Distributional Regression Forest: Random Forest probabilístico

Joaquín Amat Rodrigo j.amatrodrigo@gmail.com

Marzo, 2020

Tabla de contenidos

Versión PDF: Github

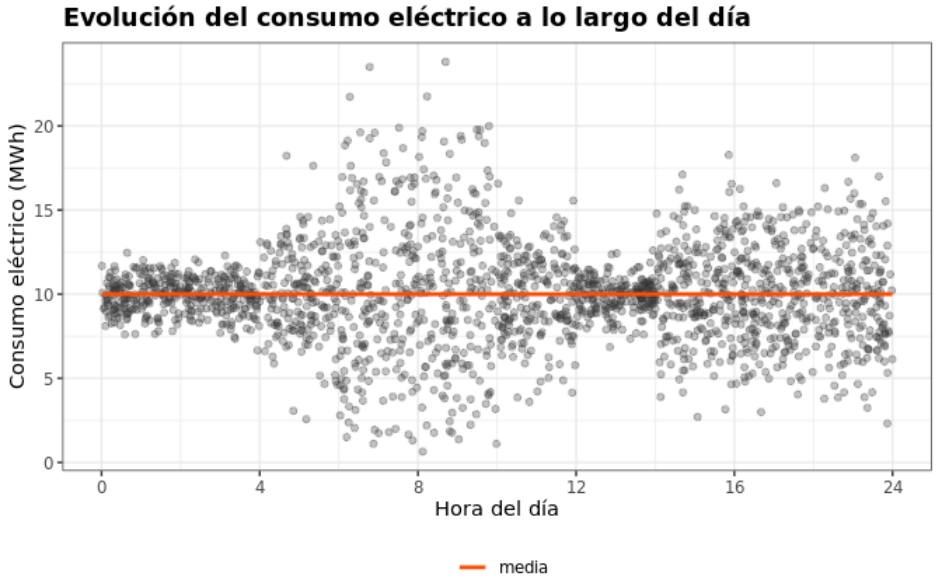
Más sobre ciencia de datos: cienciadedatos.net o joaquinamatrodrigo.github.io

Introducción

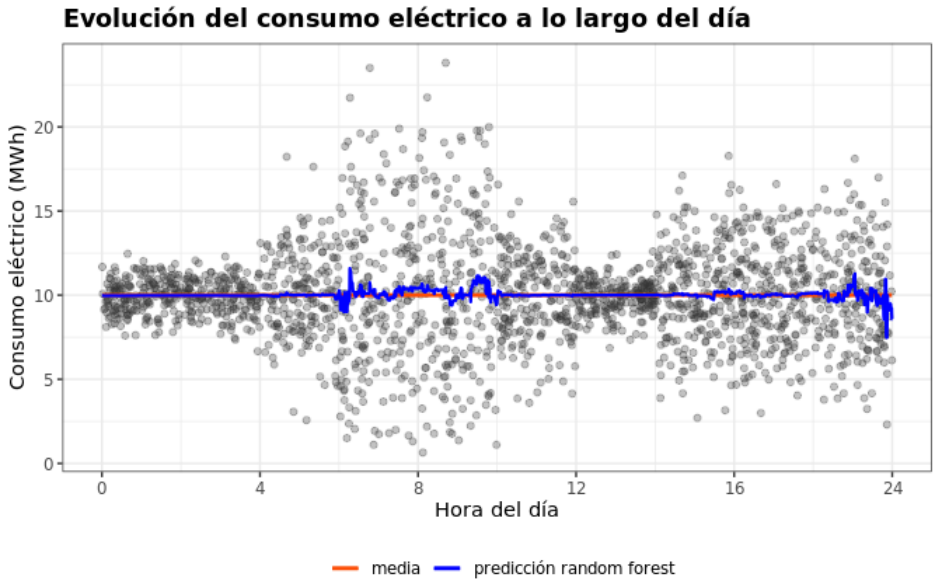
Son muchos los modelos de regresión capaces de predecir una variable respuesta Y en función de uno o varios predictores X. Algunos de ellos consideran que la relación entre Y y X es únicamente lineal (regresión lineal, GLM), mientras que otros permiten incorporar relaciones no lineales o incluso interacciones entre los predictores (SVM, Random Forest, Boosting). De una forma u otra, todos ellos tratan de inferir la relación entre X e Y.

El objetivo de la mayoría de estos algoritmos consiste en predecir el valor medio 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.



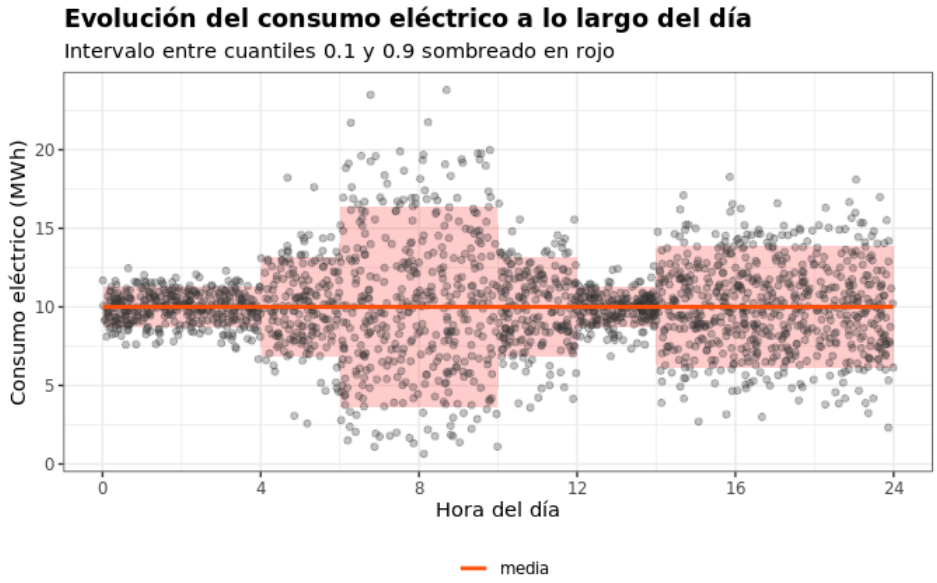
La media del consumo eléctrico es la misma durante todo el día, $\overline{consumo} = 10MWh$, sin embargo, su dispersión no es constante (heterocedasticidad). Véase el resultado de predecir el consumo medio en función de la hora del día con un modelo *random forest*.



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 MWh. 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 costes.

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 medio predicho es en torno a 10 MWh, sin embargo, la probabilidad de que se alcancen consumos de 15 MWh a las 2h es prácticamente nula mientras que esto ocurra a las 8h sí es razonable.

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 sombreando el intervalo central que acumula el 90% de probabilidad de contener las observaciones.



En casos como este, se requiere de algoritmos capaces de aprender cómo varía toda la distribución de Y en función de X. Una vez conocida la distribución, además de predecir el valor medio, se puede estimar su incertidumbre o la probabilidad de que una observación esté por encima o por debajo de un determinado límite.

Otras aplicaciones para las que es útil predecir la de la distribución condicional completa son:

- Detectar anomalías, identificando aquellas observaciones cuya probabilidad predicha es inferior a un determinado valor.
- Entrenar modelos que predicen la mediana (cuantil 0.5) en lugar de la media. Estos modelos son más robustos frente a *outliers*.

En los siguientes apartados se describe el algoritmo **Distributional Random Forest**, una adaptación de *Random Forest* capaz de aprender distribuciones.

Nota: Otras aproximaciones que tratan de resolver este mismo problema como son Quantile Regression Forest y GAMLSS.

Distributional Regression Forest

Algoritmo

Distributional Random Forest es una adaptación del algoritmo de Regression Random Forest que permite predecir, en lugar de la media, la distribución completa (los parámetros que la definen).

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 *Distributional Random Forest* sigue exactamente la misma estrategia para crear el modelo, la diferencia reside en el criterio empleado para la división de nodos y 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 toda la distribución.

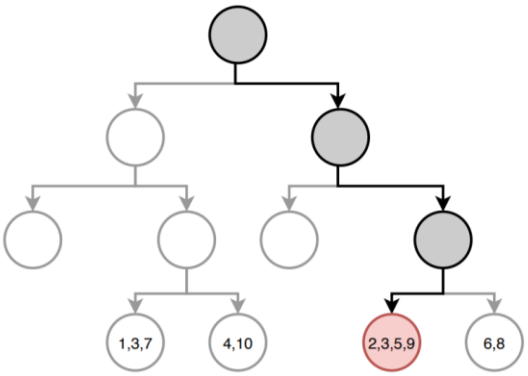
Predicción de un árbol de regresión

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 todas las observaciones de entrenamiento que están en este 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 (1 o 0) 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.

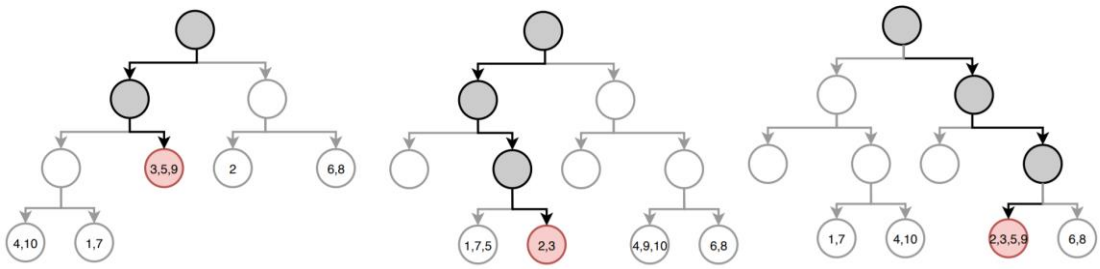
$$\mathbf{w} = (0, \frac{1}{4}, \frac{1}{4}, 0, \frac{1}{4}, 0,0,0, \frac{1}{4}, 0)$$

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 toda la distribución

La modificación necesaria para predecir toda la distribución (ver detalles en Schlosser, Lisa et al.) consiste en emplear árboles de distribución (*distributional tree*) en lugar de los árboles de regresión descritos anteriormente. ¿Cuál es la diferencia?

El primer paso la hora de ajustar un *distributional tree* es definir el tipo de distribución que sigue la variable respuesta y su función de verosimilitud (*likelihood*) que permite estimar sus parámetros θ en función de los datos observados. Por ejemplo, si se asume que la distribución de la variable respuesta es normal, con su función de *likelihood* se obtienen los valores de μ y σ más probables dados los datos disponibles. Esta estimación suele hacerse mediante el método de máxima verosimilitud (*maximum likelihood*).

$$\hat{\theta} = \underset{\theta \in \Theta}{\text{argmax}} \sum_{i=1}^{n} l\,(\theta; y_i)$$

$$l(\theta; y_i) = \log(\text{likelihood}(\theta; y_i))$$

$\hat{\theta}$ es el valor de los parámetros de la distribución que, de entre todos los posibles, maximizan la suma total del logaritmo del *likelihood*.

Una vez asumida una determinada distribución, el árbol se ajusta dividiendo los nodos en aquellos puntos que consiguen el mayor aumento en el *likelihood*. En los árboles de regresión tradicionales suele emplearse como criterio de división el error cuadrático medio.

Para obtener la predicción probabilística de una nueva observación, se recorre todo el árbol hasta llegar a un nodo terminal y se estiman los parámetros de la distribución empleando únicamente las observaciones de entrenamiento que forman parte de ese mismo nodo. Esto

equivale a emplear todas las observaciones del árbol pero ponderando con los pesos de cada observación.

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbf{w}_i^{arbol}\, l(\theta; y_i)$$

En el caso de *Distributional Regression Forest*, la predicción se consigue igual que en *Random Forest*, ponderando las predicciones de todos los árboles que forman el modelo, pero en este caso son árboles de tipo *distributional tree*.

Dos puntos cabe destacar del algoritmo *Distributional Regression Forest*:

- Se trata de modelos paramétrico en tanto que requieren asumir una determinada distribución paramétrica para la variable respuesta.
- La predicción del modelo para una nueva observación X es un valor por cada uno de los parámetros de la distribución seleccionada. Si por ejemplo, se quiere predecir el valor medio de la variable respuesta, hay que emplear el valor de los parámetros devuelto por el modelo y calcular el valor medio esperado de la distribución.

Ejemplo regresión

En **R** se pueden entrenar modelos *Distributional Regression Forest* con el paquete `disttree`.

En el paquete `ranger` y `randomForest`, ambos para la creación de modelos *random forest*, existe la opción de calcular probabilidades en sus predicciones. Esta aproximación es totalmente distinta a la de *Distributional Regression Forest*, solo se puede aplicar a problemas de clasificación y la probabilidad de cada clase se estima empleando el porcentaje de votos hacia cada una, no modelan la distribución.

```
install.packages("disttree", repos="http://R-Forge.R-project.org")
```

Datos

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

```
library(tidyverse)
# Simulación distribución no uniforme en el rango X
# ------------------------------------------------------------------------------
set.seed(12345)
n <- 2000
x <- runif(min = 0, max = 24, n = n)
y <- rnorm(
        n,
        mean = 10,
        sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
             1.5*(12 < x & x < 14.4) + 2*(x > 16.8)
     )

# 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*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(12 < x & x < 14.4) + 2*(x > 16.8)
              )

cuantil_90 <- qnorm(
                p = 0.9,
                mean = 10,
                sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(12 < x & x < 14.4) + 2*(x > 16.8)
              )
```

```
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

Se ajusta un modelo `distforest` con la función `distforest()` asumiendo que la variable respuesta sigue una distribución normal, cuyos parámetros (μ y σ) varían en función del predictor X. De entre sus argumentos cabe destacar:

- `family`: distribución de la variable respuesta. Puede ser cualquiera de las distribuciones disponibles en el paquete `gamlss.dist`. Listado de la versión `5.1-6`.
- `ntree`: número de árboles que forman el modelo.
- `minsplit`: número mínimo de observaciones para realizar una división de nodo.
- `minbucket`: número mínimo de observaciones en los nodos terminales.
- `alpha`: nivel de significancia para la selección de variables. `1` implica que no hay parada temprana en base a test estadísticos.
- `mincriterion`: valor de significancia que debe superarse (1 - p-value) para que se permita una división de nodo. `0` implica que no hay parada temprana en base a test estadísticos.

```
library(disttree)

# Entrenamiento del modelo
modelo_distforest <- disttree::distforest(
                        formula = y ~ x,
                        data = datos,
                        family = NO(),
                        type.tree = "ctree",
                        ntree = 500,
                        control = disttree_control(
                                    minsplit  = 100,
                                    minbucket = 100,
                                    alpha     = 1,
                                    mincriterion = 0
                                  )
                      )
```

Predicción

La predicción del modelo es el valor estimado de los parámetros de la distribución (en este caso μ y σ) para cada valor de X.

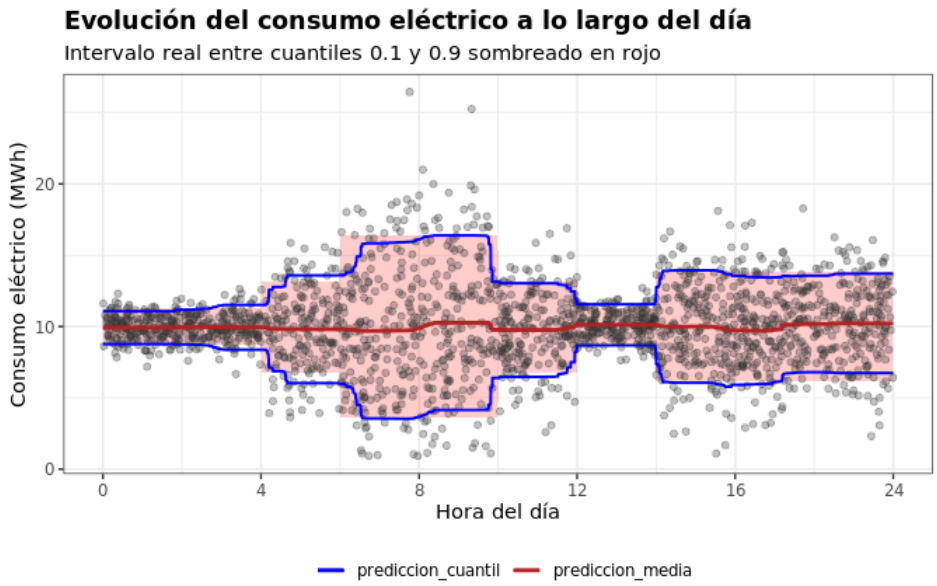
```
# Se predice la distribución para todo el rango de X
grid_predictor <- seq(0, 24, length.out = 4000)

predicciones <- predict(
                    modelo_distforest,
                    newdata = data.frame(x = grid_predictor),
                    type = "parameter"
                )
predicciones <- as.data.frame(predicciones)
predicciones <- bind_cols(data.frame(x = grid_predictor), predicciones)

predicciones %>% head()
```

```
##           x       mu     sigma
## 1 0.0000000 9.919852 0.8973017
## 2 0.0060015 9.919852 0.8973017
## 3 0.0120030 9.919852 0.8973017
## 4 0.0180045 9.919852 0.8973017
## 5 0.0240060 9.919852 0.8973017
## 6 0.0300075 9.919852 0.8973017
```

Una vez predichos los parámetros que caracterizan a la distribución, se puede calcular la probabilidad de cada observación o el intervalo que acumula un determinado porcentaje de probabilidad (intervalo cuantílico).

```
#Cálculo de los cuantiles teóricos para establecer el intervalo central que acumula
#un 90% de probabilidad empleando los parámetros predichos.
predicciones <- predicciones %>%
                mutate(
                cuantil_01_pred = purrr::pmap_dbl(
                                    .l = list(mu = mu, sigma = sigma),
                                    .f = function(mu,sigma){qnorm(p=0.1,mu,sigma)}
                                  ),
                cuantil_90_pred = purrr::pmap_dbl(
                                    .l = list(mu = mu, sigma = sigma),
                                    .f = function(mu,sigma){qnorm(p=0.9,mu,sigma)}
                                  ),
                )

p <- ggplot() +
      geom_ribbon(
        data = datos,
        aes(x = x, ymin = cuantil_01, ymax = cuantil_90),
        fill = "red",
        alpha = 0.2) +
      geom_point(
        data = datos,
        aes(x = x, y = y),
        alpha = 0.3,
        color = "gray20") +
      geom_line(
        data = predicciones,
        aes(x = x, y = cuantil_01_pred, color = "prediccion_cuantil"),
        size = 0.7) +
      geom_line(
        data = predicciones,
        aes(x = x, y = cuantil_90_pred, color = "prediccion_cuantil"),
        size = 0.7) +
      geom_line(
        data = predicciones,
        aes(x = x, y = mu, color = "prediccion_media"),
        size = 1) +
      scale_color_manual(name = "",
                         breaks = c("prediccion_cuantil", "prediccion_media"),
                         values = c("prediccion_cuantil" = "blue",
                                    "prediccion_media"   = "firebrick")) +
      labs(title = "Evolución del consumo eléctrico a lo largo del día",
           subtitle = "Intervalo real entre cuantiles 0.1 y 0.9 sombreado en rojo",
           x = "Hora del día",
           y = "Consumo eléctrico (MWh)") +
      theme_bw() +
      theme(legend.position = "bottom",
            plot.title = element_text(face = "bold"))
```

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



Si por ejemplo, se desea conocer la probabilidad de que a las 8h el consumo supere los 15 *MWh*. Primero se predicen los parámetros de la distribución a para $(X = 8)$ y después se calcula la probabilidad de $Y \geq 15$ con su función de distribución.

```
prediccion <- predict(
                modelo_distforest,
                newdata = data.frame(x = 8),
                type = "parameter"
              )
prediccion
```

```
##         mu    sigma
## 1 9.694728 4.788004
```

```
# Se calcula la probabilidad
probabilidad_consumo <- pnorm(q = 15, mean = 9.269, sd = 5.332, lower.tail = FALSE)
probabilidad_consumo
```

```
## [1] 0.1412252
```

La probabilidad de que a las 8h el consumo sea igual o superior a 15 *MWh* es del 14.1%.

Detección de anomalías (Outliers)

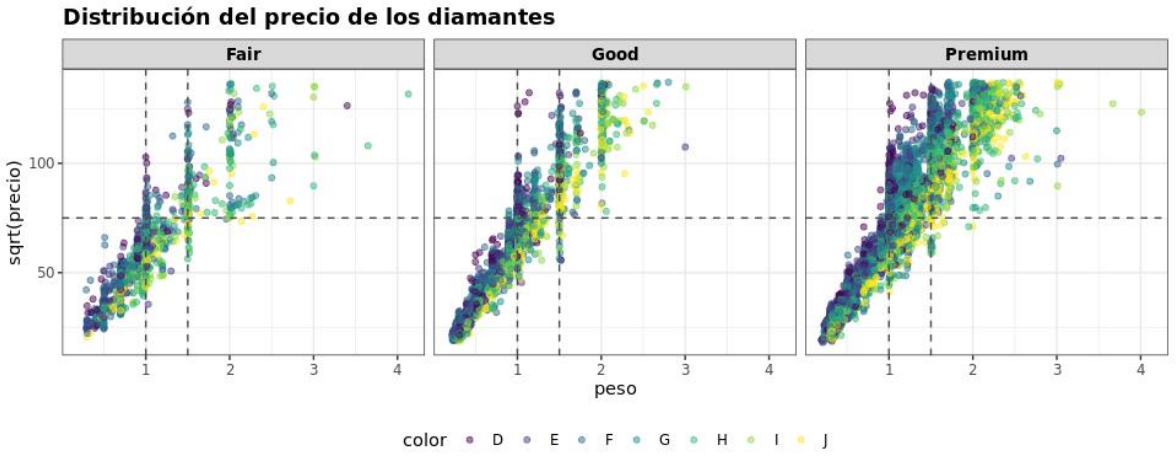
Conocer la probabilidad condicional de la variable respuesta Y permite identificar observaciones que se alejan atípicamente por encima o por debajo del valor esperado para una determinado X. Véase el siguiente ejemplo en el que se trata de identificar precios anómalos de diamantes.

Datos

```
data(diamonds)
set.seed(1234)
datos <- diamonds %>%
         filter(cut %in% c("Fair", "Good", "Premium")) %>%
         mutate(price = sqrt(price)) %>%
         sample_frac(size = 0.7)
```

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

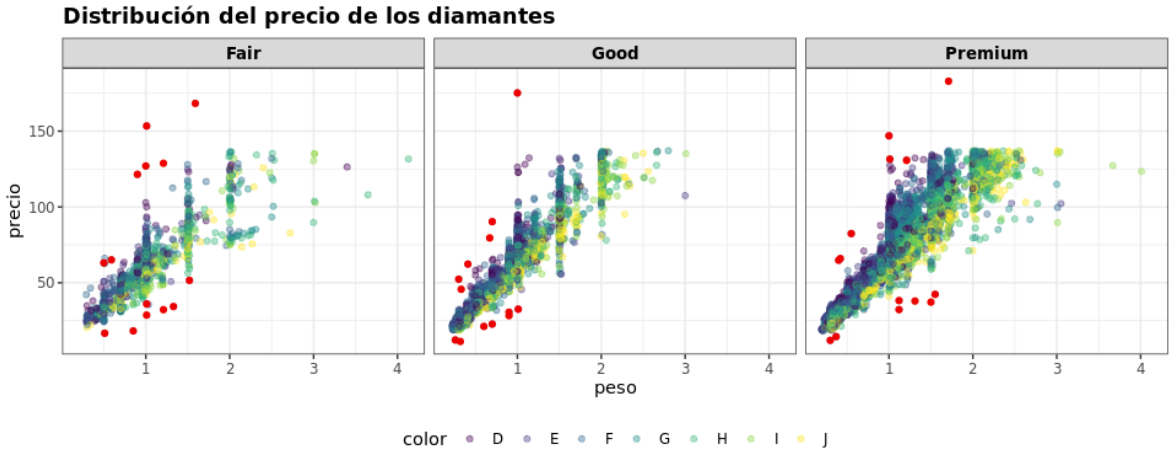
```
ggplot(data = datos, aes(x = carat, y = price, color = color)) +
geom_point(alpha = 0.5) +
geom_vline(xintercept = 1, linetype = "dashed",  color = "gray30") +
geom_vline(xintercept = 1.5, linetype = "dashed",  color = "gray30") +
geom_hline(yintercept = 75, linetype = "dashed", color = "gray30") +
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",
      plot.title = element_text(face = "bold"),
      strip.text = element_text(colour = "black", size = 10, face = 2))
```



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.

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

anomalias_1 <- datos %>% slice(id_anomalias_1)
anomalias_1 <- anomalias_1 %>%
               mutate(price = price * 2,
                      anomalia = TRUE) %>%
               filter(price < 200)

anomalias_2 <- datos %>% slice(id_anomalias_2)
anomalias_2 <- anomalias_2 %>%
               mutate(price =  price / 2,
                      anomalia = TRUE) %>%
               filter(price < 200)

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

ggplot() +
geom_point(data = datos, aes(x = carat, y = price, color = color), alpha = 0.4) +
geom_point(data = anomalias_1, aes(x = carat, y = price), color = "red2") +
geom_point(data = anomalias_2, aes(x = carat, y = price), color = "red2") +
labs(title = "Distribución del precio de los diamantes",
     x = "peso",
     y = "precio",
     color = "color") +
guides(col = guide_legend(nrow = 1)) +
facet_wrap(facets = vars(cut))  +
theme_bw() +
theme(legend.position = "bottom",
      plot.title = element_text(face = "bold"),
      strip.text = element_text(colour = "black", size = 10, face = 2))
```



Modelo

Se modela el precio de los diamantes en función de las variables *carat, cut, color, clarity, depth,* y *table*. Para no aumentar la complejidad del ejemplo, se asume que el precio de los diamantes sigue una distribución normal (`NO()`). En un caso real, convendría realizar un estudio de bondad de ajuste para identificar cual es realmente la distribución más adecuada.

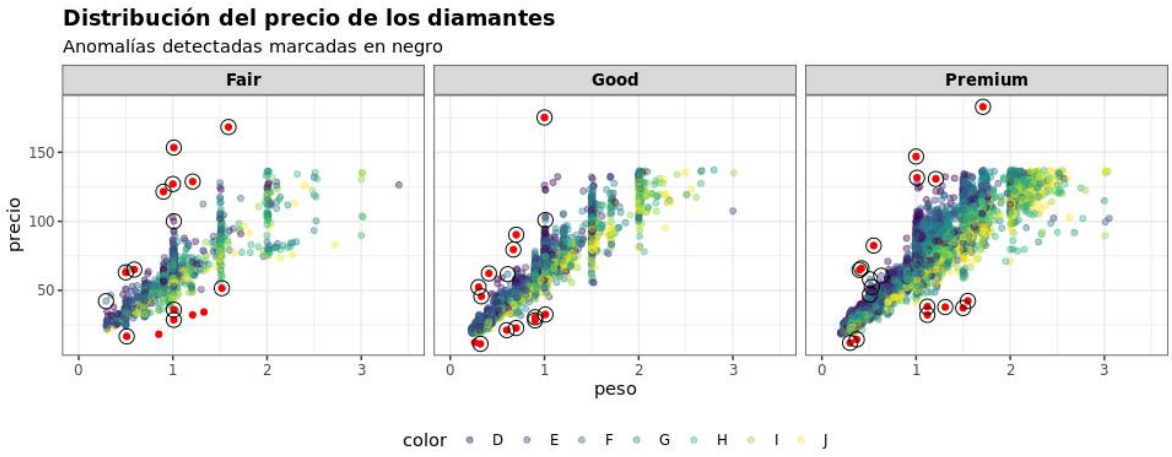
```
modelo_distforest <- disttree::distforest(
                        formula = price ~ carat+cut+color+clarity+depth+table,
                        data    = datos,
                        family  = NO(),
                        type.tree = "ctree",
                        ntree   = 500,
                        control = disttree_control(
                                    minsplit  = 50,
                                    minbucket = 20,
                                    alpha     = 1,
                                    mincriterion = 0
                                  )
                      )
```

Anomalías

Se identifica como anomalías aquellos diamantes cuyo precio tiene una probabilidad predicha inferior al 1%.

```
predicciones <- predict(
                  modelo_distforest,
                  newdata = datos,
                  type = "parameter"
                )

predicciones <- as.data.frame(predicciones)

# Se unen los datos con las predicciones
predicciones <- bind_cols(datos, predicciones)

# Cálculo de las probabilidades de precio acorde al modelo
predicciones <- predicciones %>%
                mutate(
                 prob_lower_tail = purrr::pmap_dbl(
                                      .l = list(q = price, mu = mu, sigma = sigma),
                                      .f = function(q, mu, sigma) {
                                            pnorm(q, mu, sigma, lower.tail = TRUE)
                                           }
                                    ),
                 prob_upper_tail = purrr::pmap_dbl(
                                      .l = list(q = price, mu = mu, sigma = sigma),
                                      .f = function(q, mu, sigma) {
                                            pnorm(q, mu, sigma, lower.tail = FALSE)
                                           }
                                    )
                )

# Identificación de anomalías: probabilidades inferiores al 1%
predicciones <- predicciones %>%
                 mutate(
                   pred_anomalia = ifelse(
                                     prob_lower_tail<0.01 | prob_upper_tail<0.01,
                                     TRUE,
                                     FALSE
                                   )
                 )

ggplot() +
geom_point(data = predicciones,
           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 = predicciones %>% filter(pred_anomalia == 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 = "bottom",
      plot.title = element_text(face = "bold"),
      strip.text = element_text(colour = "black", size = 10, face = 2))
```



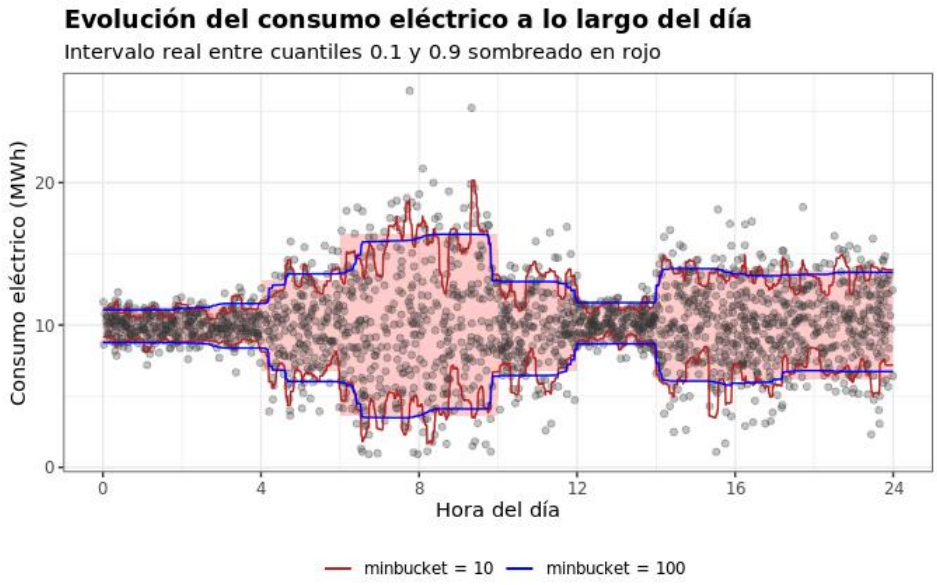

```
predicciones %>%
  dplyr::select(anomalia, pred_anomalia)%>%
  table()
```

```
##         pred_anomalia
## anomalia FALSE  TRUE
##    FALSE 14207     8
##    TRUE      4    37
```

```
rm(modelo_distforest)
```

Consideraciones prácticas

- Ha diferencia de otros modelos como los GAMLSS, también capaces de modelar toda la distribución, *Distributional Regression Forest* selecciona automáticamente los predictores relevantes.
- Por el momento, el paquete `disttree` no permite paralelización, por lo que el entrenamiento de los modelos es bastante lento.
- El número de observaciones en los nodos finales debe ser suficientemente elevado como para que las estimaciones por *máxima verosimilitud* sean mínimamente buenas y no sufran de *overfitting*. Véase la siguiente imagen como ejemplo.



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:

$$y \sim \mathcal{N}(10, (1 + 1.5(4.8 < x < 7.2) + 4(7.2 < x < 12) + 1.5(12 < x < 14.4) + 2(x > 16.8))$$

Para el rango de valores $0 < x < 4.8$ los datos se distribuyen según una normal de media 10 y desviación típica 1. Para el rango de $4.8 < x < 7.2$ la desviación típica aumenta a 2.5. Para el rango $7.2 < x < 12$ pasa a 5, a continuación de $12 < x < 14.4$ desciende 2.5 y de $14.4 < x < 14.4$ a 1. Finalmente, para $x > 16.8$ la desviación aumenta a 3. El valor medio se mantiene constante (10) teniendo en todo momento la media de 10.

```
library(dplyr)

# Simulación distribución uniforme en el rango X
# ------------------------------------------------------------------------------
set.seed(123)
x <- rep(x = seq(0, 1, length.out = 96), each = 30)
y <- rnorm(
        length(x),
        mean = 10,
        sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
             1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
     )

# 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*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
              )
cuantil_90 <- qnorm(
                p = 0.9,
                mean = 10,
                sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
              )
```

```
datos    <- data.frame(y, x, cuantil_01, cuantil_90)
# No puede haber consumos negativos
datos <- datos %>%
         filter(y >=0)

# Simulación distribución no uniforme en el rango X
# ------------------------------------------------------------------------------
set.seed(12345)
n <- 2000
x <- runif(min = 0, max = 24, n = n)
y <- rnorm(
        n,
        mean = 10,
        sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
             1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
     )
# 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*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
              )

cuantil_90 <- qnorm(
                p = 0.9,
                mean = 10,
                sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
              )
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
                                   )
        )
```

Bibliografía

Distributional Regression Forests for Probabilistic Precipitation Forecasting in Complex Terrain (2018) by Lisa Schlosser and Torsten Hothorn and Reto Stauffer and Achim Zeileis
http://arxiv.org/abs/1804.02921

Nicolai Meinshausen. 2006. Quantile Regression Forests. J. Mach. Learn. Res. 7 (December 2006), 983–999.
http://www.jmlr.org/papers/volume7/meinshausen06a/meinshausen06a.pdf