**R BES for Titanic**

Input data file

Copy below data, save to input1data.csv, and upload to Azure storage /blob1/

PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked

1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S

2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C

3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S

(No new line at the end of file)

Output data file

The output data file is myresults.csv will be written to local disk

Source code

# How this works:

#

# 1. ASSUMPTION: This code assumes that your input is already uploaded to your Azure storage account (if the web service accepts input).

# 2. Call BES to process the data in the blob.

# 3. The results get written to another Azure blob

# 4. Download the output blob to a local file

library("RCurl")

library("rjson")

# Accept SSL certificates issued by public Certificate Authorities

options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))

requestFailed = function(headers) {

return (headers["status"] >= 400)

}

printHttpError = function(headers, result) {

print(paste("The request failed with status code:", headers["status"], sep=" "))

# Print the headers - they include the requert ID and the timestamp, which are useful for debugging the failure

print(headers)

print(fromJSON(result))}

saveBlobToFile = function(blobUrl, resultsLabel) {

output\_file = "d://temp/myresults.csv" # Replace this with the location you would like to use for your output file

print(paste("Reading the result from", blobUrl, sep=" "))

blobContent = getURL(blobUrl)

fc = file(output\_file)

write(blobContent, fc)

close(fc)

print(paste(resultsLabel, " have been written to the file", output\_file, sep=" "))

}

processResults = function(result) {

first = TRUE

for (outputName in names(result$Results))

{

result\_blob\_location = result$Results[[outputName]]

sas\_token = result\_blob\_location$SasBlobToken

base\_url = result\_blob\_location$BaseLocation

relative\_url = result\_blob\_location$RelativeLocation

print(paste("The result for", outputName, "is available at the following Azure Storage location:", sep=" "))

print(paste("BaseLocation: ", base\_url, sep=""))

print(paste("RelativeLocation: ", relative\_url, sep=""))

print(paste("SasBlobToken: ", sas\_token, sep=""))

if (first) {

first = FALSE

url3 = paste(base\_url, relative\_url, sas\_token, sep="")

saveBlobToFile(url3, paste("The results for", outputName, sep=" "))

}

}

}

invokeBatchExecutionService = function() {

storage\_account\_name = "loy2018sa" # Replace this with your Azure Storage Account name

storage\_account\_key = "4oKF2tzfkDk/H6eYzHa8YwpV/pNB9oVprOpc3PNIRrL/EduRP6/o2css1tX4p47ateS8AfT2DUetjgLv4Tr3hg==" # Replace this with your Azure Storage Key

storage\_container\_name = "blob1" # Replace this with your Azure Storage Container name

connection\_string = paste("DefaultEndpointsProtocol=https;AccountName=" , storage\_account\_name , ";AccountKey=" , storage\_account\_key, sep='')

api\_key = "IJh2PfzFAh5Q4Hsj/vod6PjgOlTBWeng2f2C+89Sv/1t1Vr7KaDZfequmXPzhAZNs9KjkaklAcSuRvTLy47/yw==" # Replace this with the API key for the web service

url = "https://ussouthcentral.services.azureml.net/workspaces/ede12cb3aaf24c7e826493f4e309f1e1/services/ad3b577804c443d08f0f30b6c8028411/jobs"

authz\_hdr = paste("Bearer", api\_key, sep=" ")

payload = list(

Inputs=list(

input1=list(ConnectionString=connection\_string, RelativeLocation=paste("/", storage\_container\_name, "/input1datablob.csv", sep=''))

),

Outputs=list(

output1=list(ConnectionString=connection\_string, RelativeLocation=paste("/", storage\_container\_name, "/output2results.csv", sep=''))

),

GlobalParameters=setNames(fromJSON('{}'), character(0))

)

body = enc2utf8(toJSON(payload))

print("Submitting the job...")

h = basicTextGatherer()

hdr = basicHeaderGatherer()

# submit the job

curlPerform(url = paste(url, "?api-version=2.0", sep=""),

httpheader = c("Content-Type" = "application/json", "Authorization" = authz\_hdr),

postfields = body,

writefunction = h$update,

headerfunction = hdr$update,

verbose = FALSE

)

headers = hdr$value()

result = h$value()

if (requestFailed(headers)) {

printHttpError(headers, result)

return()

}

job\_id = substring(result, 2,nchar(result)-1) # Removes the enclosing double-quotes

print(paste("Job ID:", job\_id, sep=" "))

# start the job

print("Starting the job...")

h$reset()

hdr$reset()

curlPerform(url = paste(url, "/", job\_id, "/start?api-version=2.0", sep=""),

httpheader = c("Authorization" = authz\_hdr),

postfields = "",

writefunction = h$update,

headerfunction = hdr$update,

verbose = FALSE

)

headers = hdr$value()

result = h$value()

if (requestFailed(headers)) {

printHttpError(headers, result)

return()

}

url2 = paste(url, "/", job\_id, "?api-version=2.0", sep="")

while (TRUE) {

print("Checking the job status...")

h$reset()

hdr$reset()

curlPerform(url = url2,

httpheader = c("Authorization" = authz\_hdr),

writefunction = h$update,

headerfunction = hdr$update,

verbose = FALSE

)

headers = hdr$value()

result = h$value()

if (requestFailed(headers)) {

printHttpError(headers, result)

return()

}

result = fromJSON(result)

status = result$StatusCode

if (status == 0 || status == "NotStarted") {

print(paste("Job", job\_id, "not yet started...", sep=" "))

}

else if (status == 1 || status == "Running") {

print(paste("Job", job\_id, "running...", sep=" "))

}

else if (status == 2 || status == "Failed") {

print(paste("Job", job\_id, "failed...", sep=" "))

print(paste("Error details:", result$Details, sep=" "))

break

}

else if (status == 3 || status == "Cancelled") {

print(paste("Job", job\_id, "cancelled...", sep=" "))

break

}

else if (status == 4 || status == "Finished") {

print(paste("Job", job\_id, "finished...", sep=" "))

processResults(result)

break

}

Sys.sleep(1) # Wait one second

}

}

invokeBatchExecutionService()