**How to Create an Entry Using Your Favorite R Environment and   
Submit to Cortana Intelligence “Decoding Brain Signals” Competition**

In this tutorial, you will build a solution in your favorite R environment outside of Azure ML, and then create a valid entry to enter the *Decoding Brain Signals* Competition. You can incorporate the R scripts given in the Starter Experiment provided for this competition for feature engineering, and then create your own for model training and validation. You should also feel free to engineer other features based on your own understanding of the task and the data.

1. **Download the training data**

Find the link in the Data Description section of the Competition information and download the training dataset. In this tutorial, it is saved to local directory on your PC as *“E:\Brain\_Competition\_OnPrem\ecog\_train\_with\_labels.csv”*

1. **Copy the Starter Experiment**

Enter the competition by following steps 1 and 2 in tutorial “[15 Minutes to Build Your First Solutions for the Inaugural Microsoft Cortana Intelligence Competition: Decoding Brain Signals](http://az754797.vo.msecnd.net/competition/ecog/docs/Tutorial_1.docx)”.

The important step here is that you make a copy of the Starter Experiment into your Azure ML workspace. The R scripts included in the Execute R Script modules of the Starter Experiment cover feature engineering. However, model training, predicting, and evaluation are implemented using built-in modules in Azure ML.

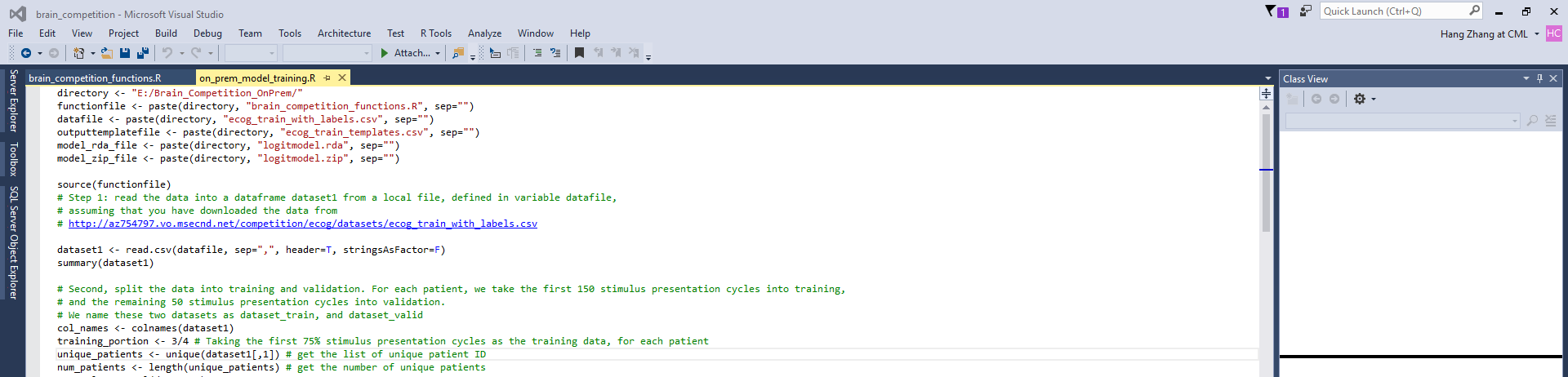
In this tutorial, you will learn how to write R scripts to implement these steps and bring the trained model into Azure ML for submission.

1. **Download the R script files**

From the Tutorial section of the Competition page, download the zip file that includes 2 R script files that will help you get started: [*on\_prem\_model\_training.R*](http://az754797.vo.msecnd.net/competition/ecog/code/on_prem_model_training.R) and [*brain\_competition\_functions.R*](http://az754797.vo.msecnd.net/competition/ecog/code/brain_competition_functions.R)and unzip them locally. The *brain\_competition\_functions.R* script defines two functions: *fh\_get\_events()* and *fh\_project\_2\_templates()*. (These two functions are also used in the *Execute R Script* modules in the Azure ML Starter Experiment for this competition.) Save these two R script files to the same directory of the downloaded training data. In this example, it is “*E:\Brain\_Competition\_OnPrem*”.

1. **Open both R script files in R Tools for Visual Studio or RStudio.**

Open these two R script files in an IDE of your choice for R development. This tutorial uses [R Tools for Visual Studio](https://www.visualstudio.com/en-us/features/rtvs-vs.aspx) (RTVS), which allows you to develop and run R scripts in Visual Studio. But you can use RStudio or other tools, and you can go bare-knuckle R as well if you’d like. After opening both files in RTVS, you will see the following UI:



1. **Run the R script file on\_prem\_model\_training.R**

Before running the R script file on\_prem\_model\_train.R, make sure you that you have packages *glmnet* and *abind* installed. If these two packages are missing, use the following two lines to install them:

*install.packages(“glmnet”)*

*install.packages(“abind”)*

This R script file *on\_prem\_model\_training.R* is the main file to run in this tutorial to build an on-premises solution for this competition. If your data and R script files are downloaded to a directory other than “*E:\Brain\_Competition\_OnPrem*\”, you need to update the variable *directory* in the first line of this file.

*directory <- "E:/Brain\_Competition\_OnPrem/"*

This R script file has the following blocks in sequence:

* 1. **Read the data from a local file into a data frame**

After the data is read into a data frame named *dataset1*, use *summary(dataset1)* to understand the basic statistics of each variable.

* 1. **Split the data into training set and validation set**

Here, the entire training data is split into training set and validation set. For each patient, the first 150 (3/4) of the 200 stimulus presentation cycles are put in the training set (*dataset\_train*), and the remaining 50 stimulus presentation cycles are put in the validation set (*dataset\_valid*).

* 1. **Create raw signal templates for each channel, stimulus type, and for each patient**

In this step, the raw signal templates for each channel, stimulus class (1 or 2), and for each patient are calculated based on the dataset *dataset\_train*. Generally speaking, the templates are just the average of signals between 200 milliseconds before the onset time of a stimulus class and 399 milliseconds after the onset time, after aligning all presentation cycles of the same stimulus class on the stimulus onset time.

After the templates are calculated, save the template data *templates* as a local file named *ecog\_train\_templates.csv*, in the same directory as the data and R script files. You will need this template file later when you operationalize the solution in Azure ML.

*write.csv(templates, file = outputtemplatefile, row.names=FALSE)*

* 1. **Project raw signals to the templates as features**

In this step, we directly call the function *fh\_project\_2\_templates*() defined in *brain\_competition\_functions.R* to calculate the features for both *dataset\_train* and *dataset\_valid*.

Name these two feature sets *erp\_train* and *erp\_valid*.

*erp\_train <- fh\_project\_2\_templates(dataset\_train, templates)*

*erp\_valid <- fh\_project\_2\_templates(dataset\_valid, templates)*

* 1. **Train a logistic regression model on the training feature data**

Now you are ready to train a logistic regression model using the *glmnet*() function. First, convert the label from 1 and 2, to 0 and 1, since *glmnet()* in R only takes 0 and 1 as the targets for binary classification model. Keep in mind you need to convert the labels back to 1 and 2 later.

Use the following two lines of code to convert the labels in both training feature set *erp\_train* and validation feature set *erp\_valid* to 0 and 1:

*erp\_train[,ncols-1] <- erp\_train[,ncols-1] - 1*

*erp\_valid[,ncols-1] <- erp\_valid[,ncols-1] - 1*

Then, generate the formula for the logistic regression model:

*formula <- paste(col\_names[2:(length(col\_names)-2)], collapse="+")*

*formula <- paste("Stimulus\_Type ~ ", formula, sep="")*

Then, train a logistic regression model with L1 regularization (a LASSO model), and get summary of this model:

*glmnetmodel <- glmnet(x=as.matrix(erp\_train[,2:(ncols-2)]), y=erp\_train[,ncols-1], alpha=1, nlambda=1, lambda=0.01) #train a LASSO model, where lambda=0.01*

*summary(glmnetmodel)*

To know how the model performs on the holdout validation data, apply this model *glmnetmodel* to predict the validation data, and calculate the performance (accuracy). Please note that accuracy is the performance metrics used to rank entries in this competition. This tutorial script yields a decent accuracy value. Your job is to figure out creative ways to improve this number without overfitting.

*valid\_pred <- predict(glmnetmodel, newx = as.matrix(erp\_valid[,2:(ncols-2)]), type="response")*

*valid\_pred[valid\_pred >= 0.5] <- 1*

*valid\_pred[valid\_pred < 0.5] <- 0*

*index <- erp\_valid[,ncols-1] == valid\_pred*

*print(paste("Validation accuracy = ", round(sum(index)/length(valid\_pred)\*100,4)), sep="")*

1. **Save the model object to a local *.rda* file**

Save the logistic regression model as a local .*rda* file.

*model\_rda\_file <- paste(directory, "logitmodel.rda", sep="")*

*save(glmnetmodel, file = model\_rda\_file)*

Make sure the model file *logitmodel.rda* is saved in same directory as the data file and R script files

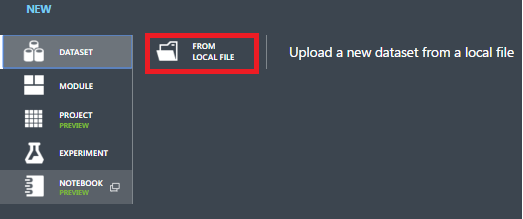
1. **Zip the .rda file and the brain\_competition\_functions.R into a .zip file**

Go to the directory where you store the R script and the .rda files, *“E:\Brain\_Competition\_OnPrem”* and zip *logitmodel.rda* and *brain\_competition\_functions.R* into a new zip file named *logitmodel.zip*.

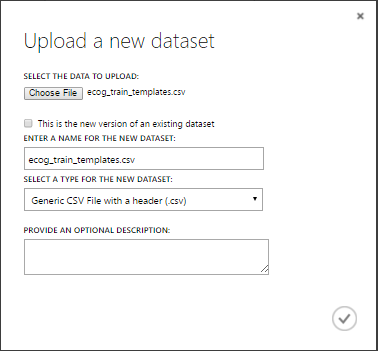
1. **Upload the files into the Azure ML** 
   1. Go to your workspace in Azure ML Studio where you copied the Starter Experiment into, and click the **“+ NEW**” button at the left bottom corner of the page.



Then, select **DATASET** and **FROM LOCAL FILE**



* 1. Upload these two files *ecog\_train\_templates.csv* and *logitmodel.zip* from your local directory one by one:

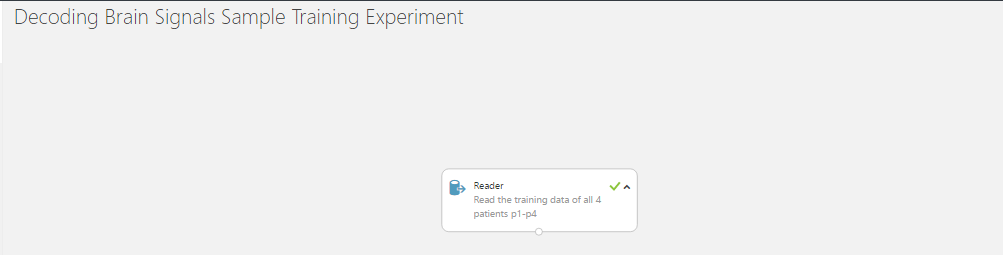


The dialog box will automatically infer the data file type based on the file extension, which are in this case CSV and ZIP. For CSV files, choose the type “Generic CSV File with a header” since the template has a header row. You can also give them new names if you want. But in this tutorial, accept the default names.

1. **Build a predictive experiment in Azure ML to operationalize the model**

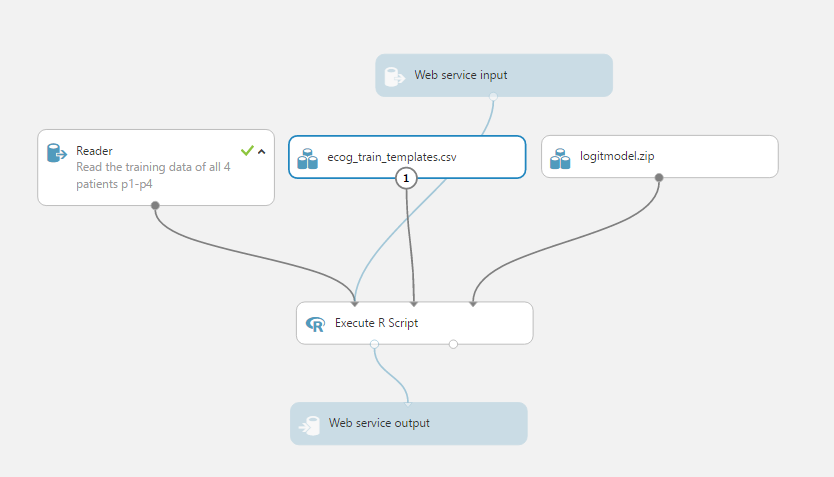
In this competition, testing data is not shared with you. Instead, you will need to create a web service API from the R code you uploaded, and let the evaluation process invoke it in to make a prediction on the testing data. The web service API is created and deployed out of a predictive experiment. Therefore, you will need to create a predictive experiment first that is able to generate the same set of features from the test data as from the training data in the training process, and call the model to make predictions based on the features of test data.

* 1. Open the Starter Experiment you copied to your workspace when you entered the competition in Step 2. Keep the Reader module, and delete all other modules. Save this experiment using a different name.



Please note that you should not build your predictive experiment from scratch via +New > Experiment because it doesn’t carry metadata for this competition, and therefore cannot be used to generate a valid entry for this competition.

* 1. Add an *Execute R Script* module, and add the *logitmodel.zip* and the *ecog\_train\_templates.csv* that you just upload in Step 8 to the experiment from the **Saved Dataset**, **My Dataset** section in the toolbox. Also, add a Web service input module and a Web service output module to the experiment. Connect them as follows:



* 1. Replace the R script in *Execute R Script* module with the following scripts. Please note the *logitmodl.zip* file is automatically unzipped and the two R scripts are dropped into the *src* folder of the sandbox R runtime in Azure ML, which is why you can reference them directly. See [this article for more information](https://azure.microsoft.com/en-us/documentation/articles/machine-learning-r-quickstart/) on how to work with R in Azure ML.

*dataset1 <- maml.mapInputPort(1) # class: data.frame*

*dataset2 <- maml.mapInputPort(2) # class: data.frame*

*library(glmnet)*

*# source the functions in defined in brain\_competition\_functions.R, which is archived in the logitmodel.zip*

*source('src/brain\_competition\_functions.R')*

*# Project data to templates (dataset2)*

*erp\_data <- fh\_project\_2\_templates(dataset1, dataset2)*

*print("load the trained model from rda file...")*

*load('src/logitmodel.rda') #load the .rda file into R, as logitmodel model object*

*#print(summary(glmnetmodel)) #you can use this line to check whether the model has been successfully loaded*

*ncols <- ncol(erp\_data)*

*valid\_pred <- predict(glmnetmodel, newx = as.matrix(erp\_data[,2:(ncols-2)]), type="response") #make predictions on the data erp\_data*

*valid\_pred[valid\_pred >= 0.5] <- 2 # rescale the predict results back to 1 and 2*

*valid\_pred[valid\_pred < 0.5] <- 1*

*valid\_pred <- as.matrix(valid\_pred, nrow=nrow(erp\_data), ncol=1)*

*ncols <- ncol(erp\_data)*

*data.set <- data.frame(as.character(erp\_data[,1]), erp\_data[,ncols], valid\_pred, stringsAsFactors = F) #only output three columns: PatientID, Stimulus\_ID, and Scored Labels, as required by the competition*

*colnames(data.set) <- c("PatientID", "Stimulus\_ID", "Scored Labels")*

*# Select data.frame to be sent to the output Dataset port*

*maml.mapOutputPort("data.set");*

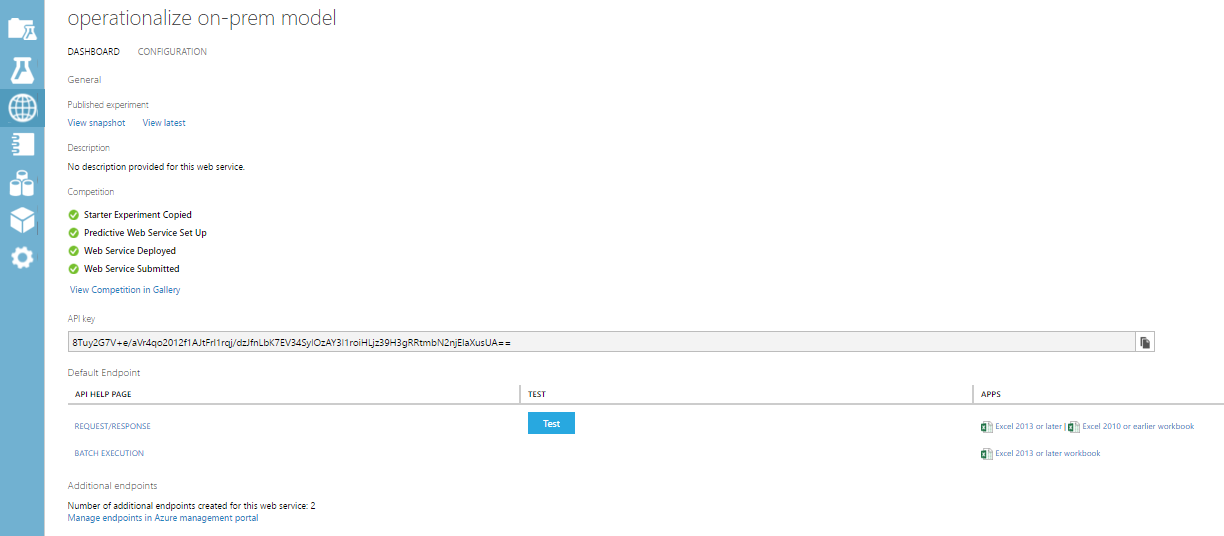
* 1. Run the experiment

Click the Run button at the bottom of the studio, the experiment will start running. It might take around 2 - 5 minutes to complete.



1. **Deploy web service, and submit for evaluation**

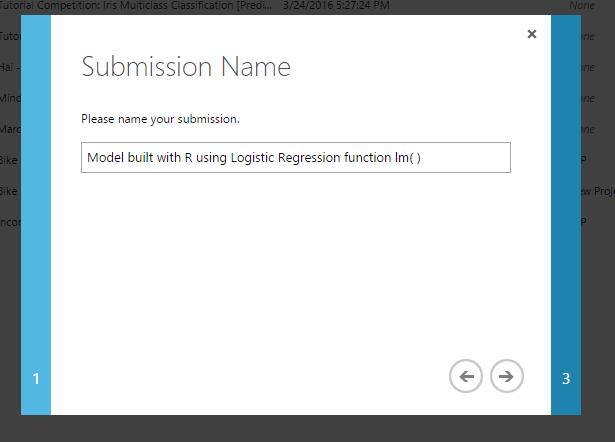
After the experiment completes successfully, click “DEPLOY WEB SERVICE”, a web service API will be created from this predictive experiment.



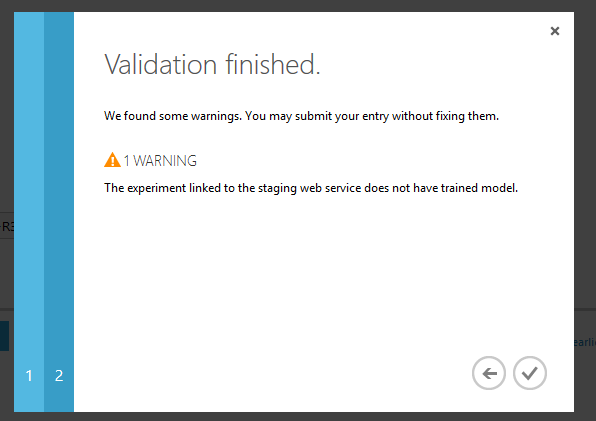
Click the SUBMIT COMPETITION ENTRY button of the web service API page, an entry submission wizard will be launched and walk you through the steps to submit.



One quick tip here is to properly name your entry. This competition allows you to submit multiple entries. The name, once submitted, cannot be changed. And this name is visible only to yourself. So you might consider an easily recognizable name for your own reference.



Also, you will likely see the following warning upon validation in the wizard. Simply ignore it. The reason of the warning is that it can’t detect a Trained Model module in the graph. This is OK since we create our trained model using R into the *.rda* file. There is no Trained Model module produced from a training experiment.



1. **Improve your model and resubmit a new entry.**

After you are able to successfully submit the first entry, you can go back to step 5 and start to refactor your script to achieve higher accuracy in your R code. You can then repackage it up and upload it into Azure ML. You can overwrite the same .csv and .zip files when uploading. But please make sure you remove the old ones from the experiment before re-adding the updated ones back. This is because Azure ML has a versioning capability that it remembers old versions of the uploaded assets until you physically remove them from the graph. Then you can re-run your experiment, re-deploy (essentially update) your web service, and submit a new entry.