ORACLE

Oracle Machine Learning for R

A component of Oracle Machine Learning that enables R for enterprise data.

August 2022, Version 1.0

Purpose statement

This document provides an overview of Oracle Machine Learning for R for Oracle Database and Oracle Autonomous Database. It is intended solely to help you assess the business benefits of Oracle Machine Learning for R and to plan your data science and I.T. projects.

Intended audience

This document is for anyone who is interested in using R at scale for data preparation, machine learning, and solution deployment involving data stored or accessible through Oracle Database or Oracle Autonomous Database.

Table of contents

Introduction

Oracle Machine Learning for R (OML4R), a component of Oracle Machine Learning on Oracle Database and Oracle Autonomous Database, makes the open source R scripting language and environment ready for enterprise data. Oracle Machine Learning for R provides a database-centric environment for end-to-end analytical processes using R. Users invoke overloaded R functions on database-resident data for data exploration and preparation, which leverages the database as a high-performance computing environment. Further, R users can invoke powerful in-database machine learning algorithms from R and deploy user-defined R functions supporting applications.

R for the Enterprise

To leverage the database as a high-performance computing environment through R, users manipulate database tables and views through R *data.frame* proxy objects with overloaded functionality that is translated to SQL – transparently, behind the scenes. This enables exploring, preparing, and analyzing data faster and at scale by keeping data in the database, while allowing the use of familiar R syntax to manipulate database data.

OML4R also provides a natural R interface for building in-database machine learning models and using those models to score data. Machine learning techniques include classification, regression, clustering, attribute importance, anomaly detection, association rules, time series, and feature extraction. The in-database algorithms support automated algorithm-specific data preparation, partitioned model ensembles, and integrated text mining. With Oracle Machine Learning for R, users can build more models on more data, and score large volume data – faster. Data scientists benefit from increased productivity, while at the same time, the powerful in-database algorithms are made more accessible to non-expert users.

On Oracle Autonomous Database, users can work with R through Oracle Machine Learning Notebooks, side-by-side with SQL and Python, choosing the most appropriate language for a given task. Using the OML4R client package, users can connect from an R engine to on-premises Oracle Database and Oracle Database Cloud Service databases. OML4R access to Oracle Autonomous Database from third-party clients will be introduced in an upcoming release.

Using embedded R execution, R users can develop user-defined R functions and manage these in the database script repository for production deployment. These user-defined functions can then be invoked via REST endpoints and a SQL interface on Oracle Autonomous Database and via a SQL interface on Oracle Database. Users can also run their user-defined R functions in a data-parallel and task-parallel manner, for example, to enable scoring native R models at scale by leveraging multiple database-spawned and controlled R engines. Results from these user-defined functions can contain both structured and image results.

R objects can also be stored in the database – as opposed to being managed in flat files – using the *datastore* capability. Datastores work together with embedded R execution for passing one or more R objects, including complex objects such as native R machine learning models, to user-defined functions.

The script repository, datastore, and embedded execution features facilitate collaboration across the data science team – enabling convenient hand-off of data science work products from data scientists to application developers for immediate deployment.

Transparency layer

The OML4R transparency layer provides the foundation for leveraging in-database performance. Using proxy objects and overloaded R functions that transparently produce SQL, users can take advantage of database table column indexes, query optimization, parallelism, in-memory caching, and table-level partitioning – which can greatly improve performance on database table operations.

Here, we see how to create a table from an R *data.frame* (`df` in the example) using the function *ore.create* and specifying the table name we wish to use, in this case *'BOSTON'*.

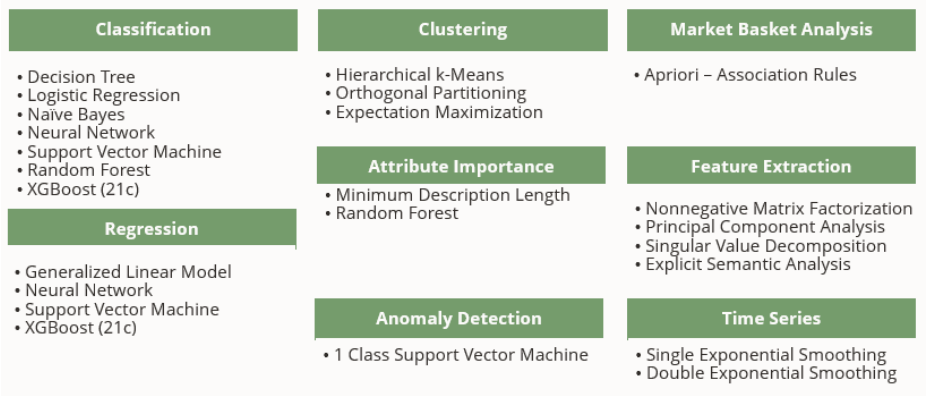
```
ore.create(df, table = 'BOSTON')
ore.sync(table = 'BOSTON')

dim(BOSTON)
head(BOSTON)
summary(BOSTON)
min(BOSTON$age)
split(BOSTON, BOSTON$chas)
```

If the table already exists in the database, we can use the function *ore.sync*, and specify the table or view name for which we want to get a proxy object. This proxy object (named `BOSTON` in the example) can then be used to invoke familiar R functions, like *dim*, *head*, *summary*, and *split*, among others, including functions from the *dplyr* package. These overloaded functions translate functionality to SQL for in-database processing.

In-database ML algorithms

OML4R provides a natural R API to the in-database algorithms using the R *Formula* specification. OML4R supports in-database algorithms for classification, regression, clustering, attribute importance, anomaly detection, association rules, time series, and feature extraction as shown below. The in-database algorithms support automatic algorithm-specific data preparation, partitioned model ensembles, and integrated text mining.

In-database models produced through the R API can be accessed using the SQL API, exported as a flat file for separate management or import to other Oracle Database instances, and deployed to OML Services.



Embedded R execution

Embedded R execution enables user-defined R functions to be run on R engines spawned and managed by the database environment. This capability can automatically load database data into the user-defined R function. Function results can contain both image and structured content, with structured content returned as tables accessible via *data.frame* proxy objects. User-defined R functions can be stored and managed in the database script repository.

On Oracle Autonomous Database, you can leverage third-party packages that are provided with OML4R in your user-defined functions, for example, to build models, score data, or generate lattice visualizations. On Oracle Database, users with appropriate database machine access permissions can custom install third-party packages.

The embedded R execution infrastructure also enables running your functions in a data-parallel and task-parallel manner – supporting "embarrassingly parallel" use cases such as scoring data using a native R machine learning model.

Embedded R execution facilitates production deployment of user-defined R functions, where you can invoke functions using a SQL interface – and on Autonomous Database, a REST interface – for application integration.

- **doEval** is the simplest and runs your function in a single R engine and returns function result, including data.frames and images as produced from the R graphics engine.
- **tableApply** also runs your function in a single R engine but passes in the data referenced by your proxy object as an R *data.frame* in the first argument of your function.
- **rowRpply** enables data parallelism by partitioning data into chunks of rows and runs your user-defined R function on each chunk, using one or more R engines spawned and controlled by the database environment.
- **groupApply** also enables data parallelism by partitioning database data by the column or columns specified by the index parameter and runs the user-defined R function on each partition using one or more R engines as specified in the parallel argument.
- **indexApply** enables task parallelism, by running your user-defined R function the specified number of times, passing the index value as the first argument to your function, using one or more R engines as specified in the parallel argument.

The following example shows using the R API to invoke a user-defined R function on Oracle Database that builds a Random Forest classification model using the *iris* data set to predict Species and stores that model in the datastore called *"RF-model-iris"*. On Oracle Database, the *randomForest* package would be installed on each database server machine node's R engine.

```
buildRFmodel <- function(dat,dsname) {
  library(randomForest)
  dat$Species <- as.factor(dat$Species)
  mod <- randomForest(Species ~ ., data=dat)
  ore.save(mod,name=dsname,overwrite = TRUE)
  TRUE
}
ore.connect(…database credentials…)
ore.scriptCreate('buildRFmodel', buildRFmodel)

ore.sync(table='IRIS')  # get ore.frame proxy object

ore.tableApply(IRIS, FUN.NAME='buildRFmodel',
               dsname= 'RF-model-iris',
               ore.connect=TRUE)
```

After defining our function, we connect to the database using *ore.connect* and store our use-defined R function in the database script repository.

Note that on Autonomous Database, the database connection is automatic, there is no explicit connection.

Next, we get a proxy object to the *IRIS* table that exists in the database, which is accessible through the R variable *IRIS*. Then, we use the *tableApply* function, which takes a proxy object, the name of our function as stored in the script repository, the name to use for the datastore instance where we store the model, and that we want to automatically establish a connection to the database within our function since we use the datastore.

The following example shows invoking the user-defined function *buildRFmodel* using the REST API and the corresponding SQL API on Oracle Autonomous Database.

```
$ curl -i -X POST --header "Authorization: Bearer ${token}"
     --header 'Content-Type: application/json'
     --header 'Accept: application/json' \
     -d'{"input":"IRIS", "parameters":"{\"dsname\":\"RF-model-
iris\"}"}' \
"${omlserver}/oml/api/r-scripts/v1/table-apply/buildRFmodel"


SELECT * FROM table(rqTableEval(
    inp_nam  => 'IRIS',
    par_lst  => '{"dsname":"RF-model-iris"}',
    out_fmt =>  'XML',
    scr_name => 'buildRFmodel'));
```

Users can also store user-defined R functions in the script repository using the SQL interface. For example, the following example stores the corresponding scoring function using the *sys.rqScriptCreate* function.

```
BEGIN
 sys.rqScriptCreate('scoreDataRF',
   'function(dat, dsname) {
      library(randomForest)
      ore.load(name=dsname)
      dat$Prediction <- predict(mod, newdata = dat)
      dat[,c("Species","Prediction")]
    }',FALSE, TRUE); -- not global and enable overwrite
END;
```

Since scoring is an embarrassingly parallel use case, we can invoke our scoring function using *ore.rowApply* and indicate the number of rows to process (10) at a time and the number of R engines to use (3). We also specify the *FUN.VALUE* with the structure of the result to return a single *data.frame* proxy object with the result.

```
PRED = ore.rowApply(IRIS,
         FUN.NAME = 'scoreDataRF',
         rows = 10,
         parallel = 3,
         dsname = 'RF-model-iris',
         ore.connect = TRUE,
         FUN.VALUE = data.frame(Species=character(),
                                Prediction=character(),
                                stringsAsFactors=FALSE))
class(PRED)   # returns an ore.frame proxy object
ore.create(PRED,table = 'BATCH_SCORES') # persist as table
with(BATCH_SCORES, table(Species, Prediction))
```

This can also be invoked using the SQL and REST APIs.

Database processing

Oracle Machine Learning for R supports data scientists and R users in running R scripts on database-resident data, thereby avoiding data movement to the client while supporting the use of familiar R functions. OML4R overloads select R functions, transparently translating these functions to SQL, which eliminates data movement while maintaining security and taking advantage of database table column indexes, query optimization, parallelism, and even partitioning. Oracle Machine Learning for R supports a wide range of in-database machine learning techniques and algorithms that have been redesigned for parallelism and scalability, taking advantage of multi-node clusters and optimized hardware such as Oracle Exadata.

While data scientists and R users interactively analyze data and develop R scripts, data and results can remain inside the database – reducing or eliminating data movement latency while enable more scalable data processing. For R users familiar with loading data from file extracts, OML4R facilitates a migration toward a database-centric architecture. Further, R users can easily push data from their desktops to the database.

OML4R is supported on Oracle Autonomous Database versions 19c and 21c, and Oracle Database versions 18c and later on select operating systems. OML4R is included with your Oracle Autonomous Database subscription, Oracle Database license, and Oracle Database Cloud Service.

Conclusion

If you have data managed by Oracle Autonomous Database or Oracle Database, empower your data scientists and other R users with the ability to explore, prepare, and analyze data at scale leveraging database performance and scalability. Additionally, give them the ability to build machine learning models at scale using a native R API to in-database algorithms. Users can also deploy their user-defined R functions leveraging database-spawned and controlled R engine.

Try Oracle Machine Learning for R on Oracle Autonomous Database for free using the Always Free cloud services on the Oracle Cloud Free Tier. Create your account today at https://oracle.com/cloud/free.

For further reading

OML4R Documentation

OML4R User's Guide:
https://docs.oracle.com/en/database/oracle/machine-learning/oml4r

Oracle Machine Learning

A family of products supporting data science projects that consists of APIs for SQL, Python, and R to deliver machine learning, statistical analysis, data preparation, and visualization, Oracle Data Miner as a drag-and-drop user interface for creating analytical workflows, OML Services for REST-based model management and deployment, OML AutoML UI for no-code machine learning modeling, and OML4Spark supporting machine learning on big data from R.

https://oracle.com/data-science/machine-learning

ROracle

ROracle is an R package for Oracle Database connectivity based on DBI. Oracle Machine Learning for R uses ROracle for database connectivity and within embedded R execution functions. ROracle can also be used for insert, update, and delete operations on database tables.

https://cran.r-project.org/web/packages/ROracle/index.html

https://www.oracle.com/database/technologies/roracle-downloads.html