Azure Machine Learning

Course Material

Part 1: Core Exercises

Contents

Lab 1: Creating an Azure Machine Learning Studio Workspace and Familiarization

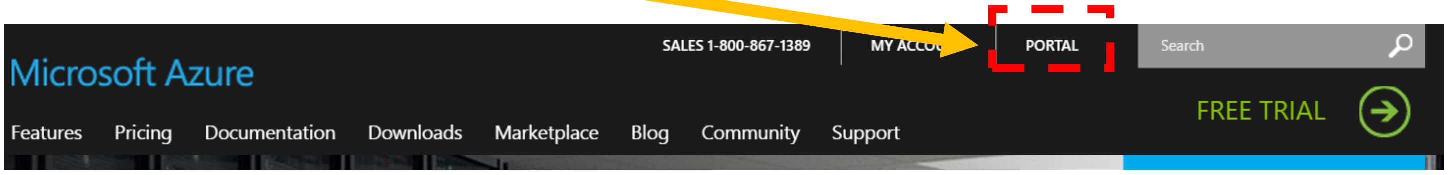
Skip this lab if you already have an Azure ML Account and know how to log in.

1. Log in using your credentials. Sign up for a free Azure trial if you do not have an account.
   a. http://azure.microsoft.com/en-us/pricing/free-trial/

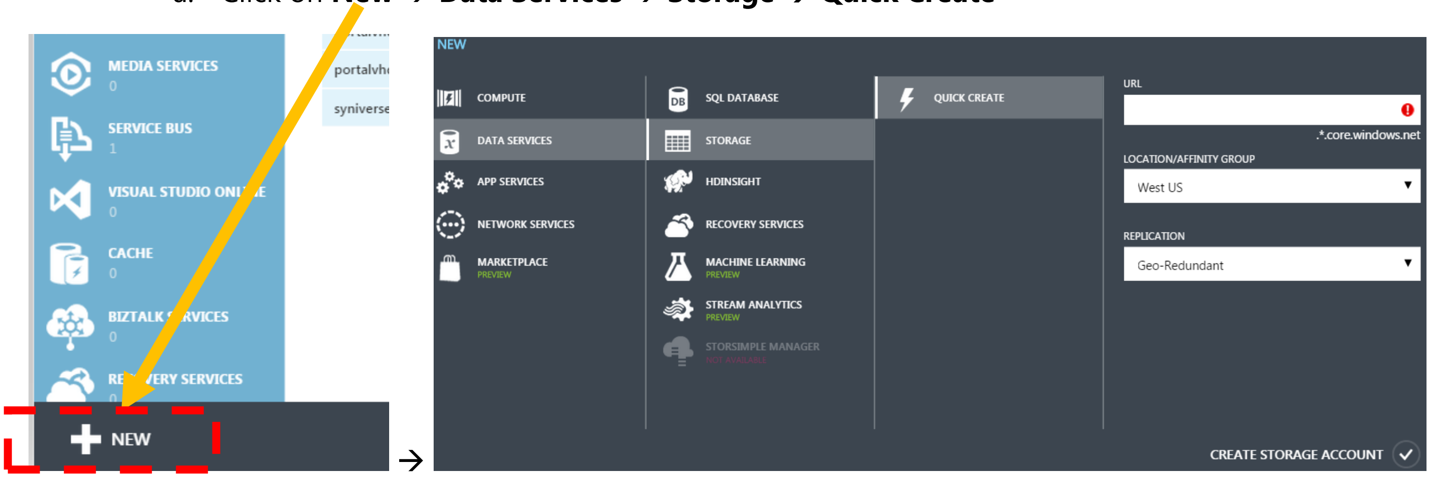
Exercise 1: Create a **dedicated** Azure Storage

Even if you already have an Azure Storage account. It is **highly recommended** that you created a **dedicated** Azure Storage account for your Azure Machine Learning Studio workspace.

1. Log into your Azure Management Portal.
   a. Go to: http://azure.microsoft.com/
   b. Click on the portal button.



2. Create a dedicated Azure Storage account for your Azure Machine Learning Studio workspace.
   a. Click on **New → Data Services → Storage → Quick Create**

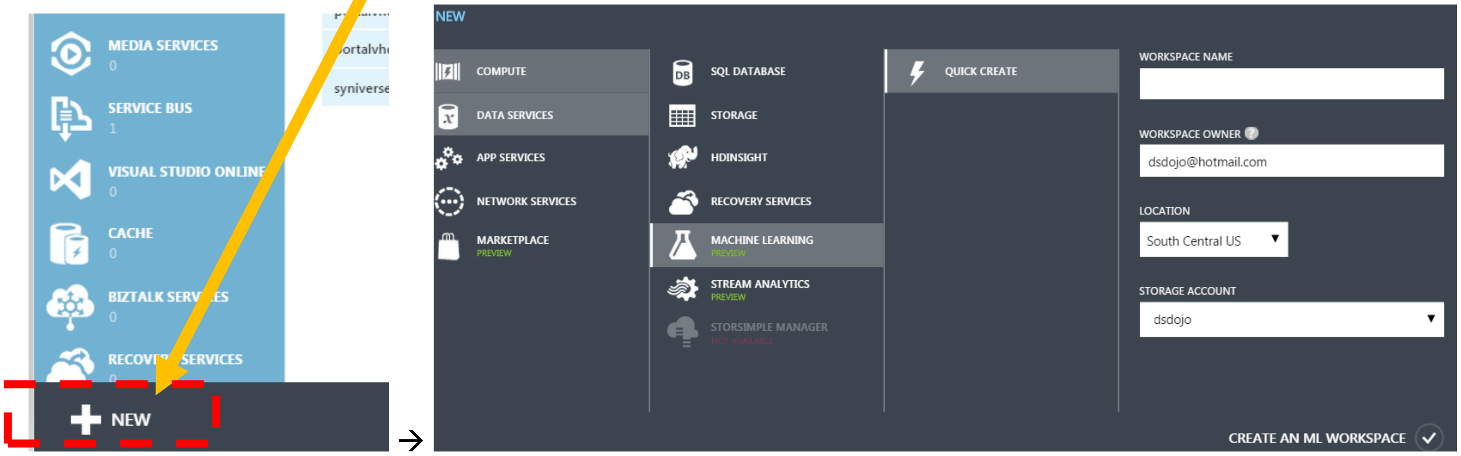


   b. Specify a URL. This will be the name as well as the HTTP URL to your storage account. Since this account will be for our Azure Machine Learning Studio account, it is best practice and **highly recommended** that you have the machinelearningstudio somewhere in the url name so there is no doubt or confusion as to what the storage account is for (so you don't accidently delete the storage in the future). The URL name will also have to be globally unique.
   c. Location/Affinity group: Select a region that's closest to you or your team of data scientists.
   d. Replication: this will control the fault tolerance of your storage. Locally or Geo-Redundancy should suffice.
   e. Hit okay and wait for your storage account to be provisioned.

Exercise 2: Creating an Azure Machine Learning Studio workspace.

Once you have a dedicated Azure storage account, you can now create an Azure Machine Learning Studio workspace.

1. Provision a new Azure Machine Learning Studio workspace (if you don't' have one already).
   a. Click on: **New → Data Services → Machine Learning → Quick Create**



WORKSPACE NAME

WORKSPACE OWNER: dsdojo@hotmail.com

LOCATION: South Central US

STORAGE ACCOUNT: dsdojo

   b. Workspace: Name your workspace. This is name has to be globally unique.
   c. Workspace Owner: set your azure account email (or someone else if you'd rather them be admin).
   d. Storage Account: reference your dedicated Azure Storage account.
2. Wait for your Machine Learning Studio workspace to be provisioned.

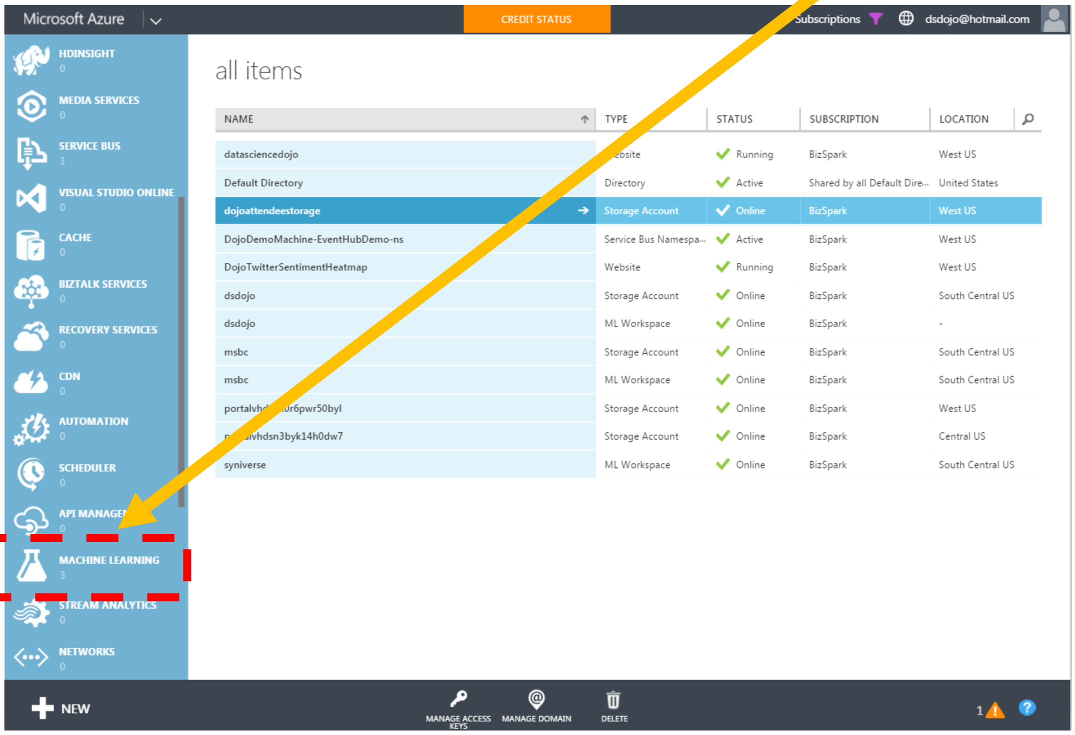
Tips and Notes:

You can invite others to work on your workspace by sharing it with them. This is the closest thing that data scientists have to a collaborative tool such as google docs. You can also copy and paste experiments across workspaces.

Exercise 3: Accessing your Azure Machine Learning workspace.

Once you have a dedicated Azure storage account, you can now create an Azure Machine Learning Studio workspace.

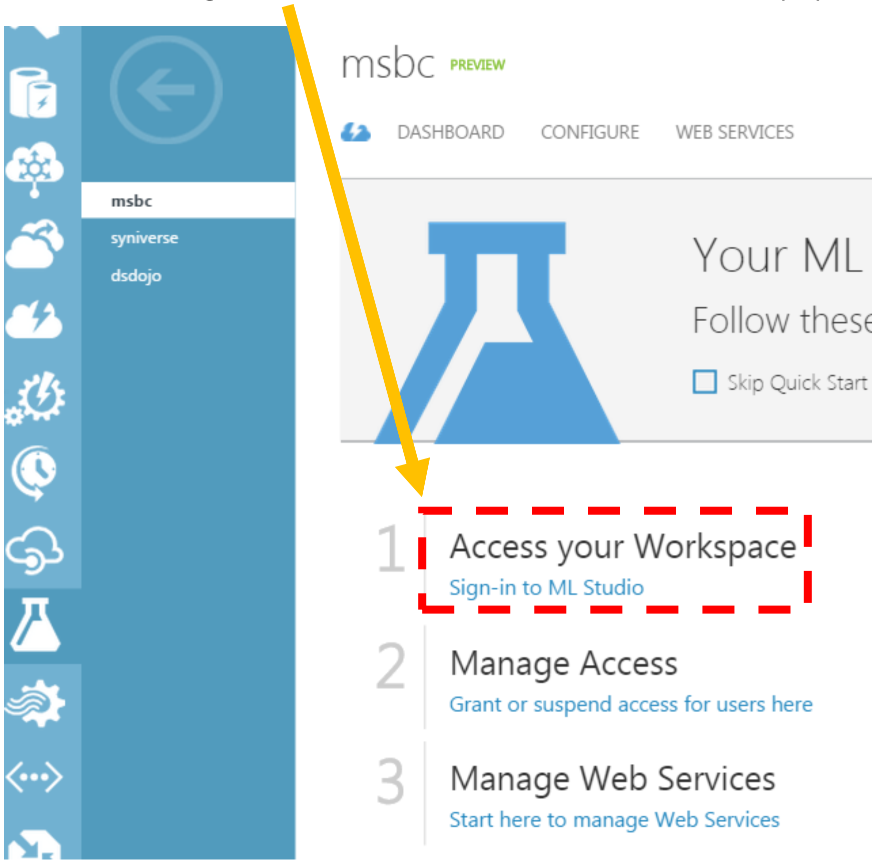
1. Within your Azure Management Portal. Click on Machine Learning.



all items

NAME	TYPE	STATUS	SUBSCRIPTION	LOCATION
datasciencedojo	Website	✓ Running	BizSpark	West US
Default Directory	Directory	✓ Active	Shared by all Default Dire...	United States
dojoattendeestorage	Storage Account	✓ Online	BizSpark	West US
DojoDemoMachine-EventHubDemo-ns	Service Bus Namespa...	✓ Active	BizSpark	West US
DojoTwitterSentimentHeatmap	Website	✓ Running	BizSpark	West US
dsdojo	Storage Account	✓ Online	BizSpark	South Central US
dsdojo	ML Workspace	✓ Online	BizSpark	-
msbc	Storage Account	✓ Online	BizSpark	South Central US
msbc	ML Workspace	✓ Online	BizSpark	South Central US
portalvh[illegible]0r6pwr50byl	Storage Account	✓ Online	BizSpark	West US
[illegible]lvhdsn3byk14h0dw7	Storage Account	✓ Online	BizSpark	Central US
syniverse	ML Workspace	✓ Online	BizSpark	South Central US

2. Click on your provisioned workspace.
3. Click "Sign-in to ML studio." A new window will now populate with your ML studio.



1 Access your Workspace
Sign-in to ML Studio

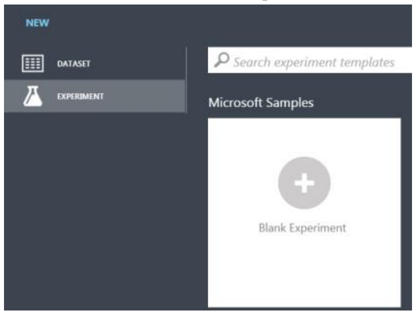
2 Manage Access
Grant or suspend access for users here

3 Manage Web Services
Start here to manage Web Services

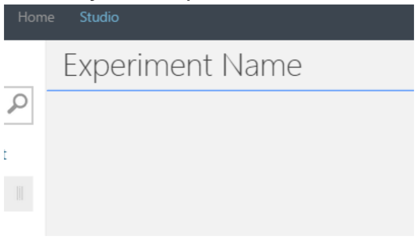
Exercise 4: Creating your first experiment.

Data Science is both an art and a science. It borrows terms from other disciplines, especially traditional sciences like chemistry. A project in data science is called an "experiment."

1. Create a new experiment
    a. Click on: **New → Experiment → Blank Experiment**



2. Name your experiment at the top.



3. You won't be able to save your experiment unless you have a module inside of it. For now, move onto the next lab. After that, you will be able to save your experiment.

Lab 2: Methods of Ingress and Egress with Azure Machine Learning Studio

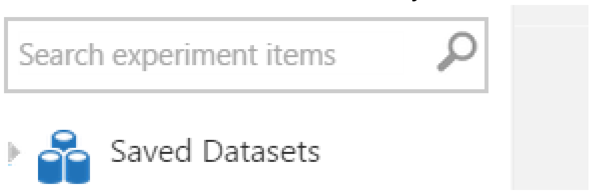
For now, **you cannot deleted saved datasets**. So think **very** carefully before naming your datasets. This may change later in future updates to Azure ML.

Exercise 1: Upload dataset to workspace from a local file

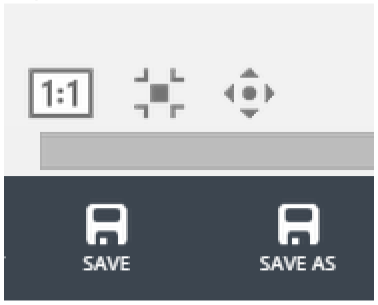
1. Visit: http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
```

   a. Notice how its comma delimited. That lets you know that it can be read as a CSV.
   b. Excel Files (xls) and delimited text files can be read as a CSV.
   c. Notice that this data does not have headers. We will define the headers later, as the model will require it.
2. Download and save the file as a CSV file. 'file-name.csv'.
3. Import the dataset into Azure ML Studio.
   a. Click on: **New → DATASET → FROM LOCAL FILE**
   b. Import as a new dataset.
   c. Please note that by default Azure ML ships with a dataset called "Iris Two Class Data". Do not confuse that with the data you just imported, so give it a unique name.
4. Go into any experiment and verify that the dataset has been imported.
   a. The data will be under a directory called "Saved Datasets" within any experiment.

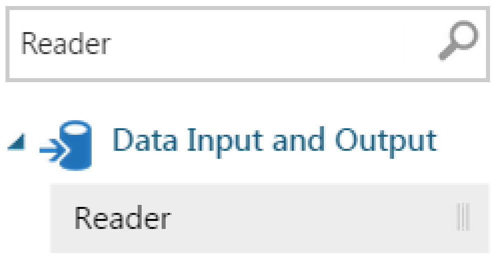


5. Now that your experiment has some modules in it, you are able to save your experiment. Save your experiment with "save as" on the menus at bottom.

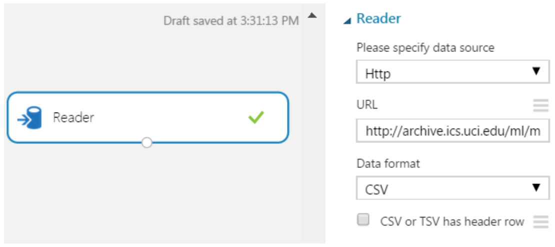


Exercise 2: Reading a data source through http

1. Drag and drop a Reader module from the menu on the left.



2. Specify Http as the data source.
3. In the URL field, enter http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data
4. Choose CSV from the Data Format drop down
5. Leave the 'CSV or TSV has header row' option unchecked



Reader

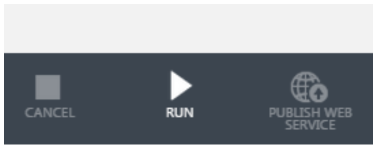
Please specify data source: Http

URL: http://archive.ics.uci.edu/ml/m

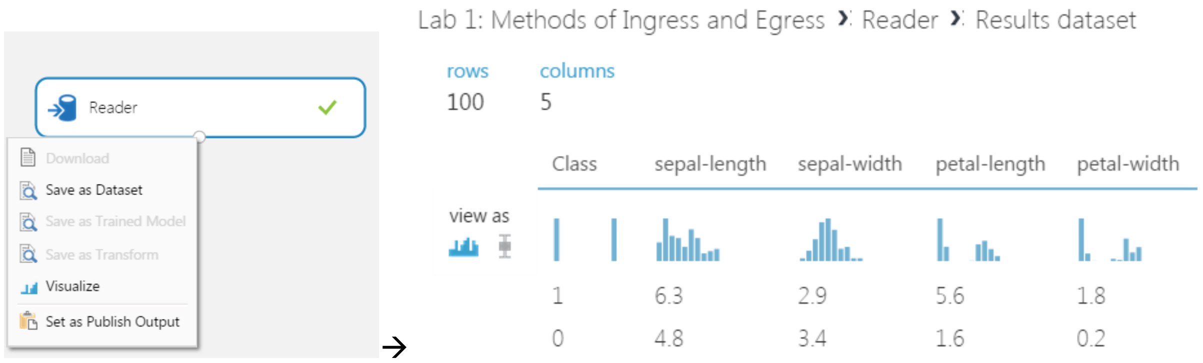
Data format: CSV

☐ CSV or TSV has header row

6. Leave Disable Upgrades unchecked (Any thoughts why this would matter?)
7. Run the experiment to execute the import and parse.



8. Preview the data by visualizing the output of the reader module. Right click the bottom middle node of the reader module to access the menu.

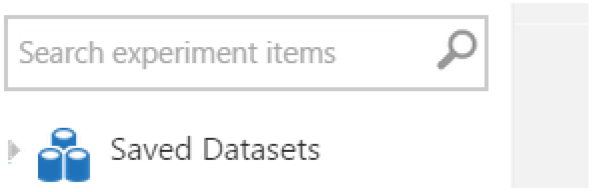


Lab 1: Methods of Ingress and Egress › Reader › Results dataset

rows 100 columns 5

Class	sepal-length	sepal-width	petal-length	petal-width
1	6.3	2.9	5.6	1.8
0	4.8	3.4	1.6	0.2

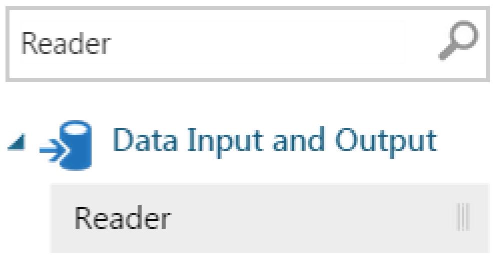
9. The data is not yet saved. Save it to the workspace by clicking "save as Dataset" in the reader's output node.
10. Go into any experiment and verify that the dataset has been imported.
    a. The data will be under a directory called "Saved Datasets" within any experiment.



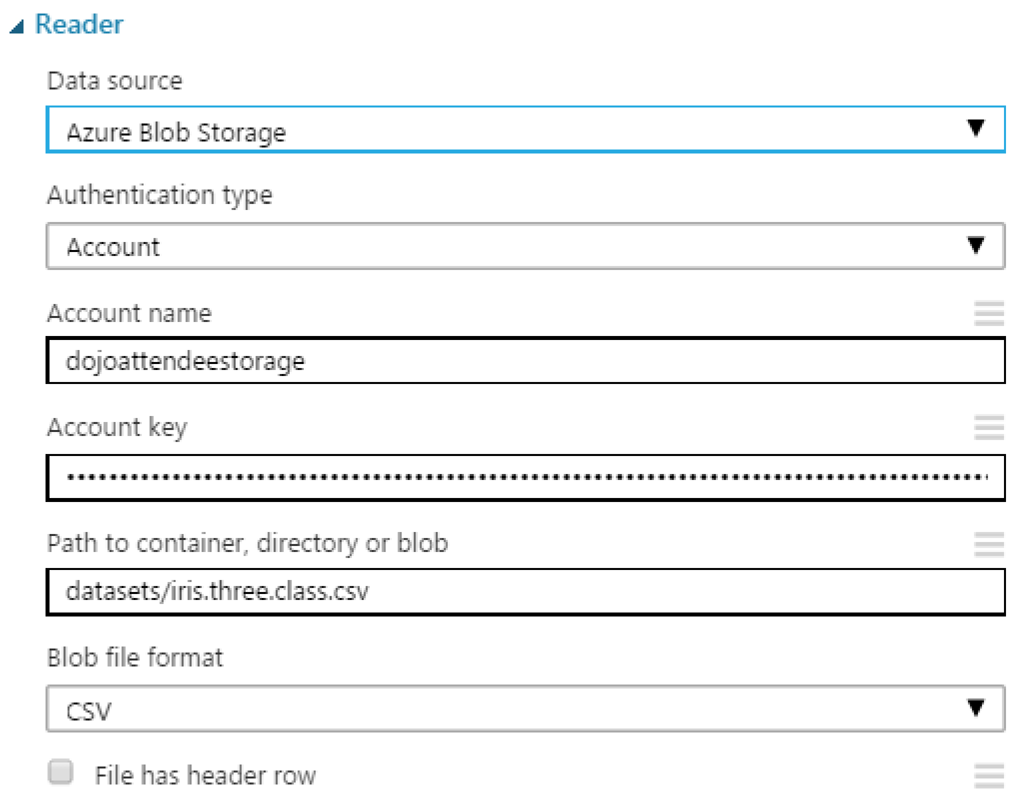
    b. Please note that by default Azure ML ships with a dataset called "Iris Two Class Data." Do not confuse that with the data you just imported.

Exercise 3: Reading a Data Set from Azure Blob Storage

1. Drag and drop a Reader module from the menu on left.



2. Input the sample blob storage account we have set up for you.
   a. **Data Source:** AzureBlobStorage
   b. **Authentication type:** Account
   c. **Account:** dojoattendeestorage
   d. **Account Key:** aKQOxU3As1BsS3yT2bhHkJ/icCICJPpL1tdWKxQ+tPBNk6DbykV4qd3HGlFPZN/3TdiUHuM/Quk9DPUeQu7M8A==
   e. **Path to container, directory:** datasets/iris.three.class.csv
      i. Note that dojoattendeestorage is the container name. Storage vaults have containers, and containers have blobs: **Storage>Container>Blob**. In this case it's: **dojoattendeestorage>datasets>iris.three.class.csv**
      ii. A blob is really just a file that's in the Azure Cloud itself. If you've done web development, this looks exactly like an FTP.
   f. **Blob file format:** CSV
   g. **File has header row:** Unchecked


Reader

Data source
Azure Blob Storage

Authentication type
Account

Account name
dojoattendeestorage

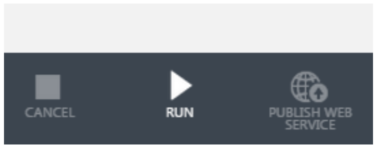
Account key
••

Path to container, directory or blob
datasets/iris.three.class.csv

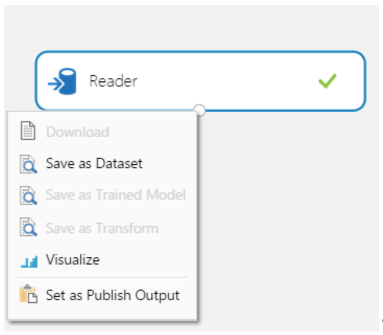
Blob file format
CSV

☐ File has header row

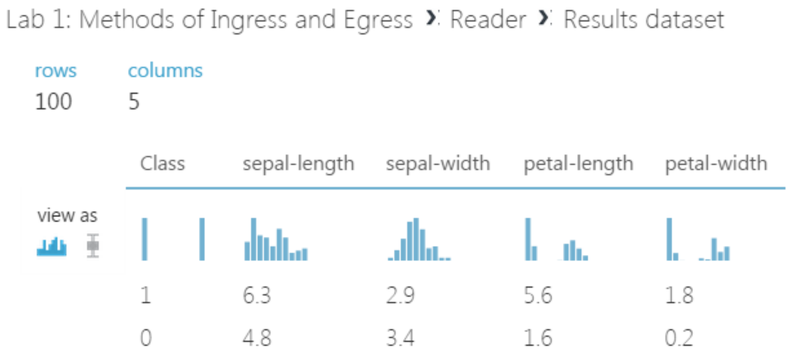
3. Run the experiment to execute the import and parse.



4. Preview the data by visualizing the output of the reader module. Right click the bottom middle node of the reader module to access the menu.



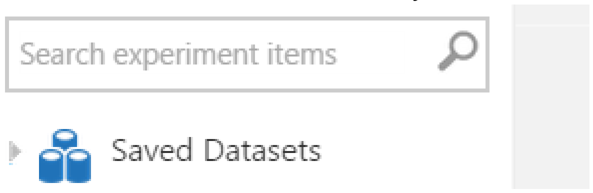
→


Lab 1: Methods of Ingress and Egress › Reader › Results dataset

rows	columns
100	5

view as	Class	sepal-length	sepal-width	petal-length	petal-width
	1	6.3	2.9	5.6	1.8
	0	4.8	3.4	1.6	0.2

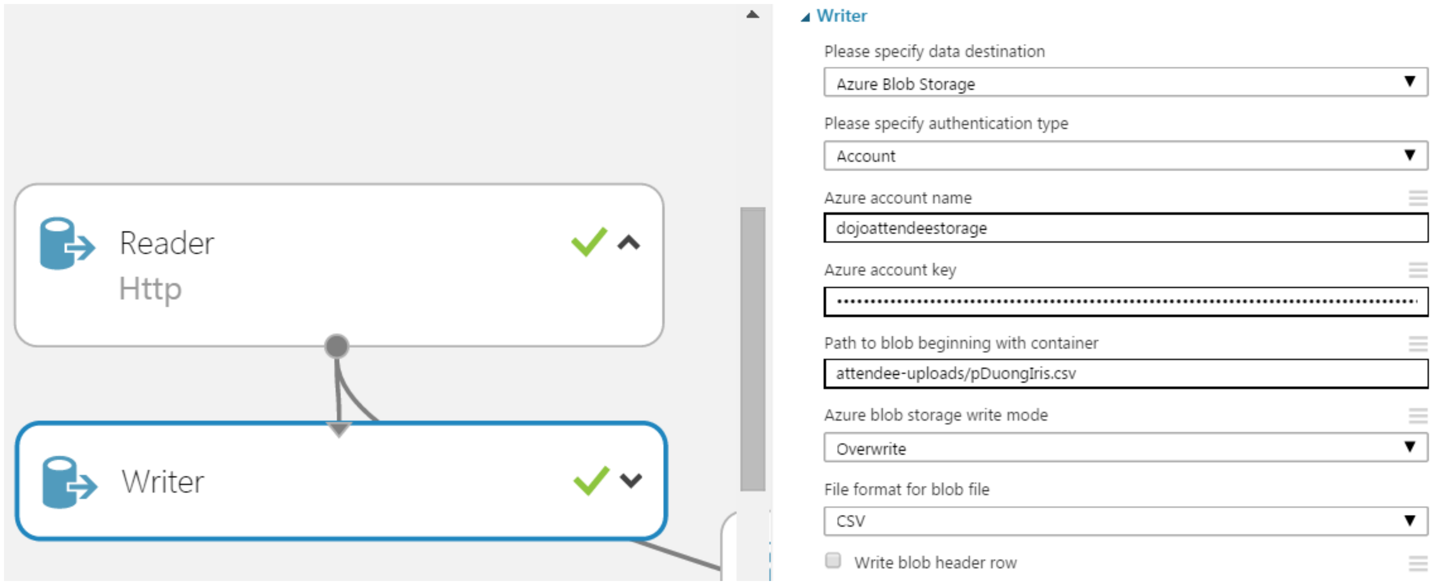
5. The data is not yet saved. Save it to the workspace by clicking "save as Dataset" in the reader's output node.
6. Go into any experiment and verify that the dataset has been imported.
   a. The data will be under a directory called "Saved Datasets" within any experiment.



   b. Please note that by default Azure ML ships with a dataset called "Iris Two Class Data". Do not confuse that with the data you just imported.

Exercise 4: Writing Datasets to an Azure Blob Storage

1. Drag any dataset into your workspace.
2. Drag a writer module in and connect the dataset to the writer.
3. Input the sample blob storage account we have set up for you.
    a. **Data Source:** AzureBlobStorage
    b. **Authentication type:** Account
    c. **Account:** dojoattendeestorage
    d. **Account Key:** aKQOxU3As1BsS3yT2bhHkJ/icCICJPpL1tdWKxQ+tPBNk6DbykV4qd3HGlFPZN/3TdiUHuM/Quk9DPUeQu7M8A==
    e. **Path to container, directory:** attendee-uploads/<file-name>.csv
        i. Normally you can name your file whatever you want. However since a lot of people will be writing to this blob, **do not name it iris.csv.** Name it a combination of your first initial, then your last name, then iris as one word. For example, if your name was John Smith, name it jSmithIris.csv. This will keep our azure storage account somewhat manageable and neater.
    f. **Azure Blob storage account write mode:** Overwrite
        i. Write mode of error will return an error if the filename already exists on the storage account.
    g. **Blob file format:** CSV
    h. **Write blob header row:** Unchecked



Writer

Please specify data destination: Azure Blob Storage

Please specify authentication type: Account

Azure account name: dojoattendeestorage

Azure account key: ••

Path to blob beginning with container: attendee-uploads/pDuongIris.csv

Azure blob storage write mode: Overwrite

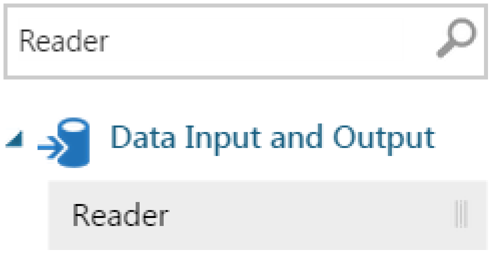
File format for blob file: CSV

☐ Write blob header row

Exercise 5: Reading Hive Tables into Azure ML

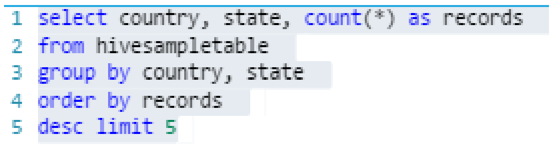
This lab will assume that you already have a Hadoop cluster setup. Please refer to the HDInsight lab on how to create a Hadoop cluster.

1. Drag and drop a Reader module from the menu on left.



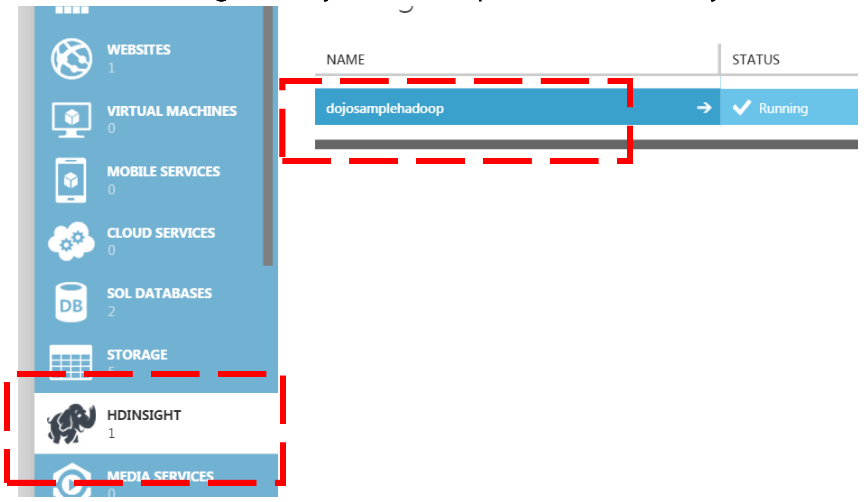
2. Populate the required fields using the values below. Open up a second window or tab to retrieve the other information.
   a. **Data Source:** HiveQuery
   b. **Hive database query:** Below is a sample query statement that you may perform. However, any of the query statements you made in the Hive lab will also work.

```
select country, state, count(*) as records
from hivesampletable
group by country, state
order by records
desc limit 5
```

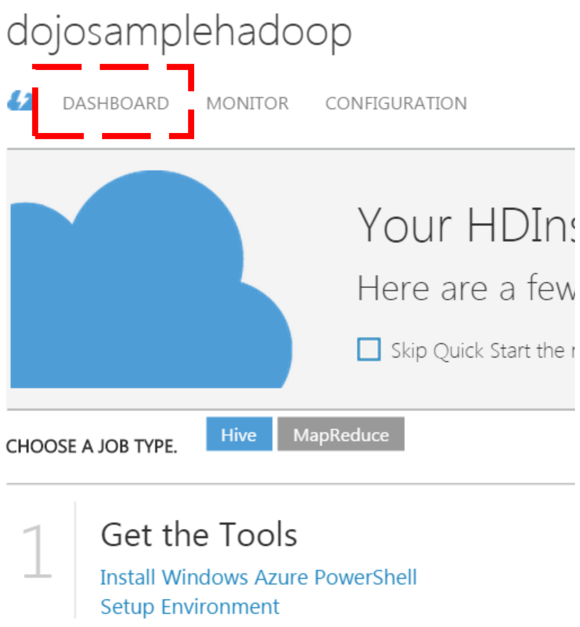


```
select country, state, count(*) as records
from hivesampletable
group by country, state
order by records
desc limit 5
```

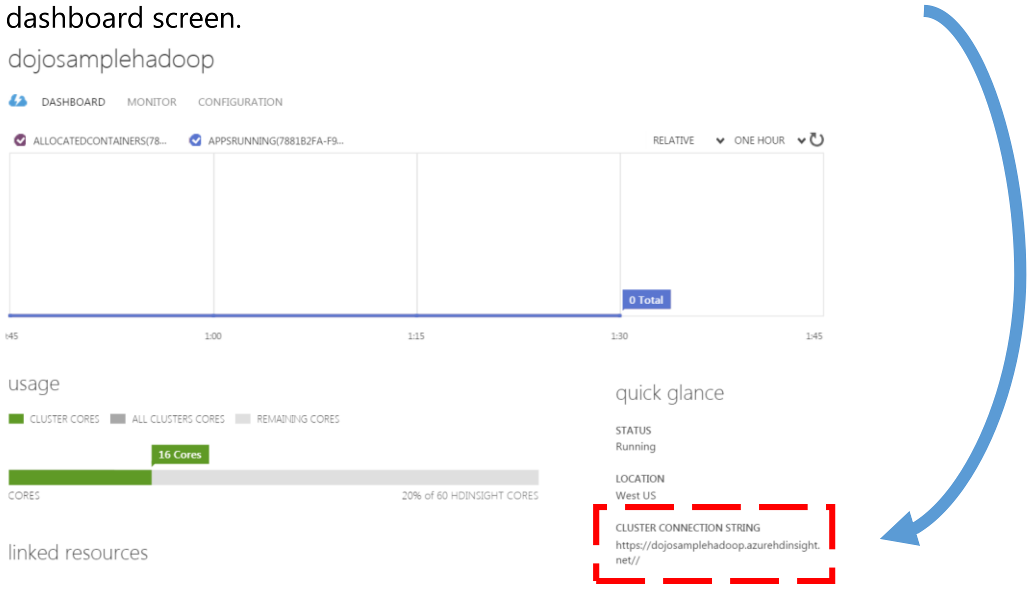
   c. **HCatalog server URI:**
      i. From your azure management portal (open this in a new window or tab): https://manage.windowsazure.com/
      ii. Click on HDInsight and your Hadoop Cluster to enter your Hadoop cluster dashboard.



iii. Click on “Dashboard.”



iv. Retrieve your cluster connection string, located on the bottom right hand corner of the dashboard screen.



STATUS
Running

LOCATION
West US

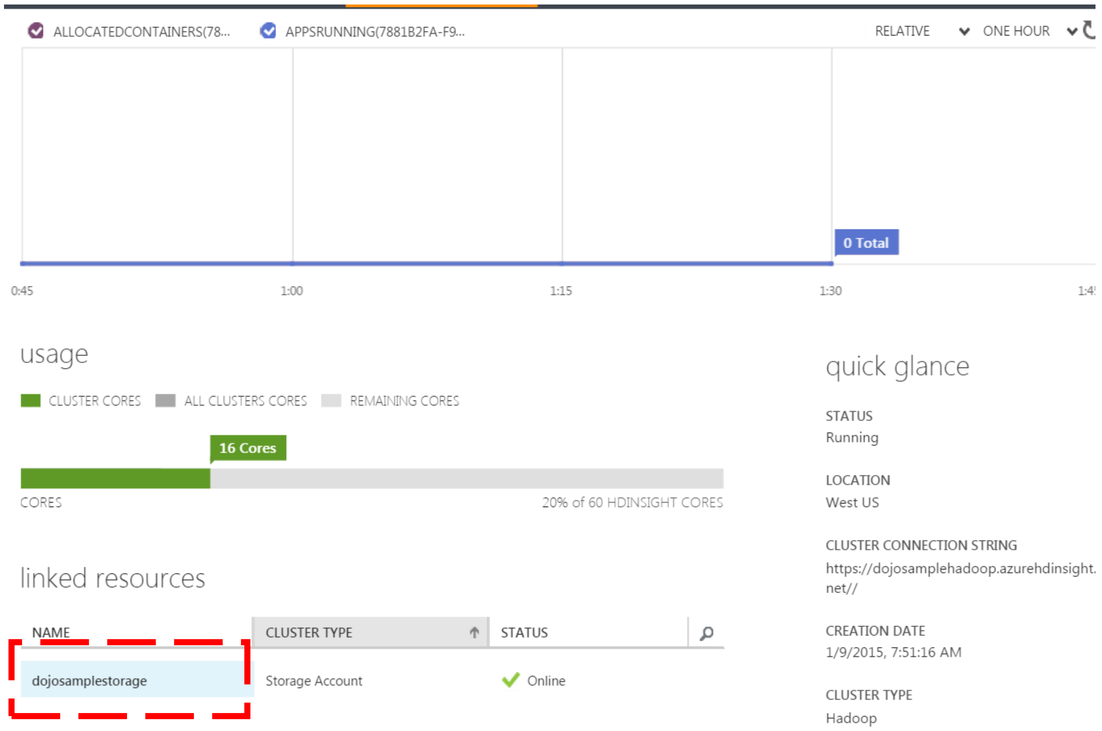
CLUSTER CONNECTION STRING
https://dojosamplehadoop.azurehdinsight.net//

In the above example it is ‘https://dojosamplehadoopcluster.azurehdinsight.net//’

v. Copy the cluster connection string into your Azure ML reader module’s HCatalog server URI section.

d. **Hadoop user account name:** admin
e. **Hadoop user account password:** The password you used when you provisioned your HDInsight Hadoop cluster.
f. **Location of output data:** Azure

g. **Azure storage account name:** To find this detail, simply scroll down on the Hadoop cluster dashboard page (the same place you got your Hadoop cluster connection string from). It will be under a table called "linked resources".



NAME	CLUSTER TYPE	STATUS
dojosamplestorage	Storage Account	✓ Online

quick glance

STATUS
Running

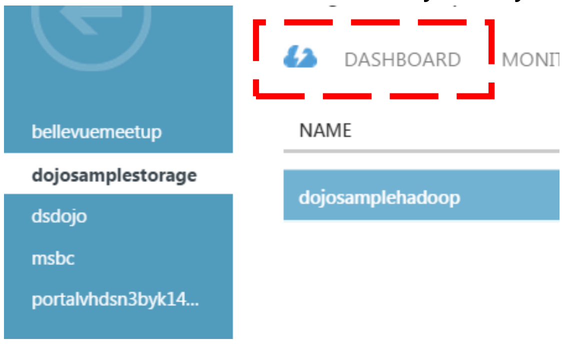
LOCATION
West US

CLUSTER CONNECTION STRING
https://dojosamplehadoop.azurehdinsight.net//

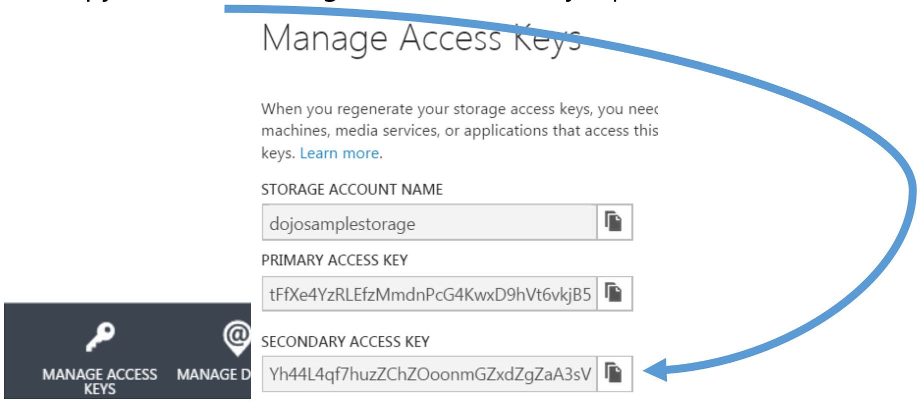
CREATION DATE
1/9/2015, 7:51:16 AM

CLUSTER TYPE
Hadoop

i. Click on the cluster and it'll take you to your storage page. Click on dashboard.



ii. Once inside, click "manage access keys" in the bottom middle of the page. You can click the copy button to the right of the read only input boxes.



Manage Access Keys

When you regenerate your storage access keys, you need machines, media services, or applications that access this keys. Learn more.

STORAGE ACCOUNT NAME
dojosamplestorage

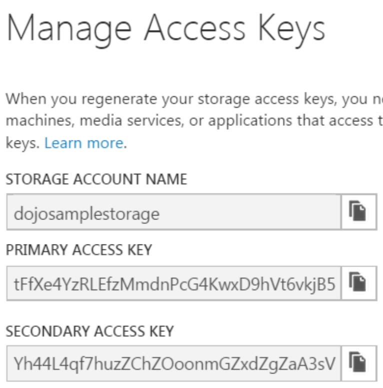
PRIMARY ACCESS KEY
tFfXe4YzRLEfzMmdnPcG4KwxD9hVt6vkjB5

SECONDARY ACCESS KEY
Yh44L4qf7huzZChZOoonmGZxdZgZaA3sV

iii. Copy the storage name into your Azure ML reader module.

h. **Azure storage key:**

i. You can copy either of the access keys given in the manage access keys window. Ideally the primary access key is the key used for important web services and is highly guarded to prevent changes (or it could break live web services that depend on this storage account, such as a Hadoop cluster). The secondary key is the one given out for others to use and can be regenerated often.


Manage Access Keys

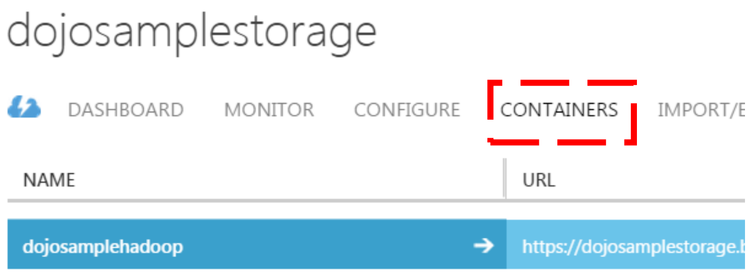
When you regenerate your storage access keys, you neec
machines, media services, or applications that access this
keys. Learn more.

STORAGE ACCOUNT NAME
dojosamplestorage

PRIMARY ACCESS KEY
tFfXe4YzRLEfzMmdnPcG4KwxD9hVt6vkjB5

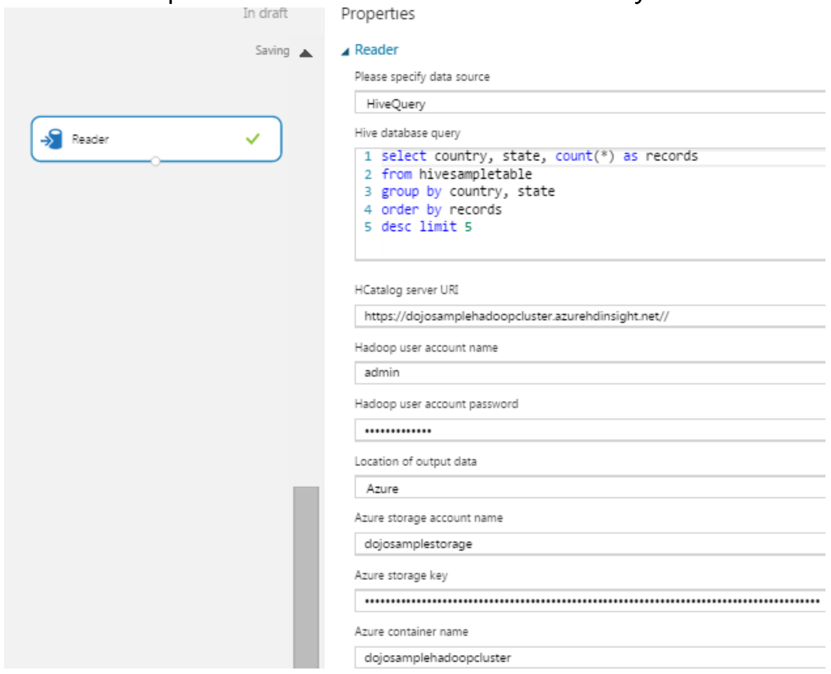
SECONDARY ACCESS KEY
Yh44L4qf7huzZChZOoonmGZxdZgZaA3sV

i. **Azure Container:** Your container name should actually be the same name as your Hadoop cluster. Just in case, you can check it by clicking on “containers” at the top.


dojosamplestorage

NAME	URL
dojosamplehadoop	https://dojosamplestorage.t

j. This is a sample of what it should look like when you’re done.



Properties

Reader

Please specify data source
HiveQuery

Hive database query

```
select country, state, count(*) as records
from hivesampletable
group by country, state
order by records
desc limit 5
```

HCatalog server URI
https://dojosamplehadoopcluster.azurehdinsight.net//

Hadoop user account name
admin

Hadoop user account password
•••••••••••••

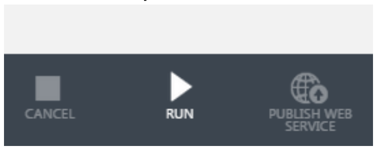
Location of output data
Azure

Azure storage account name
dojosamplestorage

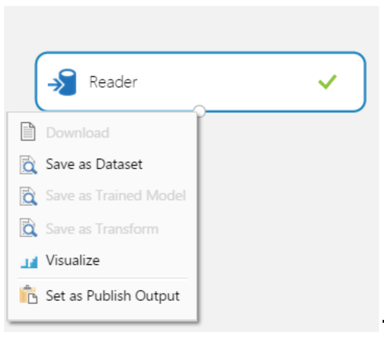
Azure storage key
••

Azure container name
dojosamplehadoopcluster

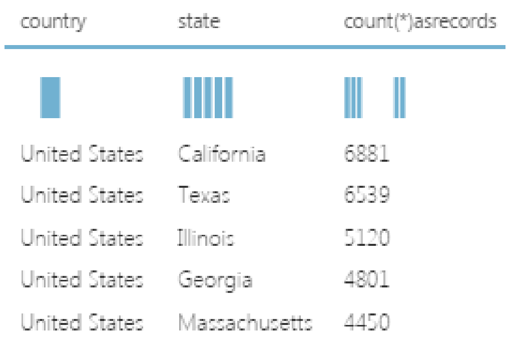
3. Run the experiment to execute the import and parse.



4. Preview the data by visualizing the output of the reader module. Right click the bottom middle node of the reader module to access the menu.



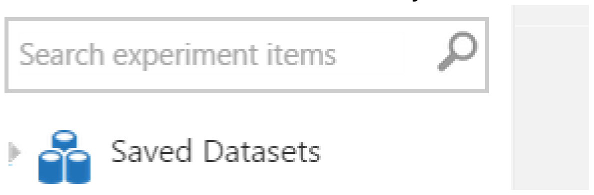
→



country	state	count(*)asrecords
United States	California	6881
United States	Texas	6539
United States	Illinois	5120
United States	Georgia	4801
United States	Massachusetts	4450

The data is not yet saved to Azure ML. Save it to the workspace by clicking "Save as Dataset" in the reader's output node.

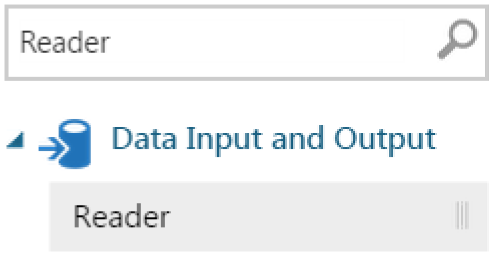
5. Go into any experiment and verify that the dataset has been imported.
    a. The data will be under a directory called "Saved Datasets" within any experiment.



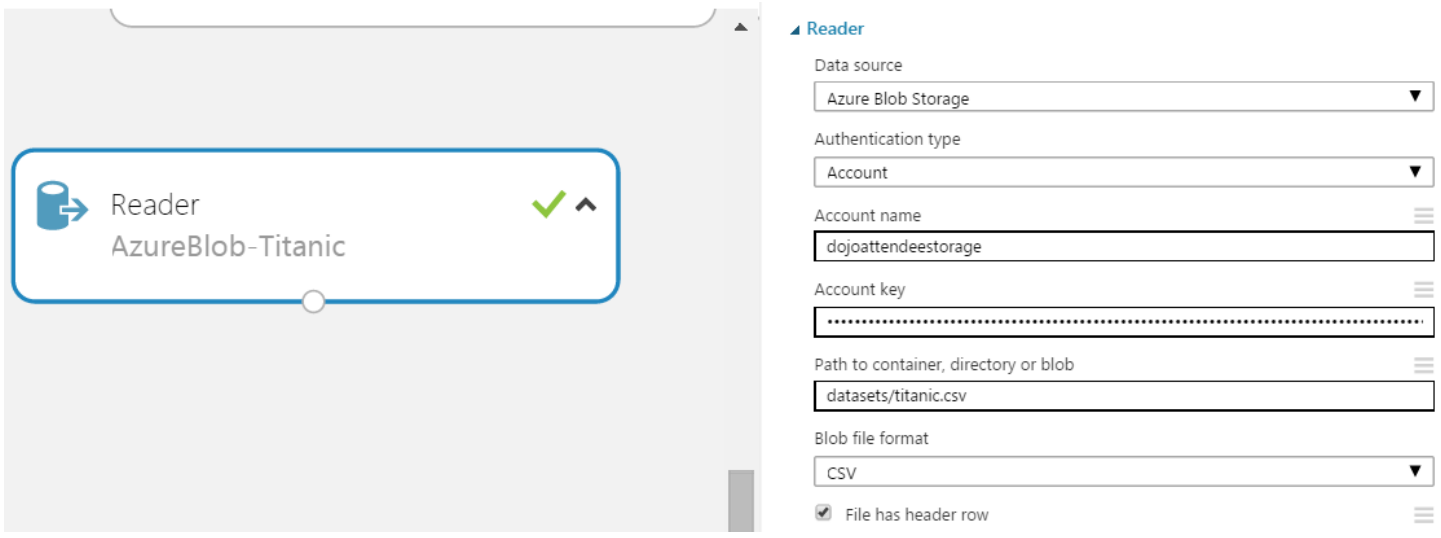
Lab 3: Visualizing, Exploring, Cleaning, and Manipulating Data

Exercise 1: Obtain the Titanic sample data

1. Drag and drop a Reader module from the menu on left.



2. Input the sample blob storage account we have set up for you.
   a. **Data Source:** AzureBlobStorage
   b. **Authentication Type:** Account
   c. **Account:** dojoattendeestorage
   d. **Account Key:** 7zXrsCGmM5LKIgZG6a4UU6LPPzVab70JdxuVSwXMRFsfQQW48RVh6GERv1/fsR5DBSAKWCXhyGdngXI6pyA4SQ==
   e. **Path to container, directory:** datasets/titanic.csv
   f. **Blob file format:** CSV
   g. **File has header row:** Checked



Reader

Data source: Azure Blob Storage

Authentication type: Account

Account name: dojoattendeestorage

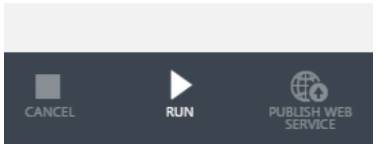
Account key: ••

Path to container, directory or blob: datasets/titanic.csv

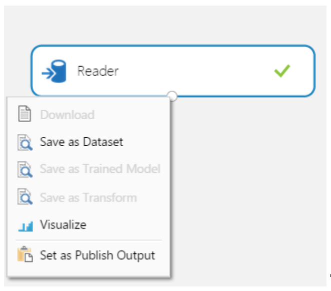
Blob file format: CSV

☑ File has header row

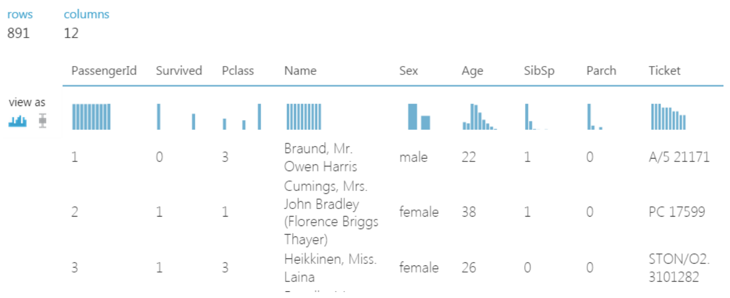
3. Run the experiment to execute the import and parse.



4. Preview the data by visualizing the output of the reader module. Right click the bottom middle node of the reader module to access the menu.



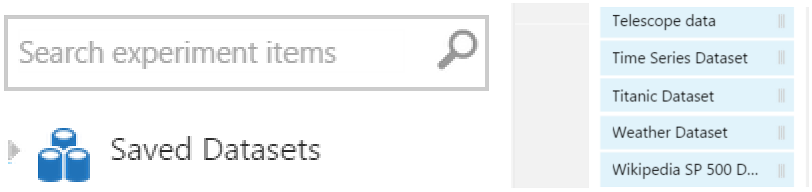
→



rows 891   columns 12

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282

5. The data is not yet saved. Save it to the workspace by clicking “Save as Dataset” in the reader’s output node.
6. Go into any experiment and verify that the dataset has been imported.
    a. The data will be under a directory called “Saved Datasets” within any experiment



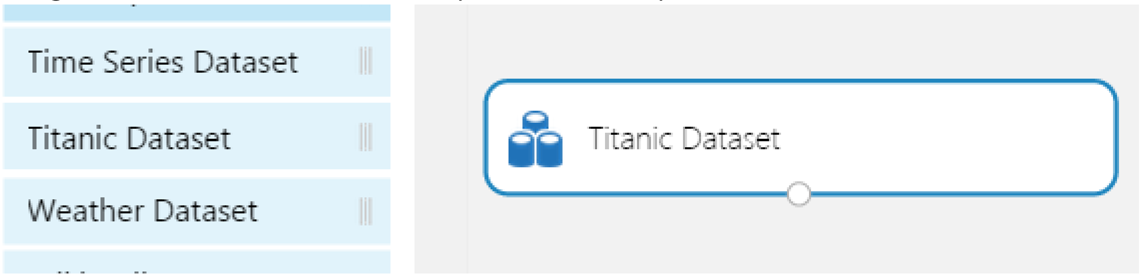
Titanic Dataset Key:

```
VARIABLE DESCRIPTIONS:
survival        Survival
                (0 = No; 1 = Yes)
pclass          Passenger Class
                (1 = 1st; 2 = 2nd; 3 = 3rd)
name            Name
sex             Sex
age             Age
sibsp           Number of Siblings/Spouses Aboard
parch           Number of Parents/Children Aboard
ticket          Ticket Number
fare            Passenger Fare
cabin           Cabin
embarked        Port of Embarkation
                (C = Cherbourg; Q = Queenstown; S = Southampton)

SPECIAL NOTES:
Pclass is a proxy for socio-economic status (SES)
 1st ~ Upper; 2nd ~ Middle; 3rd ~ Lower

Age is in Years; Fractional if Age less than One (1)
 If the Age is Estimated, it is in the form xx.5

With respect to the family relation variables (i.e. sibsp and parch)
some relations were ignored.  The following are the definitions used
for sibsp and parch.

Sibling:  Brother, Sister, Stepbrother, or Stepsister of Passenger Aboard Titanic
Spouse:   Husband or Wife of Passenger Aboard Titanic (Mistresses and Fiances Ignored)
Parent:   Mother or Father of Passenger Aboard Titanic
Child:    Son, Daughter, Stepson, or Stepdaughter of Passenger Aboard Titanic

Other family relatives excluded from this study include cousins,
nephews/nieces, aunts/uncles, and in-laws.  Some children travelled
only with a nanny, therefore parch=0 for them.  As well, some
travelled with very close friends or neighbors in a village, however,
the definitions do not support such relations.
```

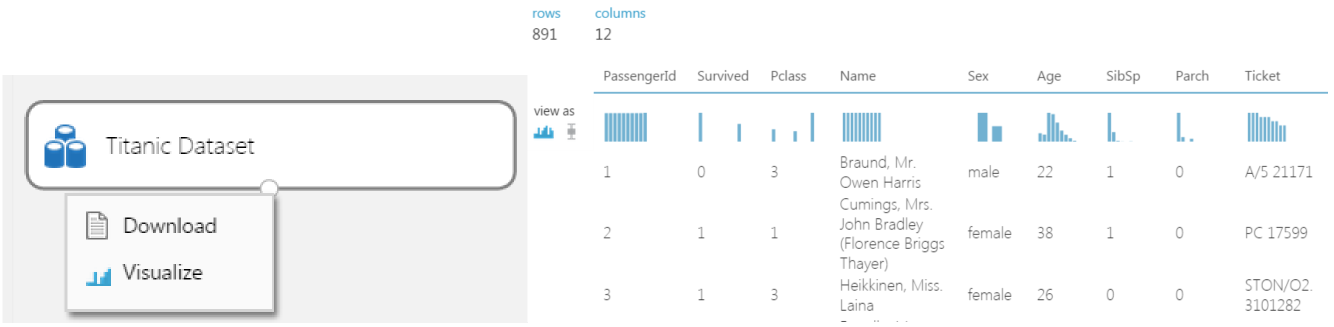
Exercise 2: Casting Columns

This data contains categorical data types. However, we must tell Azure which of those columns are categorical so that our models will not treat them as sequential numbers.

1. Drag the Titanic dataset into the experiment workspace.



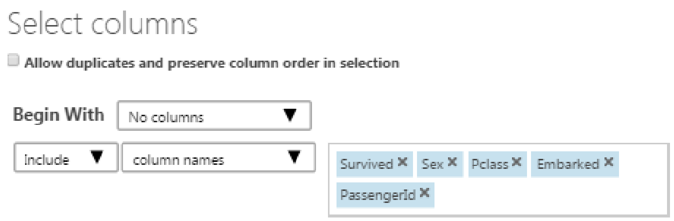
2. Verify that it's the same looking data as below by visualizing the Titanic dataset.



rows 891 columns 12

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282

3. Cast categorical values to categorical.
   a. Drag in the metadata editor module.
   b. Launch the column selector.
   c. Begin with "no columns" and "include" "column names"
      i. Add "survived," "sex," "pclass," "embarked," and "PassengerID"



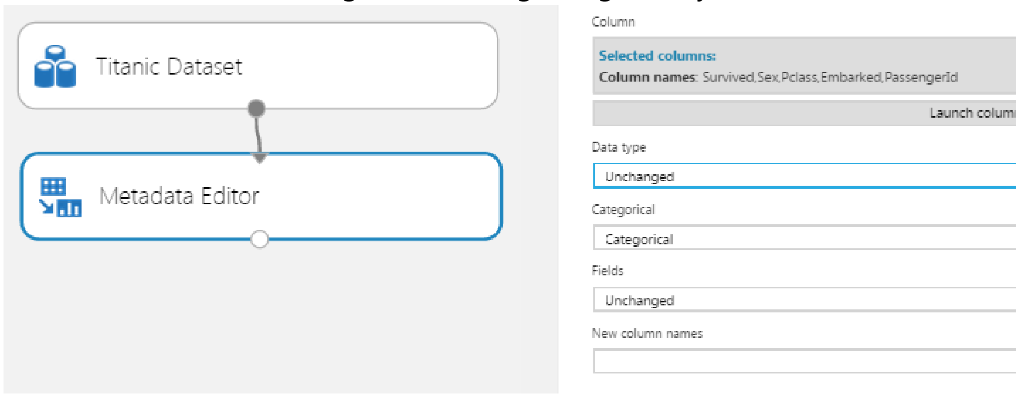
Select columns

Allow duplicates and preserve column order in selection

Begin With: No columns

Include: column names: Survived, Sex, Pclass, Embarked, PassengerId

   d. Set categorical to "categorical"
   e. Label the module as "Categorical Casting" for good style.



Column

Selected columns:
Column names: Survived,Sex,Pclass,Embarked,PassengerId

Data type: Unchanged

Categorical: Categorical

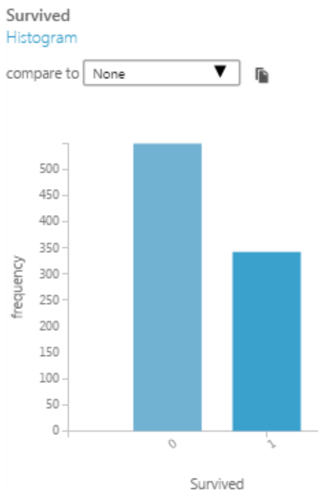
Fields: Unchanged

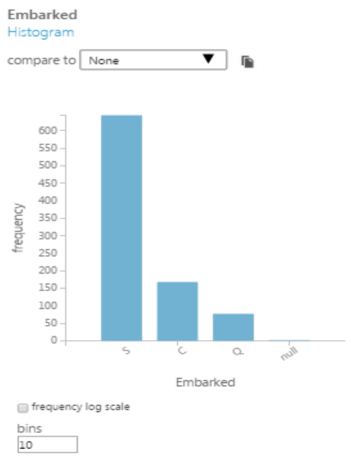
New column names:

Exercise 3: Data Visualization & Exploration

1. Histograms
   a. Click on the survived column with "view as" set to the picture of a histogram.

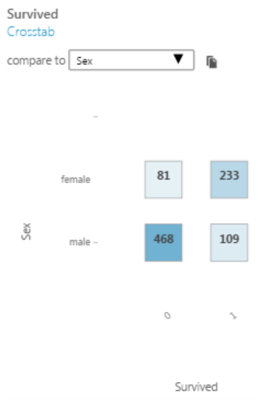
   b. A menu on the right will pop up. Expand the visualizations drop down. A histogram will now appear.
   c. What is the distribution of survived vs deceased? Did more people survive or perish?
   d. Where did people come from? Select the "embarked" column.


Survived
Histogram
compare to None

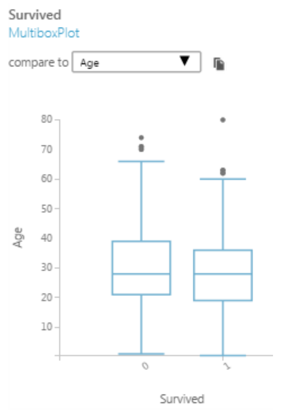

Embarked
Histogram
compare to None

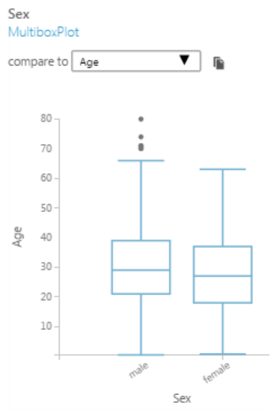
frequency log scale
bins 10

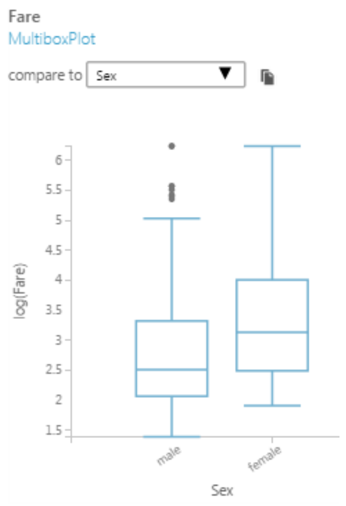
2. Comparison Visualizations
   a. Did gender play any part in survival? Select the "survived" column to view the histogram of survival, and set the histogram "compare to" to "sex". You will notice that a disproportionate amount of males died.
   b. Is there a relationship between age and survival? Set "compare to" to "age".
   c. Compare "Age" to "Sex" to see the distribution of males and female age groups.
   d. Compare "Fare" to "Sex". Make sure **"log scale"** is checked.


Survived
Crosstab
compare to Sex

Sex	0	1
female	81	233
male	468	109


Survived
MultiboxPlot
compare to Age

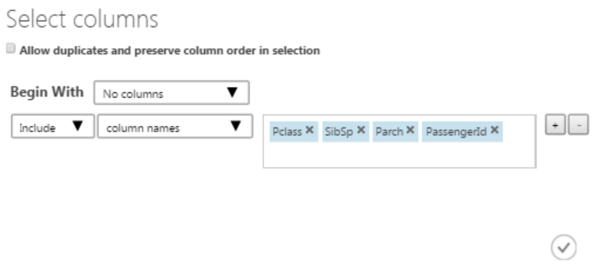

Sex
MultiboxPlot
compare to Age


Fare
MultiboxPlot
compare to Sex

Exercise 4: Renaming Your Columns

Let's rename some undescriptive column names: "Pclass", "SibSp", and "Parch"

1. Drag in a **Project Columns Module** under **Data Transformation > Manipulation**
   a. Launch the column selector and select "PassengerID," "Pclass," SibSP," and "Parch"

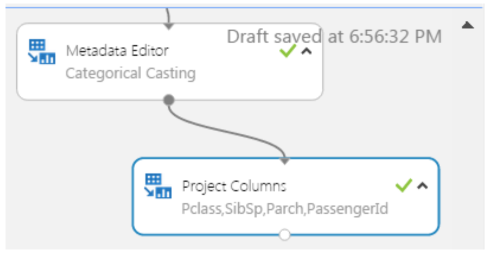


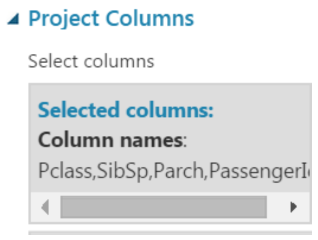
Select columns

Allow duplicates and preserve column order in selection

Begin With: No columns

Include | column names | Pclass ✕ SibSp ✕ Parch ✕ PassengerId ✕



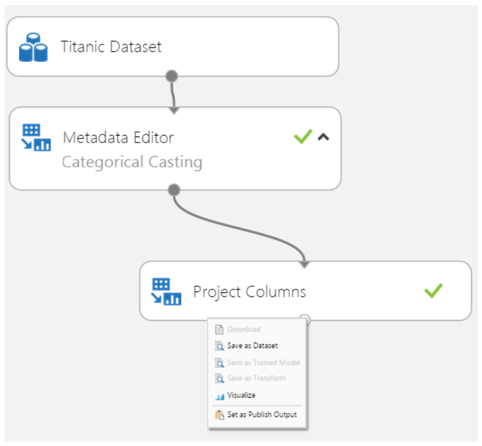


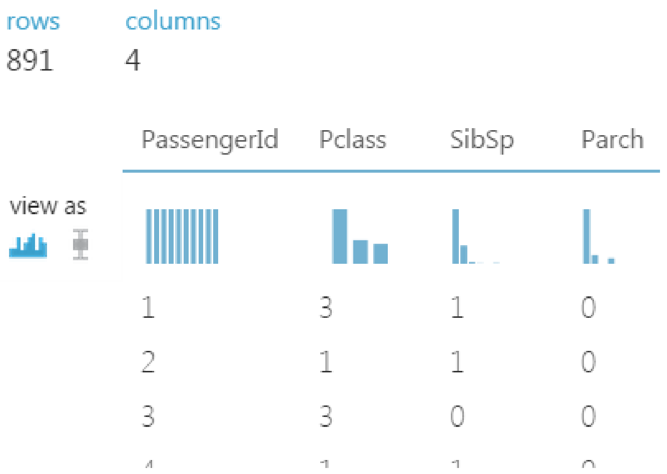
Project Columns

Select columns

Selected columns:
Column names:
Pclass,SibSp,Parch,PassengerI

   b. Run then visualize the output of the project columns module. This is what it should look like:



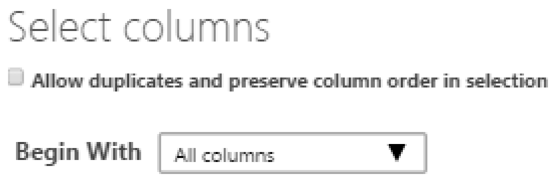


rows 891 columns 4

view as

PassengerId	Pclass	SibSp	Parch
1	3	1	0
2	1	1	0
3	3	0	0

2. Drag in another metadata editor, and connect it to the project columns module.
   a. Launch the column selector.
   b. Begin with: "All Columns"
   c. Directly below "begin with," there is a "+" and "-" button. Remove the extra parameter by clicking the minus button. Submit the popup window.



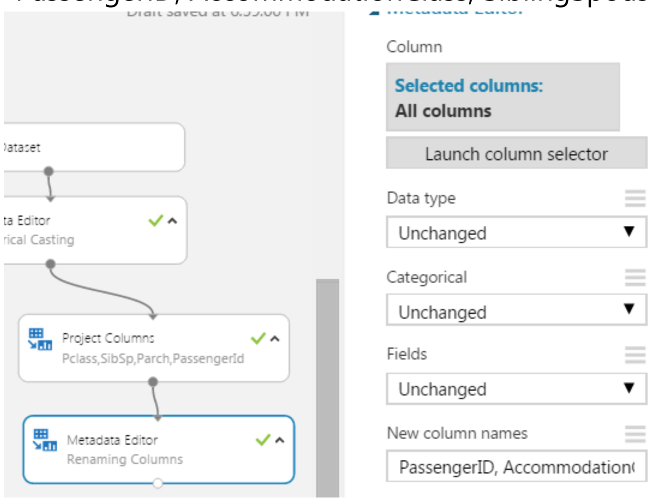
Select columns

Allow duplicates and preserve column order in selection

Begin With: All columns

   d. Leave "data type," "categorical," and "fields," to "unchanged"
   e. Under new column names, list IN ORDER what the new column names will be.

i. “PassengerID, AccommodationClass, SiblingSpouse, ParentChild”



Column

Selected columns:
All columns

Launch column selector

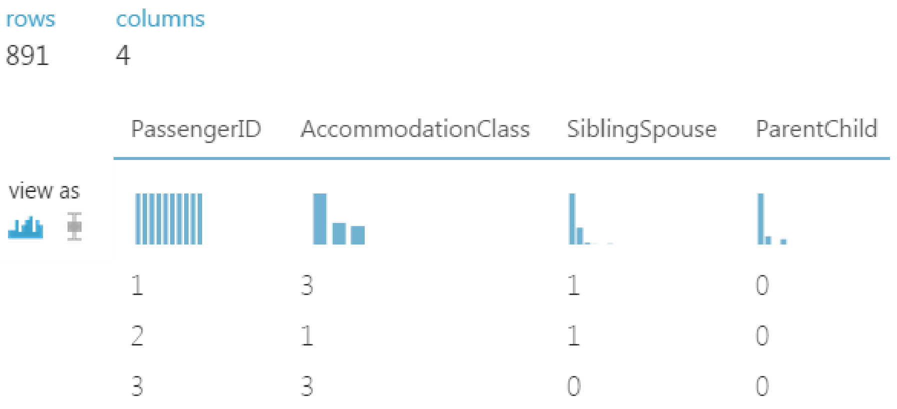
Data type: Unchanged

Categorical: Unchanged

Fields: Unchanged

New column names: PassengerID, Accommodation(

ii. Visualize the output of the metadata editor to verify that the column names changed.



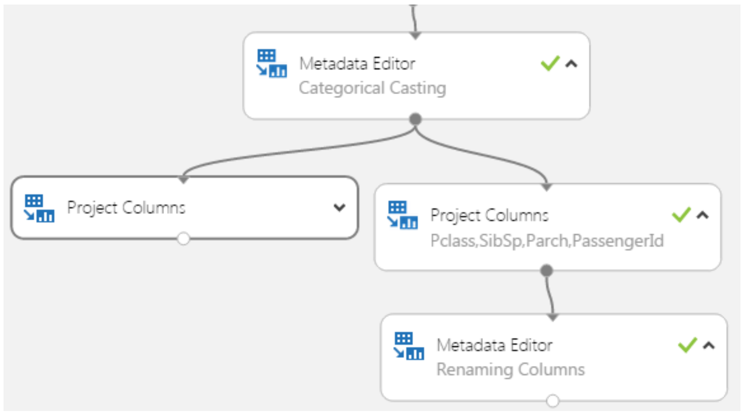
rows	columns
891	4

PassengerID	AccommodationClass	SiblingSpouse	ParentChild
1	3	1	0
2	1	1	0
3	3	0	0

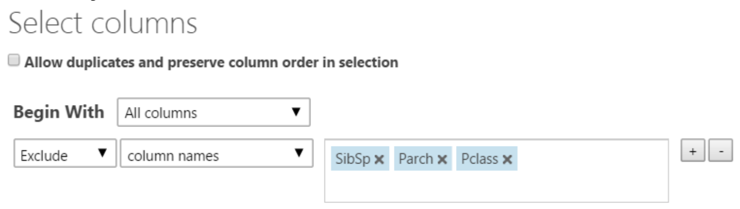
Exercise 5: Joining Tables

From the last step, we now have an isolated version of the other table. Now, we must rejoin them together.

1. Remove the columns that were renamed with the previous table projection.
    a. Drag in another project columns module and connect it to the metadata editor that performed the categorical casting. There should now be two project columns modules side by side feeding off the same metadata editor.



    b. Launch the column selector in the project columns module.
        i. Begin With: "All Columns"
        ii. Change the "include" to "exclude"
        iii. Choose to remove SibSp,Parch,Pclass. **Do not remove PassengerID**, we will need that for the join.

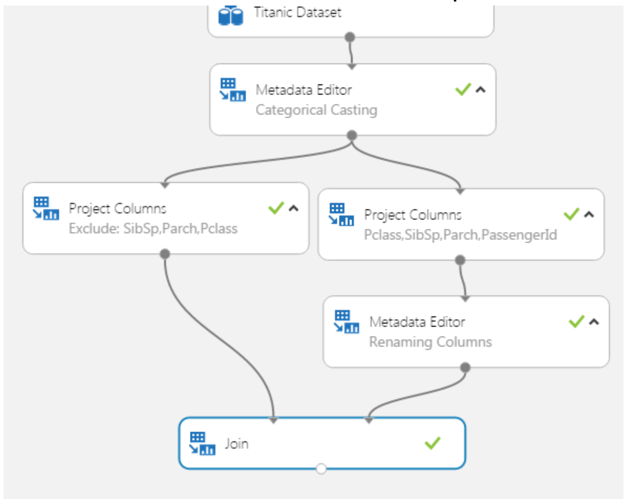

Select columns

Allow duplicates and preserve column order in selection

Begin With: All columns

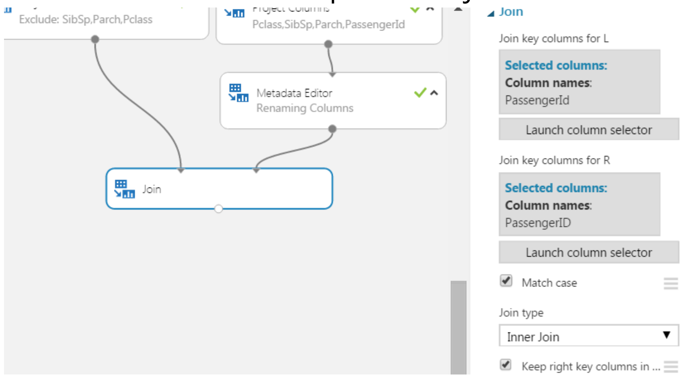
Exclude | column names | SibSp ✕ Parch ✕ Pclass ✕

2. Join the two tables together.
    a. Drag in a Join Module from **Data Transformation > Manipulation**
    b. Connect the project columns module on the left to the metadata editor on the right (the metadata editor that renamed the previous 3 columns).



    c. Join key columns for L > Launch Column Selector > PassengerID
    d. Join key columns for R > Launch Column Selector > PassengerID

e. Uncheck the box for “Keep right key columns in”. This will remove the extra PassengerID as a result of the join.
f. Run and visualize the output of the join module.



Join

Join key columns for L

Selected columns:
Column names:
PassengerId

Launch column selector

Join key columns for R

Selected columns:
Column names:
PassengerID

Launch column selector

✓ Match case

Join type

Inner Join

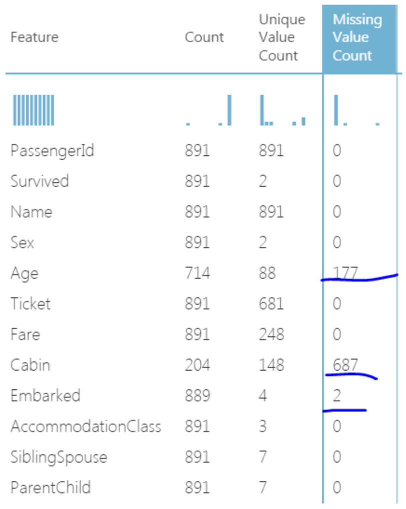
✓ Keep right key columns in ...

g. Verify that the new table has joined together properly on PassengerID.

Exercise 6: Descriptive Statistics and Scrub Missing Values

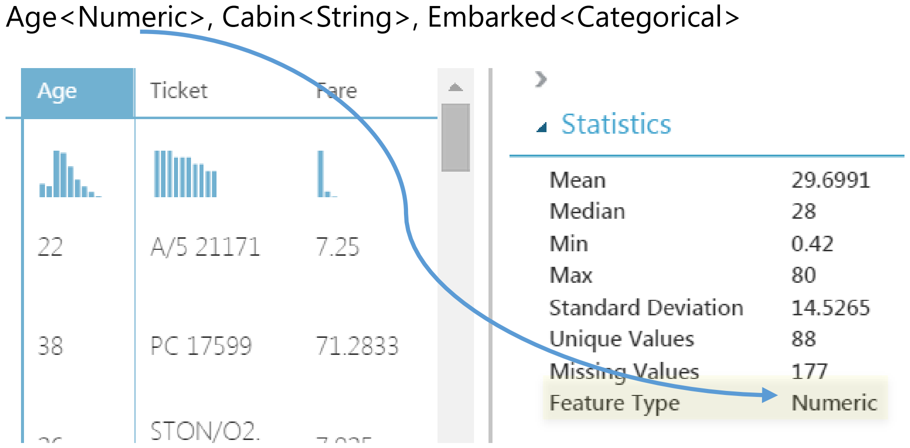
Before we can scrub missing values, we must find out where the missing values are and how to treat them. We will use the descriptive statistics module to help identify such columns.

1. Drag in a **Descriptive Statistics** module under **Statistical Functions.**
   a. Connect it to the join module. Run the experiment and visualize the descriptive statistics output.
   b. Examining some summary information about our data:
      i. What was the mean age?
      ii. How old was the oldest person onboard?
      iii. Who was the youngest?
         1. How do you interpret the youngest age?
      iv. What was the lowest price someone paid for a fare? The highest? Median fare price?
2. Identify columns that hold little to no data mining value. These will be dropped later.
   a. Passenger ID is only a primary key and holds no value outside of a database, table joins, or unique identification marker.
   b. Name also does not hold value to us in this context. You can use name to predict gender, but there are no missing values of gender.
   c. Ticket number also is an identifier which does not hold much value outside of identification.
3. Identify which columns have missing values.
   a. Age 177, Cabin 687, Embarked 2



Feature	Count	Unique Value Count	Missing Value Count
PassengerId	891	891	0
Survived	891	2	0
Name	891	891	0
Sex	891	2	0
Age	714	88	177
Ticket	891	681	0
Fare	891	248	0
Cabin	204	148	687
Embarked	889	4	2
AccommodationClass	891	3	0
SiblingSpouse	891	7	0
ParentChild	891	7	0

4. Identify the "Feature Type" of each column that contains missing values. Visualize the join output, and click on each individual column that has missing values to check their "Feature Type".
   a. Age<Numeric>, Cabin<String>, Embarked<Categorical>

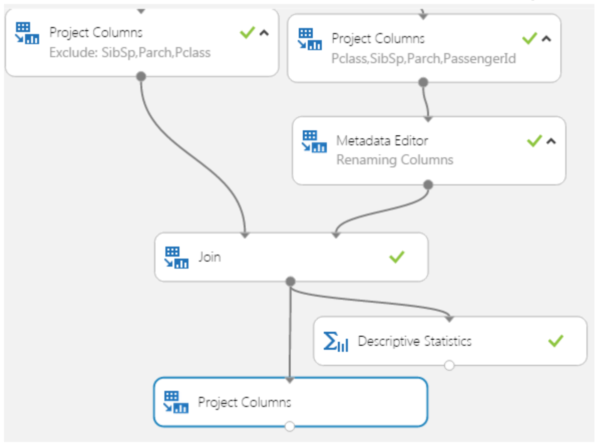


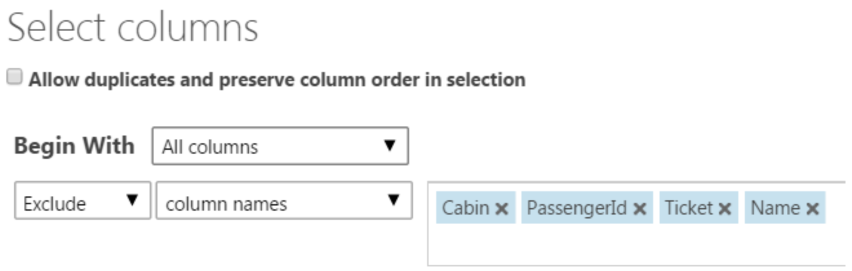
Age	Ticket	Fare
22	A/5 21171	7.25
38	PC 17599	71.2833
	STON/O2.	

Statistics

Mean	29.6991
Median	28
Min	0.42
Max	80
Standard Deviation	14.5265
Unique Values	88
Missing Values	177
Feature Type	Numeric

5. Perform an analysis on the missing values and develop a scrub strategy.
    a. Since AGE has 177 missing values, dropping the column would result in a huge loss of data. Age is also a numeric value. We can easily replace it with the "median" value and not much information loss will be present to the data. Age -> Replace with Median.
    b. Cabin is a string and is also missing 687 values. It may be best to just drop the cabin column moving forward since there are so many missing values. Cabin -> drop column.
    c. Embarked only has 2 missing values out of a total of 891 rows, which is not much. Dropping 2 rows to remove the missing values under embarked would not hurt the data very much.
6. Consider the **order of operations** used to perform the scrub. **THIS MATTERS A LOT.**
    a. Drop the cabin column first since it'll make for a smaller table moving forward and less iteration and processing for the next 2 scrubs.
    b. Replace missing age values with the median.
    c. **ALWAYS PERFORM ROW DROPS LAST,** not just on Azure ML but any data mining software or platform. Drop the 2 missing rows of embarked.
7. Drop the columns that hold little value in data mining.
    a. Drop Cabin, Name, PassengerID, and Ticket columns
    b. Use a project columns module and connect it to the output of the Join module.
        i. Begin With: "All Columns"
        ii. Change "include" to "exclude"
        iii. Select **Cabin, Ticket, Name, and Passenger** as columns to drop.



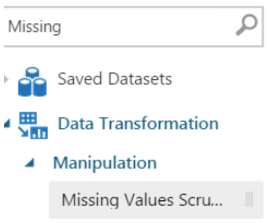

Select columns

☐ Allow duplicates and preserve column order in selection

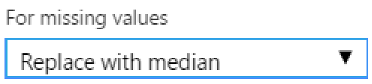
Begin With: All columns

Exclude — column names: Cabin ✕ PassengerId ✕ Ticket ✕ Name ✕

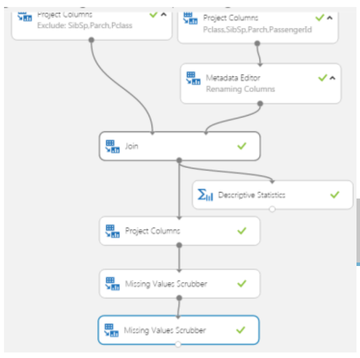
8. Replace missing values of age with the median.
    a. Drag in a missing Values Scrubber under **Data Transformation > Manipulation**

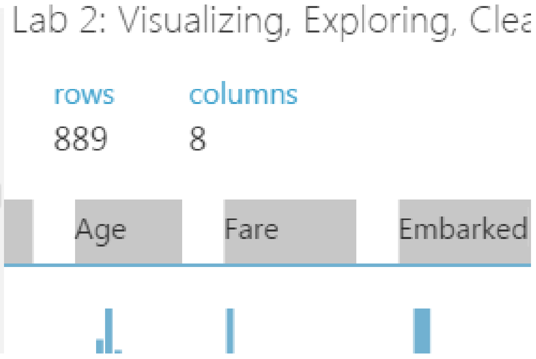


    b. Connect the scrubber module to the project columns module output.
    c. “For Missing Values” set to “Replace with Median”. This will replace all numeric missing values to the median value of the corresponding column.
        i. What will the module replace all the missing values with? Hint: look for the answer in the descriptive statistics module.
        ii. Why would it be poor practice to attach the Descriptive Statistics module AFTER this step? What would happen to our summary statistics? How would it affect our visualizations?


For missing values
Replace with median

9. Drop the remaining rows of the missing values.
    a. Drag in another missing value scrubber and connect it to the output of the previous missing value scrubber.
    b. “For Missing Values” set to “Remove entire row”
    c. Visualize the output. There should now be 889 rows, from the original 891 rows. Verify that there are now only 8 columns.

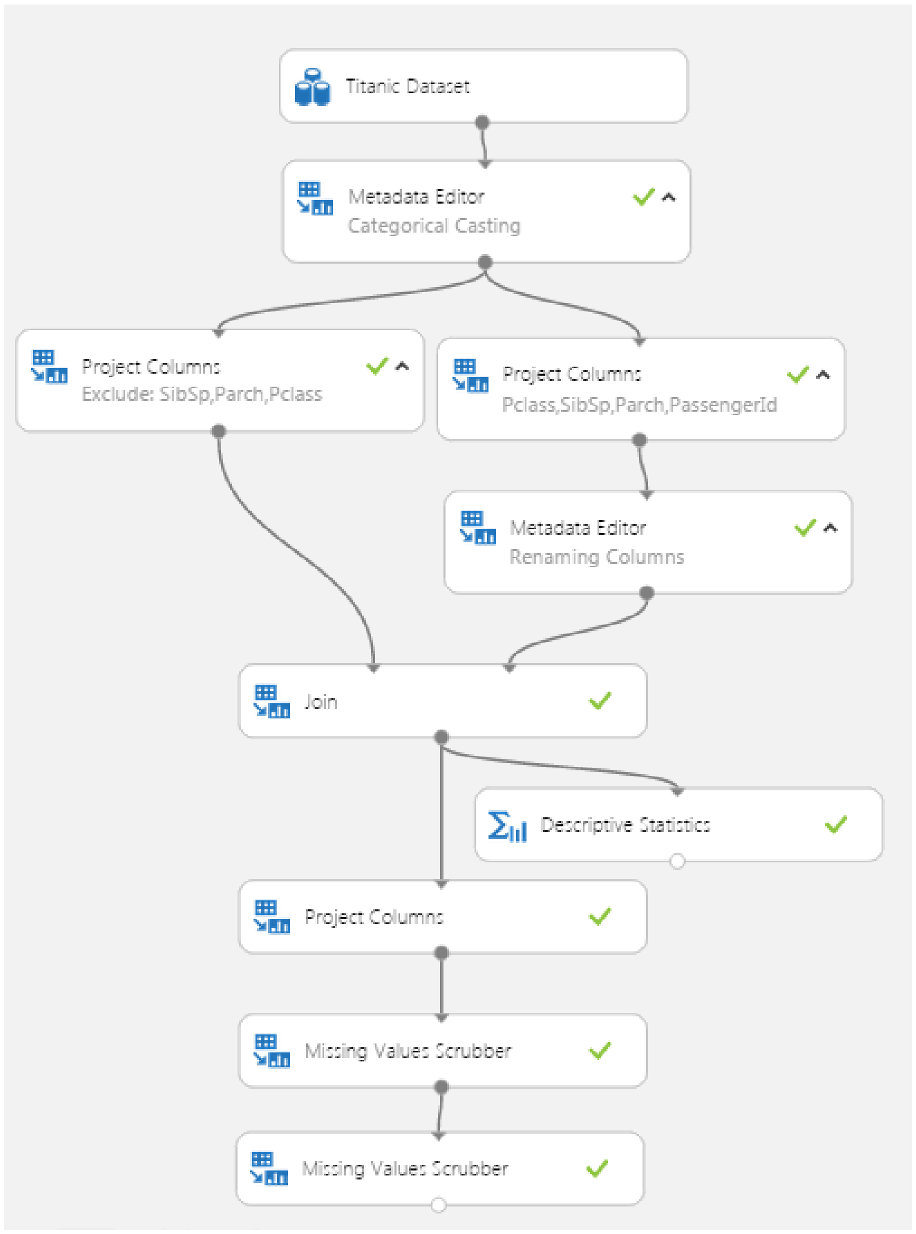



Lab 2: Visualizing, Exploring, Clea

rows	columns
889	8

We are now ready to data mine!

Final Result:

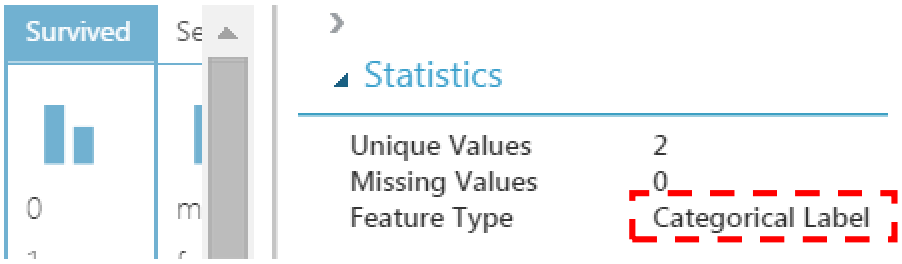


Lab 4: Building a Classification Model in Azure ML

Identify what kind of machine learning problem this is. The column that we want our machine learning model to predict is "survived," which can either be "dead" or "alive", "0" or "1." This falls into what is called a Binary Classification problem. We will now build a classification model in Azure Machine Learning Studio.

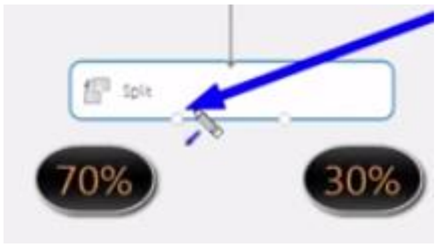
Exercise 1: Response Class & Partitioning Datasets

1. Identify the column the machine learning model will predict (also called a response class), and cast it into a "label":
   a. Drag in a metadata editor and connect it to the output of the missing values scrubber.
      i. Launch Column Selector > Survived
      ii. Change "Fields" to "Labels."
   b. Run the experiment, view the output, and click on the Survived column to make sure the feature type says "Categorical Label."



Unique Values	2
Missing Values	0
Feature Type	Categorical Label

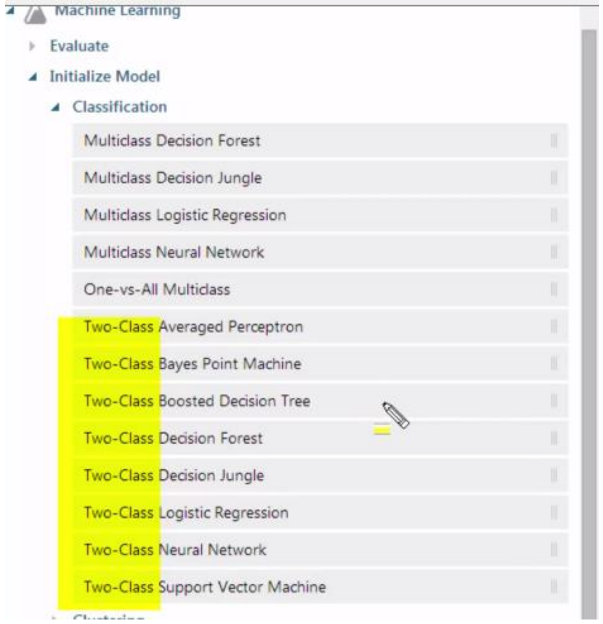
2. Create with-hold data to score and evaluate the performance of our machine learning model. Why do we need with-hold data?
   a. Drag in a Split module under **Data Transformation > Sample and Split**
   b. Connect it to the metadata editor.
   c. Set "Fraction of rows in the first output" to 0.7
      i. This means 70% of the data will output to the left node and 30% to the right.



3. To train an algorithm, drag in a "Train Model" module under **Machine Learning > Train.** Do not run the experiment and do not connect this module yet.

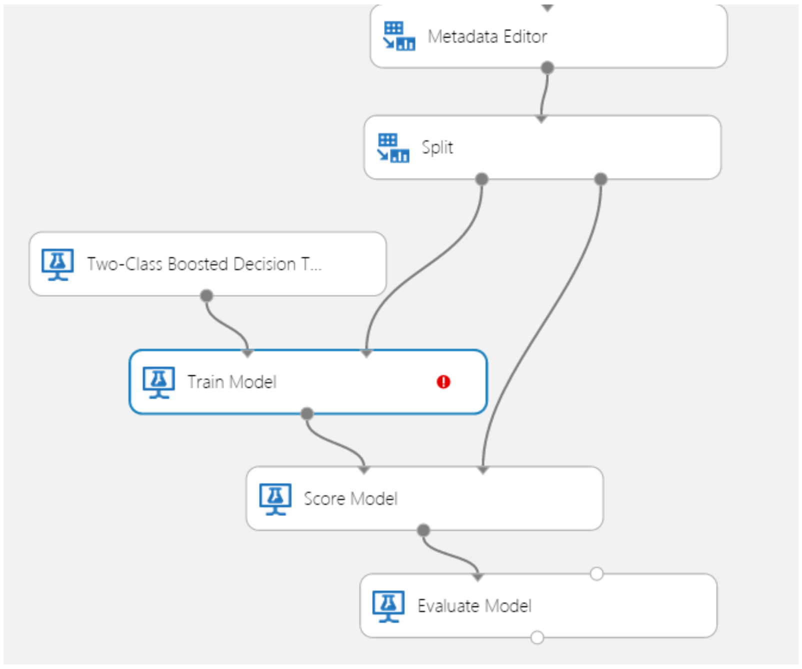
Exercise 2: Algorithm Selection & Training

1. **Under Machine Learning > Initialize Model**, drag in one module that says "two-class". You may choose any of the two-class algorithms. This is an algorithm that will be trained**.** Do not run the experiment and do not connect this module yet.

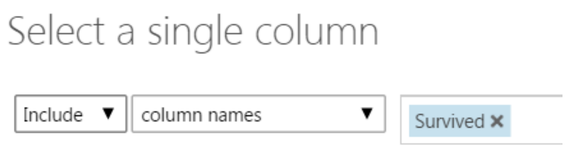

Machine Learning
Evaluate
Initialize Model
Classification
Multiclass Decision Forest
Multiclass Decision Jungle
Multiclass Logistic Regression
Multiclass Neural Network
One-vs-All Multiclass
Two-Class Averaged Perceptron
Two-Class Bayes Point Machine
Two-Class Boosted Decision Tree
Two-Class Decision Forest
Two-Class Decision Jungle
Two-Class Logistic Regression
Two-Class Neural Network
Two-Class Support Vector Machine

2. Drag in a Score Model module under **Machine Learning > Score.** Do not run the experiment and do not connect this module yet.
3. Drag in an Evaluate Model module under **Machine Learning > Evaluation.** Do not run the experiment and do not connect this module yet.
4. Connect the modules together in the following way:
    a. Connect the two-class algorithm module to the left input node of the train model module.
    b. Connect the output node of the train model module to the left input node of the score model module.
    c. Connect the output node of the score model module to the left input node of the evaluate model module.
    d. Connect the left output node of the split module (70%) to the right input node of the train model module.
    e. Connect the right output node of the split module (30%) to the right input of the score model module.

f. The result should look like this:



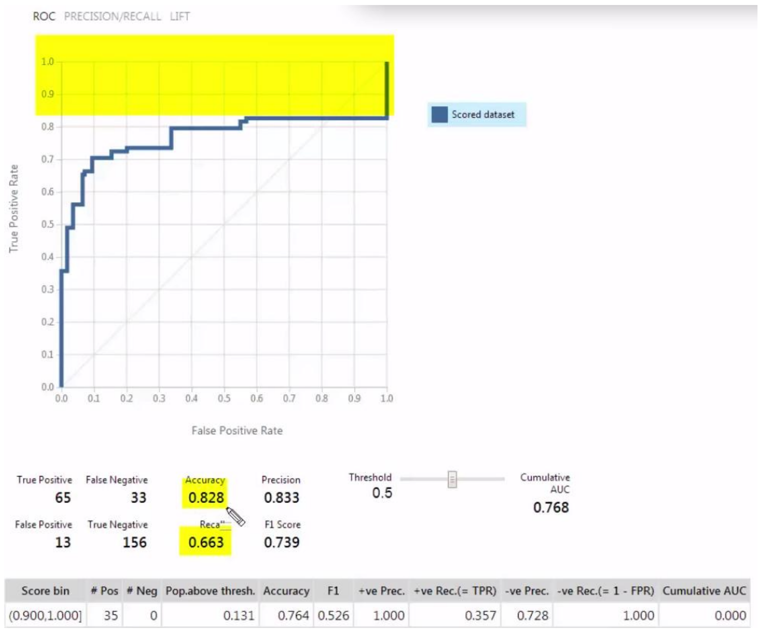
5. Tell the algorithm which attribute it’s trying to predict.
   a. Train Model Module > Launch Column Selector > Select to “include” “column names” “survived”



Select a single column

Include ▼	column names ▼	Survived ✖

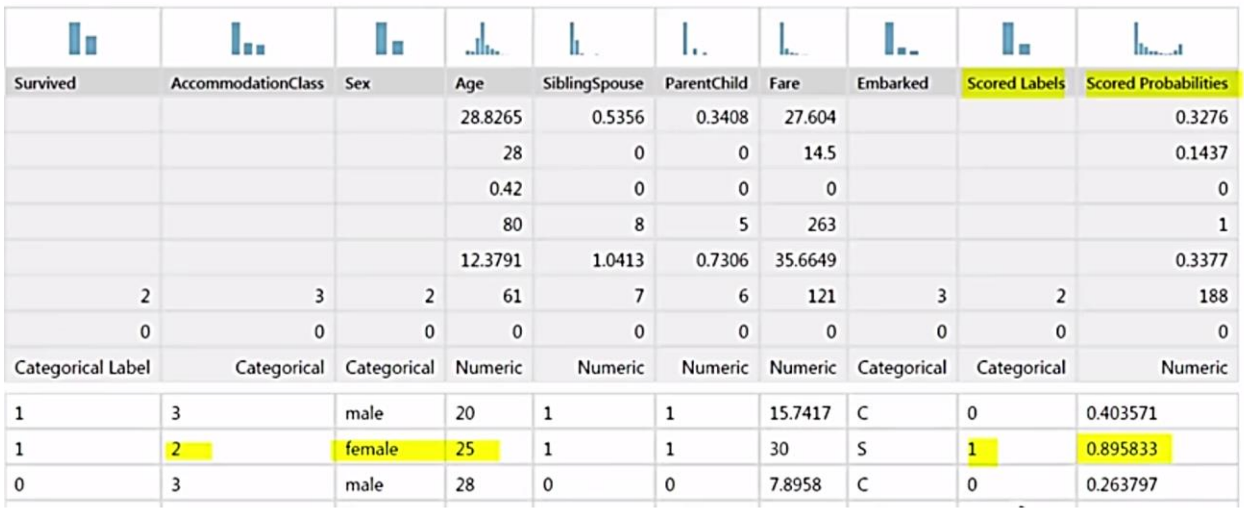
6. Run the experiment.
7. To see how the model did overall, visualize the output node of the Evaluate Model module. Your evaluate model will have different values from the one below.



True Positive	False Negative	Accuracy	Precision	Threshold	Cumulative AUC
65	33	0.828	0.833	0.5	0.768
False Positive	True Negative	Reca“	F1 Score		
13	156	0.663	0.739		

Score bin	# Pos	# Neg	Pop.above thresh.	Accuracy	F1	+ve Prec.	+ve Rec.(= TPR)	-ve Prec.	-ve Rec.(= 1 - FPR)	Cumulative AUC
(0.900,1.000]	35	0	0.131	0.764	0.526	1.000	0.357	0.728	1.000	0.000

8. To see how close the model predicted the withhold split data, visualize the score model module. Your score will have different values from the one below.



Survived	AccommodationClass	Sex	Age	SiblingSpouse	ParentChild	Fare	Embarked	Scored Labels	Scored Probabilities
			28.8265	0.5356	0.3408	27.604			0.3276
			28	0	0	14.5			0.1437
			0.42	0	0	0			0
			80	8	5	263			1
			12.3791	1.0413	0.7306	35.6649			0.3377
2	3	2	61	7	6	121	3	2	188
0	0	0	0	0	0	0	0	0	0
Categorical Label	Categorical	Categorical	Numeric	Numeric	Numeric	Numeric	Categorical	Categorical	Numeric

Survived	AccommodationClass	Sex	Age	SiblingSpouse	ParentChild	Fare	Embarked	Scored Labels	Scored Probabilities
1	3	male	20	1	1	15.7417	C	0	0.403571
1	2	female	25	1	1	30	S	1	0.895833
0	3	male	28	0	0	7.8958	C	0	0.263797

a. In the above example, it predicted that a woman in 2nd class, would have an 89% chance of survival, and it casted her as "1," meaning survived. In actuality, under "Survived" she was marked as "1," as well, so it means that the model predicted her survival correctly.
b. In the above example, it predicted that a 3rd class male, 20 years of age, would have a 40% chance of survival, and casted him as "0," meaning deceased. In actuality, under "Survived" he is marked as "1," meaning he actually survived, which means the model incorrectly predicted his survival.

Lab 5: Model Evaluation

Exercise 1: Background Information and Some Context

So you were given way more metrics than you probably know what to do with. Each machine learning problem will have its own unique goals, thus having different priorities when evaluating as “good” or “bad”. As a result, each problem will also optimize different metrics.

Which metric to optimize?

1. RoC AuC: Overall Performance
2. Precision: Relevance
3. Recall: Thoroughness
4. Accuracy: Correctness

For this lab we will be maximizing the RoC AuC, which stands for “receiver operating characteristic” “area under the curve” which is an overall measure of the model’s robustness or how well can it handle new data that it has not seen before. Once you learn how to optimize one metric, it is the same for the others.

Beginner’s Guide to RoC AuC

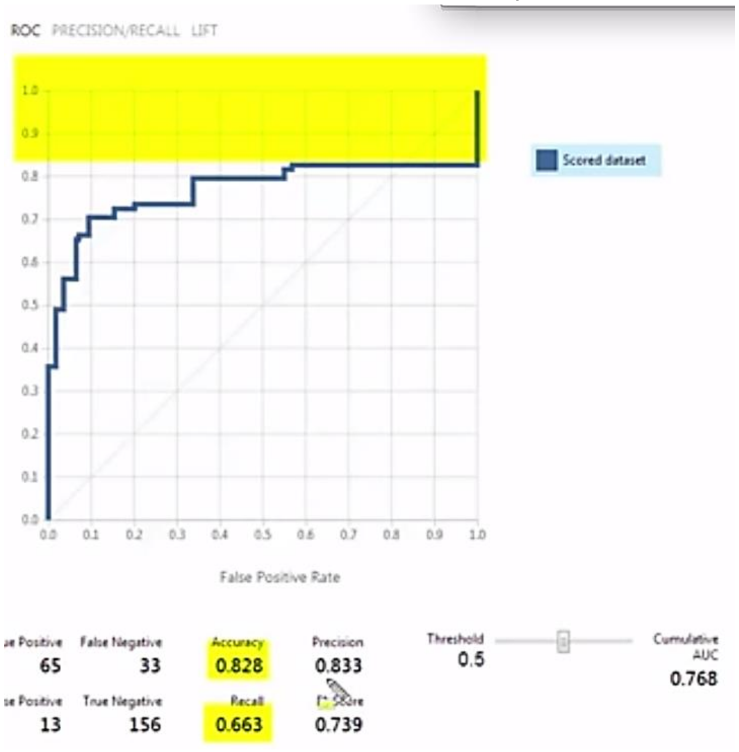
.9~1	=	Suspiciously Good
.8~.9	=	Fair
.7~.8	=	Decent Model
.5~.6	=	Worthless Model

If the RoC is less than .5, it means that flipping a coin yourself would have been more accurate. It also means you should do the opposite of what the model is telling you because it is so completely wrong.

If the RoC AuC is above .9, you might want to check your model again for bias or over-fitting, wherein the model is only good at making predictions for “known” outcomes (the existing data you used to train) and might be terrible when predicting outcomes that are “unknown” (new data coming in).

Exercise 1: Evaluate & Interpret Your Model

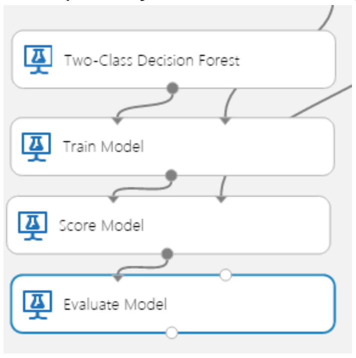
1. Evaulate the model you created in the previous lab by visualizing the Evaluate Model module.
    a. What is its RoC AuC?
        i. Are you happy with its RoC AuC?
    b. How many false positives or false negatives did it predict? What does it mean to have a false positive or false negative?
    c. What was its accuracy? Precision? Recall?
    d. Below is an example of a model's output. Your model may look nothing like this. Here is an example evaulation of the model below:
        i. The RoC AuC is 0.768, a fair model, but it may only be marginally more reliable than flipping a coin for every prediction. Notice that the top of the RoC Curve (highlighted in yellow), has a huge gap, which is bad. We want the RoC Curve to be as high as possible. Its accuracy was decent, which means of the "true" values it predicted, most of them were correct and it minimized the false positives.



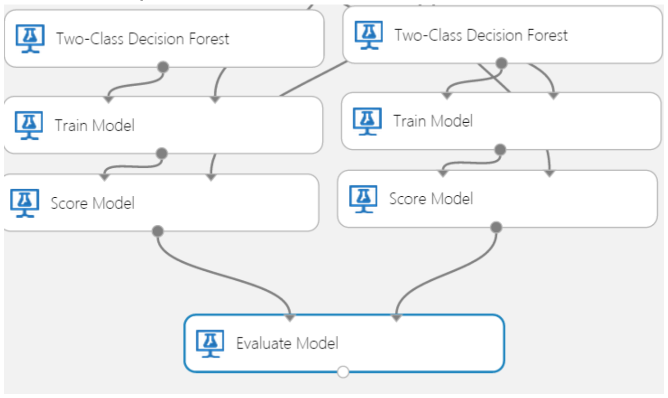
ue Positive	False Negative	Accuracy	Precision	Threshold	Cumulative AUC
65	33	0.828	0.833	0.5	0.768
se Positive	**True Negative**	**Recall**	**F1 Score**		
13	156	0.663	0.739		

Exercise 2: Fine-Tune and Optimize Your Model

1. For this example, make sure your machine learning algorithm is a Two-Class Decision Forest.
2. Compress your modules together so they are in a vertical column fashion:

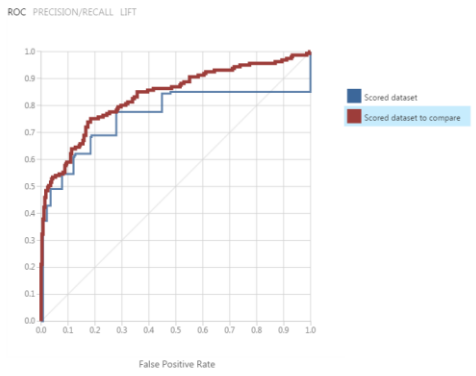


3. Select the four modules by either dragging the highlighted area over them or holding down the control key and clicking on them individually. Now copy and paste (ctrl+c, ctrl+v). Move the new module set to be next to the old one.
4. Align the three module sets side by side, delete one of the evaluate model modules, and have both score model outputs connect to the same evaluate model module.



5. Select the **right** two class decision forest algorithm module. Input the following parameters:
    a. Change "number of trees" to 100.
    b. Change "max depth" to 4.
6. Select the left two class decision forest algorithm module. Input the following parameters:
    a. Change "number of trees" to 8.
    b. Change "max depth" to 32.
7. Now we have 8 large decision trees compared to 100 small decision trees. Which do you think will run better? Run the model to find out. Visualize the evaluate model module.
8. Below is a sample evaluation response, your evaluation may look different.

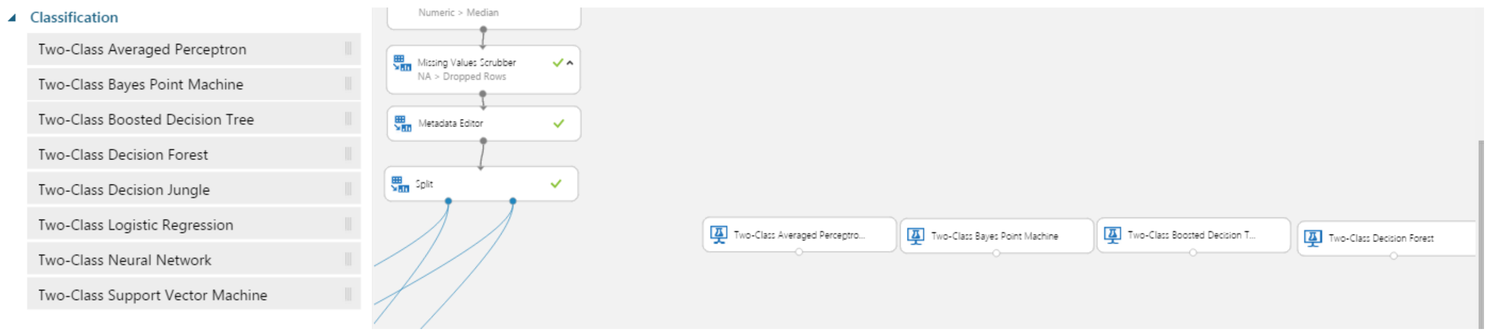
a. The right model (in red) is higher on the graph then the left model (in blue), meaning that the 100 small decision trees vastly out performs 8 large decision trees when it comes to RoC performance.



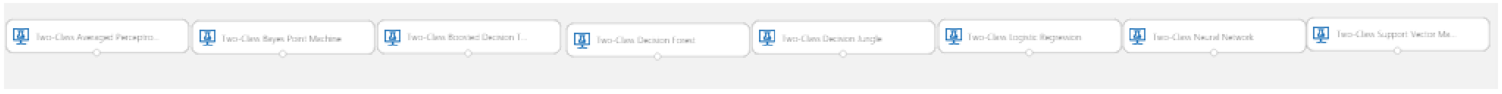
9. Continue optimizing further by tweaking the parameters of the algorithm that "lost" until you are happy with the model's performance.

Exercise 3: Fine-Tune Across Different Algorithms

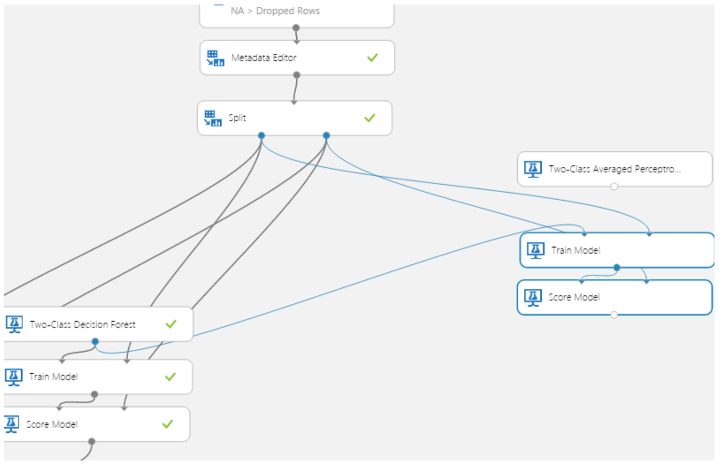
1. Try comparing all of the two-class classification models.
2. Drag in each algorithm module side-by-side


Classification
Two-Class Averaged Perceptron
Two-Class Bayes Point Machine
Two-Class Boosted Decision Tree
Two-Class Decision Forest
Two-Class Decision Jungle
Two-Class Logistic Regression
Two-Class Neural Network
Two-Class Support Vector Machine
Numeric > Median
Missing Values Scrubber
NA > Dropped Rows
Metadata Editor
Split
Two-Class Averaged Perceptro...
Two-Class Bayes Point Machine
Two-Class Boosted Decision T...
Two-Class Decision Forest

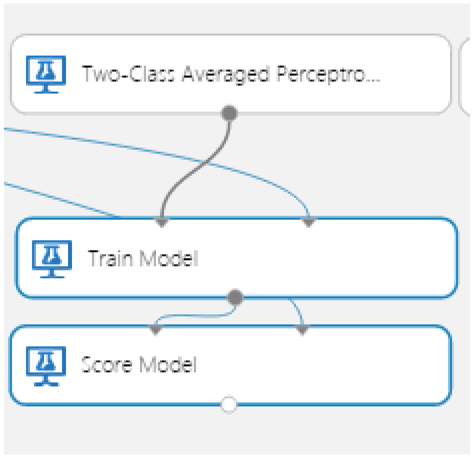
Full view:


Two-Class Averaged Perceptro...
Two-Class Bayes Point Machine
Two-Class Boosted Decision T...
Two-Class Decision Forest
Two-Class Decision Jungle
Two-Class Logistic Regression
Two-Class Neural Network
Two-Class Support Vector Ma...

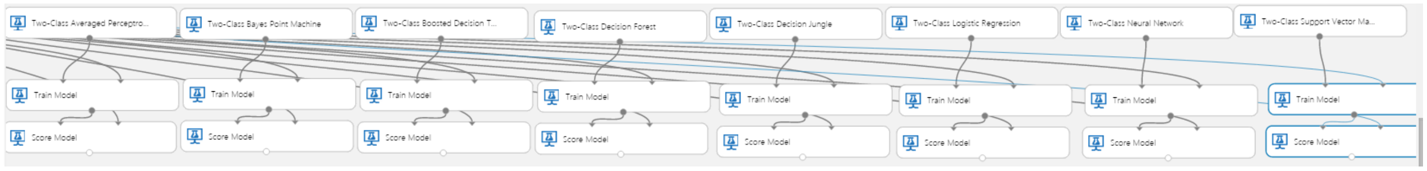
3. Select both the train model module and the score model modules together at the same time (by shift clicking) from one of the two sets of two-class decision forests we ran earlier. Paste them and connect it to the first algorithm in the chain.



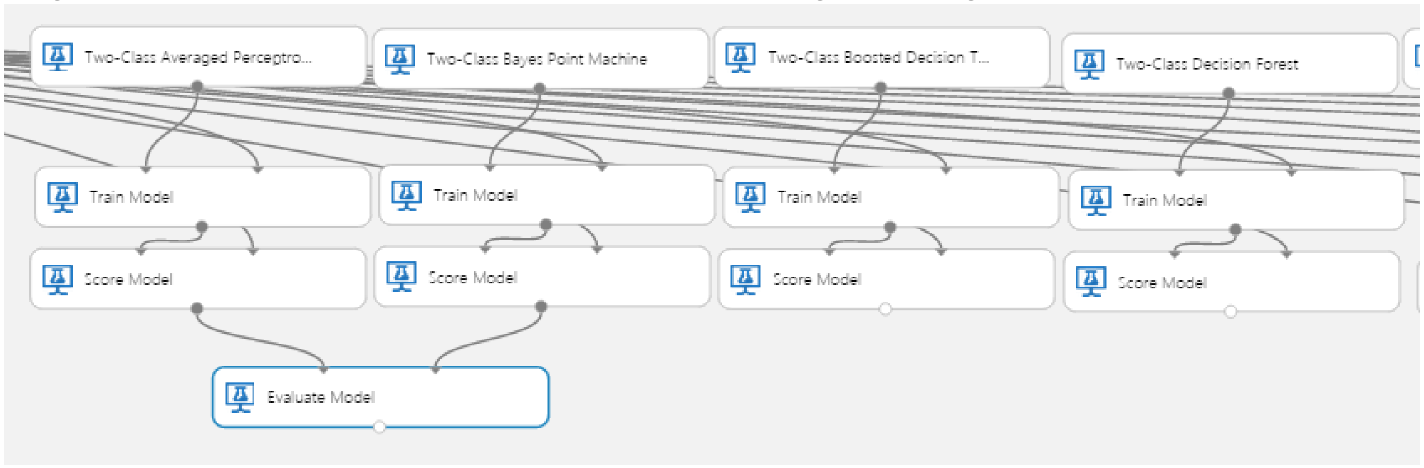
4. Redirect the left input of the train model module to connect to the output of the algorithm module.



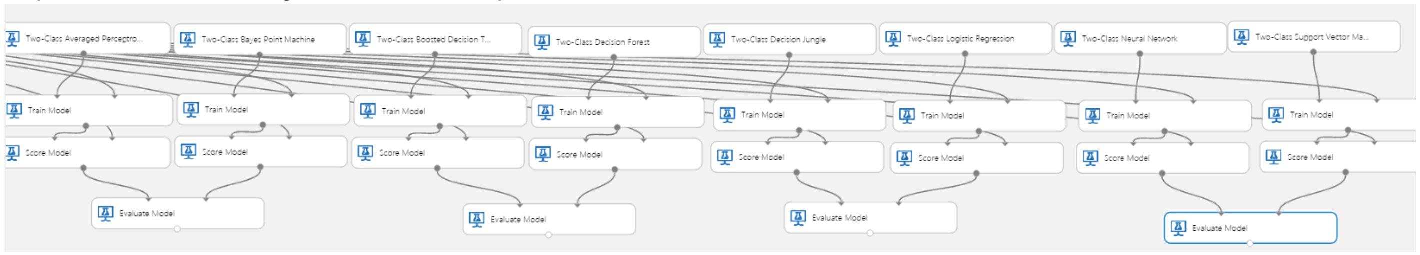
5. Repeat this for all algorithms.


Two-Class Averaged Perceptro...
Two-Class Bayes Point Machine
Two-Class Boosted Decision T...
Two-Class Decision Forest
Two-Class Decision Jungle
Two-Class Logistic Regression
Two-Class Neural Network
Two-Class Support Vector Ma...
Train Model
Score Model

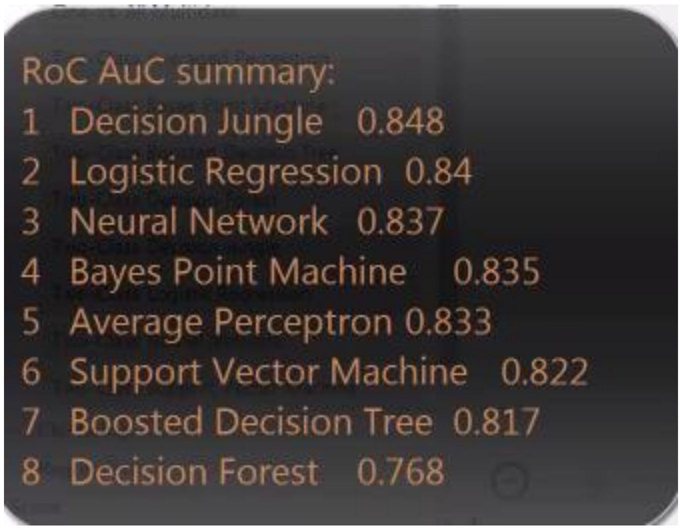
6. Drag in an evaluate model module and connect the first two algorithms together.


Two-Class Averaged Perceptro...
Two-Class Bayes Point Machine
Two-Class Boosted Decision T...
Two-Class Decision Forest
Train Model
Score Model
Evaluate Model

7. Repeat this until all algorithms are outputted to an evaluation model module.


Two-Class Averaged Perceptro...
Two-Class Bayes Point Machine
Two-Class Boosted Decision T...
Two-Class Decision Forest
Two-Class Decision Jungle
Two-Class Logistic Regression
Two-Class Neural Network
Two-Class Support Vector Ma...
Train Model
Score Model
Evaluate Model

8. Visualize each evaluate model module and rank their performance.



RoC AuC summary:

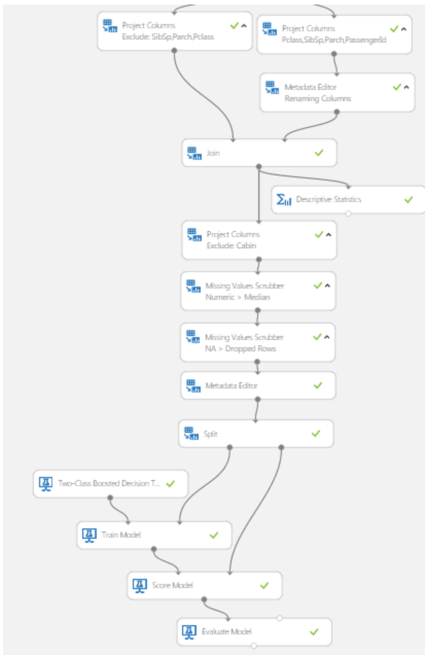
1	Decision Jungle	0.848
2	Logistic Regression	0.84
3	Neural Network	0.837
4	Bayes Point Machine	0.835
5	Average Perceptron	0.833
6	Support Vector Machine	0.822
7	Boosted Decision Tree	0.817
8	Decision Forest	0.768

   a. In the above example, the decision jungle had the best performance under default setting. We may move forward and optimize the decision jungle to create a great performing model.

Lab 6: Creating and Publishing a Predictive Model as a Web Service

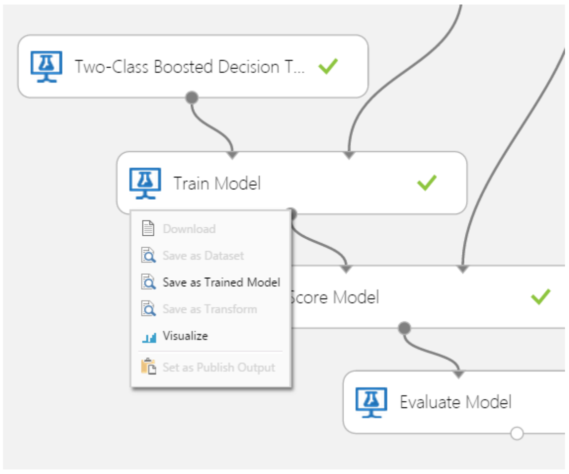
Exercise 1: Exporting & Saving Your Optimized Trained Model

1. Make sure you are happy with the model that you have trained and built.
2. Run the model to ensure that all modules have a green check mark next to them with no errors.

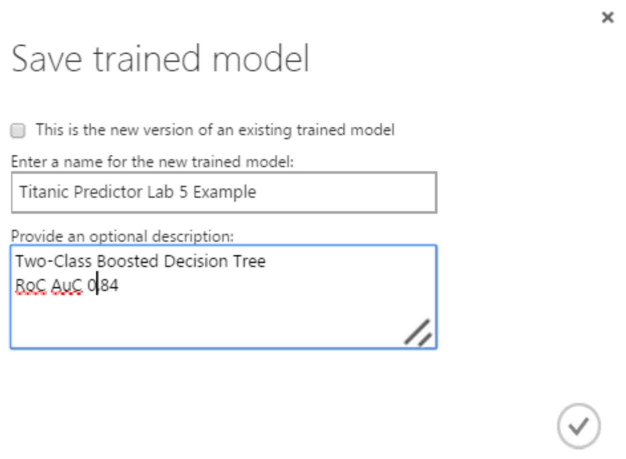


Notice how all the modules have green check marks.

3. When you are ready to export and save your model, right click on the output of the train model module, and "Save as Trained Model".



4. Choose a good name for your model. YOU WILL NOT BE ABLE TO DELETE A TRAINED MODEL ONCE YOU'VE CREATED IT. Also remember that multiple people can work on the same Azure ML experiment. You can "invite" people to work on the experiments as well, so try not to clutter the trained models folder with bad model names.
    a. It's also good practice to note what kind of algorithm it was and the metric that was used to optimize it in the description.



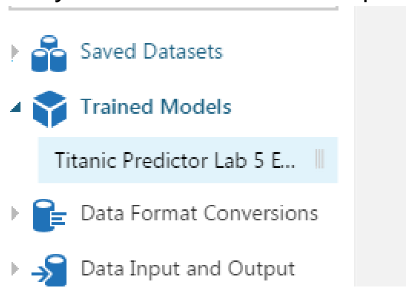
Save trained model

This is the new version of an existing trained model

Enter a name for the new trained model: Titanic Predictor Lab 5 Example

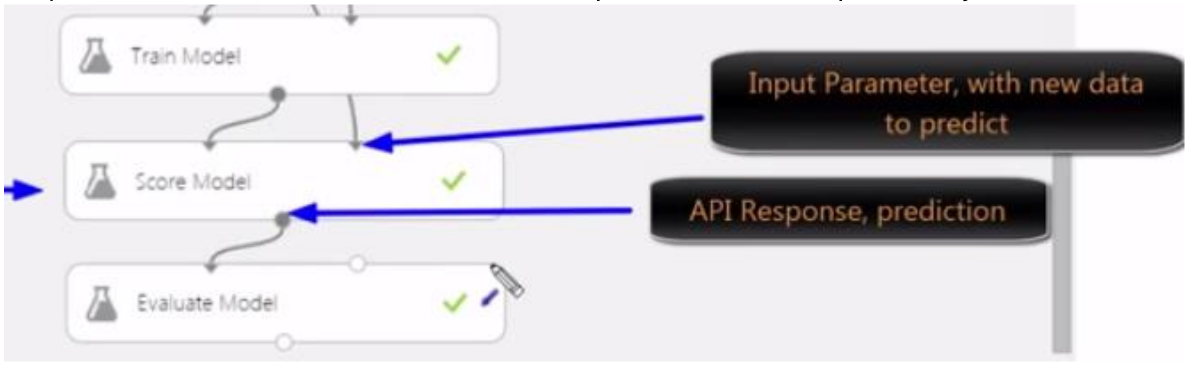
Provide an optional description: Two-Class Boosted Decision Tree
RoC AuC 0.84

5. Verify that the model was exported to the "Trained Models" tab.

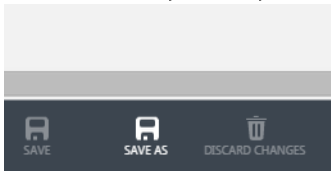


Exercise 2: Setting Up Your Experiment to be Deployed

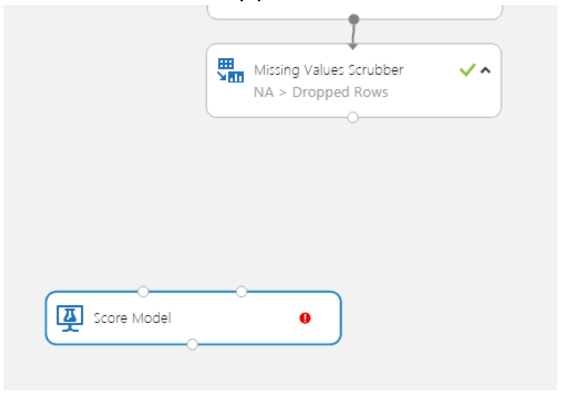
1. Context: The score model module is actually the module that makes the prediction. We want to set up our web service so that any request calls are sent into the score model module as an input and are outputted from the score model module as a prediction (scored probability and scored label).



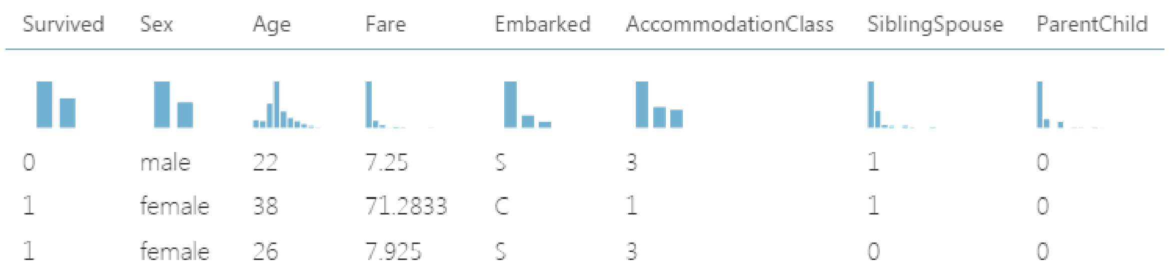
2. Create a new copy of your experiment by hitting "Save As" on the bottom middle toolbar.
    a. Rationale: In the future, if we wanted to go back and re-optimize our model, we still have our original experiment preserved. Messing with a deployed model could disrupt your apps that are dependent on the deployed model.
    b. We want to remove any unnecessary processing modules from our workspace's experiment. Once it is a web service, we do not want to be charged for unnecessary processing power, and we want to speed up each individual iteration of a request call.



3. Remove the evaluate model module by selecting it then hitting the "delete" key on your keyboard.
4. Remove the train model module, the algorithm module (the two-class decision tree), the split module, and the metadata editor that casted survived into a label.
    a. Rationale: Each of those modules were in place to train our algorithm. Since our algorithm has been trained, and preserved as a train model, they are no longer necessary.
    b. You should now be left with only the score model module and the missing values scrubber module which dropped all rows that contained missing values.

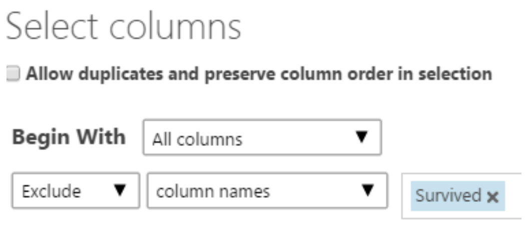


5. Removing the response class from the schema:



Survived	Sex	Age	Fare	Embarked	AccommodationClass	SiblingSpouse	ParentChild
0	male	22	7.25	S	3	1	0
1	female	38	71.2833	C	1	1	0
1	female	26	7.925	S	3	0	0

   a. The score model module requires a defined schema so it knows what kind of parameters to accept such as integer, string, or numeric.
   b. If we made this the input parameter schema right now, the web service would want a value for "Survived, Sex, Age, Fare, Embarked, AccommodationClass, SiblingSpouse, ParentChild" in that exact order in the proper typing and casting. However, there is a problem. It wants a "survived" parameter currently, but its job is to predict "survived". The model wouldn't be very accurate if I wanted to know the answer of the variable that it was trying to predict, so let's remove it.
   c. Drag in a project columns module and set it to begin with "all columns" and exclude "survived". This will drop the survived column all together.



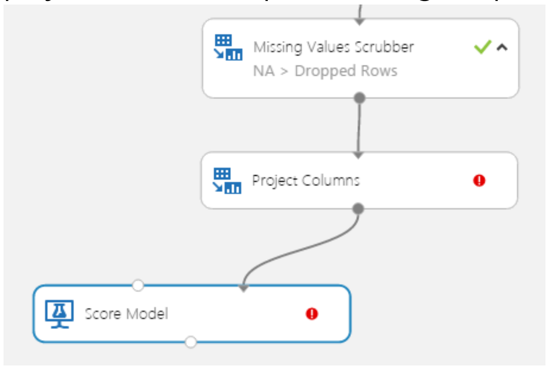
Select columns

Allow duplicates and preserve column order in selection

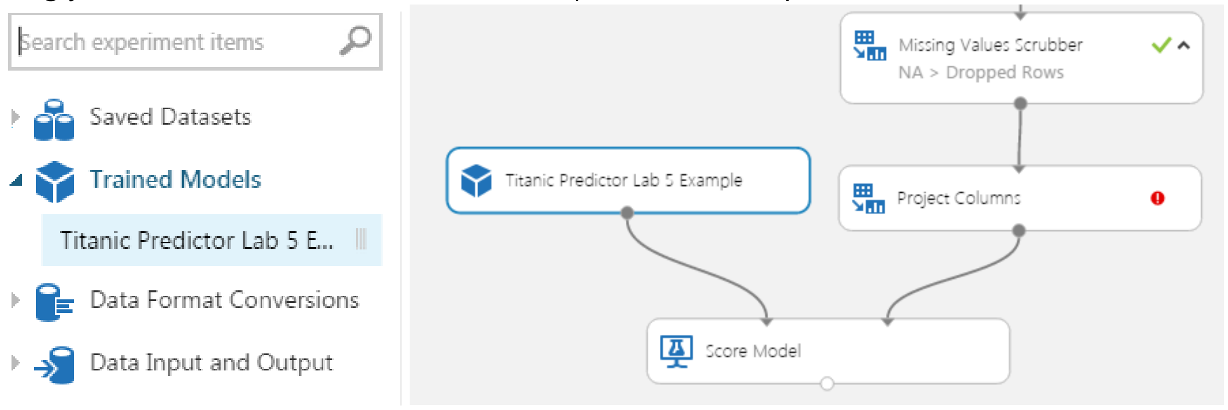
Begin With: All columns

Exclude | column names | Survived

   d. Connect the project columns input to the output of the missing values scrubber. Connect the project columns output to the right input node of the score model.

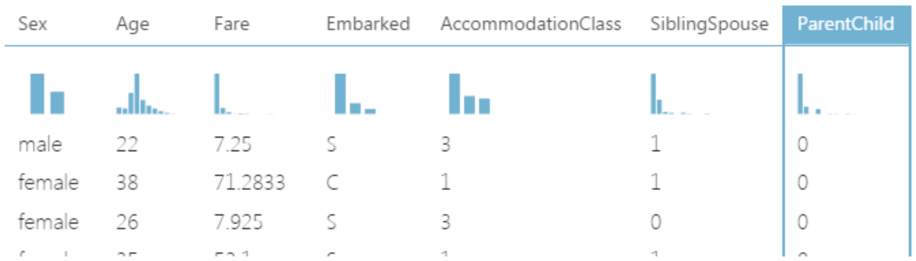


6. Drag your trained model and connect its output to the left input node of the score model.



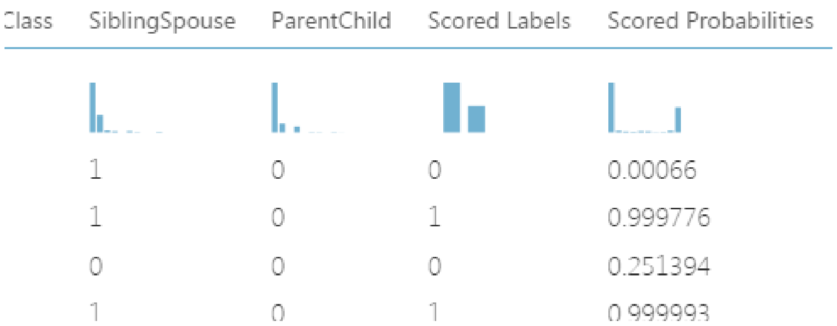
7. Run your model.

8. Visualize the project columns output to double check your schema.
    a. Your schema should be the following:
    Sex<Categorical>, Age<Numeric>, Fare<Numeric>, Embarked<Categorical>, AccommodationClass<Categorical>, SiblingSpouse<Numeric>, ParentChild<Numeric>



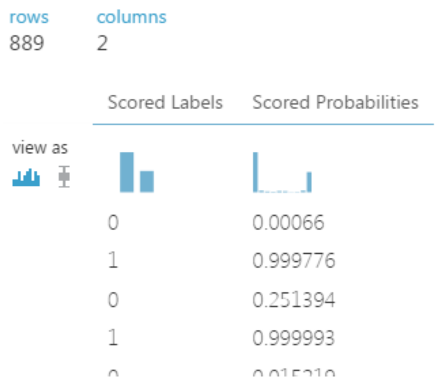
Sex	Age	Fare	Embarked	AccommodationClass	SiblingSpouse	ParentChild
male	22	7.25	S	3	1	0
female	38	71.2833	C	1	1	0
female	26	7.925	S	3	0	0

9. Visualize the output of your score model.
    a. You should see the scored probabilities column and the scored labels column.
    b. The scored probabilities is the prediction that the algorithm has made. For example, there is a 99.9776% chance of survival.
    c. The scored label is the rounded probability to “0” or “1” meaning “survived” or “dead.”



:lass	SiblingSpouse	ParentChild	Scored Labels	Scored Probabilities
	1	0	0	0.00066
	1	0	1	0.999776
	0	0	0	0.251394
	1	0	1	0.999993

    d. The only thing we wanted to be output from the score model was the scored probabilities and the scored label, yet it brought the whole table with us. Let’s project only the scored labels and scored probabilities moving forward so when you make a request call, you get a much cleaner response and less parsing will be involved for the web developer.
10. Drag in a project columns module and connect it to the score model.
    a. Begin with “no columns” and “include” “Scored Probabilities” and “Scored Labels”.
    b. Visualize the project columns output to make sure only scored labels and scored probabilities



rows 889 columns 2

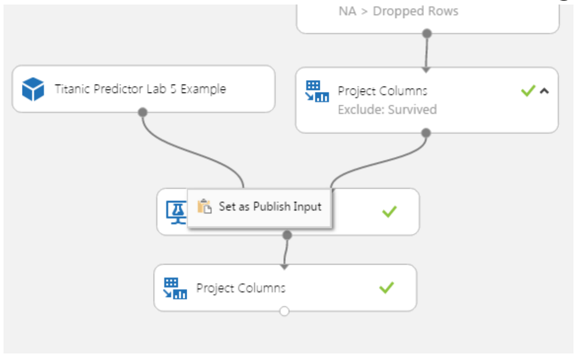
view as

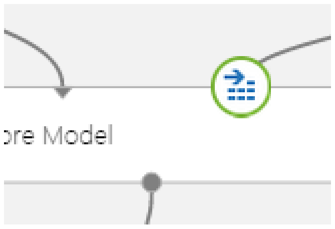
Scored Labels	Scored Probabilities
0	0.00066
1	0.999776
0	0.251394
1	0.999993

    are visible.
    c. Notice how there are 889 rows of predictions and 2 columns.
11. Set published input.
    a. Right click on the right input node of the score model module. This is where our parameters will be sent once a request call has been made.

b. Example function, CallMyFunction(parameter, parameter, parameter). The parameters would enter the node you specified as the published input node.

c. The node should turn blue and be much bigger than it previously was.

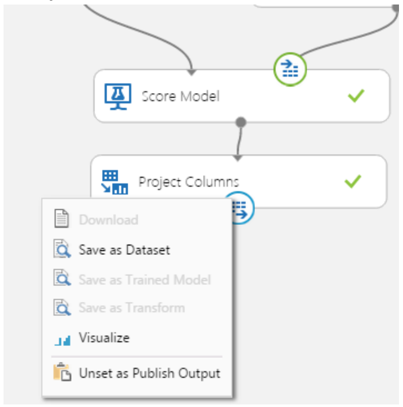


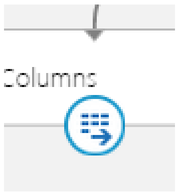


12. Set published output.

    a. Right click on the project columns output and "Set as Publish Output". This is what will be returned (callback) of a successful API response call (POST).

    b. Example function: CallMyFunction(foo){ return bar; }. "Bar" is the thing being returned from this output node.

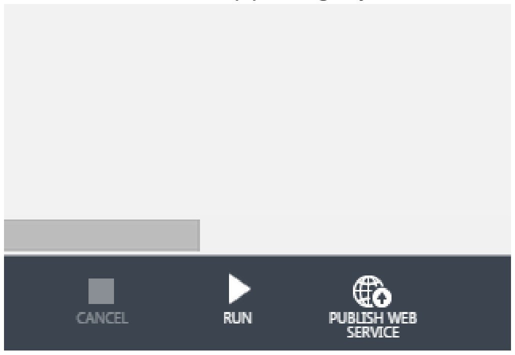




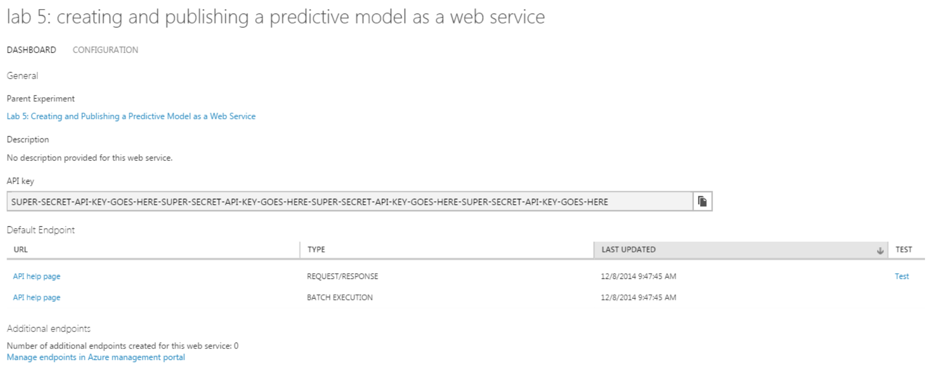
13. "Save" and Run the model one last time. The model is ready to be deployed.

Exercise 3: Deploying Your Model

1. Click on the “Publish Web Service” button. If this is greyed out for you, simply run the model again. All modules must have green check marks as well as specified output/input nodes or else the button will be disabled and appear greyed out.



2. You should see this screen after the deployment.



lab 5: creating and publishing a predictive model as a web service

DASHBOARD CONFIGURATION

General

Parent Experiment

Lab 5: Creating and Publishing a Predictive Model as a Web Service

Description

No description provided for this web service.

API key

SUPER-SECRET-API-KEY-GOES-HERE-SUPER-SECRET-API-KEY-GOES-HERE-SUPER-SECRET-API-KEY-GOES-HERE-SUPER-SECRET-API-KEY-GOES-HERE

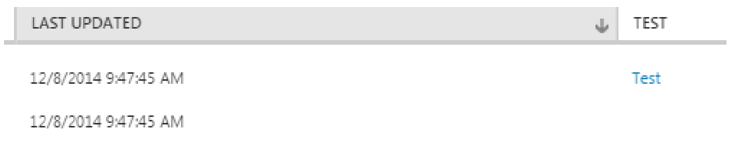
Default Endpoint

URL	TYPE	LAST UPDATED	TEST
API help page	REQUEST/RESPONSE	12/8/2014 9:47:45 AM	Test
API help page	BATCH EXECUTION	12/8/2014 9:47:45 AM	

Additional endpoints

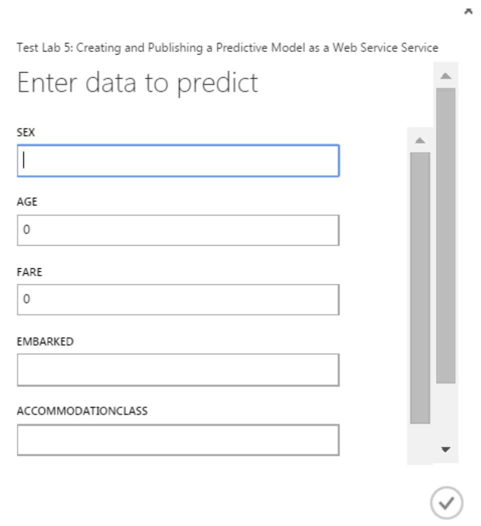
Number of additional endpoints created for this web service: 0
Manage endpoints in Azure management portal

3. Test your model by clicking on the test button on the right hand side.



LAST UPDATED	TEST
12/8/2014 9:47:45 AM	Test
12/8/2014 9:47:45 AM	

4. A form window should pop up with the same name as the columns that you inputted in the score model as a schema.



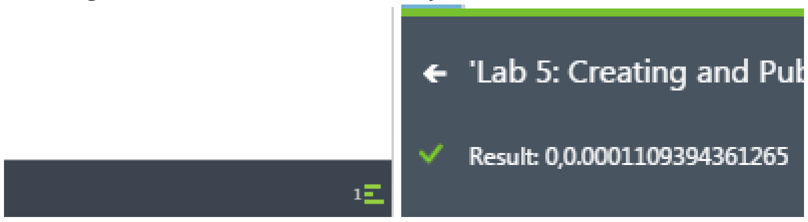
Test Lab 5: Creating and Publishing a Predictive Model as a Web Service Service

Enter data to predict

SEX	
AGE	0
FARE	0
EMBARKED	
ACCOMMODATIONCLASS	

5. Input your own variables into the form. CASE MATTERS. “male” not “Male” or “MALE” same with Embark’s ‘Q’ ‘C’ and ‘S’

6. Submit the response. Look on the bottom right hand side for a loading box. Click on it when it's done loading. Click on details. Look at your result.

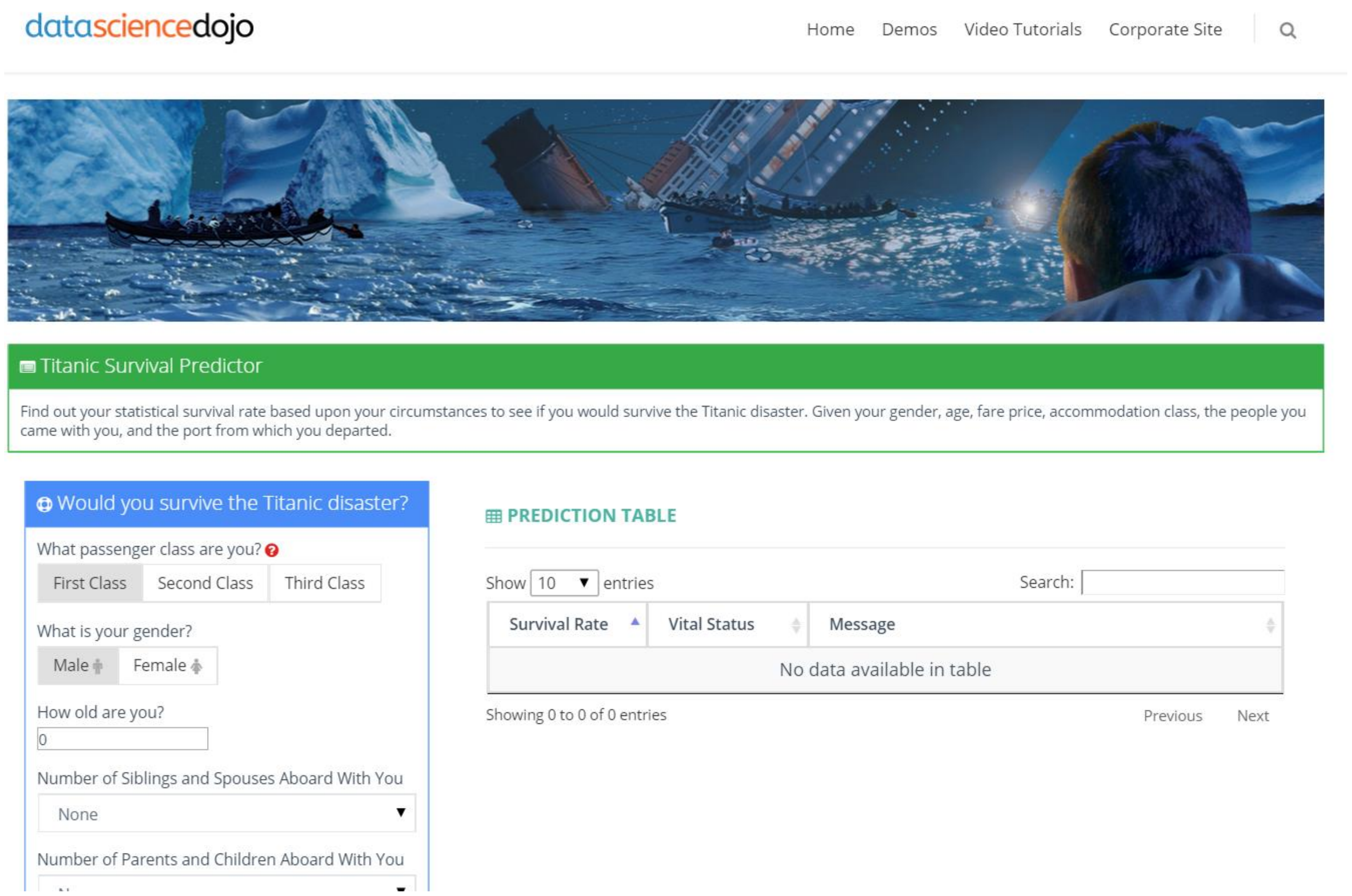


'Lab 5: Creating and Pub

Result: 0,0.0001109394361265

The above example shows a '0' for not survived followed by a '0.0001' meaning it predicted, from the parameters given, that this person would only have a 0.01% chance of survival.

7. Play around with the parameters to see how your model performs. Would you survive the titanic disaster?
8. Data Science Dojo has launched the model you created in this lab as a Django web-app. Play around with it on here: http://demos.datasciencedojo.com/demo/titanic/



Titanic Survival Predictor

Find out your statistical survival rate based upon your circumstances to see if you would survive the Titanic disaster. Given your gender, age, fare price, accommodation class, the people you came with you, and the port from which you departed.

Would you survive the Titanic disaster?

What passenger class are you?

First Class | Second Class | Third Class

What is your gender?

Male | Female

How old are you?

0

Number of Siblings and Spouses Aboard With You

None

Number of Parents and Children Aboard With You

PREDICTION TABLE

Show 10 entries

Search:

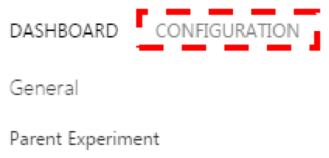
Survival Rate	Vital Status	Message
No data available in table		

Showing 0 to 0 of 0 entries

Previous Next

Exercise 4: Exploring and Testing Your API

1. Optional Step: for good style, click on the configuration button at the top.

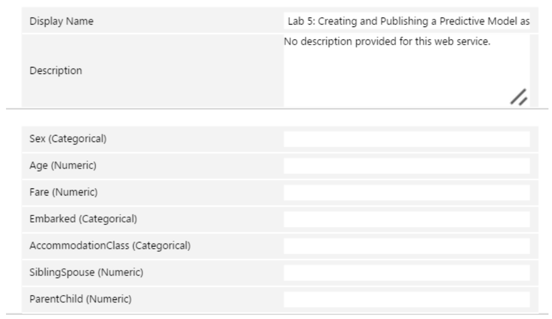


DASHBOARD CONFIGURATION

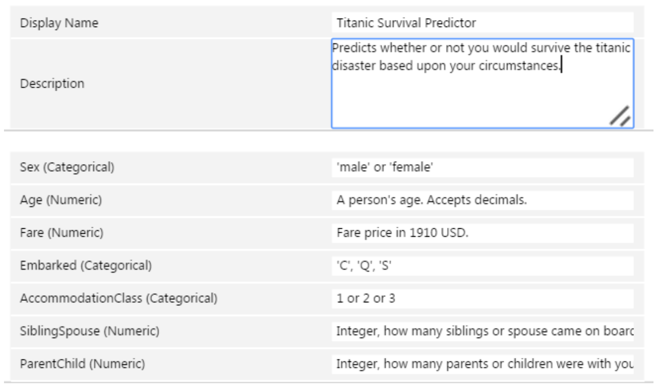
General

Parent Experiment

   a. Provide some documentation for your API.



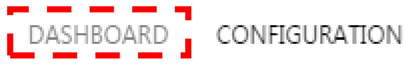
Display Name	Lab 5: Creating and Publishing a Predictive Model as
Description	No description provided for this web service.
Sex (Categorical)	
Age (Numeric)	
Fare (Numeric)	
Embarked (Categorical)	
AccommodationClass (Categorical)	
SiblingSpouse (Numeric)	
ParentChild (Numeric)	



Display Name	Titanic Survival Predictor
Description	Predicts whether or not you would survive the titanic disaster based upon your circumstances.
Sex (Categorical)	'male' or 'female'
Age (Numeric)	A person's age. Accepts decimals.
Fare (Numeric)	Fare price in 1910 USD.
Embarked (Categorical)	'C', 'Q', 'S'
AccommodationClass (Categorical)	1 or 2 or 3
SiblingSpouse (Numeric)	Integer, how many siblings or spouse came on boarc
ParentChild (Numeric)	Integer, how many parents or children were with you

   b. Save your descriptions.

2. Go back to the dashboard.

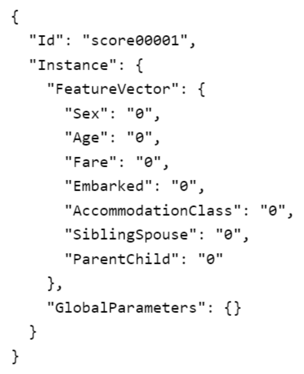


DASHBOARD CONFIGURATION

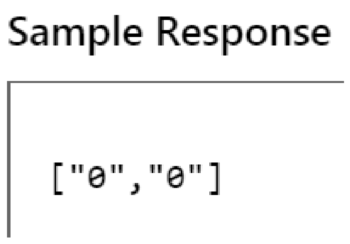
3. View the API help page for "request/response"

   a. Note the sample request. Azure will want a JSON in the exact same manner.

   b. View the sample response. If this is blank, then go back and check your model because something is wrong.



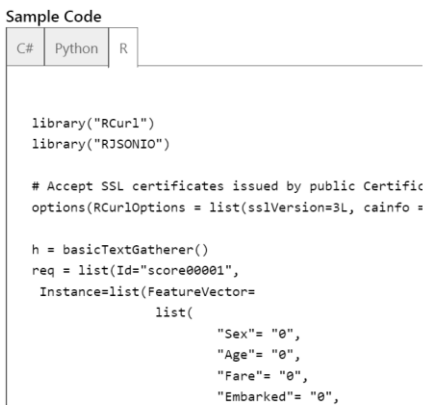
```
{
  "Id": "score00001",
  "Instance": {
    "FeatureVector": {
      "Sex": "0",
      "Age": "0",
      "Fare": "0",
      "Embarked": "0",
      "AccommodationClass": "0",
      "SiblingSpouse": "0",
      "ParentChild": "0"
    },
    "GlobalParameters": {}
  }
}
```



Sample Response

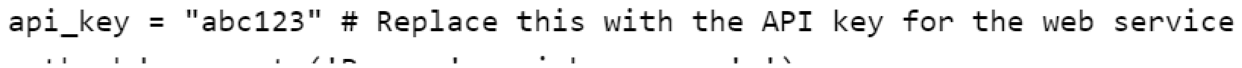
```
["0","0"]
```

4. View the pre-generated code for C#, Python, and R
    a. C# sample code requires .NET 4.5, therefore visual studio >2010+
    b. R sample code will only work for 32-bit R console. May not work with 64-bit R console or RStudio.



```
library("RCurl")
library("RJSONIO")

# Accept SSL certificates issued by public Certific
options(RCurlOptions = list(sslVersion=3L, cainfo =

h = basicTextGatherer()
req = list(Id="score00001",
 Instance=list(FeatureVector=
               list(
                       "Sex"= "0",
                       "Age"= "0",
                       "Fare"= "0",
                       "Embarked"= "0",
```

5. Notice how all the codes require your API key. The API key can be found on the previous page.

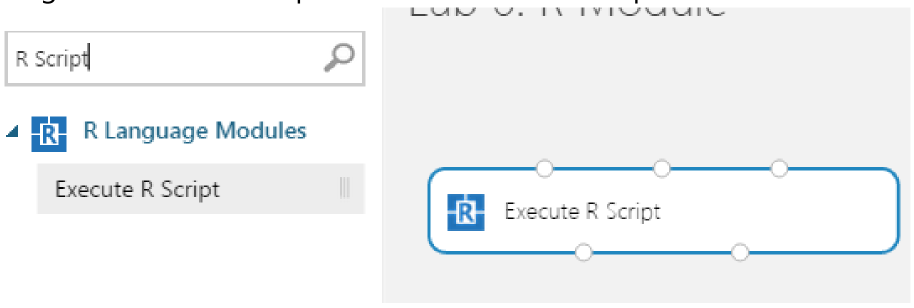


```
api_key = "abc123" # Replace this with the API key for the web service
```

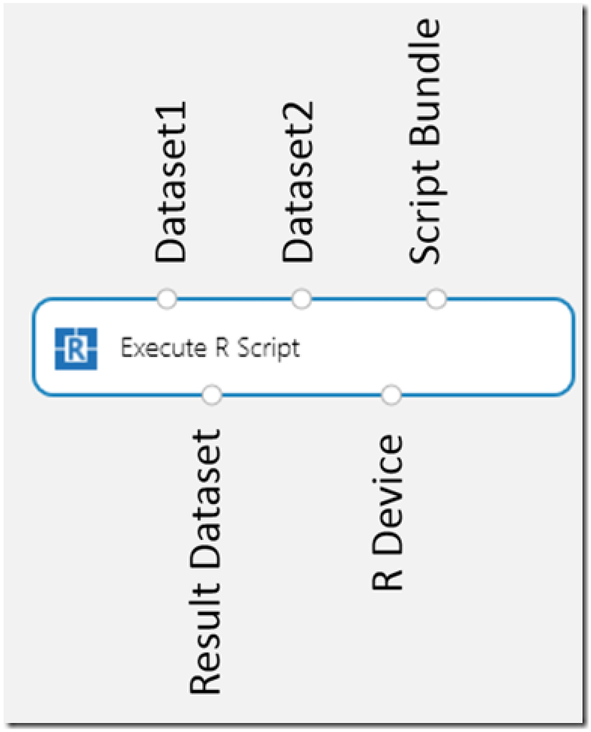
Lab 7: R Script Module

Exercise 1: Exploring the R Module

1. Create a new and blank experiment.
2. Drag an “Execute R Script” module into the workspace.



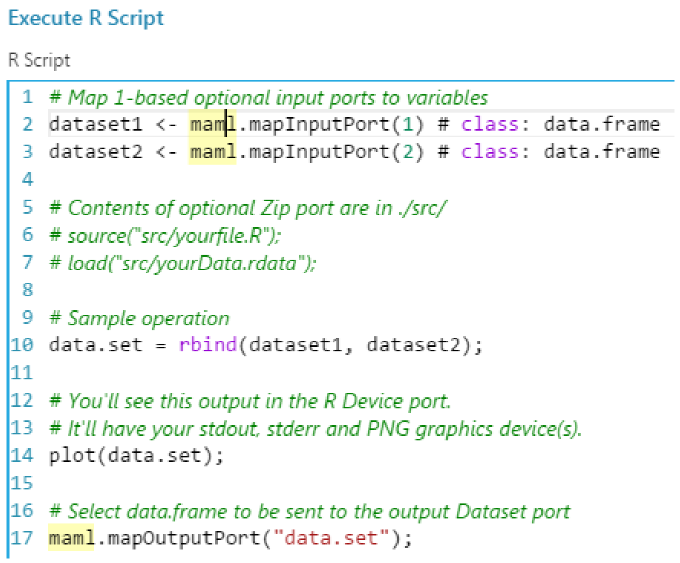
   a. Use this key to understand what each of the nodes mean:



Node	Parameter
Left Input	Input Dataset1
Middle Input	Input Dataset2
Right Input	R Script Bundle
Left Output	Output Dataset, as a data.frame
Right Output	R Device (graphs/charts)

Exercise 2: Exploring the R Script Sample Code

1. Click on the R Script module. An R Script window will open on the right side. This lets you type in raw R code.



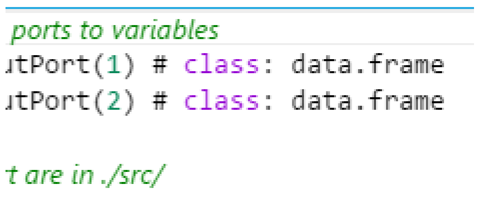
Execute R Script

R Script

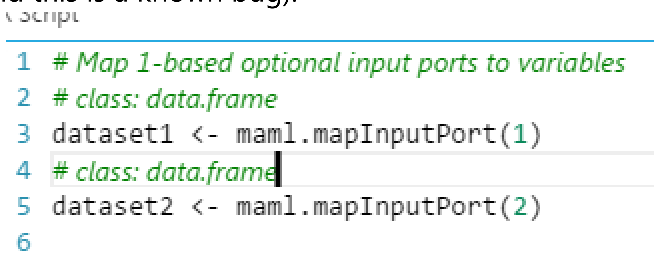
```r
# Map 1-based optional input ports to variables
dataset1 <- maml.mapInputPort(1) # class: data.frame
dataset2 <- maml.mapInputPort(2) # class: data.frame

# Contents of optional Zip port are in ./src/
# source("src/yourfile.R");
# load("src/yourData.rdata");

# Sample operation
data.set = rbind(dataset1, dataset2);

# You'll see this output in the R Device port.
# It'll have your stdout, stderr and PNG graphics device(s).
plot(data.set);

# Select data.frame to be sent to the output Dataset port
maml.mapOutputPort("data.set");
```

   a. Everything in green is a comment. The R module compiler will skip over these lines and not execute them. Comments start with ‘#’ or pound symbol in R. Unlike in R console or Rstudio, a “#” must be the first character in the line.

   b. Notice how # class: data.frame is a comment, yet it is not green, because ‘#’ was not the first character in the line. If this script were to run now, it would error. Fix it so the ‘#’ are on their own lines (Azure ML is still in beta and this is a known bug).



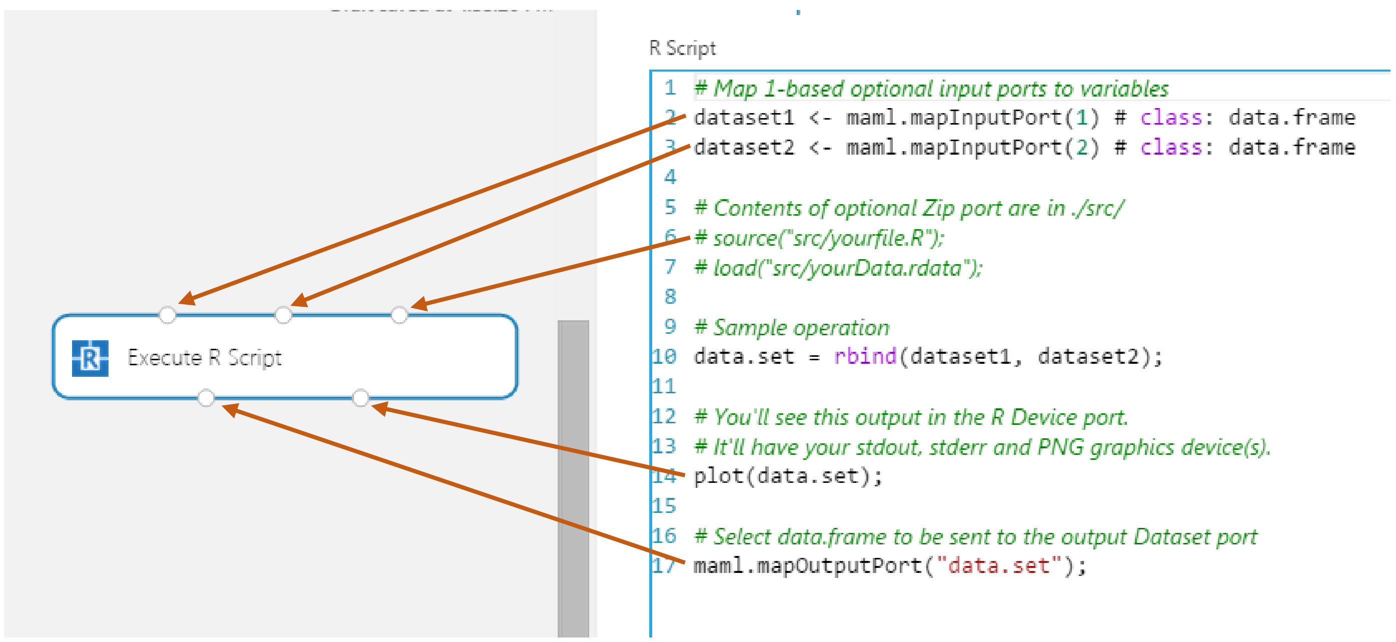
```
ports to variables
utPort(1) # class: data.frame
utPort(2) # class: data.frame

t are in ./src/
```



```r
# Map 1-based optional input ports to variables
# class: data.frame
dataset1 <- maml.mapInputPort(1)
# class: data.frame
dataset2 <- maml.mapInputPort(2)

```

2. Explanation of R Module Native Syntax

Node	Parameter	R Script Module	Explanation
Left Input	Input Dataset1	mam1.mapInputPort(1)	Reads & parses left input.
Middle Input	Input Dataset2	mam1.mapInputPort(2)	Reads & parses right input.
Right Input	R Script Bundle	Source("src/yourFile.R");	External R Scripts.
Left Output	Output Dataset	mam1.mapOutputPort("<SomeObject>")	Returns output as a dataset.
Right Output	R Device (graphs/charts)	plot(<SomeObject>);	Views R Console/graphics.



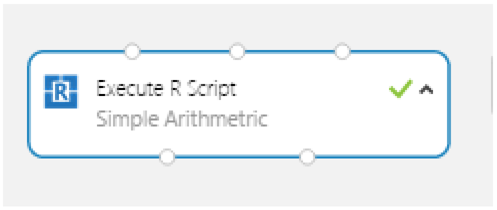
```
# Map 1-based optional input ports to variables
dataset1 <- maml.mapInputPort(1) # class: data.frame
dataset2 <- maml.mapInputPort(2) # class: data.frame

# Contents of optional Zip port are in ./src/
# source("src/yourfile.R");
# load("src/yourData.rdata");

# Sample operation
data.set = rbind(dataset1, dataset2);

# You'll see this output in the R Device port.
# It'll have your stdout, stderr and PNG graphics device(s).
plot(data.set);

# Select data.frame to be sent to the output Dataset port
maml.mapOutputPort("data.set");
```

Exercise 3: Buggy R Script

Although Azure currently supports R in the R script module, it is still quite buggy within the beta release. Some known issues are:

- R code will fail to comment a line unless '#' is the very first character in a line.
- Copying and pasting code will result in runtime errors within the module. If at first copy and paste does not work, try to type out everything when possible into the R module.
- The debugger will only tell you that there has been an error in the script and will not give you clues as to why your code failed to compile or run. You have to look for the error yourself in the trace-back log.

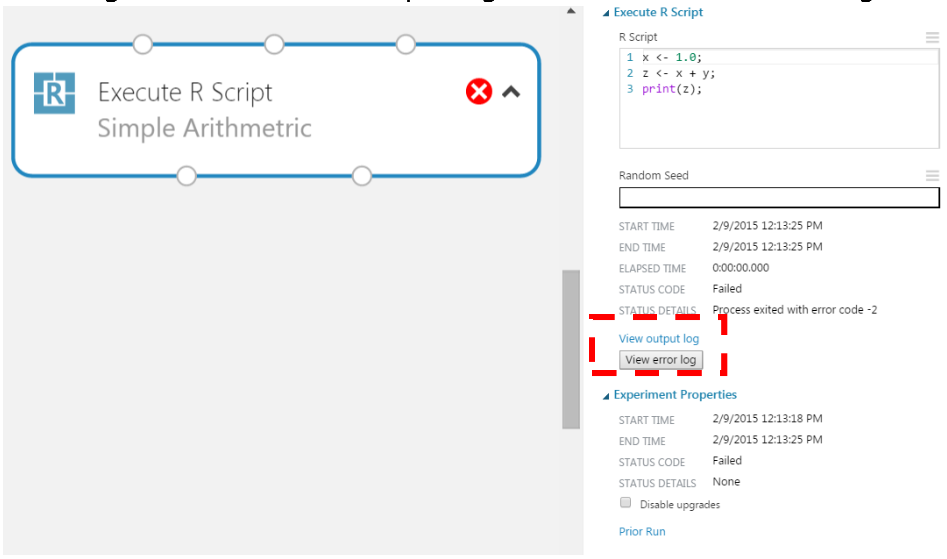
1. We will try, on purpose, to execute a bug into the R script module so you can see what to expect. Drag in an Execute R Script module if you do not have one already.



2. Insert the following code into R script module and try to run the experiment. What did you see once it finishes loading?

```
x <- 1.0;
z <- x + y;
print(z);
```

3. Why do you think the R module returned an error?
4. Troubleshooting: Click on the view output log button (above view error log).



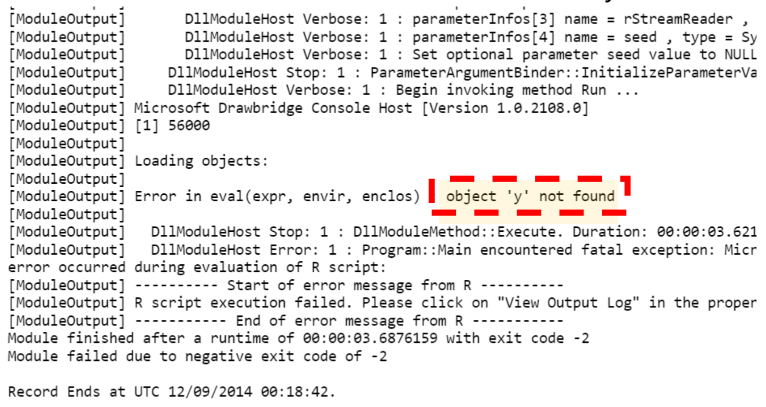
```
x <- 1.0;
z <- x + y;
print(z);
```

Random Seed

START TIME	2/9/2015 12:13:25 PM
END TIME	2/9/2015 12:13:25 PM
ELAPSED TIME	0:00:00.000
STATUS CODE	Failed
STATUS DETAILS	Process exited with error code -2

START TIME	2/9/2015 12:13:18 PM
END TIME	2/9/2015 12:13:25 PM
STATUS CODE	Failed
STATUS DETAILS	None

a. Towards the bottom of the traceback window, you will find the error.

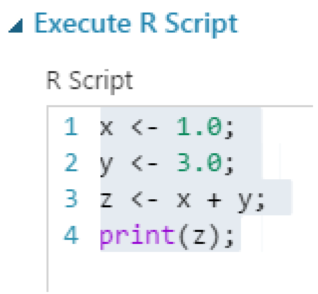


```
[ModuleOutput]       DllModuleHost Verbose: 1 : parameterInfos[3] name = rStreamReader ,
[ModuleOutput]       DllModuleHost Verbose: 1 : parameterInfos[4] name = seed , type = Sy
[ModuleOutput]       DllModuleHost Verbose: 1 : Set optional parameter seed value to NULL
[ModuleOutput]     DllModuleHost Stop: 1 : ParameterArgumentBinder::InitializeParameterVa
[ModuleOutput]     DllModuleHost Verbose: 1 : Begin invoking method Run ...
[ModuleOutput] Microsoft Drawbridge Console Host [Version 1.0.2108.0]
[ModuleOutput] [1] 56000
[ModuleOutput]
[ModuleOutput] Loading objects:
[ModuleOutput]
[ModuleOutput] Error in eval(expr, envir, enclos) : object 'y' not found
[ModuleOutput]
[ModuleOutput]   DllModuleHost Stop: 1 : DllModuleMethod::Execute. Duration: 00:00:03.621
[ModuleOutput]   DllModuleHost Error: 1 : Program::Main encountered fatal exception: Micr
error occurred during evaluation of R script:
[ModuleOutput] ---------- Start of error message from R ----------
[ModuleOutput] R script execution failed. Please click on "View Output Log" in the proper
[ModuleOutput] ----------- End of error message from R -----------
Module finished after a runtime of 00:00:03.6876159 with exit code -2
Module failed due to negative exit code of -2

Record Ends at UTC 12/09/2014 00:18:42.
```

b. It seems that an objected called ‘y’ was called but never defined.

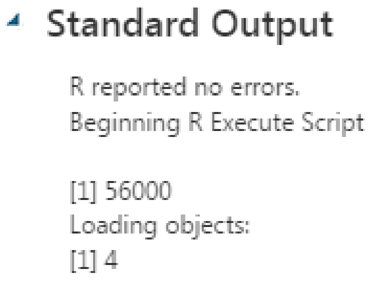
5. Correct the code with the below snippet. Call Z at the end so it outputs Z’s value to the console.

```
x <- 1.0;
y <- 3.0;
z <- x + y;
print(z);
```



```
x <- 1.0;
y <- 3.0;
z <- x + y;
print(z);
```

6. Visualize the right output node. You should see an output of 4. Notice that the R device acts in place of the R console, as well as the R plot and graphs.



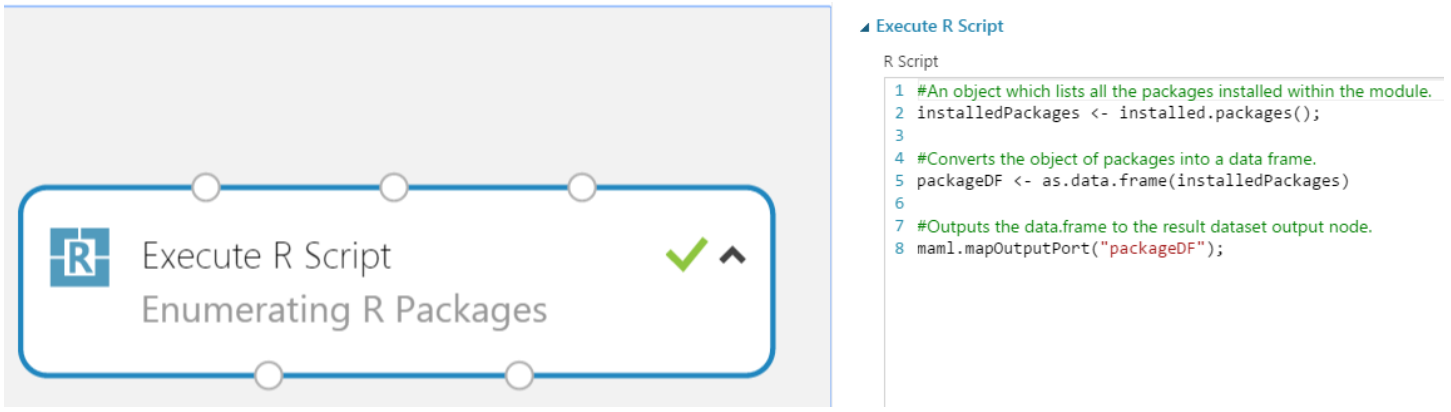
```
R reported no errors.
Beginning R Execute Script

[1] 56000
Loading objects:
[1] 4
```

Exercise 4: Enumerating R Packages

Let's see which packages, R plugins, are shipped natively with the R Module. Libraries are a powerful way to increase your data science potential and productivity without having to dig too deeply into the details of implementation. Have an algorithm that you want to implement? Chances are there is a library that does that for you.

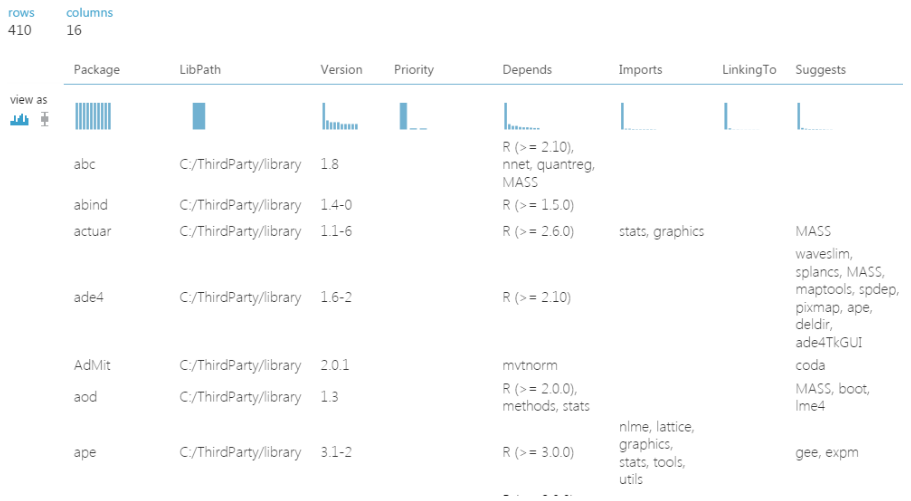
1. Drag in an Execute R Script module if you do not have one already.
2. Insert the following code into the R Script Module

```
#An object which lists all the packages installed within the module.
installedPackages <- installed.packages();

#Converts the object of packages into a data frame.
packageDF <- as.data.frame(installedPackages)

#Outputs the data.frame to the result dataset output node.
maml.mapOutputPort("packageDF");
```



```
1 #An object which lists all the packages installed within the module.
2 installedPackages <- installed.packages();
3
4 #Converts the object of packages into a data frame.
5 packageDF <- as.data.frame(installedPackages)
6
7 #Outputs the data.frame to the result dataset output node.
8 maml.mapOutputPort("packageDF");
```

3. On line 5, 'installedPackages was converted to a data.frame because mapOutputPort expects a data.frame as a parameter.



Package	LibPath	Version	Priority	Depends	Imports	LinkingTo	Suggests
abc	C:/ThirdParty/library	1.8		R (>= 2.10), nnet, quantreg, MASS			
abind	C:/ThirdParty/library	1.4-0		R (>= 1.5.0)			
actuar	C:/ThirdParty/library	1.1-6		R (>= 2.6.0)	stats, graphics		MASS
ade4	C:/ThirdParty/library	1.6-2		R (>= 2.10)			waveslim, splancs, MASS, maptools, spdep, pixmap, ape, deldir, ade4TkGUI
AdMit	C:/ThirdParty/library	2.0.1		mvtnorm			coda
aod	C:/ThirdParty/library	1.3		R (>= 2.0.0), methods, stats			MASS, boot, lme4
ape	C:/ThirdParty/library	3.1-2		R (>= 3.0.0)	nlme, lattice, graphics, stats, tools, utils		gee, expm

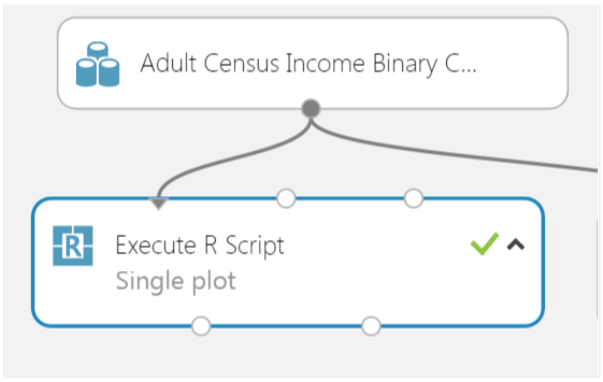
Exercise 5: ggplots, Advanced Visualizations within the R Module

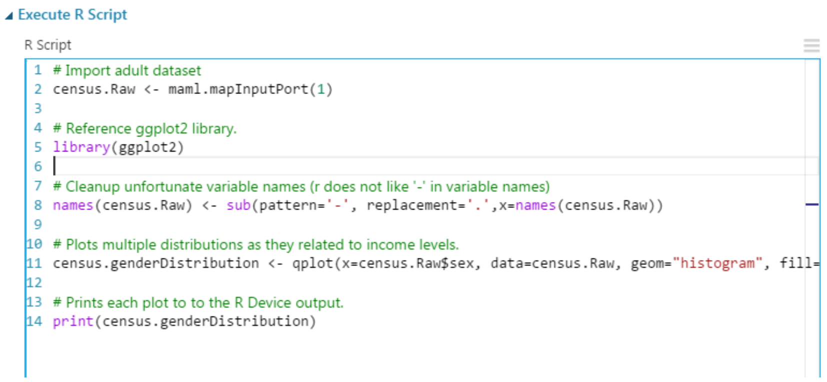
Let's try one of these plugins. Ggplot is a powerful data visualization R plugin that extends the data visualization of R beyond its normal capabilities.

1. We will be working with the "Adult Census Income Binary Classification" dataset which ships as a default dataset with your workspace. Search for it and drag it into your workspace.
2. Drag in an Execute R Script module if you do not have one already.

```
# Import adult dataset
census.Raw <- maml.mapInputPort(1)

# Reference ggplot2 library.
library(ggplot2)

# Cleanup unfortunate variable names (r does not like '-' in variable names)
names(census.Raw) <- sub(pattern='-', replacement='.',x=names(census.Raw))

# Plots multiple distributions as they related to income levels.
census.genderDistribution <- qplot(x=census.Raw$sex, data=census.Raw, geom="histogram",
fill=census.Raw$income, position="dodge")

# Prints each plot to to the R Device output.
print(census.genderDistribution)
```

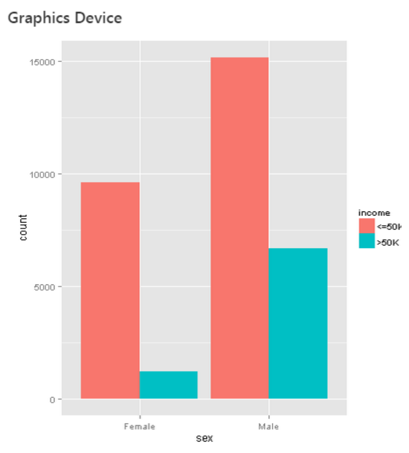
3. Insert the following code:





```
# Import adult dataset
census.Raw <- maml.mapInputPort(1)

# Reference ggplot2 library.
library(ggplot2)

# Cleanup unfortunate variable names (r does not like '-' in variable names)
names(census.Raw) <- sub(pattern='-', replacement='.',x=names(census.Raw))

# Plots multiple distributions as they related to income levels.
census.genderDistribution <- qplot(x=census.Raw$sex, data=census.Raw, geom="histogram", fill=

# Prints each plot to to the R Device output.
print(census.genderDistribution)
```

4. Visualize the R device output.



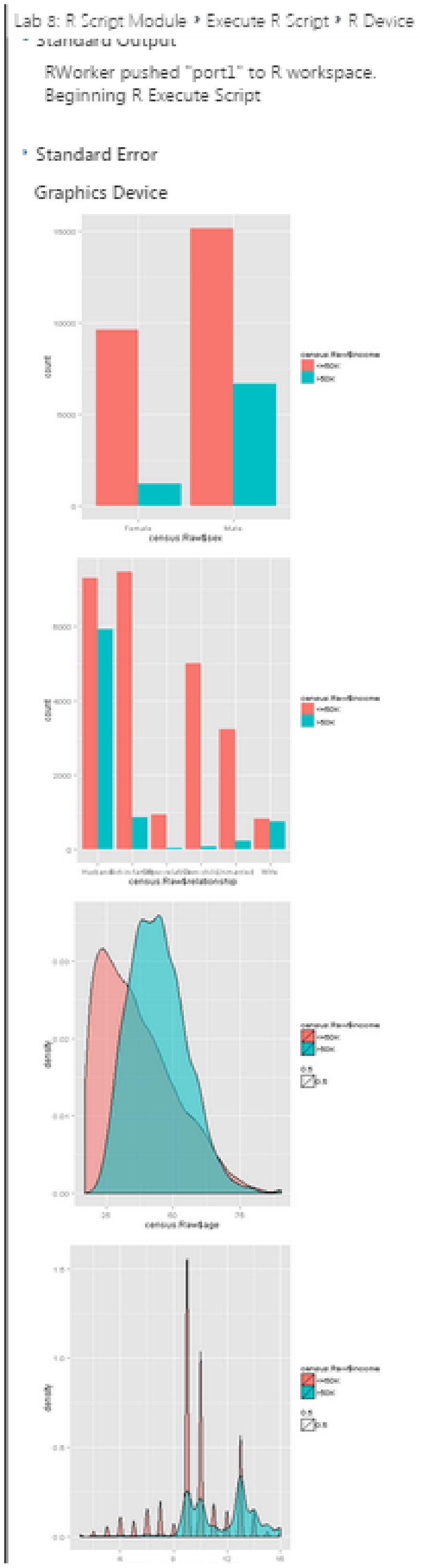
Exercise 6: Multiple plots in R

Each graphical instance in the R script module will be rendered and added to the R Device output of the R Script module. This is nice because in RStudio or R Console, you can only have 1 graphical plot at a time (calling multiple "plots" will overwrite the previous).

1. Insert the following code, which will result in 4 visualizations.

```
# Import adult dataset
census.Raw <- maml.mapInputPort(1)

# Reference ggplot2 library.
library(ggplot2)

# Cleanup unfortunate variable names (r does not like '-' in variable names)
names(census.Raw) <- sub(pattern='-', replacement='.',x=names(census.Raw))

# Plots multiple distributions as they related to income levels.
census.genderDistribution <- qplot(x=census.Raw$sex, data=census.Raw, geom="histogram",
fill=census.Raw$income, position="dodge")
census.relationshipDistribution <- qplot(x=census.Raw$relationship, data=census.Raw,
geom="histogram", fill=census.Raw$income, position="dodge");
census.ageDistribution <- qplot(x=census.Raw$age, data=census.Raw, geom="density",
alpha=0.5, fill=census.Raw$income);
census.educationDistribution <- qplot(x=census.Raw$education.num, data=census.Raw,
geom="density", alpha=0.5, fill=census.Raw$income);

# Prints each plot to to the R Device output.
print(census.genderDistribution);
print(census.relationshipDistribution)
print(census.ageDistribution)
print(census.educationDistribution)
```

2. Your output should be this:



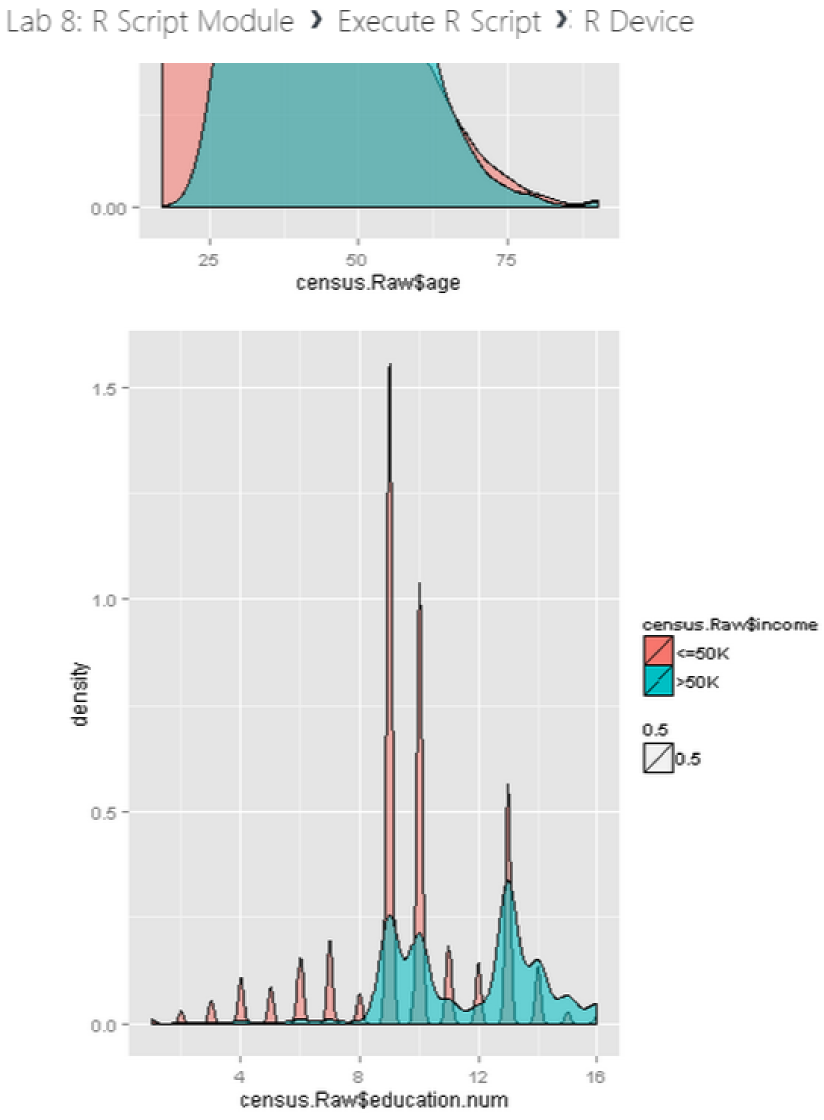
Lab 8: R Script Module › Execute R Script › R Device

Standard Output

```
RWorker pushed "port1" to R workspace.
Beginning R Execute Script
```

Standard Error

Graphics Device



Lab 8: R Script Module › Execute R Script › R Device

Exercise 7: Reading R Scripts from the R Module

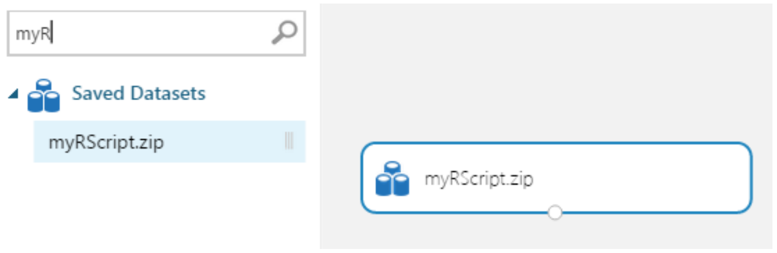
We will be creating our own R script to be used in conjunction with the R module.

1. Create a new text file in notepad (or notepad++ or any IDE you prefer).
2. Insert the following code into the text file.
3. summary() provides summary statistics of the data. Median, interquartile ranges, max, etc.

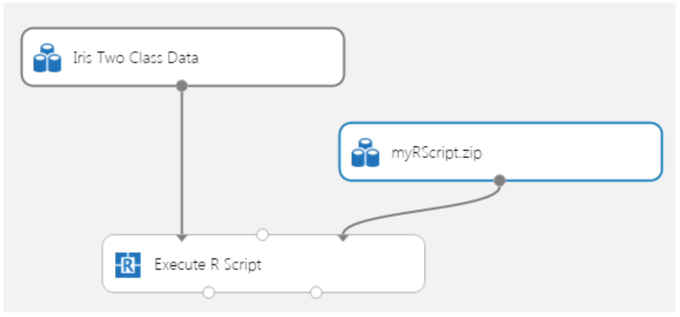
```
myPrt <- function(x){summary(x)}
```

   a. The summary function is wrapped around a function and can be called with myPrt(x), with X as the input parameter of the data that we want to summarize.

4. Save the text file as an R file named: **myRscript.R**
5. Zip the R file into **myRscript.zip**.
6. Import the zip file as a new dataset. There will be an option to select ZIP.
7. Find the myRscript.zip file from your saved datasets and drag it into your workspace.

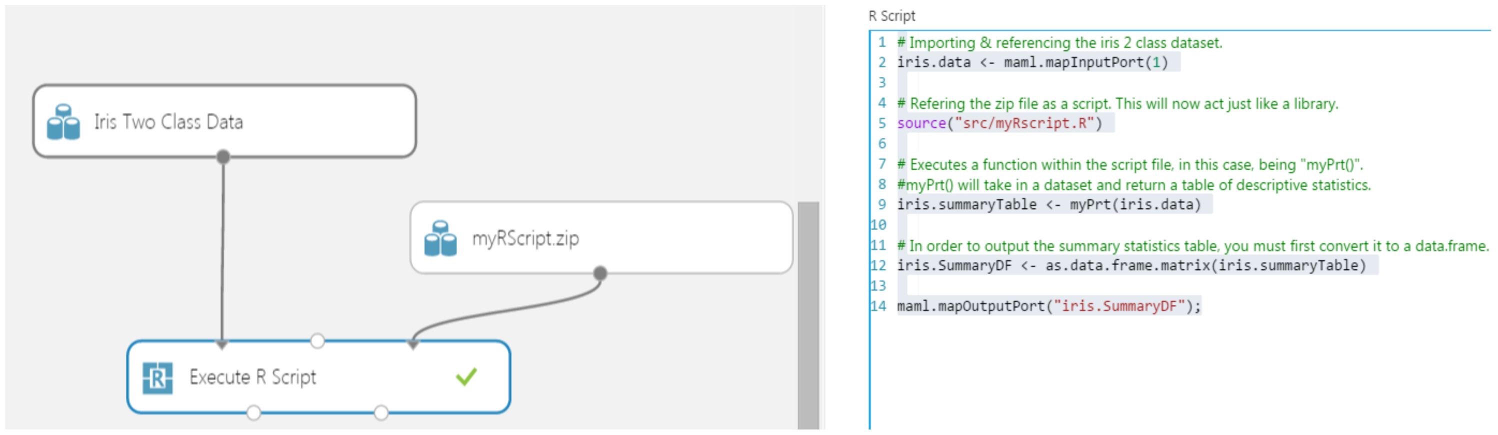


8. Drag an R Script Module and the Iris Two Class Data.
   a. Connect the Iris to the left input node of the R script module. Connect the myRscript.zip to the right input node of the R script module.



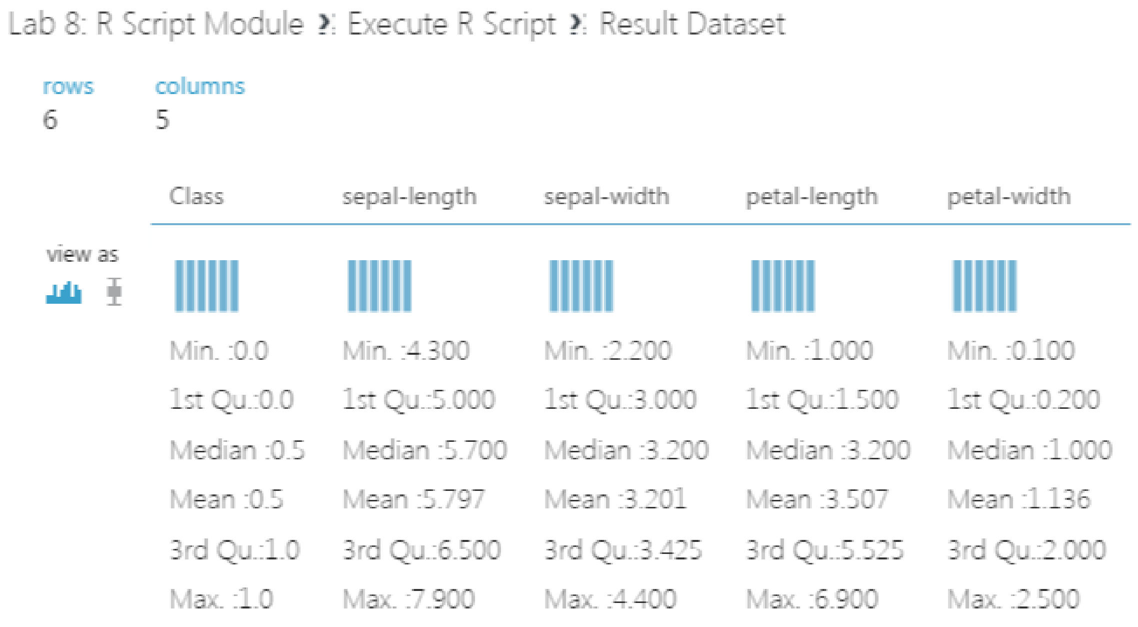
9. Insert the following code into the R script module.

```
# Importing & referencing the iris 2 class dataset.
iris.data <- maml.mapInputPort(1)

# Refering the zip file as a script. This will now act just like a library.
source("src/myRscript.R")

# Executes a function within the script file, in this case, being "myPrt()".
# myPrt()takes in a dataset and return a table of descriptive statistics.
iris.summaryTable <- myPrt(iris.data)

# Converts the table to a data.frame for output.
iris.SummaryDF <- as.data.frame.matrix(iris.summaryTable)

maml.mapOutputPort("iris.SummaryDF");
```



```
# Importing & referencing the iris 2 class dataset.
iris.data <- maml.mapInputPort(1)

# Refering the zip file as a script. This will now act just like a library.
source("src/myRscript.R")

# Executes a function within the script file, in this case, being "myPrt()".
#myPrt() will take in a dataset and return a table of descriptive statistics.
iris.summaryTable <- myPrt(iris.data)

# In order to output the summary statistics table, you must first convert it to a data.frame.
iris.SummaryDF <- as.data.frame.matrix(iris.summaryTable)

maml.mapOutputPort("iris.SummaryDF");
```

10. Visualize the right output node to view the summary statistics.



Lab 8: R Script Module › Execute R Script › Result Dataset

rows 6 columns 5

view as

Class	sepal-length	sepal-width	petal-length	petal-width
Min. :0.0	Min. :4.300	Min. :2.200	Min. :1.000	Min. :0.100
1st Qu.:0.0	1st Qu.:5.000	1st Qu.:3.000	1st Qu.:1.500	1st Qu.:0.200
Median :0.5	Median :5.700	Median :3.200	Median :3.200	Median :1.000
Mean :0.5	Mean :5.797	Mean :3.201	Mean :3.507	Mean :1.136
3rd Qu.:1.0	3rd Qu.:6.500	3rd Qu.:3.425	3rd Qu.:5.525	3rd Qu.:2.000
Max. :1.0	Max. :7.900	Max. :4.400	Max. :6.900	Max. :2.500