Regresión cuantílica: Quantile Regression Forest

Joaquín Amat Rodrigo j.amatrodrigo@gmail.com

Febrero, 2020

Tabla de contenidos

Versión PDF: Github

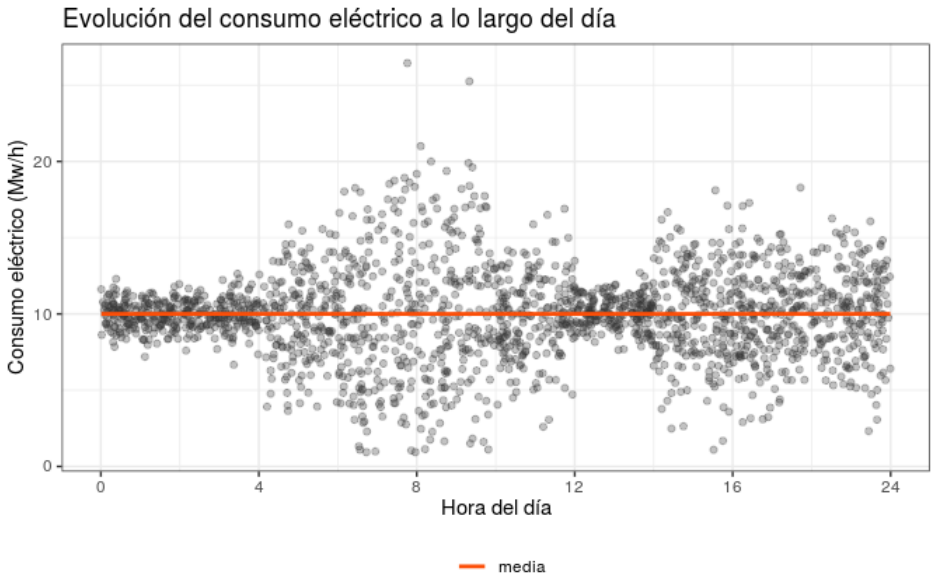
Más sobre ciencia de datos: cienciadedatos.net o joaquinamatrodrigo.github.io

Introducción

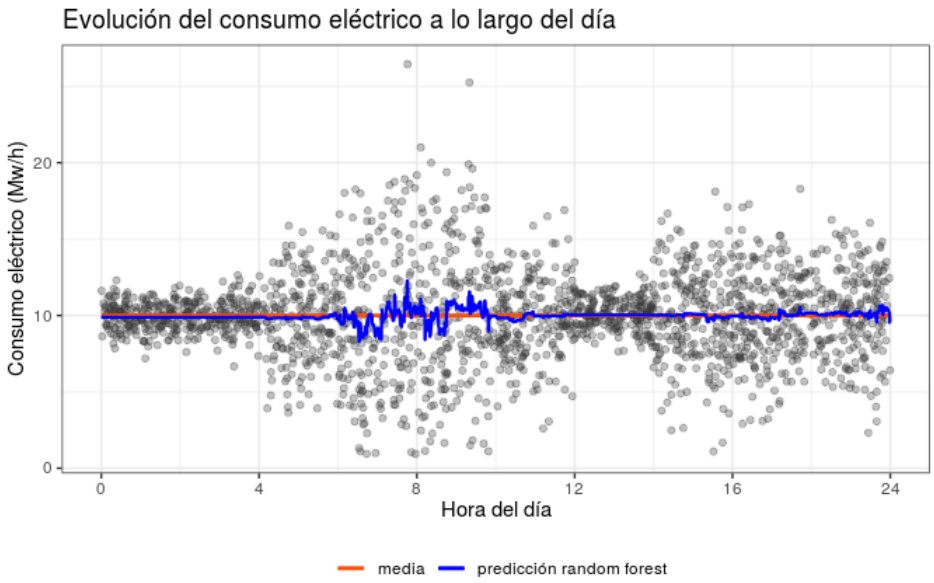
La predicción de una variable continua Y en función de uno o varios predictores X es un problema de aprendizaje supervisado que puede resolverse con múltiples métodos de machine learning (SVM, Random Forest, Boosting), todos ellos capaces de inferir de algún modo la relación entre X e Y.

El objetivo de la mayoría de estos algoritmos consiste en predecir el valor promedio de Y en función del valor de X, $E(Y|X = x)$. Aunque conocer la media condicional es de utilidad, este resultado ignora otras características de la distribución de Y que pueden ser claves a la hora de tomar decisiones, por ejemplo, su dispersión.

Véase el siguiente ejemplo simulado (y muy simplificado) sobre la evolución del consumo eléctrico de todas las casas de una ciudad en función de la hora del día. Ver $Anexo^1$ con el código empleado para la simulación.



La media del consumo eléctrico es la misma durante todo el día, $\overline{consumo} = 10Mw/h$, sin embargo, su dispersión no es constante (heterocedasticidad). Véase el resultado de predecir el consumo medio en función de la hora del día con un modelo *random forest*.

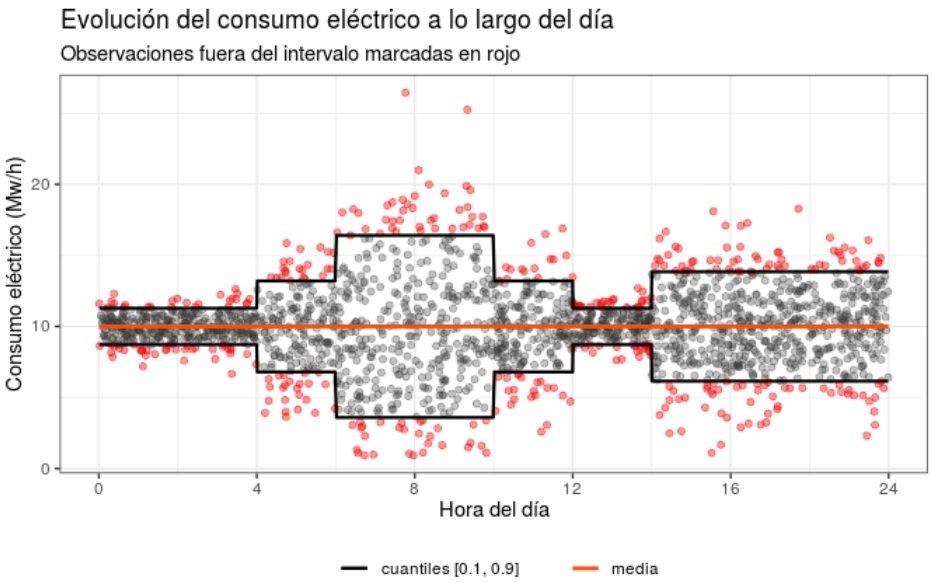


El valor predicho es muy próximo a la media real, es decir, el modelo es bueno prediciendo el consumo medio esperado. Ahora, imagínese que la compañía encargada de suministrar la electricidad debe de ser capaz de provisionar, en un momento dado, con hasta un 50% de electricidad extra respecto al promedio. Esto significa un máximo de 15 Mw/h. Estar preparado para suministrar este extra de energía implica gastos de personal y maquinaría, por lo que la compañía se pregunta si es necesario estar preparado para producir tal cantidad durante todo el día, o si, por lo contrario, podría evitarse durante algunas horas, ahorrando así gastos.

Un modelo que predice únicamente el promedio no permite responder a esta pregunta, ya que tanto para las 2h de la mañana como para las 8h, el consumo promedio predicho es en torno a 10 Mw/h, sin embargo, la probabilidad de que se alcancen consumos de 15 Mw/h a las 2h es prácticamente nula mientras que esto ocurra a las 8h sí es razonable.

Una forma de describir la dispersión de una variable es el uso de cuantiles. El cuantil de orden τ $(0 < \tau < 1)$ de una distribución es el valor de la variable X que marca un corte tal que, una proporción τ de valores de la población, es menor o igual que dicho valor. Por ejemplo, el cuantil de orden 0.36 deja un 36% de valores por debajo y el cuantil de orden 0.50 el 50% (se corresponde con la mediana de la distribución).

Dado que los datos se han simulado empleando distribuciones normales, se conoce el valor de los cuantiles teóricos para cada X. Se muestra de nuevo el mismo gráfico pero esta vez añadiendo los cuantiles 0.1 y 0.9.



Si como resultado del modelo, además de la predicción de la media, se predice también el valor de los cuantiles, se dispone de una caracterización mayor de la distribución de la variable respuesta Y, y con ello se puede responder a más preguntas. Por ejemplo, en el caso de la energía, se tendría cierta seguridad al decir que, durante los intervalos de 0h a 4h y de 12h a 14h, es poco probable que se alcancen consumos de 15 Mw/h.

Otros casos en los que la regresión cuantílica puede emplearse son:

- Identificación de regiones en las que la variable respuesta Y tiene mayor dispersión en torno a su media.
- Entrenar modelos que predicen la mediana (cuantil 0.5) en lugar de la media. Estos modelos son más robustos frente a *outliers*.
- Detectar anomalías, identificando aquellas observaciones que están fuera de un determinado intervalo cuantílico.

En los siguientes apartados se describe el algoritmo *quantile regression forest*, una adaptación de *random forest* capaz de aprender la distribución de cuantiles.

Quantile Regression Forest

Algoritmo

Quantile regression forest es una adaptación del algoritmo de Regression Random Forest que permite predecir, en lugar de la media, cualquiera de los cuantiles.

Un modelo *random forest* está formado por un conjunto de árboles de regresión individuales, cada uno ajustado empleando una muestra bootstrapping de los datos de entrenamiento. Una vez entrenado, la predicción de una nueva observación se obtiene promediando las predicciones de todos los árboles individuales que forman el modelo. El algoritmo de *quantile regression forest* sigue exactamente la misma estrategia para crear el modelo, la diferencia reside en cómo se calculan las predicciones. Con el objetivo de mostrar esta adaptación, en los siguientes apartados se describe primero cómo predicen los árboles de regresión simple, a continuación cómo lo hace *random forest* y por último la modificación para conseguir predecir cuantiles.

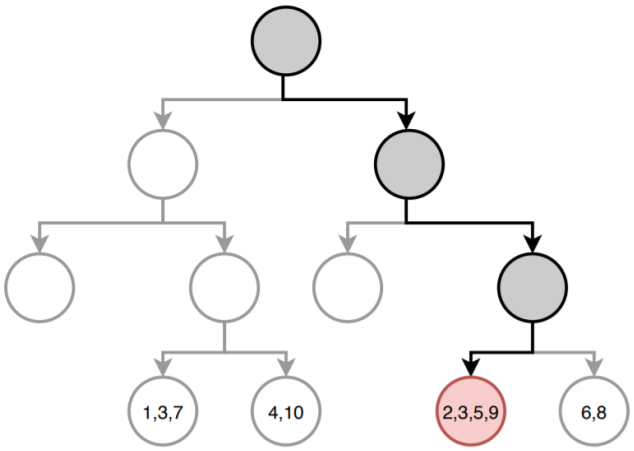
Predicción de un árbol de regresión

En el entrenamiento de un árbol de regresión simple, las observaciones se van distribuyendo por bifurcaciones (nodos) generando la estructura del árbol hasta alcanzar un nodo terminal. Cuando se quiere predecir una nueva observación, esta recorre el árbol acorde al valor de sus predictores hasta alcanzar uno de los nodos terminales. La predicción del árbol es la media de la variable respuesta de las observaciones de entrenamiento que están en el mismo nodo terminal.

Supóngase que se dispone de 10 observaciones, cada una con un valor de variable respuesta Y y unos predictores X.

id	1	2	3	4	5	6	7	8	9	10
Y	10	18	24	8	2	9	16	10	20	14
X	...	...	...	...	...	...	...	...	...	...

La siguiente imagen muestra cómo sería la predicción del árbol para una nueva observación. El camino hasta llegar al nodo final está resaltado. En cada nodo terminal se detalla el índice de las observaciones de entrenamiento que forman parte.



Predicción con un árbol de regresión: el camino hasta llegar al nodo final está resaltado. En cada nodo terminal se detalla el índice de las observaciones de entrenamiento que forman parte.

El valor predicho por el árbol es la media de la variable respuesta Y de las observaciones con id: 2, 3, 5, 9.

$$\hat{\mu} = \frac{18 + 24 + 2 + 20}{4} = 16$$

Aunque la anterior es la forma más común de obtener las predicciones de un árbol de regresión, existe otra aproximación. La predicción de un árbol de regresión puede verse como una variante de vecinos cercanos en la que, solo las observaciones que forman parte del mismo nodo terminal que la observación predicha, tienen influencia. Siguiendo esta aproximación, la predicción del árbol se define como la media ponderada de todas las observaciones de entrenamiento, donde el peso de cada observación depende únicamente de si forma parte o no del mismo nodo terminal.

$$\hat{\mu} = \sum_{i=1}^{n} \mathbf{w}_i \, Y_i$$

El valor de las posiciones del vector de pesos $\mathbf{w}$ es 1 para las observaciones que están en el mismo nodo y 0 para el resto. En este ejemplo sería:

$$\mathbf{w} = (0,1,1,0,1,0,0,0,1,0)$$

Para que la suma de todos los pesos sea 1, se dividen por el número total de observaciones en el nodo terminal seleccionado, en este caso 4.

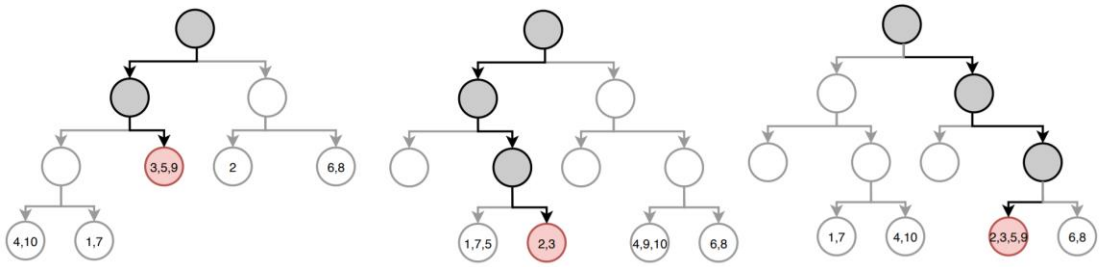
$$\mathbf{w} = (0, \frac{1}{4}, \frac{1}{4}, 0, \frac{1}{4}, 0,0,0, \frac{1}{4}, 0)$$

Así pues, el valor predicho es:

$$\hat{\mu} = (0 \text{ x } 10) + \left(\frac{1}{4} \text{ x } 18\right) + \left(\frac{1}{4} \text{ x } 24\right) + (0 \text{ x } 8) + \left(\frac{1}{4} \text{ x } 2\right) + (0 \text{ x } 9) + (0 \text{ x } 16) + (0 \text{ x } 10) + \left(\frac{1}{4} \text{ x } 20\right) + (0 \text{ x } 14) = 16$$

Predicción de random forest

La predicción de un modelo *random forest* es la media de las predicciones de todos los árboles que lo forman. Siguiendo la visión de vecinos cercanos planteada en el apartado anterior, esto equivale a la media ponderada de todas las observaciones, empleando como pesos **w** la media de los vectores de pesos de todos los árboles. Véase el siguiente ejemplo.



Predicción con random forest: en cada árbol, el camino hasta llegar al nodo final está resaltado. En cada nodo terminal se detalla el índice de las observaciones de entrenamiento que forman parte de él.

Acorde a la imagen anterior, el vector de pesos para cada uno de los tres árboles (de izquierda a derecha) es:

$$\mathbf{w}_{arbol_1} = (0,0,\frac{1}{3},0,\frac{1}{3},0,0,0,\frac{1}{3},0)$$

$$\mathbf{w}_{arbol_2} = (0,\frac{1}{2},\frac{1}{2},0,0,0,0,0,0,0)$$

$$\mathbf{w}_{arbol_3} = (0,\frac{1}{4},\frac{1}{4},0,\frac{1}{4},0,0,0,\frac{1}{4},0)$$

La media de todos los vectores de pesos es:

$$\bar{\mathbf{W}} = \frac{1}{3} + (\mathrm{W}_{arbol1} + \mathrm{W}_{arbol2} + \mathrm{W}_{arbol3}) = \left(0,\frac{1}{4},\frac{13}{36},0,\frac{7}{36},0,0,0,\frac{7}{36},0\right)$$

Una vez obtenido el vector de pesos promedio, se puede calcular la predicción con la media ponderada de todas las observaciones de entrenamiento:

$$\hat{\mu} = \sum_{i=1}^{n} \overline{\mathbf{w}}_i \, Y_i$$

$$\hat{\mu} = (0 \text{ x } 10) + \left(\frac{1}{4} \text{ x } 18\right) + \left(\frac{13}{36} \text{ x } 24\right) + (0 \text{ x } 8) + \left(\frac{1}{4} \text{ x } 2\right) + (0 \text{ x } 9) + (0 \text{ x } 16) + (0 \text{ x } 10) +$$

$$\left(\frac{1}{4} \text{ x } 20\right) + (0 \text{ x } 14) = 17.4$$

Adaptación para predecir cuantiles

La adaptación necesaria para predecir cualquier cuantil es la siguiente (ver detalles en Meinshausen, 2006). Si en lugar de emplear la media ponderada de todas las observaciones, se emplean únicamente aquellas para las que se cumple que $Y \leq y$, y se repite este cálculo para todos los valores observados y_i, se obtiene la estimación empírica de la función de distribución de Y en función de X.

$$\hat{F}\,(y|X = x) = P(Y \leq y|X = x) = \sum_{i=1}^{n} \mathbf{w}_i\,(x)\; 1_{\{Y_i \leq y\}}$$

Es decir, la probabilidad de que la variable respuesta Y tenga un valor menor o igual que y , dado un valor de los predictores X. Una vez obtenida la estimación de la distribución, se puede calcular cualquiera de sus cuantiles.

Una ventaja de este método frente a otras aproximaciones como por ejemplo *Dristributional Tregression Forest* o *GAMLSS,* es que es de tipo no paramétrico, por lo que no requiere asumir ningún tipo de distribución de la variable respuesta Y.

Ejemplo

En **R** existen varios paquetes que incorporan la adaptación de *random forest* para predecir cuantiles, dos de ellos son `quantregForest` y `ranger`. Ambos siguen el método propuesto por (Meinshausen, 2006), pero el segundo destaca por ser mucho más rápido.

Datos

Ver *Anexo*[1] para conocer más detalles de la simulación.

```
library(tidyverse)

# Simulación distribución no uniforme en el rango X
# ------------------------------------------------------------------------------
set.seed(123)
n <- 2000
x <- runif(n)
y <- rnorm(
        n,
        mean = 10,
        sd = 1 + 1.5*(0.2 < x & x < 0.3) + 4*(0.3 < x & x < 0.5) +
             1.5*(0.5 < x & x < 0.6) + 2*(x > 0.7)
    )
# Cálculo del cuantil 0.1 y 0.9 para cada posición de x simulada.
cuantil_01 <- qnorm(
                p = 0.1,
                mean = 10,
                sd = 1 + 1.5*(0.2 < x & x < 0.3) + 4*(0.3 < x & x < 0.5) +
                     1.5*(0.5 < x & x < 0.6) + 2*(x > 0.7)
              )
cuantil_90 <- qnorm(
                p = 0.9,
                mean = 10,
                sd = 1 + 1.5*(0.2 < x & x < 0.3) + 4*(0.3 < x & x < 0.5) +
                     1.5*(0.5 < x & x < 0.6) + 2*(x > 0.7)
              )
datos <- data.frame(y, x, cuantil_01, cuantil_90)
# No puede haber consumos negativos
datos <- datos %>%
         filter(y >= 0)
datos <- datos %>%
         mutate(dentro_intervalo = ifelse(
                                     y > cuantil_01 & y < cuantil_90,
                                     TRUE,
                                     FALSE
                                   )
        )
```

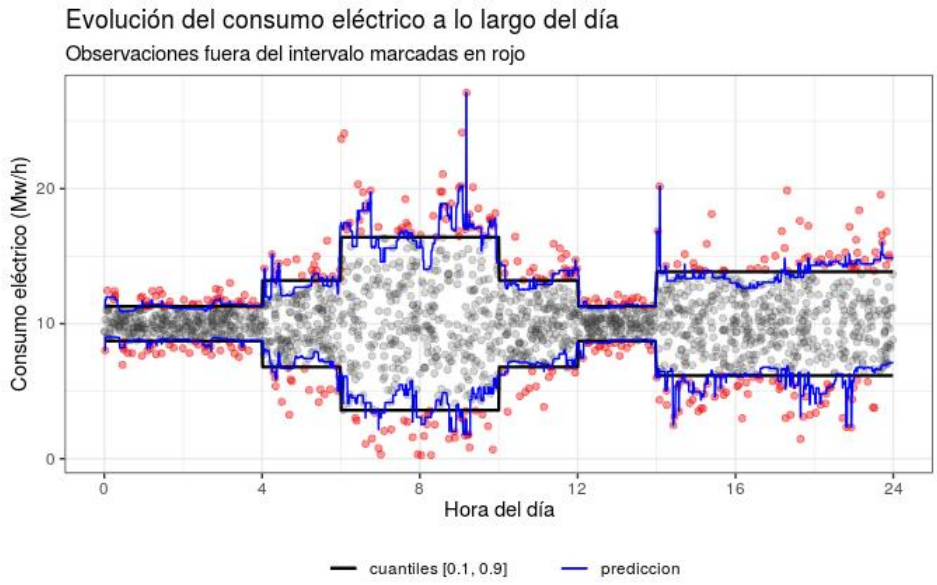
Modelo

```
library(ranger)

# Para predecir cuantiles se tiene que indicar quantreg = TRUE
modelo_qranger <- ranger(
                    x = datos %>% select(x),
                    y = datos$y,
                    num.trees = 5000,
                    min.node.size = 100,
                    quantreg = TRUE,
                    # Se emplean todos los cores disponibles -1
                    num.threads = future::availableCores() - 1,
                    seed = 123
                  )

modelo_qranger
```

```
## Ranger result
##
## Call:
##  ranger(x = datos %>% select(x), y = datos$y, num.trees = 5000,
min.node.size = 100, quantreg = TRUE, num.threads = future::availableCores() -
1, seed = 123)
##
## Type:                             Regression
## Number of trees:                  5000
## Sample size:                      1994
## Number of independent variables:  1
## Mtry:                             1
## Target node size:                 100
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       9.4406
## R squared (OOB):                  -0.05156388
```

```
# Se predice todo el rango de X para representar los cuantiles
grid_predictor <- seq(0, 1, length.out = 5000)

predicciones_cuantiles <- predict(
                            modelo_qranger,
                            data = data.frame(x = grid_predictor),
                            type = "quantiles",
                            quantiles = c(0.1, 0.9),
                            # Se emplean todos los cores disponibles -1
                            num.threads = future::availableCores() - 1
                          )
```

```r
predicciones_cuantiles <-  as.data.frame(predicciones_cuantiles$predictions)
predicciones_cuantiles <- bind_cols(
                           data.frame(x = grid_predictor),
                           predicciones_cuantiles
                          )

p <- ggplot() +
      geom_point(
        data = datos %>% filter(dentro_intervalo == TRUE),
        aes(x = x, y = y),
        alpha = 0.2,
        color = "gray20") +
      geom_point(
        data = datos %>% filter(dentro_intervalo == FALSE),
        aes(x = x, y = y),
        alpha = 0.4,
        color = "red2") +
      geom_line(aes(x = x, y = cuantil_01, linetype = "cuantiles [0.1, 0.9]"),
                size = 0.8) +
      geom_line(aes(x = x, y = cuantil_90, linetype = "cuantiles [0.1, 0.9]"),
                size = 0.8) +
      geom_line(
        data = predicciones_cuantiles,
        aes(x = x, y = `quantile= 0.1`, color = "prediccion")) +
      geom_line(
        data = predicciones_cuantiles,
        aes(x = x, y = `quantile= 0.9`, color = "prediccion")) +
      scale_color_manual(
        name = "",
        breaks = c("prediccion"),
        values = c("prediccion" = "blue")) +
      scale_linetype_manual(
        name = "",
        breaks = c("cuantiles [0.1, 0.9]", "cuantiles [0.1, 0.9]"),
        values = c("cuantiles [0.1, 0.9]" = "solid")) +
      labs(
        title = "Evolución del consumo eléctrico a lo largo del día",
        subtitle =  "Observaciones fuera del intervalo marcadas en rojo",
        x = "Hora del día",
        y = "Consumo eléctrico (Mw/h)") +
      theme_bw() +
      theme(legend.position = "bottom")

p <- p +
     scale_x_continuous(breaks = seq(0,1,length.out = 6),
                        labels = c(0, 4, 8, 12, 16, 24))
p
```



Detección de anomalías (Outliers)

Conocer los cuantiles condicionales de la variable respuesta Y permite identificar observaciones que se alejan atípicamente por encima o por debajo del valor esperado, dado sus predictores X. Véase el siguiente ejemplo en el que se trata de identificar precios anómalos de diamantes.

Datos

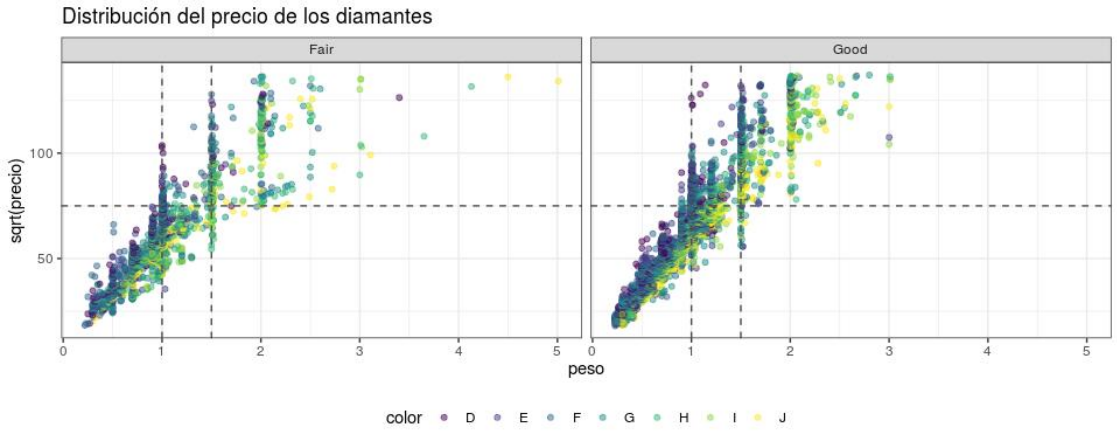
Se emplea el set de datos `diamods` del paquete disponible en `ggplot2`.

```
data(diamonds)
diamonds <- diamonds %>%
            filter(cut %in% c("Fair", "Good")) %>%
            mutate(price = sqrt(price))
```

El siguiente gráfico muestra la distribución del precio de los diamantes en función de su peso, calidad del corte y color.

```
ggplot(data = diamonds, aes(x = carat, y = price, color = color)) +
geom_point(alpha = 0.5) +
geom_vline(xintercept = 1, linetype = "dashed",  color = "gray30") +
geom_vline(xintercept = 1.5, linetype = "dashed",  color = "gray30") +
geom_hline(yintercept = 75, linetype = "dashed", color = "gray30") +
facet_wrap(facets = vars(cut))  +
```

```
labs(
  title = "Distribución del precio de los diamantes",
  x = "peso",
  y = "sqrt(precio)",
  color = "color") +
guides(col = guide_legend(nrow = 1)) +
theme_bw() +
theme(legend.position = "bottom")
```

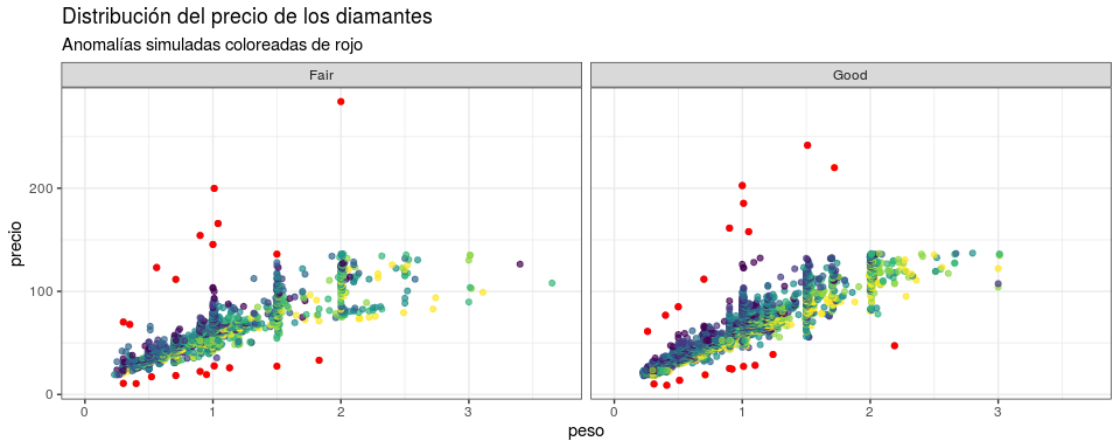


Puede verse que, dependiendo del peso, calidad del corte y color, el precio varía notablemente. Por ejemplo, apenas hay ningún diamante con un peso entre 1 y 1.5 unidades, de color *J* que tenga un precio de más de 7000$, pero sí los hay de este peso y precio con otros colores.

Se añaden varias anomalías simuladas en cada uno de los grupos.

```
# Para distribuir las anomalías se simula una por cada intervalo cuantílico.
diamonds <- diamonds %>%
            mutate(
              percentil_carat = cut(
                                 diamonds$carat,
                                 breaks = quantile(diamonds$carat, seq(0,1,by=0.1))
                                )
            ) %>% drop_na()

set.seed(1234)
id_anomalias_1 <- diamonds %>%
                  mutate(row_id = row_number()) %>%
                  group_by(percentil_carat, cut) %>%
                  sample_n(size = 1) %>%
                  pull(row_id)
id_anomalias_2 <- diamonds %>%
                  mutate(row_id = row_number()) %>%
                  group_by(percentil_carat, cut) %>%
                  sample_n(size = 1) %>%
                  pull(row_id)
```

```r
anomalias_1 <- diamonds %>% slice(id_anomalias_1)
anomalias_1 <- anomalias_1 %>%
               mutate(price = price * 2.5,
                      anomalia = TRUE)

anomalias_2 <- diamonds %>% slice(id_anomalias_2)
anomalias_2 <- anomalias_2 %>%
               mutate(price =  price / 2.5,
                      anomalia = TRUE)

# Se añaden las anomalías al set de datos
diamonds <- diamonds %>%
            mutate(anomalia = FALSE) %>%
            bind_rows(anomalias_1, anomalias_2)

ggplot() +
geom_point(data = diamonds,aes(x = carat, y = price, color = color), alpha = 0.7) +
geom_point(data = anomalias_1, aes(x = carat, y = price), color = "red") +
geom_point(data = anomalias_2, aes(x = carat, y = price), color = "red") +
lims(x = c(0,3.7)) +
labs(
  title = "Distribución del precio de los diamantes",
  subtitle = "Anomalías simuladas coloreadas de rojo",
  x = "peso",
  y = "precio",
  color = "color") +
guides(col = guide_legend(nrow = 1)) +
facet_wrap(facets = vars(cut))  +
theme_bw() +
theme(legend.position = "none")
```



Modelo

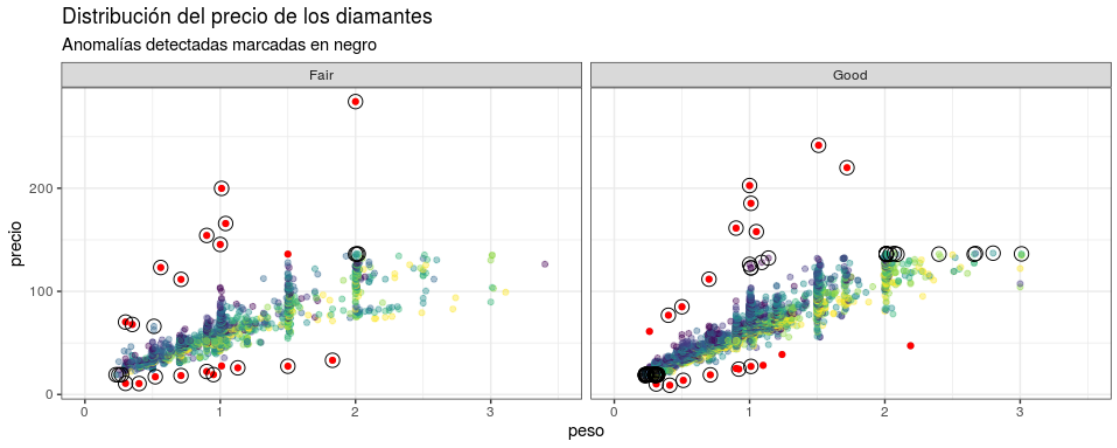
```
modelo_qranger <- ranger(
                    x = diamonds %>% select(-price),
                    y = diamonds$price,
                    num.trees = 5000,
                    max.depth = 2,
                    min.node.size = 300,
                    quantreg = TRUE,
                    # Se emplean todos los cores disponibles -1
                    num.threads = future::availableCores() - 1,
                    seed = 123
                  )
```

Anomalías

Se identifica como anomalías aquellos diamantes con un precio por debajo del percentil del 2% o por encima del percentil 98% predichos por el modelo.

```
predicciones_cuantiles <- predict(
                            modelo_qranger,
                            data = diamonds,
                            type = "quantiles",
                            quantiles = c(0.02, 0.98),
                            # Se emplean todos los cores disponibles -1
                            num.threads = future::availableCores() - 1
                          )

predicciones_cuantiles <- as.data.frame(predicciones_cuantiles$predictions)

diamonds_prediccion <- bind_cols(
                         diamonds,
                         predicciones_cuantiles
                       )

diamonds_prediccion <- diamonds_prediccion %>%
                         mutate(
                           fuera_intervalo = ifelse(
                                 price < `quantile= 0.02` | price > `quantile= 0.98`,
                                 TRUE,
                                 FALSE
                                )
                         )

ggplot() +
geom_point(data = diamonds_prediccion,
           aes(x = carat, y = price, color = color),
           alpha = 0.4) +
```

```
geom_point(data = anomalias_1,
           aes(x = carat, y = price),
           color = "red") +
geom_point(data = anomalias_2,
           aes(x = carat, y = price),
           color = "red") +
geom_point(data = diamonds_prediccion %>% filter(fuera_intervalo == TRUE),
           aes(x = carat, y = price),
           color = "black",
           shape = 1,
           size = 4) +
lims(x = c(0,3.5)) +
labs(
  title = "Distribución del precio de los diamantes",
  subtitle = "Anomalías detectadas marcadas en negro",
  x = "peso",
  y = "precio",
  color = "color") +
guides(col = guide_legend(nrow = 1)) +
facet_wrap(facets = vars(cut))  +
theme_bw() +
theme(legend.position = "none")
```




```
diamonds_prediccion %>%
  filter(anomalia == TRUE) %>%
  select(anomalia, fuera_intervalo)%>%
  table()
```

```
##         fuera_intervalo
## anomalia FALSE TRUE
##     TRUE     7   33
```

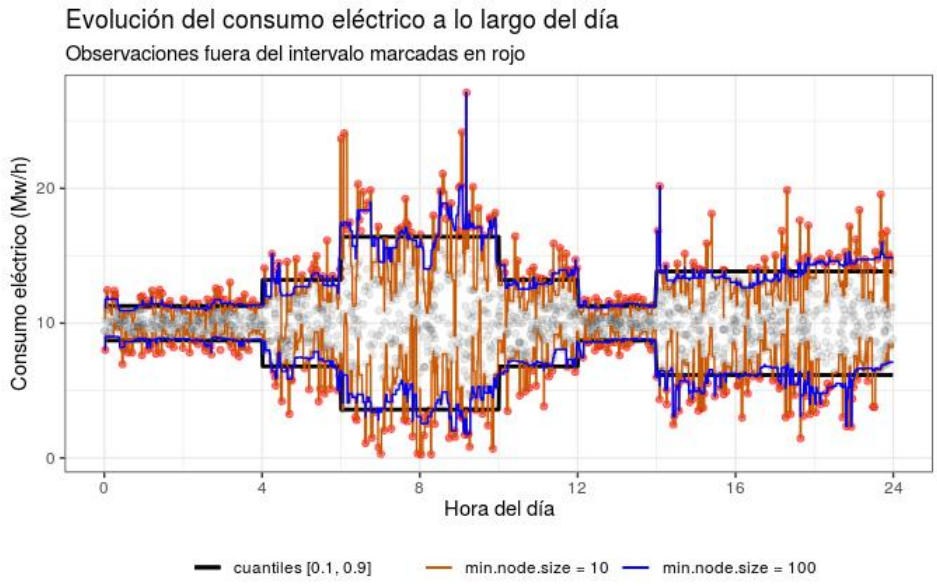
Consideraciones prácticas

Uno de los hiperparámetros de *quantile regression forest* es el número mínimo de observaciones en los nodos terminales. Cómo el algoritmo únicamente emplea aquellas observaciones para las que $Y \leq y$, si son muy pocas las observaciones en los nodos terminales, apenas las habrá que cumplan la condición. Esto se traduce en que *quantile regression forest* es notablemente sensible a *overfitting* por valores bajos de este hiperparámetro. Véase cómo afecta pasar de 10 a 100 observaciones mínimas por nodo al predecir los cuantiles.

```
# Modelo con un mínimo de 10 observaciones por nodo.
modelo_qranger_10 <- ranger(
                       x = datos %>% select(x),
                       y = datos$y,
                       num.trees = 5000,
                       min.node.size = 10,
                       quantreg = TRUE,
                       # Se emplean todos los cores disponibles -1
                       num.threads = future::availableCores() - 1,
                       seed = 123
                     )
# Modelo con un mínimo de 100 observaciones por nodo.
modelo_qranger_100 <- ranger(
                        x = datos %>% select(x),
                        y = datos$y,
                        num.trees = 5000,
                        min.node.size = 100,
                        quantreg = TRUE,
                        # Se emplean todos los cores disponibles -1
                        num.threads = future::availableCores() - 1,
                        seed = 123
                      )

# Se predice todo el rango de X para representar los cuantiles
grid_predictor <- seq(0, 1, length.out = 5000)
predicciones_cuantiles_10 <- predict(
                               modelo_qranger_10,
                               data = data.frame(x = grid_predictor),
                               type = "quantiles",
                               quantiles = c(0.1, 0.9),
                               # Se emplean todos los cores disponibles -1
                               num.threads = future::availableCores() - 1
                             )
predicciones_cuantiles_10 <- as.data.frame(predicciones_cuantiles_10$predictions)
predicciones_cuantiles_10 <- bind_cols(
                               data.frame(x = grid_predictor),
                               predicciones_cuantiles_10
                             )
```

```
predicciones_cuantiles_100 <- predict(
                                modelo_qranger_100,
                                data = data.frame(x = grid_predictor),
                                type = "quantiles",
                                quantiles = c(0.1, 0.9),
                                # Se emplean todos los cores disponibles -1
                                num.threads = future::availableCores() - 1
                              )
predicciones_cuantiles_100 <- as.data.frame(predicciones_cuantiles_100$predictions)
predicciones_cuantiles_100 <- bind_cols(
                               data.frame(x = grid_predictor),
                               predicciones_cuantiles_100
                              )

p <- ggplot() +
      geom_point(
        data = datos %>% filter(dentro_intervalo == TRUE),
        aes(x = x, y = y),
        alpha = 0.1,
        color = "gray20") +
      geom_point(
        data = datos %>% filter(dentro_intervalo == FALSE),
        aes(x = x, y = y),
        alpha = 0.5,
        color = "red") +
      geom_line(aes(x = x, y = cuantil_01, linetype = "cuantiles [0.1, 0.9]"),
                size = 1) +
      geom_line(aes(x = x, y = cuantil_90, linetype = "cuantiles [0.1, 0.9]"),
                size = 1) +
      geom_line(
        data = predicciones_cuantiles_10,
        aes(x = x, y = `quantile= 0.1`, color = "min.node.size = 10")) +
      geom_line(
        data = predicciones_cuantiles_10,
        aes(x = x, y = `quantile= 0.9`, color = "min.node.size = 10")) +
      geom_line(
        data = predicciones_cuantiles_100,
        aes(x = x, y = `quantile= 0.1`, color = "min.node.size = 100")) +
      geom_line(
        data = predicciones_cuantiles_100,
        aes(x = x, y = `quantile= 0.9`, color = "min.node.size = 100")) +
      scale_color_manual(
        name = "",
        breaks = c("min.node.size = 10", "min.node.size = 100"),
        values = c("min.node.size = 10"="#cd5700","min.node.size = 100"= "blue")) +
      scale_linetype_manual(
        name = "",
        breaks = c("cuantiles [0.1, 0.9]", "cuantiles [0.1, 0.9]"),
        values = c("cuantiles [0.1, 0.9]" = "solid")) +
      labs(
```

```
        title = "Evolución del consumo eléctrico a lo largo del día",
        subtitle =  "Observaciones fuera del intervalo marcadas en rojo",
        x = "Hora del día",
        y = "Consumo eléctrico (Mw/h)") +
      theme_bw() +
      theme(legend.position = "bottom")

p <- p +
     scale_x_continuous(breaks = seq(0,1,length.out = 6),
                        labels = c(0, 4, 8, 12, 16, 24))
p
```



Empleando `min.node.size = 10`, el modelo generado pasa por todas las observaciones de menor y mayor valor, está sobreentrenado. Por el contrario, el modelo con un mínimo de 100 observaciones por nodo (`min.node.size = 100`) se ajusta mucho mejor a los cuantiles reales. Lamentablemente, no hay forma de saber *a priori* con exactitud cuál es el valor óptimo de este hiperparámetro.

Anexos

Anexo 1

Simulación ligeramente modificada del ejemplo publicado en *XGBoostLSS – An extension of XGBoost to probabilistic forecasting Alexander März*.

La ecuación empleada para generar los datos del ejemplo es:

$$y \sim \mathcal{N}(10, (1 + 1.5(0.2 < x < 0.3) + 4(0.3 < x < 0.5) + 1.5(0.5 < x < 0.6) + 2(x > 0.7)^2)$$

Para el rango de valores $0 < x < 0.2$ los datos se distribuyen según una normal de media 10 y desviación típica 1. Para el rango de $0.2 < x < 0.3$ la desviación típica aumenta a 2.5. Para el rango $0.3 < x < 0.5$ pasa a 5 y a continuación de $0.5 < x < 0.6$ vuelve a tener una desviación típica de 2.5. Finalmente, para $x > 0.7$ la desviación aumenta a 3. El valor medio se mantiene constante (10) teniendo en todo momento la media de 10.

```
# Simulación distribución uniforme en el rango X
# ------------------------------------------------------------------------------
set.seed(123)
x <- rep(x = seq(0, 1, length.out = 60), each = 40)
y <- rnorm(
        length(x),
        mean = 10,
        sd = 1 + 1.5*(0.2 < x & x < 0.3) + 4*(0.3 < x & x < 0.5) +
             1.5*(0.5 < x & x < 0.6) + 2*(x > 0.7)
     )
# Cálculo del cuantil 0.1 y 0.9 para cada posición de x simulada.
cuantil_01 <- qnorm(
                p = 0.1,
                mean = 10,
                sd = 1 + 1.5*(0.2 < x & x < 0.3) + 4*(0.3 < x & x < 0.5) +
                     1.5*(0.5 < x & x < 0.6) + 2*(x > 0.7)
              )
cuantil_90 <- qnorm(
                p = 0.9,
                mean = 10,
                sd = 1 + 1.5*(0.2 < x & x < 0.3) + 4*(0.3 < x & x < 0.5) +
                     1.5*(0.5 < x & x < 0.6) + 2*(x > 0.7)
              )
datos <- data.frame(y, x, cuantil_01, cuantil_90)
# No puede haber consumos negativos
datos <- datos %>%
         filter(y >=0)
```

```r
# Simulación distribución no uniforme en el rango X
# ------------------------------------------------------------------------------
set.seed(123)
n <- 2000
x <- runif(n)
y <- rnorm(
        n,
        mean = 10,
        sd = 1 + 1.5*(0.2 < x & x < 0.3) + 4*(0.3 < x & x < 0.5) +
             1.5*(0.5 < x & x < 0.6) + 2*(x > 0.7)
     )

# Cálculo del cuantil 0.1 y 0.9 para cada posición de x simulada.
cuantil_01 <- qnorm(
                p = 0.1,
                mean = 10,
                sd = 1 + 1.5*(0.2 < x & x < 0.3) + 4*(0.3 < x & x < 0.5) +
                     1.5*(0.5 < x & x < 0.6) + 2*(x > 0.7)
              )

cuantil_90 <- qnorm(
                p = 0.9,
                mean = 10,
                sd = 1 + 1.5*(0.2 < x & x < 0.3) + 4*(0.3 < x & x < 0.5) +
                     1.5*(0.5 < x & x < 0.6) + 2*(x > 0.7)
              )
datos <- data.frame(y, x, cuantil_01, cuantil_90)

# No puede haber consumos negativos
datos <- datos %>%
         filter(y >=0)
```

Bibliografía

Nicolai Meinshausen. 2006. Quantile Regression Forests. J. Mach. Learn. Res. 7 (December 2006), 983–999.

Regresión cuantílica (Quantile Regression) con R Joaquín Amat Rodrigo

Trabajo fin de máster - Tema: Regresión Cuantil Isabel Martínez Silva 30 de Junio de 2010

A gentle introduction to quantile regression for ecologists, Brian S. Cade and Barry R. Noon

März, Alexander. (2019). XGBoostLSS – An extension of XGBoost to probabilistic forecasting.