EST-25134: Aprendizaje Estadístico

Profesor: Alfredo Garbuno Iñigo — Primavera, 2023 — Interpretabilidad y explicabilidad de modelos predictivos.
Objetivo: En aplicaciones de modelos predictivos usualmente se consideran modelos con alto poder predictivo. A su vez, estos modelos son altamente complejos y es dificil *explicar* el cómo una predicción es realizada a consecuencia del vector de atributos en consideración. En esta sección estudiaremos algunas de las nociones de interpretabilidad de modelos.
Lectura recomendada: Los libros de Biecek and Burzykowski [1] y Molnar [2] son dos publicaciones recientes que tratan temas de interpretabilidad y explicabilidad de modelos.

1. INTRODUCCIÓN

En aplicaciones de modelos predictivos usualmente se consideran modelos con alto poder predictivo. A su vez, estos modelos son altamente complejos y es dificil *explicar* el cómo una predicción es realizada a consecuencia del vector de atributos en consideración. En esta sección estudiaremos algunas de las nociones de interpretabilidad de modelos.

Para algunos modelos, como regresión lineal o árboles de decisión, es relativamente sencillo interpretar las relaciones entre atributos y variable respuesta.

Para ilustrar retomaremos el ejemplo de productos de `Ikea`, el cual es original de: Tune random forests for #TidyTuesday IKEA prices.

Los datos que tenemos disponibles son los siguientes.

```
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
```

Los cuales son sometidos a nuestro típico flujo de trabajo de ajuste de modelos predictivos junto con un proceso de separación de muestras para métricas de generalización y selección de hiper-parámetros.

```r
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)
```

```r
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)
```

```r
linear_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) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_predictors())
```

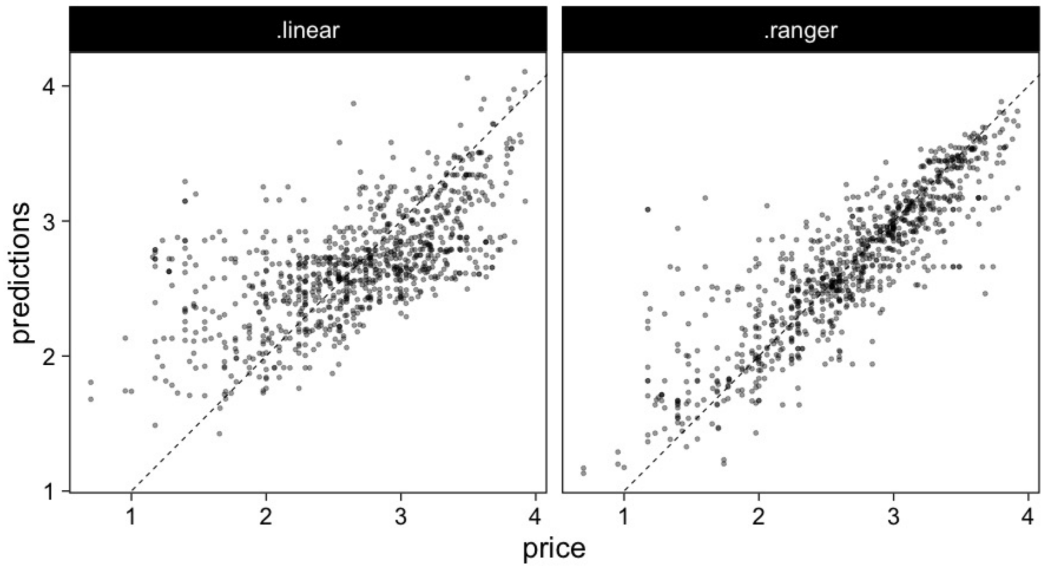
1.1. Especificación del modelo

```r
linear_spec <-
  linear_reg(penalty = 1e-3) |>
  set_mode("regression") |>
  set_engine("glmnet")

linear_workflow <-
  workflow() |>
  add_recipe(linear_recipe) |>
  add_model(linear_spec)
```

```r
ranger_spec <-
  rand_forest(trees = 1000) |>
  set_mode("regression") |>
  set_engine("ranger", importance = "impurity")

ranger_workflow <-
  workflow() |>
  add_recipe(ranger_recipe) |>
  add_model(ranger_spec)
```

```r
all_cores <- parallel::detectCores(logical = TRUE) - 1
library(doParallel)
cl <- makePSOCKcluster(all_cores)
registerDoParallel(cl)
```

```
ikea_lm <- linear_workflow |> fit(data = ikea_train)
ikea_rf <- ranger_workflow |> fit(data = ikea_train)
```



2. INTERPRETABILIDAD

Iremos explorando los conceptos necesarios para interpretabilidad conforme los necesitemos. Primero necesitaremos herramientas de trabajo desde `R`, y para esta tarea podeos usar `lime`, `vip` y `DALEXtra`.

En general podemos usar:

- `vip` para usar métodos basados en algún modelo en particular para aprovechar la estructura del modelo predictivo.
- `DALEX` para usar métodos que no requieren de una estrcutura en particular (usaremos `DALEXtra` para compatibilidad con `tidymodels`).

```
library(DALEXtra)
```

Para poder comenzar lo que tenemos que hacer es crear los objetos de `DALEX` (*moDel Agnostic Language for Exploration and eXplanation*).

```
explainer_lm <-
  explain_tidymodels(
    ikea_lm,
    data = ikea_train |> select(-price),
    y    = ikea_train |> pull(price),
    label = "linear model",
    verbose = FALSE
  )
```

```
explainer_rf <-
  explain_tidymodels(
    ikea_rf,
    data = ikea_train |> select(-price),
```

```
    y     = ikea_train ▷ pull(price),
    label = "random forest",
    verbose = FALSE
  )
```

3. MÉTODOS DE INTERPRETABILIDAD LOCAL

Los siguientes métodos que veremos son `métodos locales` es decir, tomamos una $x^\star \in \mathcal{X} \subset \mathbb{R}^p$ en particular y exploramos la respuesta a partir de este punto. Por ejemplo, consideremos como $x^\star$ la observación donde queremos explorar el modelo.

```
set.seed(123)
mueble ← ikea_test ▷ sample_n(1)
mueble
```

```
# A tibble: 1 × 6
  price name      category                          depth height width
  <dbl> <fct>     <fct>                             <dbl>  <dbl> <dbl>
1  2.98 TYSSEDAL Chests of drawers & drawer units     49    102    67
```

Sabemos de modelos lineales que los coeficientes están asociados a las contribuciones de cada predictor a la respuesta. Usualmente, interpretados bajo un principio *ceteris paribus* (interpretado en nuestro contexto: dejando constantes los demás predictores constantes).

```
ikea_lm ▷ extract_fit_parsnip() ▷
  tidy() ▷
  print(n = 5)
```

```
# A tibble: 35 × 3
  term          estimate penalty
  <chr>            <dbl>   <dbl>
1 (Intercept)   2.67      0.001
2 depth         0.104     0.001
3 height        0.155     0.001
4 width         0.237     0.001
5 name_bekant   0.00497   0.001
# ... with 30 more rows
#  Use ‘print(n = ...)‘ to see more rows
```

3.0.1. Para pensar: Un profesional de la estadística les recordaría el concepto de *ceteris paribus* en el contexto de regresión. ¿Es alrededor del vector $x^\star \in \mathcal{X}$ el que usamos para la interpretación o es alrededor del individuo promedio $\bar{x} \in \mathcal{X}$ el que usamos para interpretar el ajuste?

4. DESCOMPOSICIÓN SECUENCIAL ADITIVA

Recordemos que nuestras predicciones (en regresión) se pueden asociar a la función de regresión

$$\hat{f}(x) = \mathbb{E}[y|x]\,, \tag{1}$$

siempre y cuando utilicemos pérdida cuadrática para realizar el ajuste.

Por ejemplo en regresión lineal podemos calcular el valor esperado de la respuesta para una observación $x^\star$ por medio de

$$\mathbb{E}[y|x^\star] = \beta_0 + \beta_1 x_1^\star + \cdots + \beta_p x_p^\star\,, \tag{2}$$

donde los coeficientes β se ajustan por MCO.

En general, podemos calcular cómo cambia el valor esperado de y condicionado en que el atributo j tiene un valor de X_j por medio de

$$\iota(j, x^\star) = \mathbb{E}[Y|x^\star] - \mathbb{E}_{X_j}[\mathbb{E}[y|X_j]]\,, \tag{3}$$

donde $\iota(j, x^\star)$ mide la importancia de la variable j evaluada en el punto $x^\star$.

Por ejemplo, en regresión lineal tenemos la expresión particular de

$$\iota(j, x^\star) = \beta_j \left(x_j^\star - \mathbb{E}[X_j] \right)\,, \tag{4}$$

que podemos utilizar para expresar

$$\hat{f}(x^\star) = (\textsf{prediccion media}) + \sum_{j=1}^{p} \iota(j, x^\star)\,. \tag{5}$$

En general, cuando usamos modelos no lineales podemos pensar en

$$\iota(j, x^\star) = \mathbb{E}[Y|x^\star_{1:j}] - \mathbb{E}[y|x^\star_{1:j-1}]\,, \tag{6}$$

para preservar una suma telescópica como la anterior.

> Nota como el orden de los atributos afecta la descomposición de la predicción en términos individuales. Como es de esperarse es un mal resumen cuando hay interacción entre atributos.

Una vez que hemos decidido sobre cual individuo (observación o instancia) queremos hacer la expansión podemos usar `DALEX` para poder crear métricas de sensibilidad. Para esto utilizamos la función `predict_parts()`.

```
lm_breakdown <- predict_parts(
  explainer = explainer_lm,
  new_observation = mueble
)
lm_breakdown
```

```
                                contribution
linear model: intercept                2.665
linear model: width = 67              -0.162
linear model: category = 7             0.146
linear model: name = 568              -0.049
linear model: depth = 49               0.022
linear model: height = 102            -0.016
linear model: prediction               2.606
```

Lo mismo podemos hacer para nuestro modelo de `random forest`. En este tipo de tablas interpretamos cómo cada cambio va alejándonos de nuestro *intercepto* (la respuesta promedio de nuestro modelo predictivo).

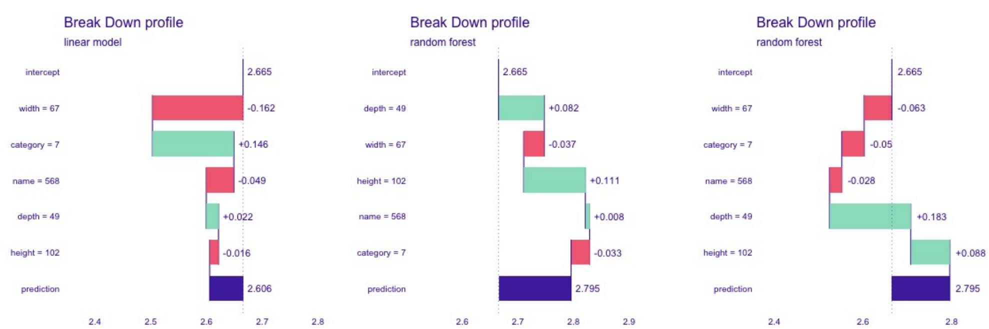
```r
rf_breakdown <- predict_parts(
  explainer = explainer_rf,
  new_observation = mueble
)
rf_breakdown
```

```
                                    contribution
random forest: intercept                   2.665
random forest: depth = 49                  0.082
random forest: width = 67                 -0.037
random forest: height = 102                0.111
random forest: name = 568                  0.008
random forest: category = 7               -0.033
random forest: prediction                  2.795
```

La interpretación cambia de acuerdo al orden en como se van presentando los cambios en los atributos y para esto podemos usar el modelo lineal como una heuristica de orden.

```r
rfor_breakdown <- predict_parts(
  explainer = explainer_rf,
  new_observation = mueble,
  order = lm_breakdown$variable_name
)
rfor_breakdown
```

```
                                    contribution
random forest: intercept                   2.665
random forest: width = 67                 -0.063
random forest: category = 7               -0.050
random forest: name = 568                 -0.028
random forest: depth = 49                  0.183
random forest: height = 102                0.088
random forest: prediction                  2.795
```



Podemos utilizar también la siguiente opción para explorar posibles contribuciones derivadas de interacciones.

```r
rfin_breakdown <- predict_parts(
  explainer = explainer_rf,
  new_observation = mueble,
  type = "break_down_interactions"
```

```
)
rfin_breakdown
```

```
                                             contribution
random forest: intercept                            2.665
random forest: depth = 49                           0.082
random forest: width:category = 67:7               -0.033
random forest: height = 102                         0.090
random forest: name = 568                          -0.009
random forest: prediction                           2.795
```

5. VALORES SHAP

La descomposición aditiva que hemos utilizado es sensible al orden en que hacemos la expansión de las atribuciones. Un problema latente si existen interacciones entre los predictores. Lo que haremos en esta sección es promediar bajo distintos órdenes las contribuciones individuales de los atributos.

Distintos órdenes, pueden presentar distintos resúmenes. Consideremos las siguientes gráficas de atribución tomadas de [1]. ¿Qué sucede con las variables de clase y tarifa?

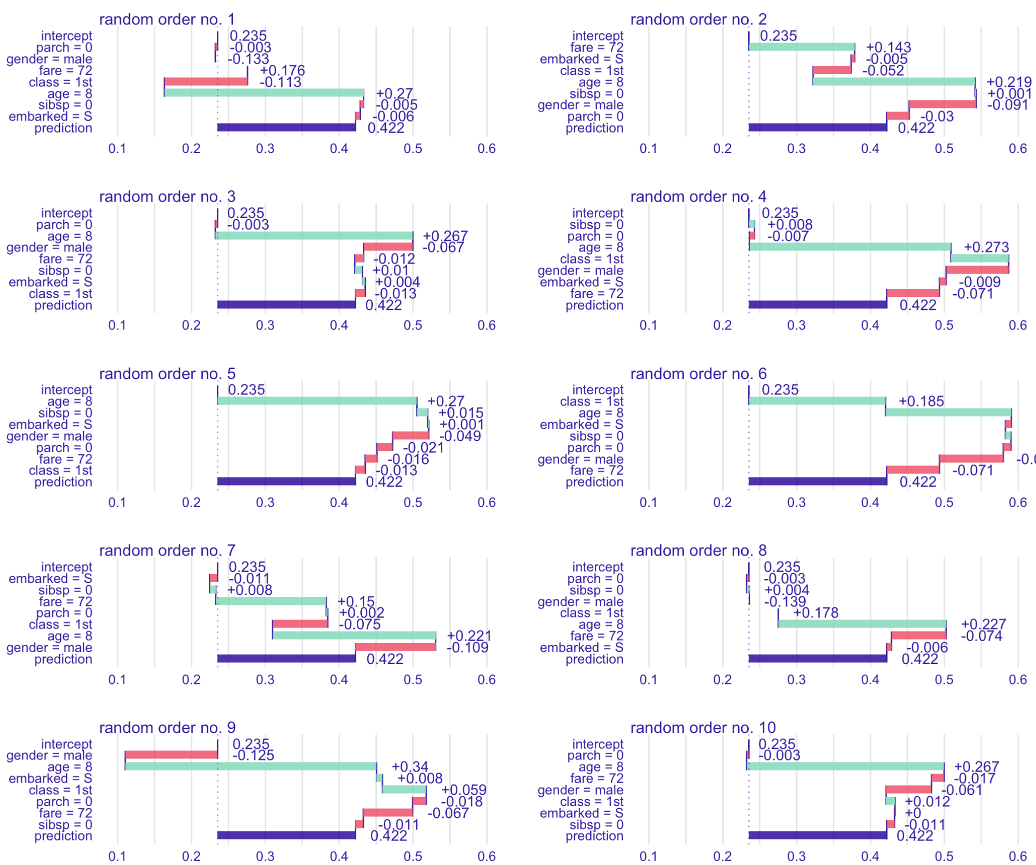
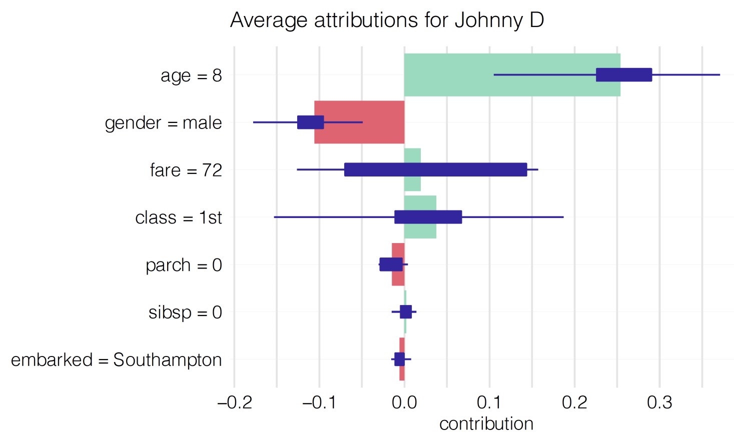


FIGURA 1. *Imagen tomada [1].*

Tomar promedios puede ayudar eliminar esta dependencia en el orden. Las barras (colores rojo y verde) muestran las contribuciones promedio, mientras que los gráficos de caja y brazos muestran la variabilidad a lo largo de los distintos órdenes.



5.1. Método

Usaremos los valores de Shapley como mecanismo explicativo de modelos. El nombre es por *SHapley Additive exPlanations* (`SHAP`) el cual usa teoría de juegos y el cual fue desarrollado para juegos cooperativos.

Lo que queremos es diagnosticar la magnitud de las contribuciones individuales a una tarea cooperativa.

5.1.1. ***Definición:*** Consideremos la predicción para $x^\star \in \mathbb{R}^p$ por parte del modelo predictivo $\hat{f}$. Consideremos que para un conjunto de índices J dejamos fijos los valores de acuerdo a las entradas de $x^\star$. Los índices que se encuentran en $J^c = \{1, \ldots, p\} \setminus J$ son variables aleatorias. Entonces denotamos por $\Delta^{\ell|J}(x^\star)$ el cambio en el valor esperado de las predicciones por incluir la ℓ-ésima variable en el conjunto fijo de valores.

5.1.2. Para pensar: Con la definición de arriba, argumenta que tenemos la siguiente igualdad

$$\Delta^{\ell|\emptyset}(x^\star) = \mathbb{E}[\hat{f}(X)|X_\ell = x_\ell^\star] - \mathbb{E}[\hat{f}(X)]\,. \tag{7}$$

5.1.3. Para pensar: Con la notación de arriba, ¿cómo escribirías nuestro término $\iota(j, x^\star)$?

5.1.4. ***Definición:*** El valor de Shapley está definido como

$$\phi(x^\star, j) = \frac{1}{p!} \sum_J \Delta^{j|\mathcal{I}(J,j)}(x^\star)\,, \tag{8}$$

donde $\mathcal{I}(J, j)$ denota todos los índices en J que se encuentran antes de j.

5.1.5. Propiedades: Los valores de Shapley gozan de las siguientes propiedades.

1. *Simetría*: si dos atributos satisfacen que bajo $S \subseteq \{1, \ldots, p\} \setminus \{j, k\}$

$$\Delta^{j|S}(x^\star) = \Delta^{k|S}(x^\star)\,, \tag{9}$$

entonces

$$\phi(x^\star, j) = \phi(x^\star, k)\,. \tag{10}$$

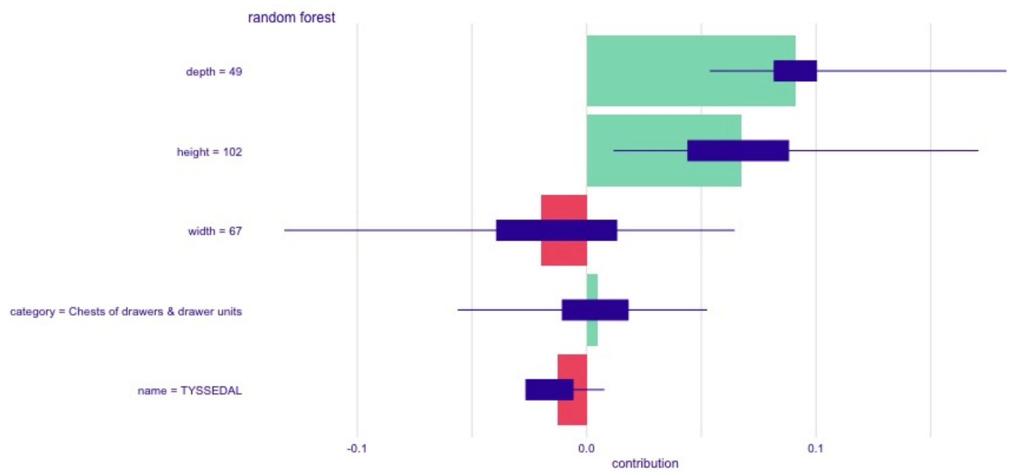
2. *Dummies*: si alguna variable no contribuye en predicciones entonces su valor de Shapley es 0.
3. *Aditividad*: Si $\hat{f} = \hat{f}_1 + \hat{f}_2$ entonces el valor de Shapley es la suma de los valores de Shapley para cada $\hat{f}_i$.

4. *Capacidad local*: La suma de los valores de Shapley es igual a las predicciones del modelo

$$\hat{f}(x^\star) - \mathbb{E}[\hat{f}(X)] = \sum_{j=1}^{p} \phi(x^\star, j)\,. \tag{11}$$

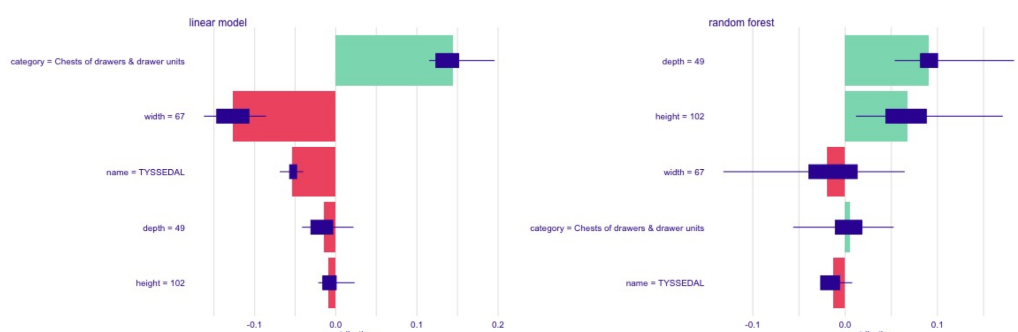
5.2. SHAP en IKEA.

```
set.seed(1801)
shap_mueble_rf ←
  predict_parts(
    explainer = explainer_rf,
    new_observation = mueble,
    type = "shap",
    B = 20
  )
```



Podemos comparar para cada modelo para diagnosticar en cada uno cómo cambian las predicciones.

```
set.seed(1801)
shap_mueble_lm ←
  predict_parts(
    explainer = explainer_lm,
    new_observation = mueble,
    type = "shap",
    B = 20
  )
```



6. EXPANSIONES LINEALES LOCALES

Podemos explorar la idea de aproximar un modelo predictivo sumamente complejo por uno altamente transparente. Esta es la idea detras de `LIME` (*Local Interpretable Model-agnostic Explanations*).

Por ejemplo, podemos utilizar la función `predict_surrogate()` de la siguiente manera. Para esto consideraremos otro modelo predictivo.

```
xgb_spec ←
  boost_tree(trees = 1000) ▷
  set_mode("regression") ▷
  set_engine("xgboost")

xgb_workflow ←
  workflow() ▷
  add_recipe(ranger_recipe ▷ step_dummy(all_nominal_predictors())) ▷
  add_model(xgb_spec)
```

```
ikea_xgb ← xgb_workflow ▷ fit(data = ikea_train)
```

Con la función de `explain_tidymodels()` construimos como antes el objeto necesario para interactuar.

```
explainer_xgb ←
  explain_tidymodels(
    ikea_xgb,
    data = ikea_train ▷ select(-price),
    y    = ikea_train ▷ pull(price),
    label = "boosted trees",
    verbose = FALSE
  )
```

Es importante definir unas opciones para interactuar con los modelos predictivos.

```
library(lime)
model_type.dalex_explainer ← DALEXtra::model_type.dalex_explainer
predict_model.dalex_explainer ← DALEXtra::predict_model.dalex_explainer
```

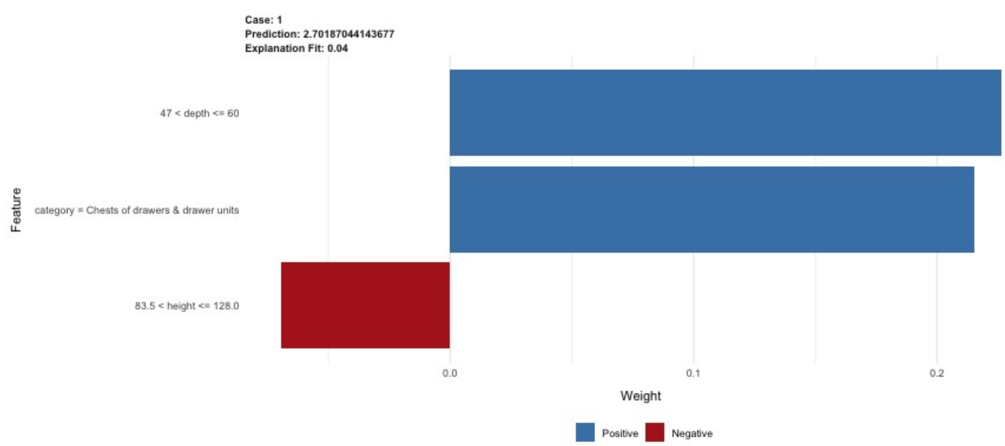
Con esto podemos ya usar la función `predict_surrogate()` que usaremos para crear nuestra aproximación local del modelo predictivo.

```
set.seed(108)
lime_mueble ← predict_surrogate(
  explainer = explainer_xgb,
  new_observation = mueble,
  n_features = 3,
  n_permutations = 500,
  type = "lime"
)

lime_mueble ▷ print(width = 85)
```

```
# A tibble: 3 × 11
  model_type case  model_r2 model_intercept model_prediction feature
  <chr>      <chr>    <dbl>           <dbl>            <dbl> <chr>
1 regression 1       0.0401            2.40             2.77 depth
2 regression 1       0.0401            2.40             2.77 category
3 regression 1       0.0401            2.40             2.77 height
  feature_value feature_weight feature_desc              data ...¹
    predi
          <dbl>          <dbl> <chr>                     <list>        <dbl>
1            49          0.226 47 < depth ≤ 60          <named list>    2.70
2             7          0.215 category = Chests of ...dra <named list>
   2.70
3           102        -0.0694 83.5 < height ≤ 128.0    <named list>    2.70
# ... with abbreviated variable name ¹prediction
```

El modelo lineal descrito arriba realiza predicciones por medio de

$$\hat{f}(x_{\text{mueble}}) = 2.40 + 0.226 \times x_1 + 0.215 \times x_2 - 0.07 \times x_3\,. \tag{12}$$



6.1. Construcción de LIME

La idea es sencilla. Tenemos un modelo predictivo $\hat{f}(x)$ que hemos ajustado con un conjunto de datos. Ahora lo que buscamos es un modelo *transparente* g tal que

$$\hat{g}(x) = \arg \min_{g \in \mathcal{G}^\star} \|g - \hat{f}\|^2 + \Omega(g)\,, \tag{13}$$

donde $\mathcal{G}^\star = \{g : \mathcal{N}(x^\star) \subset \mathcal{X} \to \mathcal{Y} | g$ es una función sencilla $\}$, $\|\cdot\|$ es una norma apropiada para modelos predictivos y $\Omega(\cdot)$ asigna una penalización por complejidad de g.

- Usualmente restringimos $\mathcal{G}^\star$ a ser un espacio de funciones lineales que incluye sólo d entradas (con $d \ll p$).
- Para generar puntos en $\mathcal{N}(x^\star)$ creamos perturbaciones a partir de $x^\star$ y evaluamos el modelo entrenado $\hat{f}$ en esos puntos.
- Ajustamos un modelo de regresión lineal para el conjunto de datos sintéticos.

6.2. Observaciones

- No hacemos supuestos sobre el modelo $\hat{f}$.
- La representación en menores dimensiones nos ayuda a mantener los atributos en un marco manejable.

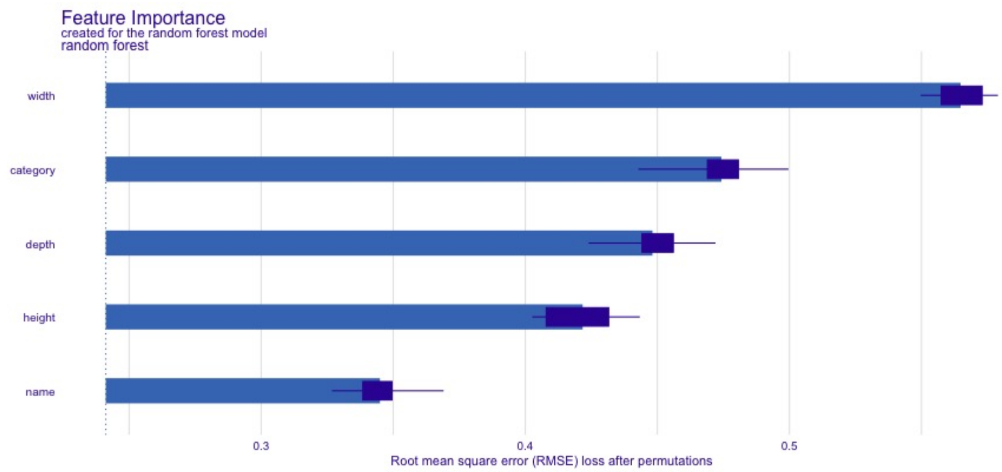
- La aproximación sólo es local.
- Puede y ha sido utilizado en modelos de texto y visión por computadora.
- Hay que recordar que sólo es una interpretación del modelo ajustado, no de los datos.
- Nosotros utilizamos sólo las funciones de `lime` pero también pueden utilizar `iml` o `localModels`, pueden ver mas aquí.

7. EXPLORACIÓN GLOBAL DE MODELOS

Los diagnósticos globales de los modelos los hemos presentado en secciones anteriores. Por ejemplo, para modelos basados en árboles, podemos usar los resúmenes de importancia de las variables (usando la librería `vip`). En `DALEX` también podemos usar la función `model_parts()` y definir la métrica que nos interesa.

```
set.seed(1804)
vip_rf ← model_parts(explainer_rf, loss_function = loss_root_mean_square)
```

De esta manera tenemos un gráfico familiar para nosotros que nos muestra la cómo afecta una variable la salida promedio del modelo $\hat{f}$.



7.1. Perfiles de dependencia parcial

Los resumenes anteriores son dependientes de la estructura del modelo mismo. Pero en muchas situaciones es de interés poder entender cómo afecta un atributo la salida esperada de las predicciones. Para esto, utilizaremos perfiles de dependencia parcial (PDP por sus siglas en inglés).

Este tipo de mecanismos nos ayudan a comparar grupos de datos bajo un mismo modelo predictivo, o incluso bajo distintos modelos.

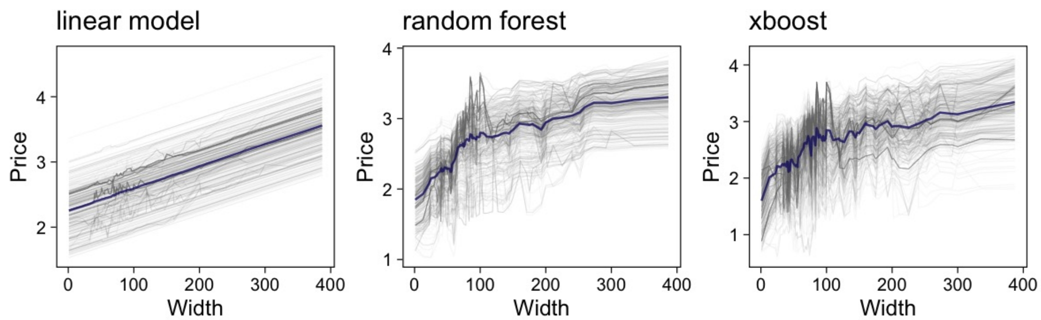
- Si dos modelos, con distinto grado de complejidad, tienen el mismo comportamiento entonces puede indicar que el riesgo de sobreajuste es reducido.
- Podemos utilizar las relaciones encontradas bajo un modelo mas complejo para hacer transformaciones que ayuden la capacidad predictiva de un modelo interpretable.
- Nos puede ayudar a entender las capacidades predictivas y comportamiento del modelo en regiones de pocas observaciones.

Alternativamente, podemos construir un gráfico que perfile el cambio de la salida del modelo predictivo utilizando un conjunto de observaciones. Es decir, construimos un diagnóstico global a partir de diagnósticos locales.

Este es el concepto de utilizar valores esperados individuales condicionados (`ICE`) o perfiles *ceteris paribus* (`CP`).

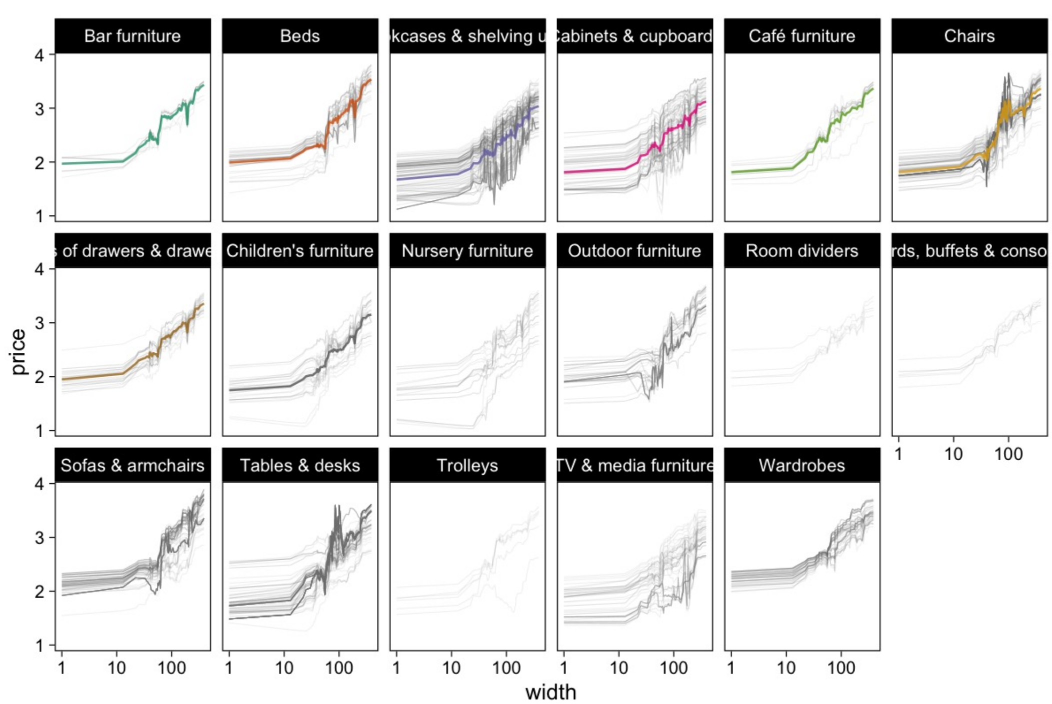
Para esto construiremos perfiles para 500 observaciones distintas en nuestro conjunto de entrenamiento y agregaremos los resultados por medio de la función `model_profile()`. En esta ocasión nos concentraremos en cómo se comporta la predicción como función parcial del ancho de los empaques (`width`).

```
set.seed(1805)
pdp_width_lm ← model_profile(explainer_lm, N = 500, variables = "width")
pdp_width_xgb ← model_profile(explainer_xgb, N = 500, variables = "width")
pdp_width_rf ← model_profile(explainer_rf, N = 500, variables = "width")
```



Adicionalmente, podemos utilizar la interacción de variables categóricas y continuas para explorar cómo afecta el cambio de un atributo en la salida del modelo en conjunto de datos.

```
set.seed(1806)
pdp_wcat ← model_profile(explainer_rf, N = 1000,
                         variables = "width",
                         groups = "category")
```



8. CONCLUSIONES

Las herramientas que presentamos en esta sección es un resumen muy breve de mas temas que se pueden explorar para entender lo que se puede explicar de un modelo.

La notación de esta sección ha sido consistente con la idea: una vez que tengo entrenado un modelo $\hat{f}$ entonces qué puedo decir sobre la relación entre los atributos y la respuesta. **No** es una relación de causalidad real entre atributos y respuesta dado que es una herramienta de post-procesamiento.

Los mecanismo estudiados en esta sección nos pueden ayudar a mejorar nuestro ciclo de modelado para incorporar mayor conocimiento en éste.

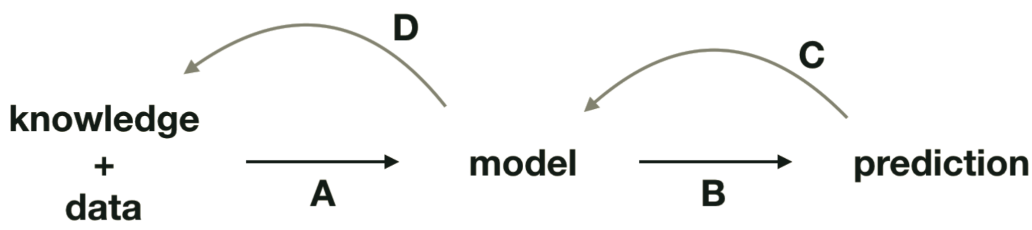


FIGURA 2. *Imagen tomada de [1].*

REFERENCIAS

[1] P. Biecek and T. Burzykowski. *Explanatory Model Analysis: Explore, Explain, and Examine Predictive Models*. Chapman & Hall/CRC Data Science Series. CRC Press, Boca Raton, first edition, 2021. ISBN 978-0-367-13559-1. 1, 7, 14

[2] C. Molnar. *Interpretable Machine Learning*. Lean Pub, 2020. 1