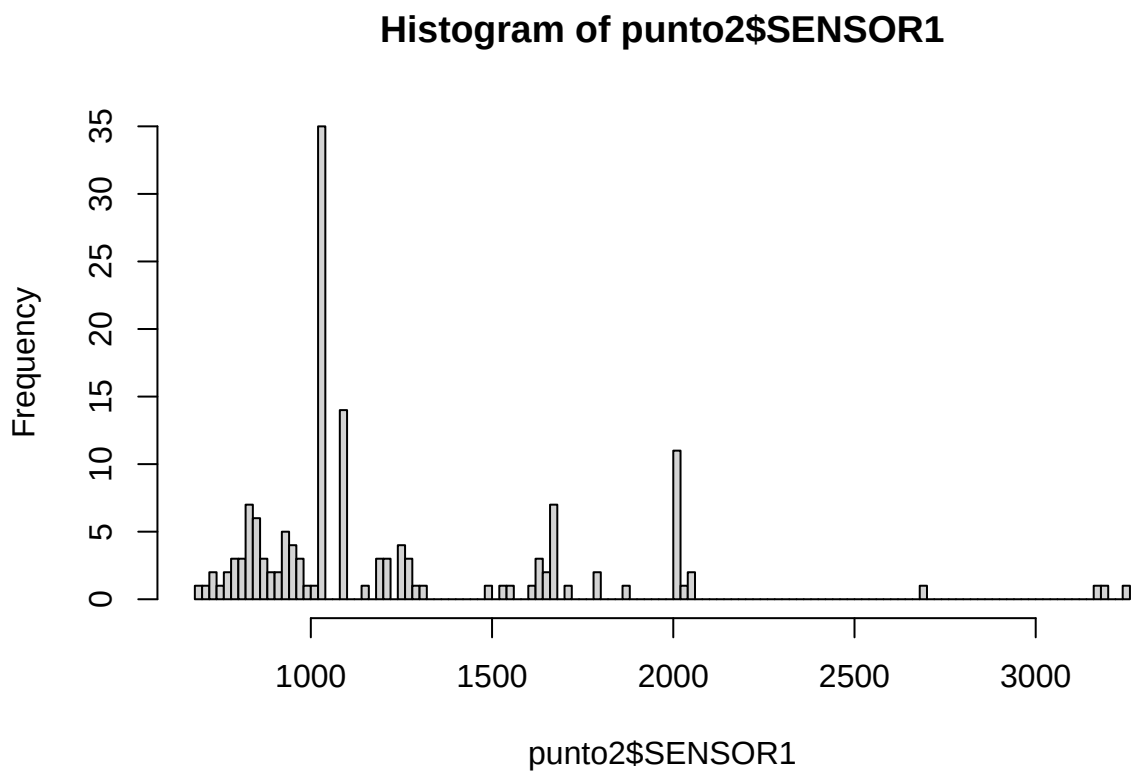
Se revisan los datos de nuestro data frame de entreno

```
summary(punto2)
```

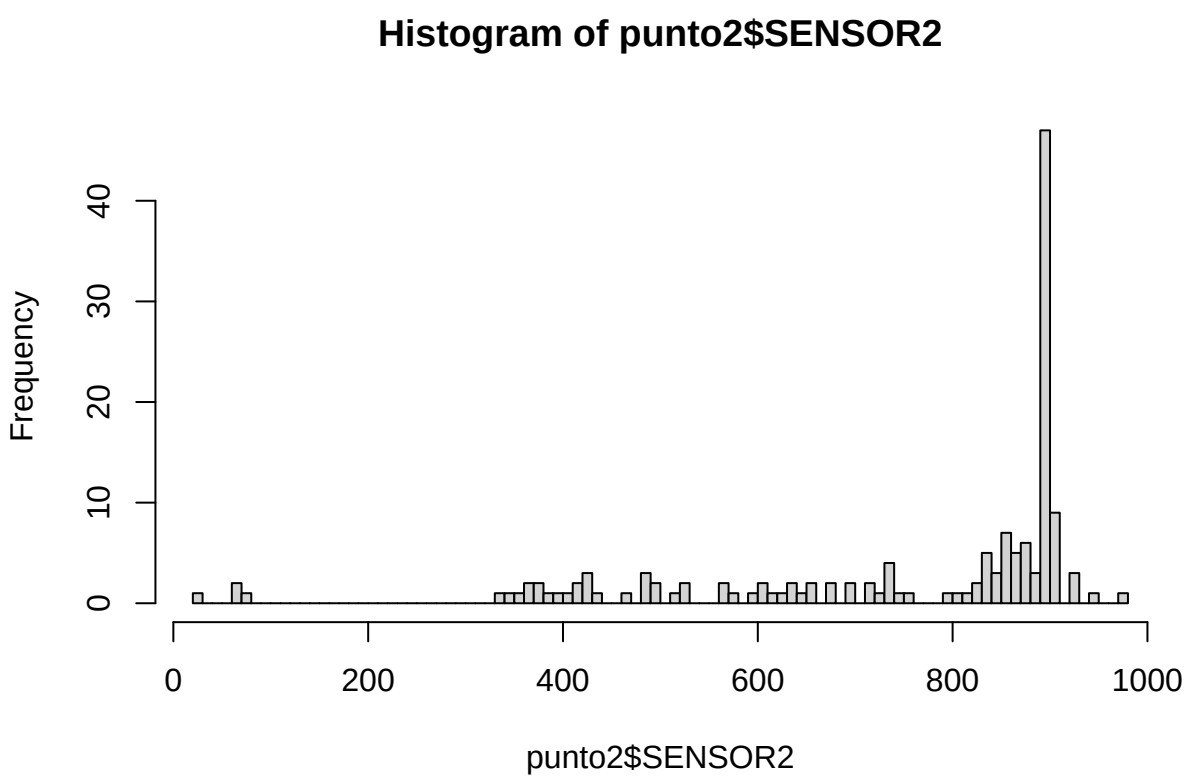
```
##     SENSOR1         SENSOR2         SENSOR3         TIPO
##  Min.   : 687   Min.   : 24.0   Min.   : 617   Length:150
##  1st Qu.: 941   1st Qu.:635.2   1st Qu.: 881   Class :character
##  Median :1025   Median :860.5   Median :1011   Mode  :character
##  Mean   :1236   Mean   :748.3   Mean   :1224
##  3rd Qu.:1445   3rd Qu.:895.0   3rd Qu.:1230
##  Max.   :3246   Max.   :973.0   Max.   :3369
```

Se realizan histogramas para poder analizar los datos de cada sensor

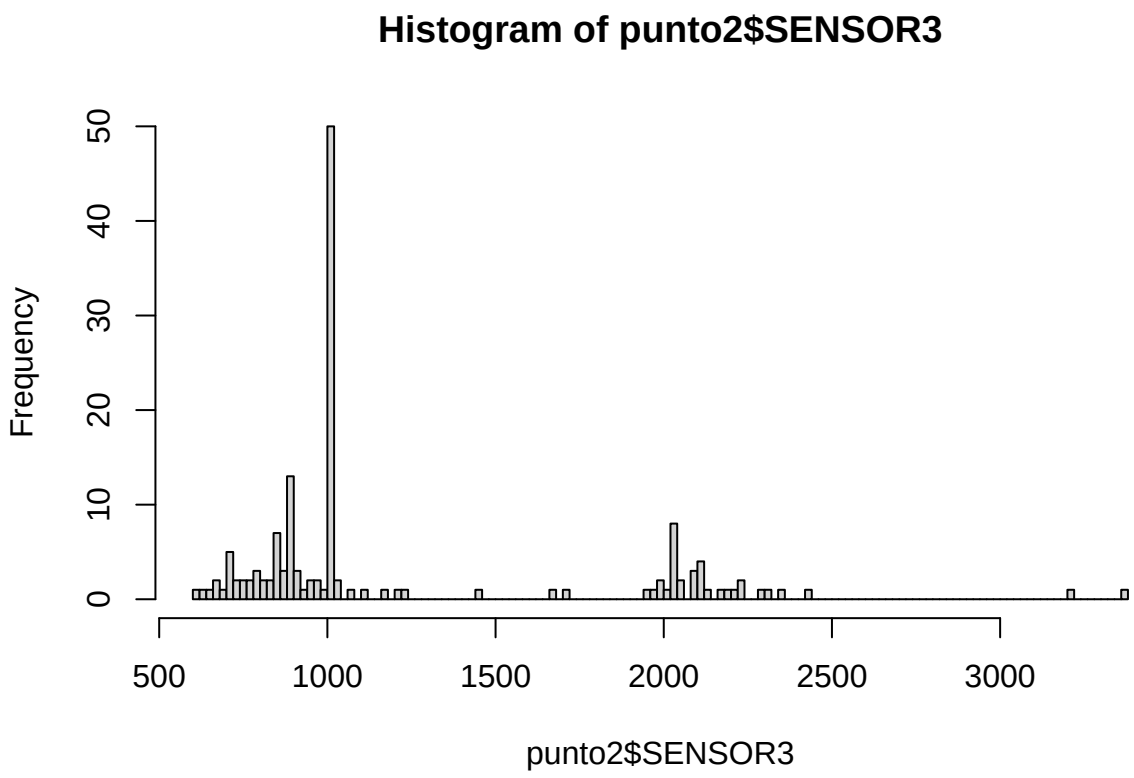
```
hist(punto2$SENSOR1, breaks = 100)
```




```
hist(punto2$SENSOR2, breaks = 100)
```

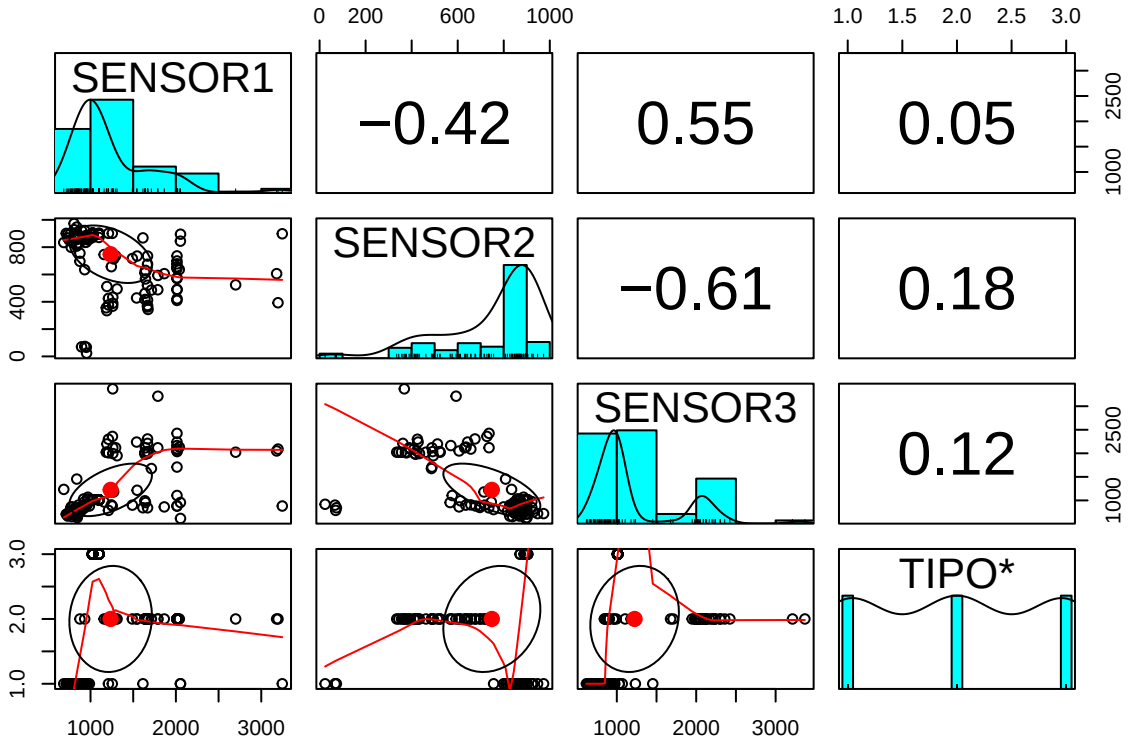



```
hist(punto2$SENSOR3, breaks = 100)
```



Se realiza una grafica de comparacion para revisar cada uno de los sensores con nuestra variable a predecir

```
library(psych)
pairs.panels(punto2[c("SENSOR1",

                      "SENSOR2",

                      "SENSOR3",

                      "TIPO")]

             ,pch=21, bg=c("red","green3","blue", "orange")[unclass(punto2$TIPO)])
```




```
predictors <- colnames(punto2)[-3]

sample.index <- sample(1:nrow(punto2)
                       ,nrow(punto2)*0.3
                       ,replace = F)

train.data <- punto2[sample.index,c(predictors,"TIPO"),drop=F]
test.data <- punto2[-sample.index,c(predictors,"TIPO"),drop=F]
```

Modelo 1 knn sin procesamiento

```
ctrl <-trainControl(method="cv",number=5)


modelo2k <- train(TIPO~.,data = punto2,method="knn",trControl=ctrl)
modelo2k
```

```
## k-Nearest Neighbors
##
## 150 samples
##   3 predictor
##   3 classes: 'CONCAVA', 'CONVEXA', 'PLANA'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
```

```
## Summary of sample sizes: 120, 120, 120, 120, 120
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.9066667  0.86
##   7  0.9000000  0.85
##   9  0.8933333  0.84
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

Modelo 2 knn con procesamiento

```
modelo3 <- train(TIPO~.,data = punto2,method="knn",preProcess=c("center","scale"),trControl=ctrl)
modelo3
```

```
## k-Nearest Neighbors
##
## 150 samples
##   3 predictor
##   3 classes: 'CONCAVA', 'CONVEXA', 'PLANA'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 120, 120, 120, 120, 120
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.9200000  0.88
##   7  0.8800000  0.82
##   9  0.8666667  0.80
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

Modelo 3 knn con grid knn

```
knnGrid <- expand.grid(k=c(1,5,10,30))
modelo4 <- train(TIPO~.,data = punto2,method="knn",preProcess=c("center","scale"),tuneGrid=knnGrid,trCo
modelo4
```

```
## k-Nearest Neighbors
##
## 150 samples
##   3 predictor
##   3 classes: 'CONCAVA', 'CONVEXA', 'PLANA'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 120, 120, 120, 120, 120
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
```

```
##    1  0.9533333  0.93
##    5  0.9200000  0.88
##   10  0.8866667  0.83
##   30  0.7933333  0.69
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

Al ver los 3 modelos podemos observar el kappa y el accuaracy ademas de que cada modelo nos dice cual k es mejor usar el modelo 3 le damos diferentes valores de k y el toma la mejor opcion en este caso la mejor opcion es el k=1

```
P1 <- predict(modelo2k,newdata=punto2.1,)
P1
```

```
##   [1] CONCAVA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONCAVA CONCAVA CONCAVA
##  [10] CONVEXA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONVEXA
##  [19] CONCAVA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONCAVA CONVEXA CONCAVA
##  [28] CONCAVA CONVEXA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA
##  [37] CONCAVA CONCAVA CONCAVA CONCAVA CONVEXA CONVEXA CONCAVA CONVEXA CONCAVA
##  [46] CONVEXA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA
##  [55] CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONVEXA CONCAVA CONCAVA
##  [64] CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA PLANA   PLANA   PLANA   PLANA
##  [73] PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA
##  [82] PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA
##  [91] PLANA   PLANA   PLANA   PLANA   PLANA   CONCAVA CONCAVA CONCAVA CONCAVA
## [100] CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONVEXA CONCAVA CONCAVA CONCAVA
## [109] CONCAVA CONCAVA PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA
## [118] PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   CONVEXA CONVEXA CONVEXA
## [127] CONVEXA PLANA   PLANA   CONCAVA CONVEXA CONVEXA CONVEXA PLANA   PLANA
## [136] PLANA   PLANA   PLANA   PLANA
## Levels: CONCAVA CONVEXA PLANA
```

```
P2 <- predict(modelo3,newdata=punto2.1,)
P2
```

```
##   [1] CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONCAVA CONCAVA CONCAVA
##  [10] CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONVEXA
##  [19] CONCAVA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA
##  [28] CONVEXA CONVEXA CONVEXA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA
##  [37] CONCAVA CONCAVA CONCAVA CONCAVA CONVEXA CONVEXA CONCAVA CONVEXA CONCAVA
##  [46] CONVEXA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA
##  [55] CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONVEXA CONCAVA CONCAVA
##  [64] CONCAVA CONCAVA CONCAVA CONCAVA CONVEXA PLANA   PLANA   PLANA   PLANA
##  [73] PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA
##  [82] PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA
##  [91] PLANA   PLANA   PLANA   PLANA   CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA
## [100] CONCAVA CONCAVA CONCAVA CONVEXA CONCAVA CONVEXA CONVEXA CONCAVA CONCAVA
## [109] CONCAVA CONCAVA PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA
## [118] PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   CONCAVA CONCAVA CONCAVA
## [127] CONCAVA PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA
## [136] PLANA   PLANA   PLANA   PLANA
## Levels: CONCAVA CONVEXA PLANA
```

```r
prediction<-punto2.1$P2 <- c(P2)


P3 <- predict(modelo4,newdata=punto2.1,)
P3
```

```
##   [1] CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONCAVA CONCAVA CONCAVA
##  [10] CONVEXA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONCAVA CONVEXA
##  [19] CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA
##  [28] CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONCAVA CONCAVA CONCAVA
##  [37] CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA
##  [46] CONVEXA CONVEXA CONVEXA CONCAVA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA
##  [55] CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONCAVA CONVEXA CONVEXA CONVEXA
##  [64] CONCAVA CONVEXA CONCAVA CONVEXA CONVEXA PLANA   PLANA   PLANA   PLANA
##  [73] PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA
##  [82] PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA
##  [91] PLANA   PLANA   PLANA   PLANA   CONCAVA CONCAVA CONCAVA CONCAVA CONVEXA
## [100] CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA CONVEXA
## [109] CONCAVA CONCAVA PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA
## [118] PLANA   PLANA   PLANA   CONCAVA PLANA   CONCAVA CONCAVA CONCAVA CONCAVA
## [127] CONCAVA PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA   PLANA
## [136] PLANA   PLANA   PLANA   PLANA
## Levels: CONCAVA CONVEXA PLANA
```