Demos

1. Running HClust and DBScan in AzureML (http://tinyurl.com/hclust-and-dbscan)

We'll also see how to setup a web service.

2. Demo of a typical ML pipeline (http://tinyurl.com/flight-delay-demo). The goal is to understand new Azure ML modules along with the pipeline.

LAB ACTIVITY - BUILD A LINEAR REGRESSION MODEL AND A LOGISTIC REGRESSION MODEL ON THE BIKE RENTAL DATA

Data Understanding

0. Download the dataset from http://tinyurl.com/bike-sharing-data

The data is about how many bikes are rented in a public sharing platform at a given datetime. Following are the columns present in the dataset -
datetime, season, holiday, workingday, weather, temp, atemp, humidity, windspeed, casual, registered, count

The target is to predict a count based on the other features. We shall use both linear and logistic regression on this target.

Experiment (Feel free to apply your own modifications)

1. Load the dataset using NEW > DATASET

2. Create a new experiment and load the dataset. Use edit metadata to convert Datetime to String format.

3. Create an RScript module which exctracts date, hour, weekday, month and year from the datetime column. Use the Execute R Script Module and the code given below.

```
==========================================================================================

dataset <- maml.mapInputPort(1)

# extracting hour, weekday, month, and year from dataset
dataset$datetime <- as.POSIXct(dataset$datetime, format = "%m/%d/%Y
%I:%M:%S %p")
dataset$hour  <- substr(dataset$datetime,  12,13)
dataset$weekday  <- weekdays(dataset$datetime)
dataset$month  <- months(dataset$datetime)
dataset$year  <- substr(dataset$datetime,  1,4)

#Preserving the column order
Count <- dataset[,names(dataset) %in% c("count")]
OtherColumns <- dataset[,!names(dataset) %in% c("count")]
dataset <- cbind(OtherColumns,Count)
```

```
# Remove single observation with weather = 4 to prevent scoring model
from failing
dataset <- subset(dataset, weather != '4')

# Return the dataset after appending the new features.
maml.mapOutputPort("dataset");
```

==========

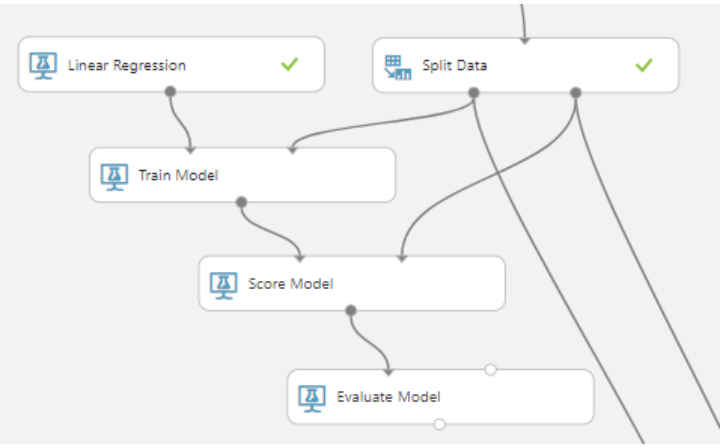
4. Visualize the data and convert the necessary 'string' variables into 'categorical'. Use Edit Metadata module here

5. Drop the 'datetime', 'casual' and 'registered' columns using the "Select Columns Module"

For Linear Regression:

6. Split the data into test/train

7. Train a linear regression model using the “Count” as the target/label variable.



8. Score on the test data and evaluate and check the error metrics.

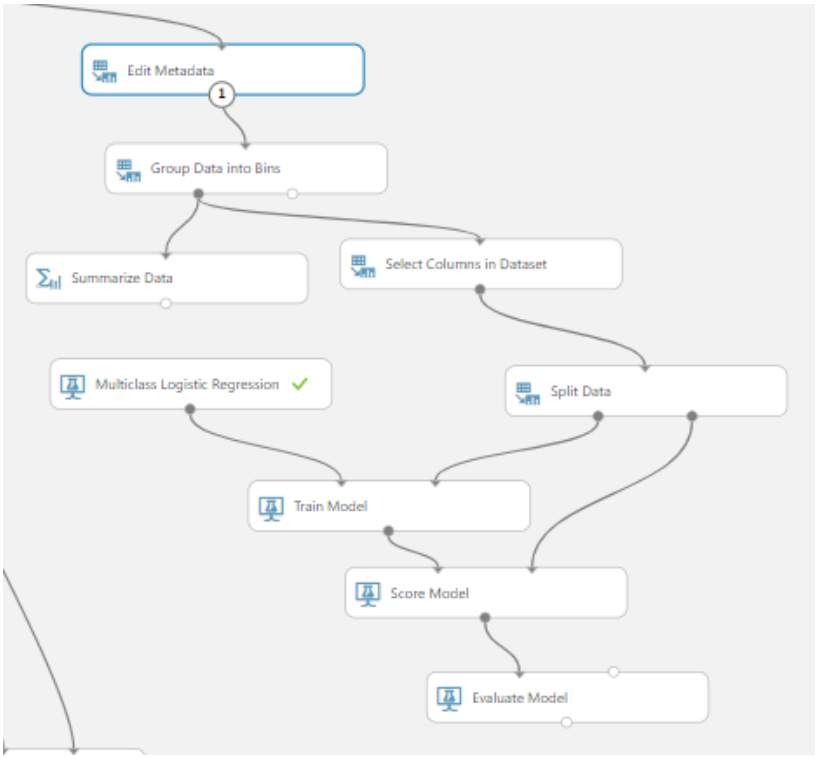
For Logistic Regression:

6. Group the target "Count" into bins using "Group Data into Bins". Select 3 or 4 bins as parameter

7. Drop the "Count column"

8. Split the data

9. Train a logistic regression model, using Multiclass Logistic Regression Module



10. Score on the test data, evaluate and check the error metrics

Once the experiments are run, modify the training parts to include an additional validation split that is input to "Tune Model Hyperparameters" for better tuning of the linear and logistic regression models. In this case we are using a Two-Class boosted tree. You should observe better accuracies with this.