Lab: R in DSX

November 30th, 2017

Author: Elena Lowery elowery@us.ibm.com

Table of contents

Overview

In this lab you will work with various R assets in DSX Local.

Required software, access, and files

- To complete this lab, you will need access to a DSX Local cluster.
- You will also need to download and unzip this GitHub repository: https://github.com/elenalowery/DSX_Local_Workshop

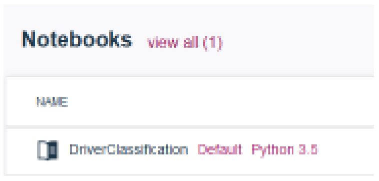
Part 1: Create a DSX Project and run R notebook and Shiny

1. Log in to a **DSX Local cluster**.

2. Navigate to the downloaded GitHub project. Navigate to the *DSX Local Projects* folder and rename *DS_for_Automotive.zip* to a unique name (for example, add your initials).

3. Create a project "from file".

4. Switch to the **Assets** tab. Open *DriverClassification* notebook.



5. Work through the notebook.

Note: notice that in the notebook we save the model into the RStudio directory. We use this approach because when we open RStudio, the model is displayed in the UI, which makes it easier for integration with the Shiny application.

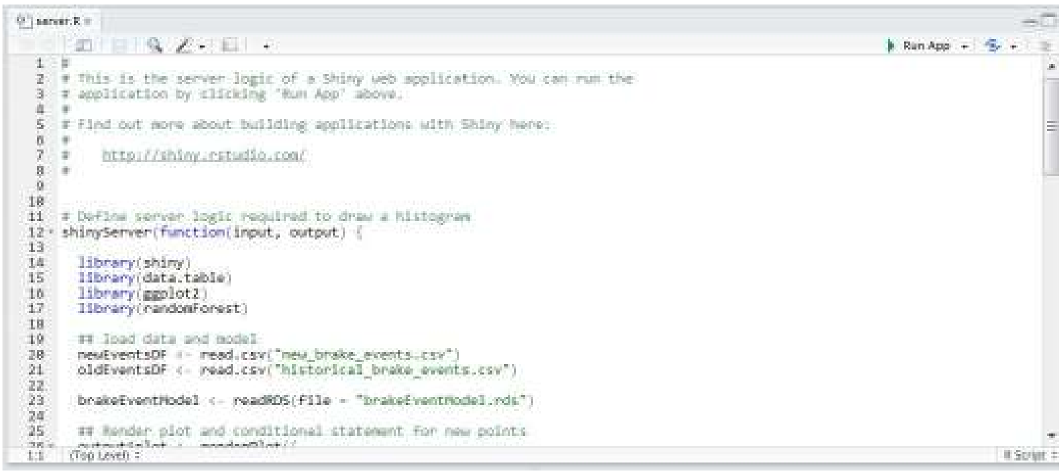
In the sample notebook we overwrite the file each time we run the "save code", but additional code can be added to add a version number to the model name.

At this time DSX doesn't provide any additional "model management features" for R models.

6. Open **RStudio** from the Project view.

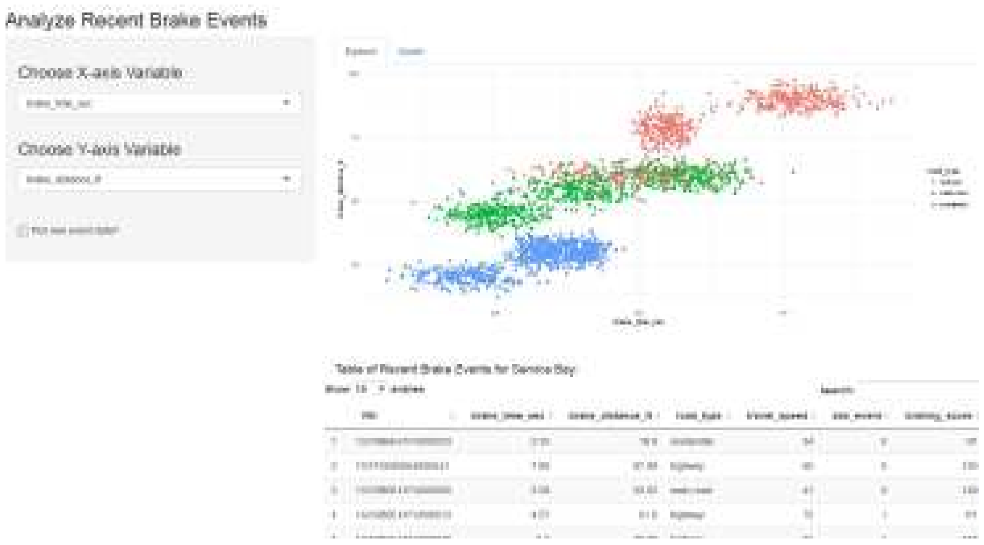
7. If *serverR* file is not already loaded in the top left window, click on the *demoBreakEvents* folder in the file explorer (right bottom window), then click on *serverR*.

   After the application is loaded, click the **Run App** button.



```r
#
# This is the server logic of a Shiny web application. You can run the
# application by clicking 'Run App' above.
#
# Find out more about building applications with Shiny here:
#
#    http://shiny.rstudio.com/
#


# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  library(shiny)
  library(data.table)
  library(ggplot2)
  library(randomForest)

  ## load data and model
  newEventsDF <- read.csv("new_brake_events.csv")
  oldEventsDF <- read.csv("historical_brake_events.csv")

  brakeEventModel <- readRDS(file = "brakeEventModel.rds")

  ## Render plot and conditional statement for new points
```

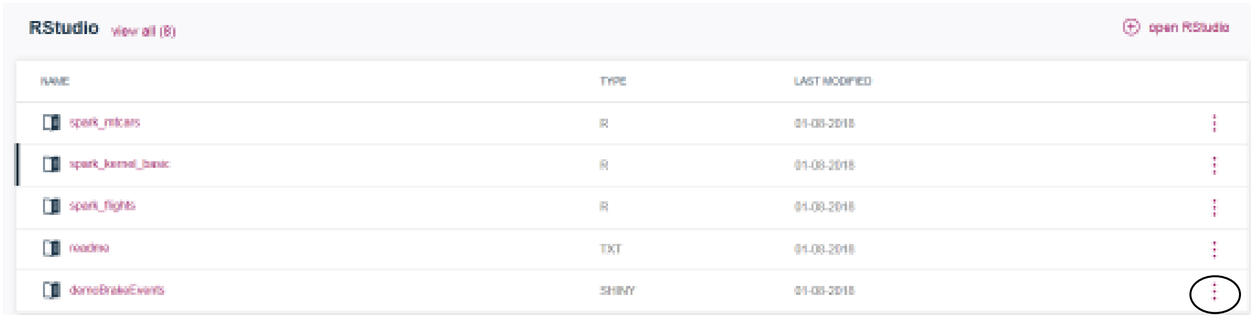
8. Select **Allow popups** to bring up the application. Test the application - both the Explore and Model tabs.


Analyze Recent Brake Events
Choose X-axis Variable
Choose Y-axis Variable
Table of Recent Brake Events for Service Bay

Part 2: Publish Shiny

In this section we will publish the Shiny application.

1. Navigate back to the **Project** view and click **Preview** from the ellipses next to the Shiny application (*demoBreakEvents*).



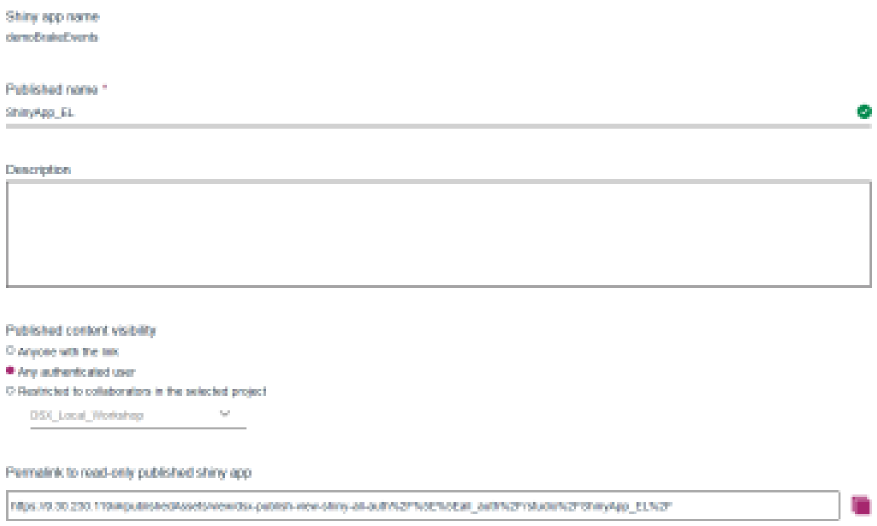
RStudio view all (8) — open RStudio

NAME	TYPE	LAST MODIFIED	
spark_mtcars	R	01-08-2018	⋮
spark_kernel_basic	R	01-08-2018	⋮
spark_flights	R	01-08-2018	⋮
readme	TXT	01-08-2018	⋮
demoBrakeEvents	SHINY	01-08-2018	⋮

2. If you get an error during preview, check with the lab instructor.

3. From the same menu click **Publish**. Provide *Published name* (try to make it unique, for example, add your initials) and select *Published content visibility* that you would like to test. Click **Publish**.

   Note: All Shiny applications in DSX are published to a shared Shiny server. At this time there is only 1 instance of the Shiny server in DSX (i.e. it's not configured for HA). However, because the Shiny server is deployed as a pod, Kubernetes will monitor its status and restart it, when needed.



Shiny app name
demoBrakeEvents

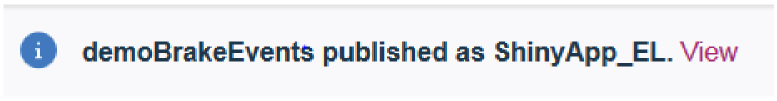
Published name *
ShinyApp_EL

Description

Published content visibility
○ Anyone with the link
[✓ Any authenticated user]
○ Restricted to collaborators in the selected project
DSX_Local_Workshop

Permalink to read-only published shiny app
[illegible]

4. Click on the status message (**View**) or navigate to **Published Assets** from the main DSX menu.

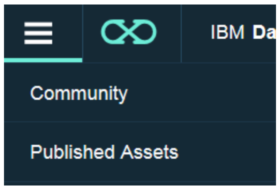


demoBrakeEvents published as ShinyApp_EL. View

5. Verify that the application works.

6. Depending on the "scope" that you published to, the Shiny application can be accessed through different views in DSX.

   - If selected *anyone with the link* or *any authenticated user*, then you can view it in the **Published Assets** accessed through the main DSX menu (in the top left corner)



   - If you published it to a specific project, then it will show up in the **Published Assets** tab of the project you published **to** (not published **from**).

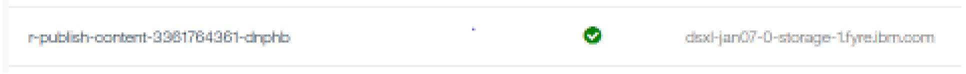
Part 3: Check Status of Shiny Server

At this time the Shiny Server is deployed as a pod in DSX, and the only way to view status is by looking at the status of the pod. You must have admin privileges to perform these functions.

There is only 1 instance of Shiny Server, but Kubernetes monitors pod status, and the pod should be automatically restarted if it goes down.

1. Switch to the **IBM Data Platform Manager** view

2. Click on Pods. Scroll down to locate the *r-publish… pod*. This is the Shiny Server.



r-publish-content-3381764361-dnphb ✓ dsxl-jan07-0-storage-1.fyre.ibm.com

3. If the pod is not running, you can redeploy it from this UI or via Kubernetes commands in ssh.

   Display all pods: `kubectl get pods --all-namespaces`

   Delete pod: `kubectl delete pod <pod name> -n <namespace>`

   Force delete for a pod (if it’s stuck in “terminating” state):
   ```
   kubectl delete pod <pod name> -n <namespace> --grace-period=0
   ```