Lab: R in Watson Studio Local

Dec 26th, 2018

Author: yfphoon@us.ibm.com

© Copyright IBM Corporation, 2017. All Rights Reserved.
All trademarks or registered trademarks mentioned herein are the property of their respective holders

Table of contents

Contents

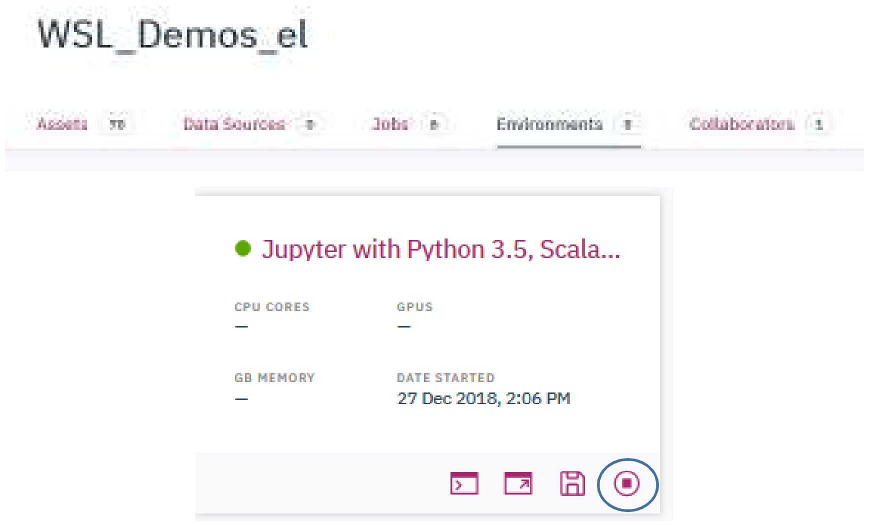
Overview

In this lab you will learn how to work with R assets in **Watson Studio Local.**

Required software, access, and files

- To complete this lab, you will need access to a **Watson Studio Local** cluster.
- You will also need to complete the following steps to import the sample project:
    - Download and unzip this GitHub repository: https://github.com/elenalowery/WSL-Workshop (if you haven't already done it in the previous lab)
    - In the **Projects** directory of the unzipped file, rename *R_Lab.zip* to a unique name, for example, add your initials.
    - Log in to WSL and create a project **From File**, using the *R_Lab.zip* file that you just renamed.
- Stop all environments that were used for other projects – this is important when sharing the cluster with other users (it will help minimize workload on the cluster).
    - In **Watson Studio** click on your project (for example, *WSL_Demos*), then **Environments**. Stop all environments that are running.

Note: do not select Save Custom Image when stopping the environment.



R in Watson Studio Local

There are two R Integrated Development Environments (IDE) in Watson Studio Local (WSL), **Jupyter** notebooks and **RStudio**. The objective of this lab is to familiarize you with the two IDEs for reading data, model building, model export, and model invocation via REST API.

The R code in **Part 1** and **2** is similar, but there are differences in retrieving data and viewing outputs that you will be aware of after working through this lab.

Part 1: Build and Save R model in Jupyter Notebook

In the **R Lab** project, open the **DriverClassification** notebook.

- Review the instructions, comments and linked documentation in the notebook
- *Follow the instructions in the notebook to add code where needed*
- *Execute each line of code in the notebook*

Part 2: Build and Save R model in R Studio

In this part of the lab, we will build the same model as we did in **Part 1**, but we will use **RStudio** to build and save the model.

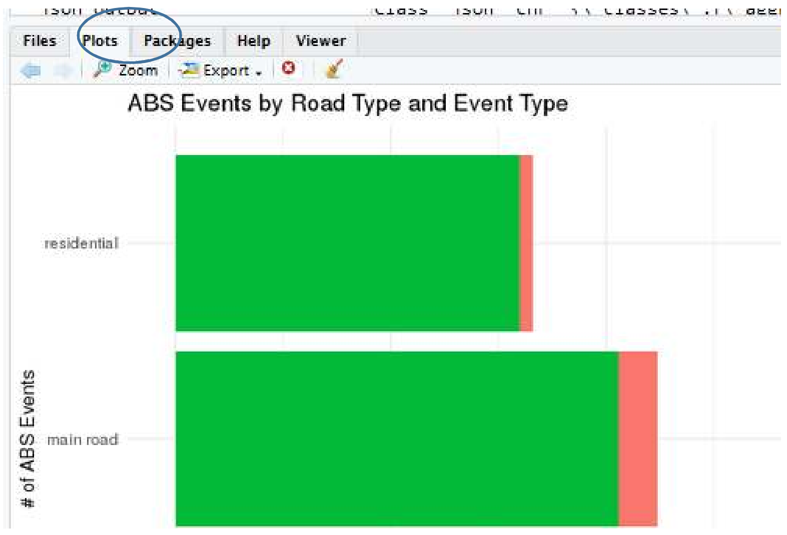
The R code in the script is mostly the same as the code in the *DriverClassification* Jupyter notebook. The main differences are:

1. Where the csv file is read from. In a notebook, the csv file is read from the *Project datasets* folder. In RStudio, the csv file is read from the *RStudio project*. When a customer is migrating a project from an external RStudio environment, they may prefer to keep csv files as a part of RStudio project.

    - It is possible to read csv files from the WSL Project. If you would like to do that, use the following code:

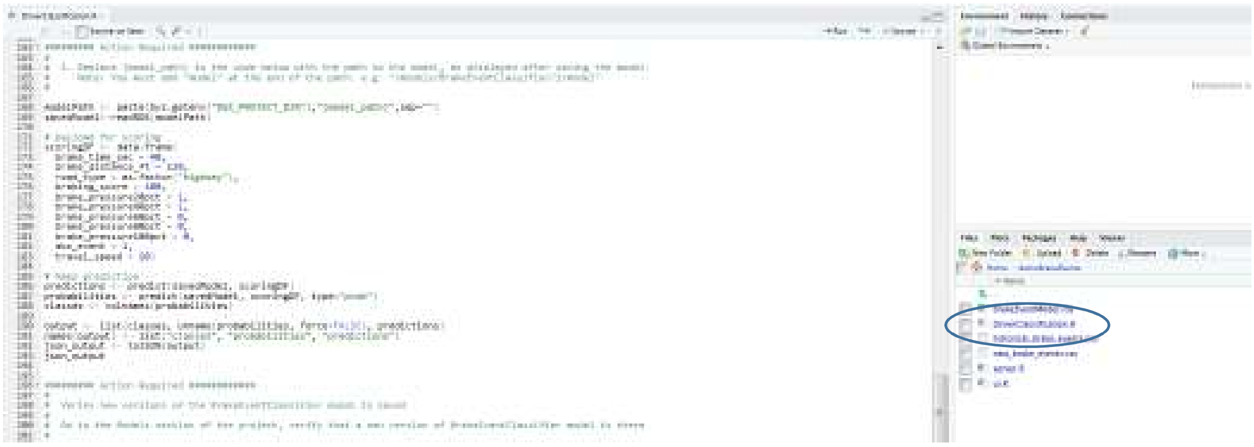
      ```
      df.data.1 < -
      read.csv(paste(Sys.getenv("DSX_PROJECT_DIR"),'/datasets/historical_brake_event
          s.csv',sep=""))
      ```

2. The graphs rendered by *ggplot* are displayed in the *Plots* tab of RStudio.



1. In the **R Lab** project click on **RStudio**, then click **Open RStudio**.

2. In **RStudio** in the **Files** view click on *DriverClassification.R* to load it.



3. Review the instructions and comments in the R script

   - *Follow the instructions in the script to add code where needed. Look for* `# ########## Action Required ##############` *as an indicator that you need to edit or add some code.*

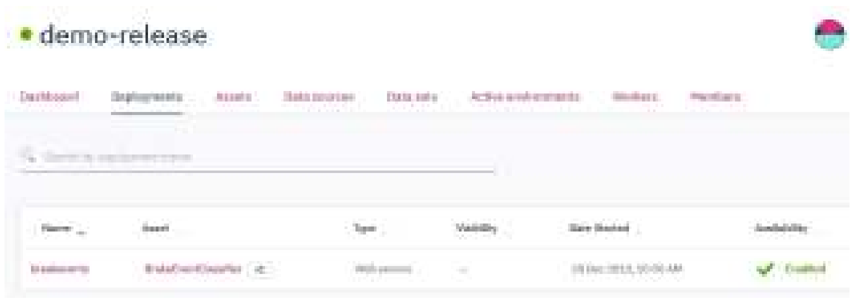
   - *Execute each line of code in the script*

Part 3: Optional Exercises

In this exercise you will deploy the **BrakeEventClassifer** model in WML and invoke the model through an API call. Then, you will edit the code in the Shiny App, so that it will invoke the model through an API call and display the results in the app:

1. Deploy your *R_Lab* project in WML.

2. Create a Web service deployment for the **BrakeEventClassifer** model. Don't forget to **launch** the project. Verify that the Web service deployment is *enabled*.

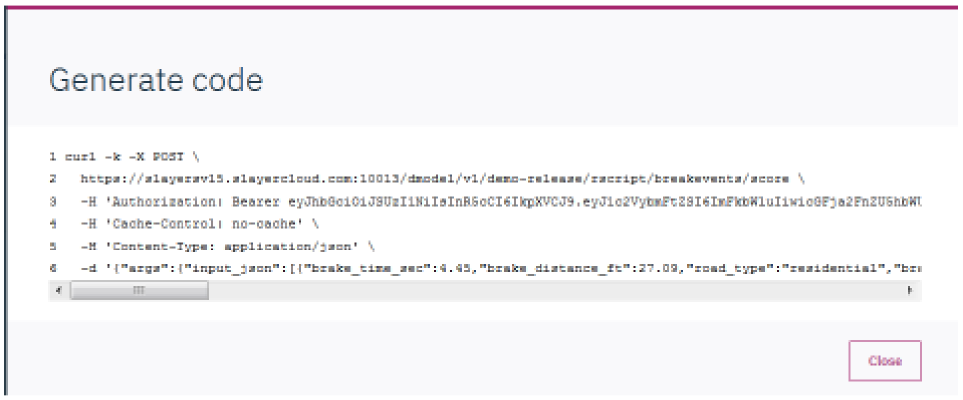


demo-release

Dashboard · Deployments · Assets · Data sources · Data sets · Active environments · Workers · Members

Name	Asset	Type	Validity	Date Started	Availability
breakevents	BrakeEventClassifier	Web service	–	[illegible]	✓ Enabled

3. Click on the deployment to bring up the deployment details. Click **Generate Code** and copy the curl command. We will use this information to invoke the Web service from the notebook and the Shiny app.



Generate code

```
curl -k -X POST \
  https://slayersv15.slayercloud.com:10013/dmodel/v1/demo-release/rscript/breakevents/score \
  -H 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwic0Fja2Fn2U5hbWl
  -H 'Cache-Control: no-cache' \
  -H 'Content-Type: application/json' \
  -d '{"args":{"input_json":[{"brake_time_sec":4.45,"brake_distance_ft":27.09,"road_type":"residential","br
```

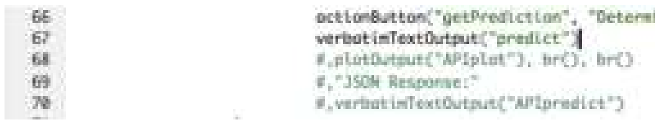
4. Switch back to Watson Studio environment. In the *DriverClassification* notebook, go to the *Optional Exercise* section at the bottom of the notebook, and follow the instructions in that section of the notebook

5. In **RStudio**, open *demoBrakeEvents/server.R*. Review the code, there are two functions, *toPredict()* which gets the predicted value from the model that is saved in RStudio, and *toPredictAPI()* gets the predicted values via an API call to the deployed model.

6. Edit the code in line 95 to replace URL of the model endpoint and replace the *Authorization Bearer* token with your generated values. (**Hint**: You may copy the

value assigned to "curl_left" from the optional exercise in the DriverClassification notebook)

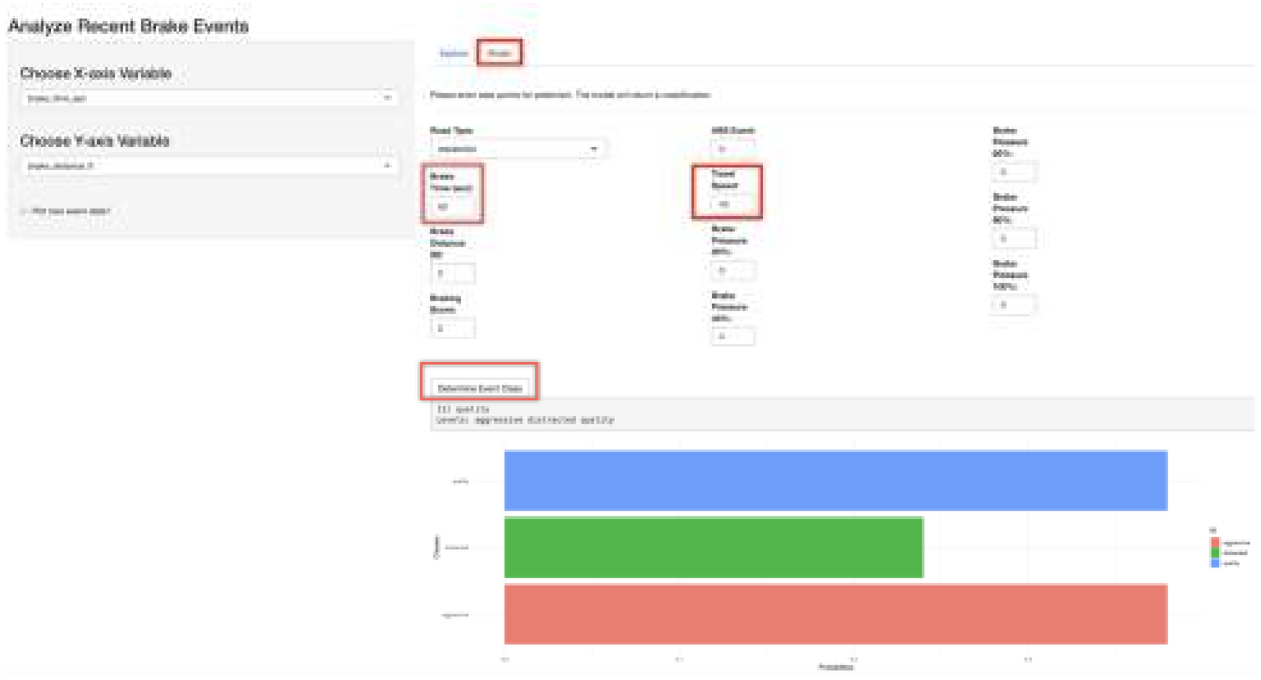
7. Open *demoBrakeEvents/ui.R*, uncomment the line 68 to 70 and save.



```
66                      actionButton("getPrediction", "Determine Event Class"),
67                      verbatimTextOutput("predict")
68                      #,plotOutput("APIplot"), br(), br()
69                      #,"JSON Response:"
70                      #,verbatimTextOutput("APIpredict")
```

8. Run the Shiny App. To see the different predictions for the input values:

   a. Click *Model*
   b. Enter different values, e.g. *Travel Speed=45*, *Brake Time=60*



```
[1] quality
Levels: aggressive distracted quality
```

Summary

You have finished the **R in WSL** lab.

In this lab you completed the following steps:

- In a notebook, you have added code to read data from the project data asset, build an R caret model and saved that model in the RStudio file system as well as WSL repository
- In RStudio, you have added code to read data from the RStudio file system, build an R caret model and saved that model in the RStudio file system as well as WSL repository

- You have deployed the project, created a deployment for the model and invoked the model through API call.
