# 15 Minutes to Build Your First Solution for the Inaugural Microsoft Cortana Intelligence Competition : Analyzing Brain Signals

## **Introduction**

This competition asks you to build machine learning models in Microsoft Cortana Intelligence Suite to decode perceptions of human subjects from brain, specifically Electrocorticographic (ECoG) signals. The data used in this competition was collected from four human subjects. 300 gray-scale images of houses (labeled as image class 1) or faces (labeled as image class 2), were displayed in a random order on a screen to the patients. ECoG signals were collected from the cortical surfaces of these patients during the experiments. The competition is to decode visual perceptions of these subjects from the ECoG signals, to predict whether the patient is seeing a house image (class 1) or a face image (class 2).

Similar work on another set of 7 patients has been published at [PLOS Computational Biology](http://journals.plos.org/ploscompbiol/article/metrics?id=10.1371/journal.pcbi.1004660). The data we used in this competition was provided by the author of this paper, Dr. Kai J. Miller. This data was collected from 4 patients at the same experiment as the 7 patients in his paper. But the 4 patients whose data is used in this competition does not overlap with the 7 patients in his paper. However, reading through Dr. Miller’s paper may be helpful for you to understand the experimental settings, the ECoG signals, and features that were created in the machine learning experiment.

The data from these 7 patients and the Matlab codes that analyzed the data and built the machine learning models to decode the perceptions from the ECoG signals have been shared by the authors of this paper at [Stanford Digit Repository](http://purl.stanford.edu/xd109qh3109). In the paper, the authors generated two sets of features from the ECoG signal: the event-related potentials (ERP, also called averaged raw potentials), and the event related broadband changes (ERBB), and reached high accuracy (> 95%) on 3-fold cross-validation. The sample training experiment (described below) only implemented the ERP signals.

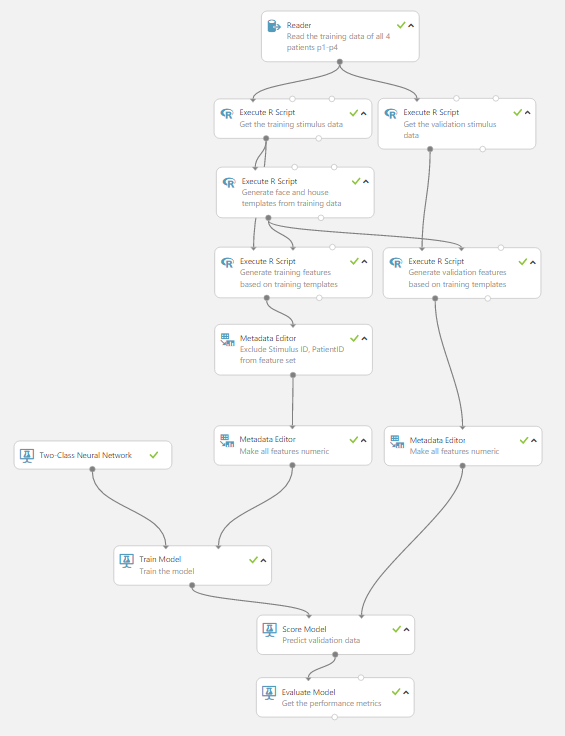
Azure Machine Learning (AzureML), a component of Microsoft’s Cortana Intelligence Suite, is a cloud-based tool that enables data scientists and big data professionals to build and operationalize predictive analytics solutions with ease. The following tutorial is a starting point for those who are interested in participating in this competition and learning about the diverse capabilities of AzureML. The sample experiments that we provide below can be easily accessed from the [Azure ML Gallery](http://gallery.azureml.net/). You can use these as a baseline and start building upon them in the rich AML Studio environment by dragging and dropping an extensive set of available algorithms or your own custom R and Python scripts.

## **Overview of the Sample Solution in this Tutorial**

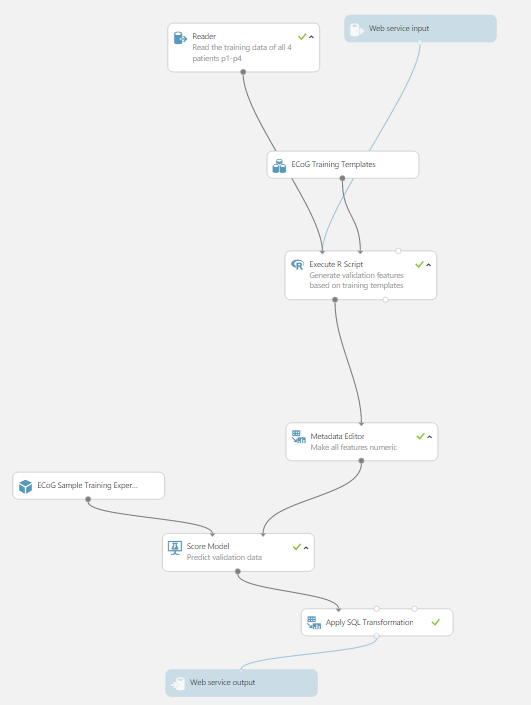
This tutorial provides instructions on how you can build a qualifying solution for this competition in just 15 minutes based on a sample training experiment we share with you in the Gallery of AzureML. With only some clicks and drag-and-drop actions, you can submit your first entry for this competition and you will be able to see yourself on the leaderboard!

First, we provide you with a sample training experiment. In the next section, you will use Azure ML to generate a predictive experiment that is built from the training experiment.

1. **Training experiment** Analyzing Brain Signals (Running Time: 7 minutes)



1. Predictive experiment. Analyzing Brain Signals [Predictive Exp.] (Running Time: 3 minutes)



## Five Steps to Build Your First Solution in 15 Minutes

Here are the five steps that you can take to build your first machine learning solution in 15 minutes for this competition.

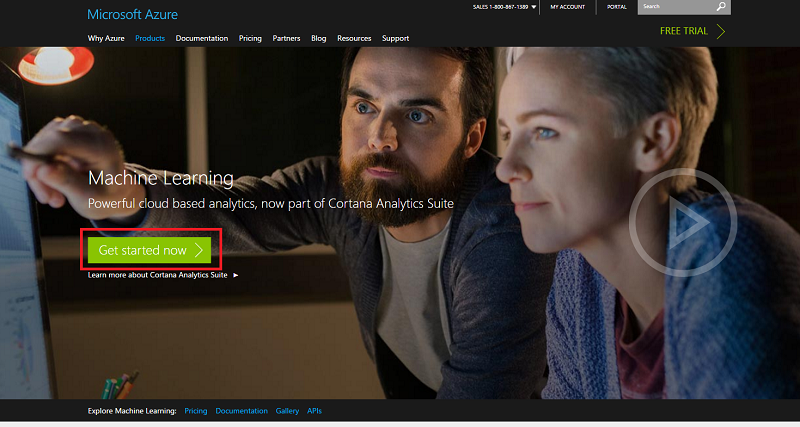
[Step 1. Sign in to Azure Machine Learning](#step1)  
[Step 2. Enter the Competition](#step2)   
[Step 3. Run the Sample Training Experiment](#step3)   
[Step 4. Build and Run a Predictive Experiment, Publish and Submit Web Service API for Evaluation](#step4)   
[Step 5. View Your Ranking on Public Leaderboard](#step5)

For more details of how to upload the shared real training data to the AzureML workspace, how we built the sample training experiment, and how you can build your new solution on AzureML for this competition, please refer to the sections in the Appendix:

[How the Sample Training Experiment Is Built: a Deep Dive](#deepdive)  
[How to Create a New Experiment for the Competition](#new)

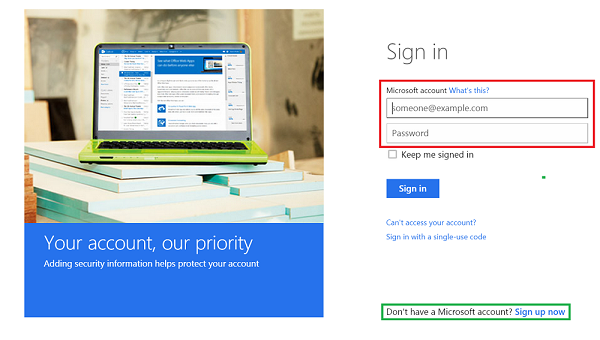
**Step 1. Sign in to Azure Machine Learning**

**Step 1.1. Open** [**Azure Machine Learning**](https://azure.microsoft.com/en-us/services/machine-learning) **web page using any browser. Then click "Get started now."**

****

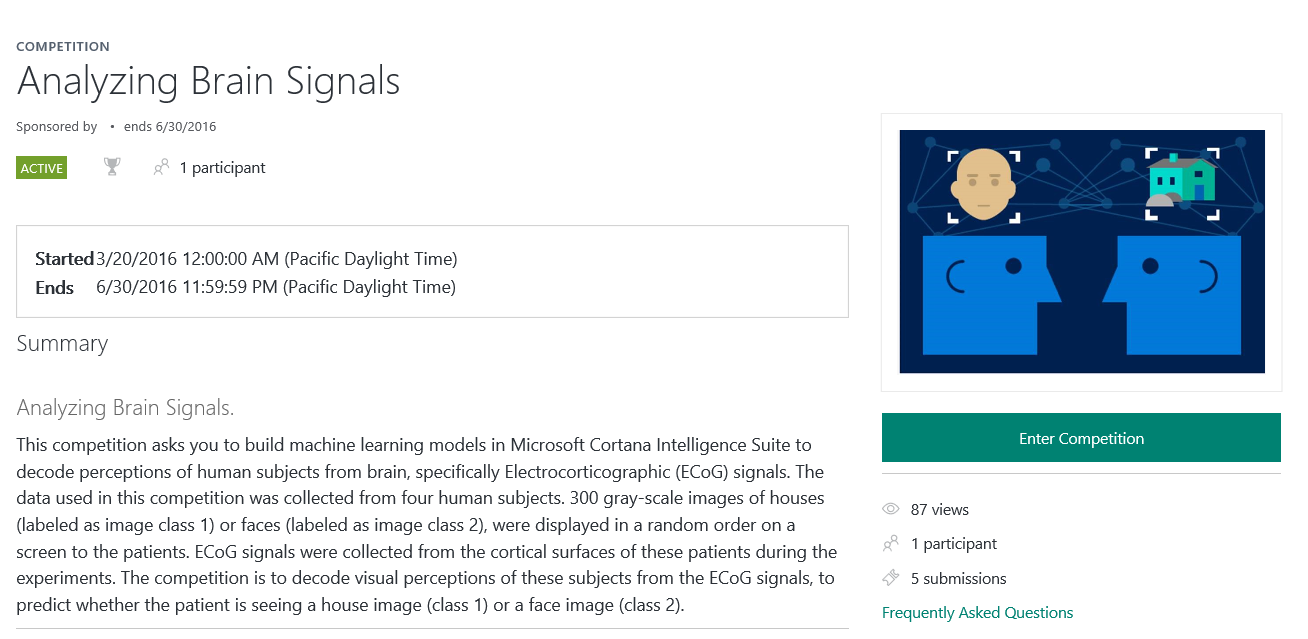
**Step 1.2. Sign in to AzureML Studio.** You will be directed to the Microsoft Sign in page. If you already have a Microsoft account, can sign in directly here. Otherwise, click on "Sign up now" link (the link in the green box) to sign up for a Microsoft account.

**Note:** Please remember that after you make a successful submission to the competition, the public leaderboard will show the name associated with the Microsoft account you are using to log in. If you prefer to remain anonymous, you’d better change the name associated with this email account before you submit.

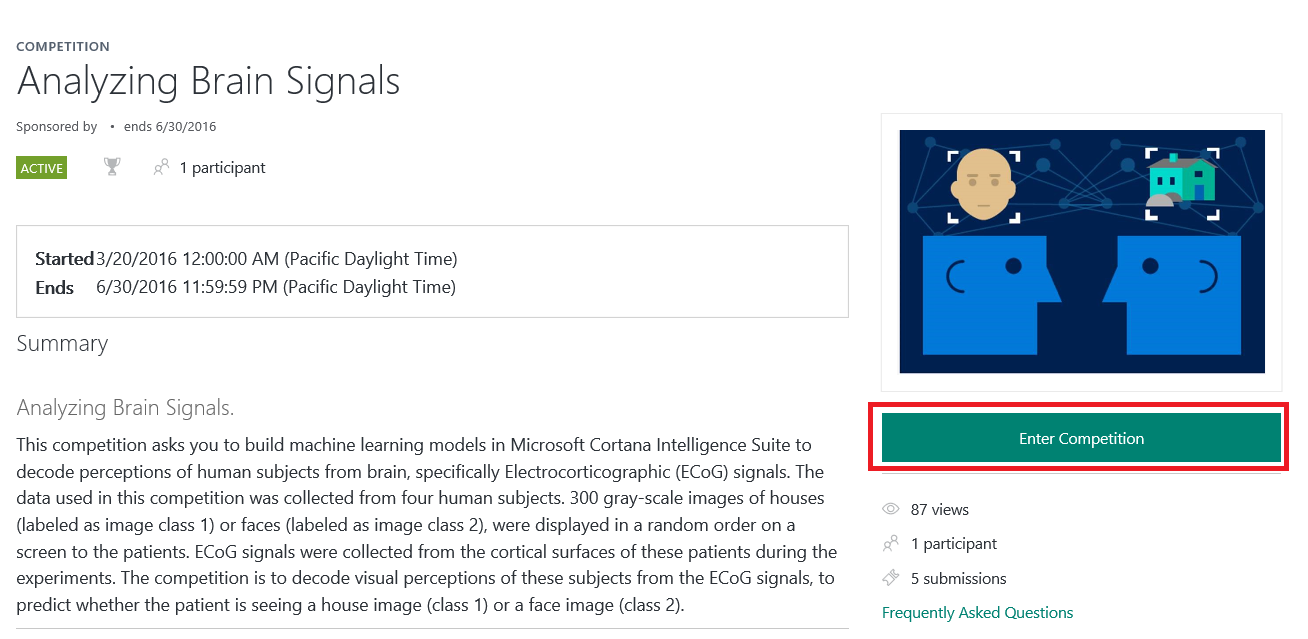
****

**Step 2. Enter the Competition**

* 1. **Visit the information page of the competition at** <https://gallery.cortanaanalytics.com/Competition/d8f8892ee8b54c94939ed2bb2bfa7301>. You will see information about this competition such as summary, description, data files, rules, prizes, leaderboard, etc. These sections provide information that is helpful for you to understand the data, the task, and the evaluation process.



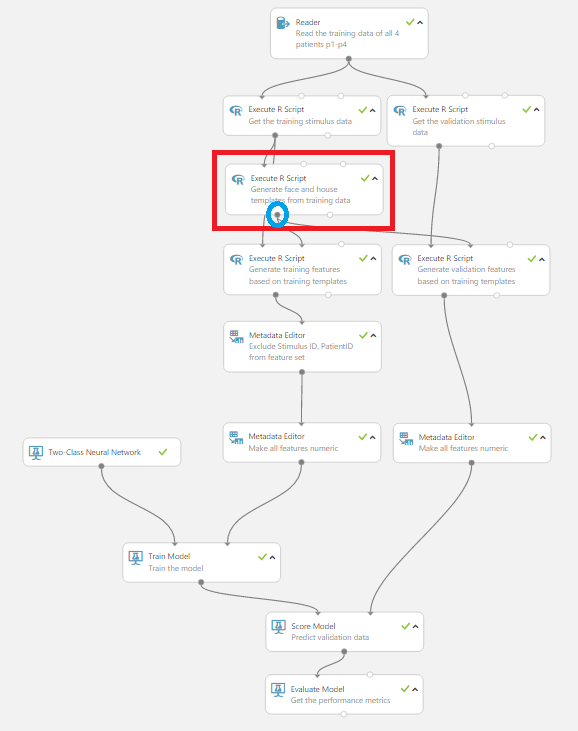
* 1. **Enter the competition.** Click **Enter Competition** to copy the sample training experiment to your AzureML workspace.

****

**Step 3. Run the Sample Training Experiment**

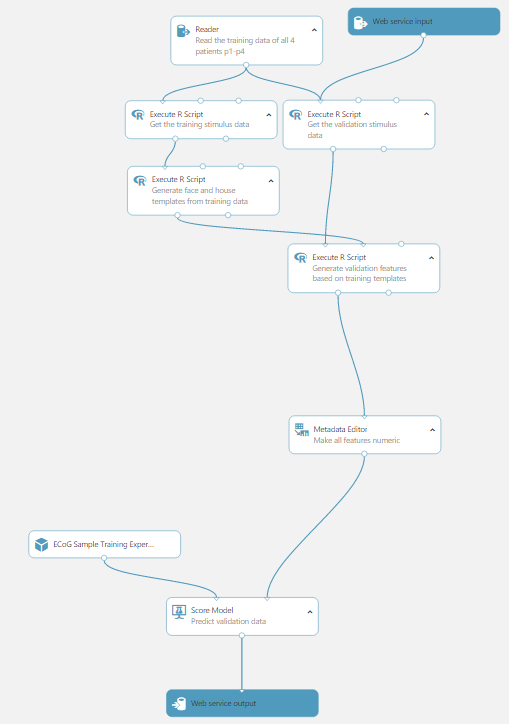
After the sample training experiment **Analyzing Brain Signals** is copied to your Azure ML workspace, click the **Run** button at the page bottom, the sample experiment will start running. It may take around 10 minutes to complete.

**After the sample training experiment completes training, right click the output from the Execute R Script module as highlighted below and save it as a dataset in your workspace**. Let’s name it as “***ECoG Training Templates***”. These are the templates built on the training stimulus presentation cycles. You will need these templates to calculate the similarities between the ECoG signals of the testing stimulus presentation cycles and these templates. These similarities are the features used to train the machine learning model in the sample training experiment.



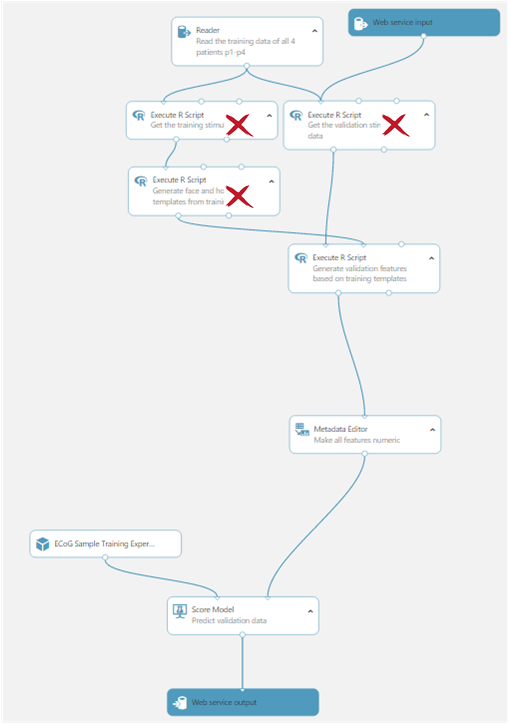
**Step 4. Build and Run the Predictive Experiment, Publish, and Submit Web Service API for Evaluation**

* 1. **Create predictive experiment automatically**. After the sample training experiment completes successfully, at the bottom of the page click **SET UP WEB SERVICE**, then click **Predictive Web Service**. As shown below, the AzureML studio automatically generates a predictive experiment using the model trained in the sample training experiment to make predictions.

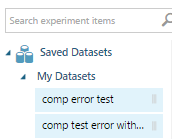


**4.2. Modify the predictive experiment**. You need to modify the automatically generated predictive experiment by following the steps below:

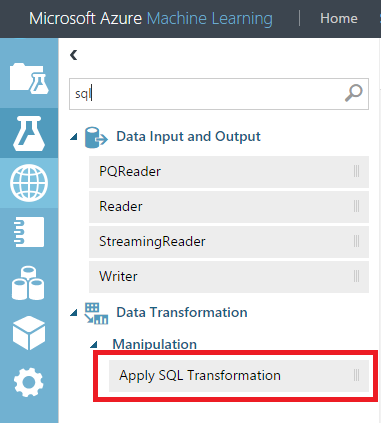
* + 1. Delete the modules that are crossed out in red in the following picture.

****

* + 1. Add the house and face templates that are saved as a dataset in your workspace to the predictive experiment. Go to Saved Datasets > My Datasets left to your experiment, you should be able to find the saved dataset there. “*ECoG Training Templates*” should be the name of the saved dataset in this example.



Add an Apply SQL Transformation module to your predictive experiment. Replace the **SQL Query Script** in the module **Apply SQL Transformation.** You can find modules on the left side of the studio. To help get these modules faster, you can input the module name in the search field **Search experiment items**. Then, from the search results, drag and drop the module you need into your experiment.

****

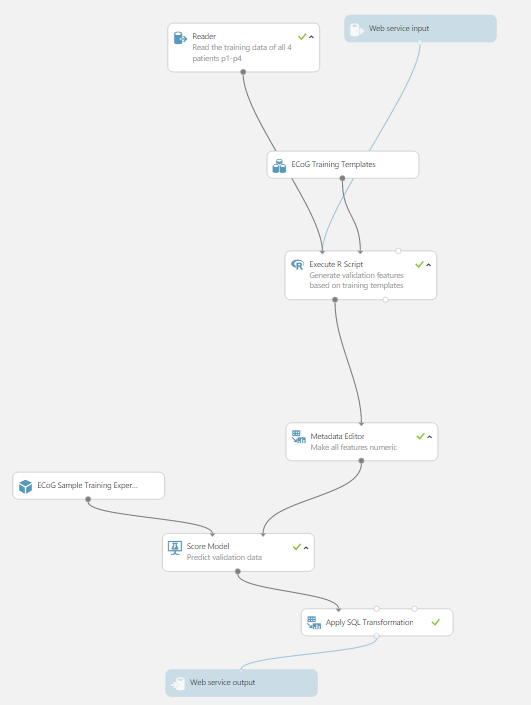
* + 1. Use the following scripts to replace the default query in the Apply SQL Transformation module. **DO NOT change** single quotes or double quotes in this query, even if you see some dashed underline in the SQL Query Script window of this module.

*select PatientID, Stimulus\_ID,*

*cast("Scored Labels" as INTEGER) as "Scored Labels"*

*from t1;*

* + 1. Connect the modules and data as follows:



**Checklist before you proceed:**

□ The reader module is connected to the **FIRST** input portal of the Execute R Script module. So is the **Web service input**.

□ The signal template data (here, ECoG Training Templates in the sample predictive experiment) is connected to the **SECOND** input portal of the Execute R Script module.

□ The output portal of **Score Model** module is connected to the **FIRST** input portal of **Apply SQL Transformation** module.

□ The **Web Service Output** is connected to the output portal of **Apply SQL Transformation**.

□ The Web Service API input and output match the requirements below.

## **Requirements on the web service API input and output schema**

The input schema of the web service API should be the same as the schema of the training data. The input data schema can be found in the data description section.

The output schema of the web service API **MUST BE** as follows:

|  |  |  |
| --- | --- | --- |
| **Column Index** | **Column Name** | **Data Type** |
| 1 | PatientID | String (p1, p2, p3, or p4) |
| 2 | Stimulus\_ID | Integer |
| 3 | Scored Labels | Integer (binary with values 1 or 2) |

Make sure that the order of the columns in your web service API output schema, column names, and data types are the same as above. Otherwise, you will not be able to submit, or you will not get a reasonable score as you might expect.

**4.3. Deploy web service API.** **RUN** the predictive experiment, and **DEPLOY WEB SERVICE** after it completes.

Click **RUN** button at the bottom of the page. The predictive experiment should complete in less than 3 minutes. Then, click **DEPLOY WEB SERVICE** to generate your web service API.



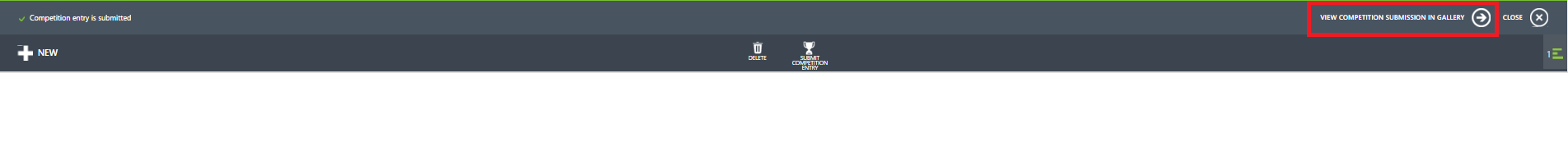
**4.4** **Submit your web service API for evaluation**

After **Deploy Web Service** is done, click **SUBMIT COMPETITION ENTRY** at the page bottom to submit and get your web service API evaluated on the testing data.

You have to agree with the terms during the submission process. You also have the opportunity to name your submission, which might be very helpful to remind you the features and/or models you use in this solution.

**Step 5. View Your Ranking on Public Leaderboard**

After your web service API is successfully taken by the evaluation process, you will see a green check mark on the left bottom corner of the page and evaluation on the public test data is underway. Click **“VIEW COMPETITION SUBMISSION IN GALLERY”** on the right bottom corner of the page, you will be redirected to the competition detail page in gallery where you can see your submission history. It may take a few minutes before your score on the public test data can be returned from the evaluation process. After that, you will be able to see your current ranking on the public leaderboard.



**Appendix**

**How the Sample Training Experiment Is Built: Deep Dive**

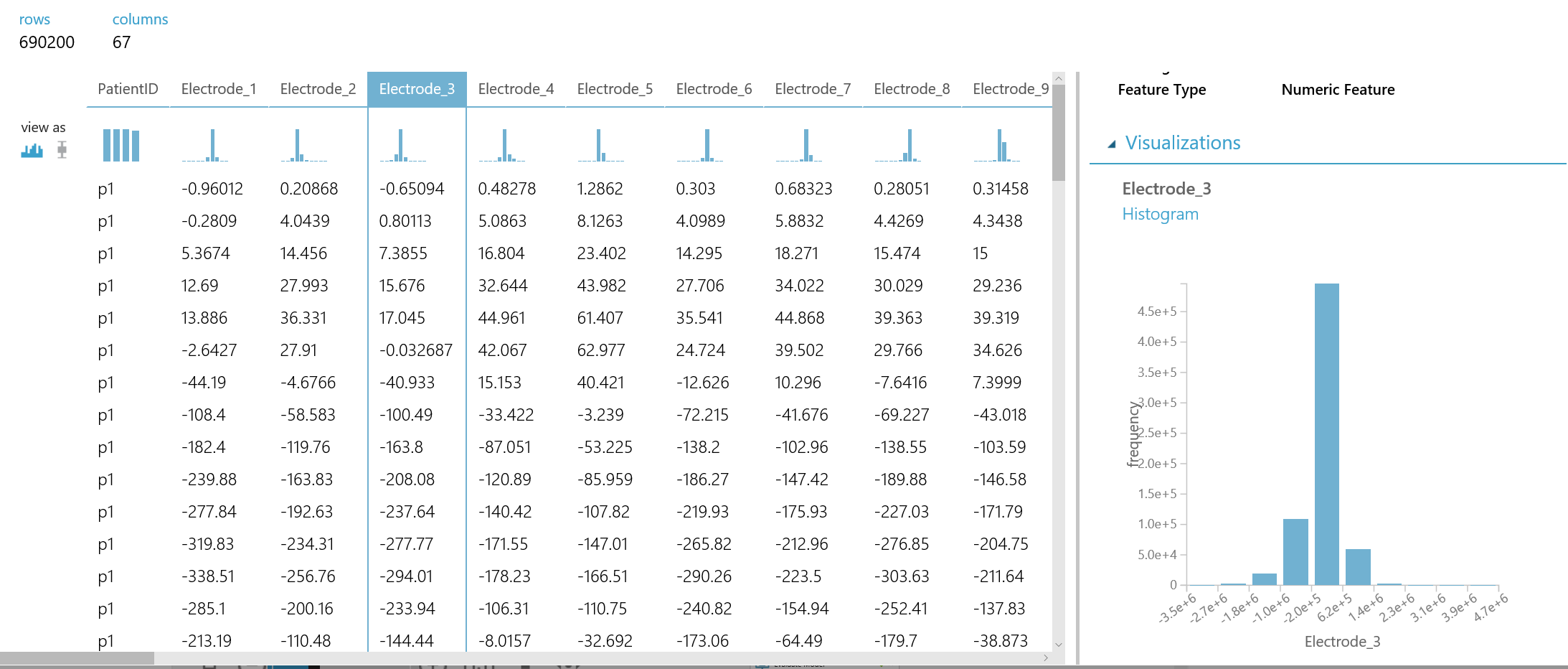
This sample training experiment illustrates some typical steps for creating an end-to-end pipeline for a machine learning task.

It consists of the following steps:

1. [Ingesting and visualizing the raw data](#ingest)
2. [Splitting the data into a training set and a validation set](#splitting)
3. [Compute the event related potential (ERP) templates for each electrode, each stimulus type (house or face), and for each patient](#templates)
4. [Calculate similarities between signals in each stimulus presentation cycle and the signal templates](#similarity)
5. [Exclude PatientID and Stimulus\_ID from Feature Set, and Change the data type of feature columns to be non-categorical](#metadata)
6. [Training a predictive model](#training)
7. [Scoring and evaluating a trained model](#scoring)

**Ingesting and visualizing the raw data**

The organizers have shared the training data in a tabular csv format in a [public URL](http://az754797.vo.msecnd.net/competition/ecog/datasets/ecog_train_with_labels.csv), which allows the reader module in Azure ML to read it into your Azure ML workspace. After the reader module completes running, you can visualize the data by simply right-clicking the output portal of the training dataset, and selecting **Visualize**. After the visualization windows pops up, you can select any column; the statistics and the histogram of the selected variable will then be shown in the right panel.



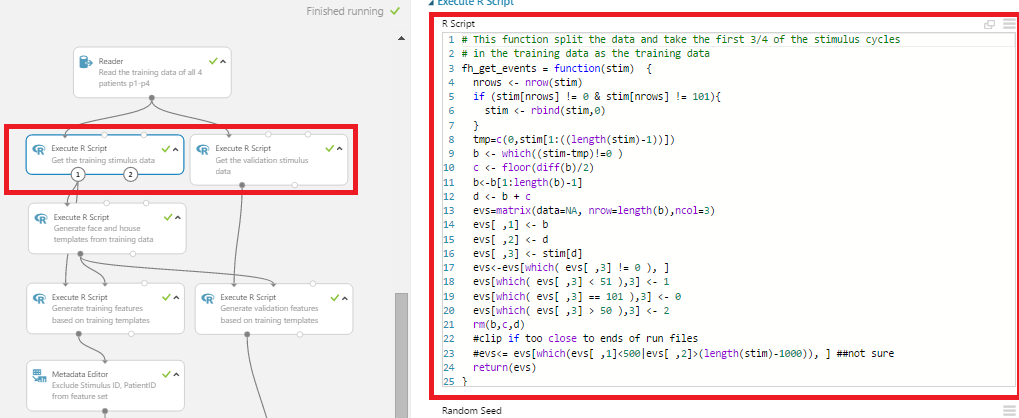
The training data has 67 columns, where column 1 is column PatientID, columns 2-65 are the signals of the 64 channels. The last two columns are Stimulus\_Type and Stimulus\_ID. Stimulus\_Type has values between 0 and 101. For each patient, in training data, Stimulus\_ID is integers ranged between 1 and 200.

Keep in mind that the numbers of implanted electrodes might differ patient to patient. The maximum number of implanted electrodes is 64. Even if a patient has 64 electrodes implanted, some electrodes might have very bad signals, and therefore these electrodes were excluded from the data provided in this competition. Some patient might have valid electrodes 1-42, some might have valid electrodes 1-64. If a patient does not have a valid electrode, the value of this electrode for this patient is replaced by ***-999999***.

Additionally, the same electrode name in different patients does not have the same physical location on the cortex. Even if the same electrode in different patients has similar location, because of the anatomical difference of the patients’ brain, it might be recording the activities of different brain functional regions.

**Splitting the data into a training set and a validation set**

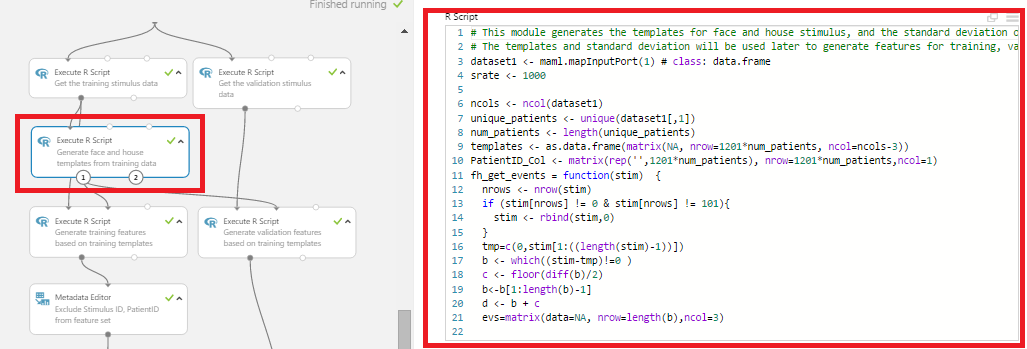
In this sample training experiment, we use two Execute R Script modules to split the data into training (150 stimulus presentation cycles for each patient) and validation data (50 stimulus presentation cycles for each patient). Since we need to reserve the completeness of each stimulus presentation cycle, in the Execute R Script module, we first define a *fh\_get\_events()* function, and call it to get the stimulus onset time of each stimulus presentation cycle. The stimulus onset time plus 399 milliseconds will be the ending time of each stimulus presentation cycle. Then, the records before the ending time of the 150th stimulus presentation cycle are put into the training data, and the records after that are put into the validation data.

****

**Compute the event related potential (ERP) templates for each electrode, each stimulus type (house or face), and for each patient**

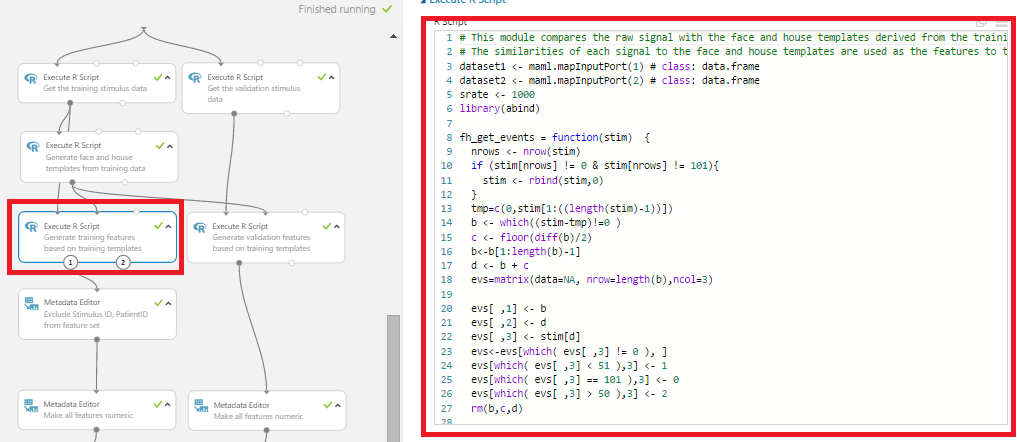
Then, we use an Execute R Script module to compute the ERP templates using the stimulus presentation cycles in the training data. Each template consists of 600 time points (between 200 milliseconds before the onset of the stimulus, and 399 milliseconds after the onset of the stimulus). Since we are generating templates for both house and face stimuli, we concatenate these two templates of each single patient by rows. In addition to that, the standard deviation of each signal in the training data is also needed to normalize the signals in the testing/validation data. So, we also record the training data standard deviation for each signal.

Therefore, the output from this module has 4804 rows. Each 1201 rows belong to a single patient. Among these 1201 rows, the first record is the training data standard deviation, rows 2-601 are the house templates, and the remaining 600 rows are the face templates.



**Calculate similarities between signals in each stimulus presentation cycle and the signal templates**

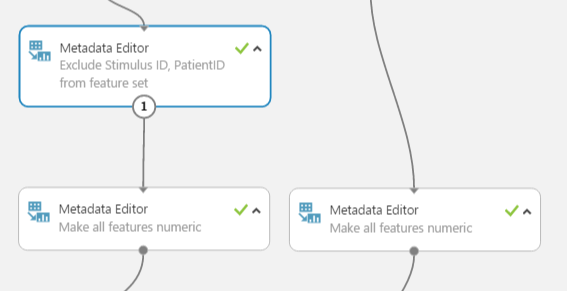
Then, the signal templates are parsed to two Execute R Script modules, where the templates are used to calculate the similarities between the signals in each stimulus presentation cycle and the templates. The output from this module will have the same number of rows as the number of stimulus presentation cycles in the training data or in the validation data. These similarities are the features used to train a machine learning model to predict the stimulus type for each stimulus presentation cycle.



**Exclude PatientID and Stimulus\_ID from Feature Set, and Change the data type of feature columns to be non-categorical**

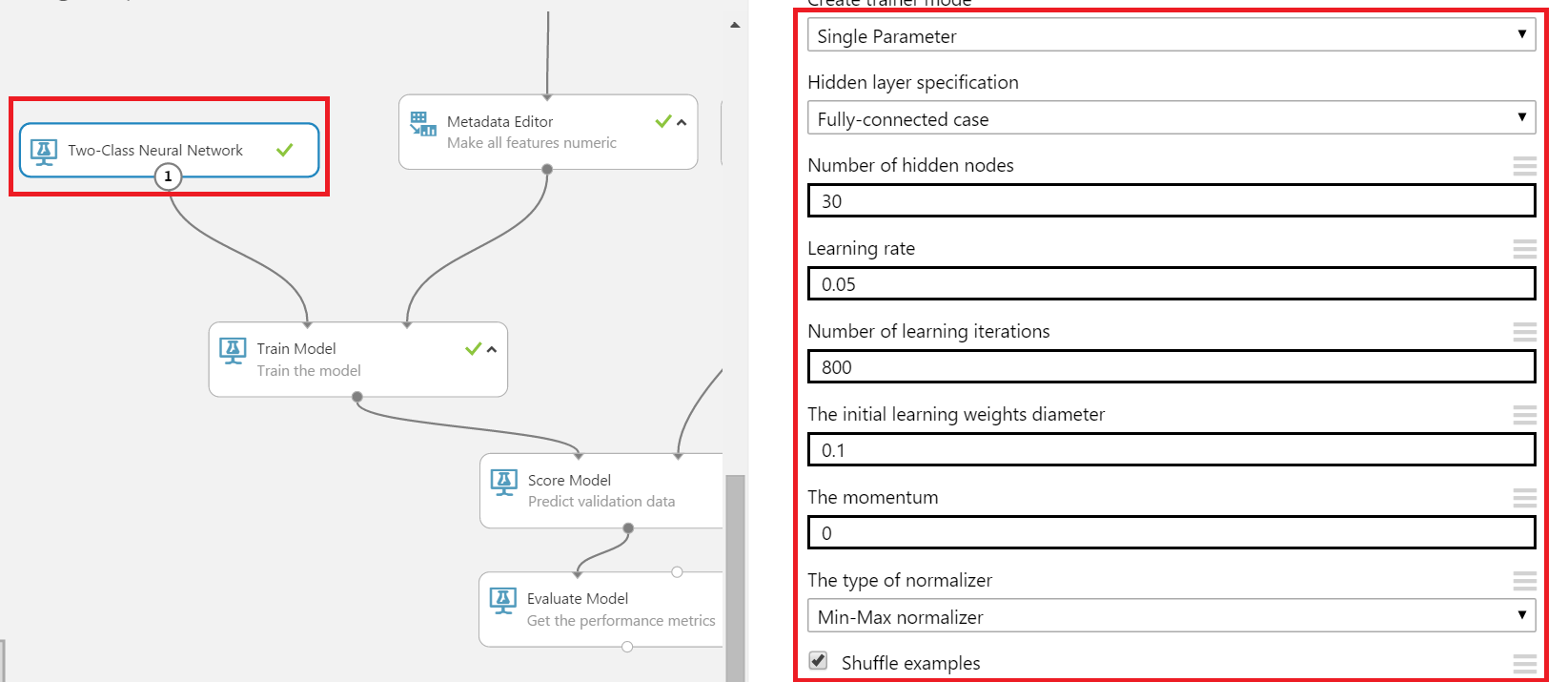
We use a Metadata Editor module to exclude columns PatientID and Stimulus\_ID from the feature set. So the model training process will not include them as explanatory variables.

All features in this sample training experiment are numerical. We use Metadata Editor modules to convert all these features to non-categorical, so that the machine learning model can treat them correctly.



**Training a predictive model**

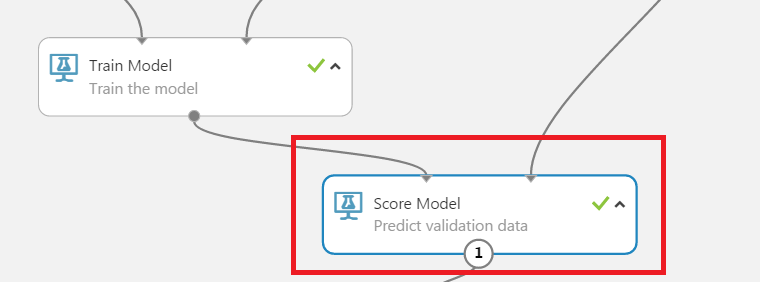
Now, train the model using the training data. We first add **Two Class Neural Network** module from the **Machine Learning->Initialize Model->Classification** menu to the experiment.



The **Train Model** module found in the **Machine Learning->Train** menu carries out the actual training of the neural network model. This module takes two inputs: the left input portal takes the model specification, and the right input portal takes the training data. The Train Model module must specify the label column. Here, indicate that the **Stimulus\_Type** column is the target column through the column selector dialog in the **Properties** box.

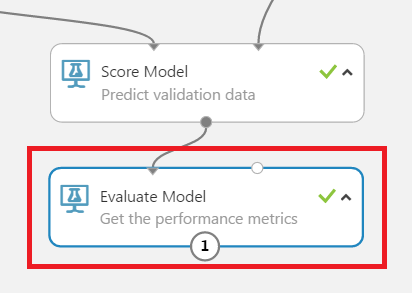
**Scoring and evaluating a trained model**

After the model is trained, we use it to predict the validation data sets. The **Score Model** module in the **Machine Learning->Score** menu accomplishes this task. It takes a trained model in its left input port and the data set to be predicted in its right input port.

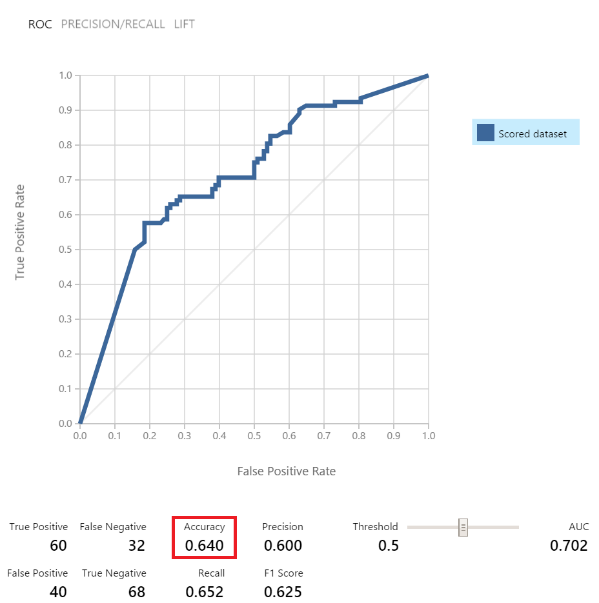


In the scenario displayed above, the Score Model module uses the trained neural network model as its first input, and predicts the validation data from the second input portal.

The scored data set can now be evaluated using the **Evaluate Model** module in the **Machine Learning->Evaluate** menu. This module takes in one or two scored data sets as input.



Evaluation, in this case, displays several metrics related to the accuracy of the model's predictions, as well as a graphical representation of the ROC curve. In this competition, we are using accuracy, which is the percentage of the number of stimulus presentation cycles in the testing data (of all four patients) that is accurately predicted by the model, to rank the participants in leaderboards.



**How to Create a New Experiment for the Competition**

If you want to create a new training experiment, click **Save** and then **Save As** to save the sample training experiment as a new one and make further edits (feature engineering, trying different models, etc.) on it.   
  
**DO NOT** directly click **NEW** to create a new experiment since experiments created this way will not be recognized as experiments for competitions. Later on web service APIs created from such training experiment will not have the button to submit for evaluation.

After the new training experiment is created, you can follow the step above to [Step 4. **Run the Scoring Experiment, Publish and Submit Web Service API for Submission**.](#step4)