Azure ML

RStudio Web Service API

Demo Guidelines

# Introduction

Currently, millions of Data Scientists and developers are using RStudio to build and train Machine Learning models but they lack the ability to share them via web services. Ideally, a user should be able to publish a web service without ever having to leave RStudio. This “In Progress” project will provide that link between RStudio and AzureML's publishing service in order to encourage RStudio users to utilize AzureML web services. Essentially, this will provide a seamless way of sharing data so that when data does need to be shared, a link can be shared with another user and the data is then duplicated for their use.

The three main goals of this project is to provide the RStudio user with Discovery, Consumption and Publishing abilities. This package will be available to users as a downloadable CRAN as long as you have an authorized workspace in AzureML that is discoverable.

# Discovery

Currently, the discovery code allows the user to retrieve a list of the web services available in their workspace given that they provide the workspace ID and the authorization token (both of which can be found in settings on the AzureML webpage). A user can also get detailed information about a specific web service, retrieve its endpoints, and the details of a specific endpoint.

# Consumption Spiel

Consuming a web service currently gives the user the option to score either a CSV file, data frame or individual requests. With either option, the user simply has to make a single-line function call and the scored probabilities are returned to the user in a data frame. There are three functions that the user can choose from. The consumeLists function takes in the API key of the published experiment, its request URL, column names, and a variable number of requests entered in data table format. The option to enter the requests in key-value format is currently being added. It also returns the scored probabilities in a data frame. The consumeFile function takes in an API key, file name and a request URL. The consumeDataFrame function takes in an API key, request URL and data frame. All functions have the option of varying the global parameters and delay time before retrying a server call in case of a server error. The functions that score a data frame and lists have the option of varying the batch size of each request sent to AzureML. Hence, the user can consume a web service from AzureML regardless of the format in which their data is stored in a single-line function call.

# Publishing Spiel

Publishing is a simple one call function for the user. Upon calling, the user simply needs to provide the workspace they want their code published to, along with the authorization token for this workspace. Additionally, they user will specify the function name and the name they would like the service saved under (serviceName). The most work they will need to put into calling publish is creating an input and output schema. The schemas will be expected to be a list that looks like ***inputSchema = list("arg1"="type", "arg2"="type", ...)***. Upon calling the publish function, this will handle the API call for the user and any consumption that may need to be done. Once the call is finished, the function will return a list with the web service details, the endpoint detains and the consumption function.

# Moving onto the Demo

For demoing purposed, we will be using a Titanic demo set. We will show you the process of calling the publishing web services and walk you through the process. **(Now demo code)**

**In this demo, we will begin in RStudio so that you can see that this is can all be done from your existing workspace. From within RStudio we will make a call to the publish function, then access the published code from the returned URL and from which we will be able to move to VisualStudio to work.**

**One of the first things we will want to do is gather your workspace ID and Authorization Token from your personal AzureML account. It will be most convenient to save these somewhere as you will be using them frequently throughout your work with our CRAN package.**

(SLIDE 2)

(SLIDE 3) **This is the functions any user can access and will be able to call. There will be official documentation available to the user within the CRAN package explaining the purpose of each function and the parameters and output.**

(SLIDE 4) **When a function asks for a schema, the function is expecting a list of the following sorts.**