Proyecto final - Aprendizaje automático

Jacinto Carrasco Castillo

30 de junio de 2016

1. Definición del problema a resolver y enfoque elegido

La base de datos utilizada es *Forest Fires Data Set*, disponible en la UCI. Es un problema de regresión donde el objetivo es predecir la extensión del área quemada en un incendio en el parque natural de *Montesinho*, situado en el noreste de Portugal. Para hacerlo usaremos datos de carácter metereológico e índices sobre el terreno y la vegetación. Estos índices pertenecen al sistema canadiense FWI (*Forest Fire Weather Index*).

2. Codificación de los datos de entrada para hacerlos útiles a los algoritmos

Los datos de entrada son, para cada dato:

- Coordenada X en el mapa del parque (1-9)
- Coordenada Y en el mapa del parque (2-9)
- Mes del año en el que se produce el incendio
- Día de la semana en el que se produce el incendio
- FFMC, DMC, DC, ISI: Índices del sistema FWI
- Temperatura (C)
- Humedad relativa (%)
- Velocidad del viento (Km/h)
- Lluvia tenida lugar media hora antes del incendio (mm/mm^2)
- Área quemada: Variable de salida, área quemada en el bosque (ha)

Establecemos el directorio de trabajo y leemos la base de datos

```
setwd("~/Documentos/Aprendizaje Automático/AA-1516/Proyecto")
ForestFires <- read.csv("forestfires.csv")
```

Comenzamos a tratar los datos para que nos sean de utilidad. Puesto que hemos considerado relevantes la fecha en la que se produjo el incendio y, pese a que son etiquetas es una variable ordinal (hay un orden y una distancia entre los diferentes meses y los distintos días de la semana que es la que nos induce a pensar que puede ser relevante guardar esta información). Pasamos el día de la semana y el mes a números.

```
months.names = c("jan","feb","mar","apr","may","jun",
                 "jul","aug","sep","oct","nov","dec")

month.n <- sapply(ForestFires$month, function(x) which(months.names==x))
```

```
dow.names = c("mon","tue","wed","thu","fri","sat","sun")

day.n <- sapply(ForestFires$day, function(x) which(dow.names==x))
```

Modificamos el `data.frame` para cambiar las variables existentes por las creadas.

```
ForestFires <- data.frame(cbind(ForestFires[c(1,2)],month.n,
                                day.n,ForestFires[5:13]))
```

Como es lógico, separamos en datos de entrenamiento y test, que no tocaremos hasta que debamos obtener el error del modelo que surja de estos datos. Establecemos también una semilla para que los resultados

sean reproducibles. Obtenemos la matriz de los datos de entrenamiento y de test para usar de manera más cómoda el modelo.

```
set.seed(3141592)
index.train <- sample(1:nrow(ForestFires),
                      size=0.8*nrow(ForestFires))

FF.train <- ForestFires[index.train,]
FF.test <- ForestFires[-index.train,]

FF.train.data <- model.matrix(area ~ ., FF.train)[,-1]
FF.test.data <- model.matrix(area ~ ., FF.test)[,-1]
```

3. Valoración del interés de las variables para el problema y selección de un subconjunto (en su caso)

En un principio podemos considerar relevantes todas las variables disponibles, puesto que las coordenadas influirán en el tipo de vegetación, accesibilidad para los bomberos que extinguirán el fuego o el tipo de terreno. En el caso del mes, los incendios mayores tendrán lugar en verano, y el día de la semana influirá en que haya mayor afluencia en el parque natural en fin de semana, con lo que habrá un mayor número de incendios. Los índices del sistema FWI tratan sobre la composición del suelo y combinaciones de estos índices con los metereológicos. El viento y las altas temperaturas harán que los incendios sean mayores, por contra la lluvia puede facilitar la extinción. Durante el desarrollo de la actividad, se estudiará la influencia de cada variable según los datos disponibles.

4. Normalización de las variables (en su caso)

Normalizamos las variables para que las variables tengan la misma influencia. Ya hemos realizado la transformación de las etiquetas con meses y días de la semana y no podemos obviar que esta transformación (pese a ser la más natural) influiría en los resultados si no normalizásemos.

```
scale.factor <- scale(FF.train.data)
FF.train.data <- scale.factor
FF.test.data <- scale(FF.test.data,
                      center = attr(scale.factor,"scaled:center"),
                      scale = attr(scale.factor,"scaled:scale"))
```

Nótese que la normalización se realiza una vez se ha separado en datos de entrenamiento y de test para que así los datos de test no influyan en la escala y poder asegurar que los resultados no están sesgados.

Para justificar la decisión inicial de utilizar todos los datos, observamos la correlación de todas las variables con respecto a la extensión quemada:

```
cor(FF.train.data, FF.train$area)
```

```
##                [,1]
## X        0.064742210
## Y        0.059510842
## month.n  0.061379270
## day.n    0.037525302
## FFMC     0.047182621
## DMC      0.078142998
## DC       0.057290608
```

```
## ISI      0.014557863
## temp     0.098551337
## RH      -0.087373129
## wind     0.017911031
## rain    -0.005764018
```

La correlación para todas las variables es muy pequeña:

```
sapply( 1:12, function(i) cor.test(FF.train.data[,i],
                                   FF.train$area)$p.value)
```

```
##  [1] 0.18914677 0.22750629 0.21321724 0.44691943 0.33882434 0.11281517
##  [7] 0.24535397 0.76801703 0.04532728 0.07612215 0.71666386 0.90703073
```

De hecho, si realizamos un test para rechazar o no rechazar la hipótesis de que la correlación entre cada variable y el área quemada sea 0, la única variable para la que se rechaza esta hipótesis con un nivel de significación de 0.05 es la temperatura, y queda cerca de ser también rechazada para la humedad relativa.

5. Selección de las técnicas y su valoración de la idoneidad de la misma.

Modelo paramétrico

La técnica paramétrica seleccionada es regresión con *weight decay*. La decisión se debe a que la correlación de las variables con el área es muy pequeña, con lo que tratamos de evitar el ruido que pudieran introducir las variables con menor influencia. El motivo por el que no se ha utilizado la regresión con regularización LASSO es que, basándonos en la correlación, no hay ningun subconjunto de variables que sea realmente influyente, con lo que al usar LASSO podríamos quedarnos con un modelo con una o ninguna variable.

Haremos validación cruzada utilizando la función `cv.glmnet` de la biblioteca `glmnet`. De esta manera, se pretende ajustar el parámetro de regularización λ.

```
cv.wdecay <- cv.glmnet(FF.train.data, FF.train$area,
                       family = "gaussian", alpha=0)
lambda.min <- cv.wdecay$lambda.min
ridge.mod = glmnet(FF.train.data, FF.train$area, alpha=0,
                   lambda=lambda.min)
```

Una vez que tenemos el modelo con el $\lambda_{\min}$ obtenemos los valores predichos y calculamos el MSE, $RSME$ y el MAD:

```
ridge.pred = predict(ridge.mod, s=lambda.min, newx=FF.test.data)

MSE.ridge.all <- mean((ridge.pred - FF.test$area)^2)
RMSE.ridge.all <- sqrt(mean((ridge.pred - FF.test$area)^2))

abs.error.ridge <- abs(ridge.pred - FF.test$area)
abs.error.ridge <- abs.error.ridge[order(abs.error.ridge)]
MAD.ridge.all <- mean(abs.error.ridge)

cat("MSE", MSE.ridge.all)
```

```
## MSE 1171.793
```

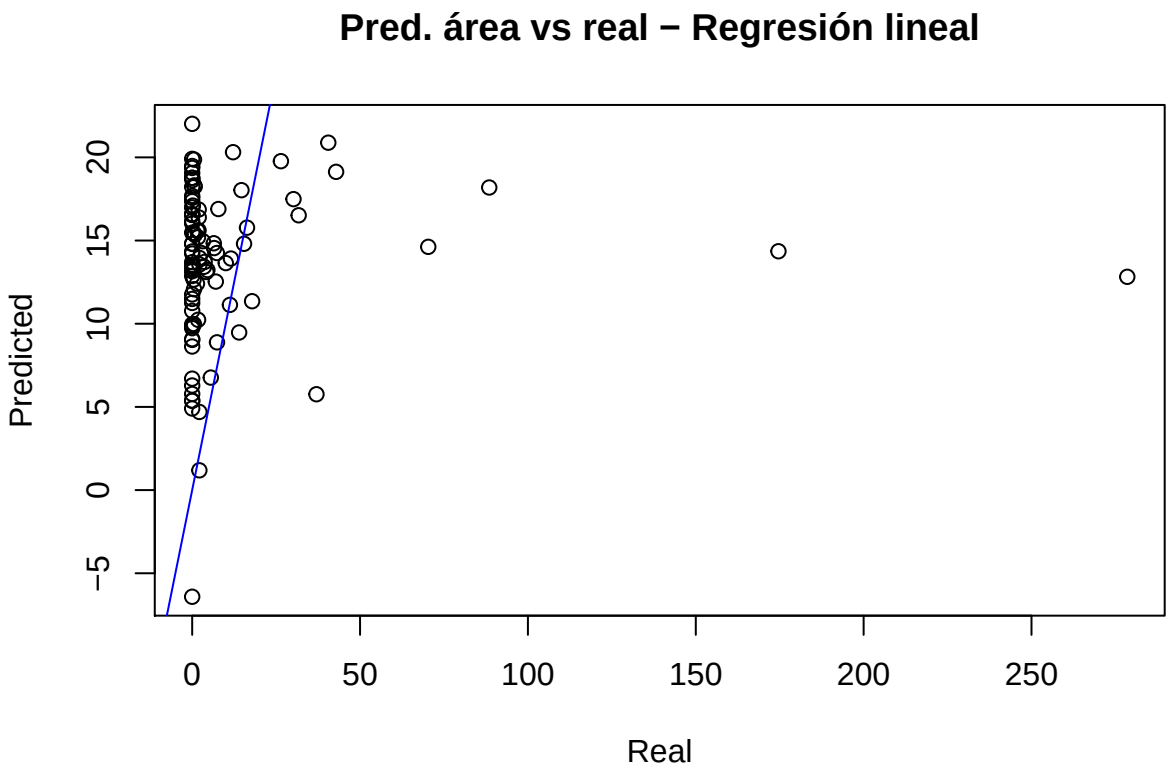
```
cat("RMSE", RMSE.ridge.all)
```

```
## RMSE 34.23146
```

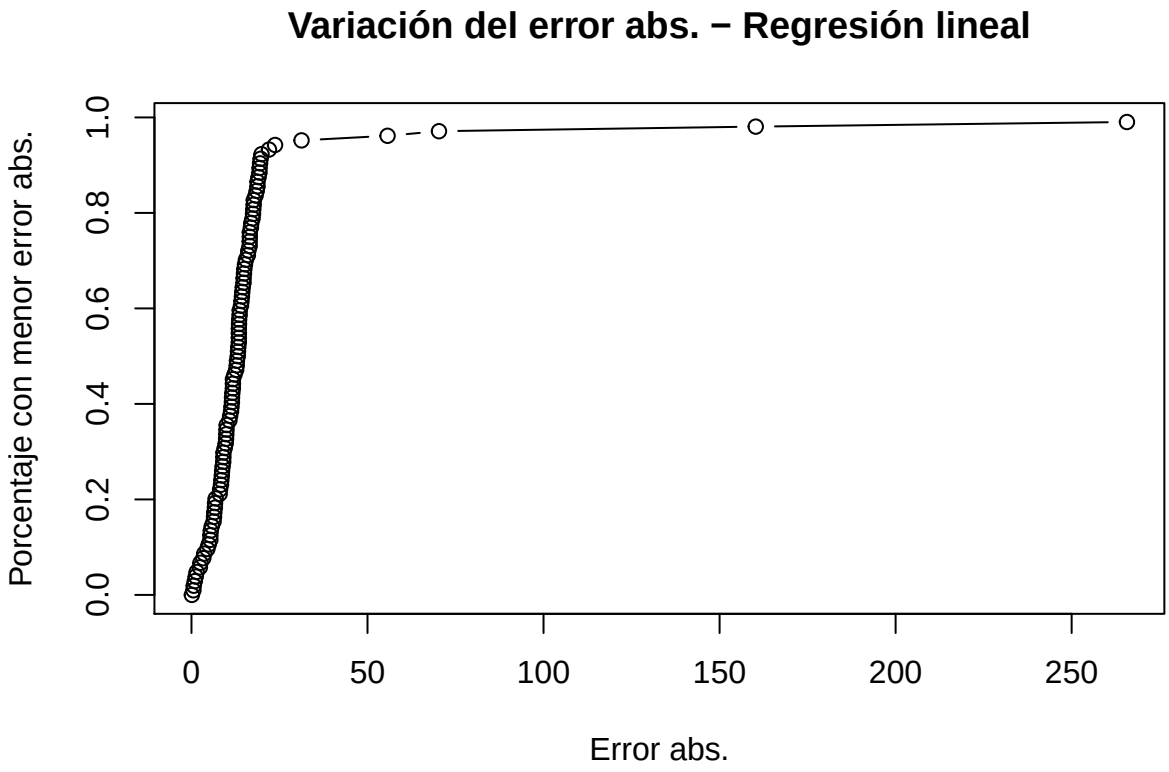
```
cat("MAD", MAD.ridge.all)
```

```
## MAD 16.81005
```

```
plot(FF.test$area, ridge.pred,
     main="Pred. área vs real - Regresión lineal",
     xlab = "Real", ylab = "Predicted")
abline(0 ,1, col = 4)
```




```
plot(abs.error.ridge,
     sapply(1:length(FF.test$area),
            function(i){
              sum(abs.error.ridge < abs.error.ridge[i])/
                length(FF.test$area)
            }),
     type="b", main="Variación del error abs. - Regresión lineal",
     xlab = "Error abs.", ylab = "Porcentaje con menor error abs.")
```



Mostramos una gráfica con la comparación del valor predicho por el modelo con el valor real. En este caso se observa como hay un valor para el que se ha predicho una extensión negativa, algo que no tiene sentido. Para buena parte de los datos (que tienen valor 0) se sobreestima la extensión, en cambio, como veremos con los demás métodos, es difícil predecir los incendios más graves y se le da un valor mucho menor.
En la segunda gráfica se observa cómo casi la totalidad de los datos tiene un error absoluto de menos de 25. Aunque es cierto que esto es fácil si tenemos en cuenta que la mayoría de los valores son pequeños, sí que se aprecia como los valores mayores que 0 y menores de 50 tienden a estar bien representados.

Modelo no paramétrico: *random forest*

Como modelo no paramétrico usaremos en primer lugar *random forest*. La elección de este modelo se debe aque se pretende identificar los valores de los atributos de los mayores incendios, que son aquellos que contribuyen en mayor medida.
Ajustamos en primer lugar el número de variables a usar, es decir el parámetro `mtry` mediante validación cruzada con la función `rfcv`.

```
cv.rf <- rfcv(FF.train.data, FF.train$area)
mtray.min <- unname(which.min(cv.rf$error.cv))
```

Una vez obtenemos que el mejor resultado se obtiene cuando se seleccionan 3 variables, ajustamos el número de árboles implementando la validación cruzada.

```
k <- 5
folds <- kfold(FF.train, k = k)

CV.error.ntree <- sapply( seq(100, 700, by=10),
        function(x){
          MSE <- 0
          for( i in 1:k ){
            rf.model <- randomForest(area ~ ., data=FF.train,
```

```
                                    subset=which(folds!=i),
                                    importance = TRUE, mtray=mtray.min,
                                    ntree=x)
            pred.area.rf <- predict( rf.model,
                                     newdata=FF.train[which(folds==i),])
            MSE <- MSE + mean((pred.area.rf -
                                FF.train$area[which(folds==i)])^2)
          }

          return(MSE/k)
        }
      )
best.ntree <- seq(100, 700, by=10)[which.min(CV.error.ntree)]
```

Entrenamos y ejecutamos el modelo con los parámetros obtenidos:

```
rf.model <- randomForest(area ~ ., data=FF.train,
                                    importance = TRUE, mtray=mtray.min,
                                    ntree=best.ntree)
pred.area.rf <- predict( rf.model, newdata=FF.test.data)
```

```
MSE.rf.all <- mean((pred.area.rf - FF.test$area)^2)
RMSE.rf.all <- sqrt(mean((pred.area.rf - FF.test$area)^2))

abs.error.rf <- abs(pred.area.rf - FF.test$area)
abs.error.rf <- abs.error.rf[order(abs.error.rf)]
MAD.rf.all <- mean(abs.error.rf)

cat("MSE", MSE.rf.all)
```

```
## MSE 1263.146
```

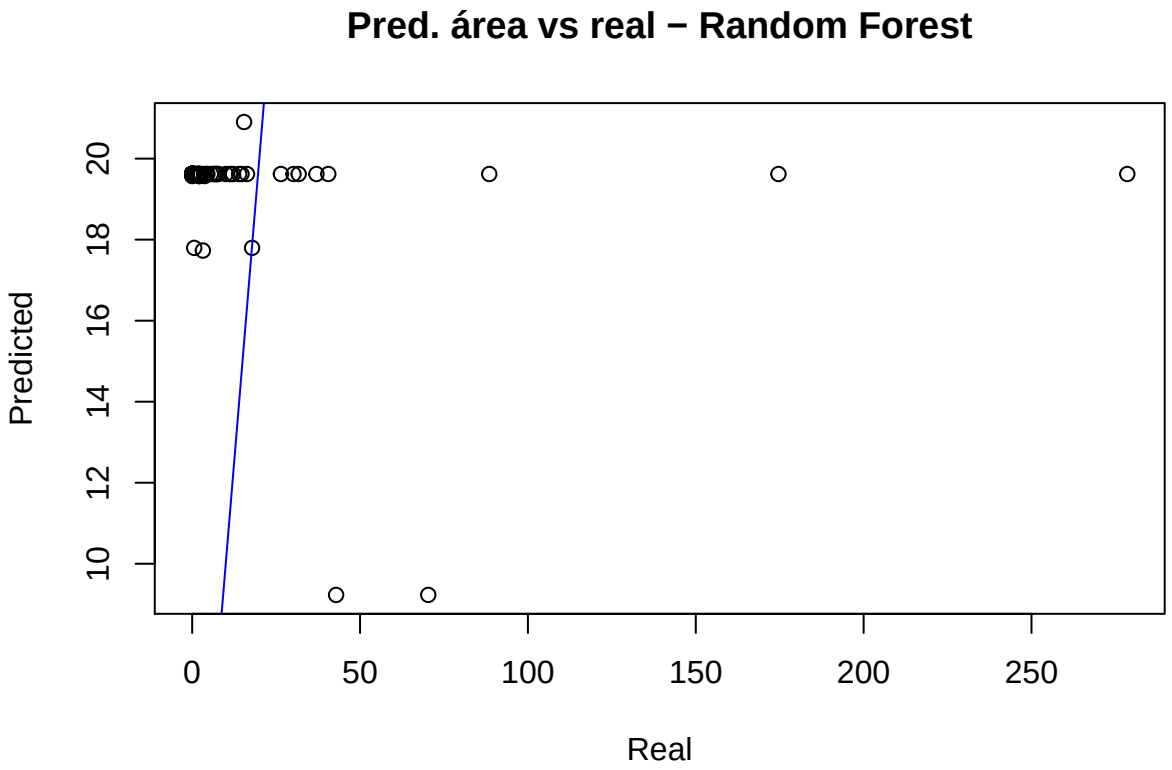
```
cat("RMSE", RMSE.rf.all)
```

```
## RMSE 35.54077
```

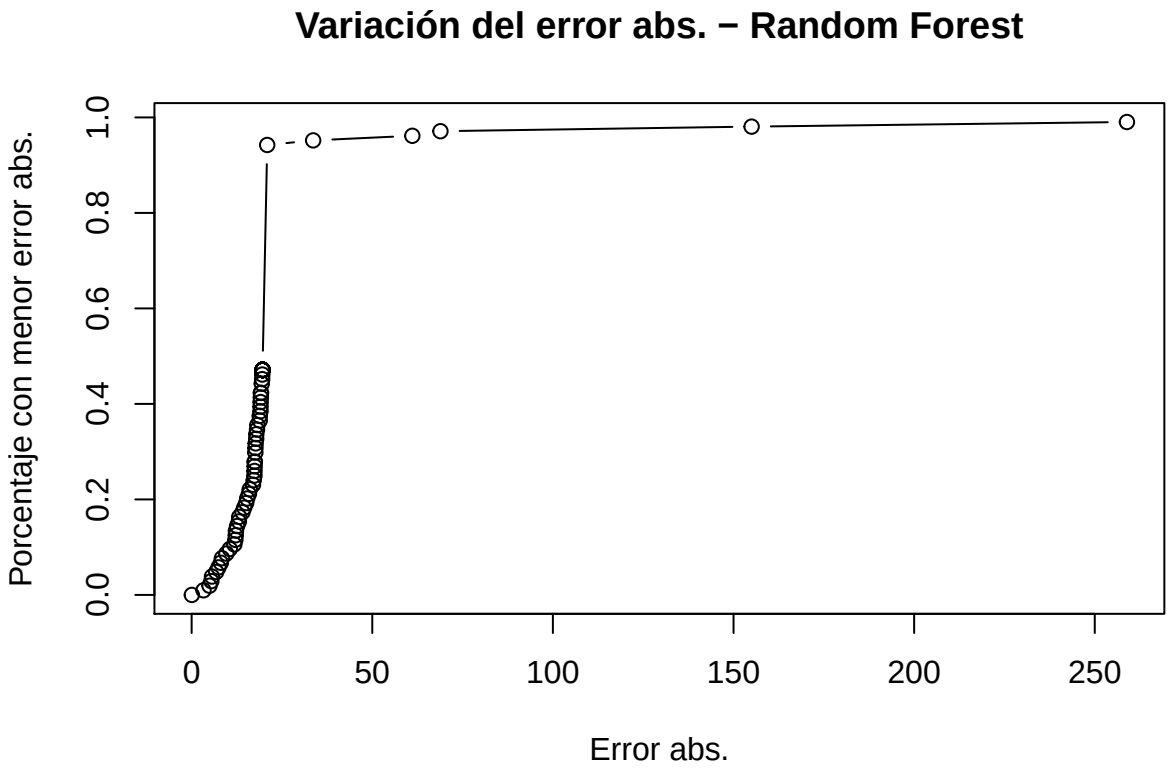
```
cat("MAD", MAD.rf.all)
```

```
## MAD 21.78636
```

```
plot(FF.test$area, pred.area.rf,
     main="Pred. área vs real - Random Forest",
     xlab = "Real", ylab = "Predicted")
abline(0 ,1, col = 4)
```




```
plot(abs.error.rf,
     sapply(1:length(FF.test$area),
            function(i){
              sum(abs.error.rf < abs.error.rf[i])/
                length(FF.test$area)
            }),
     type="b", main="Variación del error abs. - Random Forest",
     xlab = "Error abs.", ylab = "Porcentaje con menor error abs.")
```



Los resultados indican que hay un mayor error con *random forest*. Si observamos la gráfica, se observa que el rango de la variable de salida es menor y que la función de regresión calculada se mueve únicamente entre ciertos valores.

Función de base radial

Utilizando las funciones de base radial estamos aprendiendo basándonos en la semejanza a los datos disponibles. Ya por los resultados que estamos obteniendo, donde los incendios grandes son difíciles de detectar y predominan claramente los pequeños incendios, podemos intuir que también será complejo que esta familia de funciones nos proporcione buenos resultados, ya que la abundancia de áreas quemadas cercanas a 0 hará que estos valores predominen a la hora de componer la estimación para cada dato. Usaremos $\Phi(z) = e^{\frac{-z^2}{2}}$ para que así disminuya el aporte de cada dato a una nueva predicción conforme nos alejamos del dato.

Definimos las funciones que usaremos.

```
norm <- function(x){
  return(t(x)%*%x)
}

phi <- function(z){
  exp(-1/2*z^2)
}

alpha <- function(x,xn,r=10){
  return(phi(norm(x-xn)/r))
}
```

```
g <- function(x,data,label,r){
  vec_alpha <- apply(data, 1, function(xn){
```

```
    return(alpha(x,xn,r))
    })

  return(t(vec_alpha)%*%label/sum(vec_alpha))
}
```

Realizamos validación cruzada para estimar el parámetro r, que nos determine el radio de influencia de cada dato.

```
k <- 5
folds <- kfold(FF.train, k = k)

CV.error.r <- sapply( seq(1, 100, by=10),
      function(r){
        MSE <- 0
        for( i in 1:k ){
          pred.area.radial <- apply(FF.train.data[which(folds==i),], 1,
                      function(x) g(x,FF.train.data[which(folds!=i),],
                                    FF.train$area[which(folds!=i)] ,r))
          MSE <- MSE + mean((pred.area.radial -
                                FF.train$area[which(folds==i)])^2)
        }

        return(MSE/k)
      }
      )
best.r <- seq(1, 100, by=10)[which.min(CV.error.r)]
```

Al haber obtenido $r = 31$, teniendo 12 variables normalizadas, estamos diciendo que datos relativamente lejanos tienen influencia. Como consecuencia, se espera (de nuevo) una

```
pred.area.radial <- apply(FF.test.data, 1,
                          function(x) g(x, FF.train.data,
                                        FF.train$area, best.r))

MSE.radial.all <- mean((pred.area.radial - FF.test$area)^2)
RMSE.radial.all <- sqrt(mean((pred.area.radial - FF.test$area)^2))

abs.error.radial <- abs(pred.area.radial - FF.test$area)
abs.error.radial <- abs.error.radial[order(abs.error.radial)]
MAD.radial.all <- mean(abs.error.radial)

cat("MSE", MSE.radial.all)
```

```
## MSE 1165.066
```

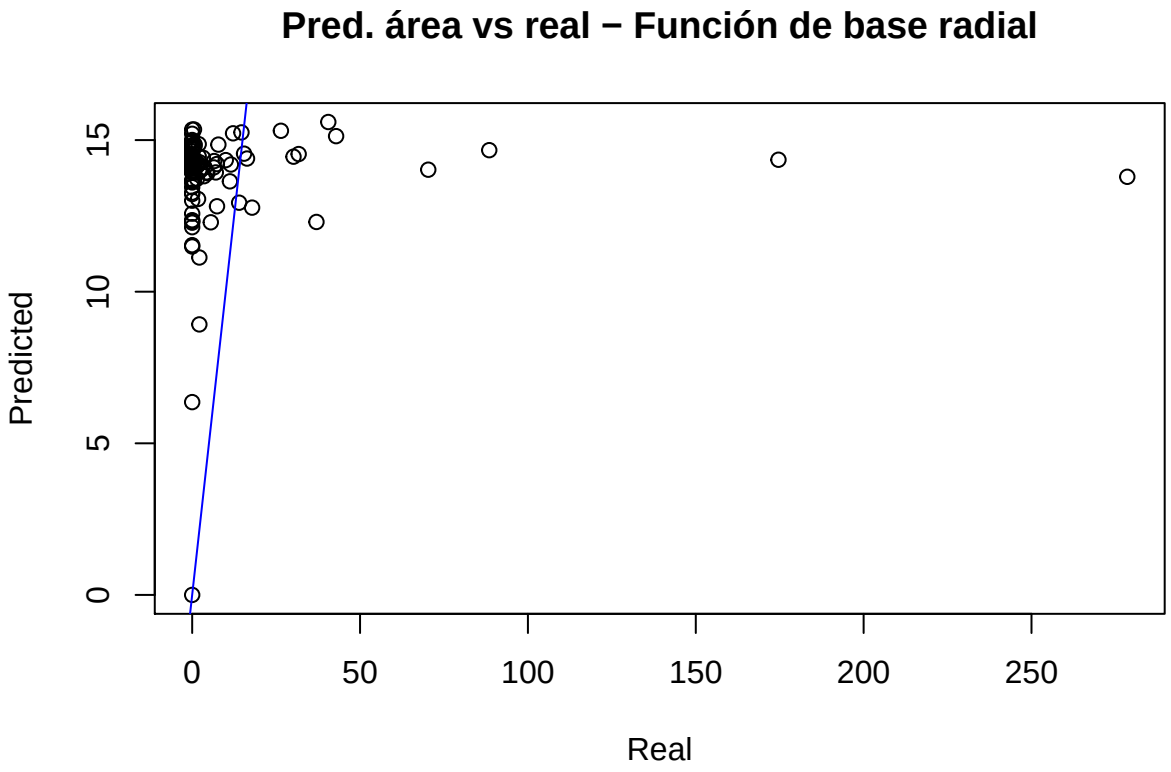
```
cat("RMSE", RMSE.radial.all)
```

```
## RMSE 34.13306
```

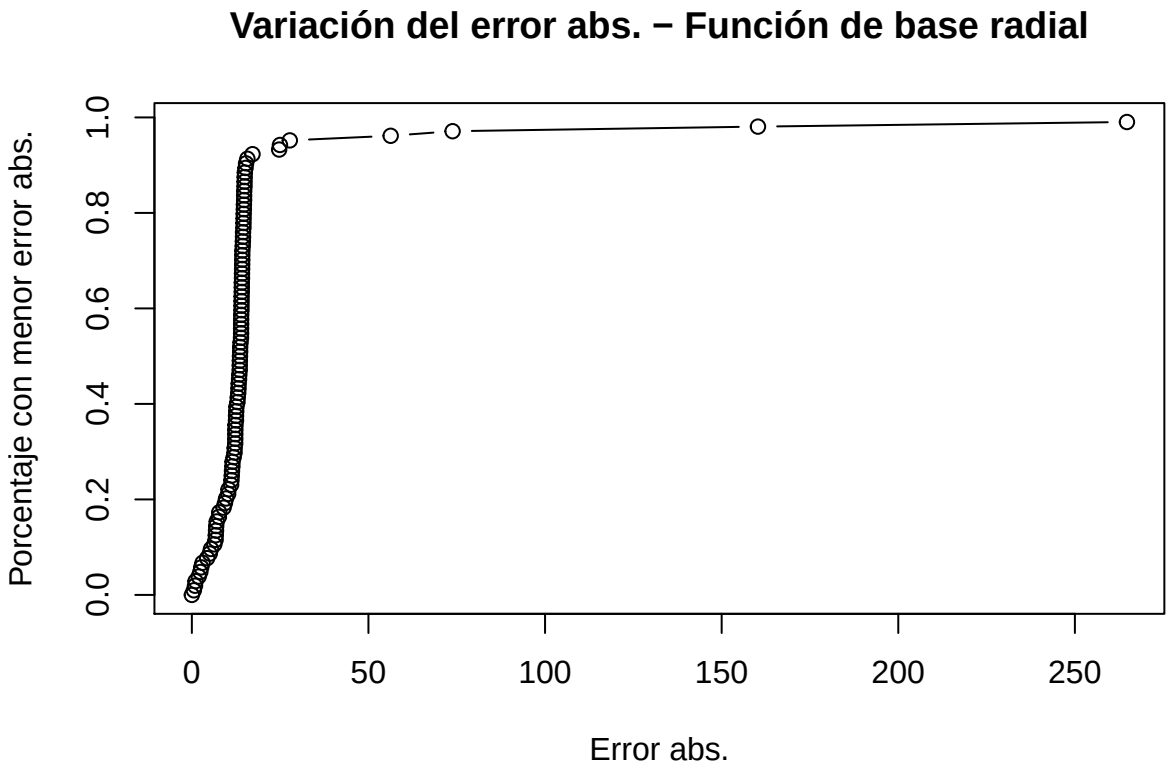
```
cat("MAD", MAD.radial.all)
```

```
## MAD 16.96699
```

```
plot(FF.test$area, pred.area.radial,
     main="Pred. área vs real - Función de base radial",
     xlab = "Real", ylab = "Predicted")
abline(0 ,1, col = 4)
```




```
plot(abs.error.radial,
     sapply(1:length(FF.test$area),
            function(i){
              sum(abs.error.radial < abs.error.radial[i])/
                length(FF.test$area)
            }),
     type="b", main="Variación del error abs. - Función de base radial",
     xlab = "Error abs.", ylab = "Porcentaje con menor error abs.")
```



De nuevo se comprueba que los incendios graves no son detectados y que la mayoría de valores asignados se concentran en torno a la media del área quemada, lo que no es muy útil si se pretende preveer el peligro de un incendio. Vemos como el error es similar al de los otros modelos.

Redes neuronales

Como última técnica usaremos redes neuronales. El beneficio que puede aportar este modelo es, de manera equivalente a la de *random forest*, encontrar los pesos que nos permitan obtener una buena aproximación.

Hacemos ahora validación cruzada para estimar el número de capas y el tamaño de cada capa. Los tamaños de las capas van creciendo conforme añadimos más capas.

```
nnet.model <- mlp(FF.train.data, FF.train$area, maxit = 1000,
                  size = best.size, learnFunc="Rprop",
                  linOut = T)
pred.area.nnet <- predict(nnet.model, newdata=FF.test.data, type="raw")

MSE.nnet.all <- mean((pred.area.nnet - FF.test$area)^2)
RMSE.nnet.all <- sqrt(mean((pred.area.nnet - FF.test$area)^2))

abs.error.nnet <- abs(pred.area.nnet - FF.test$area)
abs.error.nnet <- abs.error.nnet[order(abs.error.nnet)]
MAD.nnet.all <- mean(abs.error.nnet)

cat("MSE", MSE.nnet.all)
```

```
## MSE 2029.647
```

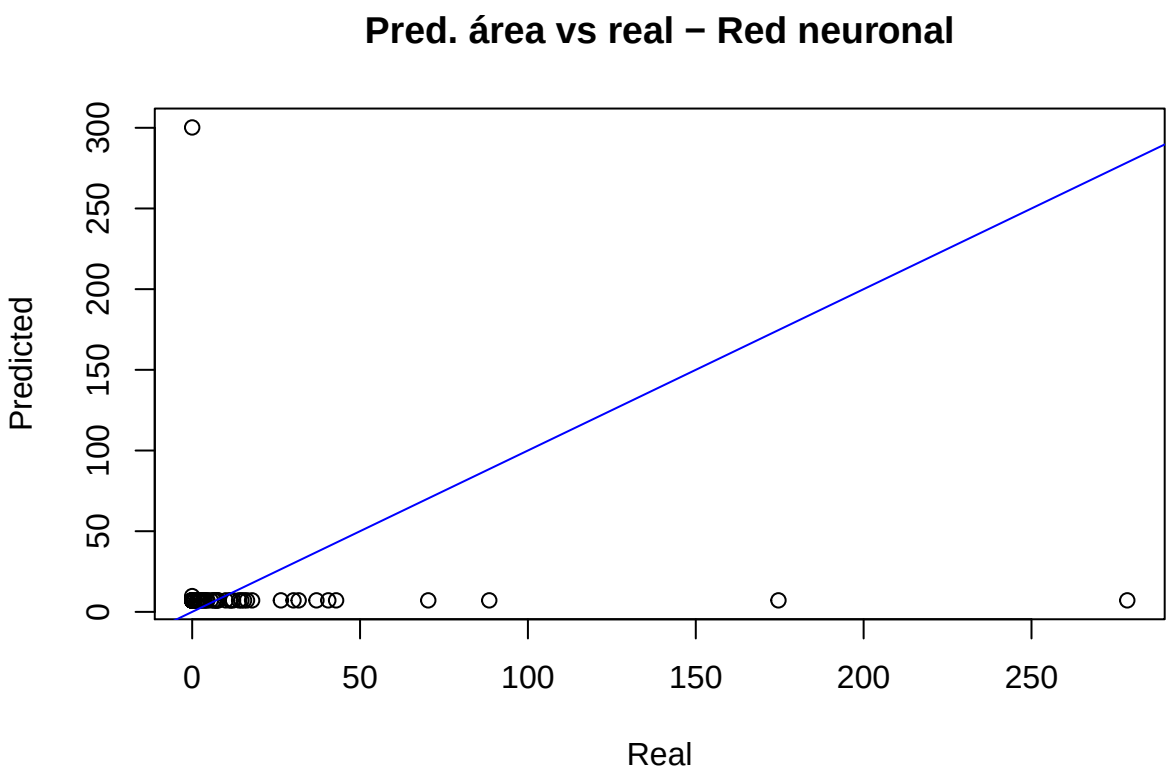
```
cat("RMSE", RMSE.nnet.all)
```

```
## RMSE 45.0516
```

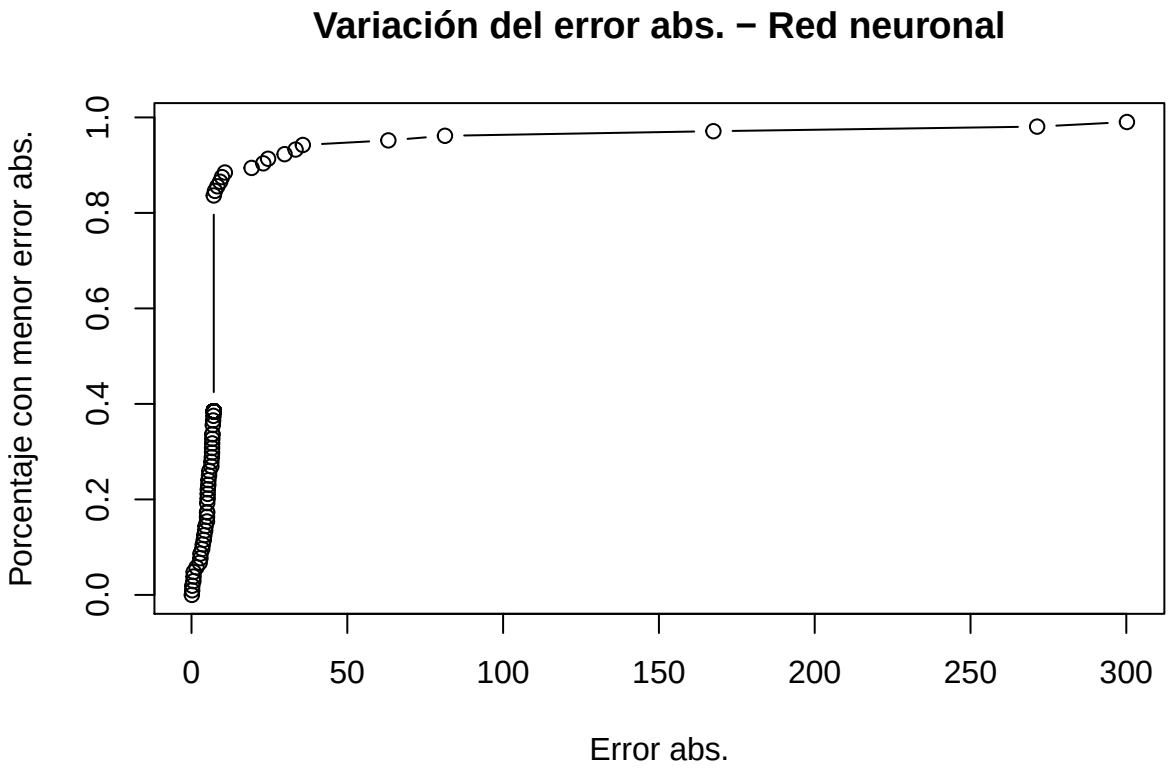
```r
cat("MAD", MAD.nnet.all)
```

```
## MAD 15.55122
```

```r
plot(FF.test$area, pred.area.nnet,
     main="Pred. área vs real - Red neuronal",
     xlab = "Real", ylab = "Predicted")
abline(0 ,1, col = 4)
```




```r
plot(abs.error.nnet,
     sapply(1:length(FF.test$area),
            function(i){
              sum(abs.error.nnet < abs.error.nnet[i])/
                length(FF.test$area)
            }),
     type="b", main="Variación del error abs. - Red neuronal",
     xlab = "Error abs.", ylab = "Porcentaje con menor error abs.")
```



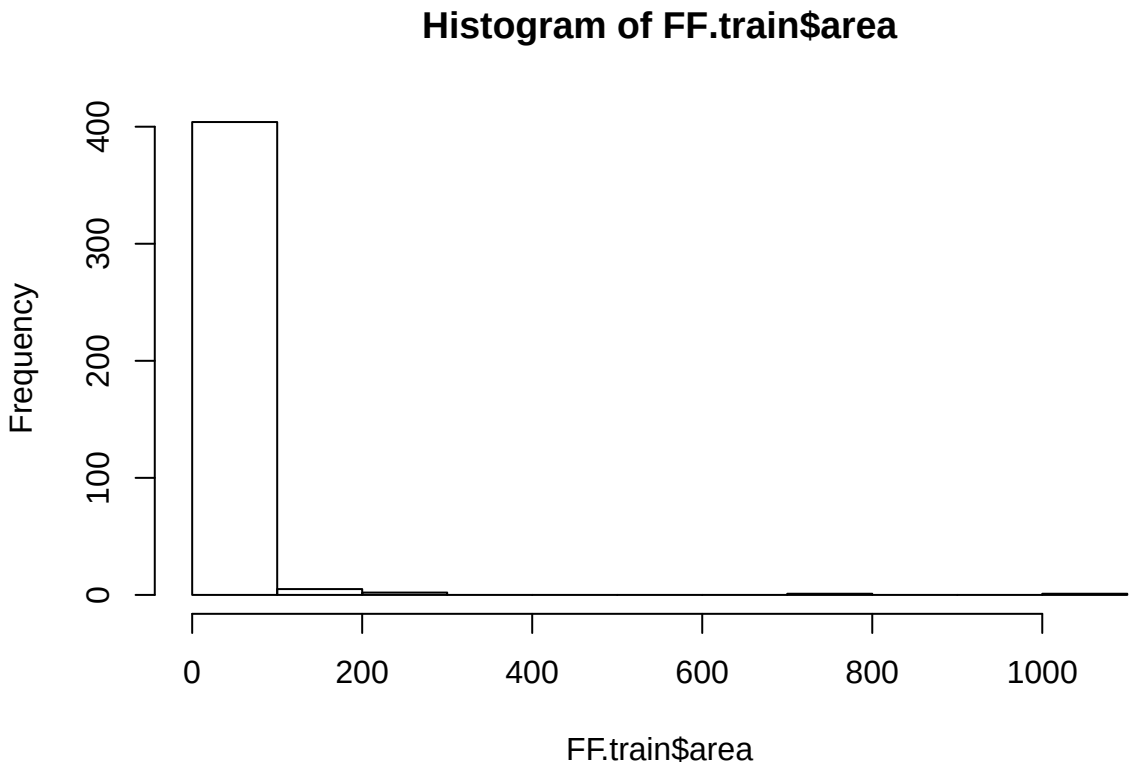
Para las redes neuronales el problema que vamos arrastrando es aún mayor. Es cierto que ofrece resultados ligeramente mejores, pero por contra, la extensión predicha por el modelo es prácticamente similar para todos los puntos, con lo que no nos resulta de utilidad.

Primer intento de mejora: Transformación con logaritmo

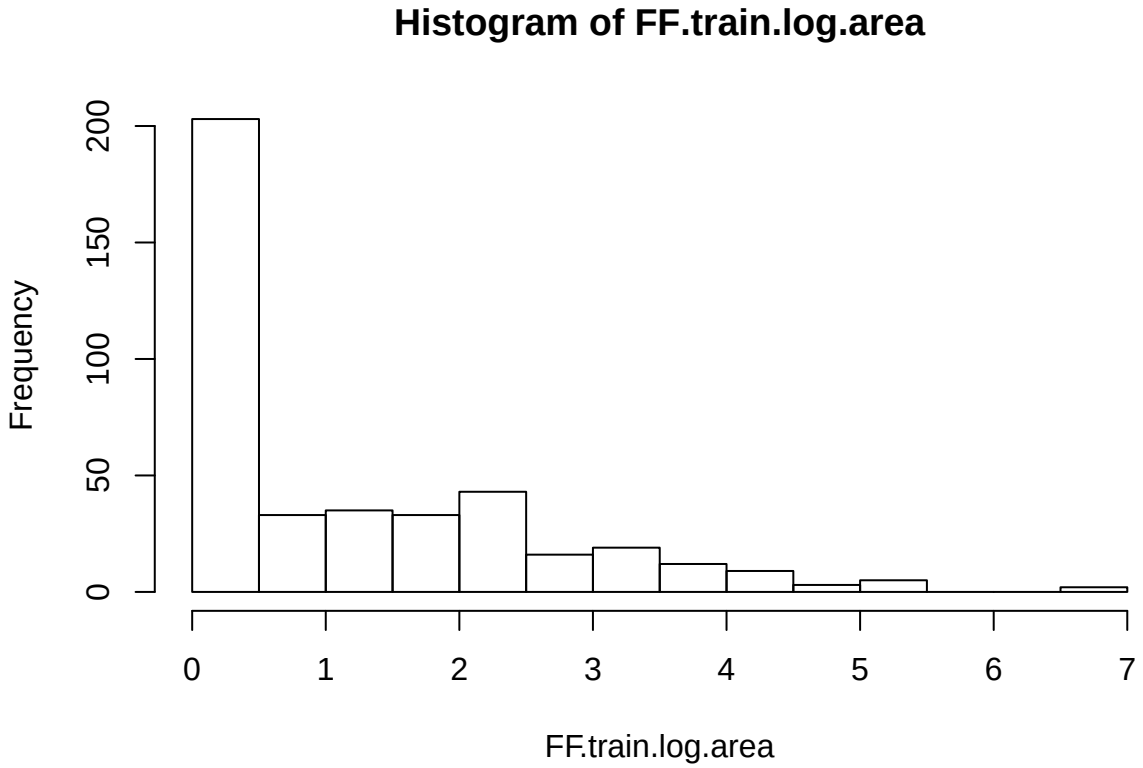
Los resultados obtenidos hasta ahora son demasiado pobres, y tienden a concentrar las predicciones, dando valores muy pequeños a incendios de una gran extensión y sobreestimando las extensiones de la mayoría de incendios, que son menores a una hectárea. Esto no es una peculiaridad de esta base de datos concreta, sino que, como es de esperar, los grandes incendios son mucho menos numerosos que los pequeños incendios.
Una transformación habitual en este tipo de distribuciones es $log(x + 1)$. De esta manera reducimos la asimetría de la distribución.

```
FF.train.log.area <- log(FF.train$area+1)
FF.test.log.area  <- log(FF.test$area+1)

hist(FF.train$area)
```




```
hist(FF.train.log.area)
```




```
FF.train.log <- data.frame(cbind(FF.train.data, FF.train.log.area))
names(FF.train.log)[13] <- "log.area"
```

Repetimos ahora los experimento, pero usando como entrada el logaritmo de las áreas.

Regresión lineal con *weight decay*

```
cv.wdecay <- cv.glmnet(FF.train.data, FF.train.log.area,
                       family = "gaussian", alpha=0)
lambda.min <- cv.wdecay$lambda.min
ridge.mod = glmnet(FF.train.data, FF.train.log.area, alpha=0,
                   lambda=lambda.min)
```

```
ridge.pred.log = exp(predict(ridge.mod,
                             s=lambda.min,
                             newx=FF.test.data))-1

MSE.ridge.log <- mean((ridge.pred.log - FF.test$area)^2)
RMSE.ridge.log <- sqrt(mean((ridge.pred.log - FF.test$area)^2))

abs.error.ridge.log <- abs(ridge.pred.log - FF.test$area)
abs.error.ridge.log <- abs.error.ridge.log[order(abs.error.ridge.log)]
MAD.ridge.log <- mean(abs.error.ridge.log)

cat("MSE", MSE.ridge.log)
```

```
## MSE 1216.954
```

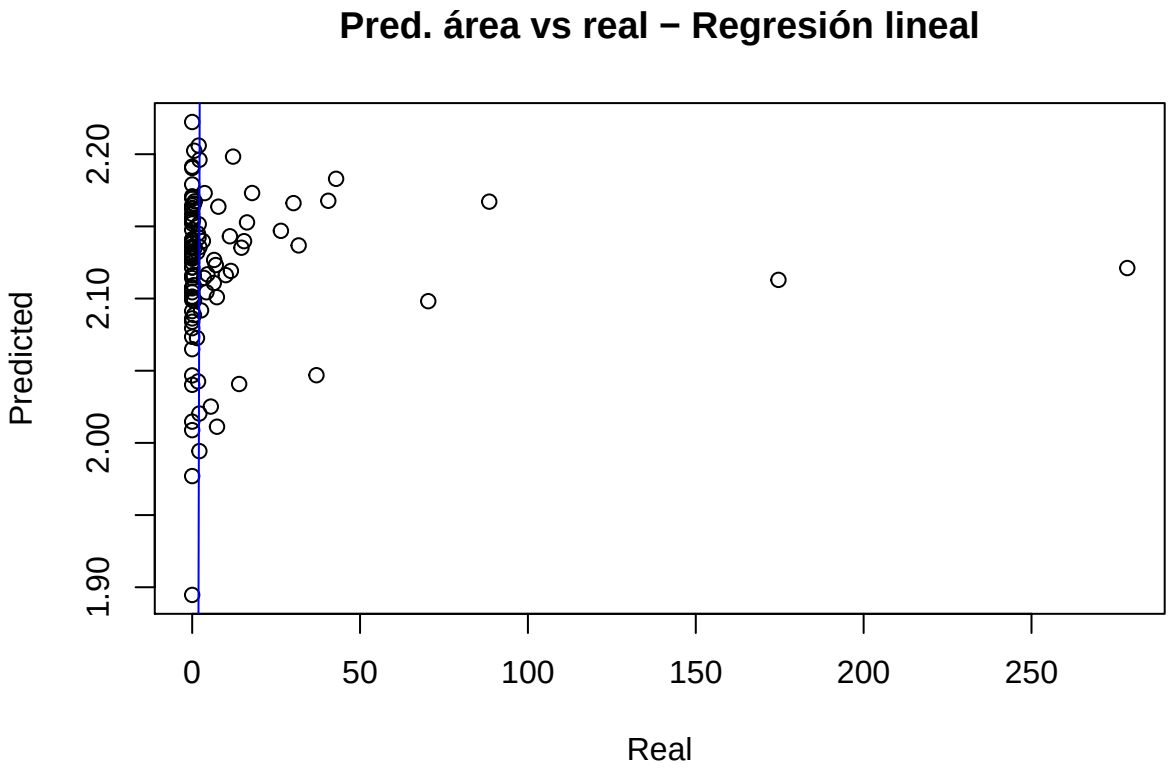
```
cat("RMSE", RMSE.ridge.log)
```

```
## RMSE 34.88486
```

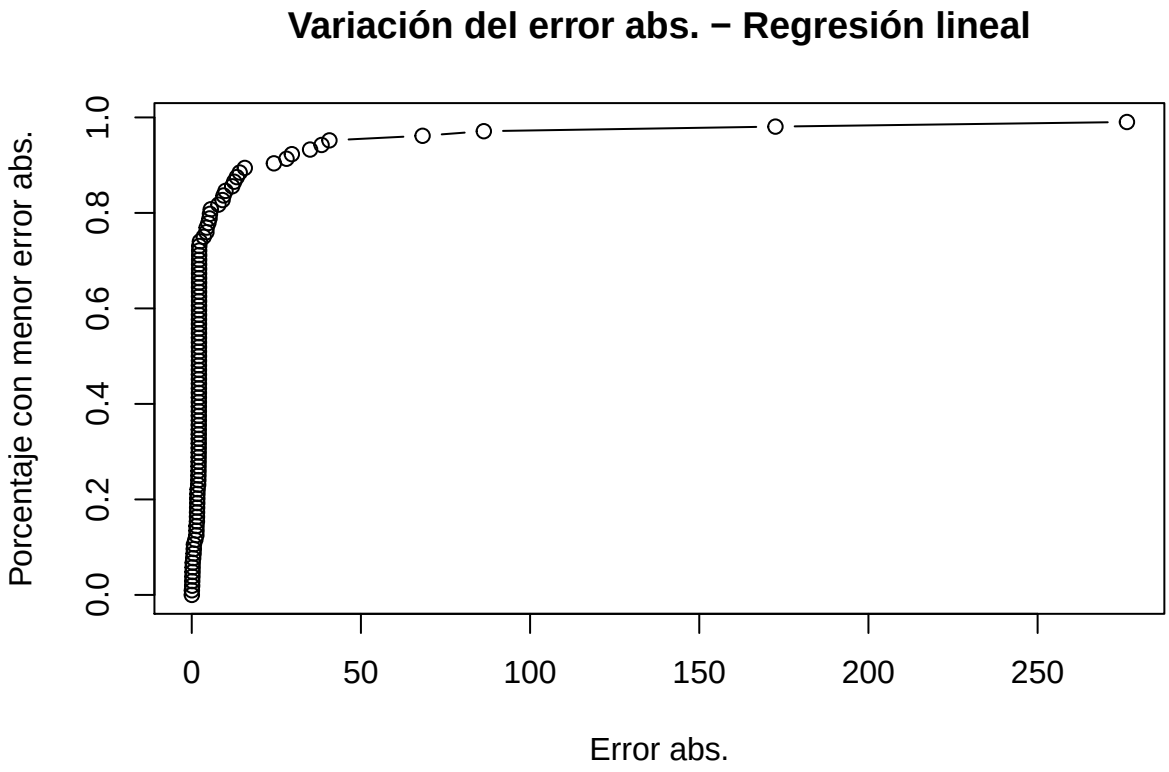
```
cat("MAD", MAD.ridge.log)
```

```
## MAD 10.31758
```

```
plot(FF.test$area, ridge.pred.log,
     main="Pred. área vs real - Regresión lineal",
     xlab = "Real", ylab = "Predicted")
abline(0 ,1, col = 4)
```




```
plot(abs.error.ridge.log,
     sapply(1:length(FF.test$area),
            function(i){
              sum(abs.error.ridge.log < abs.error.ridge.log[i])/
                length(FF.test$area)
            }),
     type="b", main="Variación del error abs. - Regresión lineal",
     xlab = "Error abs.", ylab = "Porcentaje con menor error abs.")
```



Vemos que ahora hemos introducido el problema en la regresión, donde los valores cubrían un mayor rango.

Random Forest

```
library(randomForest)
```

```
cv.rf <- rfcv(FF.train.data, FF.train.log.area)
mtray.min <- unname(which.min(cv.rf$error.cv))
```

```
k <- 5
folds <- kfold(FF.train, k = k)

MSE.ntree <- sapply( seq(100, 700, by=10),
        function(x){
          MSE <- 0
          for( i in 1:k ){
            rf.model <- randomForest(log.area ~ ., data=FF.train.log,
                                     subset=which(folds!=i),
                                     importance = TRUE, mtray=mtray.min,
                                     ntree=x)
            pred.area.rf <- predict( rf.model,
                                newdata=FF.train.data[which(folds==i),])
             MSE <- MSE + mean((pred.area.rf -
                                  FF.train.log.area[which(folds==i)])^2)
          }

          return(MSE/k)
        }
      )
```

```
best.ntree <- seq(100, 700, by=10)[which.min(MSE.ntree)]
```

```
rf.model <- randomForest(log.area ~ ., data=FF.train.log,
                         importance = TRUE, mtray=mtray.min,
                         ntree=best.ntree)
pred.area.rf.log <- exp(predict( rf.model, newdata=FF.test.data))-1

MSE.rf.log <- mean((pred.area.rf.log - FF.test$area)^2)
RMSE.rf.log <- sqrt(mean((pred.area.rf.log - FF.test$area)^2))

abs.error.rf.log <- abs(pred.area.rf.log - FF.test$area)
abs.error.rf.log <- abs.error.rf.log[order(abs.error.rf.log)]
MAD.rf.log <- mean(abs.error.rf.log)

cat("MSE", MSE.rf.log)
```

```
## MSE 1230.25
```

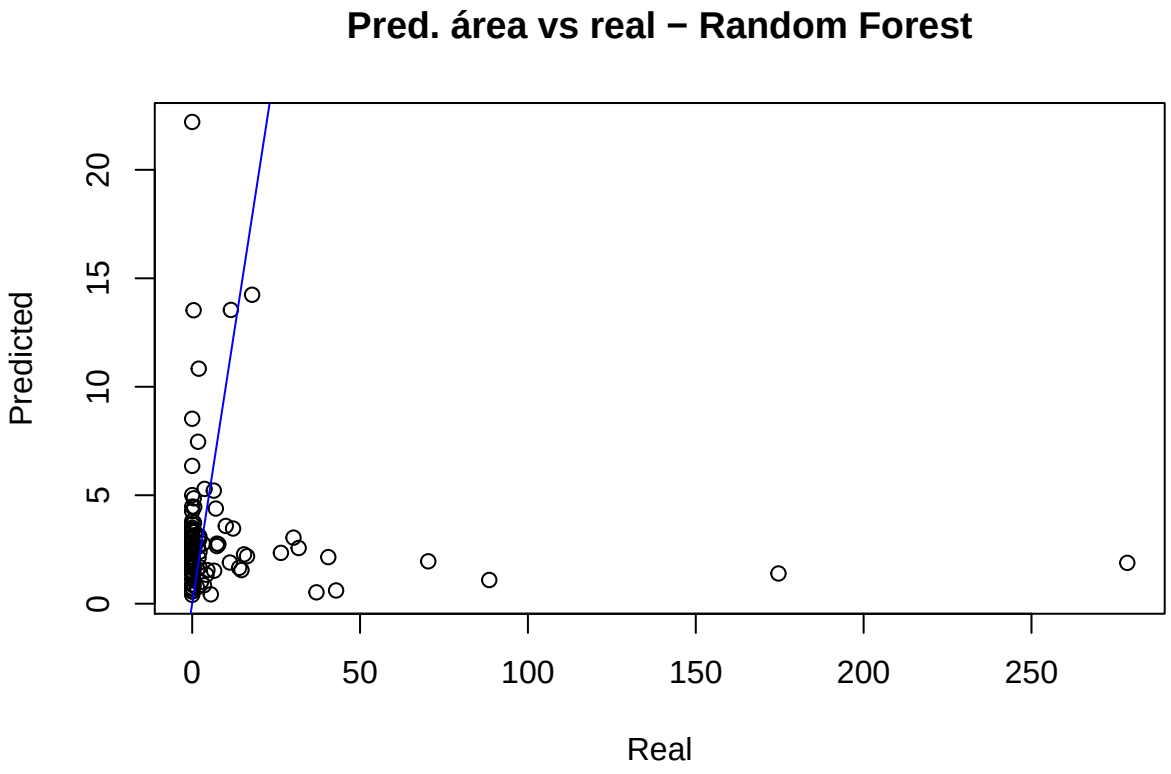
```
cat("RMSE", RMSE.rf.log)
```

```
## RMSE 35.07492
```

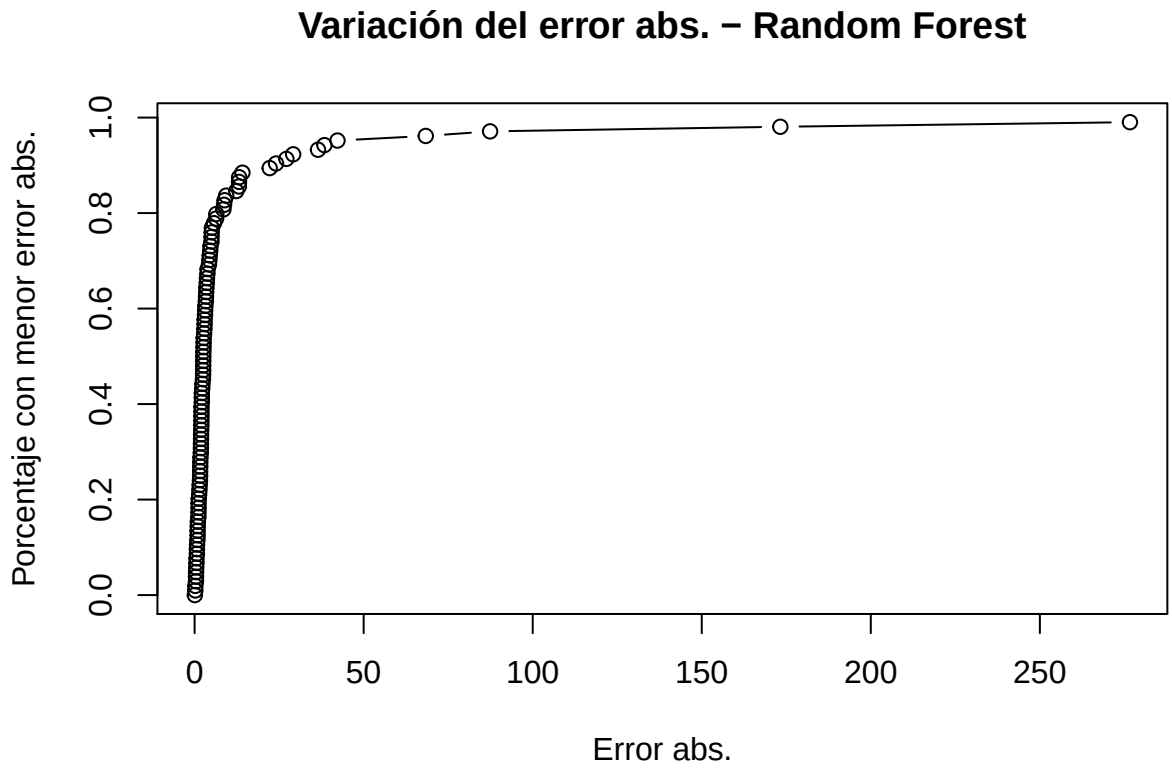
```
cat("MAD", MAD.rf.log)
```

```
## MAD 10.8276
```

```
plot(FF.test$area, pred.area.rf.log,
     main="Pred. área vs real - Random Forest",
     xlab = "Real", ylab = "Predicted")
abline(0 ,1, col = 4)
```



```r
plot(abs.error.rf.log,
     sapply(1:length(FF.test$area),
            function(i){
              sum(abs.error.rf.log < abs.error.rf.log[i])/
                length(FF.test$area)
            }),
     type="b", main="Variación del error abs. - Random Forest",
     xlab = "Error abs.", ylab = "Porcentaje con menor error abs.")
```



Función de base radial

```
## [1] 1
## [1] 11
## [1] 21
## [1] 31
## [1] 41
## [1] 51
## [1] 61
## [1] 71
## [1] 81
## [1] 91
```

```r
pred.area.radial.log <- exp(apply(FF.test.data, 1,
                          function(x) g(x, FF.train.data,
                                        FF.train.log.area, best.r)))-1

MSE.radial.log <- mean((pred.area.radial.log - FF.test$area)^2)
RMSE.radial.log <- sqrt(mean((pred.area.radial.log - FF.test$area)^2))
```

```
abs.error.radial.log <- abs(pred.area.radial.log - FF.test$area)
abs.error.radial.log <- abs.error.radial.log[order(abs.error.radial.log)]
MAD.radial.log <- mean(abs.error.radial.log)

cat("MSE", MSE.radial.log)
```

```
## MSE 1215.646
```

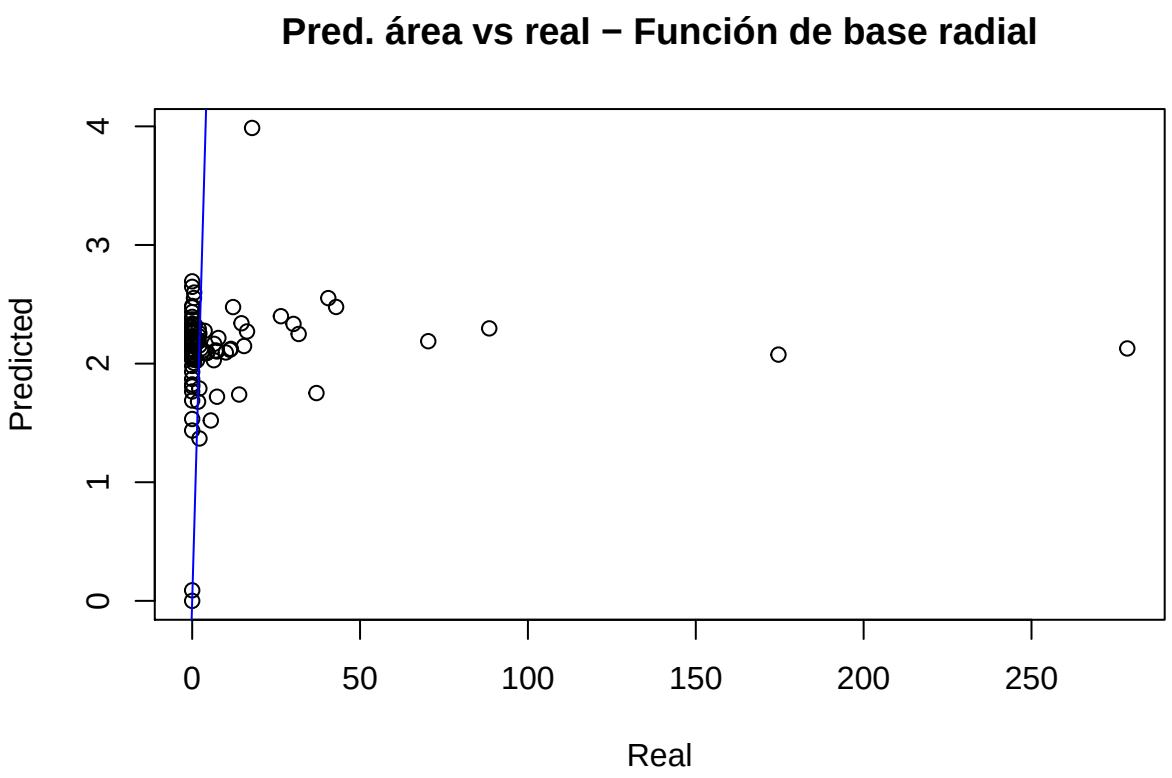
```
cat("RMSE", RMSE.radial.log)
```

```
## RMSE 34.86612
```

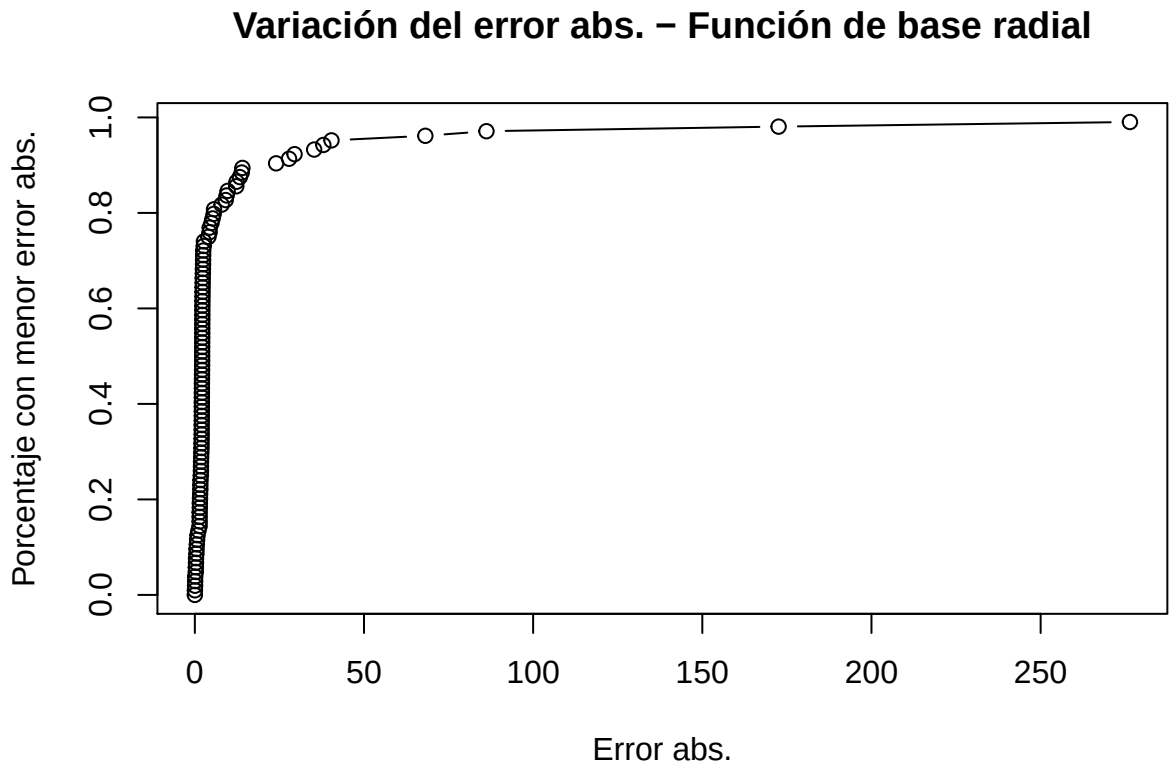
```
cat("MAD", MAD.radial.log)
```

```
## MAD 10.28518
```

```
plot(FF.test$area, pred.area.radial.log,
     main="Pred. área vs real - Función de base radial",
     xlab = "Real", ylab = "Predicted")
abline(0 ,1, col = 4)
```




```
plot(abs.error.radial.log,
     sapply(1:length(FF.test$area),
            function(i){
              sum(abs.error.radial.log < abs.error.radial.log[i])/
                length(FF.test$area)
            }),
     type="b", main="Variación del error abs. - Función de base radial",
     xlab = "Error abs.", ylab = "Porcentaje con menor error abs.")
```



Como vemos en los métodos realizados, no se obtienen mejoras, por lo que descartamos la transformación.

Segundo intento de mejora: clasificación según la magnitud de los incendios.

Los resultados obtenidos muestran una clara tendencia a obviar los incendios importantes y afirmar que todos los incendios tendrán una extensión pequeña. Esto significa que realmente no estamos haciendo una labor de regresión, sino que estamos afirmando directamente que la extensión quemada es cercana a 0. Por tanto, trataremos de realizar una clasificación previa entre incendios menores (aquellos que se corresponderán con el valor 0) y los que superan los $100m^2$, a los cuales tendrá sentido aplicar la regresión.

Creamos categorías para la clasificación de los incendios. Consideraremos incendios pequeños aquellos cuya extensión quemada sea menor a 10 ha, incendios medios los que estén entre 10 y 50 ha e incendios graves los que superen las 50 ha.

```
FF.train.categ <- FF.train$area > 0
categ <- FF.train.categ
FF.train.with.categ <- data.frame(cbind(FF.train.data,categ))

FF.test.categ <- FF.test$area > 0
categ <- FF.test.categ
FF.test.with.categ <- data.frame(cbind(FF.test.data,categ))
```

Trataremos de realizar la clasificación con *random forest*. La razón de usar este modelo es que intentamos encontrar aquellos valores con una gran extensión quemada, y al ser los valores similares a otros con áreas quemadas nulas o muy pequeñas, no nos serviría utilizar modelos como el perceptron.

```
cv.rf <- rfcv(FF.train.data, as.factor(FF.train.categ))
mtray.min <- unname(which.min(cv.rf$error.cv))
```

```
k <- 5
folds <- kfold(FF.train, k = k)

error.ntree <- sapply( seq(100, 700, by=10),
        function(x){
          error <- 0
          for( i in 1:k ){
            rf.model <- randomForest(as.factor(categ) ~ .,
                                     data=FF.train.with.categ,
                                     subset=which(folds!=i),
                                     importance = TRUE, mtray=mtray.min,
                                     ntree=x)
            pred.area.rf <- predict( rf.model,
                                     newdata=FF.train[which(folds==i),])
            error <- error + sum(pred.area.rf !=
                                   FF.train.categ[which(folds==i)])
          }

          return(error/k)
        }
      )
best.ntree <- seq(100, 700, by=10)[which.min(error.ntree)]
rf.model <- randomForest(as.factor(categ) ~ ., data=FF.train.with.categ,
                                     importance = TRUE, mtray=mtray.min,
                                     ntree=best.ntree)
pred.categ.rf <- predict( rf.model, newdata=FF.test.data)
```

Mostramos la matriz de confusión:

```
pred.categ.rf <- pred.categ.rf==1
confusionMatrix(pred.categ.rf, FF.test.categ)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE    25   20
##      TRUE     25   34
##
##                Accuracy : 0.5673
##                  95% CI : (0.4665, 0.6641)
##     No Information Rate : 0.5192
##     P-Value [Acc > NIR] : 0.1887
##
##                   Kappa : 0.1301
##  Mcnemar's Test P-Value : 0.5510
##
##             Sensitivity : 0.5000
##             Specificity : 0.6296
##          Pos Pred Value : 0.5556
##          Neg Pred Value : 0.5763
##              Prevalence : 0.4808
##          Detection Rate : 0.2404
##    Detection Prevalence : 0.4327
##       Balanced Accuracy : 0.5648
```

```
##
##        'Positive' Class : FALSE
##
```

Usamos las funciones de base radial para realizar la regresión a aquellos valores que hemos predicho que serán mayores que 0, utilizando únicamente aquellos valores de entrenamiento que lo son.

```
pred.area.clas <- sapply(1:length(pred.categ.rf),
                         function(i){
                           if(pred.categ.rf[i]==1){
                             return(g(FF.test.data[i,],
                                      FF.train.data[FF.train.categ,],
                                      FF.train$area[FF.train.categ],
                                      best.r))
                           }
                           else{
                             return( 0 )
                           }
                         })
```

```
MSE.radial.clas <- mean((pred.area.clas - FF.test$area)^2)
RMSE.radial.clas <- sqrt(mean((pred.area.clas - FF.test$area)^2))

abs.error.clas <- abs(pred.area.clas - FF.test$area)
abs.error.clas <- abs.error.clas[order(abs.error.clas)]
MAD.radial.clas <- mean(abs.error.clas)

cat("MSE", MSE.radial.clas)
```

```
## MSE 1526.719
```

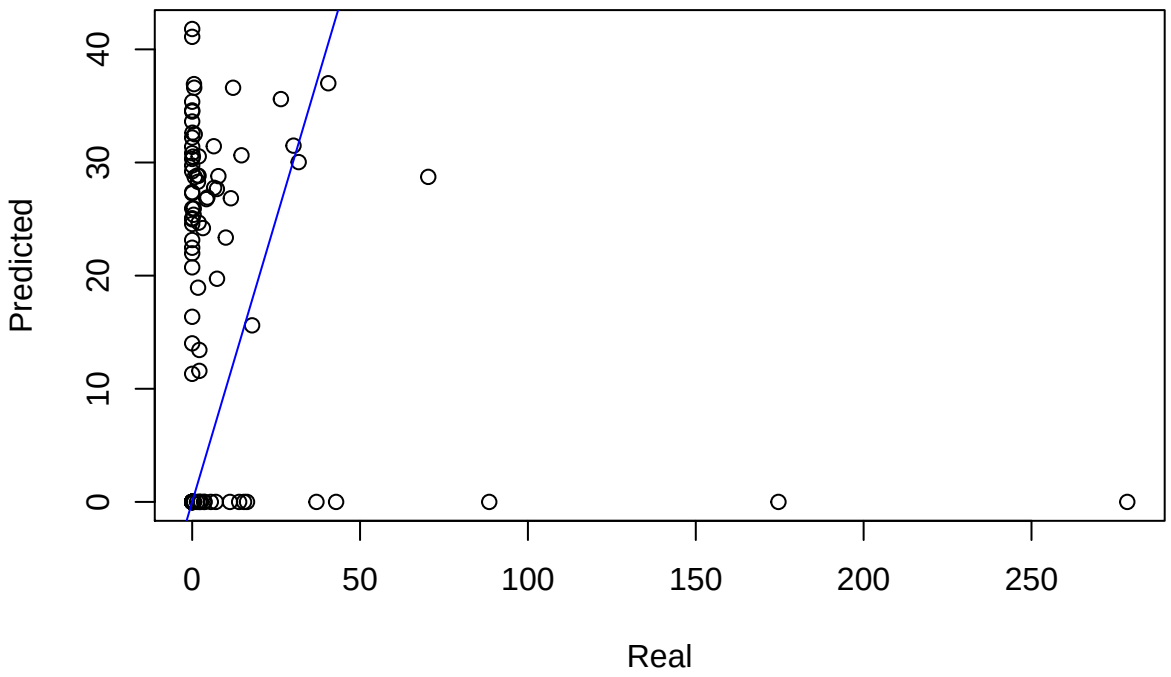
```
cat("RMSE", RMSE.radial.clas)
```

```
## RMSE 39.07325
```

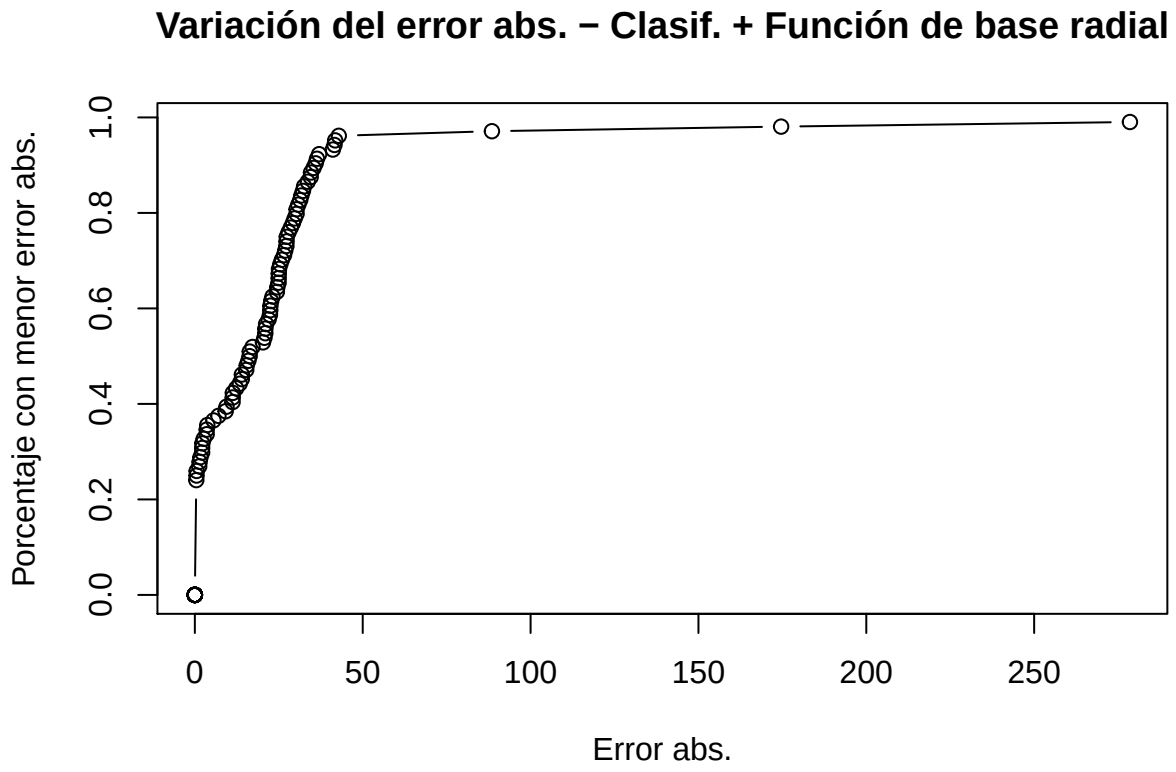
```
cat("MAD", MAD.radial.clas)
```

```
## MAD 20.26836
```

```
plot(FF.test$area, pred.area.clas,
     main="Pred. área vs real - Clasif. + Función de base radial",
     xlab = "Real", ylab = "Predicted")
abline(0 ,1, col = 4)
```




```
plot(abs.error.clas,
     sapply(1:length(FF.test$area),
            function(i){
              sum(abs.error.clas < abs.error.clas[i])/
                length(FF.test$area)
            }),
     type="b", main="Variación del error abs. - Clasif. + Función de base radial",
     xlab = "Error abs.", ylab = "Porcentaje con menor error abs.")
```



Debido a que la clasificación tampoco es buena, los resultados son peores, con lo que descartaríamos realizar esta clasificación previa.

Comparativa:

Mostramos la matriz con los resultados obtenidos.

```
##                                MSE     RMSE      MAD
## ridge                     1171.793 34.23146 16.81005
## ridge log transf.         1216.954 34.88486 10.31758
## RF                        1263.146 35.54077 21.78636
## RF log transf.            1230.250 35.07492 10.82760
## FBR                       1165.066 34.13306 16.96699
## FBR log transf.           1215.646 34.86612 10.28518
## NN                        2029.647 45.05160 15.55122
## Clasificación + regresión 1526.719 39.07325 20.26836
```

6. Referencias

- Estudio de FWI
- A Data Mining Approach to Predict Forest Fires using Meteorological Data