

Machine learning Using Stata via H2O

Eduardo Garcia Echeverri

Stata webinar, November 2025

What is machine learning?

Statistical methods that allow us to model data without the need to assume specific functional forms in our models.

We can use them to answer questions like

- Which clients are more likely to default on their loan?
- What factors explain the success of a medical procedure?
- Which genes help explain resistance to a particular disease?
- What explains Kylian Mbappé's high market value?
- etc ...

Stata's integration with H2O

H2O is a machine learning and predictive analytics platform.

- H2O integration (one time) setup

With its integration into Stata you can now:

1. Estimate high-performance predictive models easily
 - Random forest (RF)
 - Gradient boosting machine (GBM);
2. evaluate model performance;
3. tune hyperparameters to optimally select models
4. use explainability tools to gain further insights from models

... all this without leaving your familiar Stata environment.

Outline

1 Machine learning methods

- Decision trees
- Random forest
- Gradient boosting machine

2 Hyperparameter tuning

- Three-way holdout method
- Cross-validation

3 Model explainability

4 Conclusion

Outline

- 1 Machine learning methods
 - Decision trees
 - Random forest
 - Gradient boosting machine
- 2 Hyperparameter tuning
 - Three-way holdout method
 - Cross-validation
- 3 Model explainability
- 4 Conclusion

Outline

1 Machine learning methods

- Decision trees
- Random forest
- Gradient boosting machine

2 Hyperparameter tuning

- Three-way holdout method
- Cross-validation

3 Model explainability

4 Conclusion

Decision trees

Simple but powerful supervised machine learning method to predict

- binary,
- categorical,
- or continuous outcomes (features),

by partitioning data based on the values of regressors (predictors).

Constitute the building blocks of more sophisticated ML methods:

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

Discrete outcomes: Classification tree

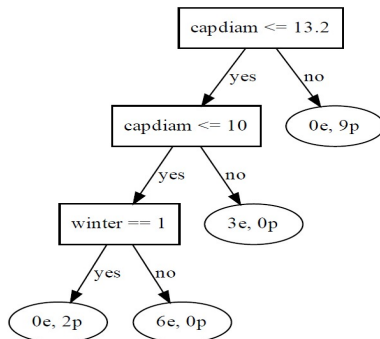
We want to predict if a mushroom is poisonous or edible based on its cap diameter and season.

	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

Classification tree – Example

Partition the data with a series of yes/no questions:

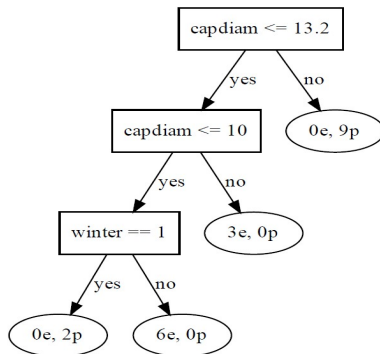


Prediction based on majority:

If $\text{capdiam} = 8.32$ and $\text{season} = w$, we predict $\widehat{\text{class}} = p$

Anatomy of decision trees

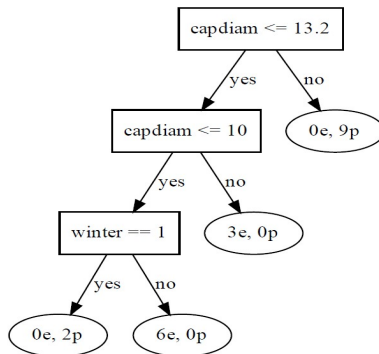
Types of nodes:



- Parent or nonterminal nodes: data is split. First is called *root*.
- Leaf or terminal nodes: data is not split.

Anatomy of decision trees

Depth of a tree:



Definition: Max. number of successive splits in a tree (3).

How do we find the tree that best fits the data?

We want our leaf nodes to be as homogeneous as possible. In each leaf, we want either

- most observations to be edible;
- or most observations to be poisonous.

How do we do this systematically?

Solving a minimization problem

First, choose an impurity measure.

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

Example:

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

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

Then, pick tree to minimize weighted total impurity of leafs

- (weight = # of obs)
- subject to constraints

We need these constraints to avoid overfitting

Extreme case of overfitting (no constraints):

Suppose we had data on 1000 edible and poisonous mushrooms and no two mushrooms have the same cap diameter.

⇒ There is a tree of depth 1000 and one observation per leaf that fits the data perfectly.

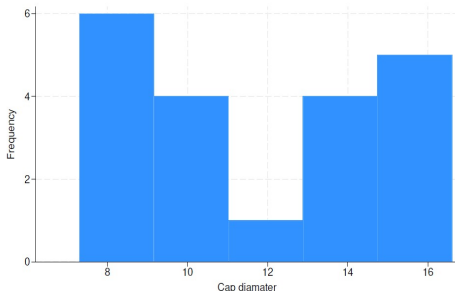
impurity = 0

This optimal tree, however, is useless for out-of-sample prediction.

- Not generalizable.

Constraints to avoid overfitting

- Set a maximum depth for the tree: `maxdepth(2)`.
- Set a minimum number of obs. per leaf: `minobsleaf(3)`.
- Restrict the set of predictors that can be used in each split.
- Require that continuous variables be split only by quantiles:



Full algorithm – constrained optimization

All the details of the algorithm can be found in our documentation:

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

Continuous outcomes: Regression tree

Growing a regression tree is similar to growing a classification tree:

1. Prediction is now the average response values in the leaf;
2. impurity measure is now $RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2$.

The constrained optimization algorithm proceeds in the same way.

Outline

1 Machine learning methods

- Decision trees
- Random forest
- Gradient boosting machine

2 Hyperparameter tuning

- Three-way holdout method
- Cross-validation

3 Model explainability

4 Conclusion

Bagging

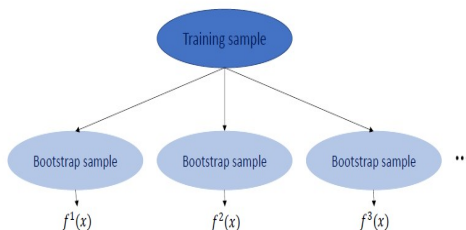
Decision trees tend to be unstable and have high variance.

- Very sensitive to small changes in the data.

This leads to high generalization error (GE).

Bagging:

A way to decrease variance (and thus GE) is bootstrap aggregation:



Random forest algorithm with B trees – Breiman (2001)

For $b = 1, 2, \dots B$:

1. Take a bootstrap subsample from training data D_b
 - (w/out replacement)
2. Fit a tree T_b in D_b
 - 2.1 Randomly choose a subset of $m \leq p$ predictors.
 - 2.2 Select node, predictor, and split point to decrease impurity measure the most.
 - 2.3 Split the selected node.
 - 2.4 Repeat until constraints are binding.

Random forest – Predictions

For a testing observation with predictors \mathbf{x} :

- Regression:

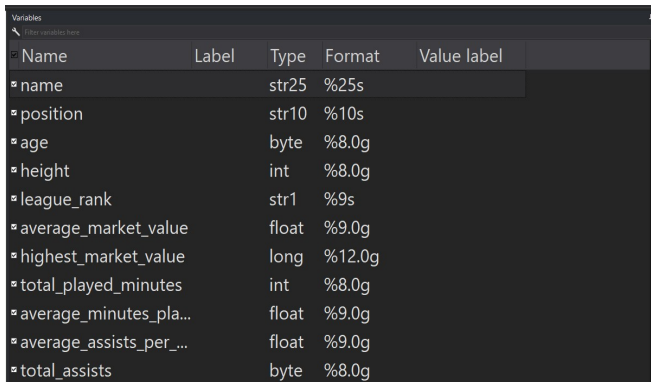
$$\hat{f}_{RF}(\mathbf{x}) = \frac{1}{B} \sum_b \hat{f}_b(\mathbf{x})$$

- Classification:

$$\hat{f}_{RF}(\mathbf{x}) = \text{most predicted class in } \hat{f}_1(\mathbf{x}), \dots, \hat{f}_B(\mathbf{x})$$

Predicting players' market value

Data: Demographics, on-field performance metrics, and market valuations for 1108 football players in 2023.



The screenshot shows the 'Variables' window in Stata, displaying a list of variables for 1108 football players in 2023. The window has a search bar at the top with the text 'Filter variables here'. Below the search bar is a table with columns: Name, Label, Type, Format, and Value label. The variables listed are:

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	

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

H2O setup

```
* 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 – Basic implementation

Fit the 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

Store results for later: `h2omlest store rf_basic`

Constraints on trees

Fit the 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	.6795719
MSE	.6795719
RMSE	.8243615
RMSLE	.0508678
MAE	.651546
R-squared	.5994857

Constraints on trees

Fit the 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

Set the number of trees

Fit the 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

Restoring results from basic model and making predictions

Make predictions on observations the model has never seen.

```
. 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_hat
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	16.552003

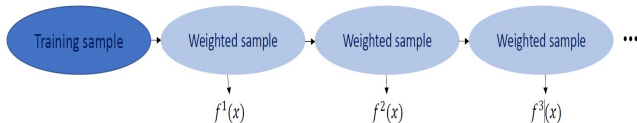
Outline

- 1 Machine learning methods
 - Decision trees
 - Random forest
 - Gradient boosting machine
- 2 Hyperparameter tuning
 - Three-way holdout method
 - Cross-validation
- 3 Model explainability
- 4 Conclusion

Boosting

We can also reduce the GE error of decision trees by reducing bias.
This can be achieved with a different ensemble procedure:

Boosting:



GBM with M trees – Regression problems, Gaussian loss

GBM algorithm (in words and simplified):

1. Fit regression tree for outcome y with predictors \mathbf{x}
 - Obtain the prediction $\hat{f}_1(\mathbf{x})$ and the prediction error \hat{e}_1
2. Fit regression tree for prediction error \hat{e}_1 with predictors \mathbf{x}
 - Obtain the prediction $\hat{f}_2(\mathbf{x})$ and the prediction error \hat{e}_2
3. Fit regression tree for prediction error \hat{e}_2 with predictors \mathbf{x}
 - Obtain the prediction $\hat{f}_3(\mathbf{x})$ and the prediction error \hat{e}_3
4. ...continue until you obtain the prediction $\hat{f}_M(\mathbf{x})$.

Prediction:

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

GBM with M trees – Regression problem, Gaussian loss

GBM algorithm (in math and simplified):

1. Set $F_0(\mathbf{x}_i) = 0$ for each $i = 1, \dots, n$.
2. For each $m = 1, 2 \dots M$:
 - 2.1 Compute $g_m(\mathbf{x}_i) = y_i - F_{m-1}(\mathbf{x}_i)$ for each $i = 1, \dots, n$.
 - 2.2 Fit a regression tree $\hat{f}_m(\cdot)$ on the training data $(\mathbf{x}_i, g_m(\mathbf{x}_i))_{i=1}^n$
 - 2.3 Update $F_{m+1}(\mathbf{x}_i) = F_m(\mathbf{x}_i) + \hat{f}_m(\mathbf{x}_i)$

Prediction: Add the *boosts*,

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

GBM with M trees – Regression problems, Gaussian loss

GBM algorithm (in math):

1. Choose learning rate $\alpha_0 \in (0, 1]$ and decay rate $r \in (0, 1]$.
 - Avoid overfitting
2. Set $F_0(\mathbf{x}_i) = 0$ for each $i = 1, \dots, n$.
3. For each $m = 1, 2, \dots, M$:
 - 3.1 Compute $g_m(\mathbf{x}_i) = y_i - F_{m-1}(\mathbf{x}_i)$ for each $i = 1, \dots, n$.
 - 3.2 Fit a regression tree $\hat{f}_m(\cdot)$ on the training data $(\mathbf{x}_i, g_m(\mathbf{x}_i))_{i=1}^n$
 - 3.3 Update $\alpha_m = \alpha_{m-1} * r$
 - 3.4 Update $F_m(\mathbf{x}_i) = F_{m-1}(\mathbf{x}_i) + \alpha_m \hat{f}_m(\mathbf{x}_i)$

Prediction: Add the *boosts*,

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

GBM – Basic implementation

Fit the 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:         = 5      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	.0696728
MSE	.0696728
RMSE	.263956
RMSLE	.0161893
MAE	.1871134
R-squared	.9516719

Adjusting the learning rate

Fit the 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

Adjusting learning and decay rates

Fit the 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	.5741583
MSE	.5741583
RMSE	.7577323
RMSLE	.0464716
MAE	.6103677
R-squared	.6017385

Quantile regression with GBM

GBM also allows us to predict conditional quantiles.

```
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)
```


Quantile regression with GBM

Comparing predicted quantiles:

```
. _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

Outline

- 1 Machine learning methods
 - Decision trees
 - Random forest
 - Gradient boosting machine
- 2 Hyperparameter tuning
 - Three-way holdout method
 - Cross-validation
- 3 Model explainability
- 4 Conclusion

Outline

- 1 Machine learning methods
 - Decision trees
 - Random forest
 - Gradient boosting machine
- 2 Hyperparameter tuning
 - Three-way holdout method
 - Cross-validation
- 3 Model explainability
- 4 Conclusion

Three-way holdout method

Steps:

1. Randomly partition the data into three:
 - training frame,
 - validation frame,
 - testing frame.
2. Define a grid of different hyperparameter configurations.
3. For each configuration:
 - Fit a model in the training frame with said configuration.
 - Evaluate performance metrics of the model in validation frame.
4. Select model that provides best performing metrics.
5. Get performance metrics of selected model in testing frame.

Example: Tuning learning and decay rates in GBM

Data: Voting behavior and social pressure (treatment).

- Mails promising to publicize turnout to household or neighbors

<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	

Source: Gerber et al (2008)

- └ Hyperparameter tuning
 - └ Three-way holdout method

H2O setup

```
* 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
```


- └ Hyperparameter tuning
 - └ Three-way holdout method

Tuning learning and decay rates in GBM

Fit the 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
	AUCPR	.4769587	.4682633
	Gini coefficient	.3551191	.3465206
	MSE	.1935435	.1939787
	RMSE	.4399358	.4404302

- └ Hyperparameter tuning
 - └ Three-way holdout method

Performance metrics in the 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

Outline

- 1 Machine learning methods
 - Decision trees
 - Random forest
 - Gradient boosting machine
- 2 Hyperparameter tuning
 - Three-way holdout method
 - Cross-validation
- 3 Model explainability
- 4 Conclusion

K -fold cross-validation

1. Randomly partition the data into two:
 - training frame,
 - testing frame.
2. Randomly split the training data into K folds.
3. Define a grid of different hyperparameter configurations.
4. For each configuration:
 - 4.1 For each fold $k = \{1, 2 \dots K\}$,
 - Fit a model in the other $K - 1$ folds with said configuration
 - Evaluate performance metric of the model in fold k
 - 4.2 Average performance metric over the K folds
5. Select configuration with best average performance metric.
6. Fit the model in training data with the selected configuration.
7. Get performance metrics of selected model in testing frame.

Fit the RF:

Progress (%): 0 100

Response: 1n mkval

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

Method: Cartesian

Metric: Deviance

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

```
Number of trees      = 435
                    actual = 435
```

```

Tree depth:                               Pred. sampling value = -1
      Input max = 20                      Sampling rate      = .632
      min = 16                            No. of bins cat.   = 1,024
      avg = 18.9                          No. of bins root  = 1,024
      max = 20                            No. of bins cont. = 20
Min. obs. leaf split = 1                 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

Performance metrics in the 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

Outline

- 1 Machine learning methods
 - Decision trees
 - Random forest
 - Gradient boosting machine
- 2 Hyperparameter tuning
 - Three-way holdout method
 - Cross-validation
- 3 Model explainability
- 4 Conclusion

Asking questions to our machine learning model

Beyond just predictions, I might be interested in knowing

- How does the market valuation of a player change with his age?
- Is age more important than position in determining a player's valuation?
- Why is Kylian Mbappe so valuable?

Explainable methods can help us with this questions.

Partial dependence plots

Plots how the expected response changes as a predictor changes

- While keeping all other predictors fixed.

Mathematically, the partial dependence for predictor s at point x_s

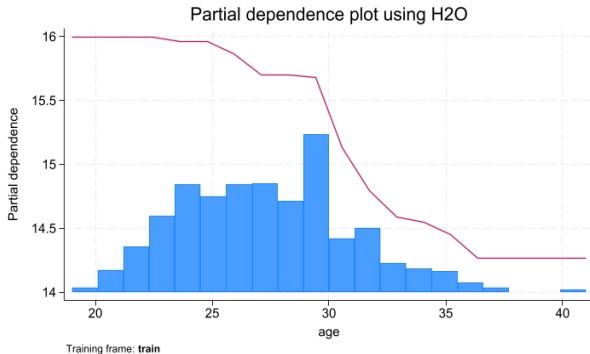
$$\hat{f}_{PD}(x_s) = \sum_{i=1}^n \hat{f}_{RF/GBM}(x_s, X_{-s,i})$$

where

$X_{-s,i}$: all predictors except s for observation i .

Partial dependence plots

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



Variable importance

Relative influence of a predictor on a model's performance. In tree T , the variable importance of predictor k is

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

where,

N_{kT} : set of nodes associated with predictor k in tree T

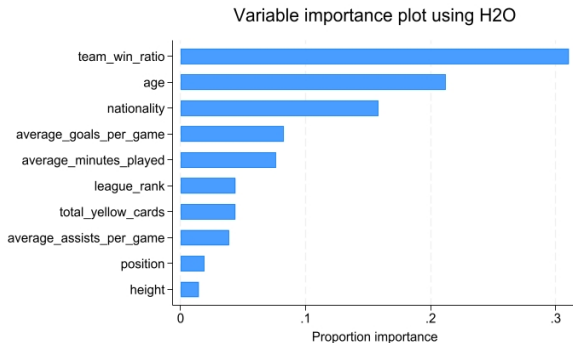
ι_j : Relative improvement of impurity measure at node j

In an ensemble method consisting of 100 trees

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

Variable importance

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



Shapley additive explanation (SHAP) values

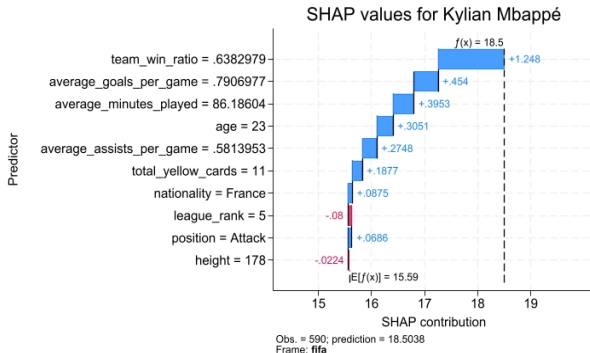
Shapley value measures the contribution of a predictor to the prediction for an observation

- Full algorithm

Very useful to answer questions of why.

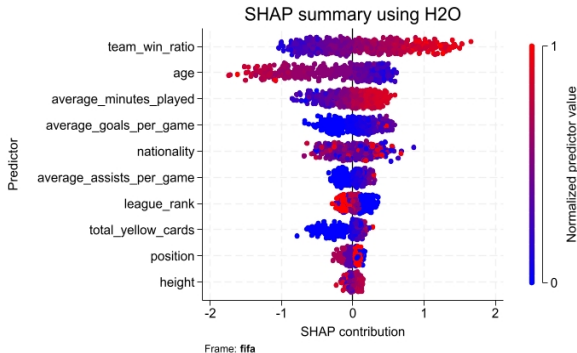
Why is Kylian Mbappé's valuation so high?

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



Summarizing the SHAP values

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



Outline

- 1 Machine learning methods
 - Decision trees
 - Random forest
 - Gradient boosting machine
- 2 Hyperparameter tuning
 - Three-way holdout method
 - Cross-validation
- 3 Model explainability
- 4 Conclusion

Conclusion

H2O's integration into Stata allows us to:

1. Estimate high-performance predictive models easily
 - Random forest (RF)
 - Gradient boosting machine (GBM);
2. evaluate model performance;
3. tune hyperparameters to optimally select models
4. use explainability tools to gain further insights from models

More machine learning features in Stata

cate: Conditional average treatment effect estimation

- Uses an honest random forest algorithm to estimate ancilliary models.

lasso: Least absolute shrinkage and selection operator.

- Linear regression
- Logit regression
- Treatment effect estimation
- Survival models...

Learning more...

1. `help` command
 - Access to all our **documentation**.
2. www.stata.com
 - Access to all our **documentation**;
 - **Frequently asked questions**.
3. www.youtube.com/@statacorp/featured
4. tech-support@stata.com
 - **Specific questions** about our software.

Thank you!

References

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.