Localización de personas con señales WIFI

UN PROFUNDO ANÁLISIS Y DISEÑO DE UN MODELO DE PREDICCIÓN

Juan Pablo Delzo Melendez
ANALISTA DE DATOS

Contenido

1. Introducción

Un equipo técnico realizo el trabajo de recolectar datos de señales WIFI emitidos por personas que se tenía el conocimiento a priori tanto de sus ubicaciones en coordenadas UTM como de sus posiciones relativas dentro de un conjunto de tres edificios colindantes en el Campus Universitario de la Universidad Jaume I de Castellón - Valencia.

Figura 1: Vista en planta del Campus Universitario de la Universidad Jaume I

El trabajo consistió en la ubicación de 18 individuos, en total todos ellos realizaron 21048 desplazamientos dentro de las infraestructuras con el objetivo de ser localizados en diferentes áreas, donde para cada localización estaban activos 520 aparatos WIFI, todos ellos distribuidos en todo el conjunto de los edificios.

En total se obtuvo un cuadro conformado por 529 columnas y 21048 filas, donde a continuación se detalla el contenido:

- Las primeras 520 columnas corresponden a las señales emitidas por los 520 aparatos WIFI en unidades dbm, donde la menor señal percibida corresponde a 104 dbm (decibelio-milivatio) y la máxima percibida corresponde a 0 dbm. Por convención, las señales no detectadas se colocaron como valor al número 100.
- La columna 521 corresponden los valores de Longitud del individuo.
- La comuna 522 corresponde los valores de Latitud del individuo.
- La columna 523 corresponde al nivel de la planta (0 ,1 ,2,3, 4).
- La columna 524 corresponde al número del edificio (1,2,3)
- La columna 525 corresponde al número del espacio interior en la planta.
- La columna 526 corresponde a la posición relativa en el espacio interior de la planta.
- La columna 527 corresponde al número de usuario
- La comuna 528 corresponde al número identificador de móvil.
- La columna 529 corresponde a la unidad de tiempo donde el usuario realizo la acción de ser detectado.

Por último, los datos están divididos en Training Data y Test Data, que contienen 19.937 y 1.111 filas respectivamente.

La pregunta inicial es si a partir de estos datos recolectados se puede crear un modelo que pueda predecir la ubicación de las personas.

2. Objetivos

- El principal objetivo es predecir independientemente la Latitud, Longitud y el Número de Planta del Usuario (Floor) con los primeros 520 atributos que proporcionan los datos de señales WIFI.
- Se realizará una inspección general de los datos donde se analizará alguna posible transformación.
- Teniendo en cuenta las variables del error de predicción y coste computacional, se diseñará el camino que optimice los variables en conjunto.
- Finalmente, se realizará una serie de análisis de los errores del modelo predictor.

3. Inspección preliminar de las señales WIFI

El número total de las señales WIFI en el Training Data viene a ser 10’367.240, mientras que en Test Data la cantidad es 577.720.

Ahora observemos las características principales que ofrecen las distribuciones de frecuencias de las señales detectadas en ambos conjuntos:

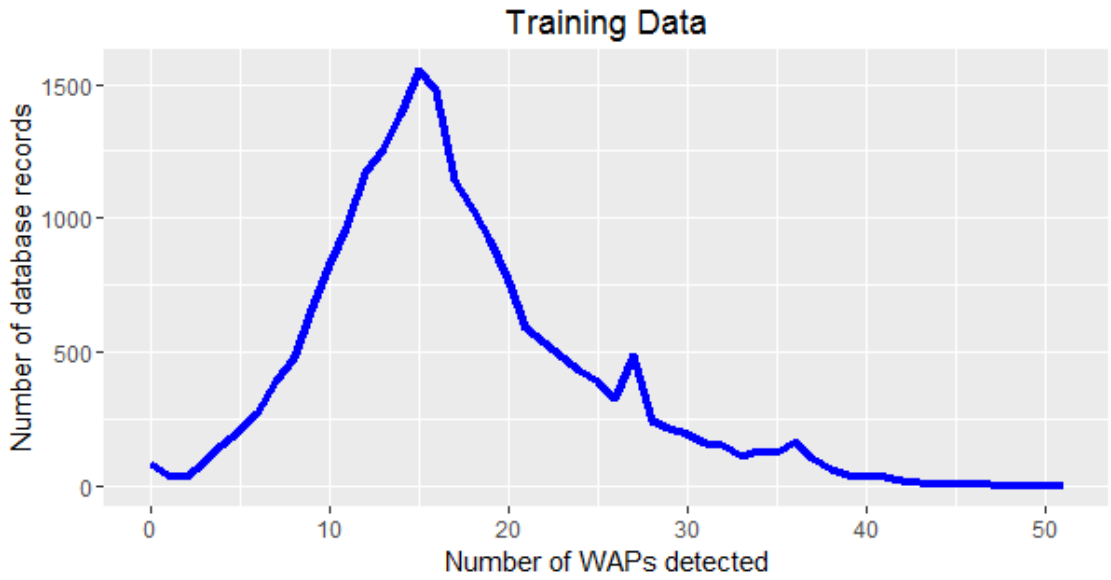

Gráfico 1: Distribución de frecuencias del número de señales detectadas del conjunto de 520 aparatos WIFI por cada episodio de prueba en el Training Data

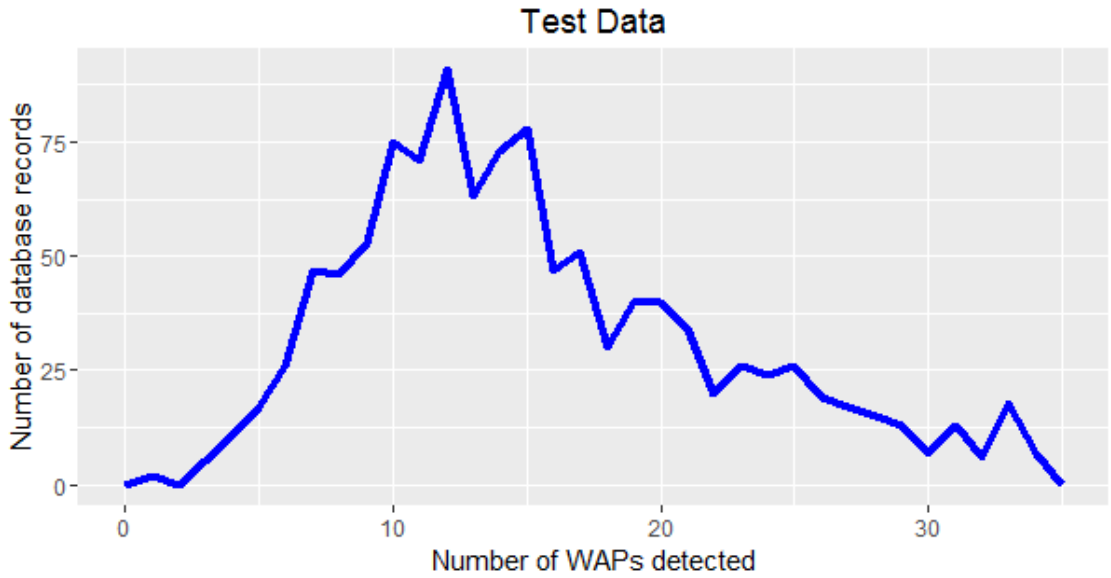

Gráfico 2: Distribución de frecuencias del número de señales detectadas del conjunto de 520 aparatos WIFI por cada episodio de prueba en el Test Data

Se pone en evidencia que como máximo el 10% del total de los aparatos detectan señales; por tanto, ello amerita una mejor observación en el global de los datos.

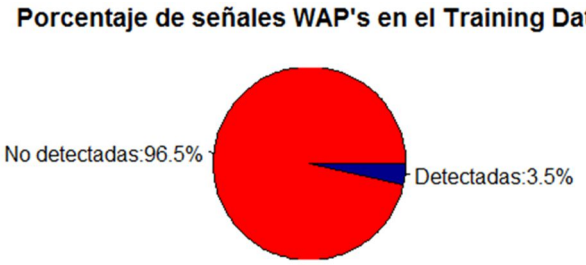

Figura 2: Porcentaje de señales en el Training Data

Existen 10'008.477 señales no detectadas y recordando que las señales no detectadas por convención tienen como valor 100, entonces ellos pueden generan un sobrecoste computacional debido al volumen de datos a trabajar.

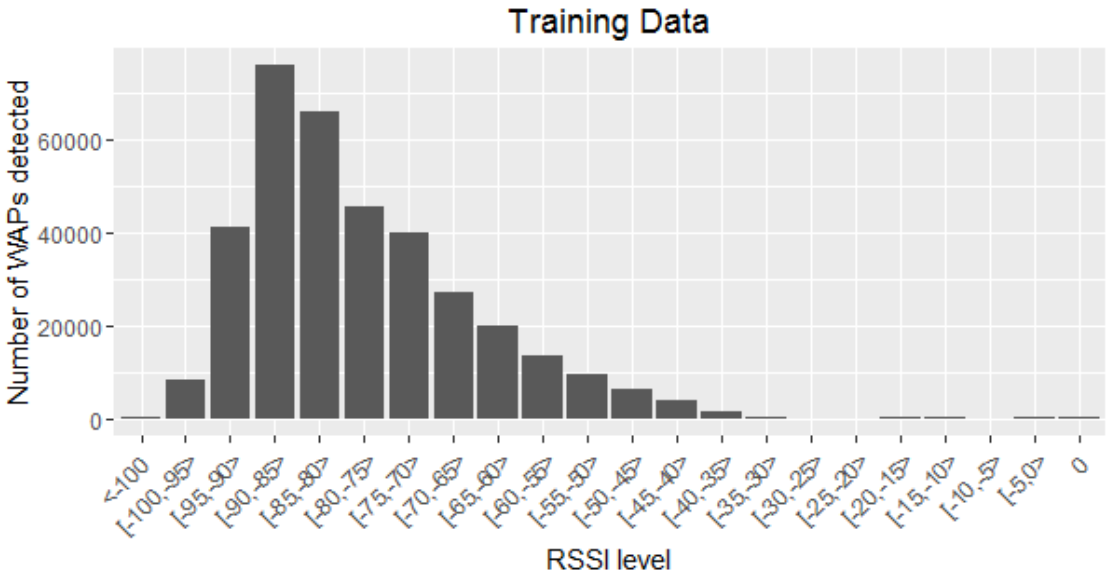

Gráfico 3: Distribución de frecuencias del número de veces del nivel de intensidad de la señal detectada del Training Data

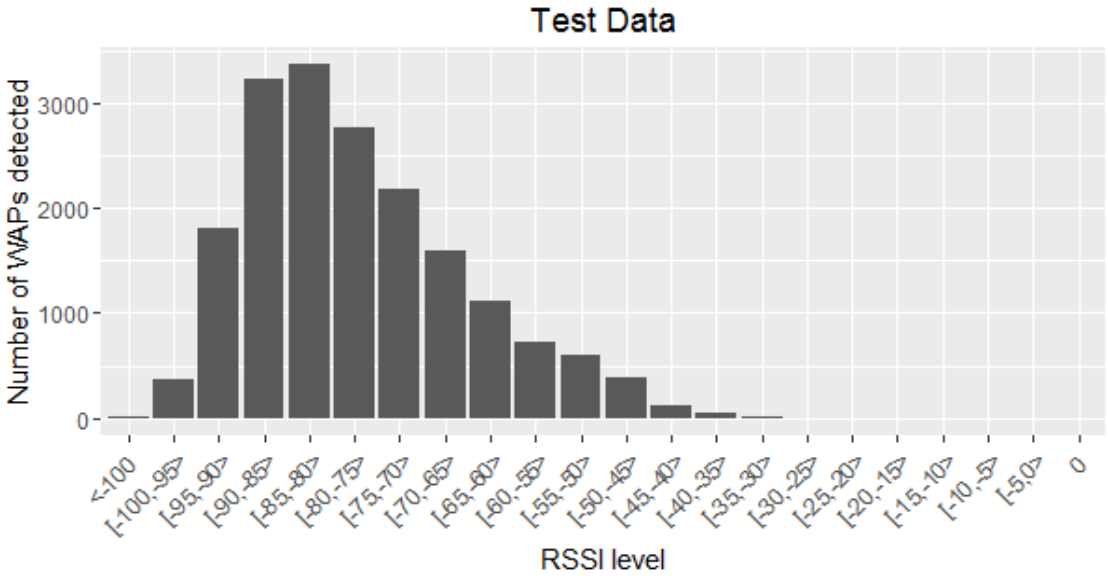

Gráfico 4: Distribución de frecuencias del número de veces del nivel de intensidad de la señal detectada del Test Data

En ambos datos, se observa que el mayor número de señales detectadas corresponden a intensidades entre -90 a -80 dbm, mientras que la intensidad aumenta de magnitud el número de señales detectadas disminuye a un ritmo cuya función se asemeja a la función $e^{-x} - 1$.

En el Training Data se han detectado 137 valores de 0 dbm, que representa el 0.001% del total de señales detectadas. Mientras que en el Test Data no existe ningún valor igual a 0 dbm.

4. Pre proceso

Debido al gran volumen de datos, se está considerando el método de **K-Nearest Neighbors (KNN)** debido al menor coste computacional sin sacrificar significativamente la precisión de los resultados. Para aplicar los métodos KKN se tiene que cumplir los dos siguientes pasos:

4.1 Transformación de los valores de las señales no detectadas

A pesar de que existe un sub conjunto de 137 valores con señal 0 en el Training Data que son una inmensa menoría y aprovechando tanto la no existencia de valores 0 en el Test Data y la tendencia de ambas graficas ya comentadas en el anterior apartado.

Entonces, en la mira de reducir el coste computacional que se puede expresar en unidad de tiempo, **se transformará los valores 100 a 0**, porque contribuirá a reducir el coste del cálculo de infinitas distancias euclidianas en el método KKN porque el resto de los valores de las coordenadas tienen valores negativos, sabiendo que dicha transformación no creara errores significativos evaluando el modelo en Test Data.

4.2 Búsqueda de K óptimo

Teniendo en cuenta las variables: el error de precisión y el coste computacional; y ante la búsqueda del K*, se crea la pregunta ¿qué camino a seguir tal que optimice dichas variables?

La primera opción que salta a la vista es aplicar Cross-validation, sabemos que el error medio es bajo, pero tendría un alto coste computacional de acuerdo a nuestros valores de entrada; por tanto, el método queda descartado.

La segunda opción sería aplicar una sola prueba de evaluación con el Training Data entero, este método tiene un bajo coste computacional pero un alto error; por tanto, este método también queda descartado.

Entonces, para crear un método que no genere un alto coste computacional con un error aceptable sería aplicando **un Hold Out de tres o cuatro diferentes formar** que particione el Training Data en datos de entrenamiento llamado Train y en datos de prueba llamado Test.

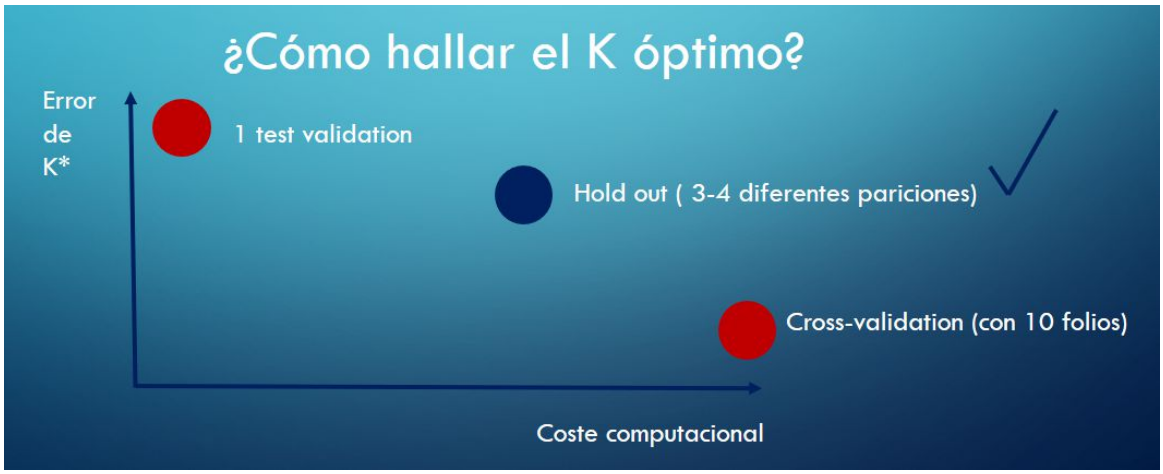

Figura 3: Croquis resumen de la elección del método de validación para determinar el K optimo

5. Algoritmo

Se diseñaron dos bucles, uno para determinar el modo de partición de los valores iniciales y otro bucle para el valor K.

A continuación, se describe las principales características del diseño del algoritmo.

5.1 Partición del Training Data

En total se tiene 529 atributos con 21048 instancias, donde se dividió el 95% de las instancias llamándose Training Data y el 5% restante se denominó Test Data.

Las proporciones de los valores de entrada del Training Data como el Test Data tienen la relación de 95/5. Por tanto, se está estableciendo que el volumen de datos del Train y del Test tengan igual proporción respecto a los datos de partida.

5.2 Modos de partición del Training Data

En la estimación de Longitud y Latitud, la partición del Training Data será tres diferentes modos. Mientras para la estimación del número de piso (Floor), la partición del Training Data será de cuatro modos diferentes.

Los modos de partición de los datos serán utilizando la herramienta set.seed del siguiente modo para cada atributo objetivo:

Atributos:	Nº set.seed():
Latitud, Longitud	1,123,10000
Floor	1,123,10000,1234123

Cuadro 1: Nro. de set.seed aplicado para la partición de instancias del Training Data

5.3 Valores de K

Con el fin de intentar reducir más el coste computacional, se utilizaron los primeros 37 valores impares.

5.4 KKNN

Se aplicó el método KKNN (https://cran.r-project.org/web/packages/kknn/kknn.pdf , página 5) del paquete Cran, donde la principal ventaja es que tu manejas que valor de K que quieres que ejecute el método y arroja directamente la predicción en la columna "fitted.values" de su matriz de resultados.

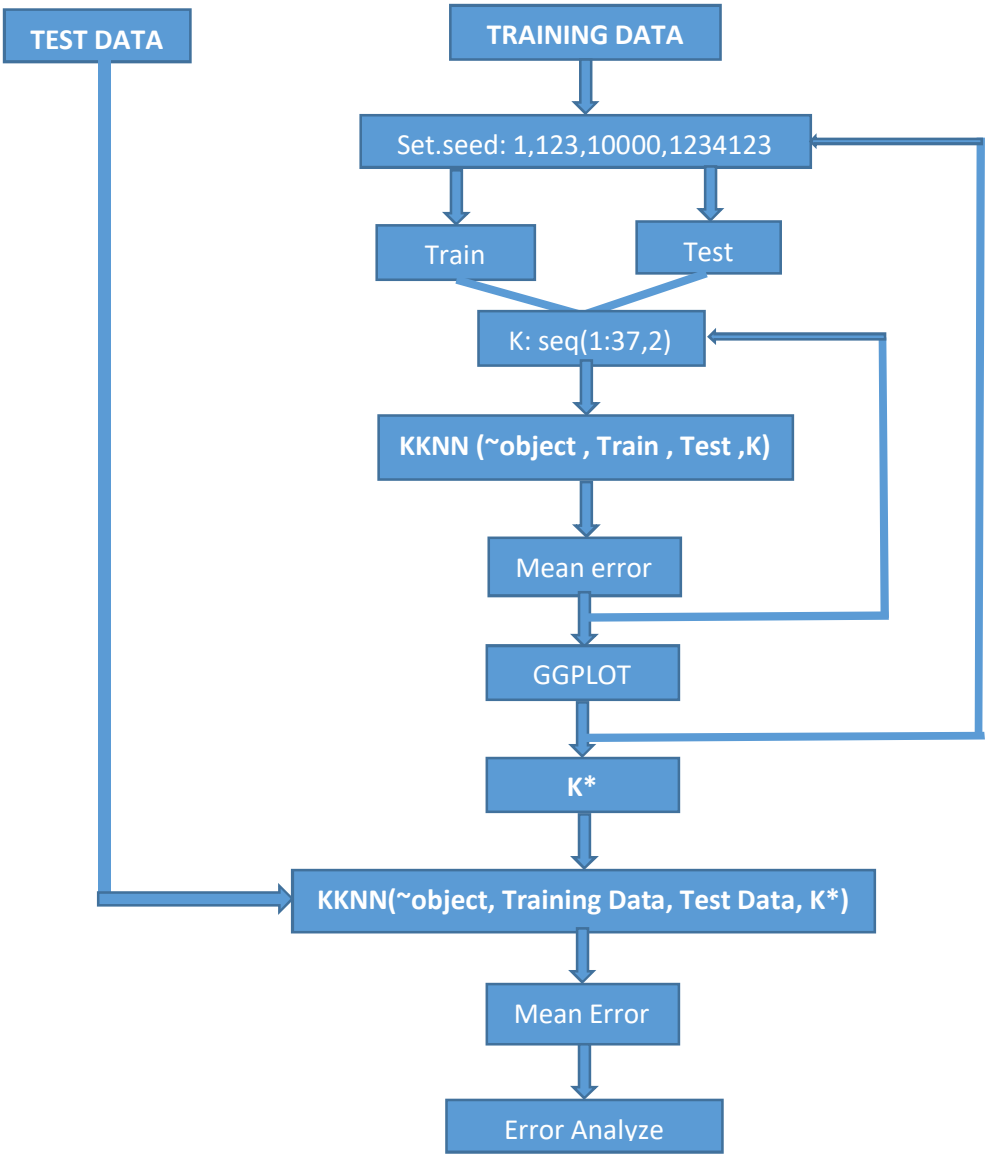

Figura 4: Diagrama de flujo de la programación para aplicar el método KKN

6. LONGITUD

Para iniciar, se escogieron las particiones SET.SEED del R-Studio con valores 1,123,10000.

6.1 SET.SEED (1)

```
> head(Resultadosetseed1)
  NºNeighbour      Error
1           1 0.03612225
2           3 0.02937524
3           5 0.03189123
4           7 0.03406363
5           9 0.03607115
6          11 0.03791819
```

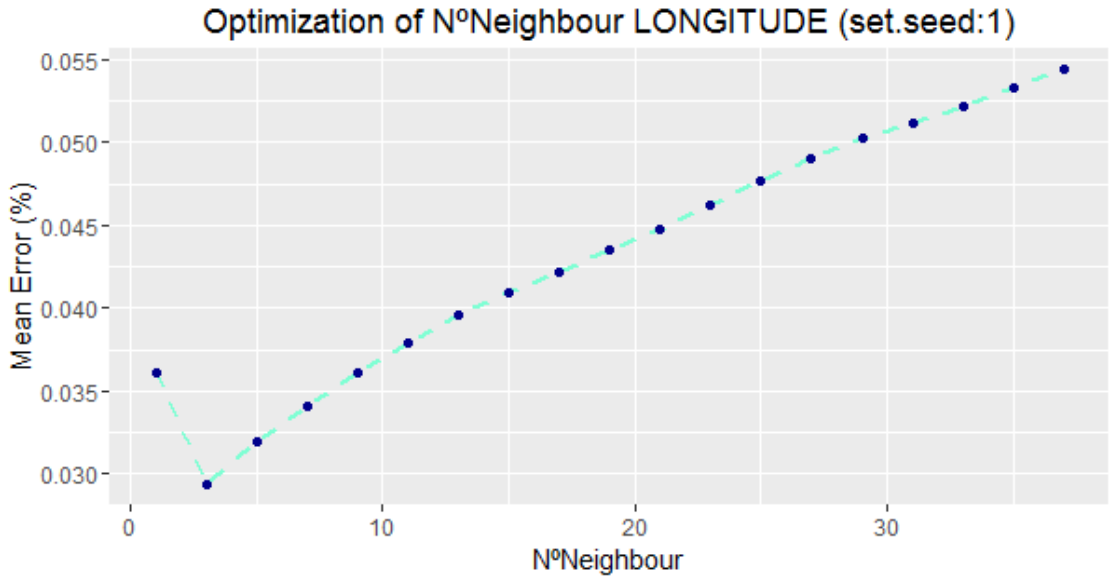

Gráfico 5: Error medio del modelo de predicción de KKN en el Test en función de NºNeighbour para un set.seed(1)

6.2 SET.SEED (123)

```
> head(Resultadosetseed123)
  NºNeighbour      Error
1           1 0.04341497
2           3 0.03771788
3           5 0.03855962
4           7 0.04132365
5           9 0.04333881
6          11 0.04494562
```

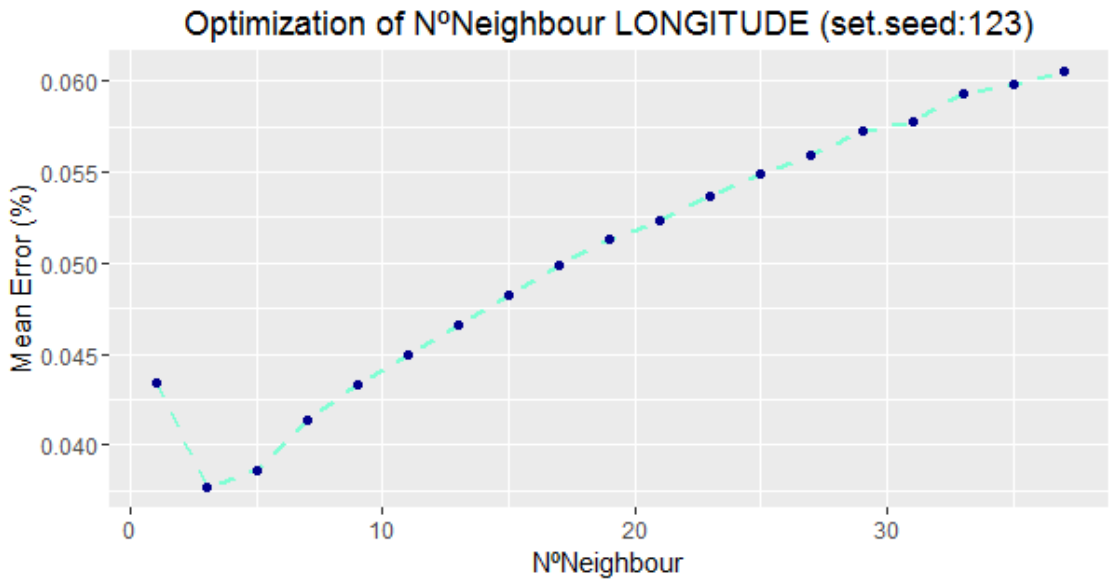

Gráfico 6: Error medio del modelo de predicción de KKN en el Test en función de NºNeighbour para un set.seed(123)

6.3 SET.SEED (10000)

```
> head(Resultadosetseed10000)
  NºNeighbour      Error
1           1 0.03188106
2           3 0.03017730
3           5 0.03108581
4           7 0.03250319
5           9 0.03455786
6          11 0.03666030
```

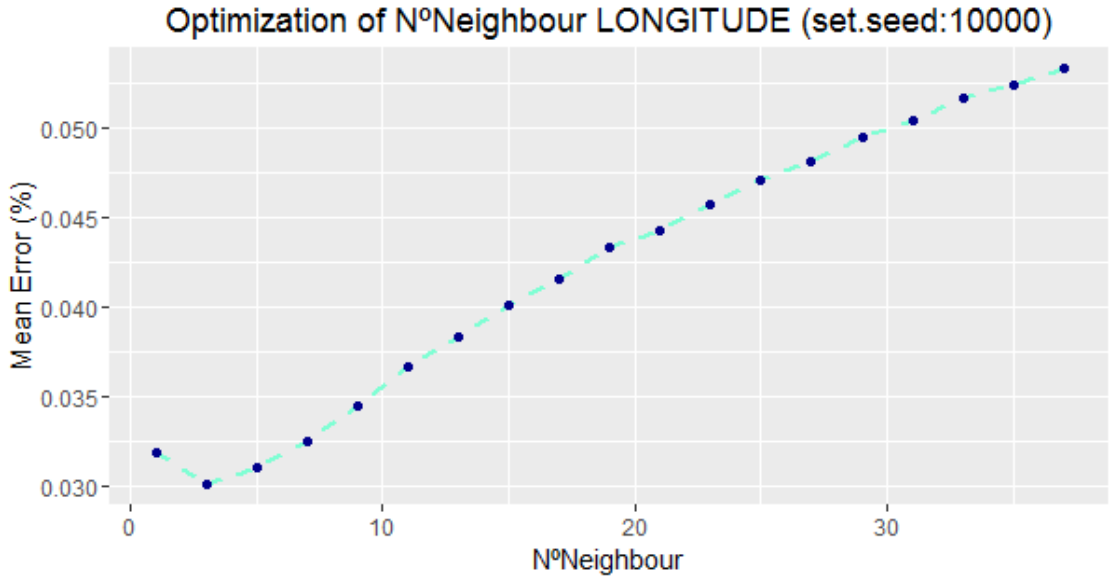

Gráfico 7: Error medio del modelo de predicción de KKN en el Test en función de NºNeighbour para un set.seed(10000)

De los tres gráficos creados, todos ellos coinciden que el K optimo viene a ser el valor de 3; por tanto, estamos considerando que aquel valor será el K optimo en el modelo creado en el Training Data.

6.4 Evaluación de la calidad del modelo y análisis de los errores

Se entrenó el modelo en el Training Data con K igual 3, luego se evaluó su calidad en el Test Data. Los resultados son los siguientes:

```
> mean_percent_error
[1] 0.1226653

> head(tabla_resultado)
        Actual Predicted Absolute percent error
[1,] -7515.917 -7455.046            0.809894371
[2,] -7383.867 -7384.246            0.005132846
[3,] -7374.302 -7370.011            0.058193710
[4,] -7365.825 -7372.803            0.094735044
[5,] -7641.499 -7648.411            0.090455950
[6,] -7338.807 -7322.213            0.226121431


> summary(error_resultado)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-76.4200  -4.6930  -0.1719   0.3705   4.1990 180.2000
```

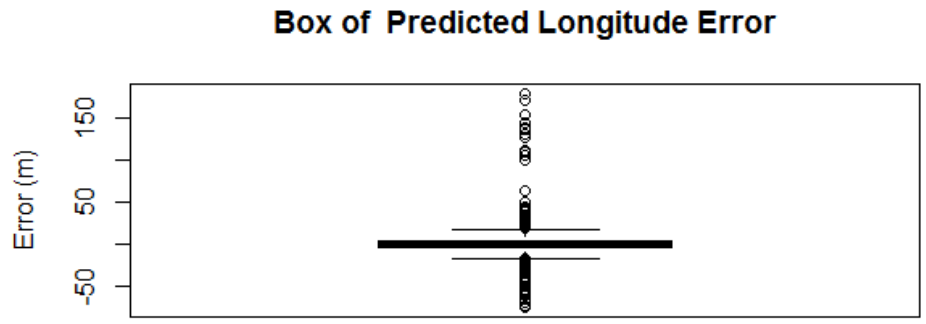

Gráfico 8: Boxplot de los errores calculados en la predicción de la Longitud

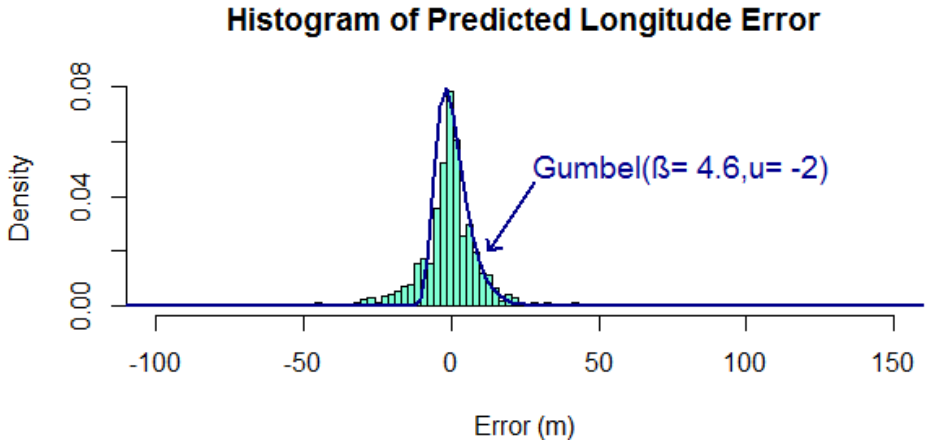

Grafico 9: Histograma de los errores con la función densidad de ajuste

7. LATITUD

Análogamente en la predicción de la Longitud, se escogieron las particiones SET.SEED del R-Studio con valores 1,123,10000.

7.1 SET.SEED(1)

```
> head(Resultadosetseed1)
  NºNeighbour        Error
1           1 3.663807e-07
2           3 3.524738e-07
3           5 3.553937e-07
4           7 3.710692e-07
5           9 4.007663e-07
6          11 4.082505e-07
```

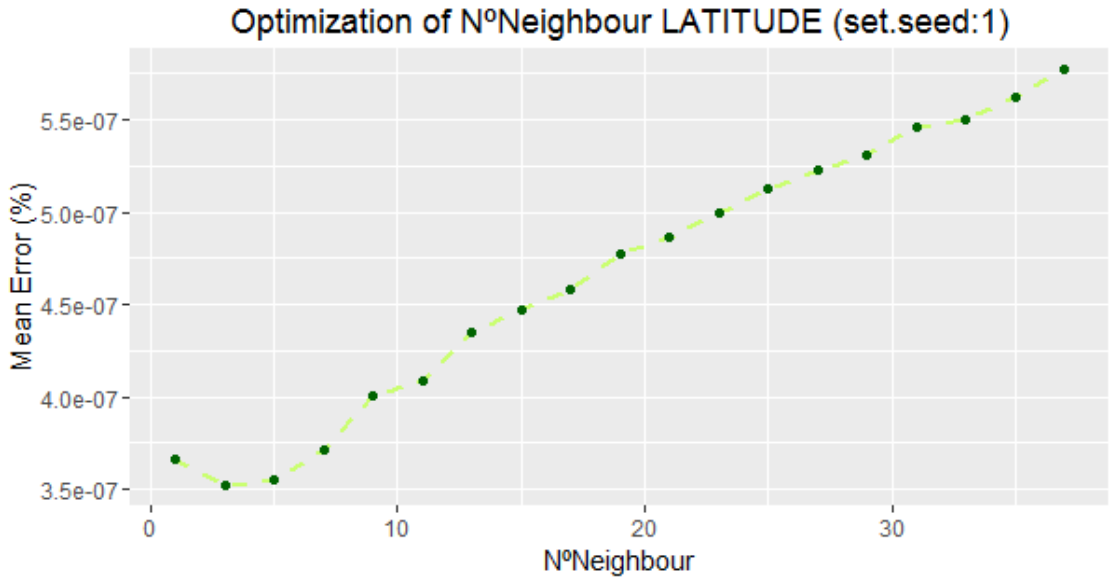

Gráfico 10: Error medio del modelo de predicción de KKN en el Test en función de NºNeighbour para un set.seed(1)

7.2 SET.SEED(123)

```
> head(Resultadosetseed123)
  NºNeighbour        Error
1           1 3.991966e-07
2           3 3.825191e-07
3           5 4.085488e-07
4           7 4.312187e-07
5           9 4.420558e-07
6          11 4.665637e-07
```

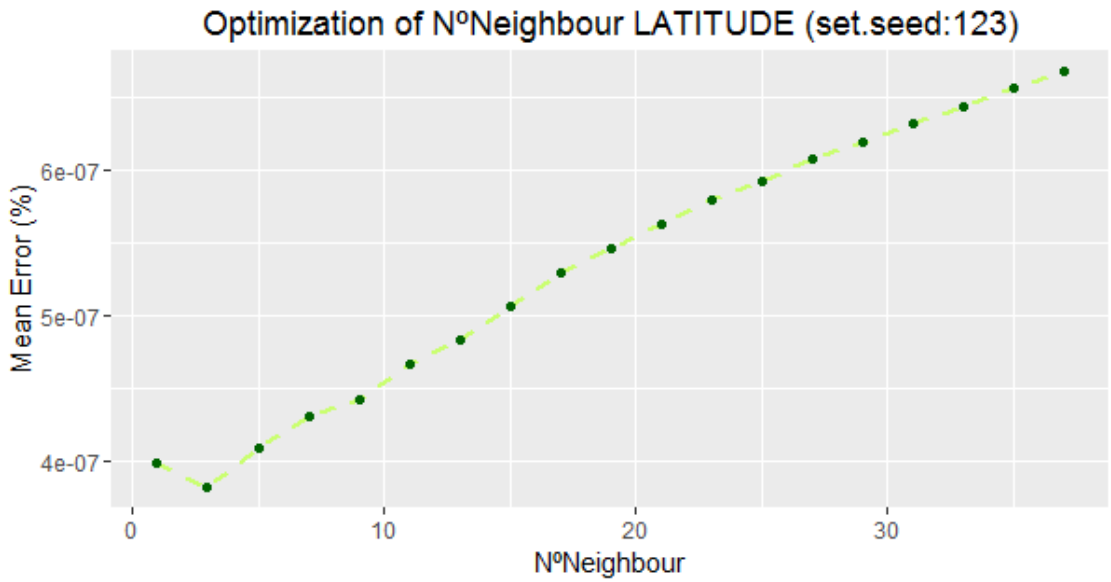

Gráfico 11: Error medio del modelo de predicción de KKN en el Test en función de NºNeighbour para un set.seed(123)

7.3 SET.SEED(10000)

```
> head(Resultadosetseed10000)
  NºNeighbour        Error
1           1 3.579986e-07
2           3 3.382521e-07
3           5 3.670773e-07
4           7 3.876481e-07
5           9 4.073633e-07
6          11 4.341327e-07
```

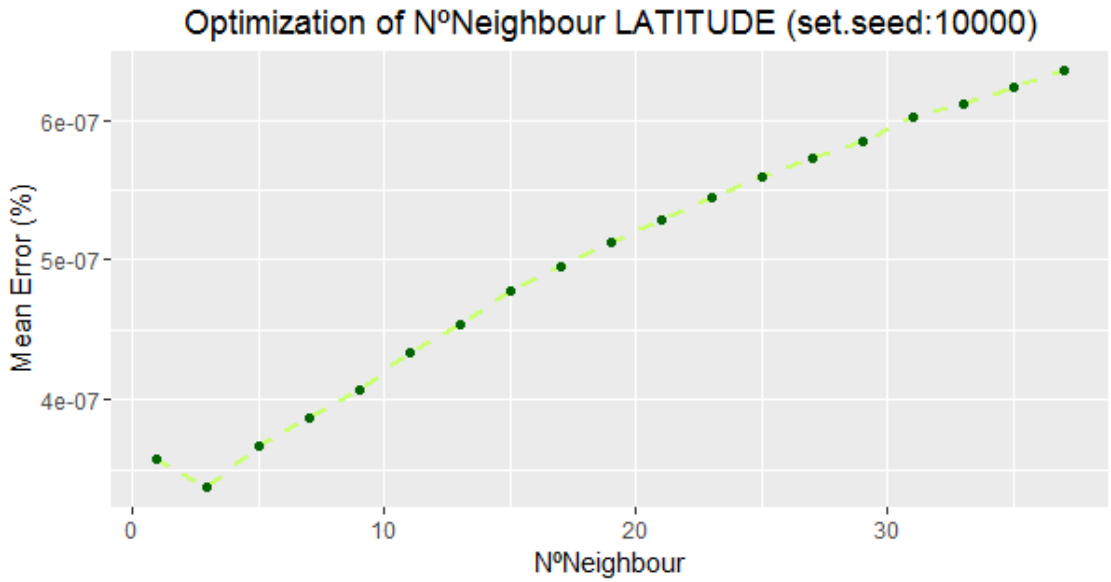

Gráfico 12: Error medio del modelo de predicción de KKN en el Test en función de NºNeighbour para un set.seed(10000)

De los tres gráficos creados, todos ellos coinciden que el K optimo viene a ser el valor de 3; por tanto, estamos considerando que aquel valor será el K optimo en el modelo creado en el Training Data.

7.4 Evaluación de la calidad del modelo y análisis de los errores

Se entrenó el modelo en el Training Data con K igual 3, luego se evaluó su calidad en el Test Data. Los resultados son los siguientes:

```
> mean_percent_error
[1] 0.000182544

> head(tabla_resultado)
      Actual Predicted Absolute percent error
[1,] 4864890   4864887           5.789101e-05
[2,] 4864840   4864841           3.137351e-05
[3,] 4864847   4864848           2.507330e-05
[4,] 4864843   4864849           1.330679e-04
[5,] 4864922   4864926           6.911177e-05
[6,] 4864825   4864820           1.222478e-04

> summary(error_resultado)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-104.8000   -7.7950   -0.8774   -1.9610    4.3380  120.0000
```

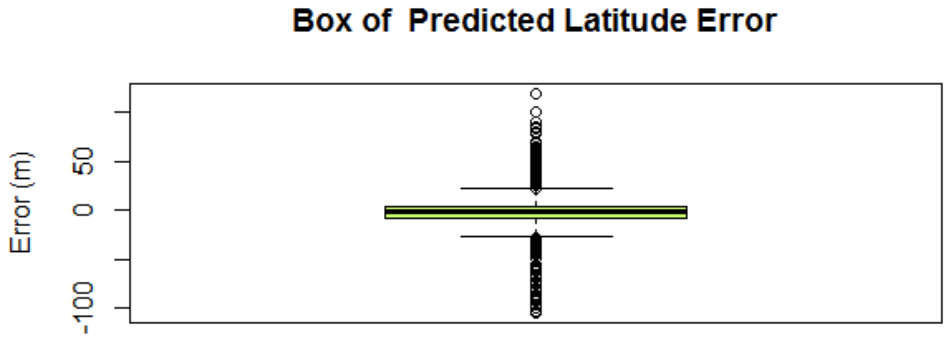

Gráfico 13: Boxplot de los errores calculados en la predicción de la Latitud

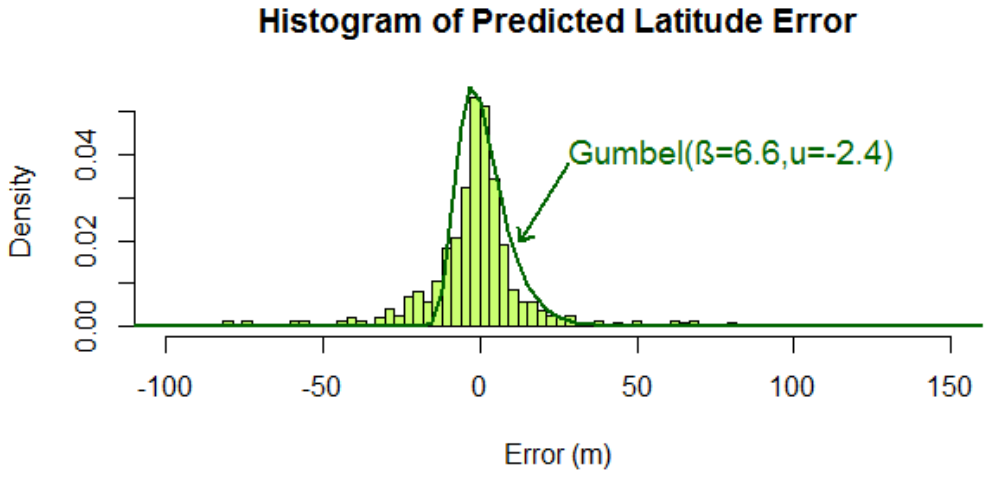

Grafico 14: Histograma de los errores con la función densidad de ajuste

8. Análisis del error en las distancias en planta

Una vez que se obtuvo los valores de las coordenadas de Longitud y Latitud en UTM, se realizó el cálculo de la distancia que existe entre el punto real y del punto predecido.

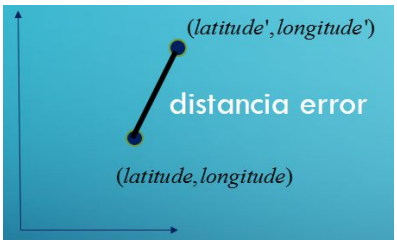

Figura 5: Croquis del error de la distancia entre el punto real y el punto predecido.

Los resultados son los siguientes:

```
> head(distanciaerror)
[1] 66.569236 16.549680  4.292170  7.060765  9.350010 17.529586

> summary(distanciaerror)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
  0.1425   5.4150   9.9950  17.4000  19.8700 201.4000

> mean(distanciaerror)
[1] 17.3958
```

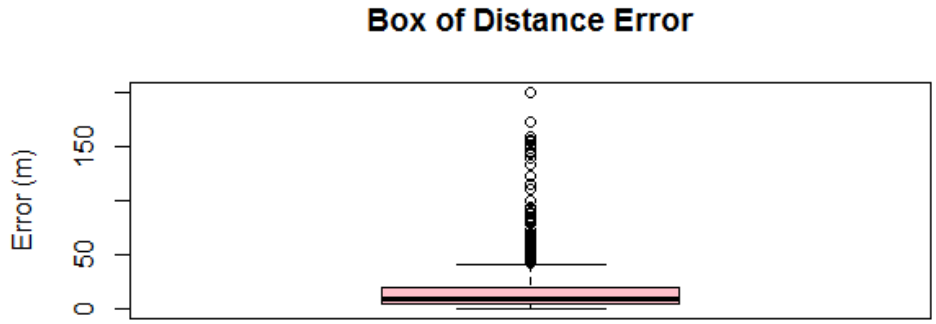

Gráfico 15: Boxplot de los errores en la distancia en planta

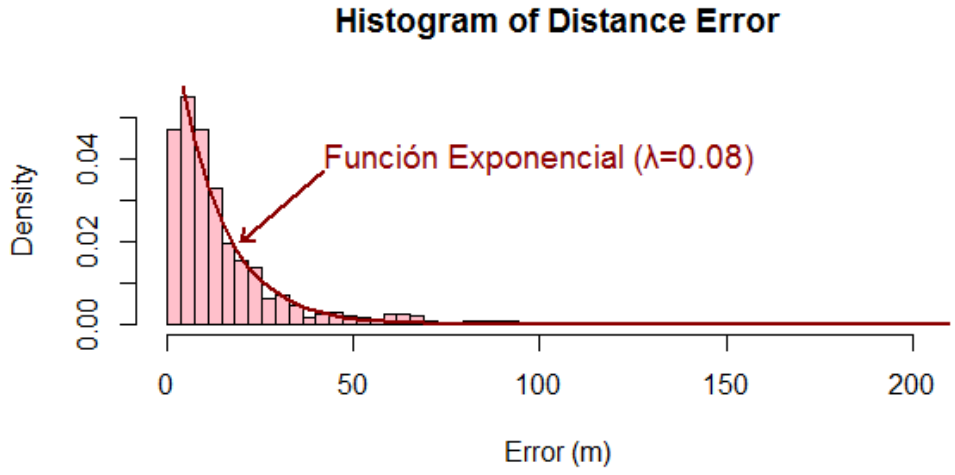

Grafico 16: Histograma del error en la distancia en planta con la función densidad de ajuste

9. FLOOR

Se escogieron las particiones SET.SEED del R-Studio con valores 1,123,10000,1234123.

9.1 SET.SEED(1)

```
> head(Resultadosetseed1)
  NºNeighbour  Accuracy
1           1 0.9849246
2           3 0.9849246
3           5 0.9869347
4           7 0.9919598
5           9 0.9909548
6          11 0.9869347
```

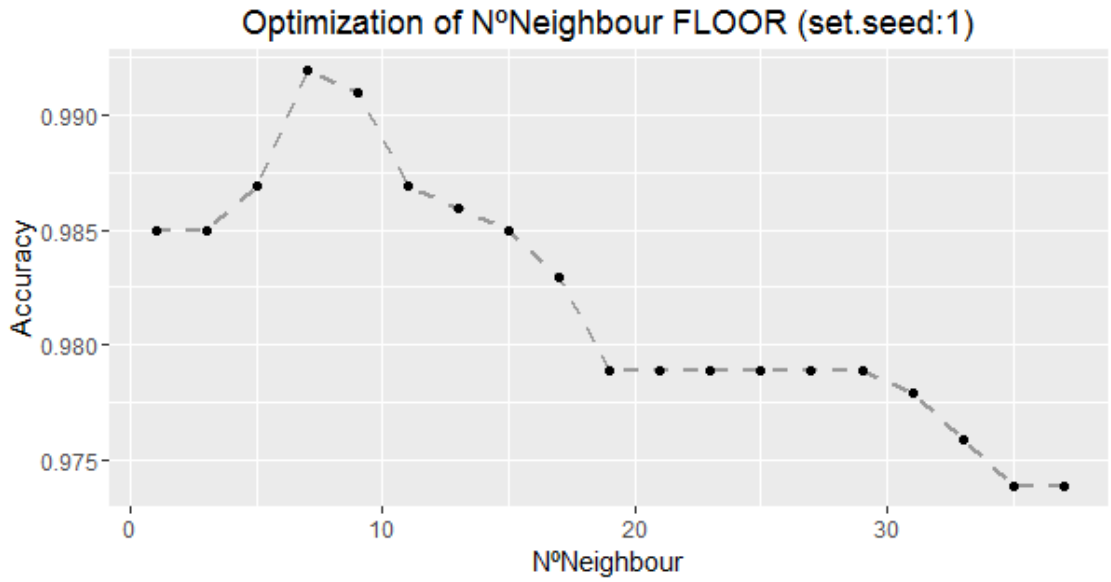

Gráfico 18: Error medio del modelo de predicción de KKN en el Test en función de NºNeighbour para un set.seed(1)

9.2 SET.SEED(123)

```
> head(Resultadosetseed123)
  NºNeighbour  Accuracy
1           1 0.9899497
2           3 0.9899497
3           5 0.9919598
4           7 0.9929648
5           9 0.9929648
6          11 0.9939698
```

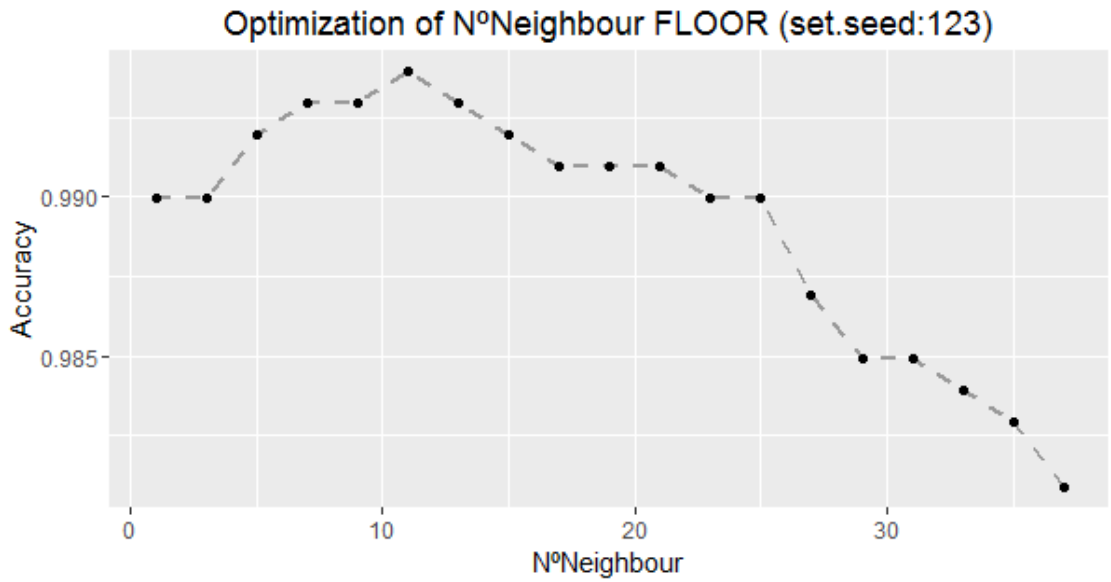

Gráfico 19: Error medio del modelo de predicción de KKN en el Test en función de NºNeighbour para un set.seed(123)

9.3 SET.SEED(10000)

```
> head(Resultadosetseed10000)
  NºNeighbour  Accuracy
1           1 0.9849246
2           3 0.9839196
3           5 0.9839196
4           7 0.9849246
5           9 0.9839196
6          11 0.9839196
```

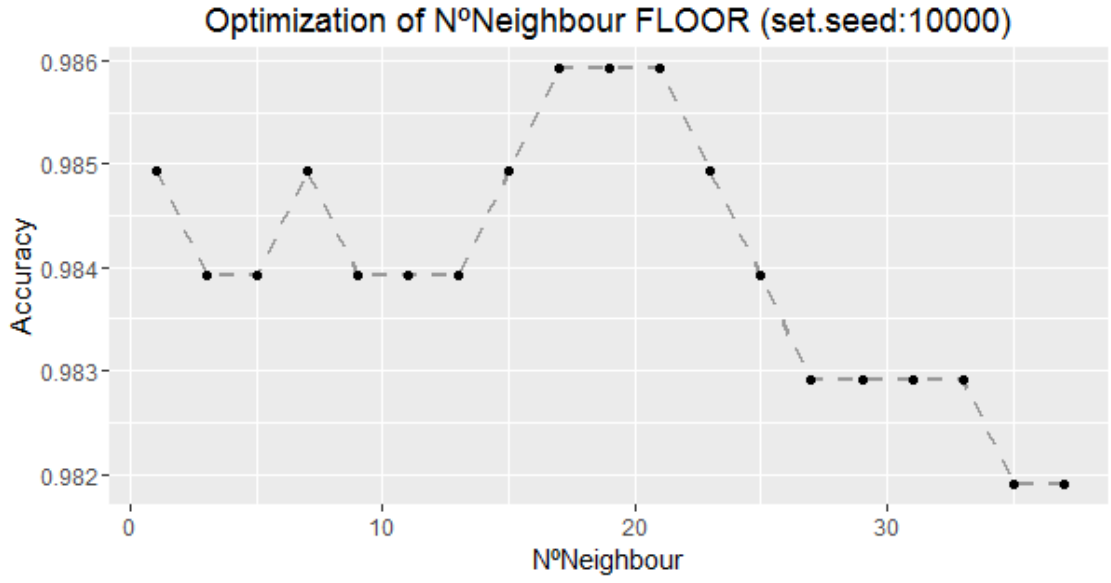

Gráfico 20: Error medio del modelo de predicción de KKN en el Test en función de NºNeighbour para un set.seed(10000)

9.4 SET.SEED(1234123)

```
> head(Resultadosetseed1234123)
  NºNeighbour  Accuracy
1           1 0.9819095
2           3 0.9839196
3           5 0.9829146
4           7 0.9849246
5           9 0.9849246
6          11 0.9839196
```

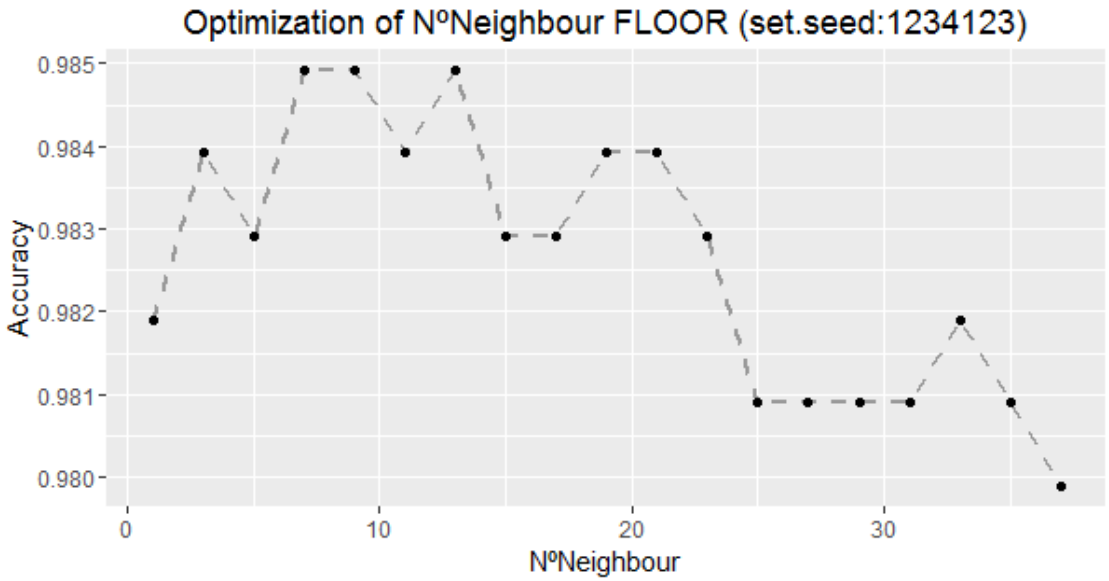

Gráfico 21: Error medio del modelo de predicción de KKN en el Test en función de NºNeighbour para un set.seed(1234123)

9.5 Criterio de decisión: Promedio de las precisiones medias

En vista de que no existe coincidencias del K optimo entre los cuatro gráficos resultantes, se procedió a promediar las precisiones medias. A continuación, se muestra sus resultados:

```
> head(MeanAccuracy)
  NºNeighbour  Accuracy
1           1 0.9854271
2           3 0.9856784
3           5 0.9864322
4           7 0.9886935
5           9 0.9881910
6          11 0.9871859
```

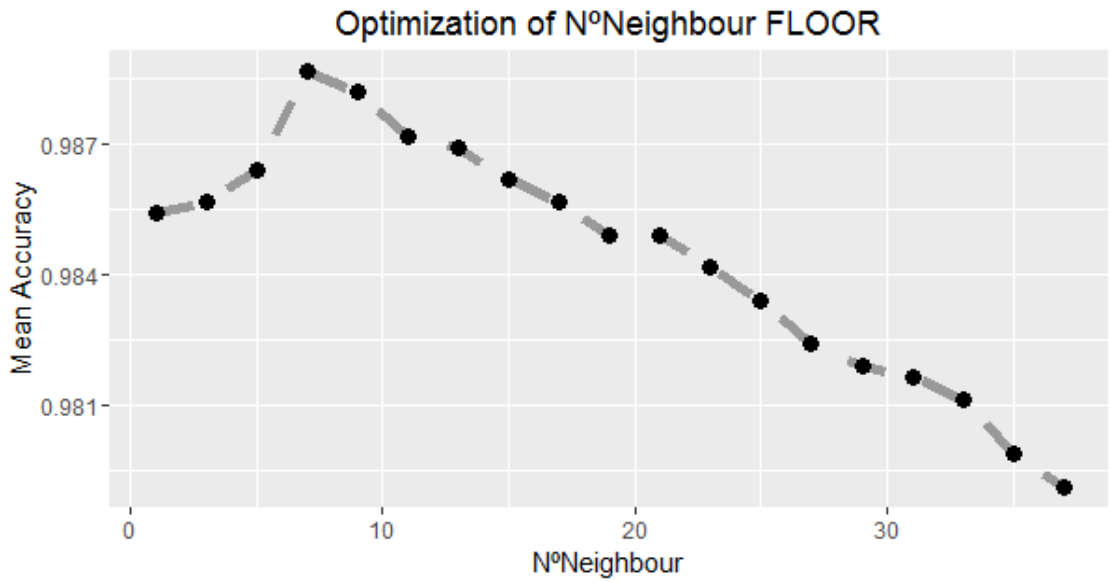

Gráfico 22: Error promedio de la precisión media para cada valor de NºNeighbour

Donde se observa en el valor de K=7 ofrece una mayor precisión media.

9.6 Evaluación de la calidad del modelo y análisis de precisión

Entrenando el método de KKN con el Training Data para un valor de K=7, luego prediciendo los valores de FLOOR en el Test Data, donde finalmente se calculó la matriz de confusión, el valor de precisión, el número de Kappa y el gráfico ROC, los resultados fueron lo siguiente:

```
> head(tabla_resultado)
  Predicted Actual
1         3      1
2         4      4
3         4      4
4         4      4
5         2      2
6         2      2
```

```
> table(tabla_resultado)
         Actual
Predicted   0   1   2   3   4
        0 114  43   6   3   1
        1   8 356  10   2   0
        2   8  55 214  12   0
        3   2   7  75 153  17
        4   0   1   1   2  21
```

```
> ROC
         FPR       TPR
1 0.05319149 0.6826347
2 0.25951557 0.9468085
3 0.39763780 0.7404844
4 0.16000000 0.6023622
```

```
> Accuracy
[1] 0.7722772
> Kappa
[1] 0.6899055
```

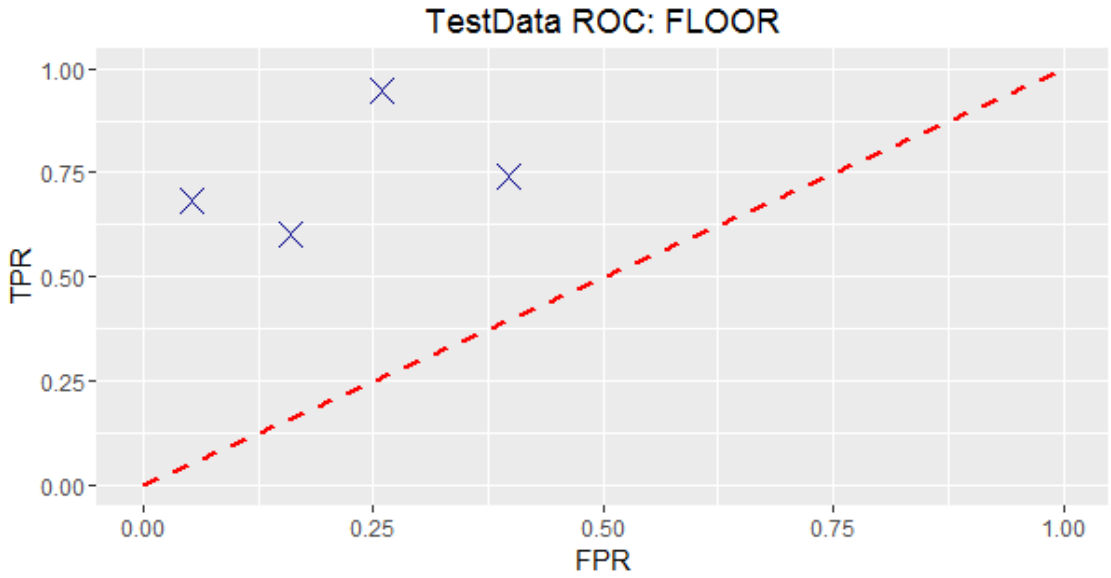

Gráfico 23: Gráfico ROC en la predicción del número de piso

Se concluye que el modelo de predicción esta entre regular a bueno; por tanto, es un modelo que se está aceptando.

10. ¿Cómo se puede mejorar?

- El modelo desarrollado tiene virtudes y defectos. La mejora se puede centrar en continuar con la filosofía de aplicar el método KKN pero con una transformación de los valores WAP's con tal de no generar posibles grandes errores de acuerdo a la configuración de otros posibles Tests Data. La función de transformación propuesta es la siguiente:

$$X' = \begin{cases} \dfrac{x - X_{\min}}{X_{\max-\{100\}} - X_{\min}} + 1 & Si: x < 0 \\ 0 & Si: x = 100 \end{cases}$$

  El intervalo [1,2] del rango de la función corresponden a las señales detectadas, mientras que el valor 0 del rango corresponden a las señales no detectadas.
- En los cuadros de resultados de la predicción de Longitud y Latitud, se pueden añadir gráficos que ayuden a visualizar la precisión de los modelos evaluados.

11. ANEJOS

A continuación, se ha extraído los principales pasos en las programaciones hechas en el programa R-Studio.

11.1 Transformación de los datos iniciales

```
#Descargando los datos del Training Data y Test Data desde la internet:
trainingData <- read.csv("trainingData.csv", header=TRUE, sep=",")
testData <- read.csv("validationData.csv", header=TRUE, sep=",")
```

```
#Función para transformar valores de 100 a 0:
editarvalores100 <- function(input) {
            output <- matrix (rep(0,nrow(input)*ncol(input)),nrow(input),ncol(input))
             for (i in 1: nrow(output)) {
              for (j in 1: ncol(output)) {
                                        if ( input[i,j] != 100) { output[i,j] <- input[i,j]}
                                    }
                                    }
 colnames(output) <- colnames(input)
 output <- data.frame(output)
 return(output)
                                     }
```

```
#Transformando tanto el Training Data como el Test Data:
trainingData <- editarvalores100(trainingData)
testData <- editarvalores100(testData)
```

11.2 LONGITUD

```
#Determinando el train y el test en el Training Data:
train <- trainingData[,1:521]
test <- testData[,1:521]
```

```
#Determinación del K optimo:
#Para tal objetivo, se ha creado la función que parta desde el número del modo de partición:
library(kknn)
library(ggplot2)
BusquedaK <- function(Nºset.seed) {
set.seed(Nºset.seed)
```

```
indxTrain <- createDataPartition(y = train$LONGITUDE,p = 0.95,list = FALSE)
training <- train[indxTrain,]
testing <- train[-indxTrain,]
tabla <- matrix( c(rep(0, 38)),19,2)
 n <- 1
 for (K in seq(1,37,2)) {
 longitude.kknn <- kknn(LONGITUDE~., training, testing, distance = 1, kernel = "triangular",
k= K)
 error<-mean(abs((testing$LONGITUDE-longitude.kknn$fitted.values)/testing$LONGITUDE))*100
tabla[n,] <- cbind(K,error)
          n <- n +1
                              }
resultados <- data.frame(NºNeighbour= tabla[,1], Error= tabla[,2])
return(resultados)
                                                  }
```

```
#Resultados por set.seed: 1,123,10000:
Resultadosetseed1<- BusquedaK(1)
head(Resultadosetseed1)
p <- ggplot(data= Resultadosetseed1, aes(x= NºNeighbour ,y= Error))
p <- p + geom_line(linetype= "dashed", color= "aquamarine", lwd= 1)
p<- p+ geom_point(color= "dark blue")
p <- p + labs (title="Optimization of NºNeighbour LONGITUDE (set.seed:1)", y= "Mean Error
(%)")
p
Resultadosetseed123<- BusquedaK(123)
head(Resultadosetseed123)
p <- ggplot(data= Resultadosetseed123, aes(x= NºNeighbour ,y= Error))
p <- p + geom_line(linetype= "dashed", color= "aquamarine", lwd= 1)
p <- p + geom_point(color= "dark blue")
p <- p + labs (title="Optimization of NºNeighbour LONGITUDE (set.seed:123)", y= "Mean
Error (%)")
p
Resultadosetseed10000<- BusquedaK(10000)
head(Resultadosetseed10000)
p <- ggplot(data= Resultadosetseed10000, aes(x= NºNeighbour ,y= Error))
p <- p + geom_line(linetype= "dashed", color= "aquamarine", lwd= 1)
p <- p + geom_point(color= "dark blue")
p <- p + labs (title="Optimization of NºNeighbour LONGITUDE (set.seed:10000)", y= "Mean
Error (%)")
p
```

```
#Entrenando en el train y evaluación de la calidad en el test K=3:
resultado <- kknn(LONGITUDE~., train, test, distance = 1, kernel = "triangular", k= 3)
tabla_resultado <- cbind(test$LONGITUDE, resultado$fitted.values, abs((test$LONGITUDE-
resultado$fitted.values)/test$LONGITUDE)*100
colnames(tabla_resultado) <- c("Actual", "Predicted", "Absolute percent error")
LONGITUDE_prediction <- tabla_resultado$Predicted
```

```
mean_percent_error <- mean(tabla_resultado[,3])
error_resultado <- tabla_resultado$Actual – tabla_resultado$Predicted
head(tabla_resultado)
```

```
#ANALISIS DE LOS ERRORES:
#Boxplot:
boxplot(error_resultado, ylab= "Error (m)" , col= "aquamarine", main= "Box of  Predicted Longitude Error"  )
```

```
#Probando con función Gumbel:
HISTOGRAMA <- function(LONGITUDEerror,B,u) {
mybinsize <- IQR(LONGITUDEerror)/4
mysd <- sd(LONGITUDEerror)
mymin <- min(LONGITUDEerror) - mysd*5
mymax <- max(LONGITUDEerror) + mysd*5
mybins <- seq(mymin,mymax, mybinsize)
hist(LONGITUDEerror, col= "aquamarine", freq=FALSE, breaks= mybins, xlim= c(-100,150), xlab="Error (m)" ,main= "Histogram of Predicted Longitude Error" )
z <- exp(-(mybins-u)/B)
gumbel <- (z*exp(-z))/B
points (mybins,gumbel, type= "l", col="dark blue",lwd=2)
                                                    }
HISTOGRAMA(error_resultado,4.6,-2)
text(78,0.05, "Gumbel(ß= 4.6,u= -2)"  ,cex=1.2,col= "dark blue")
arrows(28, 0.045, x1 = 13, y1 = 0.02, length = 0.08, angle = 40, code = 2, col = "dark blue",lwd = 2)
```

11.3 LATITUD

```
#Determinando el train y el test en el Training Data:
train <- cbind(trainingData[,1:520], trainingData[522])
test <- testData[,1:520],trainingData[522])
```

```
#Determinación del K optimo:
#Para tal objetivo, se ha creado la función que parta desde el número del modo de partición:
library(kknn)
library(ggplot2)
BusquedaK <- function(Nºset.seed) {
set.seed(Nºset.seed)
indxTrain <- createDataPartition(y = train$LATITUDE,p = 0.95,list = FALSE)
training <- train[indxTrain,]
testing <- train[-indxTrain,]
tabla <- matrix( c(rep(0, 38)),19,2)
n <- 1
```

```
 for (K in seq(1,37,2)) {
 latitude.kknn <- kknn(LATITUDE~., training, testing, distance = 1, kernel = "triangular", k=
K)
 error<-mean(abs((testing$LATITUDE-latitude.kknn$fitted.values)/testing$LATITUDE))*100
tabla[n,] <- cbind(K,error)
n <- n +1
                              }
resultados <- data.frame(NºNeighbour= tabla[,1], Error= tabla[,2])
return(resultados)
                                                  }
```

```
#Resultados por set.seed: 1,123,10000:
Resultadosetseed1<- BusquedaK(1)
head(Resultadosetseed1)
p <- ggplot(data= Resultadosetseed1, aes(x= NºNeighbour ,y= Error))
p <- p + geom_line(linetype= "dashed", color= "dark olive green 1", lwd= 1)
p <- p + geom_point(color= "dark green")
p <- p + labs (title="Optimization of NºNeighbour LATITUDE (set.seed:1)", y= "Mean Error
(%)")
p
Resultadosetseed123<- BusquedaK(123)
head(Resultadosetseed123)
p <- ggplot(data= Resultadosetseed123, aes(x= NºNeighbour ,y= Error))
p <- p + geom_line(linetype= "dashed", color= "dark olive green 1", lwd= 1)
p <- p + geom_point(color= "dark green")
p <- p + labs (title="Optimization of NºNeighbour LATITUDE (set.seed:123)", y= "Mean
Error (%)")
p
Resultadosetseed10000<- BusquedaK(10000)
head(Resultadosetseed10000)
p <- ggplot(data= Resultadosetseed10000, aes(x= NºNeighbour ,y= Error))
p <- p + geom_line(linetype= "dashed", color= "dark olive green 1", lwd= 1)
p <- p + geom_point(color= "dark green")
p <- p + labs (title="Optimization of NºNeighbour LATITUDE (set.seed:10000)", y= "Mean
Error (%)")
p
```

```
#Entrenando en train y evaluación de la calidad en el test  K=3:
resultado <- kknn(LATITUDE~., train, test, distance = 1, kernel = "triangular", k= 3)
tabla_resultado <- cbind(test$LATITUDE, resultado$fitted.values, abs((test$LATITUDE-
resultado$fitted.values)/test$LATITUDE)*100
colnames(tabla_resultado) <- c("Actual", "Predicted", "Absolute percent error")
LATITUDE_prediction <- tabla_resultado$Predicted
mean_percent_error <- mean(tabla_resultado[,3])
error_resultado <- tabla_resultado$Actual – tabla_resultado$Predicted
head(tabla_resultado)
```

```
#ANALISIS DE LOS ERRORES:
#Boxplot:
boxplot(error_resultado, ylab= "Error (m)" , col= "dark olive green 1", main= "Box of Predicted Latitude Error"  )
```

```
#Probando con función Gumbel:
HISTOGRAMA <- function(LATITUDEerror,B,u) {
mybinsize <- IQR(LATITUDEerror)/4
mysd <- sd(LATITUDEerror)
mymin <- min(LATITUDEerror) - mysd*5
mymax <- max(LATITUDEerror) + mysd*5
mybins <- seq(mymin,mymax, mybinsize)
hist(LATITUDEerror, col= "dark olive green 1", freq=FALSE, breaks= mybins, xlim= c(-100,150), xlab="Error (m)" ,main= "Histogram of Predicted Latitude Error" )
z <- exp(-(mybins-u)/B)
gumbel <- (z*exp(-z))/B
points (mybins,gumbel, type= "l", col="dark green",lwd=2)
                                                    }
HISTOGRAMA(error_resultado,4.6,-2)
text(80,0.04, "Gumbel(ß= 6.6,u= -2.4)"  ,cex=1.2,col= "dark green")
arrows(28, 0.038, x1 = 13, y1 = 0.02, length = 0.08, angle = 40, code = 2, col = "dark green",lwd = 2)
```

11.4 Análisis del error en las distancias en planta

```
distanciaerror <- sqrt((testData$LONGITUDE–LONGITUDE_prediction)^2 +
(testData$LATITUDE- LATITUDE_prediction)^2)
mean(distanciaerror)
head(distanciaerror)
```

```
#Análisis con función Exponencial:
histo <- function(variable,landa) {
mybinsize <- IQR(variable)/4
mysd <- sd(variable)
mymin <- min(variable)
mymax <- max(variable) + mysd/2
mybins <- seq(mymin,mymax, mybinsize)
hist(variable,col= "pink", freq=FALSE, breaks= mybins,xlab="Error (m)", main="Histogram of Distance Error")
points (mybins,landa*exp(-landa*mybins),type= "l", col="dark red",lwd=2 )
                                        }
histo(distanciaerror,0.08)
```

```
text(100,0.04,"Función Exponencial (λ=0.08)",cex=1.2 ,col= "dark red")
arrows(42, 0.037, x1 = 20, y1 = 0.02, length = 0.08, angle = 40, code = 2, col = "dark red",lwd
= 2)
boxplot(distanciaerror, ylab= "Error (m)", col= "pink", main="Box of Distance Error")
summary(distanciaerror)
```

11.5 FLOOR

```
#Determinando el train y el test en el Training Data:
train <- cbind(trainingData[,1:520], trainingData[523])
test <- testData[,1:520],trainingData[523])
train$FLOOR <- as.factor(train$FLOOR)
test$FLOOR <- as.factor(test$FLOOR)
```

```
#Determinación del K optimo:
#Para tal objetivo, se ha creado la función que parta desde el número del modo de
partición:
library(kknn)
library(dplyr)
library(ggplot2)
BusquedaK <- function(Nºset.seed) {
set.seed(Nºset.seed)
indxTrain <- createDataPartition(y = train$FLOOR,p = 0.95,list = FALSE)
training <- train[indxTrain,]
testing <- train[-indxTrain,]
tabla <- matrix( c(rep(0, 38)),19,2)
        n <- 1
 for (K in seq(1,37,2)) {
 floor.kknn <- kknn(FLOOR~., training, testing, distance = 1, kernel = "triangular", k= K)
 comparisontable <- data.frame( actual= testing$FLOOR, predicted=
floor.kknn$fitted.values)
accuracy <- nrow(filter( comparisontable, actual == predicted))/nrow(comparisontable)
tabla[n,] <- cbind(K,accuracy)
        n <- n +1
                        }
 resultados <- data.frame(NºNeighbour= tabla[,1], Error= tabla[,2])
 return(resultados)
                                    }
```

```
#Resultados por set.seed: 1,123,10000, 1234123
Resultadosetseed1<- BusquedaK(1)
head(Resultadosetseed1)
p <- ggplot(data= Resultadosetseed1, aes(x= NºNeighbour ,y= Accuracy))
p <- p + geom_line(linetype= "dashed", color= "gray 60", lwd= 1)
```

```
p <- p + geom_point(color= "gray 0")
p <- p + labs (title="Optimization of NºNeighbour FLOOR (set.seed:1)", y= "Accuracy")
p
Resultadosetseed123<- BusquedaK(123)
head(Resultadosetseed123)
p <- ggplot(data= Resultadosetseed123, aes(x= NºNeighbour ,y= Accuracy))
p <- p + geom_line(linetype= "dashed", color= "gray 60", lwd= 1) + geom_point(color= "gray 0")
p <- p + labs (title="Optimization of NºNeighbour FLOOR (set.seed:123)", y= "Accuracy")
p
Resultadosetseed10000<- BusquedaK(10000)
head(Resultadosetseed10000)
p <- ggplot(data= Resultadosetseed10000, aes(x= NºNeighbour ,y= Accuracy))
p <- p + geom_line(linetype= "dashed", color= "gray 60", lwd= 1) + geom_point(color= "gray 0")
p <- p + labs (title="Optimization of NºNeighbour FLOOR (set.seed:10000)", y= "Accuracy")
Resultadosetseed1234123<- BusquedaK(1234123)
head(Resultadosetseed1234123)
p <- ggplot(data= Resultadosetseed1234123, aes(x= NºNeighbour ,y= Accuracy))
p <-  p + geom_line(linetype= "dashed", color= "gray 60", lwd= 1)
p <- p + geom_point(color= "gray 0")
p <- p + labs (title="Optimization of NºNeighbour FLOOR (set.seed:1234123)", y= "Accuracy")
p
```

```
# Cálculo de la precisión media:
MeanAccuracy <- (Resultadoset.seed1 + Resultadoset.seed123 + Resultadoset.seed10000 + Resultadoset.seed1234123)/4
head(MeanAccuracy)
p <- ggplot(data= MeanAccuracy, aes(x=NºNeighbour,y= Accuracy))
p <- p + geom_line(linetype= "dashed", color= "gray 60", lwd= 2)
p <- p + geom_point(color= "gray 0", fill="gray 0", size=3)
p <- p + labs ( y= "Mean Accuracy", title="Optimization of NºNeighbour FLOOR")
p
```

```
#Evaluar el modelo:
resultado <- kknn(FLOOR~., train, test, distance = 1, kernel = "triangular", k= 7)
tabla_resultado <- data.frame(Predicted=resultado$fitted.values, Actual= test$FLOOR)
head(tabla_resultado)
Matriz_Confusion <-table(tabla_resultado)
nºROC <- nrow(Matriz_Confusion)-1
ROC <- data.frame(FPR= c(rep(0,nºROC)), TPR= c(rep(0,nºROC)) )
sumadiagonal <- 0
sumaninj <- 0
for (i in 1:nºROC ) {
fpr <- (1-Matriz_Confusion[i+1,i+1]/sum(Matriz_Confusion[i+1,]))
tpr <- Matriz_Confusion[i,i]/sum(Matriz_Confusion[i,])
ROC[i,] <- cbind(fpr,tpr)
```

```
sumadiagonal <- sumadiagonal + Matriz_Confusion[i,i]
sumaninj <- sumaninj + sum(Matriz_Confusion[i,])*sum(Matriz_Confusion[,i])
 if ( i == nºROC) { sumadiagonal <- sumadiagonal + Matriz_Confusion[nºROC +1, nºROC +1]
 sumanninj <- sumaninj + sum(Matriz_Confusion[nºROC +
1,])*sum(Matriz_Confusion[,nºROC + 1])}
                              }
Pa <- sumaninj/(sum(Matriz_Confusion)^2)
Accuracy <- sumadiagonal /sum(Matriz_Confusion)
Kappa <- (Accuracy  - Pa)/(1- Pa)
p <- ggplot(ROC, aes( x= FPR, y= TPR)) + geom_point(shape= 4, col= "dark blue", lwd= 4)
p <- p  + xlim(0,1) + ylim(0,1)
P <- p + geom_segment( aes(x = 0, y = 0, xend = 1, yend = 1),linetype= "dashed", col=
"red", lwd=1)
p <- p + labs( title= "TestData ROC: FLOOR")
p
```