Package ‘sparkxgb’

February 23, 2021

**Type** Package

**Title** Interface for 'XGBoost' on 'Apache Spark'

**Version** 0.1.1

**Maintainer** Yitao Li <yitao@rstudio.com>

**Description**
A 'sparklyr' <https://spark.rstudio.com/> extension that provides an R interface for 'XGBoost' <https://github.com/dmlc/xgboost> on 'Apache Spark'. 'XGBoost' is an optimized distributed gradient boosting library.

**License** Apache License (>= 2.0)

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 3.1.2)

**Imports** sparklyr (>= 1.3), forge (>= 0.1.9005)

**RoxygenNote** 7.1.1

**Suggests** dplyr, purrr, rlang, testthat

**NeedsCompilation** no

**Author** Kevin Kuo [aut] (<https://orcid.org/0000-0001-7803-7901>), Yitao Li [aut, cre] (<https://orcid.org/0000-0002-1261-905X>)

**Repository** CRAN

**Date/Publication** 2021-02-23 10:20:02 UTC

**R topics documented:**

xgboost_classifier .......................................................... 2
xgboost_regressor ............................................................ 6

**Index** 11
xgboost_classifier  XGBoost Classifier

Description

XGBoost classifier for Spark.

Usage

xgboost_classifier(
  x,
  formula = NULL,
  eta = 0.3,
  gamma = 0,
  max_depth = 6,
  min_child_weight = 1,
  max_delta_step = 0,
  grow_policy = "depthwise",
  max_bins = 16,
  subsample = 1,
  colsample_bytree = 1,
  colsample_bylevel = 1,
  lambda = 1,
  alpha = 0,
  tree_method = "auto",
  sketch_eps = 0.03,
  scale_pos_weight = 1,
  sample_type = "uniform",
  normalize_type = "tree",
  rate_drop = 0,
  skip_drop = 0,
  lambda_bias = 0,
  tree_limit = 0,
  num_round = 1,
  num_workers = 1,
  nthread = 1,
  use_external_memory = FALSE,
  silent = 0,
  custom_obj = NULL,
  custom_eval = NULL,
  missing = NaN,
  seed = 0,
  timeout_request_workers = 30 * 60 * 1000,
  checkpoint_path = "",
  checkpoint_interval = -1,
  objective = "multi:softprob",
  base_score = 0.5,
train_test_ratio = 1,
num_early_stopping_rounds = 0,
objective_type = "classification",
eval_metric = NULL,
maximize_evaluation_metrics = FALSE,
um_class = NULL,
base_margin_col = NULL,
thresholds = NULL,
weight_col = NULL,
features_col = "features",
label_col = "label",
prediction_col = "prediction",
probability_col = "probability",
raw_prediction_col = "rawPrediction",
uid = random_string("xgboost_classifier_"),
...
)

Arguments

x A spark_connection, ml_pipeline, or a tbl_spark.

formula Used when x is a tbl_spark. R formula as a character string or a formula. This is used to transform the input dataframe before fitting, see ft_r_formula for details.

eta Step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features and eta actually shrinks the feature weights to make the boosting process more conservative. [default=0.3] range: [0,1]

gamma Minimum loss reduction required to make a further partition on a leaf node of the tree. the larger, the more conservative the algorithm will be. [default=0]

max_depth Maximum depth of a tree, increase this value will make model more complex / likely to be overfitting. [default=6]

min_child_weight Minimum sum of instance weight(hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning. In linear regression mode, this simply corresponds to minimum number of instances needed to be in each node. The larger, the more conservative the algorithm will be. [default=1]

max_delta_step Maximum delta step we allow each tree’s weight estimation to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help making the update step more conservative. Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced. Set it to value of 1-10 might help control the update. [default=0]

grow_policy Growth policy for fast histogram algorithm.

max_bins Maximum number of bins in histogram.
subsample
Subsample ratio of the training instance. Setting it to 0.5 means that XGBoost randomly collected half of the data instances to grow trees and this will prevent overfitting. [default=1] range:(0,1]

colsample_bytree
Subsample ratio of columns when constructing each tree. [default=1] range: (0,1]
colsample_bylevel
Subsample ratio of columns for each split, in each level. [default=1] range: (0,1]
lambda
L2 regularization term on weights, increase this value will make model more conservative. [default=1]
alpha
L1 regularization term on weights, increase this value will make model more conservative, defaults to 0.
tree_method
The tree construction algorithm used in XGBoost. options: 'auto', 'exact', 'approx' [default='auto']
sketch_eps
This is only used for approximate greedy algorithm. This roughly translated into O(1 / sketch_eps) number of bins. Compared to directly select number of bins, this comes with theoretical guarantee with sketch accuracy. [default=0.03] range: (0, 1)
scale_pos_weight
Control the balance of positive and negative weights, useful for unbalanced classes. A typical value to consider: sum(negative cases) / sum(positive cases). [default=1]
sample_type
Parameter for Dart booster. Type of sampling algorithm. "uniform": dropped trees are selected uniformly. "weighted": dropped trees are selected in proportion to weight. [default="uniform"]
normalize_type
Parameter of Dart booster. type of normalization algorithm, options: 'tree', 'forest'. [default="tree"]
rate_drop
Parameter of Dart booster. dropout rate. [default=0.0] range: [0.0, 1.0]
skip_drop
Parameter of Dart booster. probability of skip dropout. If a dropout is skipped, new trees are added in the same manner as gbtree. [default=0.0] range: [0.0, 1.0]
lambda_bias
Parameter of linear booster L2 regularization term on bias, default 0 (no L1 reg on bias because it is not important.)
tree_limit
Limit number of trees in the prediction; defaults to 0 (use all trees.)
num_round
The number of rounds for boosting.
num_workers
number of workers used to train xgboost model. Defaults to 1.
nthread
Number of threads used by per worker. Defaults to 1.
use_external_memory
The tree construction algorithm used in XGBoost. options: 'auto', 'exact', 'approx' [default='auto']
silent
0 means printing running messages, 1 means silent mode. default: 0
custom_obj
Customized objective function provided by user. Currently unsupported.
custom_eval
Customized evaluation function provided by user. Currently unsupported.
missing The value treated as missing. default: Float.NaN
seed Random seed for the C++ part of XGBoost and train/test splitting.
timeout_request_workers the maximum time to wait for the job requesting new workers. default: 30 minutes
checkpoint_path The hdfs folder to load and save checkpoint boosters.
checkpoint_interval Param for set checkpoint interval (>= 1) or disable checkpoint (-1). E.g. 10 means that the trained model will get checkpointed every 10 iterations. Note: checkpoint_path must also be set if the checkpoint interval is greater than 0.
base_score Param for initial prediction (aka base margin) column name. Defaults to 0.5.
train_test_ratio Fraction of training points to use for testing.
num_early_stopping_rounds If non-zero, the training will be stopped after a specified number of consecutive increases in any evaluation metric.
objective_type The learning objective type of the specified custom objective and eval. Corresponding type will be assigned if custom objective is defined options: regression, classification.
eval_metric Evaluation metrics for validation data, a default metric will be assigned according to objective(rmse for regression, and error for classification, mean average precision for ranking). options: rmse, mae, logloss, error, merror, mlogloss, auc, aucpr, ndcg, map, gamma-deviance
maximize_evaluation_metrics Whether to maximize evaluation metrics. Defaults to FALSE (for minization.)
num_class Number of classes.
base_margin_col Param for initial prediction (aka base margin) column name.
thresholds Thresholds in multi-class classification to adjust the probability of predicting each class. Array must have length equal to the number of classes, with values > 0 excepting that at most one value may be 0. The class with largest value p/t is predicted, where p is the original probability of that class and t is the class’s threshold.
weight_col Weight column.
features_col Features column name, as a length-one character vector. The column should be single vector column of numeric values. Usually this column is output by ft_r_formula.
label_col Label column name. The column should be a numeric column. Usually this column is output by ft_r_formula.
prediction_col Prediction column name.
probability_col
Column name for predicted class conditional probabilities.

raw_prediction_col
Raw prediction (a.k.a. confidence) column name.

uid
A character string used to uniquely identify the ML estimator.

... Optional arguments; see Details.

---

**xgboost_regressor**

**XGBoost Regressor**

**Description**

XGBoost regressor for Spark.

**Usage**

```r
xgboost_regressor(
  x,
  formula = NULL,
  eta = 0.3,
  gamma = 0,
  max_depth = 6,
  min_child_weight = 1,
  max_delta_step = 0,
  grow_policy = "depthwise",
  max_bins = 16,
  subsample = 1,
  colsample_bytree = 1,
  colsample_bylevel = 1,
  lambda = 1,
  alpha = 0,
  tree_method = "auto",
  sketch_eps = 0.03,
  scale_pos_weight = 1,
  sample_type = "uniform",
  normalize_type = "tree",
  rate_drop = 0,
  skip_drop = 0,
  lambda_bias = 0,
  tree_limit = 0,
  num_round = 1,
  num_workers = 1,
  nthread = 1,
  use_external_memory = FALSE,
  silent = 0,
  custom_obj = NULL,
)```
custom_eval = NULL,
missing = NaN,
seed = 0,
timeout_request_workers = 30 * 60 * 1000,
checkpoint_path = "",
checkpoint_interval = -1,
objective = "reg:linear",
base_score = 0.5,
train_test_ratio = 1,
num_early_stopping_rounds = 0,
objective_type = "regression",
eval_metric = NULL,
maximize_evaluation_metrics = FALSE,
base_margin_col = NULL,
weight_col = NULL,
features_col = "features",
label_col = "label",
prediction_col = "prediction",
uid = random_string("xgboost_regressor_"),
...}

Arguments

x           A spark_connection, ml_pipeline, or a tbl_spark.

formula     Used when x is a tbl_spark. R formula as a character string or a formula. This is used to transform the input dataframe before fitting, see ft_r_formula for details.

eta          Step size shrinkage used in update to prevent overfitting. After each boosting step, we can directly get the weights of new features and eta actually shrinks the feature weights to make the boosting process more conservative. [default=0.3] range: [0,1]

gamma        Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger, the more conservative the algorithm will be. [default=0]

max_depth    Maximum depth of a tree, increase this value will make the model more complex / likely to be overfitting. [default=6]

min_child_weight

Minimum sum of instance weight(hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning. In linear regression mode, this simply corresponds to minimum number of instances needed to be in each node. The larger, the more conservative the algorithm will be. [default=1]

max_delta_step

Maximum delta step we allow each tree’s weight estimation to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help making the update step more conservative. Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced. Set it to value of 1-10 might help control the update. [default=0]
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grow_policy</td>
<td>Growth policy for fast histogram algorithm.</td>
</tr>
<tr>
<td>max_bins</td>
<td>Maximum number of bins in histogram.</td>
</tr>
<tr>
<td>subsample</td>
<td>Subsample ratio of the training instance. Setting it to 0.5 means that XGBoost randomly collected half of the data instances to grow trees and this will prevent overfitting. [default=1] range:(0,1]</td>
</tr>
<tr>
<td>colsample_bytree</td>
<td>Subsample ratio of columns when constructing each tree. [default=1] range: (0,1]</td>
</tr>
<tr>
<td>colsample_bylevel</td>
<td>Subsample ratio of columns for each split, in each level. [default=1] range: (0,1]</td>
</tr>
<tr>
<td>lambda</td>
<td>L2 regularization term on weights, increase this value will make model more conservative. [default=1]</td>
</tr>
<tr>
<td>alpha</td>
<td>L1 regularization term on weights, increase this value will make model more conservative, defaults to 0.</td>
</tr>
<tr>
<td>tree_method</td>
<td>The tree construction algorithm used in XGBoost. options: 'auto', 'exact', 'approx' [default='auto']</td>
</tr>
<tr>
<td>sketch_eps</td>
<td>This is only used for approximate greedy algorithm. This roughly translated into O(1 / sketch_eps) number of bins. Compared to directly select number of bins, this comes with theoretical guarantee with sketch accuracy. [default=0.03] range: (0,1)</td>
</tr>
<tr>
<td>scale_pos_weight</td>
<td>Control the balance of positive and negative weights, useful for unbalanced classes. A typical value to consider: sum(negative cases) / sum(positive cases). [default=1]</td>
</tr>
<tr>
<td>sample_type</td>
<td>Parameter for Dart booster. Type of sampling algorithm. &quot;uniform&quot;: dropped trees are selected uniformly. &quot;weighted&quot;: dropped trees are selected in proportion to weight. [default=&quot;uniform&quot;]</td>
</tr>
<tr>
<td>normalize_type</td>
<td>Parameter of Dart booster. type of normalization algorithm, options: 'tree', 'forest'. [default=&quot;tree&quot;]</td>
</tr>
<tr>
<td>rate_drop</td>
<td>Parameter of Dart booster. dropout rate. [default=0.0] range: [0.0, 1.0]</td>
</tr>
<tr>
<td>skip_drop</td>
<td>Parameter of Dart booster. probability of skip dropout. If a dropout is skipped, new trees are added in the same manner as gbtree. [default=0.0] range: [0.0, 1.0]</td>
</tr>
<tr>
<td>lambda_bias</td>
<td>Parameter of linear booster L2 regularization term on bias, default 0 (no L1 reg on bias because it is not important.)</td>
</tr>
<tr>
<td>tree_limit</td>
<td>Limit number of trees in the prediction; defaults to 0 (use all trees.)</td>
</tr>
<tr>
<td>num_round</td>
<td>The number of rounds for boosting.</td>
</tr>
<tr>
<td>num_workers</td>
<td>number of workers used to train xgboost model. Defaults to 1.</td>
</tr>
<tr>
<td>nthread</td>
<td>Number of threads used by per worker. Defaults to 1.</td>
</tr>
<tr>
<td>use_external_memory</td>
<td>The tree construction algorithm used in XGBoost. options: 'auto', 'exact', 'approx' [default='auto']</td>
</tr>
<tr>
<td>silent</td>
<td>0 means printing running messages, 1 means silent mode. default: 0</td>
</tr>
</tbody>
</table>
custom_obj  Customized objective function provided by user. Currently unsupported.
custom_eval Customized evaluation function provided by user. Currently unsupported.
missing    The value treated as missing. default: Float.NaN
seed       Random seed for the C++ part of XGBoost and train/test splitting.
timeout_request_workers
the maximum time to wait for the job requesting new workers. default: 30 minutes
checkpoint_path
The hdfs folder to load and save checkpoint boosters.
checkpoint_interval
Param for set checkpoint interval (>= 1) or disable checkpoint (-1). E.g. 10
means that the trained model will get checkpointed every 10 iterations. Note:
checkpoint_path must also be set if the checkpoint interval is greater than 0.
objective Specify the learning task and the corresponding learning objective. options:
reg:linear, reg:logistic, binary:logistic, binary:logitraw, count:poisson, multi:softmax,
base_score Param for initial prediction (aka base margin) column name. Defaults to 0.5.
train_test_ratio Fraction of training points to use for testing.
num_early_stopping_rounds
If non-zero, the training will be stopped after a specified number of consecutive
increases in any evaluation metric.
objective_type The learning objective type of the specified custom objective and eval. Corresponding
type will be assigned if custom objective is defined options: regression, classification.
eval_metric Evaluation metrics for validation data, a default metric will be assigned according
to objective(rmse for regression, and error for classification, mean average
precision for ranking). options: rmse, mae, logloss, error, merror, mlogloss, auc,
aucpr, ndcg, map, gamma-deviance
maximize_evaluation_metrics Whether to maximize evaluation metrics. Defaults to FALSE (for minization.)
base_margin_col Param for initial prediction (aka base margin) column name.
weight_col  Weight column.
features_col Features column name, as a length-one character vector. The column should be
single vector column of numeric values. Usually this column is output by
ft_r_formula.
label_col    Label column name. The column should be a numeric column. Usually this
column is output by ft_r_formula.
prediction_col Prediction column name.
uid         A character string used to uniquely identify the ML estimator.
...         Optional arguments; see Details.
Details

When x is a tbl_spark and formula (alternatively, response and features) is specified, the function returns a ml_model object wrapping a ml_pipeline_model which contains data pre-processing transformers, the ML predictor, and, for classification models, a post-processing transformer that converts predictions into class labels. For classification, an optional argument predicted_label_col (defaults to "predicted_label") can be used to specify the name of the predicted label column. In addition to the fitted ml_pipeline_model, ml_model objects also contain a ml_pipeline object where the ML predictor stage is an estimator ready to be fit against data. This is utilized by ml_save with type = "pipeline" to facilitate model refresh workflows.

Value

The object returned depends on the class of x.

- spark_connection: When x is a spark_connection, the function returns an instance of a ml_estimator object. The object contains a pointer to a Spark Predictor object and can be used to compose Pipeline objects.
- ml_pipeline: When x is a ml_pipeline, the function returns a ml_pipeline with the predictor appended to the pipeline.
- tbl_spark: When x is a tbl_spark, a predictor is constructed then immediately fit with the input tbl_spark, returning a prediction model.
- tbl_spark, with formula: specified When formula is specified, the input tbl_spark is first transformed using a RFormula transformer before being fit by the predictor. The object returned in this case is a ml_model which is a wrapper of a ml_pipeline_model.

See Also

See http://spark.apache.org/docs/latest/ml-classification-regression.html for more information on the set of supervised learning algorithms.
Index

* ml algorithms
  xgboost_regressor, 6

ft_r_formula, 3, 5, 7, 9

ml_save, 10

xgboost_classifier, 2
xgboost_regressor, 6