encode_option — Encoding schemes for categorical predictors

Description Syntax Option Reference Also see

Description

The `encode()` option specifies the encoding scheme to use for categorical predictors in machine learning models implemented by the `h2oml` *gbm* and `h2oml` *rf* commands. The encoding scheme determines how a machine learning method splits categorical predictors, which can affect model performance. This entry introduces encoding schemes for categorical predictors that are available in H2O and that may be selected via the `encode()` option. For more details, see https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/algo-params/categorical_encoding.html. For an introduction to predictor encoding, see Kuhn and Johnson (2020).

Syntax

command `...[ , ... encode(`*encode_type*`) ... ]`

command is one of `h2oml gbregress`, `h2oml gbbinclass`, `h2oml gbmulticlass`, `h2oml rfregress`, `h2oml rfbinclass`, or `h2oml rfmulticlass`.

encode_type	Description
`enum`	map labels of categorical predictors to integers; the default
`enumfreq`	map labels for 10 most frequent levels of each categorical predictor to integers; combine all other levels to an 11th integer
`onehotexplicit`	generate a binary predictor for each level of each categorical predictor
`binary`	convert levels of categorical predictors into binary digit representation
`eigen`	generate new predictors for a categorical predictor based on eigenvalues of the one-hot-encoding matrix
`label`	map labels of categorical predictors to integers; ensure order is preserved
`sortbyresponse`	map levels of categorical predictors to integers; order by average response within levels

Option

`encode(`*encode_type*`)` specifies the H2O encoding scheme to be used for categorical predictors. The selected encoding scheme does not modify the existing H2O frame. The predictors generated by the encoding scheme are entirely virtual; they are created at the algorithmic level rather than at the memory level. Therefore, they cannot be accessed directly. However, it can be helpful to think of the predictors as physically generated.

encode_type may be one of `enum`, `enumfreq`, `onehotexplicit`, `binary`, `eigen`, `label`, or `sortbyresponse`.

`enum` maps the labels of categorical predictors into integers, which are then used by the machine learning method for splitting decisions. For example, if a categorical predictor has the levels {cat, dog, horse, cow, turtle, unicorn}, then the `enum` option maps those levels to $\{0, 1, 2, 3, 4, 5\}$. The machine learning method may split the levels as $\{0, 2, 4\}$ and $\{1, 3, 5\}$. This is the default scheme.

`enumfreq` reduces the levels of each categorical predictor to the 10 most frequent levels. All other levels, if any, are grouped into a separate 11th level. This option is useful when the number of levels of categorical predictors is very large and some of the categories are very rare and might not provide useful information. In reporting postestimation results, this option adds suffix `.top_10_levels` to the names of the categorical predictors.

`onehotexplicit` internally generates a new binary predictor for each level of each categorical predictor. For example, if a categorical predictor has the observations {cat, dog, cat, cat, dog, unicorn, unicorn} then three new predictors will be generated with cat $= \{1, 0, 1, 1, 0, 0, 0\}$, dog $= \{0, 1, 0, 0, 1, 0, 0\}$, and unicorn $= \{0, 0, 0, 0, 0, 1, 1\}$. This is the most well-known encoding scheme in machine learning. In reporting postestimation results, this option adds suffix *.level* to the names of the categorical predictors, where *level* corresponds to the class of the predictor, including missing values, which are labeled as class `NA`.

`binary` converts the levels of each categorical predictor into binary digits, with each binary digit representing a new separate predictor. The encoding process begins by assigning a numeric value to each level of the categorical predictor, starting from 1. For example, the observations of the categorical predictor {cat, dog, cat, cat, dog, unicorn, unicorn} are converted to the sequence $\{1, 2, 1, 1, 2, 3, 3\}$. The binary code for each numeric value is then determined, with 1 being represented by 01, 2 by 10, and 3 by 11. Then the observations are converted to the binary code $\{01, 10, 01, 01, 10, 11, 11\}$, with the digits of the binary number forming separate predictors. In our example, there are two new encoded predictors: $\{0, 1, 0, 0, 1, 1, 1\}$ and $\{1, 0, 1, 1, 0, 1, 1\}$. Binary encoding is useful when the number of categories is very large. However, H2O limits the number of new encoded predictors to 32. In reporting postestimation results, this option adds suffix `:`# to the names of the categorical predictors, where # varies from 1 to the maximum number of newly generated predictors. In the above example, the maximum number of generated predictors is 2.

`eigen` generates k new projected predictors per categorical predictor, such that the projections of the matrix generated from one-hot-encoding of the categorical predictor is in k-dimensional eigenspace. Currently, H2O uses $k = 1$. For details, see https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/algo-params/categorical_encoding.html. In reporting postestimation results, this option adds suffix `.Eigen` to the names of the categorical predictors.

`label` maps the labels of categorical predictors into integers, ensuring that the ordinal nature of each encoded predictor is preserved. For example, if an encoded predictor has values $\{0, 1, 2, 3, 4, 5\}$, a possible split could be $\{0, 1, 2\}$ and $\{3, 4, 5\}$, but not $\{0, 3, 4\}$ and $\{1, 2, 5\}$.

`sortbyresponse` maps the levels of categorical predictors into integers according to the ascending order of the average value of the response for each level. Thus, the level with the lowest average response value is assigned to 0, the level with second-lowest average response is assigned to 1, and so on.

Reference

Kuhn, M., and K. Johnson. 2020. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Boca Raton, FL: CRC Press.

Also see

[H2OML] **h2oml** — Introduction to commands for Stata integration with H2O machine learning

[H2OML] ***h2oml gbm*** — Gradient boosting machine for regression and classification

[H2OML] ***h2oml rf*** — Random forest for regression and classification