

Hands-On-Lab: Exploring Sales Data from Enterprise Systems through Machine Learning

1. Introduction

With developing Big Data, the functionality to support the work of data scientists is one of the new areas on the market. IBM Data Science Experience (DSX) is the premier offering that seamlessly allows data scientists and data engineers to connect data from multiple domains, to analyze it, and visually to explore the data for insights. The solutions that were not possible previously or required months of development effort can now be done in DSX in a very fast and intuitive manner.

R has become the de-facto standard in the domain of data science: it supports multiple machine learning algorithms and ability to visualize the outcomes of the research.

DSX brings together the data science development experience accumulated in R, Python, Scala, and Java, intuitive data connectivity and processing capabilities of Spark, and the state-of-art dynamic visualization technology using Brunel, Pixiedust, and RStudio.

In this lab we will explore how a data scientist utilizes DSX and IBM Bluemix cloud services to easily analyze data using machine learning techniques and to visualize the outcomes using DSX, R, and Brunel. For the lab, we have chosen two algorithms to demonstrate supervised and unsupervised machine learning in DSX. Decision tree-based classification is one of the domains that allowed scientists to have direct insights into the reasoning behind classification choices. Association rules algorithms support market basket analysis.

The lab shows how to use these machine learning algorithms for uncovering the data insights in sales data.

2. Lab Components: DSX

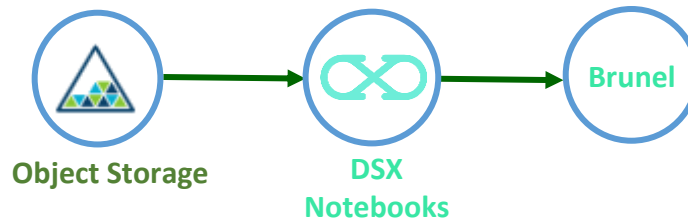
1. Source Data Repository: **Object Storage Service**

NOTE: This usually is the data hosted in a secure enterprise environment that could be securely retrieved through a network tunnel using the BlueMix Secure gateway. Since getting access to a data hosted in an enterprise infrastructure can be a

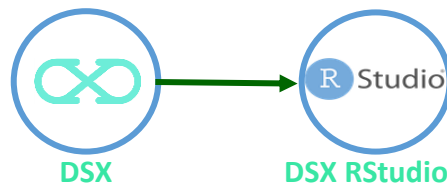
compliance and privacy issue for this lab, we will be using sample datasets hosted in the IBM Object Storage that would be used as a source data set.

2. Exploring sales data (for this lab, the data from IBM sample DB “GOSALES” was used):

a. Using Decision Tree: DSX Notebooks, Brunel, R



b. Using Association Rules: DSX RStudio



3. Before You Begin

1. Download Lab-DSX-ML.zip archive from the github.com location below and extract the data file (transactions.csv) to your laptop:

<https://github.com/ibmdataworks/datafirst/tree/master/datascientist/machinelearning/archive>

Here is the content of the downloaded archive:

- Folder “data” contains the sample data “transactions.csv”
- Folder “lab” contains the following files:
 - machine-learning-with-DSX-lab.ipynb – the notebook for the machine learning lab using decision trees;
 - ml-lab-installation.ipynb – the notebook for installing software packages for the machine learning lab (decision trees) ;
 - RStudio-apriori-demo.R – R code for the machine learning lab using association rules;
 - RStudio-apriori-demo-installation.R – R code for installing software packages for the machine learning lab (association rules)

2. Log in to your IBM DSX account at:

<http://datascience.ibm.com/>

NOTE: If you don't have a DSX account then get started by registering at
<https://datasciencex.typeform.com/to/hWECJ3>

3. A quick outline of the procedures:

- a. Decision tree lab (based on the DSX notebooks):

- i. Provision an Object Storage service and load a file with sales data (transactions.csv) for the analysis
- ii. Provision an Apache Spark service to run notebooks on
- iii. Load the sample notebooks with the lab (machine-learning-with-DSX-lab.ipynb) and the installation of the software packages (ml-lab-installation.ipynb) needed for the lab;
- iv. Run the code in the notebook with the installation sequence
- v. Run the code in the notebook with the lab

- b. Association rules lab (based on RStudio):

- i. Start RStudio
- ii. Load the data file into RStudio (transactions.csv into ~/data)
- iii. Load the R-code with the lab (RStudio-apriori-demo.R) and the installation R-code for the software packages (RStudio-apriori-demo-installation.R) needed for the lab
- iv. Run the R-code with the installation sequence
- v. Run the R-code with the lab

NOTE: If the same environment is sequentially used by different users, then the installation procedure needs to be done once per environment and the user only needs to go through executing the code with the lab without need to spend time on the installation procedures – there is no need to reinstall the labs for the same environment.




NOTE: In the instructions the sequence “Menu > Submenu > Final Item” would represent the sequence of users’ actions where they choose first “Menu” from the menu bar, then the “Submenu” item in the opened submenu list, and finally “Final Item” in the yet another opened menu list.

Start of Lab: Decision Tree

4. Decision Tree Lab Installation

4.1 Initializing Source Data Repository: Object Storage Service

Use the Object Storage Service that is created for your account. You can alternatively create a new Object Storage Service by following these instructions:

1. In DSX, go to user profile settings 
2. Go to manage Bluemix account 
3. From the Bluemix dashboard catalog menu search for “Object Storage”
4. Click on the Object Storage Icon 
5. Choose the default pre-filled values in the fields (optionally rename the “Service name:” to DS_DSX_ML_ObjectStorage), select the “Free Pricing Plan” and click “Create” at the bottom of the page.
6. Click on “Actions->Add Container” and enter “notebooks” as the container name.

NOTE: Once a new container is created, you get this message "There are no files in the container you selected. Add files or view Object Storage documentation."

7. Click on “Add files”.

8. Select and add the files previously downloaded (transactions.csv) to this Container.

NOTE: Your data files are now moved to the Object Storage into the Bluemix.


4.2 Initializing Apache Spark: Apache Spark Service

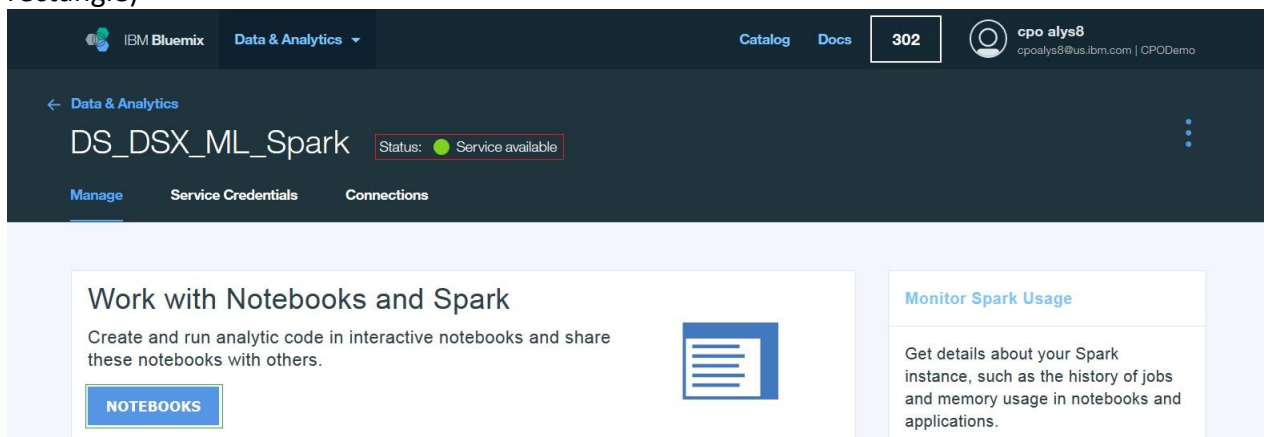
Use the Apache Spark Service that is created for your account. You can alternatively create a new Apache Spark Service by following these instructions (if you are already in the UI managing Bluemix account, please start from the step 3):

1. In DSX, go to user profile settings

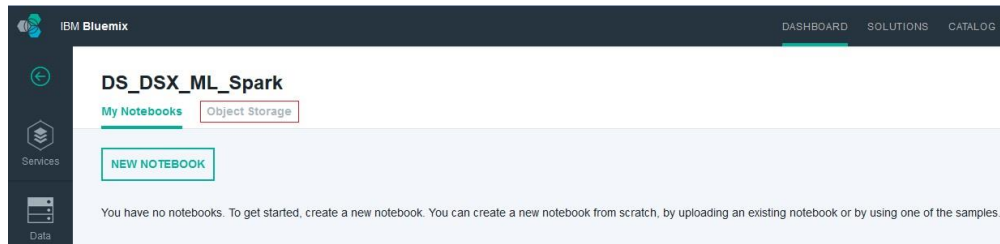


Manage Bluemix Account

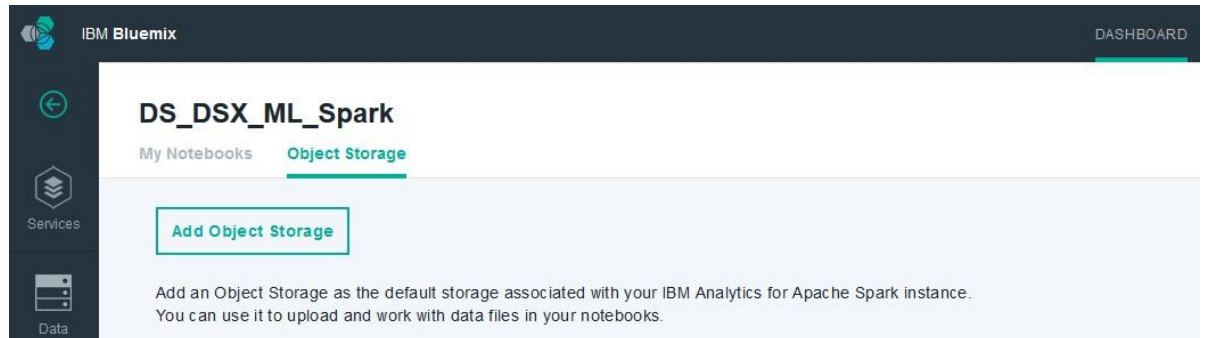
2. Go to manage Bluemix account
3. From the Bluemix dashboard catalog menu search for “Apache Spark”
4. Click on the Apache Spark Service Icon 
5. Choose the default pre-filled values in the fields (optionally rename the “Service name:” to DS_DSX_ML_Spark), select the needed “Pricing Plan” and click “Create” at the bottom of the page.
6. Ensure that your service is up and running – the page that you are transferred after creating the service shows the status of the service (this is highlighted with the red rectangle)



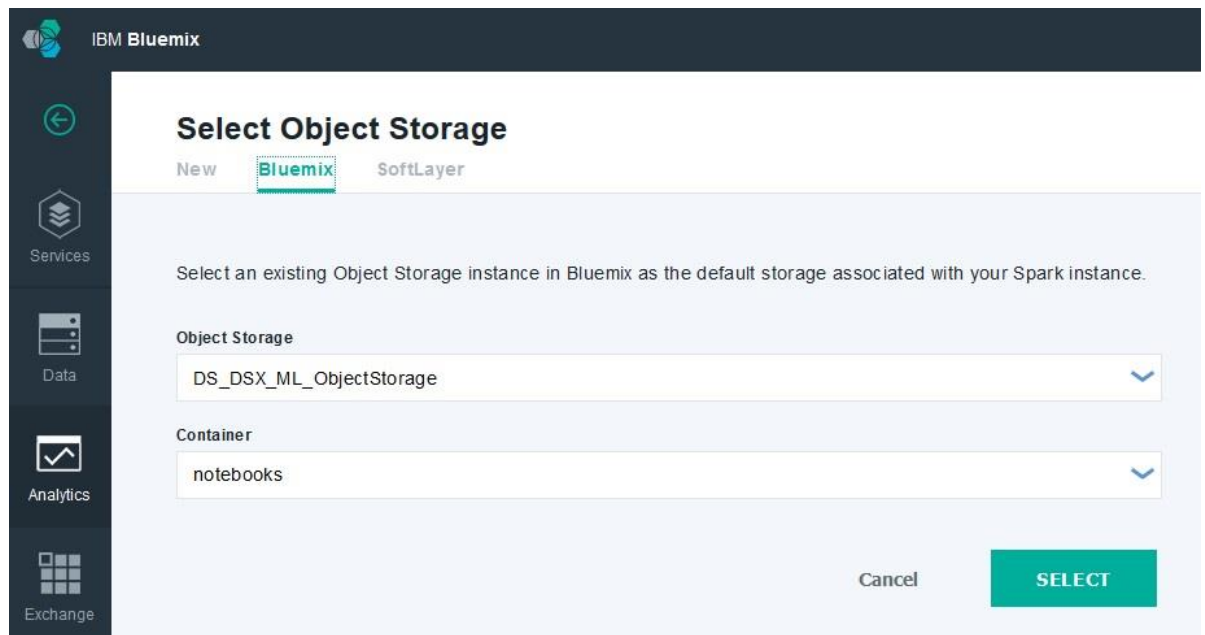
7. Connect your Apache Spark service to your Object Storage service:
 - a. Click on “NOTEBOOKS” button (it is highlighted in the green rectangle on the previous snapshot)
 - b. Click on Object Storage tab (it is highlighted with the red rectangle):



c. Click on “Add Object Storage”:



d. In the new screen, switch to Bluemix and select the Object Storage service and the container that you provisioned for this lab:



e. Click on “Select”: the provisioning of Apache Spark Service for DSX has been finished

8. Switch back to the tab with DSX in your browser

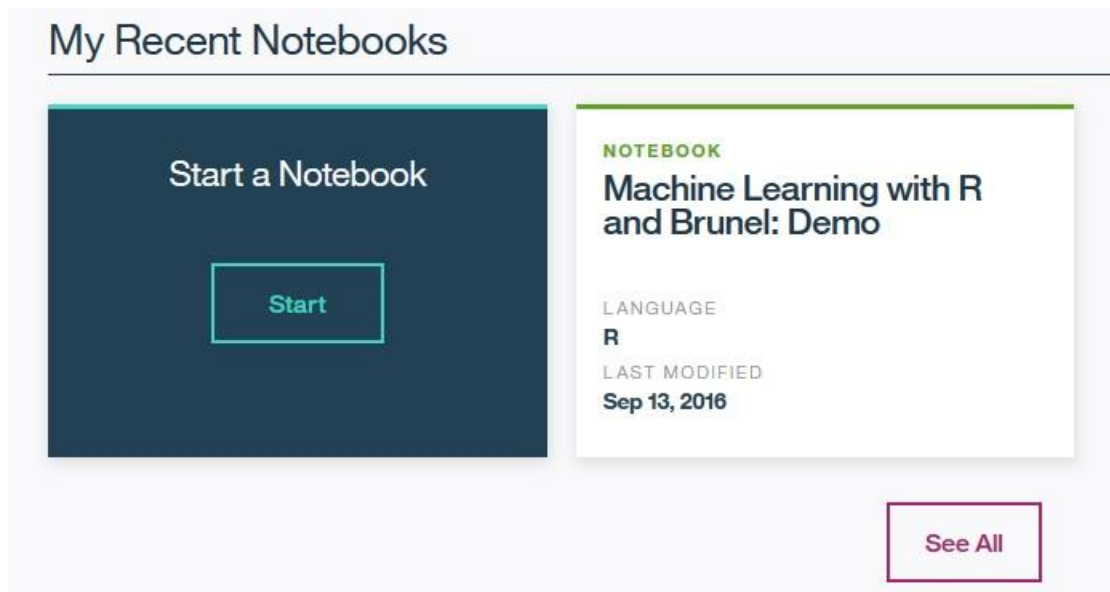
NOTE: a more detailed instructions or alternative approaches on setting up an Apache Spark service can be found at <https://console.ng.bluemix.net/docs/services/AnalyticsforApacheSpark/index.html>

4.3 Importing Notebooks for Machine Learning Lab

1. From DSX, select DSX data science mode (the button to click on is highlighted with the red rectangle):



2. From the DSX home page in Data Science mode, click on “Start” to create a new notebook:



3. In the next screen named “Create Notebook”, switch to “From File” tab, name the notebook “ML Lab Installation”, and choose the notebook file on your disk from the archive: ml-lab-installation.ipynb; alternatively you can switch to “From URL” tab and use the following “Notebook URL”:

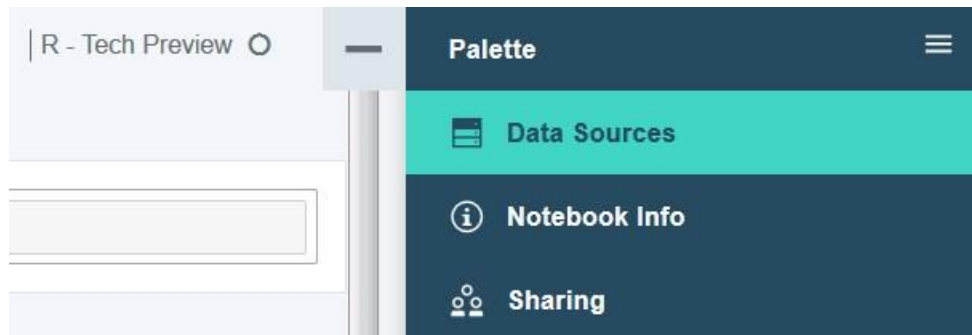
<https://github.com/ibmdataworks/datafirst/blob/master/datascientist/machinelearning/labs/ml-lab-installation.ipynb>

4. Choose the default pre-filled values in the fields (Project: None, Spark Service: the service that you provisioned for this lab)
5. Choose to Trust this Notebook to run with your Privileges and click on Create Notebook
6. When DSX loads the notebook, save the current state by clicking on File > Save and Checkpoint from the menu
7. Return back to DSX home page
8. Load the second notebook “Machine Learning with DSX - Lab” (from the file machine-learning-with-DSX-lab.ipynb, or from URL <https://github.com/ibmdataworks/datafirst/blob/master/datascientist/machinelearning/labs/machine-learning-with-DSX-lab.ipynb>) by following the same steps 1-7 as above

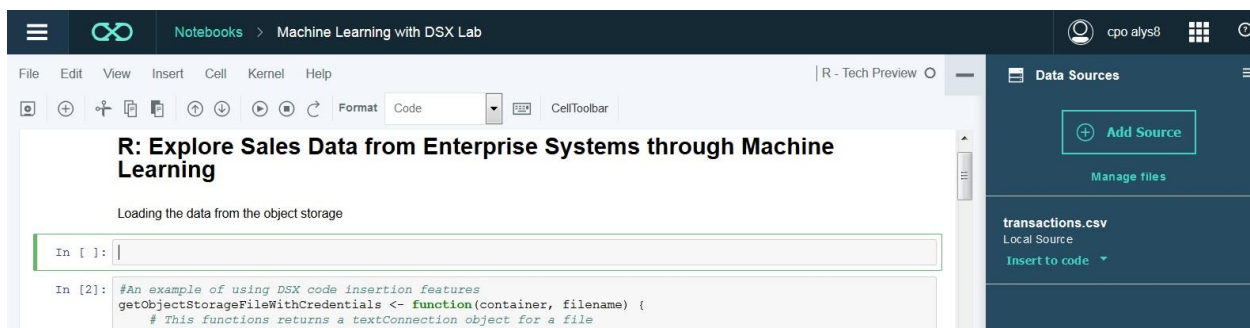
4.4 Switch to the Provisioned Source Data Repository in DSX Lab Notebook

This step allows to avoid overloading the default Object Storage service by switching to the provisioned Object Storage service.

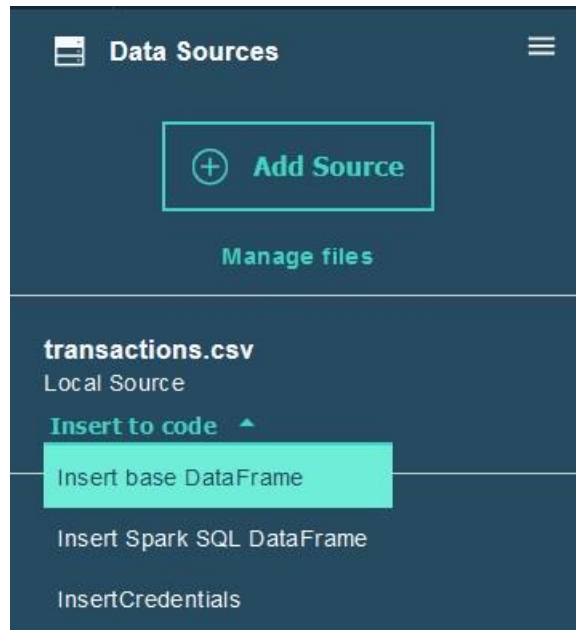
1. From the loaded notebook “Machine Learning with DSX Lab” click on Data Sources:



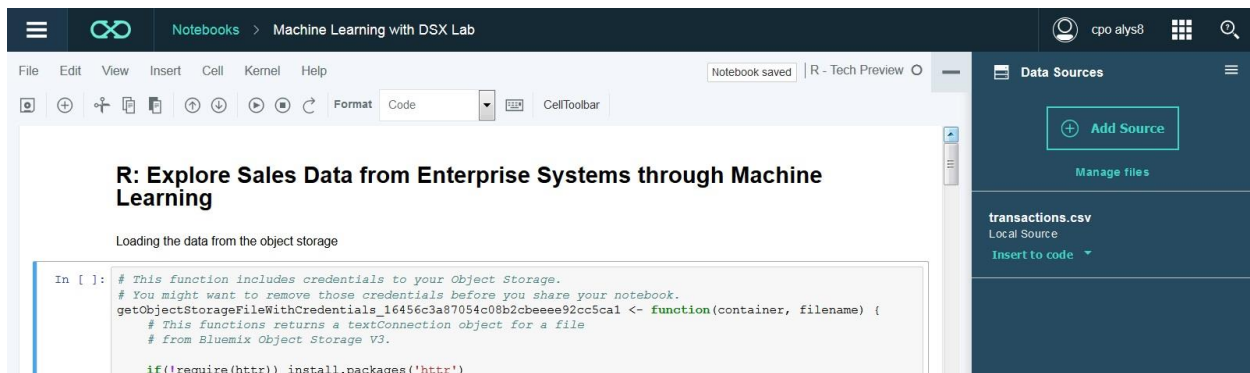
2. The expanded Data Sources would show `transact.csv` under “Manage files” section
3. Identify the cell with the implementation of `getObjectStorageFileWithCredentials` and replace the code to use your provisioned Object Storage service:
 - a. Place your cursor to the cell with the default `getObjectStorageFileWithCredentials` implementation
 - b. Create an empty code cell just above the code cell with the default `getObjectStorageFileWithCredentials` by clicking on the following menu items: “Insert” > “Insert Cell Above” and place your cursor into the new cell



- c. Clicking on “Insert to code” on Transactions.csv will show the options to insert the code: choose “insert base DataFrame” :



d. Here is a similar code that will be inserted into your new cell:



e. Replace the existing implementation of `getObjectStorageFileWithCredentials` (starts with “<- function” and finishes with the end of block “}”) with the generated code in the new cell for `getObjectStorageFileWithCredentials_<unique sequence>`; Here is the example of the highlighted code that needs to be replaced:

```

    })
  })
  data <- content(httr::GET(url = access_url, add_headers ("Content-Type" = "application/json", "X-Auth-Token" = x_subject_token)), as="text")
  textConnection(data)
}

df.data.1 <- read.csv(file = getObjectStorageFileWithCredentials_16456c3a87054c08b2cbeee92cc5ca1("notebooks", "transactions.csv"))
head(df.data.1)

In [2]: #An example of using DSX code insertion features
getObjectStorageFileWithCredentials <- function(container, filename) {
  # This function returns a textConnection object for a file
  # from Bluemix Object Storage V3.

  if(!require(httr)) install.packages("httr")
  if(!require(RCurl)) install.packages("RCurl")
  library(httr, RCurl)
  auth_url <- paste("https://identity.open.softlayer.com", "/v3/auth/tokens", sep = "")
  auth_args <- paste("auth": [{"password": ["user": ["domain": ["id": "", "email": "681e6e237043b94c0b04b02014"], "password": "",
  "api_key": "api_key", "name": "", "admin": "2048a46f554f1b04f4e44eb087026960361005", "method": [{"password": ""}], sep=""})
  auth_response <- httr::POST(url = auth_url, body = auth_args)
  x_subject_token <- headers(auth_response)[["X-Subject-Token"]]
  auth_body <- content(auth_response)
  access_url <- unlist(lapply(auth_body[["token"]][["catalog"]], function(catalog){
    if(catalog[["type"]] == "object-store") {
      lapply(catalog[["endpoints"]], function(endpoints){
        if(endpoints[["interface"]] == "public" && endpoints[["region_id"]] == "dallas") {
          paste(endpoints[["url"]], container, filename, sep="/")
        }
      })
    }
  })
  data <- content(httr::GET(url = access_url, add_headers ("Content-Type" = "application/json", "X-Auth-Token" = x_subject_token)), as="text")
  textConnection(data)
}

df <- read.csv(file = getObjectStorageFileWithCredentials("notebooks", "transactions.csv"))
print("A sample of loaded data")
head(df)

No encoding supplied: defaulting to UTF-8.
[1] "A sample of loaded data"

Out[2]:
  PRODUCT_LINE PRODUCT_TYPE CUST_ORDER_NUMBER CITY STATE COUNTRY GENDER AGE MARITAL_STATUS PROFESSION

```

Take the new code (highlighted with the green rectangle) and place it instead of the old code (highlighted with the red rectangle):

```

    })
  })
  data <- content(httr::GET(url = access_url, add_headers ("Content-Type" = "application/json", "X-Auth-Token" = x_subject_token)), as="text")
  textConnection(data)
}

df.data.1 <- read.csv(file = getObjectStorageFileWithCredentials_16456c3a87054c08b2cbeee92cc5ca1("notebooks", "transactions.csv"))
head(df.data.1)

In [2]: #An example of using DSX code insertion features
getObjectStorageFileWithCredentials <- function(container, filename) {
  # This function returns a textConnection object for a file
  # from Bluemix Object Storage V3.

  if(!require(httr)) install.packages("httr")
  if(!require(RCurl)) install.packages("RCurl")
  library(httr, RCurl)
  auth_url <- paste("https://identity.open.softlayer.com", "/v3/auth/tokens", sep = "")
  auth_args <- paste("auth": [{"password": ["user": ["domain": ["id": "", "email": "681e6e237043b94c0b04b02014"], "password": "",
  "api_key": "api_key", "name": "", "admin": "2048a46f554f1b04f4e44eb087026960361005", "method": [{"password": ""}], sep=""})
  auth_response <- httr::POST(url = auth_url, body = auth_args)
  x_subject_token <- headers(auth_response)[["X-Subject-Token"]]
  auth_body <- content(auth_response)
  access_url <- unlist(lapply(auth_body[["token"]][["catalog"]], function(catalog){
    if(catalog[["type"]] == "object-store") {
      lapply(catalog[["endpoints"]], function(endpoints){
        if(endpoints[["interface"]] == "public" && endpoints[["region_id"]] == "dallas") {
          paste(endpoints[["url"]], container, filename, sep="/")
        }
      })
    }
  })
  data <- content(httr::GET(url = access_url, add_headers ("Content-Type" = "application/json", "X-Auth-Token" = x_subject_token)), as="text")
  textConnection(data)
}

df <- read.csv(file = getObjectStorageFileWithCredentials("notebooks", "transactions.csv"))
print("A sample of loaded data")
head(df)

No encoding supplied: defaulting to UTF-8.
[1] "A sample of loaded data"



Out[2]:
  PRODUCT_LINE PRODUCT_TYPE CUST_ORDER_NUMBER CITY STATE COUNTRY GENDER AGE MARITAL_STATUS PROFESSION

```

f. Remove the cell with the newly generated code after replacing the default implementation of getObjectStorageFileWithCredentials


- g. Check point: after the modifications, the section code should still define a data frame variable df which is used in the notebook; the modifications should be done only for replacing getObjectStorageFileWithCredentials with the newly generated code for the new Object Storage service


4.5 Installing Software Libraries and Packages

1. From the DSX home page go to “See All” in “My Recent Notebooks” section
2. Open “ML Lab Installation” notebook in the list by clicking on the name of the notebook
3. Execute every code section in the order in which the sections appear by clicking on the button  or by using the menu Cell> Run Cells
4. Ensure that there are no installation failures before proceeding to the lab
5. Stop the kernel (File > Stop Kernel) and go back to the list of notebooks (click on the Notebooks:  Notebooks >)

NOTE: the software packages installation may take a few minutes, but it needs to be done only once per account

5. Running Decision Tree Lab

1. From the DSX home page go to “See All” in “My Recent Notebooks” section
2. Open “Machine Learning with DSX - Lab” notebook in the list by clicking on the name of the notebook
3. Execute every code section in the order in which the sections appear by clicking on the button  or by using the menu Cell> Run Cells. The lab covers the following actions:
 - a. Declaring the libraries used in the lab
 - b. Loading the data from the Object Storage into a data frame
 - c. Transforming the data for using with C5.0
 - d. Training the classification model (C5.0)
 - e. Transforming the classification model to visualize it in Brunel
 - f. Using a tree map for visualizing and exploring the decision tree in Brunel

- g. Using a tree for exploring the decision tree in Brunel
 - h. Showing the native R visualization of the decision tree for comparison
4. [Optional step] Clean-up the output of all sections to prepare the lab for the next user: click on Cell>All Output>Clear
 5. Stop the kernel (File > Stop Kernel) and go back to the list of notebooks (click on the Notebooks:  Notebooks >)


End of Lab: Decision Tree

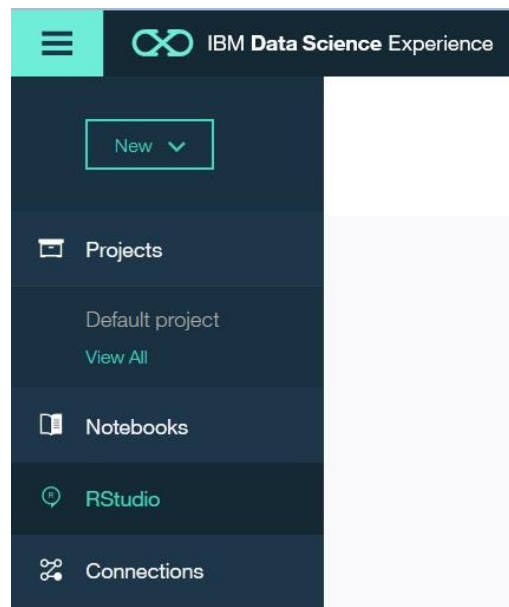
6. Association Rules Lab Installation

6.1 Starting RStudio

1. Select DSX data science mode (please use the highlighted item to get to this menu):

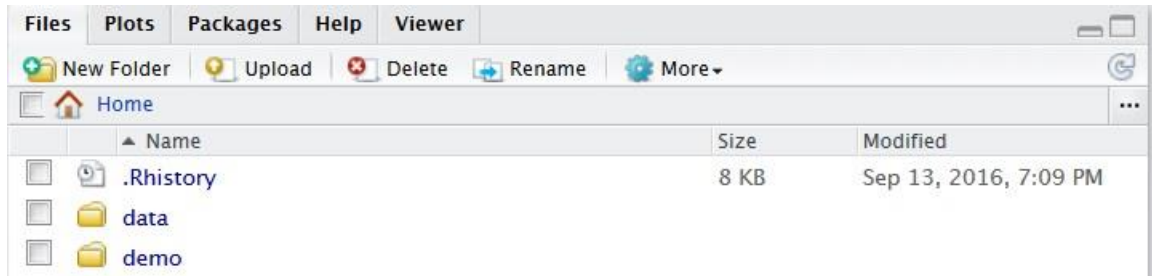


2. Click on the menu icon  on the home page of DSX in Data Science mode
3. Select RStudio in the open tool bar: RStudio session starts



6.2 Importing Source Code and Data for Machine Learning Lab in RStudio

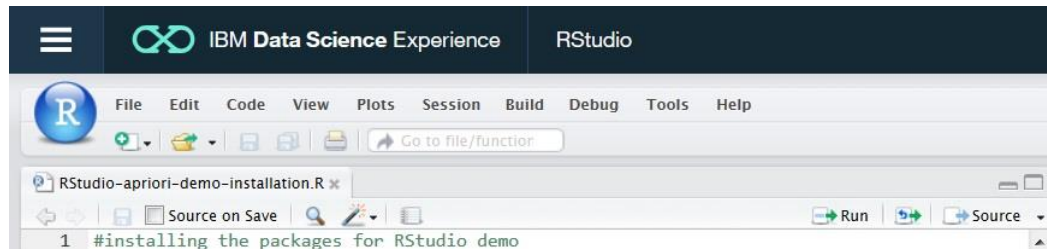
1. In “Files” tab use “New folder” to create 2 folders in the user’s home directory - data and demo (please do not mix it with the “File” menu item in the main menu and locate “Files” in the frame depicted here):




2. Using “Upload” button upload transactions.csv into the data folder and RStudio-apriori-demo-installation.R, RStudio-apriori-demo.R into the demo folder

6.3 Installing Software Libraries and Packages


1. Double-click on the name of the file RStudio-apriori-demo-installation.R: RStudio will open the source code:



2. Run the code in RStudio-apriori-demo-installation.R using the Run button : please decline the options to update any packages while installing the new packages
3. Check point: ensure that all packages install without errors
4. Close the source code editor window for RStudio-apriori-demo-installation.R

NOTE: the software packages installation may take a few minutes, but it needs to be done only once per account

7. Running Association Rules Lab

1. Click on the name of the file RStudio-apriori-demo.R: RStudio will open the source code
2. Execute every code section in the order in which the sections appear by clicking on the button . The lab covers the following actions:
 - a. Declaring the libraries used in the lab
 - b. Loading the sales data into a data frame
 - c. Data wrangling with R: transforming data to the form required by arules package for Apriori algorithm
 - d. Applying Apriori algorithm
 - e. Reviewing the generated rules in the console window
 - f. Visualizing the rules with arulesViz package

3. [Optional step] Quit RStudio

End of Lab: Association Rules
