categorical_encoding

- Available in: GBM, DRF, Deep Learning, K-Means, Aggregator, XGBoost, Isolation Forest
- Hyperparameter: yes

Description

This option specifies the encoding scheme to use for handling categorical features. Available schemes include the following:

GBM/DRF/Isolation Forest

- `auto` or `AUTO`: Allow the algorithm to decide (default). For GBM, DRF, and Isolation Forest, the algorithm will perform **Enum** encoding when `auto` option is specified.
- `enum` or `Enum`: Leave the dataset as is, internally map the strings to integers, and use these integers to make splits - either via ordinal nature when `nbins_cats` is too small to resolve all levels or via bitsets that do a perfect group split. Each category is a separate category; its name (or number) is irrelevant. For example, after the strings are mapped to integers for **Enum**, you can split {0, 1, 2, 3, 4, 5} as {0, 4, 5} and {1, 2, 3}.
- `enum_limited` or `EnumLimited`: Automatically reduce categorical levels to the most prevalent ones during training and only keep the **T** (10) most frequent levels.
- `one_hot_explicit` or `OneHotExplicit`: N+1 new columns for categorical features with N levels
- `binary` or `Binary`: No more than 32 columns per categorical feature
- `eigen` or `Eigen`: *k* columns per categorical feature, keeping projections of one-hot-encoded matrix onto *k*-dim eigen space only
- `label_encoder` or `LabelEncoder`: Convert every **enum** into the integer of its index (for example, level 0 -> 0, level 1 -> 1, etc.) The categories are lexicographically mapped to numbers and lose their categorical nature, becoming ordinal. After the strings are mapped to integers, you can split {0, 1, 2, 3, 4, 5} as {0, 1, 2} and {3, 4, 5}.
- `sort_by_response` or `SortByResponse`:(GBM/DRF) Reorders the levels by the mean response (for example, the level with lowest response -> 0, the level with second-lowest response -> 1, etc.). This is useful in GBM/DRF, for example, when you have more levels than `nbins_cats`, and where the top level splits now have a chance at separating the data with a split. Note that this requires a specified response column.

Deep Learning

- `auto` or `AUTO`: Allow the algorithm to decide. For Deep Learning, the algorithm will perform One Hot Internal encoding when `auto` is specified.
- `one_hot_internal` or `OneHotInternal`: Leave the dataset as is. This internally expands each row via one-hot encoding on the fly. (default)
- `binary` or `Binary`: No more than 32 columns per categorical feature
- `eigen` or `Eigen`: *k* columns per categorical feature, keeping projections of one-hot-encoded matrix onto *k*-dim eigen space only
- `enum_limited` or `EnumLimited`: Automatically reduce categorical levels to the most prevalent ones during training and only keep the **T** (10) most frequent levels.
- `label_encoder` or `LabelEncoder`: Convert every enum into the integer of its index (for example, level 0 -> 0, level 1 -> 1, etc.). The categories are lexicographically mapped to numbers and lose their categorical nature, becoming ordinal. After the strings are mapped to integers, you can split {0, 1, 2, 3, 4, 5} as {0, 1, 2} and {3, 4, 5}. This is useful for keeping the number of columns small for XGBoost or DeepLearning, where the algorithm otherwise perform ExplicitOneHotEncoding.
- `sort_by_response` or `SortByResponse`: Reorders the levels by the mean response (for example, the level with lowest response -> 0, the level with second-lowest response -> 1, etc.). Note that this requires a specified response column.

Note: For Deep Learning, this value defaults to `one_hot_internal`. Similarly, if `auto` is specified, then the algorithm performs `one_hot_internal` encoding.

Aggregator

- `auto` or `AUTO`: Allow the algorithm to decide. For Aggregator, the algorithm will perform One Hot Internal encoding when `auto` is specified.
- `one_hot_internal` or `OneHotInternal`: Leave the dataset as is. This internally expands each row via one-hot encoding on the fly. (default)
- `binary` or `Binary`: No more than 32 columns per categorical feature
- `eigen` or `Eigen`: *k* columns per categorical feature, keeping projections of one-hot-encoded matrix onto *k*-dim eigen space only
- `label_encoder` or `LabelEncoder`: Convert every enum into the integer of its index (for example, level 0 -> 0, level 1 -> 1, etc.). The categories are lexicographically mapped to numbers and lose their categorical nature, becoming ordinal. After the strings are mapped to integers, you can split {0, 1, 2, 3, 4, 5} as {0, 1, 2} and {3, 4, 5}. This is useful for keeping the number of columns small.
- `enum_limited` or `EnumLimited`: Automatically reduce categorical levels to the most prevalent ones during Aggregator training and only keep the **T** (10) most frequent levels.

XGBoost

- `auto` or `AUTO`: Allow the algorithm to decide (default). In XGBoost, the algorithm will automatically perform `one_hot_internal` encoding. (default)
- `one_hot_internal` or `OneHotInternal`: On the fly N+1 new cols for categorical features with N levels
- `one_hot_explicit` or `OneHotExplicit`: N+1 new columns for categorical features with N levels
- `binary`: No more than 32 columns per categorical feature
- `label_encoder` or `LabelEncoder`: Convert every enum into the integer of its index (for example, level 0 -> 0, level 1 -> 1, etc.) The categories are lexicographically mapped to numbers and lose their categorical nature, becoming ordinal. After the strings are mapped to integers, you can split {0, 1, 2, 3, 4, 5} as {0, 1, 2} and {3, 4, 5}.
- `sort_by_response` or `SortByResponse`: Reorders the levels by the mean response (for example, the level with lowest response -> 0, the level with second-lowest response -> 1, etc.). This is useful, for example, when you have more levels than `nbins_cats`, and where the top level splits now have a chance at separating the data with a split. Note that this requires a specified response column.
- `enum_limited` or `EnumLimited`: Automatically reduce categorical levels to the most prevalent ones during training and only keep the **T** (10) most frequent levels, and then internally do one hot encoding in the case of XGBoost.

K-Means

- `auto` or `AUTO`: Allow the algorithm to decide (default). For K-Means, the algorithm will perform Enum encoding when `auto` option is specified.
- `enum` or `Enum`: Leave the dataset as is, internally map the strings to integers, and use these integers to make splits - either via ordinal nature when `nbins_cats` is too small to resolve all levels or via bitsets that do a perfect group split. Each category is a separate category; its name (or number) is irrelevant. For example, after the strings are mapped to integers for Enum, you can split {0, 1, 2, 3, 4, 5} as {0, 4, 5} and {1, 2, 3}.
- `one_hot_explicit` or `OneHotExplicit`: N+1 new columns for categorical features with N levels
- `binary` or `Binary`: No more than 32 columns per categorical feature
- `eigen` or `Eigen`: *k* columns per categorical feature, keeping projections of one-hot-encoded matrix onto *k*-dim eigen space only
- `label_encoder` or `LabelEncoder`: Convert every enum into the integer of its index (for example, level 0 -> 0, level 1 -> 1, etc.) The categories are lexicographically mapped to numbers and lose their categorical nature, becoming ordinal. After the strings are mapped to integers, you can split {0, 1, 2, 3, 4, 5} as {0, 1, 2} and {3, 4, 5}.
- `sort_by_response` or `SortByResponse`: Reorders the levels by the mean response (for example, the level with lowest response -> 0, the level with second-lowest response -> 1, etc.). Note that this requires a specified response column.
- `enum_limited` or `EnumLimited`: Automatically reduce categorical levels to the most prevalent ones during training and only keep the **T** (10) most frequent levels.

Related Parameters

- None

Example

R | Python

```
import h2o
from h2o.estimators.gbm import H2OGradientBoostingEstimator
h2o.init()
h2o.cluster().show_status()

# import the airlines dataset:
# This dataset is used to classify whether a flight will be delayed 'YES' or not "NO"
# original data can be found at http://www.transtats.bts.gov/
airlines= h2o.import_file("https://s3.amazonaws.com/h2o-public-test-
data/smalldata/airlines/allyears2k_headers.zip")

# convert columns to factors
airlines["Year"]= airlines["Year"].asfactor()
airlines["Month"]= airlines["Month"].asfactor()
airlines["DayOfWeek"] = airlines["DayOfWeek"].asfactor()
airlines["Cancelled"] = airlines["Cancelled"].asfactor()
airlines['FlightNum'] = airlines['FlightNum'].asfactor()

# set the predictor names and the response column name
predictors = ["Origin", "Dest", "Year", "UniqueCarrier", "DayOfWeek", "Month", "Distance",
"FlightNum"]
response = "IsDepDelayed"

# split into train and validation sets
train, valid= airlines.split_frame(ratios = [.8], seed = 1234)

# try using the `categorical_encoding` parameter:
encoding = "one_hot_explicit"

# initialize the estimator
airlines_gbm = H2OGradientBoostingEstimator(categorical_encoding = encoding, seed =1234)

# then train the model
airlines_gbm.train(x = predictors, y = response, training_frame = train, validation_frame =
valid)

# print the auc for the validation set
airlines_gbm.auc(valid=True)
```