H_2O.ai

Filters & Logical Operations

- Logical Operators
  - `h2o_frame[x].logical_negation()`
  - `h2o_frame[x] & h2o_frame[y]`
  - `h2o_frame[x] | h2o_frame[y]`

- Comparison Operators
  - `h2o_frame[x] {==, !=, <, <=, >=, >} value`
  - `h2o_frame[x] {==, !=, <, <=, >=, >} h2o_frame[y]`

- Logical Data Summaries
  - `h2o_frame[x].all()         # includes NAs`
  - `h2o_frame[x].any()         # includes NAs`
  - `h2o_frame[x].any_na_rm() # disregards NAs`

Filters & Logical Operations

```
test.ifelse(yes, no)
```

Arguments

`test`	A logical description of the condition to be met (>, <, =, etc…)
`yes`	The value to return if the condition is TRUE.
`no`	The value to return if the condition is FALSE.

Equivalent to `[y if t else n for t,y,n in zip(self,yes,no)]`

Note: Only numeric values can be tested, and only numeric results can be returned.

Filters & Logical Operations

- Logical Operators
  - `h2o_frame[x].logical_negation()`
  - `h2o_frame[x] & h2o_frame[y]`
  - `h2o_frame[x] | h2o_frame[y]`
- Comparison Operators
  - `h2o_frame[x] {==, !=, <, <=, >=, >} value`
  - `h2o_frame[x] {==, !=, <, <=, >=, >} h2o_frame[y]`
- Logical Data Summaries
  - `h2o_frame[x].all()         # includes NAs`
  - `h2o_frame[x].any()         # includes NAs`
  - `h2o_frame[x].any_na_rm() # disregards NAs`