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.

```
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)
```

```
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

```
linear_spec ←
  linear_reg(penalty = 1e-3) ▷
  set_mode("regression") ▷
  set_engine("glmnet")

linear_workflow ←
  workflow() ▷
  add_recipe(linear_recipe) ▷
  add_model(linear_spec)
```

```
ranger_spec ←
  rand_forest(trees = 1000) ▷
  set_mode("regression") ▷
  set_engine("ranger")

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

```
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)
```

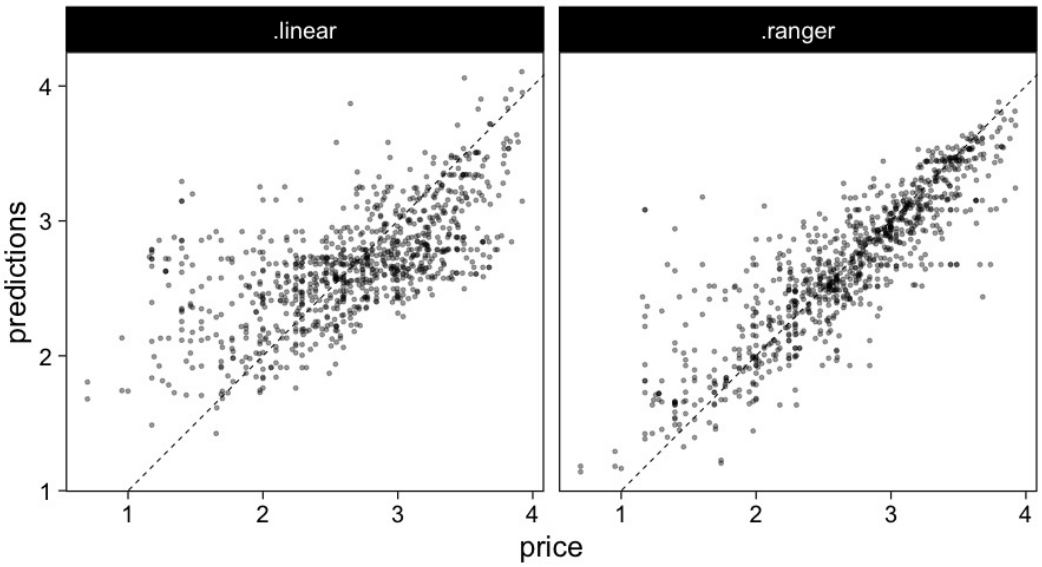
2. INTERPRETABILIDAD

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 $x_0 \in \mathcal{X} \subset \mathbb{R}^p$ en particular y exploramos la respuesta a partir de este punto. Por ejemplo, consideremos como x_0 la localidad donde queremos explorar el modelo.

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

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).

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

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 $0 \in \mathcal{X}$ 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. EXPANSIONES LINEALES LOCALES

Una vez que hemos decidido sobre cual individuo (observación o instancia) queremos hacer la expansión podemos usar `DALEX` para poder crear métricas de sensibilidad de cambios del valor promedio de la predicción derivado de cambios individuales en los atributos.

```
lm_breakdown ← predict_parts(explainer = explainer_lm, new_observation =
    mueble)
lm_breakdown
```

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

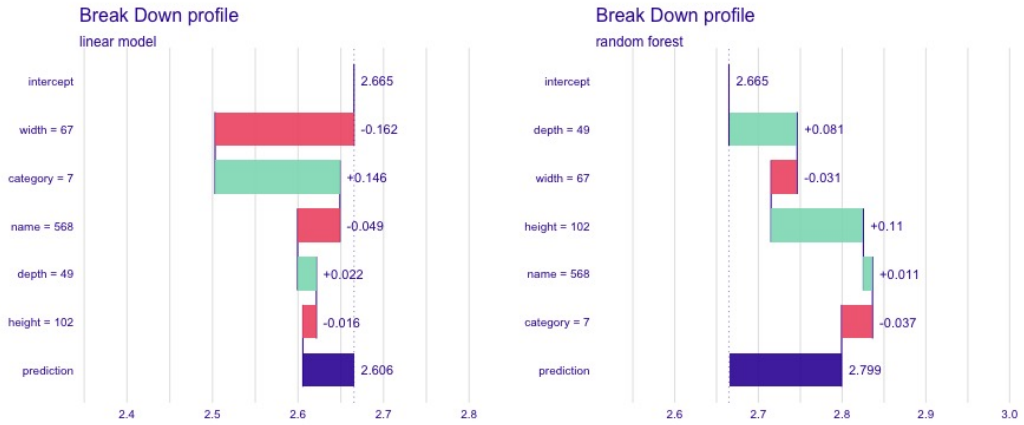
```
rf_breakdown ← predict_parts(explainer = explainer_rf, new_observation =
    mueble)
rf_breakdown
```

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.

```
predict_parts(explainer = explainer_rf,
              new_observation = mueble,
              order = lm_breakdown$variable_name)
```

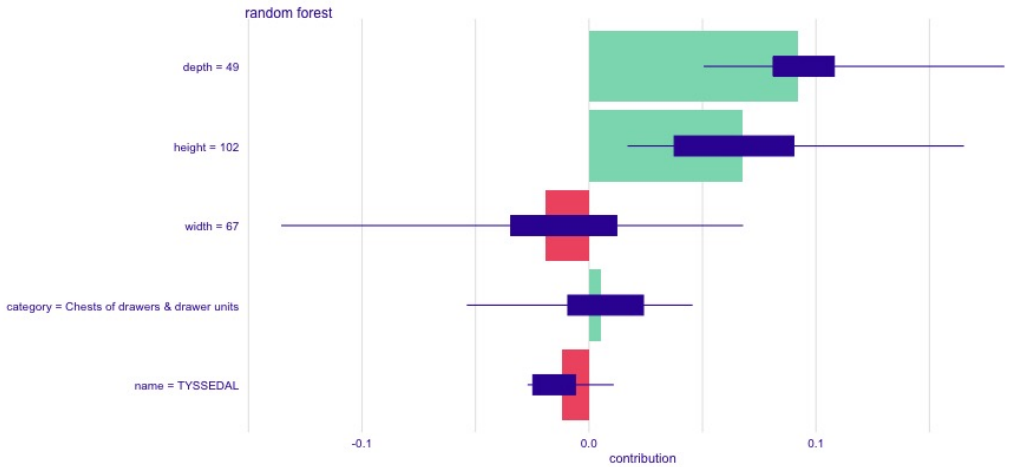
```
                            contribution
random forest: intercept           2.665
random forest: width = 67         -0.062
random forest: category = 7       -0.049
```

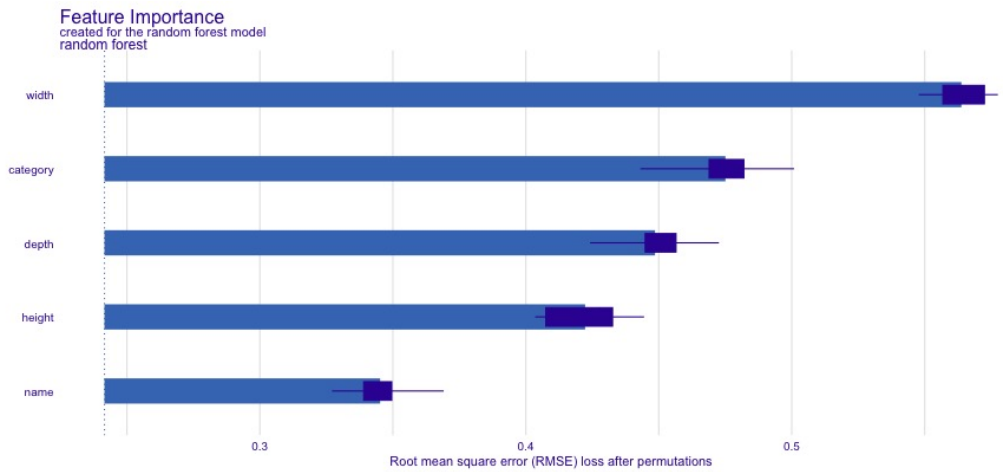
```
random forest: name = 568          -0.027
random forest: depth = 49           0.183
random forest: height = 102         0.090
random forest: prediction           2.799
```



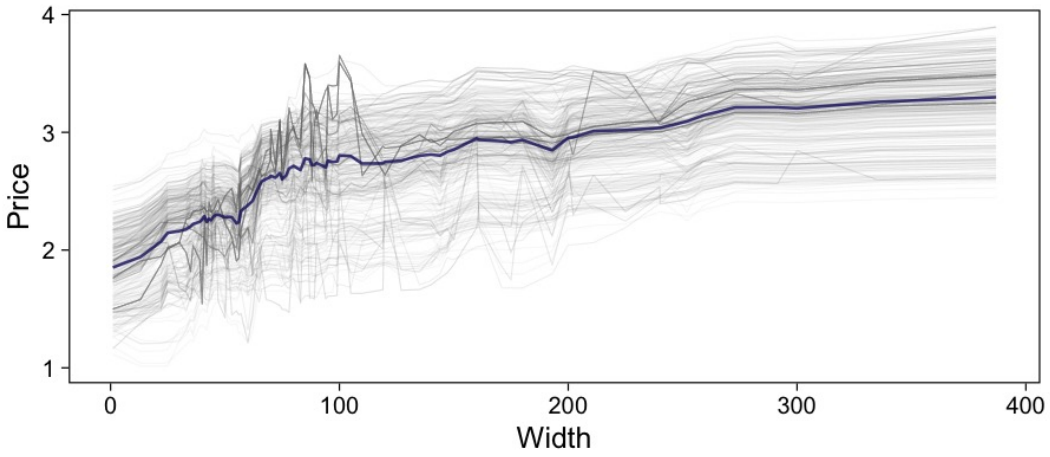
5. SHAP VALUES

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

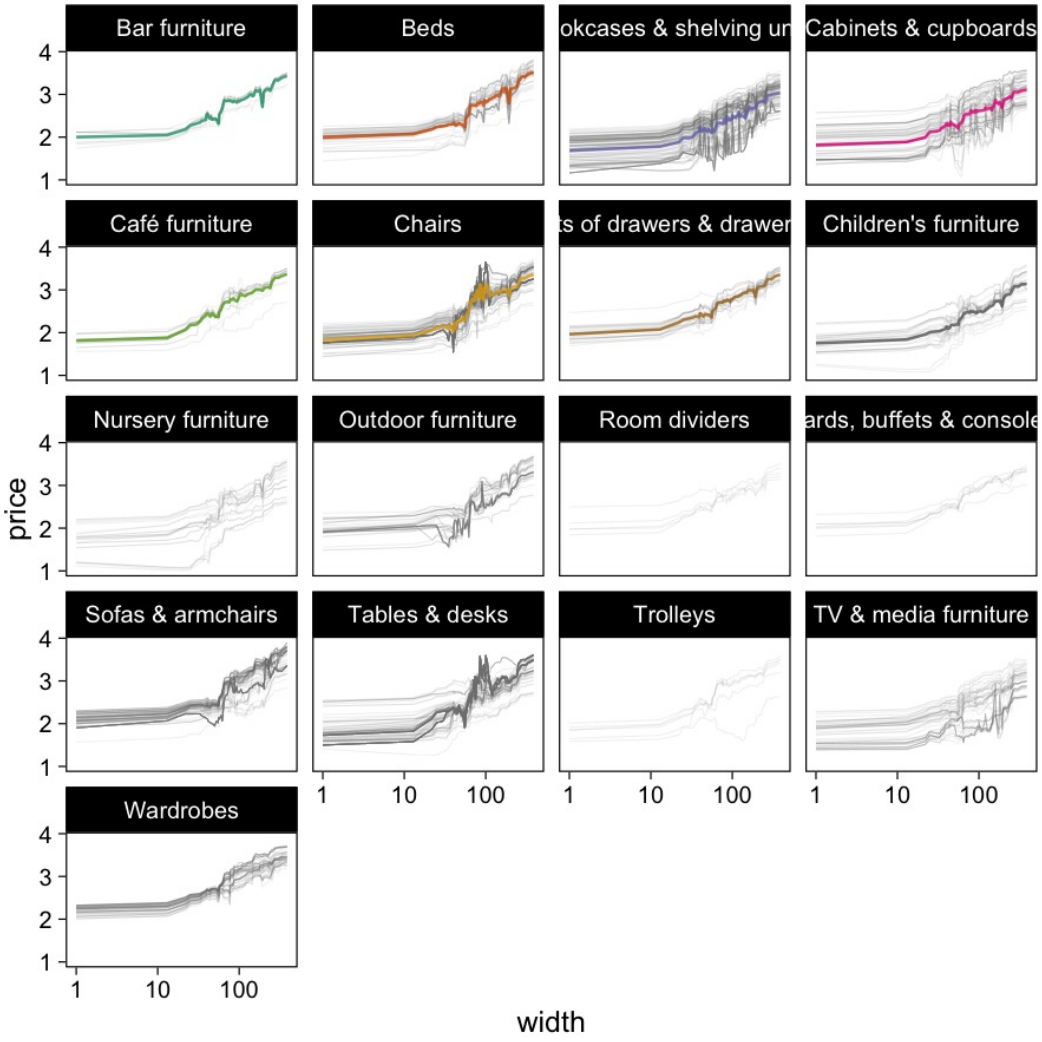





```
set.seed(1805)
pdp_width ← model_profile(explainer_rf, N = 500, variables = "width")
```




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



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

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