kNN, Linear regression, and multilinear regression

Eduard Romero / Juan David Cepeda / Carlos Bejarano

2023-04-18

1.1 kNN, Linear regression, and multilinear regression

For the acquisition of our data, we will be using the Arduino micro controller with two sensors, an HC-SR04 ultrasonic sensor and the other an analog SHARP 0A41SK sensor.

The HC-SR04 ultrasonic sensor works by sending out high frequency sound waves and then measuring the time it takes for those sound waves to bounce back off of an object and return to the sensor. The sensor consists of two main components: a transmitter and a receiver.

Code Arduino for sensor Ultrasonic HC-SR04

```
# Pin digital 12 Trigger sensor
const int Trigger = 12;
# Pin Digital 11 Echo sensor
const int Echo = 11;

void setup() {
  # Begin comunication Serial
  Serial.begin(9600);
  # Set pin as Output
  pinMode(Trigger, OUTPUT);
  # Set pin as Input
  pinMode(Echo, INPUT);
  digitalWrite(Trigger, LOW);
}

void loop()
{
  # Variable for calculation time Echo
  long t;

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);
   # Send Pulse 10uS
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH);

  Serial.print("Time: ");
  Serial.print(t);
  Serial.println();
  delay(2000);
}
```

The Sharp GP2Y0A21 infrared sensor works by emitting an infrared (IR) beam and then measuring the distance to an object based on the reflection of that beam. The sensor consists of an IR emitter and a receiver.

Code Arduino for sensor SHARP GP2Y0A21

```
void setup() {
  // Comunicación seria a 9600 baudios
  Serial.begin(9600);
}

void loop() {
  // Leemos la entrada analógica 0 :
  int ADC_SHARP = analogRead(A0);
  Serial.println(ADC_SHARP);
  delay(10);
}
```

1.2 Predict Model

- 1.2.1 Pre-process your data (if required) and to perform an Exploratory Data Analysis.
- 1.2.2 Train a linear model per sensor to predict the distance detected by each sensor.

Analysis for the sensor ADC1 - ADC2

```
# Import the librarys
library (tidyverse)
```

```
## Warning: package 'readr' was built under R version 4.2.3
```
```
library (caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```
```
library(psych)
```

```
## Warning: package 'psych' was built under R version 4.2.3
```
```
library(ggplot2)

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

#Read CSV File
Sensors1 <- read_csv(file = paste0(parentFolder, "/Datasets/Sensor.csv")) %>% as.data.frame()
#Show our Dataset Sensors1
Sensors1
```

```
##     ADC1 ADC2 DISTANCE
## 1   636  471       10
## 2   642  472       10
## 3   635  471       10
## 4   642  471       10
## 5   636  472       10
## 6   874  341       15
## 7   873  342       15
## 8   873  345       15
```

```
## 9    874   345       15
## 10   874   340       15
## 11  1153   271       20
## 12  1157   298       20
## 13  1157   270       20
## 14  1157   266       20
## 15  1156   276       20
## 16  1418   217       25
## 17  1424   218       25
## 18  1452   217       25
## 19  1451   217       25
## 20  1425   217       25
## 21  1699   176       30
## 22  1696   188       30
## 23  1702   188       30
## 24  1700   187       30
## 25  1701   189       30
## 26  1999   166       35
## 27  1999   166       35
## 28  2000   167       35
## 29  2000   166       35
## 30  1999   166       35
## 31  2259   145       40
## 32  2260   145       40
## 33  2265   145       40
## 34  2289   128       40
## 35  2283   145       40
## 36  2565   132       45
## 37  2542   132       45
## 38  2548   128       45
## 39  2541   133       45
## 40  2548   132       45
## 41  2841   120       50
## 42  2847   120       50
## 43  2840   119       50
## 44  2866   120       50
## 45  2841   119       50
## 46  3140   112       55
## 47  3168   111       55
## 48  3168   112       55
## 49  3160   111       55
## 50  3162   112       55
## 51  3365   103       60
## 52  3359   103       60
## 53  3392   103       60
## 54  3366   103       60
## 55  3361   103       60
```
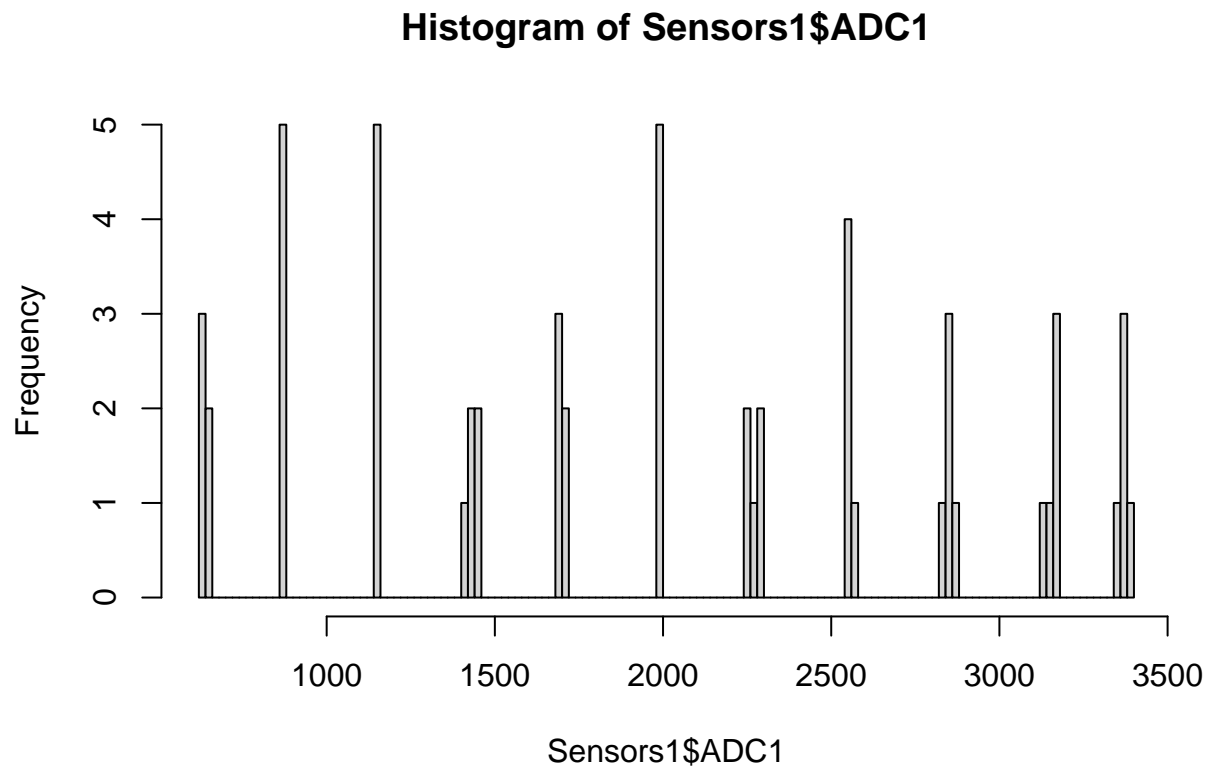
```
# Give our a summary for variables ADC1 ADC2 And DISTANCE
summary(Sensors1)
```

```
##       ADC1            ADC2          DISTANCE
##  Min.   : 635   Min.   :103   Min.   :10
##  1st Qu.:1157   1st Qu.:120   1st Qu.:20
##  Median :1999   Median :166   Median :35
```

```
##  Mean  :2000   Mean  :206   Mean  :35
##  3rd Qu.:2840  3rd Qu.:268  3rd Qu.:50
##  Max.  :3392   Max.  :472   Max.  :60
```
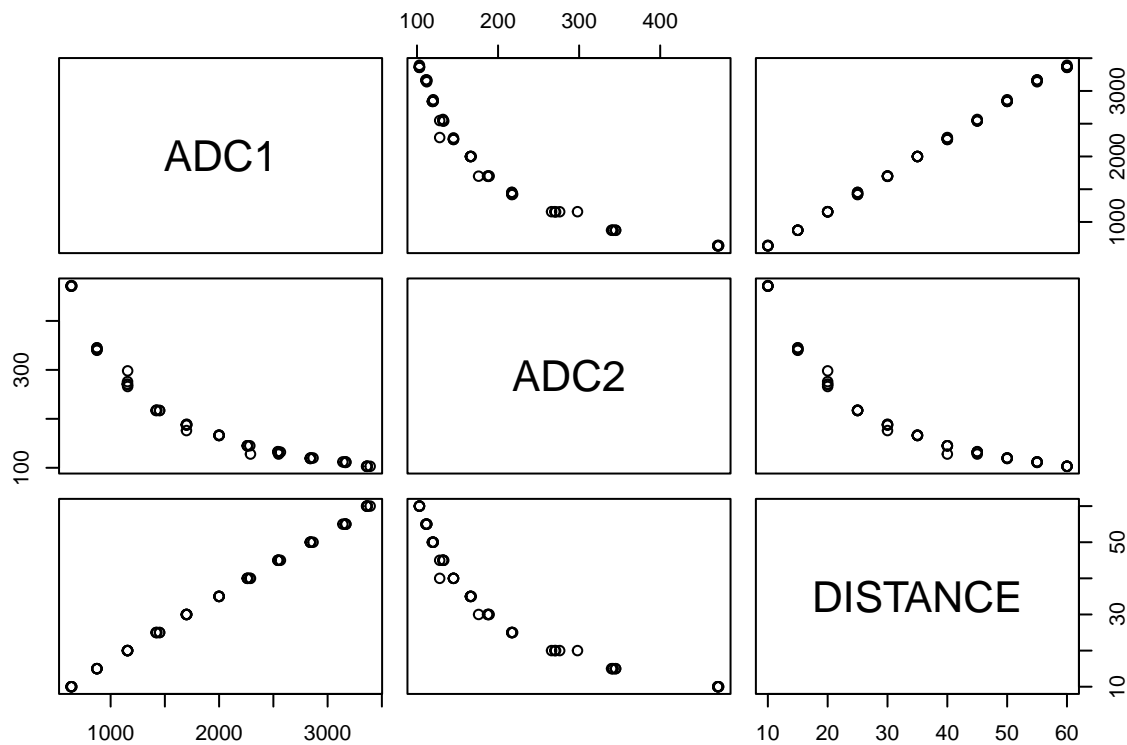
```r
# Histogram of the linear model ADC1
hist(Sensors1$ADC1,breaks = 100)
```

Histogram of Sensors1$ADC1


```r
# Histogram of the linear model ADC2
hist(Sensors1$ADC2,breaks = 100)
```

Histogram of Sensors1$ADC2
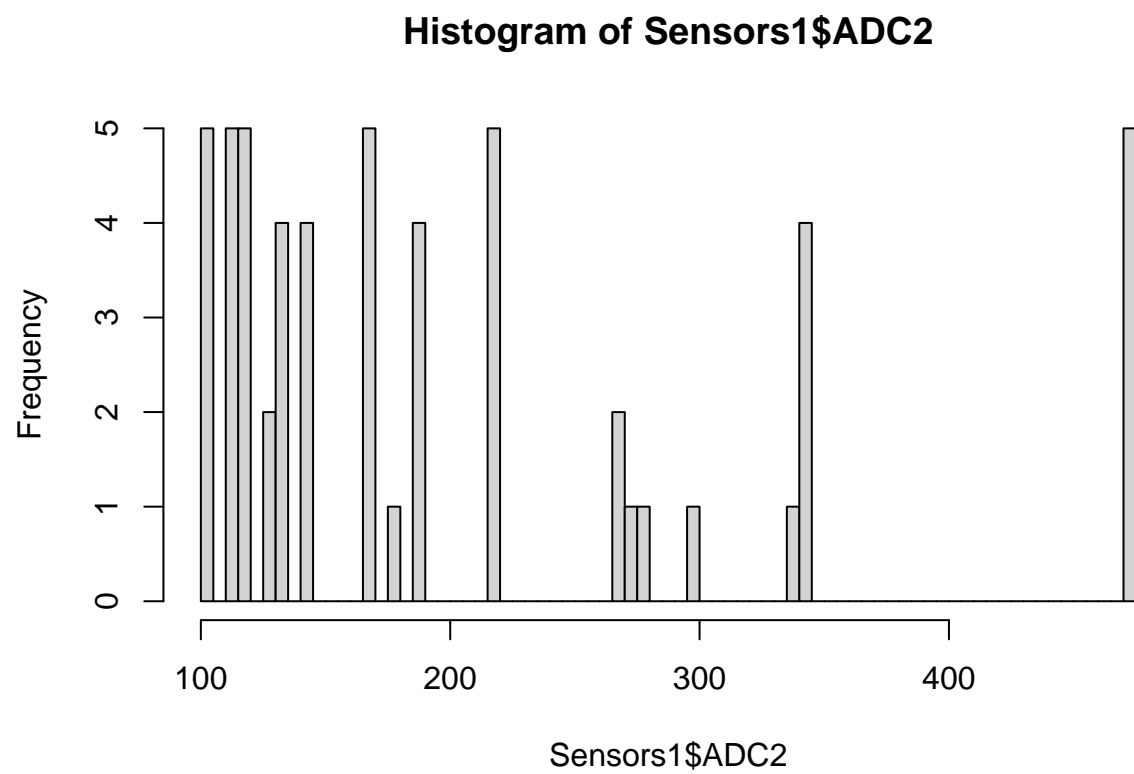

```
pairs(Sensors1[-c(7,8)], pch = 21
, bg = c("red", "green3", "blue")[unclass(Sensors1$DISTANCE)])
```

```r
# Select the corresponding columns
Data1 <- Sensors1[, c("DISTANCE", "ADC1")]
Data2 <- Sensors1[, c("DISTANCE", "ADC2")]
# Split data into training and test sets ADC1 - ADC2
set.seed(123)
trainIndex1 <- createDataPartition(Data1$DISTANCE, p = 0.8, list = FALSE)
trainData1 <- Data1[trainIndex1, ]
testData1 <- Data1[-trainIndex1, ]

trainIndex2 <- createDataPartition(Data2$DISTANCE, p = 0.8, list = FALSE)
trainData2 <- Data2[trainIndex2, ]
testData2 <- Data2[-trainIndex2, ]
# Train the linear model for distance and ADC1 - ADC2
Model1 <- lm(DISTANCE ~ ADC1, data = trainData1)
Model2 <- lm(DISTANCE ~ ADC2, data = trainData2)
# Evaluate model performance for distance and ADC1 - ADC2
predictions1 <- predict(Model1, newdata = testData1)
rmse1 <- sqrt(mean((predictions1 - testData1$DISTANCE)^2))
cat(sprintf("RMSE for DISTANCE y ADC1: %.2f\n", rmse1))
```

```
## RMSE for DISTANCE y ADC1: 0.35
```

```r
predictions2 <- predict(Model2, newdata = testData2)
rmse2 <- sqrt(mean((predictions2 - testData2$DISTANCE)^2))
cat(sprintf("RMSE for DISTANCE y ADC2: %.2f\n", rmse2))
```

```
## RMSE for DISTANCE y ADC2: 6.77
```

> [!NOTE]
> An analysis of our predictors will be that the ADC1 sensor will have a better response to implement the ADC1 sensor. The value of RMSE closer to 0 will indicate a greater precision in the prediction of the implemented model.

- 1.2.4 Train a multilinear regression using the data from the 2 sensors to predict the distance to the wall.

```
# Train multilinear regression model
AllModel <- lm(DISTANCE ~ ADC1 + ADC2, data = Sensors1)
# Predict using the trained model
predicted_values <- predict(AllModel, Sensors1)
# Evaluate model performance
rmse <- caret::RMSE(predicted_values, Sensors1$DISTANCE)
r_squared <- summary(AllModel)$r.squared
# Print model summary and evaluation metrics
print(summary(AllModel))
```

```
##
## Call:
## lm(formula = DISTANCE ~ ADC1 + ADC2, data = Sensors1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.78861 -0.17296  0.01641  0.20323  0.84470
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.9743746  0.4334817   2.248   0.0289 *
## ADC1         0.0174501  0.0001211 144.059  < 2e-16 ***
## ADC2        -0.0042143  0.0009714  -4.338 6.63e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3507 on 52 degrees of freedom
## Multiple R-squared:  0.9995, Adjusted R-squared:  0.9995
## F-statistic: 5.587e+04 on 2 and 52 DF,  p-value: < 2.2e-16
```

```
cat(paste0("RMSE: ", rmse, "\n"))
```

```
## RMSE: 0.341013403282297
```

```
cat(paste0("R-squared: ", r_squared, "\n"))
```

```
## R-squared: 0.999534839435127
```

- 1.2.5 Test the performance in your models by using cross-validation.

```
#Cross-validation for Model 1 ADC1
set.seed(123)
Model1 <- train(DISTANCE ~ ADC1, data = Sensors1, method = "lm", trControl = trainControl(method = "cv"
print(Model1)
```

```
## Linear Regression
##
## 55 samples
##  1 predictor
##
```

```
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 49, 48, 49, 50, 49, 51, ...
## Resampling results:
##
##   RMSE        Rsquared  MAE
##   0.4127814   0.99962   0.32821
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
#Cross-validation for Model 2 ADC2
set.seed(123)
Model2 <- train(DISTANCE ~ ADC2, data = Sensors1, method = "lm", trControl = trainControl(method = "cv"
print(Model2)
```

```
## Linear Regression
##
## 55 samples
##  1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 49, 48, 49, 50, 49, 51, ...
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   6.971404    0.880278   6.138437
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

News prediction from the trainning Datasets

To predict the distance from a model we created the dataset PModel.CSV containing data from the ADC1 and ADC2 sensor and sampled every 100mm.

```r
PModel <- read_csv(file = paste0(parentFolder, "/Datasets/PModel.csv")) %>% as.data.frame()
#The head() function is very useful for getting a quick idea of the data in an object, especially if th
head(PModel)
```

```
##    ADC1 ADC2
## 1   627  475
## 2   633  475
## 3   627  486
## 4   633  475
## 5   627  475
## 6  1169  309
```

```r
#Summary of our Dataset
summary(PModel)
```

```
##       ADC1            ADC2
##  Min.   : 627    Min.   :107.0
##  1st Qu.:1035    1st Qu.:136.5
##  Median :1756    Median :214.5
##  Mean   :1889    Mean   :254.8
##  3rd Qu.:2612    3rd Qu.:350.5
```

```
## Max.   :3412    Max.    :486.0
```

```
#create variable for predictions Firts Model
P1 <- predict(Model1,newdata=PModel)
print(P1)
```

```
##        1        2        3        4        5        6        7        8
## 10.40050 10.50802 10.40050 10.50802 10.40050 20.11387 20.49022 20.49022
##        9       10       11       12       13       14       15       16
## 20.49022 20.38269 41.27899 41.38652 40.79511 41.38652 41.38652 60.31147
##       17       18       19       20
## 59.75591 60.20394 59.86344 59.75591
```
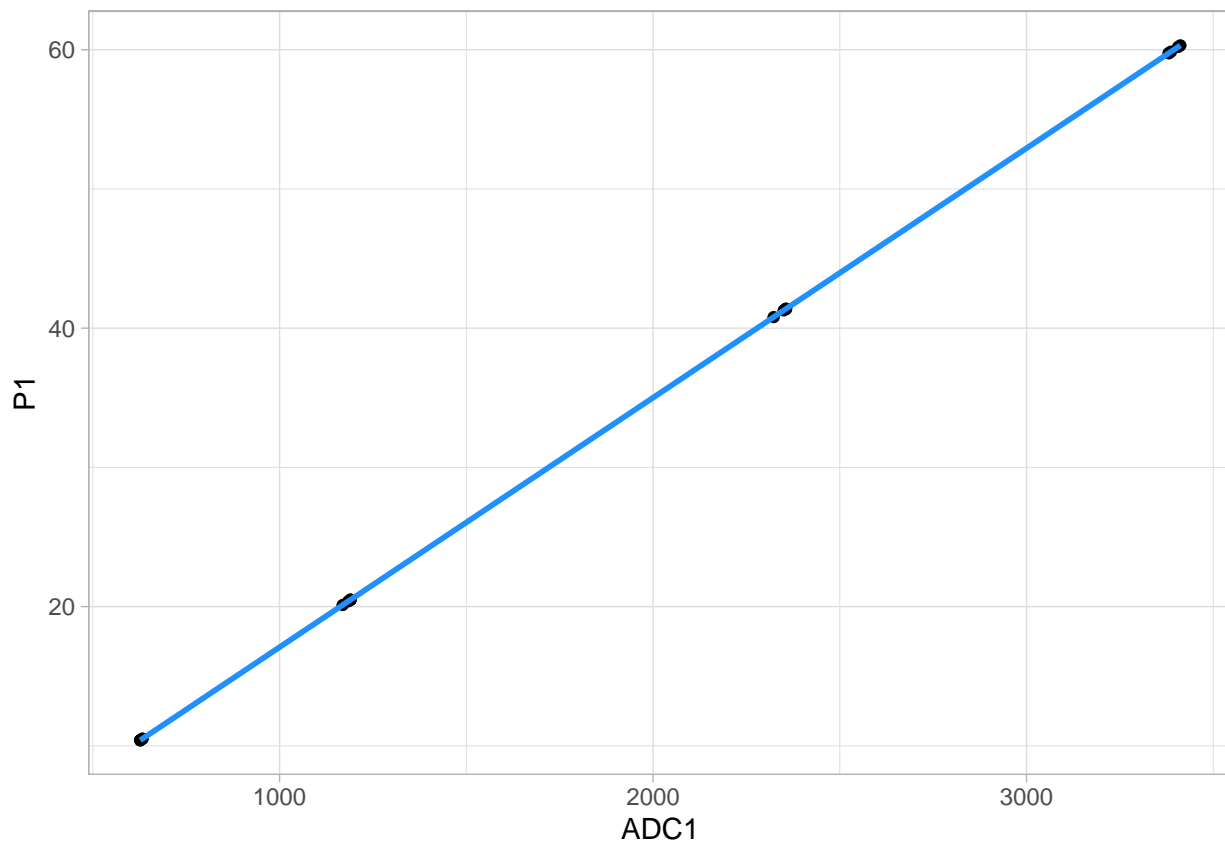
```
#We plot the graph that relates the distance that is the predictor and the value of the ADC1 sensor.
ggplot(PModel, aes(x=ADC1, y=P1)) +
geom_point() +
geom_smooth(method='lm', formula=y~x, se=FALSE, col='dodgerblue1') +
theme_light()
```



```
#Create variable for predictions Second Model
P2 <- predict(Model2,newdata =PModel)
print(P2)
```

```
##        1        2        3        4        5        6        7        8
##  0.115937  0.115937 -1.310742  0.115937  0.115937 21.645816 25.536758 25.666456
##        9       10       11       12       13       14       15       16
## 25.666456 25.017965 42.397506 42.397506 42.138110 42.916298 42.397506 47.844825
##       17       18       19       20
```

9

```
## 47.844825 47.844825 47.326032 47.844825
```

```
#We plot the graph that relates the distance that is the predictor and the value of the ADC2 sensor.
ggplot(PModel, aes(x=ADC2, y=P2)) +
geom_point() +
geom_smooth(method='lm', formula=y~x, se=FALSE, col='dodgerblue1') +
theme_light()
```
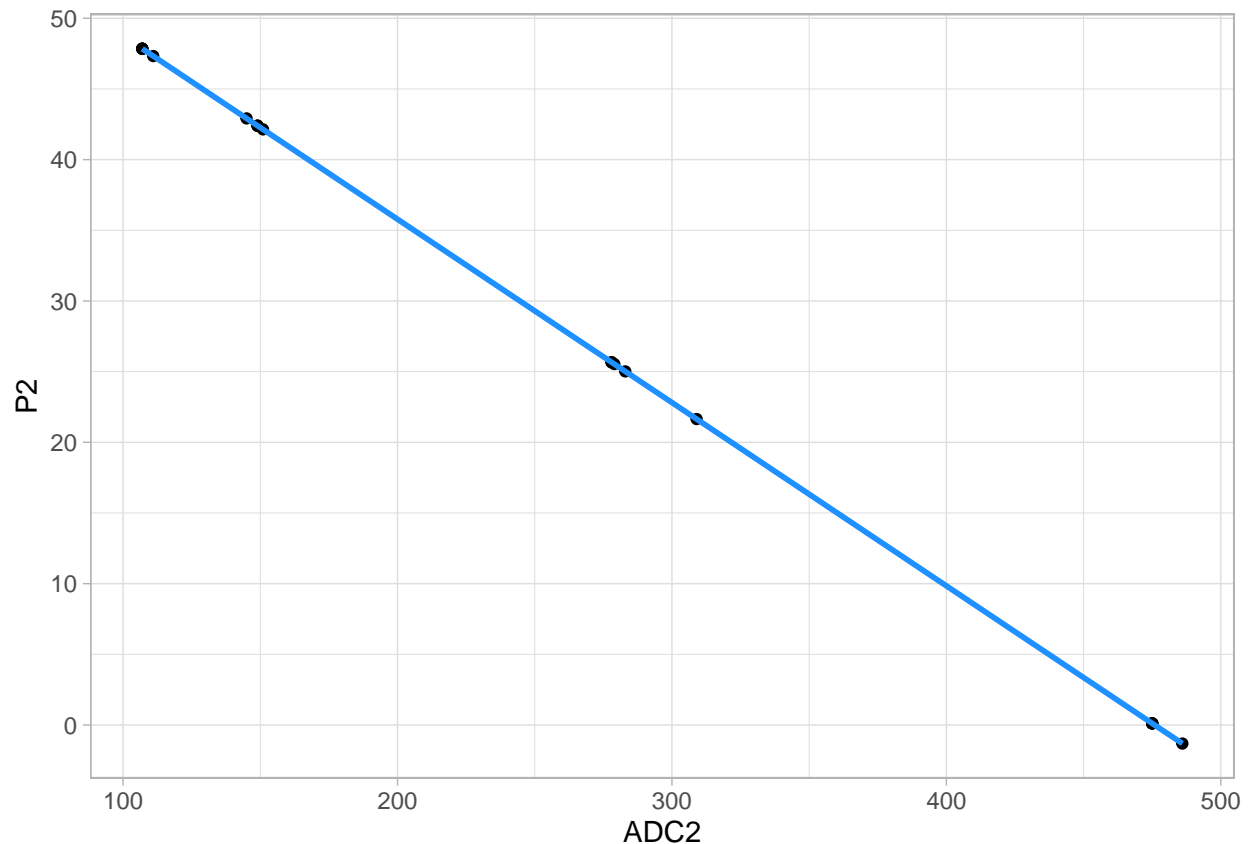


```
#Create variable for predictions model multilinear
P3 <- predict(AllModel,newdata =PModel)
print(P3)
```

```
##         1         2         3         4         5         6         7         8
##  9.913829 10.018530  9.867472 10.018530  9.913829 20.071369 20.564250 20.568465
##         9        10        11        12        13        14        15        16
## 20.568465 20.442692 41.354258 41.458959 40.874676 41.475816 41.458959 60.063298
##        17        18        19        20
## 59.522343 59.958597 59.610187 59.522343
```

```
#We plot the graph that relates the distance that is the predictor between the multi-linear model ADC1+
ggplot(PModel, aes(x=ADC1+ADC2, y=P3)) +
geom_point() +
geom_smooth(method='lm', formula=y~x, se=FALSE, col='dodgerblue1') +
theme_light()
```
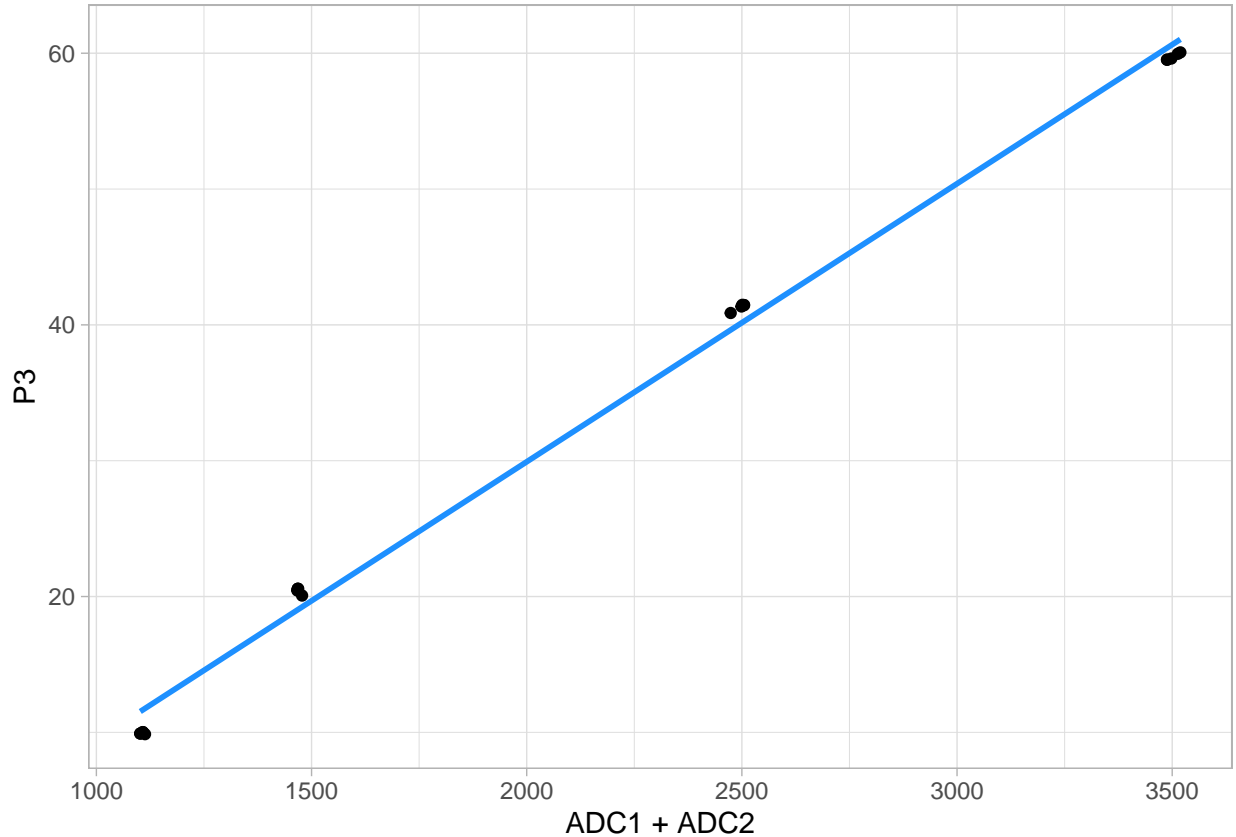
- 1.2.7 Discuss all your codes and results.

- The sensor 1 ADC1 has a linear behavior, this is an ultrasonic sensor that works by mechanical waves, training our model we obtain that this is the best response to use it as a prediction model for our distance variable.

- The sensor 2 ADC2 is a SHARP sensor of analog infrared operation, where its behavior is not linear, another characteristic of this sensor is the closer the object of detection, its variable will increase, it is inversely proportional to the distance.

- We have two Datasets the first one called Sensors1 which contains values for the training for the 3 models, the first model obtained a better response obtaining a RMSE value of 0.4127814 , which is an efficient predictor value to implement models for distance detection,

- The second model obtained a non-optimal response with an RMSE value of 6.9711404, however by performing a better data analysis we can improve the efficiency of our model.

- The third model is a multilinear model where we take into account both ADC1 and ADC2 sensors to make a distance prediction, the combination of these data gives us an RMSE of 0.34101 which tells us that the effectiveness of this combined model is good for predictions.

- After analyzing the graphs and values obtained from our models, Model 1 is the most optimal after training to predict our variable which in this case is distance.

- RMSE is a measure of the average distance between the predicted and actual values, and it is calculated as the square root of the mean of the squared differences between the predicted and actual values.

- MAE is a measure of the average absolute difference between the predicted and actual values, and it is calculated as the mean of the absolute differences between the predicted and actual values.

11

- R-squared (R^2) is a statistical measure that represents the proportion of the variance in the dependent variable,ranges from 0 to 1, where 0 means that the model explains none of the variance and 1 means that the model explains all the variance.

Predictions of a categorical variable

Develop a system (hardware and software) that includes the 2 sensors used in part 1. The system needs to let you capture data that is useful to determine if a wall in front of the system is flat, convex, or concave.