Package: xgb2sql (via r-universe)

August 31, 2024

Type Package

Title Convert Trained 'XGBoost' Model to SQL Query

Version 0.1.2

Description This tool enables in-database scoring of 'XGBoost' models built in R, by translating trained model objects into SQL query. 'XGBoost' <https://github.com/dmlc/xgboost> provides parallel tree boosting (also known as gradient boosting machine, or GBM) algorithms in a highly efficient, flexible and portable way. GBM algorithm is introduced by Friedman (2001) <doi:10.1214/aos/1013203451>, and more details on 'XGBoost' can be found in Chen & Guestrin (2016) <doi:10.1145/2939672.2939785>.

Author Chengjun Hou [aut, cre], Abhishek Bishoyi [aut]

Maintainer Chengjun Hou <chengjun.hou@gmail.com>

URL https://github.com/chengjunhou/xgb2sql

BugReports https://github.com/chengjunhou/xgb2sql/issues

Depends R (>= 3.1.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports xgboost (>= 0.81.0.1), data.table (>= 1.12.0)

Suggests ggplot2, knitr, rmarkdown

RoxygenNote 6.1.1

VignetteBuilder knitr

Repository https://chengjunhou.r-universe.dev

RemoteUrl https://github.com/chengjunhou/xgb2sql

RemoteRef HEAD

RemoteSha b2c9e9dc96515acd0b18df9453c7e48dd503dabd

Contents

`booster2sql` *Transform XGBoost model object to SQL query.*

Description

This function generates SQL query for in-database scoring of XGBoost models, providing a robust and efficient way of model deployment. It takes in the trained XGBoost model `xgbModel`, name of the input database table `input_table_name`, and name of a unique identifier within that table `unique_id` as input, writes the SQL query to a file specified by `output_file_name`. Note that the input database table should be generated from the raw table using the one-hot encoding query output by `onehot2sql()`, or to provide the one-hot encoding query as input `input_onehot_query` to this function, working as sub-query inside the final model scoring query.

Usage

```
booster2sql(xgbModel, print_progress = FALSE, unique_id = NULL,
  output_file_name = NULL, input_table_name = NULL,
  input_onehot_query = NULL)
```

Arguments

`xgbModel` The trained model object of class `xgb.Booster`. Current supported booster is `booster="gbtree"`, supported `objective` options are:

- – `reg:linear`: linear regression.
- – `reg:logistic`: logistic regression.
- - `binary:logistic`: logistic regression for binary classification, output probability.
- – `binary:logitraw`: logistic regression for binary classification, output score before logistic transformation.
- - `binary:hinge`: hinge loss for binary classification. This makes predictions of 0 or 1, rather than producing probabilities.
- - `count:poisson`: poisson regression for count data, output mean of poisson distribution.
- - `reg:gamma`: gamma regression with log-link, output mean of gamma distribution. It might be useful, e.g., for modeling insurance claims severity, or for any outcome that might be gamma-distributed.
- - `reg:tweedie`: Tweedie regression with log-link. It might be useful, e.g., for modeling total loss in insurance, or for any outcome that might be Tweedie-distributed.

`print_progress` Boolean indicator controls whether the SQL generating progress should be printed to console.

`unique_id` A row unique identifier is crucial for in-database scoring of XGBoost model. If not given, SQL query will be generated with id name "ROW_KEY".

`output_file_name`
File name that the SQL syntax will write to. It must not be empty in order for this function to run.

`input_table_name`
Name of raw data table in the database, that the SQL query will select from. If not given, SQL query will be generated with table name "MODREADY_TABLE".

`input_onehot_query`
SQL query of one-hot encoding generated by `onehot2sql`. When `input_table_name` is empty while `input_onehot_query` is not, the final output query will include `input_onehot_query` as sub-query.

Value

The SQL query will write to the file specified by `output_file_name`.

Examples

```
library(xgboost)
# load data
df = data.frame(ggplot2::diamonds)
head(df)

# data processing
out <- onehot2sql(df)
x <- out$model.matrix[,colnames(out$model.matrix)!='price']
y <- out$model.matrix[,colnames(out$model.matrix)=='price']

# model training
bst <- xgboost(data = x,
               label = y,
               max.depth = 3,
               eta = .3,
               nround = 5,
               nthread = 1,
               objective = 'reg:linear')

# generate model scoring SQL script with ROW_KEY and MODREADY_TABLE
booster2sql(bst, output_file_name='xgb.txt')
```

`onehot2sql` *Prepare training data in R so that it is ready for XGBoost model fitting. Meta information is stored so the exact transformation can be applied to any new data. It also outputs SQL query performing the exact one-hot encoding for in-database data preparation.*

Description

This function performs full one-hot encoding for all the categorical features inside the training data, with all NAs inside both categorical and numeric features preserved. Other than outputting a matrix `model.matrix` which is the data after processing, it also outputs `meta` information keeping track of all the transformation the function performs, while SQL query for the transformation is kept in output `sql` and write to the file specified by `output_file_name`. If `meta` is specified as input to the function, the transformation and the corresponding SQL query will follow what is kept in `meta` exactly.

Usage

```
onehot2sql(data, meta = NULL, sep = "_", ws_replace = TRUE,
  ws_replace_with = "", unique_id = NULL, output_file_name = NULL,
  input_table_name = NULL)
```

Arguments

`data`	Data object of class `data.frame` or `data.table`.
`meta`	Optional, a list keeps track of all the transformation that has been taken on the categorical features.
`sep`	Separation symbol between the categorical features and their levels, which will be the column names inside the output `model.matrix`, default to "_".
`ws_replace`	Boolean indicator controls whether white-space and punctuation inside categorical feature levels should be replaced, default to TRUE.
`ws_replace_with`	Replacing symbol, default to " which means all white-space and punctuation should be removed.
`unique_id`	A row unique identifier is crucial for in-database scoring of XGBoost model. If not given, SQL query will be generated with id name "ROW_KEY".
`output_file_name`	Optional, a file name that the SQL query will write to.
`input_table_name`	Name of raw data table in the database, that the SQL query will select from. If not given, SQL query will be generated with table name "INPUT_TABLE".

Value

A list of 1). `meta` data tracking the transformation; 2). matrix `model.matrix` is the data after processing which is ready for XGBoost fitting; 3). SQL query `sql` performing the exact one-hot encoding in the database.

Examples

```
library(data.table)
### load test data
df = data.frame(ggplot2::diamonds)
head(df)
```

```
d1 = data.frame(ggplot2::diamonds)
d1[1,2] = NA  # NA on 1st row cut
d1[2,5] = NA  # NA on 2nd row depth
head(d1)

d2 = data.table(ggplot2::diamonds)
d2[, cut:=factor(cut, ordered=FALSE)]
d2[, clarity:=as.character(clarity)]
d2[, tsdt:=as.IDate('2017-01-05')]
d2[1:3, tsdt:=tsdt-1]
head(d2)

### out is obtained for training data
out <- onehot2sql(df)
out1 <- onehot2sql(d1)  # NA is kept in the output
out2 <- onehot2sql(d2)  # all non-numeric features will be treated as categorical

### perform same transformation for new data when meta is given
# test-1: new data has column class change
newdata = df[1:5,]
newdata$cut = as.character(newdata$cut)
onehot2sql(newdata, meta=out$meta)$model.matrix

# test-2: new data has NA
newdata = df[1:5,]
newdata[1,1]=NA; newdata[2,1]=NA; newdata[3,2]=NA; newdata[3,3]=NA; newdata[5,4]=NA
onehot2sql(newdata, meta=out$meta)$model.matrix

# test-3: newdata has column with new elements
newdata = d2[1:5,]
newdata[5,clarity:='NEW']; newdata[1,tsdt:=as.IDate('2017-05-01')]
onehot2sql(newdata, meta=out2$meta)$model.matrix

# test-4: newdata has new columns
newdata = d2[1:5,]
newdata[,new_col:=1]
onehot2sql(newdata, meta=out2$meta)$model.matrix

# test-5: newdata is lacking some columns
newdata = d2[1:5,]
newdata[,cut:=NULL]
onehot2sql(newdata, meta=out2$meta)$model.matrix
```

Index