

Solve the KDD Cup 2015 problem from a Sample Experiment in Azure Machine Learning

Each year KDD Cup brings the data science community together to fight for a coveted spot on their leader-board via a data competition. This year's challenge, [KDD Cup 2015](#) requires participants to predict the likelihood of a student dropping out from a MOOC platform, XuetangX. This is a typical customer churn analysis problem. Gaining a better understanding of customer churn is a top priority for not just MOOC platforms but almost all businesses.

Azure machine learning is a cloud based tool that enables data scientists and big data professionals to build and operationalize such predictive analytics solutions with ease. We have put together this tutorial on how to build a predictive customer churn analytics model by using the KDD CUP MOOC problem. This tutorial could be a great starting point for those of you who are looking to participate in the KDD Cup competition. These sample experiments that we provide below can be easily accessed from the [Azure ML Gallery](#) . 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.

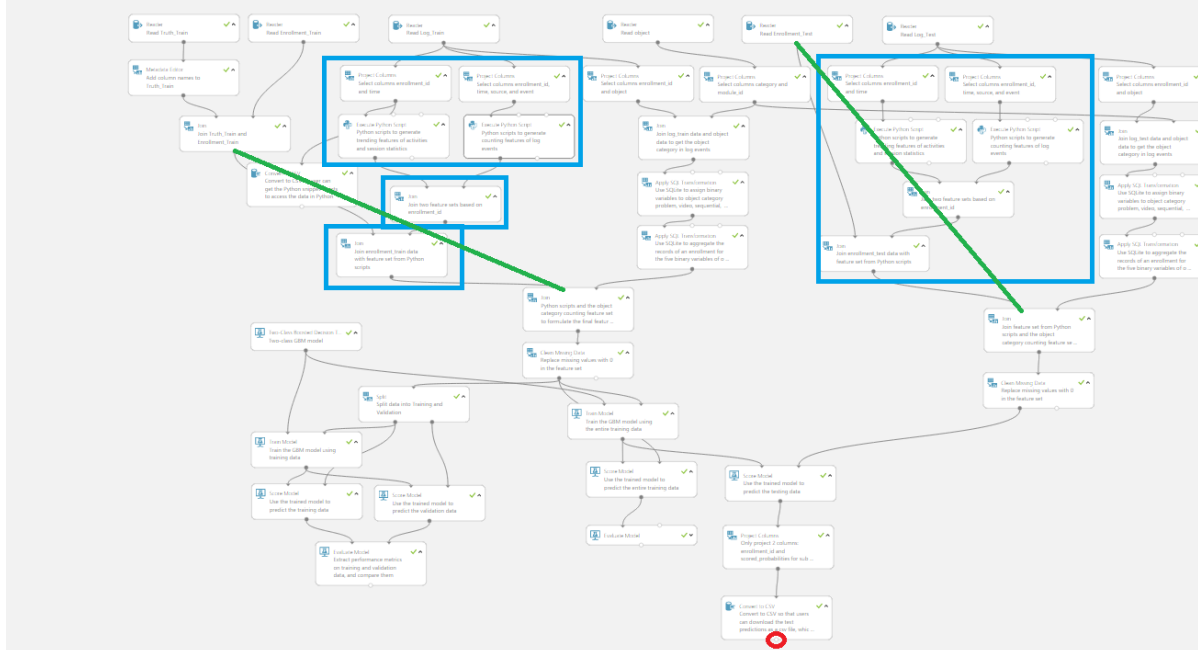
The tutorial provides two sample experiments: a) KDD Cup 2015: Customer Churn Prediction (Low) and b) KDD Cup 2015: Customer Churn Prediction (High) . The Low-version experiment is just a subset of the High-version experiment. It provides a quick overview of Azure ML. They differ in the following ways:

- I. The Low-version experiment is lower on complexity than the High-version experiment. It creates a smaller set of features, and trains a smaller gradient boosted decision tree model (maximum number of leaves per tree=50 and number of trees=600 in Low-version vs maximum number of leaves per tree=100 and number of trees=1200 in High-version), than the High-version experiment.
- II. The Low-version experiment has lower performance (AUC=0.853 on public leaderboard) than High-version experiment (AUC=0.873 on public leaderboard).
- III. The Low-version experiment, because it is simpler, runs faster than the High-version experiment. The Low-version experiment can be completed in 18 minutes whereas the High-version experiment takes 2 hours.

Below are the graphs of these two sample experiments. The highlighted modules are the extra additions in the high accuracy sample experiment and help in improving the accuracy of the model. Since the Low-version experiment is just a subset of the High-version one, in the remainder of this blog, we only describe the High-version experiment in details.

1. High-version (AUC=0.873)

Finished running ✓



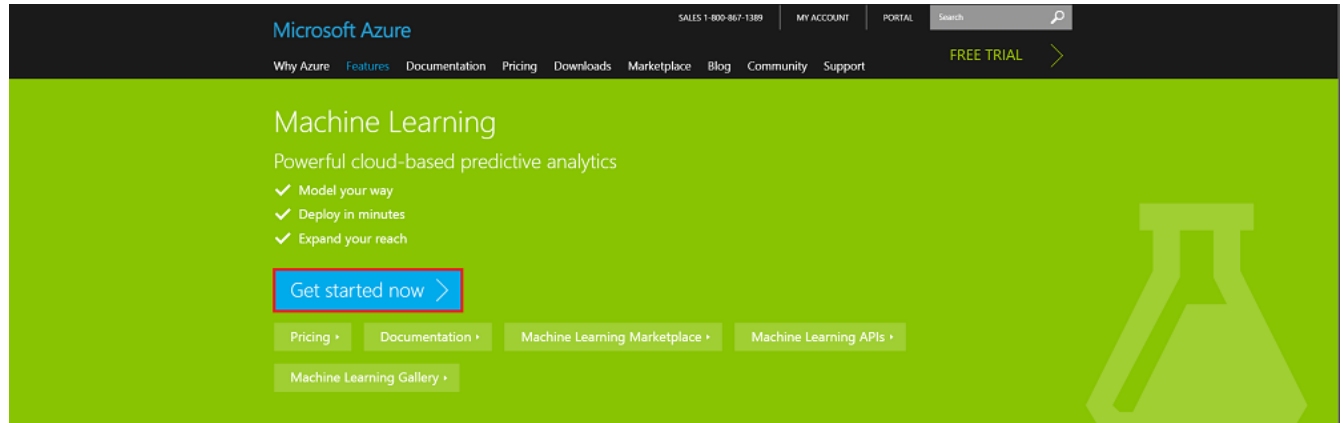
Finished running ✓

- Sign in to Azure Machine Learning for Free Trial and Copy Sample Experiment to Your Workspace
- Run Experiment and Download Test Predictions for Submission
- Deep Dive into the Sample Experiment (High)

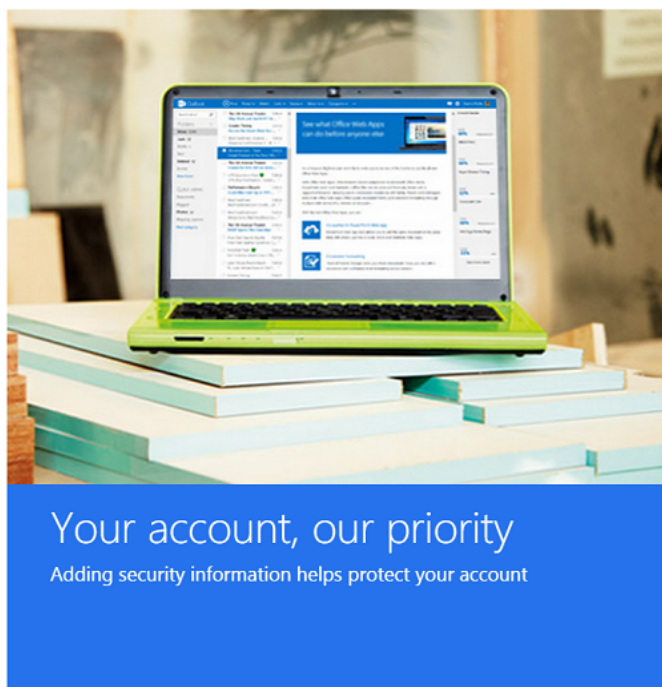
- Debug Python scripts using Azure Machine Learning Python Client library
- Publish Your Experiment to Azure Machine Learning Gallery
- Other Modules That Might Be Useful

Sign in to Azure Machine Learning for Free Trial and Copy Sample Experiment to Your Workspace

1. Open Azure Machine Learning web page using any browser. Then click "Get started now".



2. You will be directed to the Microsoft Sign in page. If you already have a Microsoft account, such as @outlook.com, @live.com, or @hotmail.com, you can sign in directly here. Otherwise, click on "Sign up now" link (the link in the green box) to sign up a Microsoft account.



Sign in

Microsoft account [What's this?](#)

☐ Keep me signed in

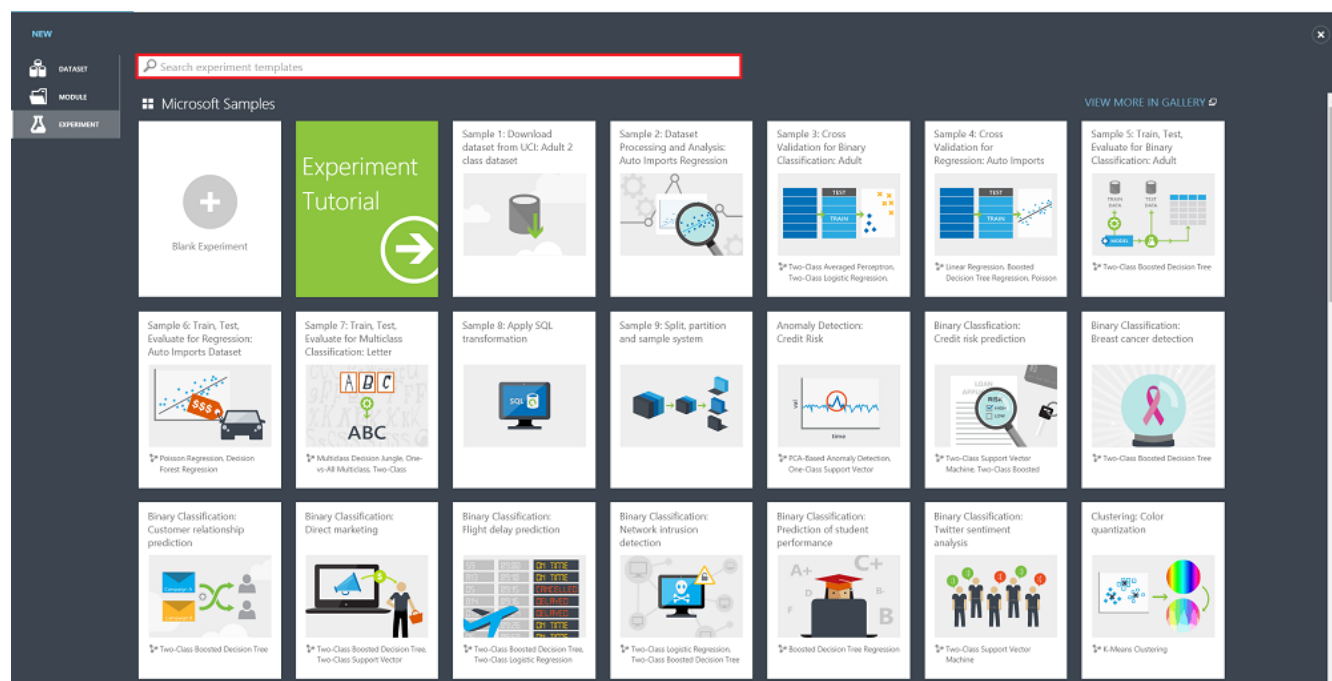
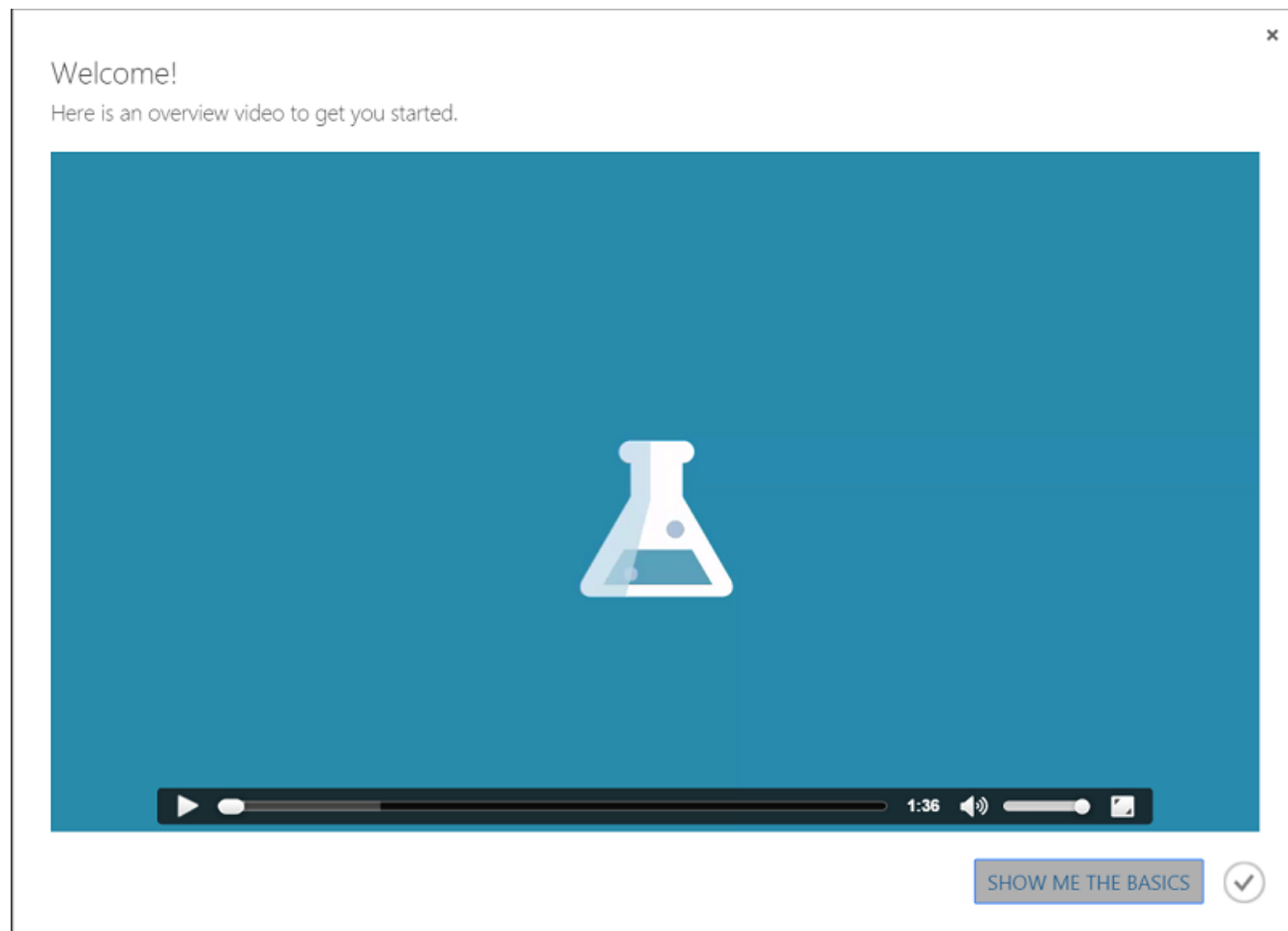
[Sign in](#)

[Can't access your account?](#)
[Sign in with a single-use code](#)

[Don't have a Microsoft account? Sign up now](#)

3. After you sign in (or sign up for a new Microsoft account), you will be greeted by a Welcome video.

Close this video window, and the sample experiments published by Microsoft will be shown in the page.



4. In the search text box, input "KDD Cup 2015" and search. In the returned sample experiments, move the cursor to "KDD Cup 2015 Sample Experiment: Customer Churn Prediction (High)" and

click the green button "Open in Studio". The sample experiment will be copied to your AML workspace.

[AZURE.NOTE] Please be noted that the free-tier trial of Azure ML has the limitation that an experiment has 1 hour maximum running duration. Time-out error will be thrown out if an experiment does not complete running in 1 hour. Information can be found in [Azure Machine Learning Pricing information](#).

In order to run the high-version experiment, you can subscribe to the 1-month free trial, which will give you 1-month free trial and 200\$ free credit. The free trial expires when 1 month is reached, or the balance goes to zero, whichever happens first. Please follow the instructions in [Get Started with Azure Machine Learning](#) on how to subscribe to Azure and then to Azure Machine Learning. When you subscribe to Azure ML, you will be asked to input your credit card information. You **will not be charged** for free trial. The purpose of the credit card information is just to verify that the subscriber is a person, but not a robot.

Run Experiment and Download Test Predictions for Submission

After the sample experiment is copied to your AML workspace, click the Run button at the page bottom, the sample experiment will start running. It may take around 2 hours to complete. After the experiment completes successfully, right click the output portal of the **Convert to CSV** module (red-circled in the following figure), and select **Download**, the predictions on test data will be downloaded as a .csv file to your local machine.

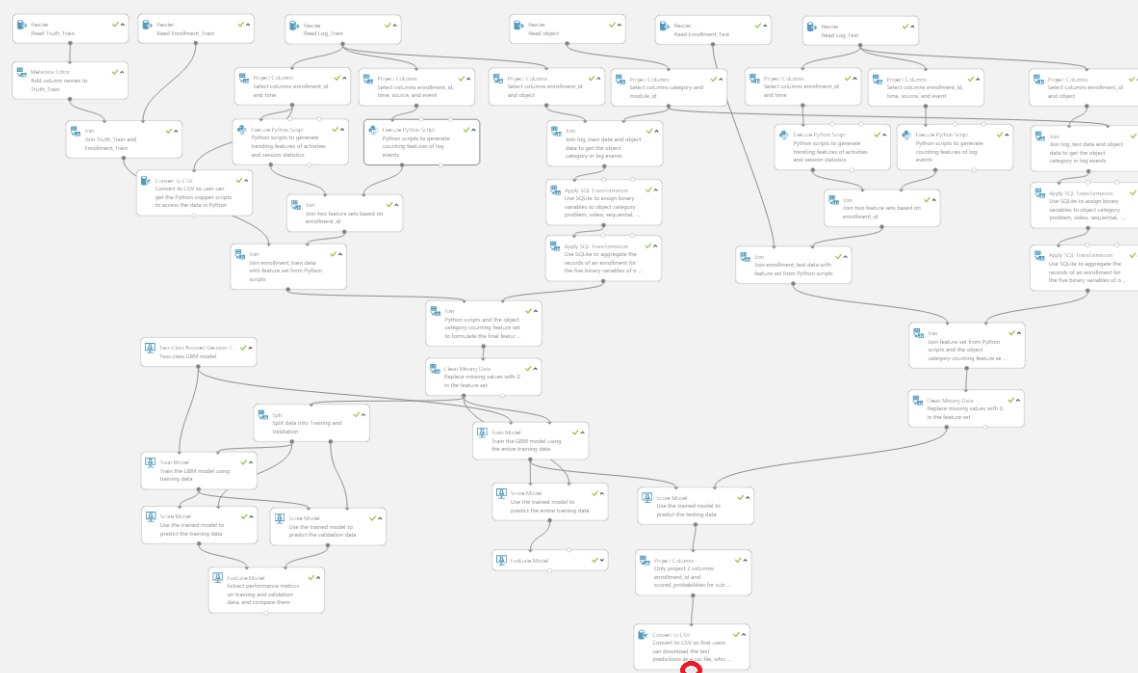
Before you submit the downloaded test predictions to KDD Cup 2015 competition website for evaluation, you need to edit the csv file by **removing the header line** since the competition website does not allow header line in the submission files. The test predictions from this sample experiment should give you AUC 0.8733 on the public leaderboard.

Deep Dive into the Sample Experiment (High)

The sample experiment has steps popularly seen in a typical end to end pipeline of a machine learning task. The end to end pipeline is shown in the following graph.

KDDCup2015 Sample Experiment: Customer Churn Prediction (High)

Finished running ✓



It consists of the following steps:

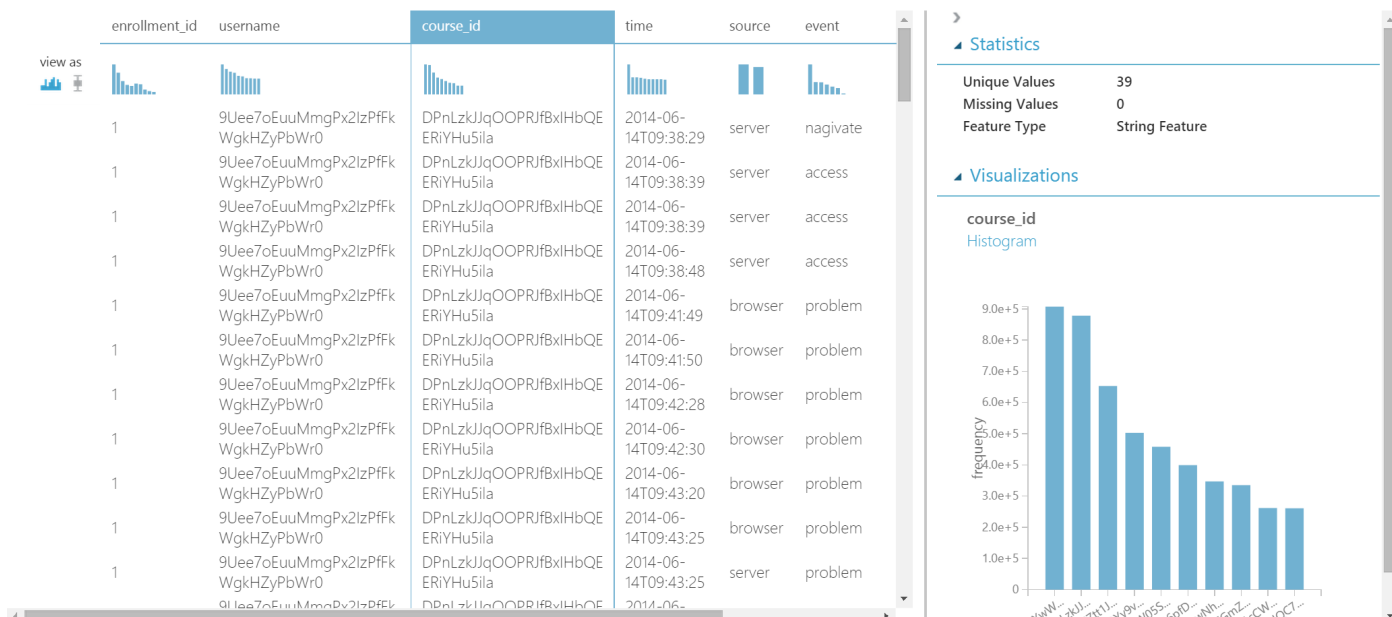
1. Ingesting and visualizing the raw data;
2. Feature engineering on both training and testing data;
3. Data cleaning by handling missing values;
4. Splitting training data into train and validation sets;
5. Training a machine learning model on train data, and validate it on validation data;
6. Fine tuning the parameters of the machine learning model to optimize the performance on validation data;
7. With the optimized parameters, training a machine learning model on the entire training data;
8. Using the model trained on the entire training data to predict the testing data;

Ingesting and visualizing the raw data

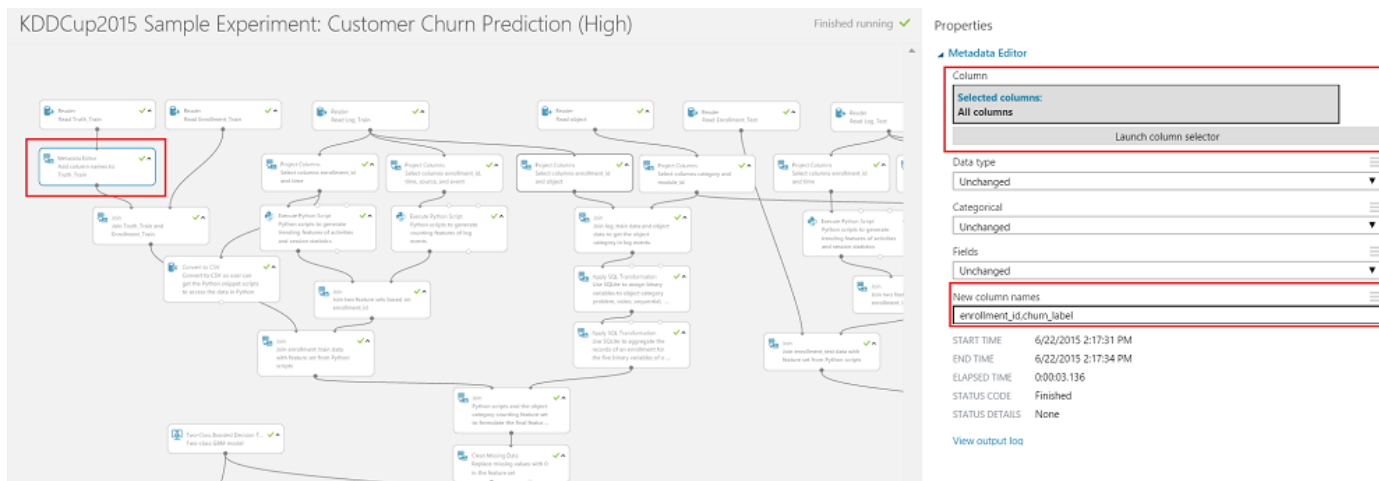
Since the organizers have already provided the data in a tabular .csv format, the raw data can be read directly into AML workspace. All six raw data files, truth_train.csv, enrollment_train.csv, log_train.csv, object.csv, enrollment_test.csv, and log_test.csv are read into AML workspace by six Reader modules, respectively. Upon the approval of the competition organizers, Microsoft shared these six unzipped .csv files in a public Azure blob storage container. Then, the reader modules can read these .csv files by specifying the web URL.

After the data is read into AML workspace, you can visualize the data by simply right clicking the output portal of the reader module, and selecting "Visualize". After the visualization windows pops up, you can select any column, and the statistics and the histogram of the selected variable will be shown in the right panel.

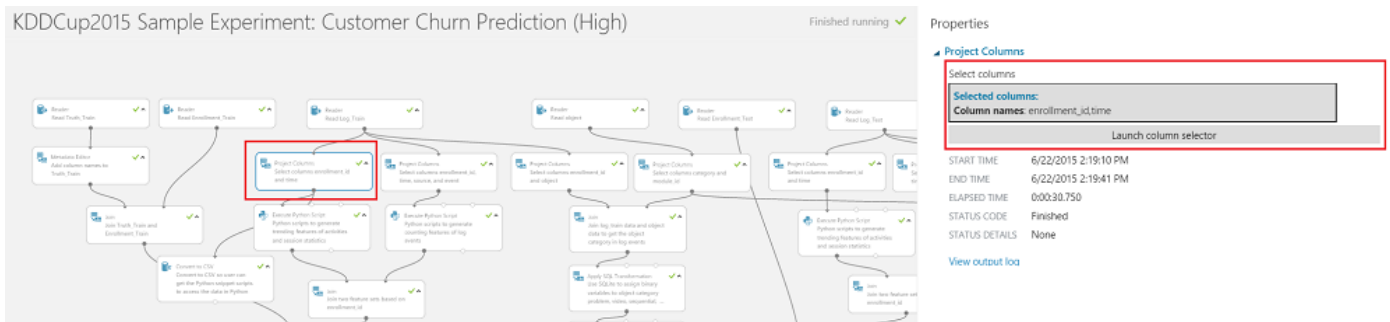
KDDCup2015 Sample Experiment: Customer Churn P... > Reader > Results dataset

rows
8157277columns
7

After truth_train.csv is read into AML workspace, we use a module "Metadata Editor" to add column names to this dataset since the raw data does not have a headerline. Click the "Metadata Editor", you can see the property panel of this module on the right. We added column names "enrollment_id" and "churnlabel" to the dataset.



Before we pass the data to modules for feature engineering, we use modules "Project Columns" to only pass the needed columns to the corresponding feature engineering module. For instance, after "log_train.csv" is read into the AML workspace, the "Project Columns" module after the reader module only selects "enrollment" and "time" columns since the feature engineering module that follows only extracts features based on the timestamps of event logs for each enrollment.

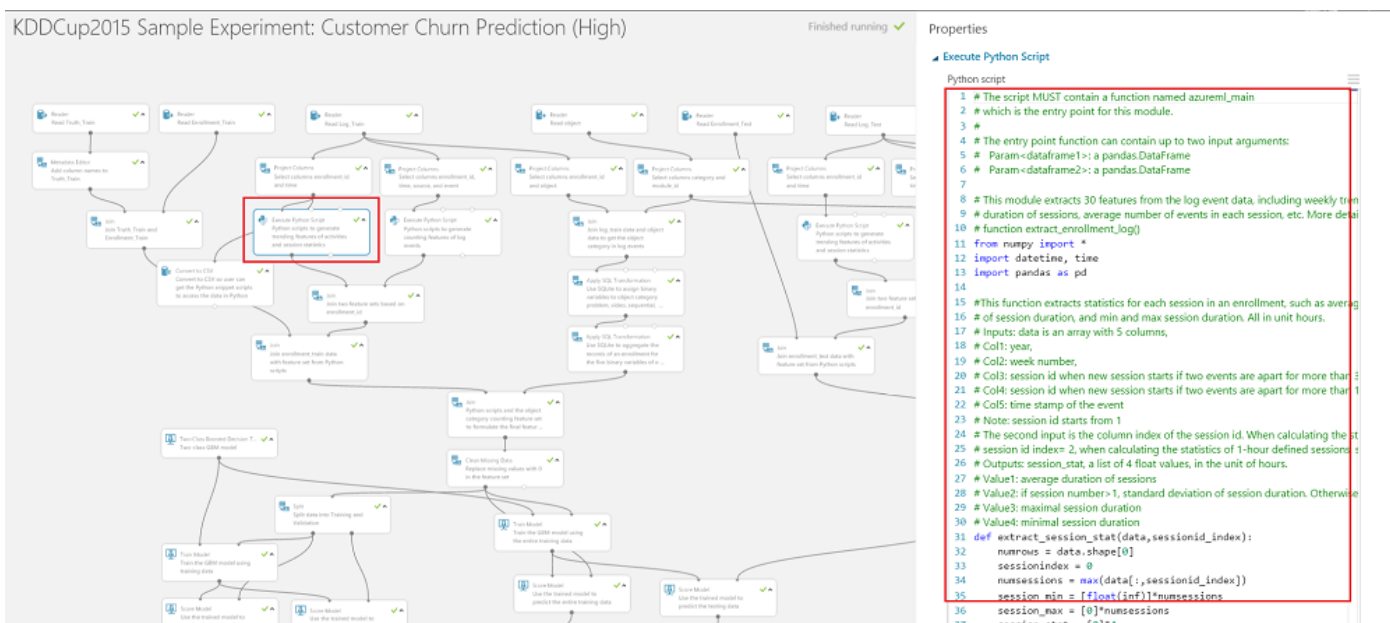


Feature engineering on both training and testing data

In most of the cases, feature engineering plays a critical role in machine learning. Since the purpose of this sample experiment is to show the end to end process of tackling the KDD Cup 2015 tasks, we do not provide very sophisticated features. You might have to develop more advanced features in order to improve the performance.

In this sample experiment, we use two Execute Python Script (EPS) modules to generate two sets of features based on the event logs of training data. Then, the same two EPS modules are copied and pasted on the experiment to generate the same two sets of features on testing data.

Click the first EPS module, the Python scripts that we develop to generate the features for each of the enrollment_id will show in the Python Script editor box. The main (entry) function is named **azureml_main**, which takes two input parameters **dataframe1** and **dataframe2**, representing the datasets from the first and second input portals of this module, respectively. The **azureml_main** function outputs a dataframe **dataframe1**. You cannot change the main function name, input and output parameter names. You can customize your own feature engineering/data processing scripts within the **azureml_main** function. You can also define your own functions to be called by **azureml_main** function.



This EPS module generates 30 features for each enrollment_id (This feature set is skipped in Low-

version experiment.). The output of this module has 31 columns since the first column is **enrollment_id**. The EPS module has detailed comments which list the definitions of all 30 features. To name a few, the features include:

1. Trending slope of the weekly log event counts, starting from the week of the first event, to the week of the last event.
2. Numbers of events in the last (first) week of the enrollment.
3. Number of events in the second last week of the enrollment.
4. Average (standard deviation, minimal, maximal) number of weekly log events in the enrollment period.
5. Numbers of sessions where a session starts if two events are apart for more than 3(1) hours. We name these sessions as 3-hour and 1-hour sessions, respectively.
6. Average numbers of events in 3-hour (1-hour) sessions.

The second EPS module generates 40 features, based on the event log timing and event types. Here is the complete list of features from this EPS module. (**This feature set is skipped in Low-version experiment.**)

1. Features 1-7: numbers of events in Monday to Sunday, respectively.
2. Features 8-31: numbers of events in hours 0-23, respectively.
3. Features 32-38: counts of 7 types of events, respectively.
4. Features 39-40: counts of events of 2 source types, respectively.

We also use two **Apply SQL Transformation** module to generate features that count the number of events that five different categories of objects are accessed in the log data. These five categories are "Problem", "Video", "Sequential", "Chapter", and "Combinedopened". We do not consider

The data needed by the **Apply SQL Transformation** module requires data from both log data and object data. So, we have a **Join** module to join these two sets of data by keyword **object** in log data and **module_id** in object data. **Join** module enables users to consolidate data easily.

The SQL syntax in **Apply SQL Transformation** is SQLite, very similar to SQL Server or MySQL syntax, but slightly different. You can find the detailed description of the SQLite syntax from the [SQLite website](#).

After all three sets of features are ready, we consolidate them together by using **Join** modules to create two feature sets, one for training, and the other one for testing. Training data needs one extra joining with the truth_train.csv data so that every row in training data has a churn label (1 or 0).

Data cleaning by handling missing values

In this example, we use **Clean Missing Data** module to handle missing values in the feature set. Click the **Clean Missing Data** module, and in the Properties panel on the right, you can see that we replace missing values with 0. You can choose other cleaning mode such as replacing missing values with mean, median, etc.

KDDCup2015 Sample Experiment: Customer Churn Prediction (High) Finished running ✓

Properties

Clean Missing Data

Columns to be cleaned
Selected columns:
All columns

Launch column selector

Minimum missing value ratio
0

Maximum missing value ratio
1

Cleaning mode
Custom substitution value

Replacement value
0

Generate missing value indicator column

START TIME: 6/22/2015 4:21:43 PM
END TIME: 6/22/2015 4:21:49 PM
ELAPSED TIME: 00:06.240
STATUS CODE: Finished
STATUS DETAILS: None
[View output log](#)

Splitting training data into train and validation sets

In order to train a model and validate it, we need to split the data into train and validation sets. This is done by **Split** module. Click this module, from its **Property** panel on the right, you can see that this module randomly splits the data into 75% and 25%, where 75% goes to the left output portal, and the remaining goes to the right output portal. In the next steps, we take the 75% data as the train set to train a model, and the remaining 25% as the validation set to validate the model and fine tune the model parameters.

KDDCup2015 Sample Experiment: Customer Churn Prediction (High) Finished running ✓

Properties

Split

Splitting mode
Split Rows

Fraction of rows in the first output dataset
0.75

☒ Randomized split

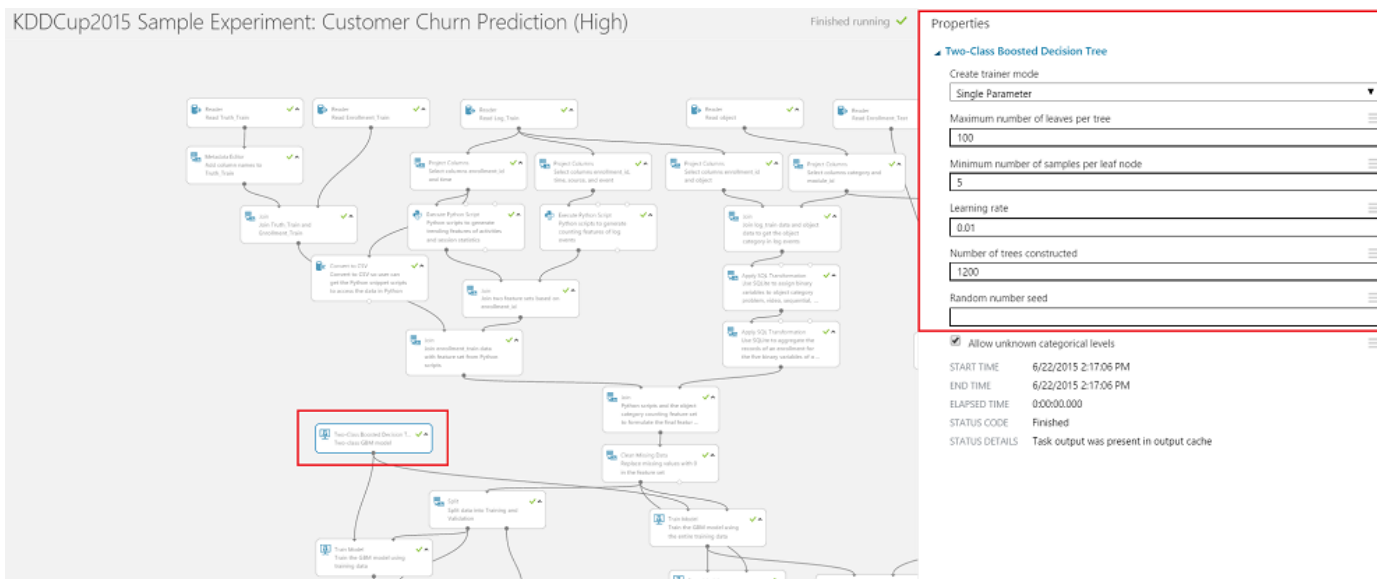
Random seed
0

Stratified split
False

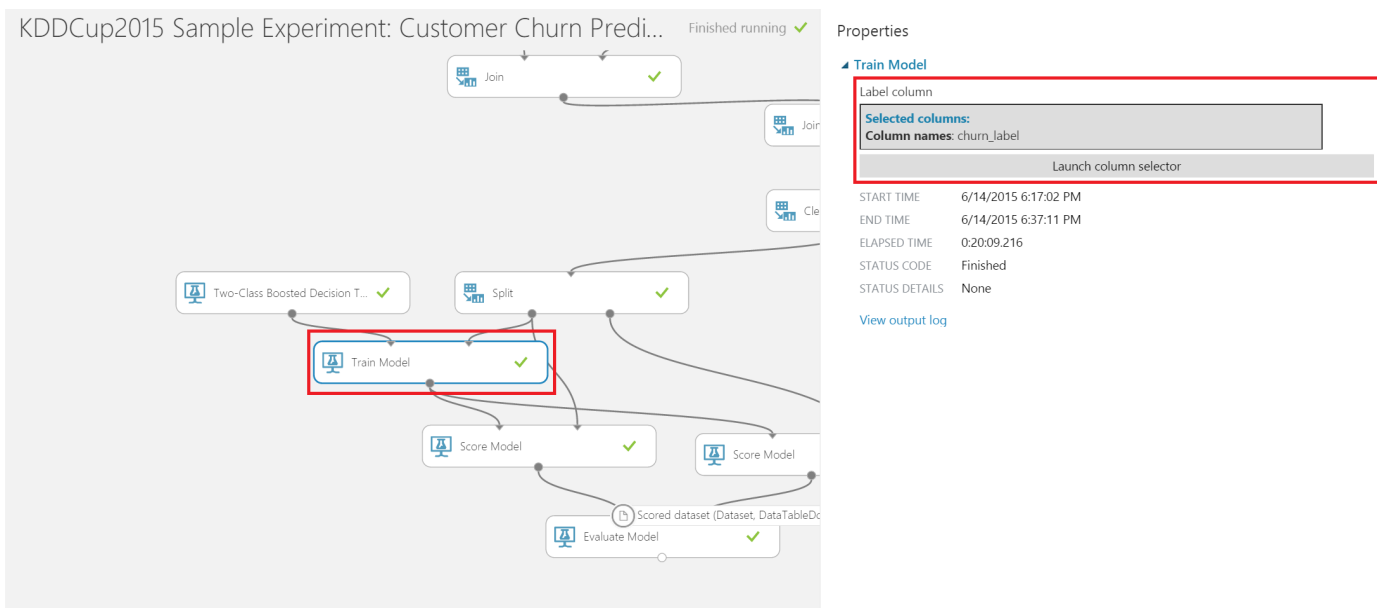
START TIME: 6/22/2015 4:22:11 PM
END TIME: 6/22/2015 4:22:17 PM
ELAPSED TIME: 00:06.086
STATUS CODE: Finished
STATUS DETAILS: None
[View output log](#)

Training a machine learning model on train data, and validate it on validation data

To train a model, we first specify what model we want to train, and the parameters of the model. In this sample experiment, we train a **Two-Class Boosted Decision Tree** model. Click this module, the **Property** panel lists the parameter settings of this model. In this experiment, we are training 1200 trees, with learning rate 0.01, maximum number of leaves per tree 100, and minimum node size 5.



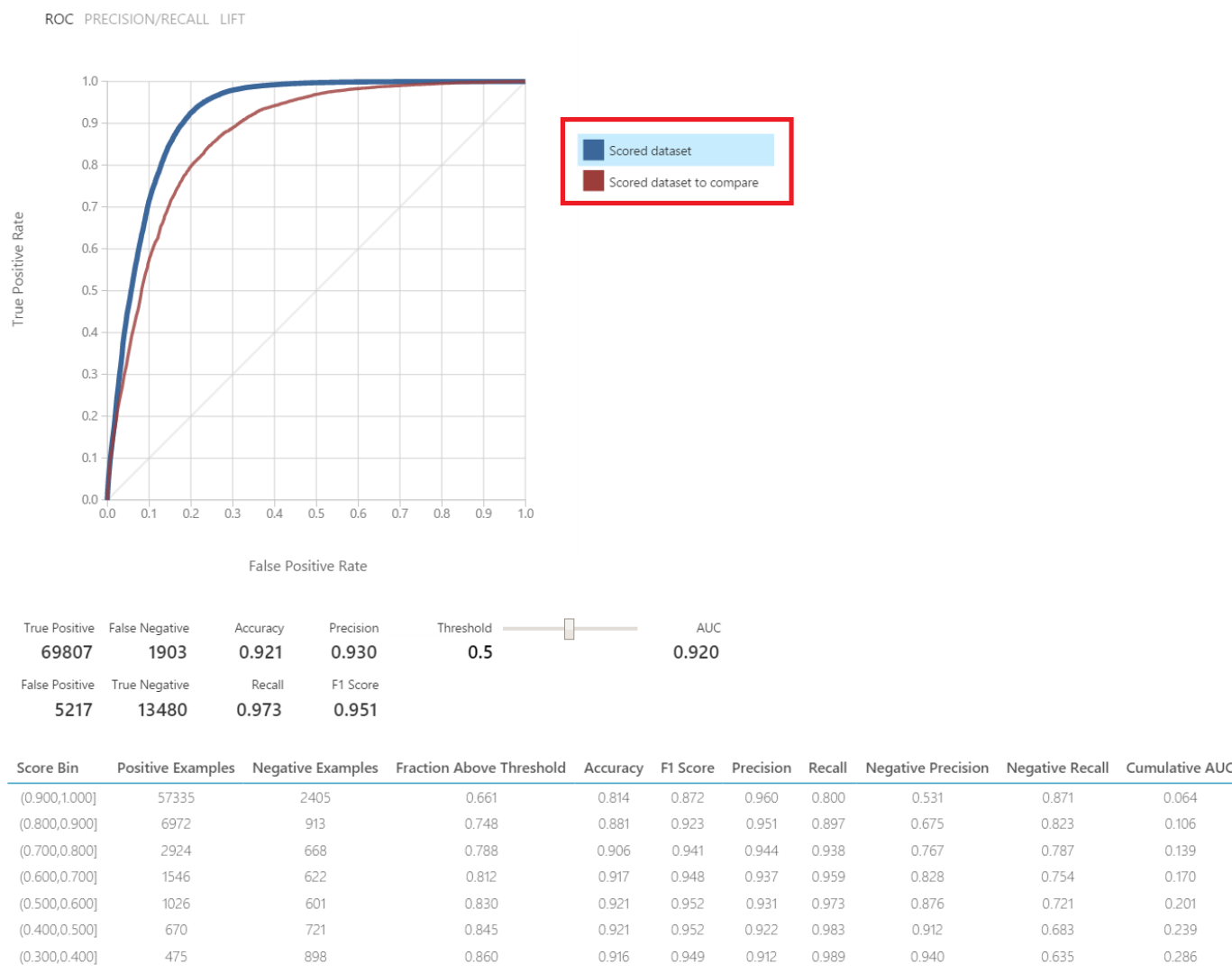
Then, we use a **Train Model** module to actually train the model. The **Train Model** module takes two inputs, the model specification and the training data. In **Train Model** module, we also specify that `churn_label` is the target column in the training data.



After the model is trained, we use two **Score Model** modules to predict the labels of data in train and validation sets, respectively. The **Score Model** module takes two inputs, the left input portal is the trained model, and the right input portal is the dataset you want to predict. It outputs a dataset which has all columns of the input data, and has two extra columns, Scored Labels and Scored Probabilities.

We use **Evaluate Model** module to calculate performance metrics for each of the input datasets. In this sample experiment, we input the predictions from both train and validation sets to the **Evaluation Model** module. Right click the output portal of this module, and select **Visualize**, the performance metrics and ROC curves of the model on these two datasets will show up.

KDDCup2015 Sample Experiment: Customer Churn Prediction > Evaluate Model > Evaluation results

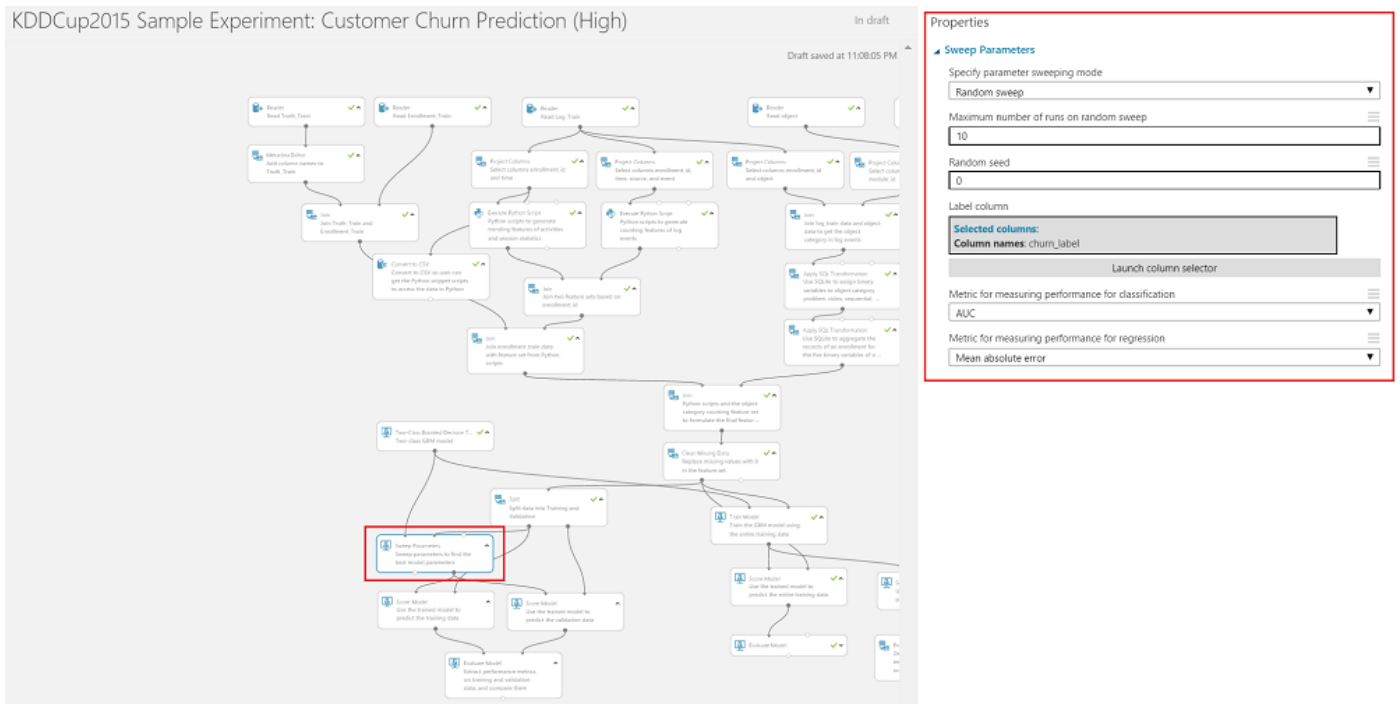


Clicking the blue or red legend on the top right corner of the ROC curve, the performance metrics switch between train and validation set. In this sample experiment, the AUC on train set is 0.92, and on validation set is 0.871.

Fine tuning the parameters of the machine learning model to optimize the performance on validation data

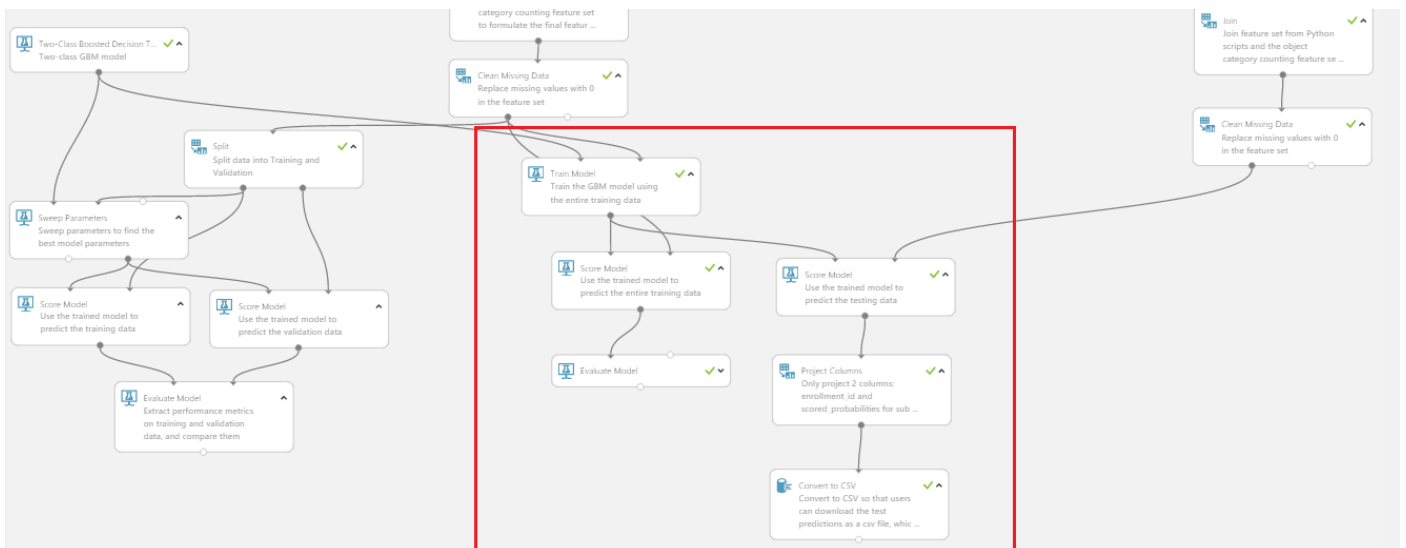
If you are not satisfied with the model performance, you can change the parameter settings and rerun the experiment to see whether the performance is improved. Alternatively, you can use **Sweep Parameters** to replace **Train Model** module. In the **Sweep Parameters** module, you can specify how many runs you want to sweep (number of different combinations of parameters), the target column, and the metric to determine the best model. The right output portal is the returned best model. You can use it to score the train and validation sets to see how it performs on these two sets.

KDDCup2015 Sample Experiment: Customer Churn Prediction (High)



With the optimized parameters, training a machine learning model on the entire training data

Now, we assume that we have found the best model on the feature set, we use the optimized parameters to train a new model on the whole training set (including both train and validation sets).



Using the model trained on the entire training data to predict the testing data

After the model is trained on the entire training feature set, we use this model to score the training and testing sets. Since the KDD Cup 2015 website only accepts submissions with two columns: enrollment_id and a column valued between 0 and 1 representing the churn probability, we use **Project Columns** module to only output **enrollment_id** and **Scored Probabilities** columns. We also use a **Convert to**

CSV module to convert the data to csv format. After that, right click the output portal of the **Convert to CSV** module, you can directly save the data as a .csv file on your local machine.

Before you submit the .csv file to the competition website for evaluation, you need to remove the headerline of the .csv file since the website does not accept submissions with headerlines.

	A	B	C
1	enrollmen	Scored Probabilities	
2	2	0.034251	
3	8	0.021106	
4	10	0.972852	
5	11	0.009207	
6	15	0.370365	
7	17	0.017495	
8	19	0.860107	
9	21	0.709814	
10	24	0.114623	
11	25	0.966817	
12	27	0.035207	
13	29	0.525528	
14	33	0.466208	

Debug Python scripts using Azure Machine Learning Python Client Library

In most of the cases, you need to debug your Python scripts before they can be successfully run in EPS. Azure Machine Learning releases a Python client library, which allows you to access any dataset in your AML workspace into your own Python development workspace. It will enable you to develop and debug your code locally. It improves the code development efficiency significantly.

You need to first install Azure Machine Learning Python client library on the machine you want to use to access the AML datasets. Instructions of installation can be found [in the Github repository](#).

In this sample experiment, the dataset that is input to the first EPS module is also passed to a **Convert to CSV** module. After the **Convert to CSV** module completes, right click its output portal, and select "Generate Data Access Code", a window will pop out with the Python codes to access the code. You just

✕

GENERATE DATA ACCESS CODE

Use this code to access your data

To programmatically access this dataset, simply copy the code snippet into your favorite development environment. [Learn More](#)

Note: this code includes your workspace access token, which provides full access to your workspace. It should be treated like a password.

CODE SNIPPET

Python

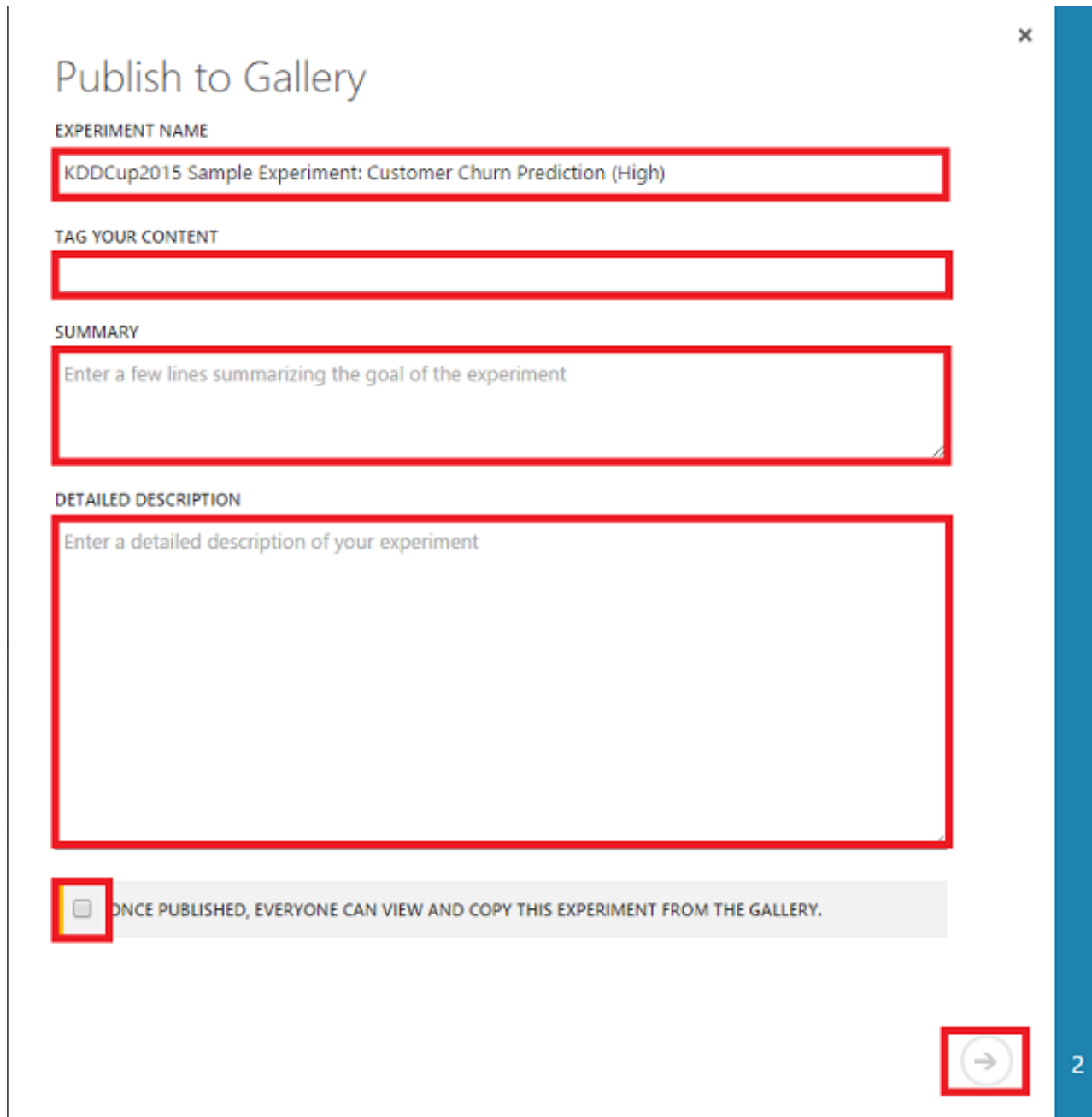
```
from azureml import Workspace  
ws = Workspace(  
    workspace_id='f1c582e1-2d11-4733-b8-3908-adcaa0e1f...',  
    authorization_token='eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...'  
)  
experiment = ws.experiments['6bf8ab45cca74738b8a3908adcaa0e1f...']  
ds = experiment.get_intermediate_dataset(  
    node_id='8f91ad56-6a31-4735-aa4d-defcd8910b5b-5877',  
    port_name='Results dataset',  
    data_type_id='GenericCSV'  
)  
frame = ds.to_dataframe()
```

☐ USE SECONDARY TOKEN

/C:/CloudML/Projects/KDDCup2015/blog_chiragedits%20(2)/blog/kddcup-sample-experiment-step-by-step-tutorial.html 15/22

Gallery

After your experiment completes, you can publish your experiment to gallery to share with others. At the bottom of the browser, click ****Publish to Gallery****. You will be asked to provide the name, summary, and the description to the experiment.



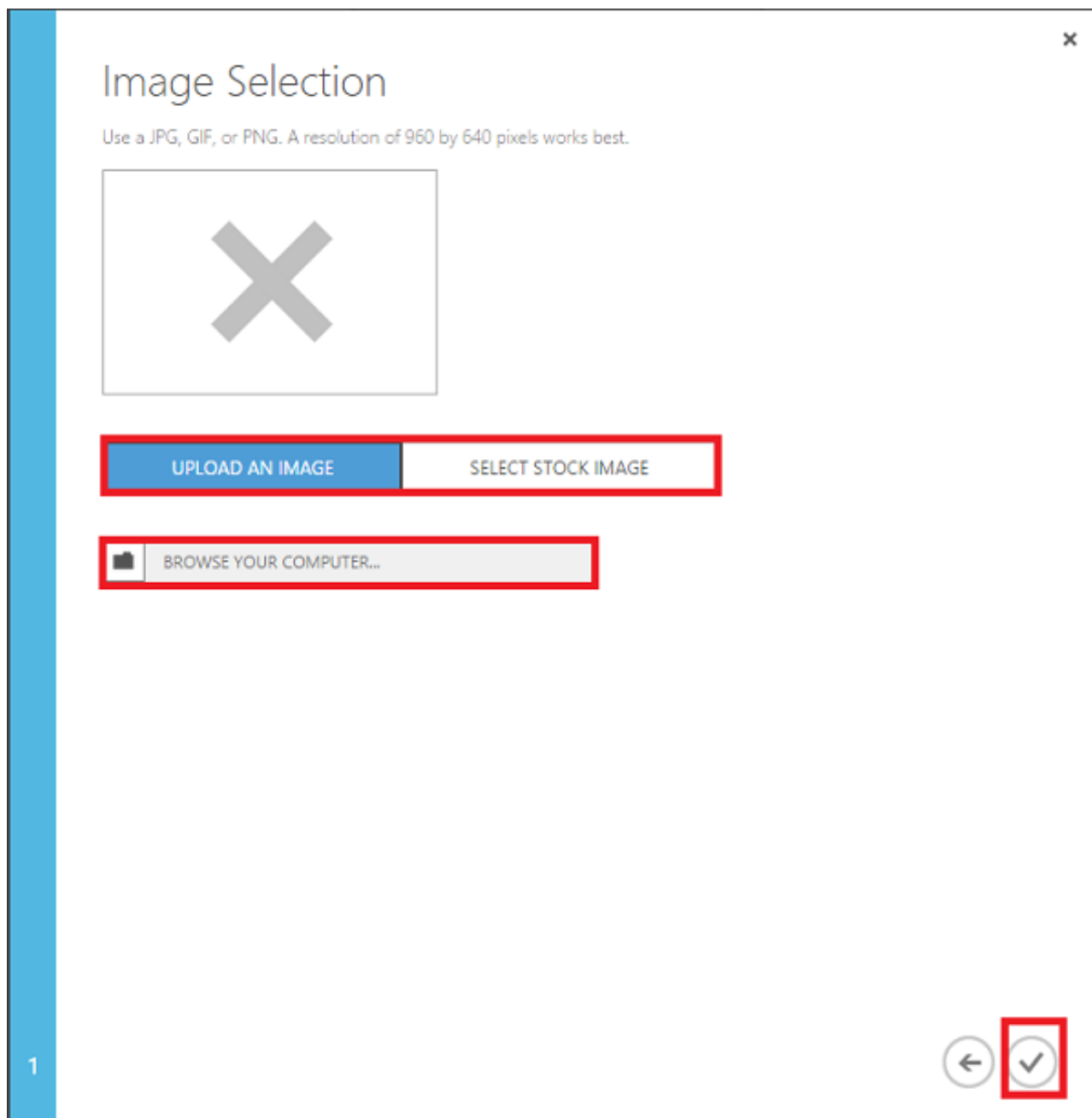
The screenshot shows a 'Publish to Gallery' dialog box with a close button (x) in the top right corner. The form contains the following sections:

- EXPERIMENT NAME**: A text input field containing 'KDDCup2015 Sample Experiment: Customer Churn Prediction (High)'.
- TAG YOUR CONTENT**: A text input field.
- SUMMARY**: A text input field with placeholder text 'Enter a few lines summarizing the goal of the experiment'.
- DETAILED DESCRIPTION**: A large text input field with placeholder text 'Enter a detailed description of your experiment'.
- PUBLISH**: A button with a checkmark icon, highlighted by a red box.

Below the input fields, there is a grey bar with the text: **ONCE PUBLISHED, EVERYONE CAN VIEW AND COPY THIS EXPERIMENT FROM THE GALLERY.**

At the bottom right of the dialog, there is a button with a right arrow icon, highlighted by a red box. To the right of this button, the number '2' is visible.

After you input the information, and agree with the term to allow others to view and copy the experiment, click the check mark and go to the second step. You will be asked to select an image for the sample experiment. You are suggested to select an image close to the purpose of this experiment so that the other users can easily grasp the idea from the graph.





Then, click the check mark and your experiment will be published. After the publishing completes, you should be able to see the following page, which provides you a URL to your sample experiments. You can share this URL by sending emails or directly twitting it.

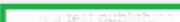


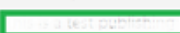
Click on the check mark, you will be directed to the page of your experiment in AML gallery. This is also the page the others are going to land on your experiment in AML gallery.


Browse ▾ Search for entities by name, algorithms or tags 🔍


EXPERIMENT


 **KDDCup2015 Sample Experiment: Customer Churn Prediction (High)**
by  June 18, 2015

Summary




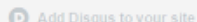
Description


0 Comments Azure ML Gallery  Login ▾


♥ Recommend  Share Sort by Best ▾



 Start the discussion...

Be the first to comment.

 Subscribe  Add Disqus to your site  Privacy




DISQUS

 [The image you previously uploaded]

 EDIT  DELETE

OPEN IN STUDIO

👁 0 views
⬇ 0 downloads

 Tweet  Share 

Tags

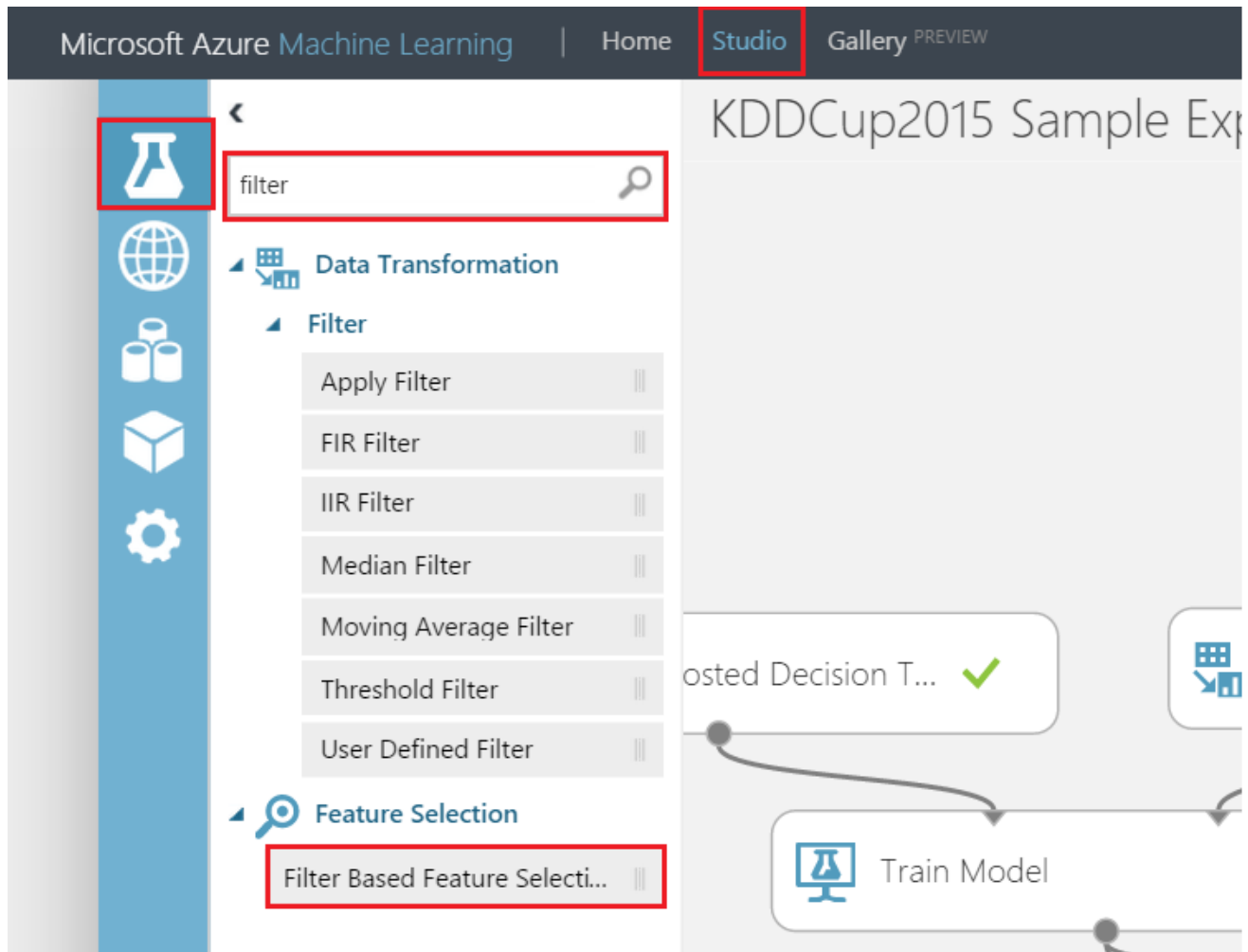
Customer Churn MOOC KDD Cup 2015

Algorithms

Two-Class Boosted Decision Tree

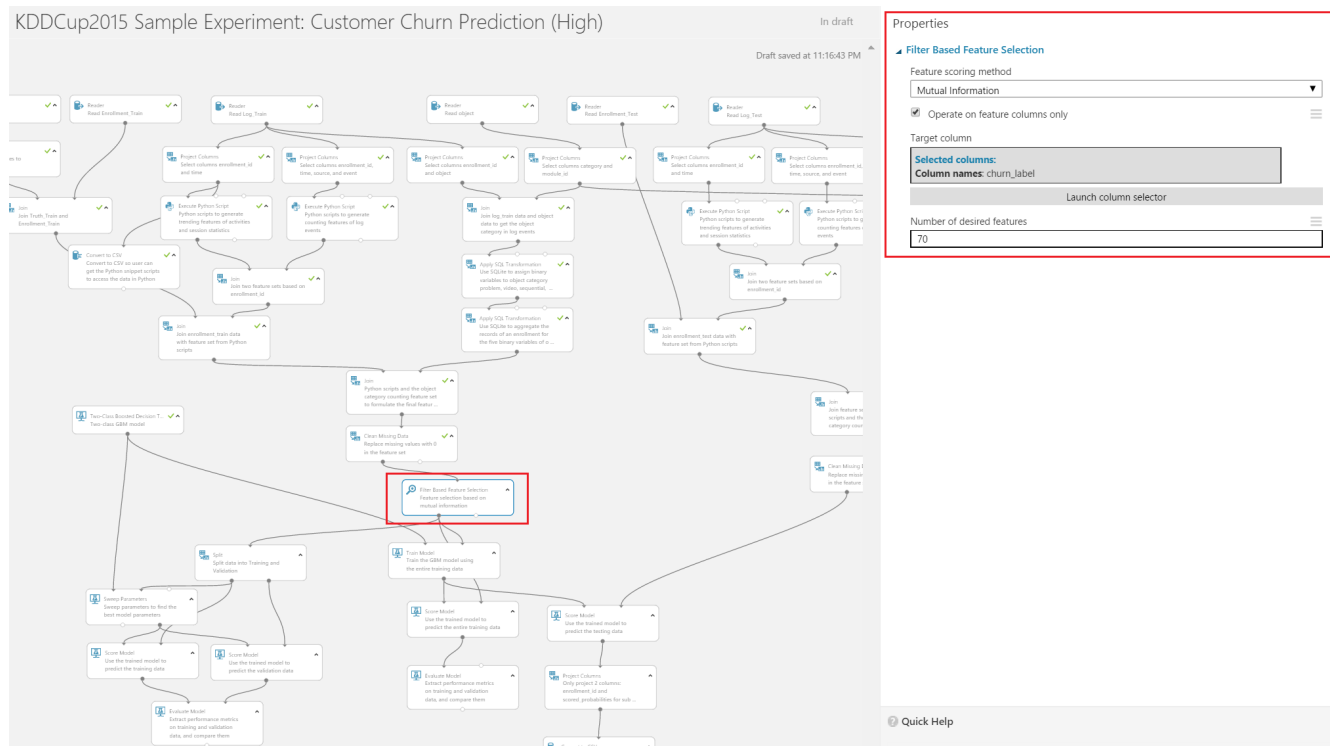
Other Modules That Might Be Useful

Azure ML has many other modules that might be helpful for this customer churn prediction task. For the sake of space, we cannot include all of them in these sample experiments. Here, we are giving a few modules and their example usages. You can find all these modules in Azure ML like the following figure.



1. Filter based feature selection.

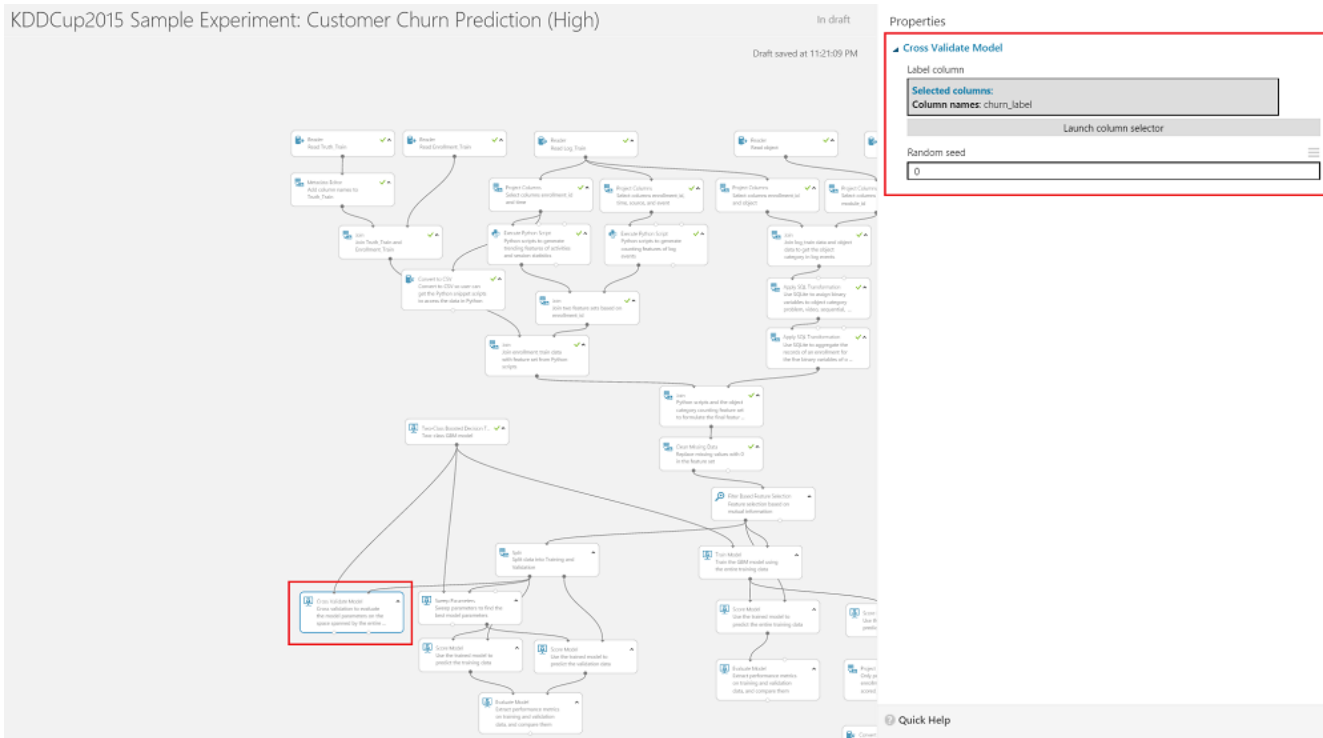
After your feature set is ready, in many times you want to do feature selection and train models only on the selected features. Azure ML has a module **Filter Based Feature Selection** to do this job for you. You just need to connect your feature set to the input portal of this module, select the criteria you want to use to measure the relationship between the feature and the target, specify the target, and tell the module how many features you want to select. Then, the module will output the top features for you to build model on. The following figure is an example.



2. Cross Validate Model

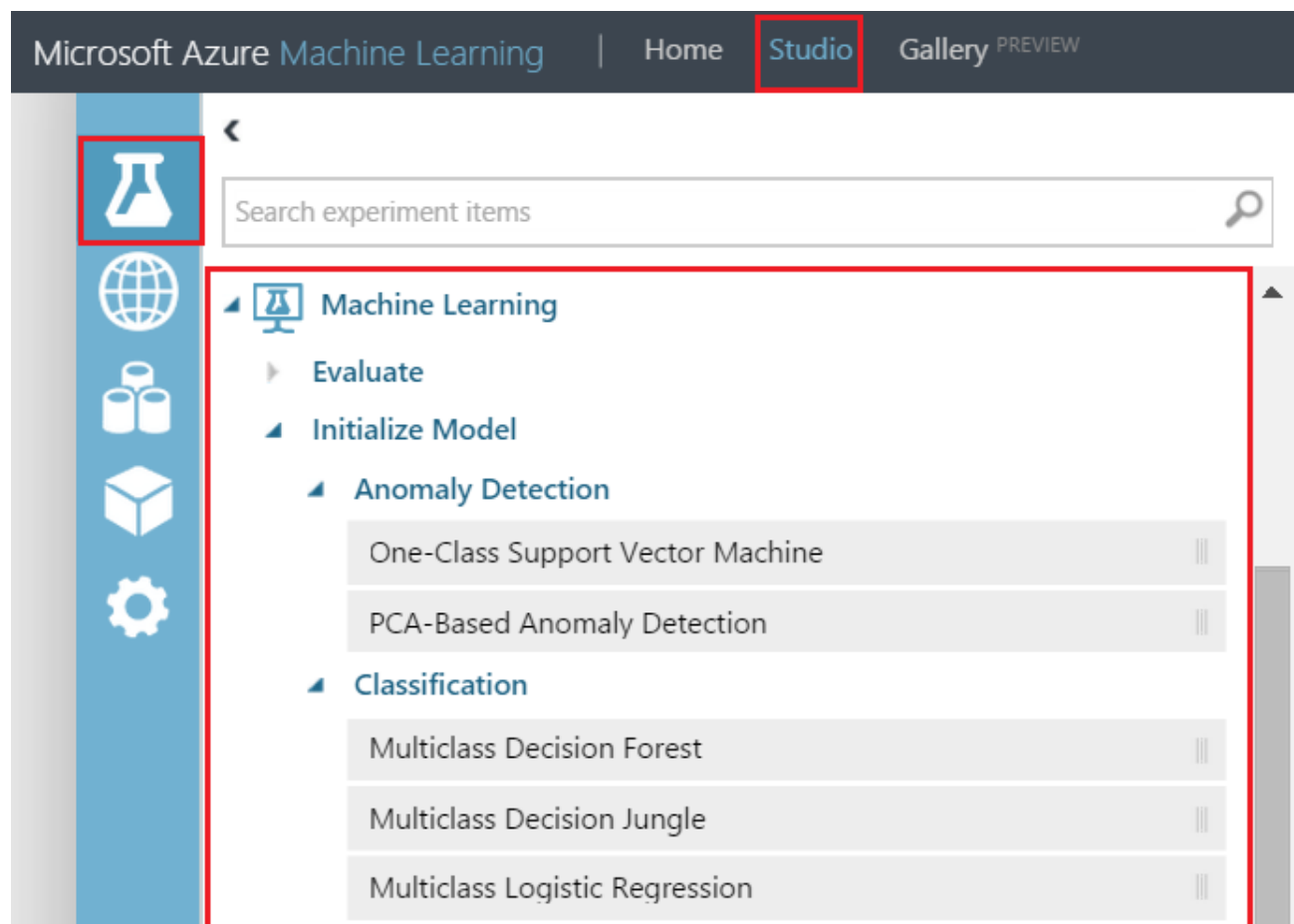
Cross validation is widely used to determine how the models trained under certain parameter setting perform in predicting new observations in the entire space spanned by the entire training set. In **Cross Validate Model** module, by default, the training data is split into 10 folds randomly. It trains 10 models with the same parameter setting. Each model is trained by leaving one fold out of the model training process. Then, the trained model is used to predict the left out fold. This module takes two inputs: model specification and training data. It outputs two datasets: the scoring results on the 10 left-out folds by their corresponding model and the performance metrics on each of the 10 left-out folds. The following figure is an example.

KDDCup2015 Sample Experiment: Customer Churn Prediction (High)



3. Other classification models

Azure ML has implemented many other classification models, and many regression models. You can see a complete list of models like the following figure. You can refer to [Microsoft Azure Machine Learning: Algorithm Cheat Sheet](#) to determine which model to use.



Multiclass Neural Network	
One-vs-All Multiclass	
Two-Class Averaged Perceptron	
Two-Class Bayes Point Machine	
Two-Class Boosted Decision Tree	
Two-Class Decision Forest	
Two-Class Decision Jungle	
Two-Class Locally-Deep Support Vector Machine	
Two-Class Logistic Regression	
Two-Class Neural Network	
Two-Class Support Vector Machine	
▲ Clustering	
K-Means Clustering	
▲ Regression	
Bayesian Linear Regression	
Boosted Decision Tree Regression	
Decision Forest Regression	
Fast Forest Quantile Regression	
Linear Regression	
Neural Network Regression	
Ordinal Regression	

Summary

Hopefully these experiments allow you to kickstart your participation for the KDD Cup as well as learn more about Azure ML. For questions reach out to : hangzh@microsoft.com.