EST-25134: Aprendizaje Estadístico

Profesor: Alfredo Garbuno Iñigo — Primavera, 2023 — Modelos en ensamble.
Objetivo: En esta sección estudiaremos una forma de incorporar varios modelos para crear un modelo predictivo mas fuerte que un modelo individual. La estrategia está basada en una técnica de remuestreo que ya hemos estudiado previamente.
Lectura recomendada: Sección 8.2 de [2]. Capítulo 15 de [1].

1. INTRODUCCIÓN

Anteriormente, vimos los modelos predictivos basados en `árboles de decisión` (regresión y clasificación). En esta sección del curso estudiaremos `distintas estrategias` para combinarlos para obtener un mejor modelo predictivo al costo de `interpretabilidad`. Algunos de estos modelos continúan representando el estado del arte en competencias como Kaggle para `datos tabulares`.

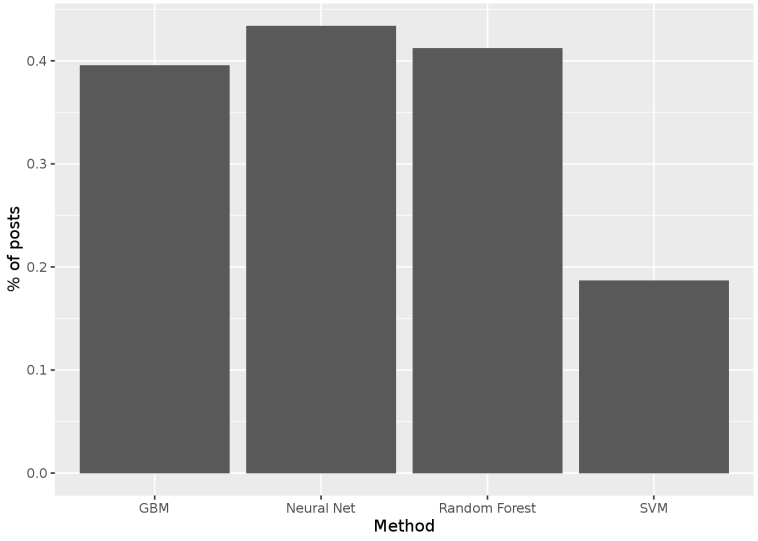


FIGURA 1. *Algoritmos que tienden a quedar en los primeros lugares en las competencias de Kaggle. Tomado de aquí.*

2. REMUESTREO O BOOTSTRAP

Utilizar técnicas de remuestreo nos permite cuantificar la variabilidad de un estimador estadístico sin necesidad de invocar un régimen asintótico para el procedimiento. Asimismo, nos permite controlar, hasta cierto punto, la variabilidad de nuestros estimadores.

Por ejemplo, consideremos la situación en donde tenemos una muestra de n observaciones $Z_1, \ldots, Z_n$ las cuales tienen una varianza σ^2. Es fácil demostrar que la varianza de la media $\bar{Z}_n$ tiene una varianza σ^2/n.

Esto quiere decir, que podemos promediar para reducir la varianza estimada.

2.0.1. Para pensar: Usualmente no tenemos acceso al proceso generador de datos (ya sea $\mathbb{P}_{X,Y}$ ó $\mathbb{P}_X$). ¿Qué estrategia podemos utilizar?

2.1. Bootstrap

Podemos utilizar la muestra $z_1, \ldots, z_n \overset{\text{iid}}{\sim} \pi$ como un *proxy* de la población de la cual queremos generar observaciones. En este sentido, consideramos que la función de acumulación empírica (`ECDF`, por sus siglas en inglés) es un *buen* estimador de la función de probabilidad (ó `CDF` por sus siglas en inglés)

$$\pi[X \leq x] \approx \hat{\pi}_n[X \leq x] = \frac{1}{n}\sum_{i=1}^{n} I_{[z_i \leq x]}\,. \tag{1}$$

Con este procedimiento podemos generar B conjuntos de datos

$$z_1^{(b)}, \ldots, z_n^{(b)} \overset{\text{iid}}{\sim} \hat{\pi}_n\,, \qquad b = 1, \ldots, B\,, \tag{2}$$

para obtener estimadores $\hat{\theta}_n^{(b)} = t(z_1^{(b)}, \ldots, z_n^{(b)})$ y, a través de un promedio, obtener un estimador

$$\bar{\theta}_{B,n}^{(\text{bag})} = \frac{1}{B}\sum_{b=1}^{B} \hat{\theta}_n^{(b)}\,, \tag{3}$$

con varianza que se reduce a una tasa $1/B$.

El muestreo $z_1^{(b)}, \ldots, z_n^{(b)} \overset{\text{iid}}{\sim} \hat{\pi}_n$ implica tomar muestras **con** reemplazo del conjunto de datos observado. Nota que las remuestras son del mismo tamaño que la muestra original. Es decir, cada remuestra b tiene n observaciones. Como el procedimiento es con reemplazo, esto puede ocasionar que pueda haber algunas observaciones que repitan en la remuestra.

2.2. Ejemplo: Suavizadores

La estrategia de remuestreo nos puede ayudar a cuantificar la estabilidad de ciertos estimadores. Por ejemplo, consideremos los datos que teníamos sobre el ingreso para un conjunto de 150 observaciones. El interés es construir un suavizador que relacione `Edad` con `Ingreso`. Utilizaremos un suavizador de *splines* con 15 grados de libertad, ver Fig. 2.

A través de remuestreo podemos cuantificar la estabilidad de dicha estimación, ver Fig. 3.

La estabilidad también la podemos graficar por medio de intervalos de confianza. Ver Fig. 4.

3. BOOTSTRAPPED AGGREGATION: BAGGING

En el contexto de modelado predictivo nos interesa `estimar la relación` que existe entre atributos x y la respuesta y por medio de una $f : \mathcal{X} \mapsto \mathcal{Y}$. Dicho estimador, lo denotamos por $\hat{f}_n$ haciendo énfasis en que ha sido construido con una muestra de tamaño n al minimizar una función de pérdida adecuada.

Si utilizamos *bootstrap*, para cada uno de los B conjuntos de entrenamiento estimamos $\hat{f}_n^{(b)}$ para poder hacer predicciones por medio de

$$\hat{f}_{B,n}^{(\text{bag})}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_n^{(b)}(x)\,, \tag{4}$$

esto lo llamamos `bootstrap aggregation` o `bagging`.

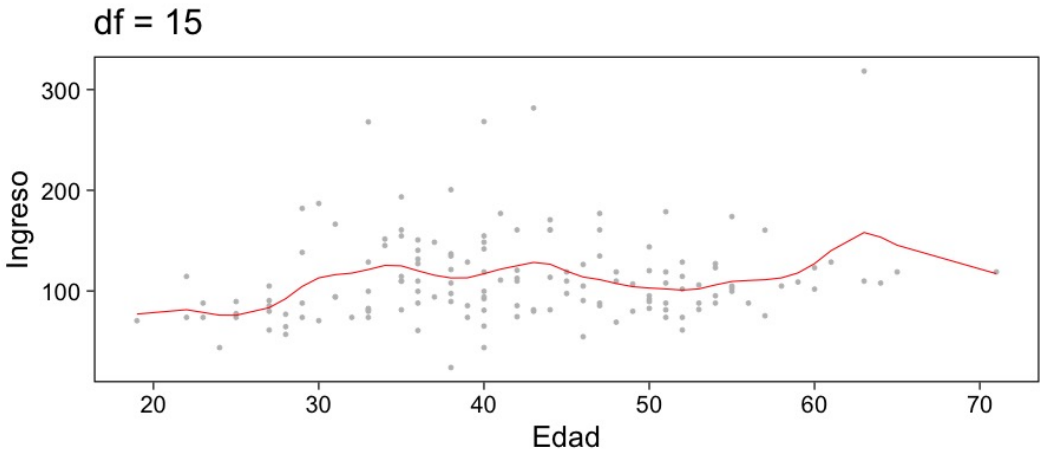


FIGURA 2. *Suavizador por* splines *con 15 grados de libertad.*

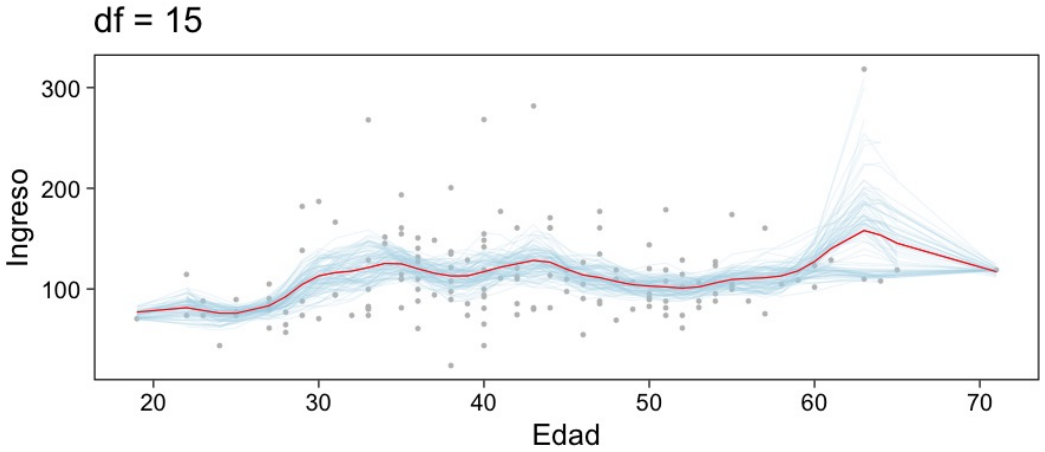


FIGURA 3. *Suavizador por* splines *con 15 grados de libertad, réplicas con remuestreo.*

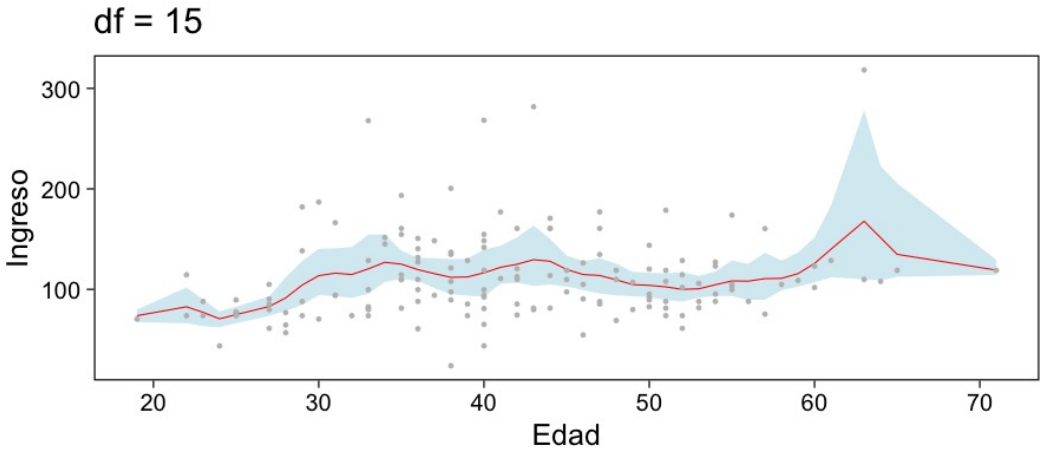


FIGURA 4. *Suavizador por* splines *con 15 grados de libertad, réplicas con remuestreo.*

3.1. En problemas de clasificación

Para problemas de clasificación podemos considerar las predicciones de cada uno de los modelos $\hat{f}_n^{(b)}$ y tomar la clase con `más votos` dentro del conjunto de B predictores.

3.2. Error de generalización

- Usando *bootstrap* entrenamos con cada uno de los conjuntos de datos remuestreados.
- Cada conjunto remuestreado utiliza, en promedio, 2/3 de los datos originales.
- El conjunto no utilizado lo llamamos `conjunto fuera de bolsa` (*out-of-bag*, `OOB`).
- Podemos obtener predicciones para cada observación $i = 1, \ldots, n$ cuando se encuentra en algún conjunto `OOB`. En promedio, tenemos $B/3$ predicciones para cada observación, las cuales podemos promediar para obtener la predicción final.
- Esto es un estimador de `LOO-CV` utilizando `bagging`.

3.3. Observaciones

- El estimador $\hat{f}_{B,n}^{(\mathsf{bag})}$ es un estimador Monte Carlo. ¿De qué?
- El estimador $\hat{f}_{B,n}^{(\mathsf{bag})} \to \hat{f}_n$ con $B \to \infty$ en cada uno de los puntos a evaluar x.
- Cuando los atributos están altamente correlacionados los árboles de decisión pueden presentar varianza alta. En esta situación `bagging` puede suavizar la varianza y reducir el error de generalización.

3.4. Bagging, regresión y `MSE`

- Si estamos en tareas de regresión y medimos el error de generalización por medio de pérdida cuadrática obtenemos lo siguiente

$$\mathbb{E}\left(y - \hat{f}^*(x)\right)^2 \geq \mathbb{E}\left(y - \mathbb{E}\hat{f}^*(x)\right)^2, \tag{5}$$

  donde $\hat{f}^*$ es una estimación por medio de una remuestra y $\mathbb{E}\hat{f}^*$ es el valor esperado de las estimaciones de f utilizando remuestras.
- Por lo tanto, *bagging* podrá disminuir el `MSE`.

3.5. Bagging y clasificación

- En problemas de clasificación, no tenemos descomposición aditiva de sesgo y varianza.
- El uso de `bagging` puede hacer de un mal clasificador, algo todavía peor. Consideremos el caso siguiente.

3.5.1. Bagging *y clasificadores:* Supongamos que tenemos un clasificador binario que asigna $Y = 1$ para todo x con probabilidad 0.4. ¿Cuál es el la tasa de error de clasificación de este modelo? ¿Cuál sería la tasa de error de clasificación de un consenso con este modelo?

3.5.2. Bagging *y la sabiduría de las masas:* Supongamos que tenemos una colección de clasificadores independientes donde cada uno tiene una tasa de error de $\varepsilon < 0.5$, y sea

$$S_1(x) = \sum_{b=1}^{B} I[G^{(b)}(x) = 1], \tag{6}$$

el voto por consenso de que la instancia x pertenezca a la clase 1. Dado que los clasificadores son independientes entonces

$$S_1(x) \sim \mathsf{Binomial}(B, 1 - \varepsilon), \tag{7}$$

donde

$$\mathbb{P}(\text{ clasificación correcta }) = \mathbb{P}(S_1 > B/2) \approx 1\,, \tag{8}$$

con B suficientemente grande.

El resultado anterior se conoce como `Sabiduría de las masas` en donde se asume que cada clasificador es un clasificador `débil`. Con tasa de error ligeramente menor al azar. Para que el consenso de dichos clasificadores tenga buenos resultados se necesita, además, que los clasificadores sean `independientes`.

3.6. Observaciones

- Utilizar `bagging` en un problema de clasificación con árboles no es un procedimiento que utilice árboles independientes. Por lo tanto no hay garantía de que el consenso mejore el error de clasificación.

4. BOSQUES ALEATORIOS

El modelo propuesto de Bosques aleatorios (`RF` por sus siglas en inglés) ayuda a decorrelacionar un conjunto de árboles. Para lograr esto seguimos utilizando remuestreo para seleccionar conjuntos de datos de entrenamiento. Al mismo tiempo, con cada conjunto de remuestras, utilizamos un conjunto de m predictores al azar para entrenar. Esto es, utilizamos para cada remuestra, un subconjunto distinto de predictores para entrenar un árbol.

Usualmente consideramos $m \approx \sqrt{p}$. Esto permite restringir el espacio de búsqueda y dejar de utilizar consistentemente los mismos predictores en cada remuestra.

4.1. Motivación

Si consideramos la situación donde tenemos B variables iid cada una con varianza σ^2 entonces el promedio tendrá varianza igual σ^2/B. Si las variables son sólo id con correlación positiva ρ , entonces el promedio tendrá varianza igual a

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2\,. \tag{9}$$

Incluso aunque tomemos un número suficiente de árboles para controlar el segundo término, el primer término no desvanece con $B \to \infty$. Es por esto que bosques aleatorios busca reducir la correlación entre árboles al permitir que se ajusten a conjuntos aleatorios (en observaciones y predictores) por medio de remuestreo.

4.2. Sobre-ajuste

- El consenso de votos tiende a ser robusto contra sobre-ajustar y si utilizamos una B (el número de árboles) suficientemente grande estabilizamos la variabilidad del error de generalización.
- Usualmente tenemos problemas de sobre-ajuste cuando el número de predictores es alto y el número de predictores relevantes para la predicción es pequeño.

4.3. Análisis de ajuste

La predicción de un bosque aleatorio se realiza por medio de

$$\hat{f}_{\mathsf{RF}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{(b)}(x) = \frac{1}{B}\sum_{b=1}^{B} T\left(x; \Theta(\mathcal{D}_n^{(b)})\right)\,, \tag{10}$$

donde $T(x;\Theta)$ denota la predicción de un árbol utilizando los parámetros (variables de selección, puntos de corte) Θ. La notación $\Theta(\mathcal{D}_n)$ hace énfasis en que los parámetros que gobiernan el árbol fueron escogidos utilizando el conjunto de datos $\mathcal{D}_n$. El término $\mathcal{D}_n^{(b)}$ hace énfasis en que el conjunto de entrenamiento es una remuestra del conjunto original.

El predictor tiende a satisfacer la siguiente igualdad (ley de los grandes números, $B \to \infty$)

$$\hat{f}_{\mathsf{RF}}(x) = \mathbb{E}_{\Theta|\mathcal{D}_n} T\left(x;\Theta(\mathcal{D}_n)\right) , \tag{11}$$

donde hacemos énfasis en que es un valor esperado condicional en los datos de entrenamiento.

Nos interesa evaluar el `error estándar` de dicho estimador. Lo cual escribimos como

$$\mathsf{SE}\left(\hat{f}_{\mathsf{RF}}(x)\right)^2 = \mathbb{V}\left(\hat{f}_{\mathsf{RF}}(x)\right) = \rho(x)\cdot\sigma^2(x) , \tag{12}$$

donde:

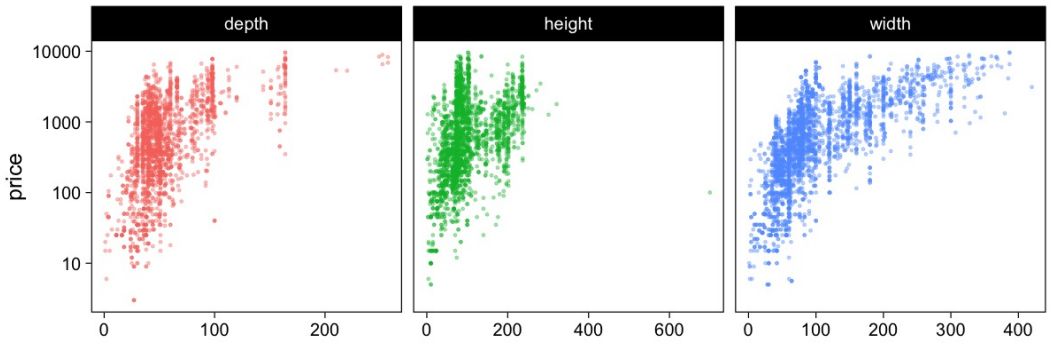
- $\rho(x)$ es la correlación entre dos árboles

$$\rho(x) = \mathsf{Corr}\left[T\left(x;\Theta_i(\mathcal{D}_n)\right), T\left(x;\Theta_j(\mathcal{D}_n)\right)\right] . \tag{13}$$

- $\sigma^2(x)$ es la varianza de cualquier árbol

$$\sigma^2(x) = \mathbb{V}\left(T\left(x;\Theta(\mathcal{D}_n)\right)\right) . \tag{14}$$

5. APLICACIÓN: PREDICCIÓN DE PRECIOS IKEA




```
ikea_df ← ikea ▷
  select(price, name, category, depth, height, width) ▷
  mutate(price = log10(price)) ▷
  mutate_if(is.character, factor)

ikea_df ▷ print(n = 5)
```

```
# A tibble: 3,694 × 6
  price name                  category      depth height width
  <dbl> <fct>                 <fct>         <dbl>  <dbl> <dbl>
1  2.42 FREKVENS              Bar furniture    NA     99    51
2  3.00 NORDVIKEN             Bar furniture    NA    105    80
3  3.32 NORDVIKEN / NORDVIKEN Bar furniture    NA     NA    NA
4  1.84 STIG                  Bar furniture    50    100    60
5  2.35 NORBERG               Bar furniture    60     43    74
# ... with 3,689 more rows
#  Use 'print(n = ...)' to see more rows
```

```
set.seed(123)
ikea_split ← initial_split(ikea_df, strata = price)
ikea_train ← training(ikea_split)
ikea_test ← testing(ikea_split)

set.seed(234)
ikea_folds ← vfold_cv(ikea_train, strata = price)
```

```
library(textrecipes)
ranger_recipe ←
  recipe(formula = price ~ ., data = ikea_train) ▷
  step_other(name, category, threshold = 0.01) ▷
  step_clean_levels(name, category) ▷
  step_impute_knn(depth, height, width)

ranger_spec ←
  rand_forest(mtry = tune(), min_n = tune(), trees = 1000) ▷
  set_mode("regression") ▷
  set_engine("ranger")

ranger_workflow ←
  workflow() ▷
  add_recipe(ranger_recipe) ▷
  add_model(ranger_spec)

set.seed(8577)
## Create a cluster object and then register:
cl ← makePSOCKcluster(6)
doParallel::registerDoParallel(cl)

ranger_tune ←
  tune_grid(ranger_workflow,
            resamples = ikea_folds,
            grid = 11,
            control = control_grid(parallel_over = "resamples", verbose = TRUE
               )
            )
```

```
show_best(ranger_tune, metric = "rmse")
```

```
# A tibble: 5 × 8
   mtry min_n .metric .estimator  mean     n std_err .config
  <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1     2     4 rmse    standard   0.323    10 0.00602 Preprocessor1_Model10
2     5     6 rmse    standard   0.331    10 0.00562 Preprocessor1_Model06
3     4    10 rmse    standard   0.332    10 0.00570 Preprocessor1_Model05
4     3    18 rmse    standard   0.339    10 0.00569 Preprocessor1_Model01
5     2    21 rmse    standard   0.343    10 0.00561 Preprocessor1_Model08
```

```
show_best(ranger_tune, metric = "rsq")
```

```
# A tibble: 5 × 8
   mtry min_n .metric .estimator  mean     n std_err .config
  <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1     2     4 rsq     standard   0.752    10  0.0106 Preprocessor1_Model10
2     5     6 rsq     standard   0.740    10  0.0100 Preprocessor1_Model06
3     4    10 rsq     standard   0.738    10  0.0101 Preprocessor1_Model05
4     3    18 rsq     standard   0.728    10  0.0104 Preprocessor1_Model01
5     2    21 rsq     standard   0.723    10  0.0107 Preprocessor1_Model08
```

```
final_rf ← ranger_workflow ▷
  finalize_workflow(select_best(ranger_tune))

final_rf
```

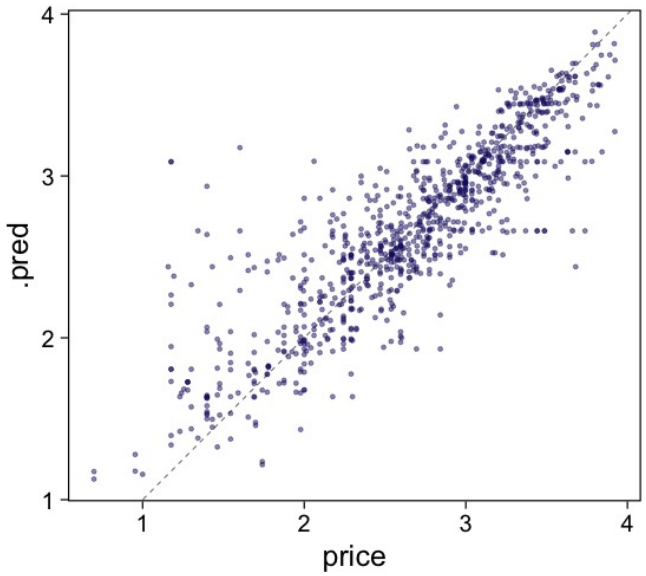
```
Warning message:
No value of ‘metric‘ was given; metric ’rmse’ will be used.
== Workflow ====================================================================
Preprocessor: Recipe
Model: rand_forest()

-- Preprocessor ----------------------------------------------------------------
3 Recipe Steps●

 step_other()●
 step_clean_levels()●
 step_impute_knn()

-- Model -----------------------------------------------------------------------
Random Forest Model Specification (regression)

Main Arguments:
  mtry = 2
  trees = 1000
  min_n = 4

Computational engine: ranger
```

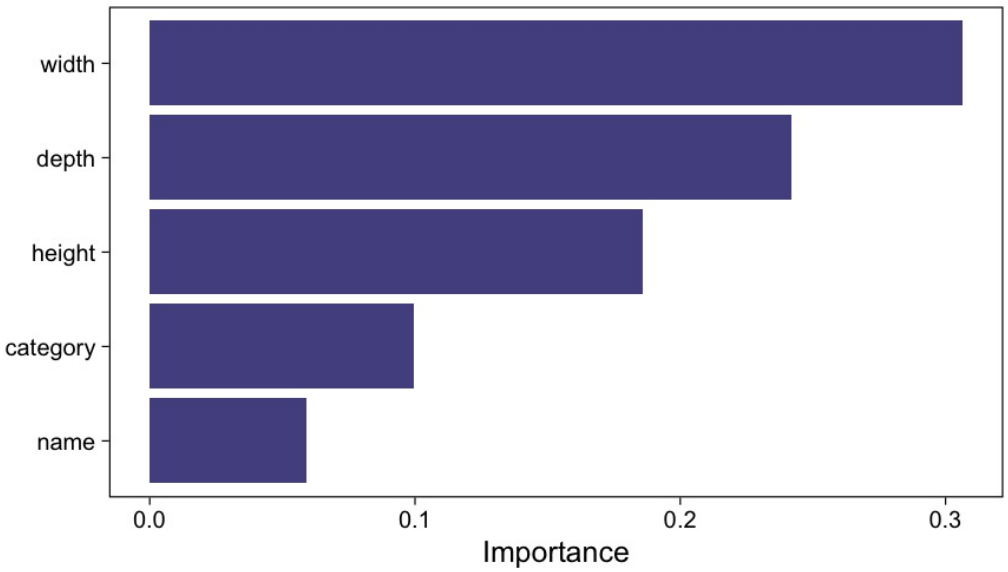
```
ikea_fit ← last_fit(final_rf, ikea_split)
ikea_fit
```

```
# Resampling results
# Manual resampling
# A tibble: 1 × 6
  splits             id               .metrics .notes   .predictions .workflow
  <list>             <chr>            <list>   <list>   <list>       <list>
1 <split [2770/924]> train/test split <tibble> <tibble> <tibble>     <workflow
    >
```

```
collect_metrics(ikea_fit)
```

```
# A tibble: 2 × 4
  .metric .estimator .estimate .config
  <chr>   <chr>          <dbl> <chr>
1 rmse    standard       0.318 Preprocessor1_Model1
2 rsq     standard       0.752 Preprocessor1_Model1
```





6. CONCLUSIONES

- Los bosques aleatorios son uno de los métodos más generales de predicción.
- Son fáciles de entrenar, usualmente ajustando dos parámetros por validación cruzada.
- Heredan ventajas de los árboles. Por ejemplo, las predicciones siempre se encuentran en el rango de las observaciones.
- Pueden ser lentos en predicción.
- Tienen capacidad de extrapolación limitada.

REFERENCIAS

[1] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, New York, NY, 2009. ISBN 978-0-387-84857-0 978-0-387-84858-7. . 1

[2] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics. Springer US, New York, NY, 2021. 1