

# Machine learning en Stata usando H2O

Eduardo García Echeverri

Seminario web Stata, November 2025

# ¿Qué es el machine learning?

**Métodos estadísticos** que permiten modelar datos sin tener que suponer formas funcionales específicas en nuestros modelos.

Con ellos, podemos responder preguntas como

- ¿Qué clientes no pagarán su crédito con mayor probabilidad?
- ¿Qué factores explican el éxito de un procedimiento médico?
- ¿Qué genes explican la resistencia a una enfermedad?
- ¿Por qué es **Kylian Mbappé** tan altamente valorado?
- etc ...

# La integración de Stata con H2O

**H2O** es una plataforma de **machine learning** y análisis predictivo.

- Estableciendo la integración con H2O (una única vez)

Con su integración con Stata ahora podemos:

1. **Estimar** fácilmente modelos de alto desempeño predictivo
  - **Random forest** (RF)
  - **Gradient boosting machine** (GBM)
2. Evaluar el desempeño de los modelos
3. Calibrar hiperparámetros para seleccionar el modelo óptimo
4. Usar **explainability tools** para indagar más en los modelos

... todo esto sin dejar el ambiente familiar **Stata**.

# Agenda

- 1 Métodos de machine learning
  - Árboles de decisión
  - Random forest
  - Gradient boosting machine
- 2 Calibración de hiperparámetros
  - Método de three-way holdout
  - Cross-validation
- 3 Model explainability
- 4 Conclusión

# Agenda

- 1 Métodos de machine learning
  - Árboles de decisión
  - Random forest
  - Gradient boosting machine
- 2 Calibración de hiperparámetros
  - Método de three-way holdout
  - Cross-validation
- 3 Model explainability
- 4 Conclusión

# Agenda

- 1 Métodos de machine learning
  - Árboles de decisión
  - Random forest
  - Gradient boosting machine
- 2 Calibración de hiperparámetros
  - Método de three-way holdout
  - Cross-validation
- 3 Model explainability
- 4 Conclusión

# Árboles de decisión

Simple pero efectivo método de **supervised machine learning** para predecir respuestas (outcomes)

- binarias,
- categóricas,
- o continuas,

**particionando** los datos según los valores de los predictores.

Usados para construir métodos de ML más sofisticados:

- **Random forest** (RF);
- **Gradient boosting machine** (GBM).

# Respuestas discretas: Árboles de clasificación

Queremos predecir si un hongo es venenoso o comestible basados en su diámetro y su estación.

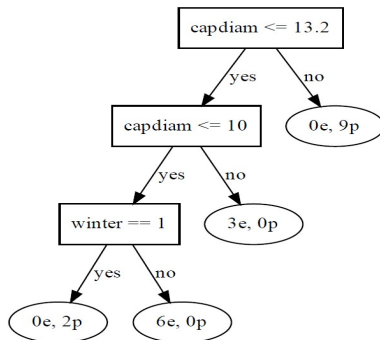
	capdiam	season	class
1.	7.3	s	e
2.	7.68	s	e
3.	8.4	s	e
4.	8.86	w	p
5.	9.03	s	e
6.	9.1	s	e
7.	9.59	w	p
8.	9.59	s	e
9.	10.42	w	e
10.	10.5	s	e

11.	12.85	s	e
12.	13.55	w	p
13.	14.07	w	p
14.	14.17	s	p
15.	14.64	s	p
16.	14.85	s	p
17.	14.86	s	p
18.	15.26	w	p
19.	15.34	s	p
20.	16.6	w	p



# Árboles de clasificación – Ejemplo

**Particionar** los datos con una serie de preguntas de sí o no:

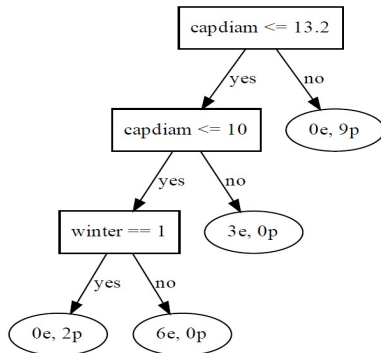


**Predicción basada en mayoría:**

Si  $\text{capdiam} = 8.32$  y  $\text{season} = w$ , predichimos  $\widehat{\text{class}} = p$

# Anatomía de los árboles de decisión

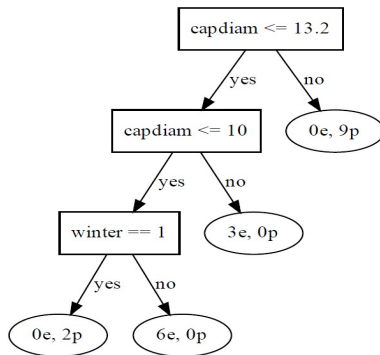
## Tipos de nodos:



- Nodos **parent** o nonterminal: datos son **partidos**. 1ro: **root**.
- Nodos **leaf** o terminal: datos no son **partidos**.

# Anatomía de los árboles de decisión

## Profundidad de un árbol:



**Definición:** Máximo número de particiones sucesivas (3).

## ¿Cómo encontrar el árbol que mejor se ajusta a los datos?

Queremos unos nodos leaf **tan homogéneos como sea posible**.

En cada leaf, queremos que

- **la mayoría** de los hongos sean **comestibles**;
- **o que la mayoría** sean **venenosos**.

¿Cómo hacemos esto **sistemáticamente**?

# Resolviendo un problema de minimización

Primero, escoger un **impurity measure**.

- Misclassification rate:  $1 - \max(\hat{p}_e, \hat{p}_p)$
- Gini index:  $2 * \hat{p}_e * \hat{p}_p$
- Cross entropy:  $-\hat{p}_e \ln(\hat{p}_e) - \hat{p}_p \ln(\hat{p}_p)$

**Ejemplo:**

(4e, 4p):  $\text{Gini} = 2 * 0.5 * 0.5 = 0.5$

(7e, 1p):  $\text{Gini} = 2 * 7/8 * 1/8 = 7/32 \approx 0.22$

Tomar el árbol que **minimiza impureza total ponderada**

- (**peso** de cada leaf = # de obs)
- sujeto **restricciones**

# Necesitamos estas restricciones para evitar overfitting

## Caso extremo de overfitting (sin restricciones):

Suponga que tenemos **1000** hongos **comestibles y venenosos**

- Todos los hongos tienen distinto diámetro.

⇒ Existe un árbol de **profundidad 1000** y **una obs por hoja** que **se ajusta a los datos perfectamente**.

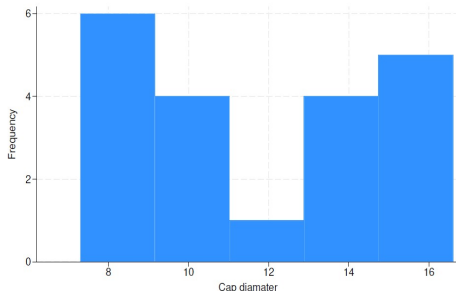
$\text{impurity} = 0$

Este árbol óptimo es **pésimo para predecir fuera de la muestra**.

- No es **generalizable**.

# Restricciones para evitar el overfitting

- Establecer una **profundidad máxima**: `maxdepth(2)`.
- Establecer **mínimo de observaciones** por hoja: `minobsleaf(3)`.
- **Restringir predictores** disponibles para particionar los datos.
- Requerir que variables continuas sean partidas **solo en cuantiles**:



## Algoritmo completo – optimización restringida

Todos los detalles del algoritmo **greedy** usado para estimar **árboles de decisión** lo puedes encontrar en **nuestra documentación**:

[\[H2OML\] Machine Learning Using H2O](#)



## Respuestas continuas: Árboles de regresión

Estimar **árboles de regresión** es **similar** a los árboles de clasificación:

1. **Predicción** es ahora la **respuesta promedio** en la hoja
2. **Impurity measure** es ahora  $RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ .

El algoritmo de optimización **greedy** procede **igual**.

# Agenda

## 1 Métodos de machine learning

- Árboles de decisión
- Random forest
- Gradient boosting machine

## 2 Calibración de hiperparámetros

- Método de three-way holdout
- Cross-validation

## 3 Model explainability

## 4 Conclusión

# Bagging

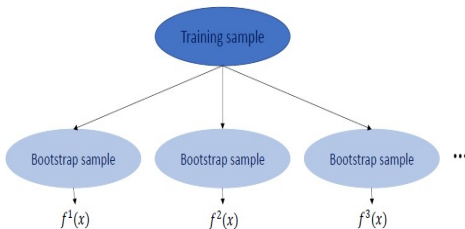
**Árboles de decisión son inestables y tener alta varianza.**

- Muy sensibles a pequeños cambios en datos.

Esto lleva a un alto **generalization error** (GE).

## Bagging:

Una manera de reducir varianza (y GE) es **bootstrap aggregation**:



# Random forest con $B$ árboles – Breiman (2001)

Para cada  $b = 1, 2, \dots B$ :

1. Tome una submuestra **bootstrap** del training data  $D_b$ 
  - (sin remplazamiento)
2. **Estime un árbol**  $T_b$  usando  $D_b$ 
  - 2.1 Aleatoriamente tome un **subconjunto** de  $m \leq p$  **predictores**.
  - 2.2 Seleccione el nodo, predictor, y punto de partición que **más decrezca el impurity measure**.
  - 2.3 **Particione** el nodo seleccionado.
  - 2.4 **Repita** hasta que **las restricciones lo permitan**.

# Random forest – Predicciones

Para una observación de testeo con predictores  $\mathbf{x}$ :

- **Regresión:**

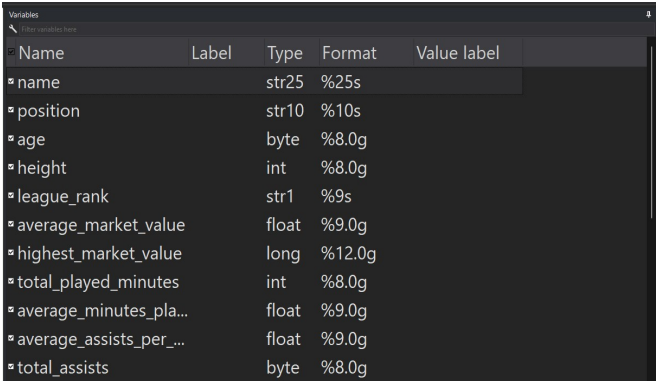
$$\hat{f}_{RF}(\mathbf{x}) = \frac{1}{B} \sum_b \hat{f}_b(\mathbf{x})$$

- **Clasificación:**

$$\hat{f}_{RF}(\mathbf{x}) = \text{Clase más predicha en } \hat{f}_1(\mathbf{x}), \dots, \hat{f}_B(\mathbf{x})$$

# Prediciendo el valor de mercado de jugadores de fútbol

**Datos:** Demográficos, métricas de desempeño en cancha, y valor de mercado para 1108 jugadores de fútbol 2023.



The screenshot shows the 'Variables' window in Stata, which lists the variables in the current dataset. The window has a search bar at the top and a table of variables below. The variables listed are: name, position, age, height, league\_rank, average\_market\_value, highest\_market\_value, total\_played\_minutes, average\_minutes\_pla..., average\_assists\_per..., and total\_assists. Each variable entry is preceded by a small square icon with a plus sign, indicating it is a variable in the dataset.

Name	Label	Type	Format	Value label
name		str25	%25s	
position		str10	%10s	
age		byte	%8.0g	
height		int	%8.0g	
league_rank		str1	%9s	
average_market_value		float	%9.0g	
highest_market_value		long	%12.0g	
total_played_minutes		int	%8.0g	
average_minutes_pla...		float	%9.0g	
average_assists_per_...		float	%9.0g	
total_assists		byte	%8.0g	

Fuente: Ahmed (2023) and Ocac et al (2023).

# Configuración de los datos y marcos en H2O

```
* H2O setup
use https://www.stata.com/users/lil/fifa, clear
gen ln_mkval = ln(average_market_value)
h2o init
_h2oframe put, into(fifa) current
_h2oframe toenum position nationality league_rank, replace
_h2oframe describe
_h2oframe split fifa, into(train test) split (0.8, 0.2) rseed(19)
_h2oframe change train
global predictors position age height nationality ///
league_rank average_minutes_played ///
average_goals_per_game average_assists_per_game ///
total_yellow_cards team_win_ratio
```

# Random forest – Implementación básica

## Estimación del random forest:

```
h2oml rfregress ln_mkval $predictors, h2orseed(19)
```

```
Progress (%): 0 100
```

```
Random forest regression using H2O
```

```
Response: ln_mkval
```

```
Frame:
```

```
Number of observations:
```

```
Training: train
```

```
Training = 877
```

```
Model parameters
```

```
Number of trees = 50
```

```
actual = 50
```

```
Tree depth:
```

```
Input max = 20
```

```
min = 17
```

```
avg = 18.9
```

```
max = 20
```

```
Min. obs. leaf split = 1
```

```
Pred. sampling value = -1
```

```
Sampling rate = .632
```

```
No. of bins cat. = 1,024
```

```
No. of bins root = 1,024
```

```
No. of bins cont. = 20
```

```
Min. split thresh. = .00001
```

```
Metric summary
```

Metric	Training
Deviance	.6068654
MSE	.6068654
RMSE	.7790156
RMSLE	.0480274
MAE	.6133148
R-squared	.6423362

**Guardar los resultados para después:** `h2omlest store rf_basic`



# Restricciones en los árboles

## Estimación del random forest:

```
h2oml rfregress ln_mkval $predictors, h2orseed(19) ///
predsampvalue(2) minobsleaf(5) maxdepth(10)
```

Progress (%): 0 1.9 100

Random forest regression using H2O

Response: **ln\_mkval**

Frame:

Training: **train**

Number of observations:

Training = **877**

Model parameters

Number of trees = **50**

actual = **50**

Tree depth:

Input max = **10**

min = **10**

avg = **10.0**

max = **10**

Min. obs. leaf split = **5**

Pred. sampling value = **2**

Sampling rate = **.632**

No. of bins cat. = **1,024**

No. of bins root = **1,024**

No. of bins cont. = **20**

Min. split thresh. = **.00001**

Metric summary

Metric	Training
Deviance	<b>.6795719</b>
MSE	<b>.6795719</b>
RMSE	<b>.8243615</b>
RMSLE	<b>.0508678</b>
MAE	<b>.651546</b>
R-squared	<b>.5994857</b>

# Restricciones en los árboles

## Estimación del random forest:

```
h2oml rfregress ln_mkval $predictors, h2orseed(19) ///
binsroot(4) binscont(4)
```

Progress (%): 0 3.9 100

Random forest regression using H2O

Response: ln\_mkval

Frame:

Training: train

Number of observations:

Training = 877

Model parameters

Number of trees = 50

actual = 50

Tree depth:

Input max = 20

min = 16

avg = 19.0

max = 20

Min. obs. leaf split = 1

Pred. sampling value = -1

Sampling rate = .632

No. of bins cat. = 1,024

No. of bins root = 4

No. of bins cont. = 4

Min. split thresh. = .00001

Metric summary

Metric	Training
Deviance	.7169355
MSE	.7169355
RMSE	.8467204
RMSLE	.0522315
MAE	.6616292
R-squared	.5774649

# Cambiar el número de árboles

## Estimación del random forest:

```
h2oml rfregress ln_mkval $predictors, h2orseed(19) ///
ntrees(100)
```

Progress (%): 0 100

Random forest regression using H2O

Response: **ln\_mkval**

Frame:

Training: **train**

Number of observations:

Training = **877**

Model parameters

Number of trees = **100**

actual = **100**

Tree depth:

Input max = **20**

min = **16**

avg = **18.9**

max = **20**

Min. obs. leaf split = **1**

Pred. sampling value = **-1**

Sampling rate = **.632**

No. of bins cat. = **1,024**

No. of bins root = **1,024**

No. of bins cont. = **20**

Min. split thresh. = **.00001**

### Metric summary

Metric	Training
Deviance	.5791324
MSE	.5791324
RMSE	.7610075
RMSLE	.0469173
MAE	.6012824
R-squared	.6586809

# Restaurar resultados del modelo básico y predecir

Hacer predicciones en observaciones que el modelo **no ha visto**.

```
. h2omltest restore rf_basic  
(results rf_basic are active now)  
  
. h2omlpredict ln_mkval_hat, frame(test)  
Progress (%): 0 100
```

```
. _h2oframe change test  
  
. _h2oframe list name ln_mkval ln_mkval_hat  
  
      name              ln_mkval  ln_mkval~t  
1 Abdelhamid Sabiri      14.731801  15.8144164  
2 Ajdin Hrustić          14.9141226  15.0215935  
3 Alban Lafont           16.405777  15.5534924  
4 Alessio Romagnoli      16.7599487  16.4266851  
5 Alex Meret             16.4365501  16.4052839  
6 Alexandre Lacazette    16.7599487  16.6609668  
7 Alexandre Letellier    12.8992195  14.4452275  
8 Alphonso Davies        18.0640049  16.6030938  
9 Amadou Diawara        15.8949518  15.3939597  
10 Andreas Christensen   17.3708591  16.5552003
```

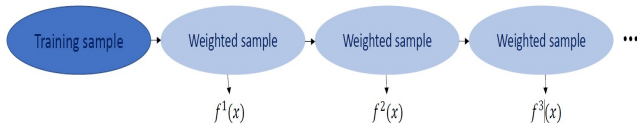
# Agenda

- 1 Métodos de machine learning
  - Árboles de decisión
  - Random forest
  - Gradient boosting machine
- 2 Calibración de hiperparámetros
  - Método de three-way holdout
  - Cross-validation
- 3 Model explainability
- 4 Conclusión

# Boosting

Podemos bajar el **GE** de los árboles de decisión **reduciendo sesgo**.  
Esto se puede hacer con un **ensemble procedure** diferente:

## Boosting:



# GBM con $M$ árboles – Regresión, Gaussian loss

## Algoritmo GBM (simplificado):

1. **Estimar árbol** para respuesta  $y$  con predictores  $\mathbf{x}$ 
  - Obtener **predicción**  $\hat{f}_1(\mathbf{x})$  y **error de predicción**  $\hat{e}_1$
2. **Estimar árbol** para el error  $\hat{e}_1$  con predictores  $\mathbf{x}$ 
  - Obtener **predicción**  $\hat{f}_2(\mathbf{x})$  y **error de predicción**  $\hat{e}_2$
3. **Estimar árbol** para el error  $\hat{e}_2$  con predictores  $\mathbf{x}$ 
  - Obtener **predicción**  $\hat{f}_3(\mathbf{x})$  y **error de predicción**  $\hat{e}_3$
4. ...**continuar** hasta obtener predicción  $\hat{f}_M(\mathbf{x})$ .

## Predicción:

$$\hat{f}_{GBM}(\mathbf{x}_i) = \sum_m \hat{f}_m(\mathbf{x}_i)$$

# GBM con $M$ árboles – Regresión, Gaussian loss

## Algoritmo GBM (simplificado):

1. Inicializar  $F_0(\mathbf{x}_i) = 0$  para  $i = 1, \dots, n$ .
2. Para cada  $m = 1, 2, \dots, M$ :
  - 2.1 Computar  $g_m(\mathbf{x}_i) = y_i - F_{m-1}(\mathbf{x}_i)$  para  $i = 1, \dots, n$ .
  - 2.2 Estimar **árbol**  $\hat{f}_m(\cdot)$  en el training data  $(\mathbf{x}_i, g_m(\mathbf{x}_i))_{i=1}^n$
  - 2.3 Actualizar  $F_{m+1}(\mathbf{x}_i) = F_m(\mathbf{x}_i) + \hat{f}_m(\mathbf{x}_i)$

**Predicción:** Agregar los *boosts*,

$$\hat{f}_{GBM}(\mathbf{x}_i) = \sum_{m=1}^M \hat{F}_m(\mathbf{x}_i)$$



# GBM con $M$ árboles – Regresión, Gaussian loss

## Algoritmo GBM

1. Escoger **learning rate**  $\alpha_0 \in (0, 1]$  y **decay rate**  $r \in (0, 1]$ .
  - **Evitar overfitting**
2. Inicializar  $F_0(\mathbf{x}_i) = 0$  para  $i = 1, \dots, n$ .
3. Para cada  $m = 1, 2, \dots, M$ :
  - 3.1 Computar  $g_m(\mathbf{x}_i) = y_i - F_{m-1}(\mathbf{x}_i)$  para  $i = 1, \dots, n$ .
  - 3.2 Estimar **árbol**  $\hat{f}_m(\cdot)$  en el training data  $(\mathbf{x}_i, g_m(\mathbf{x}_i))_{i=1}^n$
  - 3.3 Actualizar  $\alpha_m = \alpha_{m-1} * r$
  - 3.4 Actualizar  $F_m(\mathbf{x}_i) = F_{m-1}(\mathbf{x}_i) + \alpha_m \hat{f}_m(\mathbf{x}_i)$

**Predicción:** Agregar los *boosts*,

$$\hat{f}_{GBM}(\mathbf{x}_i) = \sum_{m=1}^M \hat{F}_m(\mathbf{x}_i)$$

# GBM – Implementación básica

## Estimar el GBM:

```
h2oml gbregress ln_mkval $predictors, h2orseed(19)
```

```
Progress (%): 0 100

Gradient boosting regression using H2O

Response: ln_mkval
Loss:      Gaussian
Frame:
  Training: test      Number of observations:
                        Training =    231

Model parameters

Number of trees      = 50      Learning rate        = .1
                    actual = 50 Learning rate decay = 1
Tree depth:         Input max = 5      Pred. sampling rate = 1
                    min = 5           Sampling rate        = 1
                    avg = 5.0         No. of bins cat.      = 1,024
                    max = 5           No. of bins root     = 1,024
Min. obs. leaf split = 10         No. of bins cont.     = 20
                                   Min. split thresh. = .00001
```

### Metric summary

Metric	Training
Deviance	.0696728
MSE	.0696728
RMSE	.263956
RMSLE	.0161893
MAE	.1871134
R-squared	.9516719

# Ajustar learning rate

## Estimar el GBM:

```
h2oml gbregress ln_mkval $predictors, h2orseed(19) ///
lrate(1)
```

Gradient boosting regression using H2O

Response: `ln_mkval`

Loss: Gaussian

Frame:

Training: `test`

Number of observations:

Training = 231

Model parameters

Number of trees = 50

actual = 50

Tree depth:

Input max = 5

min = 5

avg = 5.0

max = 5

Min. obs. leaf split = 10

Learning rate = 1

Learning rate decay = 1

Pred. sampling rate = 1

Sampling rate = 1

No. of bins cat. = 1,024

No. of bins root = 1,024

No. of bins cont. = 20

Min. split thresh. = .00001

Metric summary

Metric	Training
Deviance	.0000176
MSE	.0000176
RMSE	.0042
RMSLE	.0002628
MAE	.0030394
R-squared	.9999878

# Ajustar el learning rate y el decay rate

## Estimar GBM:

```
h2oml gbregress ln_mkval $predictors, h2orseed(19) ///
lrate(0.2) lratedecay(0.5)
```

Progress (%): 0 14.0 100

Gradient boosting regression using H2O

Response: **ln\_mkval**

Loss: Gaussian

Frame:

Number of observations:

Training: **test**

Training = **231**

Model parameters

Number of trees = **50**

Learning rate = **.2**

actual = **19**

Learning rate decay = **.5**

Tree depth:

Pred. sampling rate = **1**

Input max = **5**

Sampling rate = **1**

min = **5**

No. of bins cat. = **1,024**

avg = **5.0**

No. of bins root = **1,024**

max = **5**

No. of bins cont. = **20**

Min. obs. leaf split = **10**

Min. split thresh. = **.00001**

Metric summary

Metric	Training
Deviance	<b>.5741583</b>
MSE	<b>.5741583</b>
RMSE	<b>.7577323</b>
RMSLE	<b>.0464716</b>
MAE	<b>.6103677</b>
R-squared	<b>.6017385</b>

# Regresión de cuantil con GBM

GBM también nos permite predecir **cuantiles condicionales**.

```
h2oml gbregress ln_mkval $predictors, h2orseed(19) ///  
lrate(0.2) lratedecay(0.5) loss(quantile, alpha(0.25))  
h2omlpredict ln_mkval_hat_25, frame(test)
```

```
h2oml gbregress ln_mkval $predictors, h2orseed(19) ///  
lrate(0.2) lratedecay(0.5) loss(quantile, alpha(0.50))  
h2omlpredict ln_mkval_hat_50, frame(test)
```

```
h2oml gbregress ln_mkval $predictors, h2orseed(19) ///  
lrate(0.2) lratedecay(0.5) loss(quantile, alpha(0.75))  
h2omlpredict ln_mkval_hat_75, frame(test)
```

# Regresión de cuantil con GBM

Comparando **cuantiles predichos**:

```
. _h2oframe change test  
  
. _h2oframe list name ln_mkval_hat_25 ln_mkval_hat_50 ln_mkval_hat_75  
  
   name                ln_mkval~5  ln_mkval~0  ln_mkval~5  
1 Abdelhamid Sabiri      14.6062016  15.1758842  16.1193891  
2 Ajdin Hrustić          14.9407623  14.9720399  15.9597527  
3 Alban Lafont           14.4135028  15.2484461  15.9656129  
4 Alessio Romagnoli      15.1042727  16.3522333  16.8244258  
5 Alex Meret             15.0734366  15.4652781  16.5228535  
6 Alexandre Lacazette    15.4454864    15.28308  16.4480355  
7 Alexandre Letellier    14.2204296  15.1062561  15.9661407  
8 Alphonso Davies        15.3600749  16.1329853  17.0863218  
9 Amadou Diawara        14.8038634  15.8877665  16.4491213  
10 Andreas Christensen   15.3600749  16.0921205  17.219734
```

# Agenda

- 1 Métodos de machine learning
  - Árboles de decisión
  - Random forest
  - Gradient boosting machine
- 2 Calibración de hiperparámetros
  - Método de three-way holdout
  - Cross-validation
- 3 Model explainability
- 4 Conclusión

# Agenda

- 1 Métodos de machine learning
  - Árboles de decisión
  - Random forest
  - Gradient boosting machine
- 2 Calibración de hiperparámetros
  - Método de three-way holdout
  - Cross-validation
- 3 Model explainability
- 4 Conclusión



# Método Three-way holdout

## Pasos:

1. Aleatoriamente **divida los datos** en tres:
  - **training** frame,
  - **validation** frame,
  - **testing** frame.
2. Defina **cuadrícula** de configuraciones de **hiperparámetros**.
3. Para cada configuración:
  - **Estimar modelo en training** frame con la configuración.
  - **Evaluar la métrica de desempeño** en **validation** frame.
4. Seleccionar modelo con la mejor **métrica de desempeño**.
5. **Métricas desempeño** del modelo elegido en **testing frame**.

## Ejemplo: Calibrar learning rate y decay rates en GBM

**Data:** Abstención electoral y **presión social** (tratamiento).

- Correos prometiendo revelar la abstención a vecinos y amigos

<input checked="" type="checkbox"/> Name	Label	Type	Format	Value label
<input checked="" type="checkbox"/> gender	Gender	byte	%10.0g	genlbl
<input checked="" type="checkbox"/> g2000	Voted in 2000 general election	byte	%10.0g	yesno
<input checked="" type="checkbox"/> g2002	Voted in 2002 general election	byte	%10.0g	yesno
<input checked="" type="checkbox"/> g2004	Voted in 2004 general election	byte	%10.0g	yesno
<input checked="" type="checkbox"/> p2000	Voted in 2000 primary election	byte	%10.0g	yesno
<input checked="" type="checkbox"/> p2002	Voted in 2002 primary election	byte	%10.0g	yesno
<input checked="" type="checkbox"/> p2004	Voted in 2004 primary election	byte	%10.0g	yesno
<input checked="" type="checkbox"/> treatment	Treatment is assigned	byte	%10.0g	yesno
<input checked="" type="checkbox"/> voted	Voted	byte	%10.0g	yesno
<input checked="" type="checkbox"/> age	Age	int	%9.0g	

Fuente: Gerber et al (2008)

- └ Calibración de hiperparámetros
  - └ Método de three-way holdout

## Configuración de los datos y marcos en H2O

```
* H2O setup
use https://www.stata-press.com/data/r19/socialpressure, clear
describe
h2o init
_h2oframe put, into(social) current
_h2oframe split social, into(train valid test) split (0.6, 0.3, 0.1) rseed(19)
_h2oframe change train
global predictors gender g2000 g2002 p2000 p2002 p2004 treatment age
```

- Calibración de hiperparámetros
  - Método de three-way holdout

# Calibrando learning rate y decay rates en GBM

## Estimar el GBM:

```
gbbinclass voted $predictors, h2orseed(19) validframe(valid) ///
lrate(0.1(0.1)1) lratedecay(0.1(0.1)1)
```

Progress (%): 0 100

Gradient boosting binary classification using H2O

Response: voted

Loss: Bernoulli

Frame:                      Number of observations:  
 Training: train                      Training = 137,858  
 Validation: valid                      Validation = 68,509

Tuning information for hyperparameters

Method: Cartesian

Metric: Log loss

Hyperparameters	Grid values		
	Minimum	Maximum	Selected
Learning rate	.1	1	.2
Learning rate decay	.1	1	1

Model parameters

Number of trees	= 50	Learning rate	= .2
actual	= 27	Learning rate decay	= 1
Tree depth:		Pred. sampling rate	= 1
Input max	= 5	Sampling rate	= 1
min	= 0	No. of bins cat.	= 1,024
avg	= 4.8	No. of bins root	= 1,024
max	= 5	No. of bins cont.	= 20
Min. obs. leaf split	= 10	Min. split thresh.	= .00001

Metric summary

	Metric	Training	Validation
Log loss		.5697287	.5706692
	Mean class error	.3896366	.3921537
	AUC	.6775596	.6732603
Gini coefficient	AUCPR	.4769587	.4682633
		.3551191	.3465206
	MSE	.1935435	.1939787
	RMSE	.4399358	.4404302

# Métricas de desempeño en testing frame

```
h2omlestat metrics, frame(test)
```

```
. h2omlestat metrics, frame(test)

Performance metrics using H2O
Gradient boosting binary classification

Response: voted
Loss:      Bernoulli
Frame:     test

Number of observations = 23,094
```

Metric	test
Log loss	.5701197
Mean class error	.395349
AUC	.6737583
AUCPR	.4761043
Gini coefficient	.3475166
MSE	.1933754
RMSE	.4397447

# Agenda

- 1 Métodos de machine learning
  - Árboles de decisión
  - Random forest
  - Gradient boosting machine
- 2 Calibración de hiperparámetros
  - Método de three-way holdout
  - Cross-validation
- 3 Model explainability
- 4 Conclusión

# $K$ -fold cross-validation

1. Aleatoriamente **dividir los datos** en dos:
  - **training** frame,
  - **testing** frame.
2. **Dividir el training** frame en  $K$  folds.
3. Defina **cuadrícula** de configuraciones de **hiperparámetros**.
4. Para cada configuración:
  - 4.1 Para cada fold  $k = \{1, 2 \dots K\}$ ,
    - **Estimar modelo** en los otros  $K - 1$  folds con la configuración
    - **Evaluar la métrica de desempeño** del modelo en el fold  $k$
  - 4.2 **Promediar la métrica de desempeño** en los  $K$  folds
5. Tomar configuración con mejor **métrica promedio**.
6. **Estimar modelo** en training frame con configuración elegida.
7. **Métricas desempeño** del modelo elegido en **testing frame**.

# Calibrar número de árboles en el Random Forest

## Estimar el RF:

```
rfregress ln_mkval $predictors, h2orseed(19) ///
ntrees(50(5)450) cv(3)
```

Progress (%): 0 100

Random forest regression using H2O

Response: ln\_mkval

```
Frame:           Number of observations:
  Training: train           Training =    877
                        Cross-validation =    877
Cross-validation: Random   Number of folds    =    3
```

Tuning information for hyperparameters

Method: Cartesian

Metric: Deviance

Hyperparameters	Grid values		
	Minimum	Maximum	Selected
Number of trees	50	450	435

Model parameters

```
Number of trees    = 435
                  actual = 435

Tree depth:
  Input max = 20
           min = 16
           avg = 18.9
           max = 20
Min. obs. leaf split = 1

Pred. sampling value = -1
Sampling rate        = .632
No. of bins cat.     = 1,024
No. of bins root     = 1,024
No. of bins cont.    = 20
Min. split thresh.   = .00001
```

Metric summary

Metric	Cross-validation	
	Training	validation
Deviance	.5779539	.6239669
MSE	.5779539	.6239669
RMSE	.7602328	.7899158
RMSLE	.0469321	.0486451
MAE	.5998969	.6245509
R-squared	.6593755	.6322571



# Métricas desempeño en testing frame

```
h2omlestat metrics, frame(test)
```

```
. h2omlestat metrics, frame(test)
```

```
Performance metrics using H2O  
Random forest regression
```

```
Response: ln_mkval
```

```
Frame: test
```

```
Number of observations = 231
```

Metric	test
Deviance	.5532929
MSE	.5532929
RMSE	.7438366
RMSLE	.045243
MAE	.5982727
R-squared	.6162116

# Agenda

- 1 Métodos de machine learning
  - Árboles de decisión
  - Random forest
  - Gradient boosting machine
- 2 Calibración de hiperparámetros
  - Método de three-way holdout
  - Cross-validation
- 3 Model explainability
- 4 Conclusión

# Hacerle preguntas a nuestro model de machine learning

**Más allá de predecir**, yo también puedo estar interesado en saber

- ¿Cómo cambia la valoración de un jugador con su edad?
- ¿Es la edad más importante que la posición en la valoración de un jugador?
- ¿Por qué es **Kylian Mbappe** tan valioso?

Los explainable methods nos pueden ayudar con estas preguntas.

## Partial dependence plots

Grafican cómo la **respuesta esperada** cambia con un **predictor**

La **partial dependence** del predictor  $s$  en el punto  $x_s$

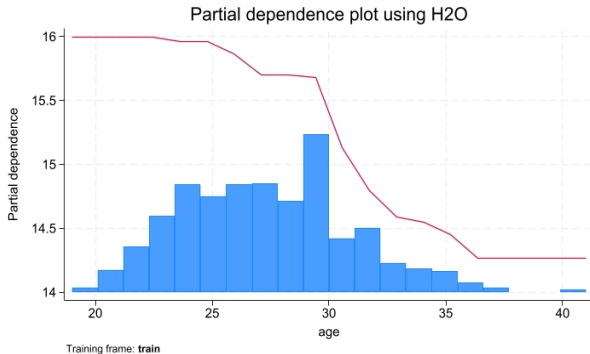
$$\hat{f}_{PD}(x_s) = \sum_{i=1}^n \hat{f}_{RF/GBM}(x_s, X_{-s,i})$$

donde,

$X_{-s,i}$ : todos los predictores excepto  $s$  en la observación  $i$ .

# Partial dependence plots

```
h2oml gbregress ln_mkval $predictors, h2orseed(19) cv(10)  
h2omlgraph pdp age
```



## Variable importance

Influencia de un predictor en el desempeño de un modelo.

En el árbol  $T$ , la **variable importance** del predictor  $k$  es

$$I_k^2(T) = \sum_{j \in N_{kT}} \iota_j^2$$

donde,

$N_{kT}$ : los **nodos asociados con el predictor**  $k$  en árbol  $T$

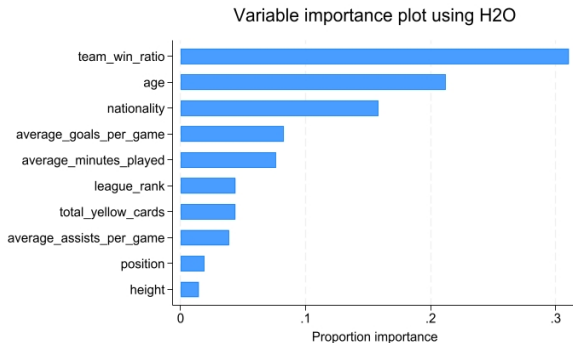
$\iota_j$ : Mejora relativa del **impurity measure** en el nodo  $j$

En un **ensemble method** con 100 árboles

$$I_k^2 = \frac{1}{100} \sum_{t=1}^{100} I_k^2(T_t)$$

# Variable importance

```
h2oml gbmregress ln_mkval $predictors, h2orseed(19) cv(10)  
h2omlgraph varimp
```



## Shapley additive explanation (SHAP) values

El valor de Shapley mide la **contribución de un predictor** a la predicción **de una observación**

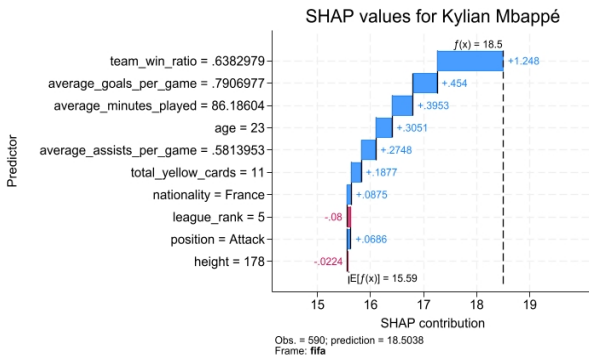
- Algoritmo completo

Muy útil para responder preguntas de **por qué**.



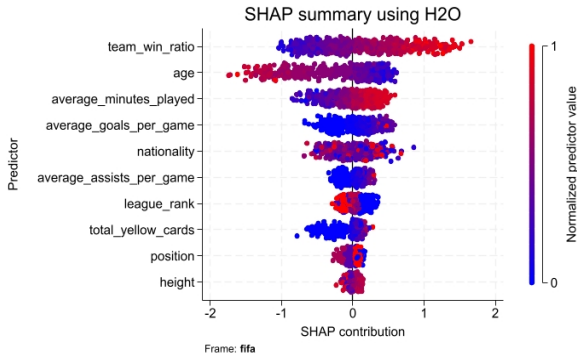
# ¿Por qué es Kylian Mbappé tan valioso?

```
h2oml gbregress ln_mkval $predictors, h2orseed(19) cv(10)
h2omlgraph shapvalues, obs(590) ///
title("SHAP values for Kylian Mbappé") frame(fifa)
```



# Resumir los SHAP values

```
h2oml gbmregress ln_mkval $predictors, h2orseed(19) cv(10)  
h2omlgraph shapsummary
```



# Agenda

- 1 Métodos de machine learning
  - Árboles de decisión
  - Random forest
  - Gradient boosting machine
- 2 Calibración de hiperparámetros
  - Método de three-way holdout
  - Cross-validation
- 3 Model explainability
- 4 Conclusión

# Conclusión

La integración de H2O con Stata nos permite:

1. **Estimar** fácilmente modelos de alto desempeño predictivo
  - **Random forest** (RF)
  - **Gradient boosting machine** (GBM);
2. Evaluar el desempeño del modelo;
3. Calibrar hiperparámetros para seleccionar el modelo óptimo
4. Usar **explainability tools** para indagar más en los modelos

# Más herramientas de machine learning en Stata

`cate`: Estimación de **Conditional average treatment effect**

- Usa **honest random forest** para estimar modelos auxiliares.

`lasso`: Least absolute shrinkage and selection operator.

- Regresión **lineal**
- Regresión **logit**
- Estimación de **efectos de tratamiento**
- Modelos de **supervivencia** ...

## Aprender más...

1. Comando `help`
  - Acceso a toda nuestra **documentación**.
2. [www.stata.com](http://www.stata.com)
  - Acceso a toda nuestra **documentación**;
  - **Frequently asked questions**.
3. [www.youtube.com/@statacorp/featured](http://www.youtube.com/@statacorp/featured)
4. [tech-support@stata.com](mailto:tech-support@stata.com)
  - **Preguntas específicas** sobre el software.

¡Gracias!

# Referencias

1. Breiman, L. 2001. Random forests. *Machine Learning* 45: 5–32.
2. Ahmed, M. 2023. Football players data. Kaggle.
3. Ocak, M., and H. Bal. 2023. Fifa-overall-prediction (commit 141de05).
4. Friedman, J. H., T. J. Hastie, and R. J. Tibshirani. 2000. Additive logistic regression: A statistical view of boosting. *Annals of Statistics* 28.
5. Gerber, A. S., D. P. Green, and C. W. Larimer. 2008. Social pressure and voter turnout: Evidence from a large-scale field experiment. *American Political Science Review* 102: 33–48.
6. Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Boca Raton, FL: Chapman and Hall/CRC.