Retail Churn Template: Step 1 of 4, tagging data – AzureAI Gallery

## **Summary**

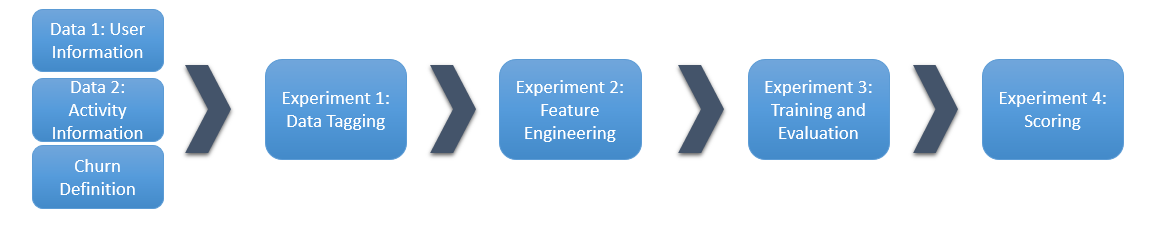
This template demonstrates the steps to build a retail customer churn prediction model.

## **Description**

# Retail Churn Prediction Template

Predicting Customer Churn is an important problem for banking, telecommunications, retail and many others customer related industries. As a part of the Azure Machine Learning offering, Microsoft is providing this template which can help retail companies predict customer churns. This template provides pre-configured machine learning modules along with custom Python scripts in the **Execute Python Script** Module for solving the customer churn prediction problem for the Retail Stores. This template focuses on binary churn prediction, i.e. classifying the users as churners or non-churners.

The overall template is divided into 4 major experiments with each containing one of the following steps:



Here are the links to each step (experiment) of the template:

[**Retail Churn Template: Step 1 of 4, tagging data**](http://go.microsoft.com/fwlink/?LinkId=626665)

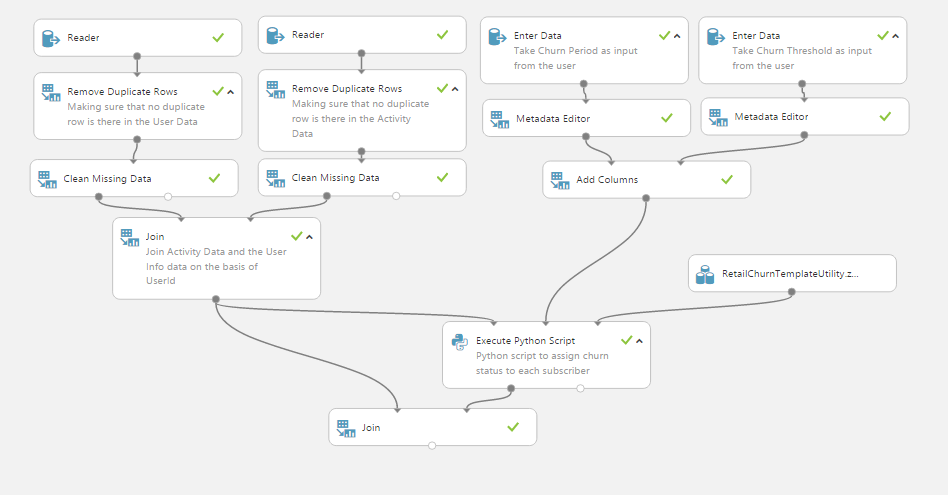
[**Retail Churn Template: Step 2 of 4, feature engineering**](http://go.microsoft.com/fwlink/?LinkId=626666)

[**Retail Churn Template: Step 3 of 4, train and evaluate model**](http://go.microsoft.com/fwlink/?LinkId=626669)

[**Retail Churn Template: Step 4 of 4, scoring**](http://go.microsoft.com/fwlink/?LinkId=626670)

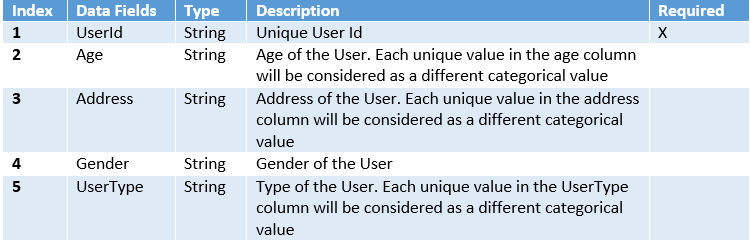
## **Step 1: Tagging Data**

Detail of this experiment can be seen in the following figure

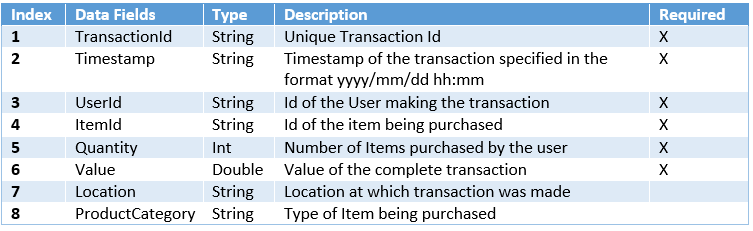


Retail Churn Template takes two data sets as input: User Information data set and the Activity Information data set. Any data following the schema of the User Information Data set and the Activity Data set can be used with the churn template. Furthermore, this churn Template is generalized to handle different churn definitions on the granularity of number of days as input.

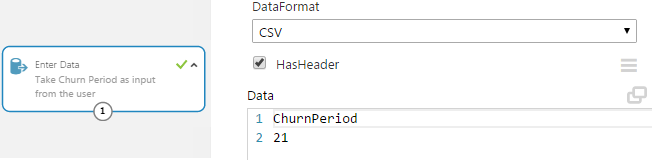
Schema of User Information Data is shown in the following table



Similarly, schema of Activity Data is shown in the following table



Some of the fields like Gender and UserType are having "Unknown" value because they were not available in this data set. This template is designed to be generalized so it works regardless of the availability of the optional fields. Furthermore, this template depends on the definition of the Churn which has to be provided by the user as shown in the following figure



Each substep in this experiment is explained as follows

### **Data Input**

**1.0** Read User and Activity data. The Reader reads the User Information Data from the file RetailChurn\_UserInfoData.csv and the Activity Information Data from the file RetailChurn\_ActivityInfoData.csv on the public blob. These data sets can be replaced by other data sets with the same names provided the data schema is followed.

**1.1** Read churn conditions. The **Enter Data** Module is used to read the churn definition provided by the user. Two parameters must be specified: 1) Churn period and 2) Churn threshold. These two parameters are used to identify churners as explained in step 1.6 below.

### **Data Cleaning and Merging**

**1.2** \*Remove Duplicate Rows\* modules is used to make sure that there are no duplicate entries in either of the datasets.

**1.3** \*Clean missing data\* module is used to replace the missing fields in the data sets with Unknown value

**1.4** Similarly **Metadata Editor** module is used to change the type of the churn period to int.

**1.5** Cleaned User Information data and the Activity Information data are merged into one data sets using the **join** module which is passed on as input to following **Execute Python Script**.

### **Churn Labeling**

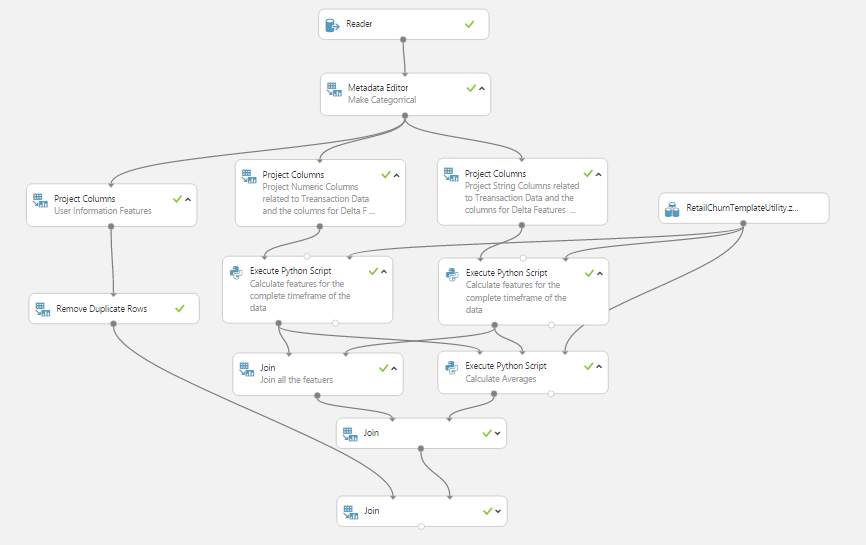
**1.6** The output from step 1.5 and the churn parameters are provided as input to Churn Labeling Python script in **Execute Python Script** module. This script mainlu uses the module **RetailChurnTemplateUtility** to label each of customer as churner or non churner. The output is a tuple of .

Churners are those customers whose activities are less than the churn threshold during the last n days seen in the Activity Information Data where n is equal to the churn period. Note that in order to identify churners, the Activity Information Data must contain activties that are dated prior to the start of the churn period. Currently, the template assumes that the Activity Information Data spans a period that is at least twice longer than the churn period. For instance, if the churn period is set to 21, the Activity Information Data should contain activities for at least 42 days of which the last 21 days are used to identify the churners.

**1.7** The labeled user data from **1.6** is merged back with the joined **user+activity** data using the **join** module, which is the final tagged data. The output can be written to template user's blob account as RetailChurnTemplate\_TaggedData.csv or saved as a Azure ML dataset named RetailChurnTemplate- TaggedData. It is going to be used in next step of this template.

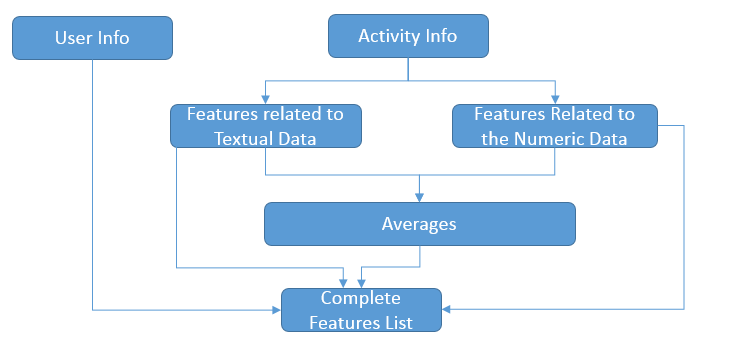
Note that the template user may assign the churn labels to users offline. In that case, the user can skip the labeling step and just join it with the transnational data. Moving from one step to another in this template, the only requirement is that the output of the preceding step should follow the input requirements of the following step.

## **Step 2: Feature Engineering**

The purpose of this step is to generate advanced features that can help in accurate prediction of the churn status. The experiment snapshot is shown below: 

The tagged input data(RetailChurnTemplate\_TaggedData.csv) to this experiment consists of some fields (numeric fields) for which we may be interested in total sum ( for example: Quantity, Value etc) while for some others (textual/string fields) we may be interested in counting the number of unique entries ( for example, Location, Address, Product Category). This observation is the basis of the Feature Generation process that we have developed. For textual fields we are interested in calculating the number of unique values for each of the user while for the numeric features we are interested in calculating the total aggregate and standard deviation for each of the user.

The complete work flow of the feature generation process is as shown in the following figure



### **Data Input**

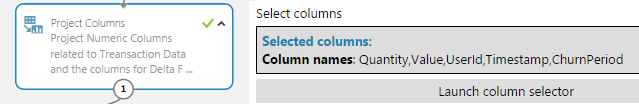
**2.1** This feature engineering experiment reads the tagged data saved by the Tagging experiment (RetailChurnTemplate\_TaggedData.csv) through Reader Module. Then it uses **Metadata editor** to change some of the fields in the input data as categorical. Advantage of this transformation is that Azure ML automatically handles the categorical features and adds dummy values internally.

### **User Information Features Projection**

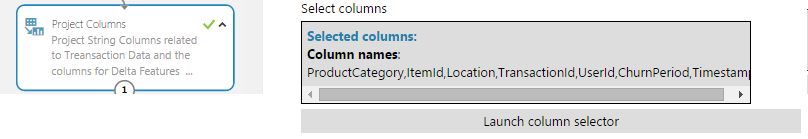
**2.2** Most of the features related to the User Information are categorical features and they are automatically handled by Azure ML, hence they are separated from rest of the data using **Project Columns** Module and later on merged with the derived features at the end.

### **Activity Information Projection**

**2.3** The numeric data is to be handled separately from the textual data. First of all the numeric activity related fields are separated from the other data using **Project columns** Module and then passed on to the Python script in **Execute Python Script** module. Note that along with the numeric fields some other fields like User Id etc should also be projected to make sure that the advanced features can be generated.

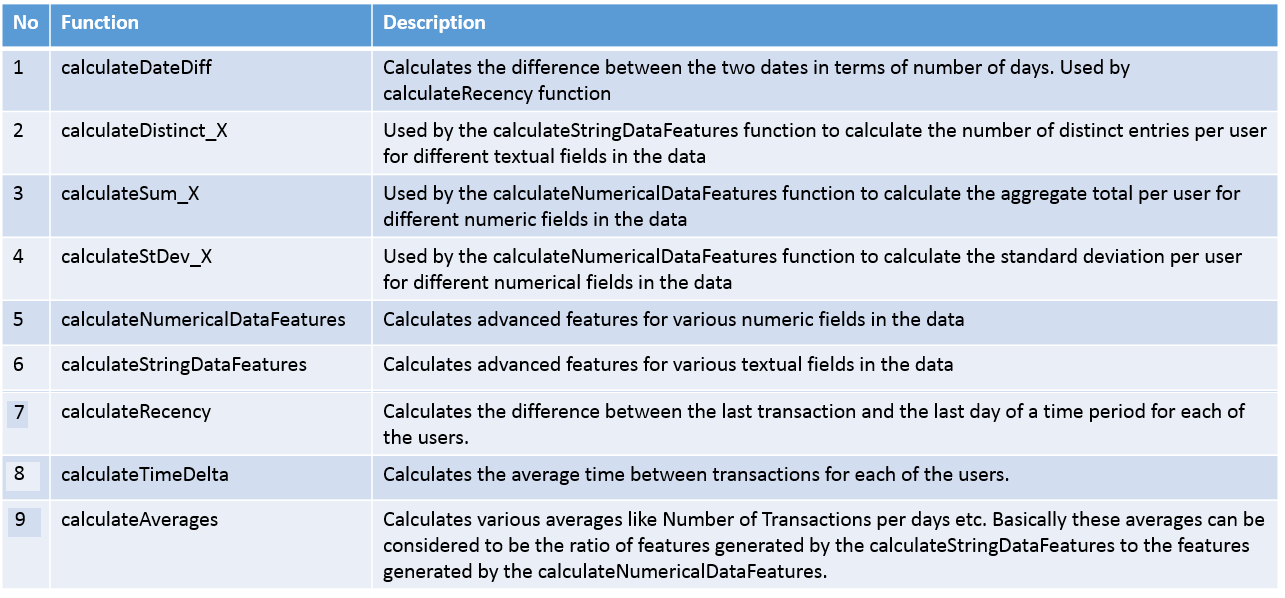


**2.4** Similarly, the textual or categorical fields ( The features for which we are interested in calculating the number of unique features) are separated from the other features as shown in the following figure.



### **Features Generation**

**2.5** The numeric features and textual features are passed on to separate **Execute Python Script** modules to generate derived features. The **RetailChurnTemplateUtility** python module contains the RetailChurnTemplateUtiltiy class which has the following helper functions to generate the advanced features for numeric and categorical input fields:



The two main functions used for the feature generation are **calculateNumericalDataFeatures** and **calculateStringDataFeatures** for numeric and categorical features respectively. They take a parameter isDevelopment. Value of True means this is for development, and the input transaction data includes the labeling period, hence the labeling period (in this case, the data of last 21 days) is excluded from feature engineering. In the scoring or web service step, the indicator is set to False for production which then utilize the entire data up to the latest date in the dataset for feature engineering.

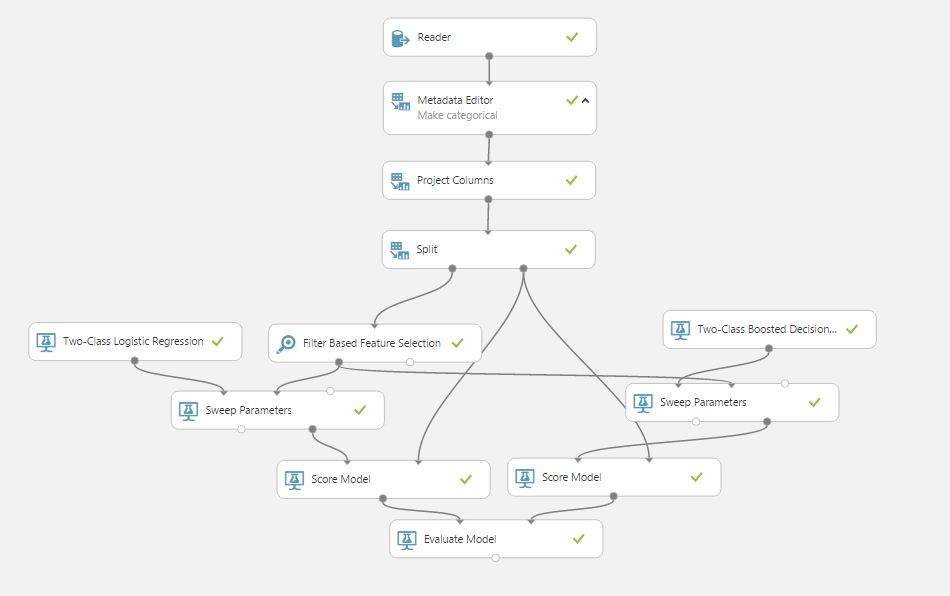
The results of these two execute python scripts module is merged together using the **Join** module. Furthermore, the output is also sent to another **Execute Python Script** module that calls the calculateAverages function of the RetailChurnTemplateUtility class. The final output of this experiment is a dataframe containing all the features ( numerical, textual, and the averages) combined together into one dataframe.

The output of this experiment is going to be used by the Training and Evaluation experiment next hence it should be saved as RetailChurnTemplate\_FeatureEngg.csv

You may observe that the churn rate is different at the end of step 1 and step 2 due to two main reasons. First of all, the output of step 1 is still at transaction level while the step 2 output is aggregated at user level. Secondly, some of the users may be having transactions only in the churn period or may not be having enough transactions, so these users get filtered out in the labeling process and the feature generation process.

## **Step 3: Training and Evaluation**

The purpose of this experiment is to train different classifiers and evaluate their performance on the test data. As the Feature Engineering experiment may generate many different features, we can perform feature selection to reduce it down.



### **Data Input**

**3.1** The reader module reads the data generated by the Feature Engineering experiment(RetailChurnTemplate\_FeatureEngg.csv) from a public blob

### **Data Cleaning and Filtering**

**3.2** The categorical variables in the data are converted into Categorical type so that they can be automatically handled by Azure ML. User Id column is removed from the data because these should not be used for training of the classifiers.

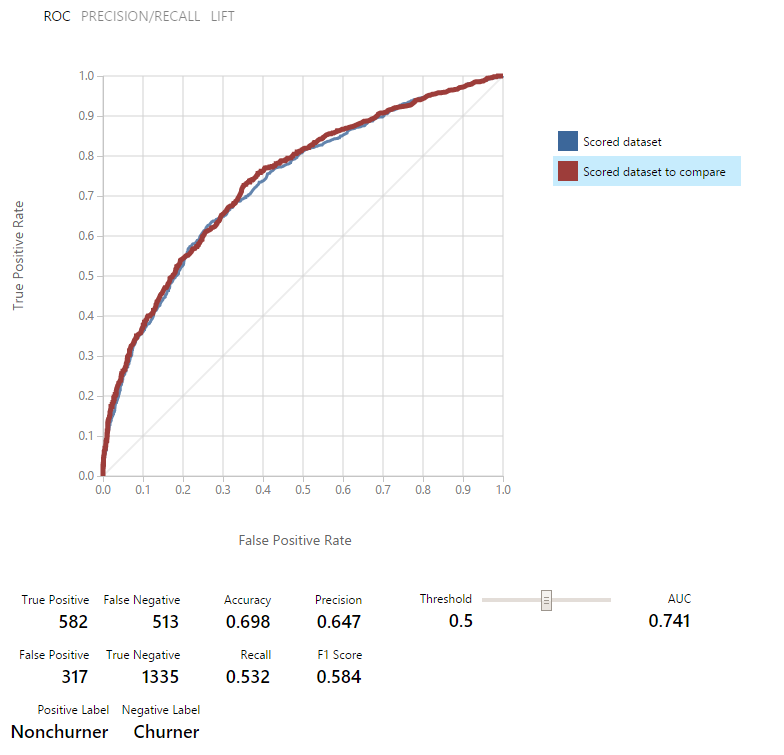
**3.3** Test/ Train Split The split module is used to split the data into Training and Testing Sets.

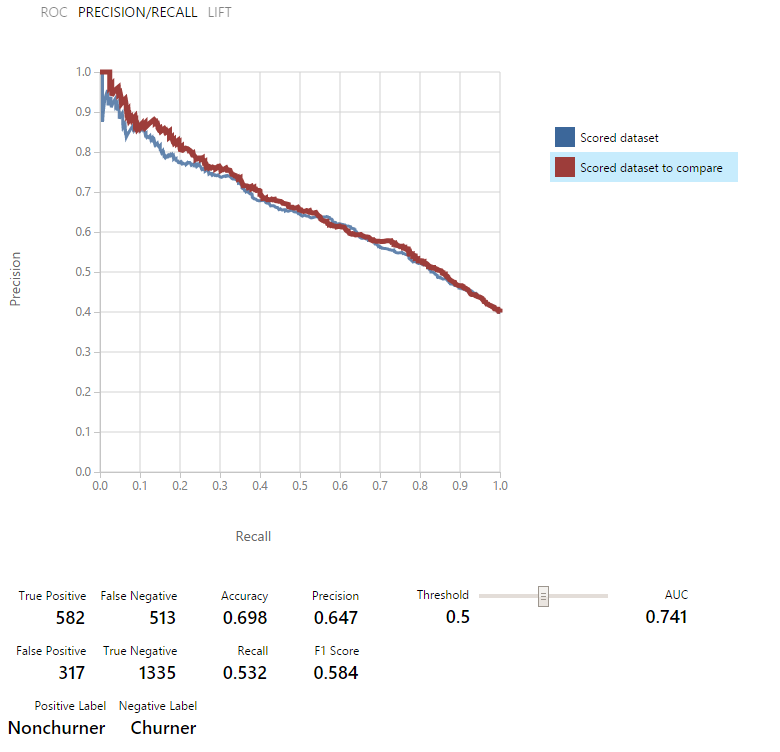
**3.4** Feature Selection Feature Selection module is used to select total number of desired features based on the Mutual Information score.

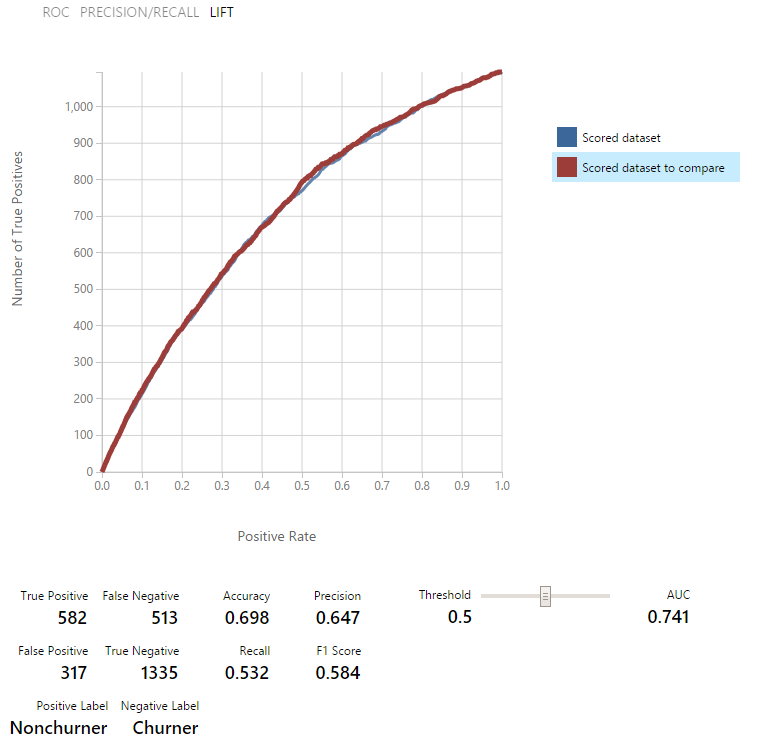
**3.5** Classification Models For this particular template, two class Logistic Regression and two class Boosted Decision Trees Classifiers are used and their parameters are optimized using the Sweep parameters modules.

**3.6** Model Scoring and Evaluation

In the end Score Model and Evaluate Model modules are used to compare the performance of the Logistic Regression and Boosted Decision Tree. The ROC Curve, Precision/Recall trends and the Lift Curves for this particular data set can be seen in the following figures:



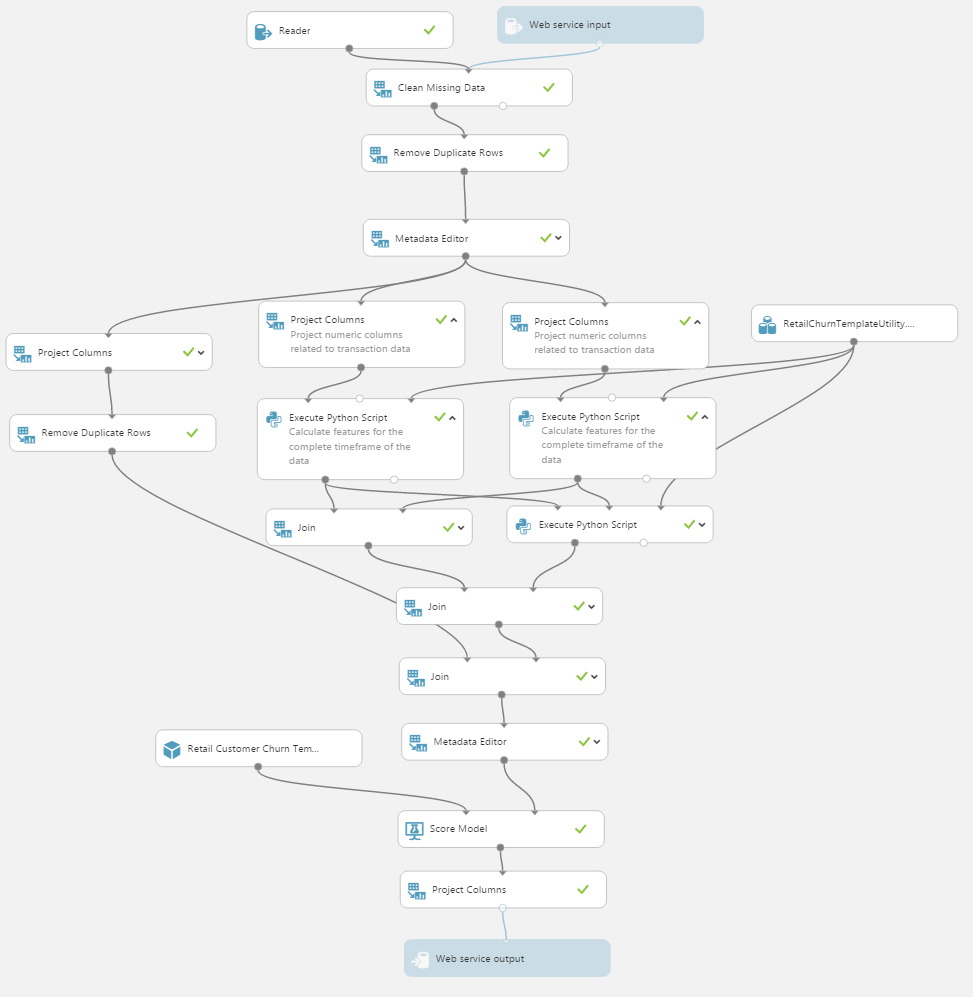




The Boosted Decision Tree is performing slightly better than the Logistic Regression. Trained Boosted Decision Tree should be saved as a Retail Churn -Trained Model as it is going to be used in the scoring experiment. The user may evaluate different classifiers and the classifier performing the best on the data should be saved as Retail Customer Churn Template -Trained Model.

## **Step 4: Scoring**

The trained model is ready for the scoring or predictive experiment. For this experiment, we connect the data processing, feature engineering modules, saved models or transformations and scoring module into one experiment and pass the scoring data (combined user and activity data) as input as shown in the following graphic.



Note that in the feature engineering Python Script modules, the indicator isDevelopment is set to false, as it is for production. By turning off this indicator, it will utilize all the data up to the latest date available in the input dataset for feature engineering, without leaving a churn period unused as done in step 2.

This can then be published as a web service after identifying input and output, running the experiment, and upon finish clicking the "Deploy Web Service" button. The web service input is specified as the same format as the reader, and the output are Scored Labels and Scored Probabilities --the output of the **Project Columns** module, as shown in the graph. The web service and be accessed in RRS or BES modes as shown in [accessing web service](https://azure.microsoft.com/en-us/documentation/articles/machine-learning-walkthrough-6-access-web-service/)

## **Summary**

Microsoft Azure Machine Learning provides a cloud-based platform for data scientists to easily build and deploy machine learning applications. The purpose of this Retail Customer Churn Template provides an easy to use template that can be used with different datasets and different definitions of Churn, which can be extended by users. This template also demonstrates the capability of the AzureML studio to handle data cleaning and processing using Python libraries like Pandas and Numpy. The features generation process described in this experiment can be used in other experiments as well.

