

Azure Machine Learning Studio

Course Material

Part 2: Additional Exercises

Contents

Extra Lab 1: Build a Machine Learning Model out of the Iris Data..... 3

 Exercise 1: Reading the data 3

 Exercise 2: Preprocessing..... 3

 Exercise 3: Build a Machine Learning Model..... 4

 Exercise 4: Evaluating Your Model..... 5

Extra Lab 2: Entity Extractor 6

 Exercise 1: Creating the Entity Extractor Model..... 7

 Exercise 2: Modifying the Output in R 8

Extra Lab 3: Interpreting Attribute Significance Using Linear Regression..... 11

 Exercise 1: Building a Modified Linear Regression Model for Data Analysis 11

 Exercise 2: Obtaining Insight from Your Model..... 12

Extra Lab 4: More R in Azure ML 13

 Extra Exercise 1: Hello World..... 13

 Extra Exercise 2: Pie Charts 14

 Extra Exercise 3: Feature Engineering with Azure ML & R 16

Extra Lab 1: Build a Machine Learning Model out of the Iris Data

Exercise 1: Reading the data

Read the data from UCI machine learning repository:

Data: <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>

MetaData: <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.names>

- Read the documentation on the page and familiarize yourself with the data.
- How would you read data from this location?
- Does the data have headers? If not, what do you think is needed?
- Are the columns casted properly? If not, which should be casted? Why does it need to be casted?
- Explore the data. Try to visualize the data.
- How would you get boxplots, histograms, and scatterplots?
- How do you segment boxplots by different flower types?

Exercise 2: Preprocessing

Explore the data and consider the following questions for preprocessing:

- Are there any column names that could be named better?
- Are there missing values to consider?
- What is our response class? What do we do with it to ensure that the model interprets it correctly?
- What type of machine learning problem is this, and which algorithm should be applied in this situation?
Classification, regression, or clustering?

Do not look at the next page until you have answered these questions or at least tried to answer the questions. We want you to spend some time brainstorming on the questions above. It will help a lot. Trust us!

The predictive modeling problem in this case is a multi-class classification problem.

- We know it is a multi-class classification problem because the response class is a categorical value rather than a sequential number. We know that it is multi-class instead of two-class because there are 3 possible responses: "Iris-setosa", "Iris-versicolor", and "Iris-virginica".
- What kind of algorithm will you use for this?

Exercise 3: Build a Machine Learning Model

- Rename the columns to what has been described in #7 here: <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.names>
- Casting: The column "class" is our response class - the variable that our machine learning model will be predicting. As a result, it must be cast into a label. Class also represents categorical data because 0 and 1 are in place for a species of plant rather than a sequential number. We must cast the class column into a categorical type. The name "Class" for column is not a clear description of what this attribute contains, it should be renamed to species.
- Missing values: Each of the 4 predictor variables are numeric and would accept either the median or the mean when scrubbing missing values. The missing categorical value of "species," the response class, should be removed since there is only one missing value, which would result in a final dataset of 150 rows from 151, not numerical.

Build a machine learning model from the information above. Optimize the machine learning model. Consider the following questions as you build your model:

- Do you have to consider splitting the data? If not, why? Why would you need to split in the first place? How much data should be split?
- Is there a certain metric that should be maximized in this case? True positive rate, false positive rate, accuracy, AuC?
- Try a few different algorithms and try to compare them. How would you do that?

Do not look at the next page until you have answered these questions or at least tried to answer the questions. We want you to spend some time brainstorming on the questions above. It will help a lot. Trust us!

Exercise 4: Evaluating Your Model

- Do you have to consider splitting the data? If not, why? Why would you need to split in the first place?
How much data should be split?
 - When training a model, you should always split the data. You want to hold out some data so that you can test the model's predictive performance against data it has not seen yet. A model is no good to us if it is only optimized to predict known values because we want to test its ability to deal with the unknown.
 - In general, you should do a 70/30 split: 70% to train the data and 30% to validate and score the algorithm that was just trained.
- Is there a certain metric that should be maximized in this case?
 - This is a low risk scenario case where the false positives and false negatives are not detrimental or urgent to maximize or minimize. In general, you want to optimize for the RoC AuC for a model's overall performance when there are little to no consequences for false positives or false negatives. The RoC AuC is a good measure of the model's robustness to anticipate for unknown values.
- What algorithm did you ultimately decided to go with and why?
 - Whichever multi-class classification algorithm that yielded the highest RoC AuC
 - Try to beat your neighbors with a higher RoC AuC by tweaking your algorithm's parameters or trying different algorithms.
- Continue optimizing until you have the highest possible RoC AuC.
- Share your RoC AuC with your neighbors. How well did you do? Did you beat them? If not, ask them for tips! See which algorithms and settings they used!

Extra Lab 2: Entity Extractor

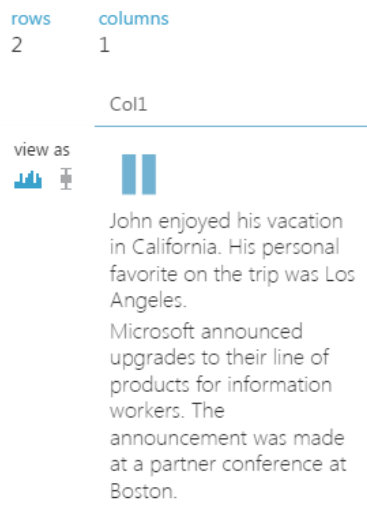
*Disclaimer: Due to the latest Azure ML update, there is a known issue with deploying this experiment. However, you can still perform the experiment for practice.

The purpose of this lab is to show you that you can do more than just machine learning models with Azure Machine Learning Studio. An entity extractor is something that can extract persons, places, or organizations from any given string. For example, here is a case where all of *Harry Potter and the Sorcerer's Stone* was pasted into the entity extractor, and these were the top results.

Entity	Type	Frequency
Harry	Person	1227
Ron	Person	382
Hermione	Person	205
Snape	Person	129
Dudley	Person	126
Neville	Person	102
McGonagall	Person	97
Hagrid	Person	89
Vernon	Person	75
Dursley	Person	52

Exercise 1: Creating the Entity Extractor Model

- 1. Create a new, blank experiment.
- 2. Drag in a "Named Entity Recognition" module under "Text Analytics".
- 3. Drag in the "Named Entity Recognition Sample Articles" under "Datasets". This is what the visualized data should contain.



- a. It contains a sample sentence to test against. We shall see if it can correctly identify "John", "California", "Los Angeles", "Microsoft", and "Boston" as entities.
- 4. Feed the Named Entity Recognition Sample Articles dataset into the Named Entity Recognition module's left input node.
- 5. Visualize the data.

The screenshot shows a data visualization interface displaying a table of entity recognition results. At the top, it indicates 'rows: 5' and 'columns: 5'. To the left, there is a 'view as' section with a bar chart icon selected. The table has the following columns: Article, Mention, Offset, Length, and Type. The data rows are as follows:

Article	Mention	Offset	Length	Type
0	John	0	4	PER
0	California	29	10	LOC
0	Los Angeles	79	11	LOC
1	Microsoft	0	9	ORG
1	Boston	133	6	LOC

- a. Not only did it correctly identify the entities, but it quantified their type, offsets, and length.

Exercise 2: Modifying the Output in R

Azure ML can only return one line at a time. However, the named entity recognition outputs a data set. Therefore, submitting this as a dataset would only return the first row, being "0, John, 0, 4, PER" which is not meaningful in the grand scheme of things. Therefore, we must parse the entirety of the dataset into one long string separated by commas.

1. Drag in an R Script module and insert the following code.

```
# Map 1-based optional input ports to variables
# class: data.frame
dataset1 <- mam1.mapInputPort(1)

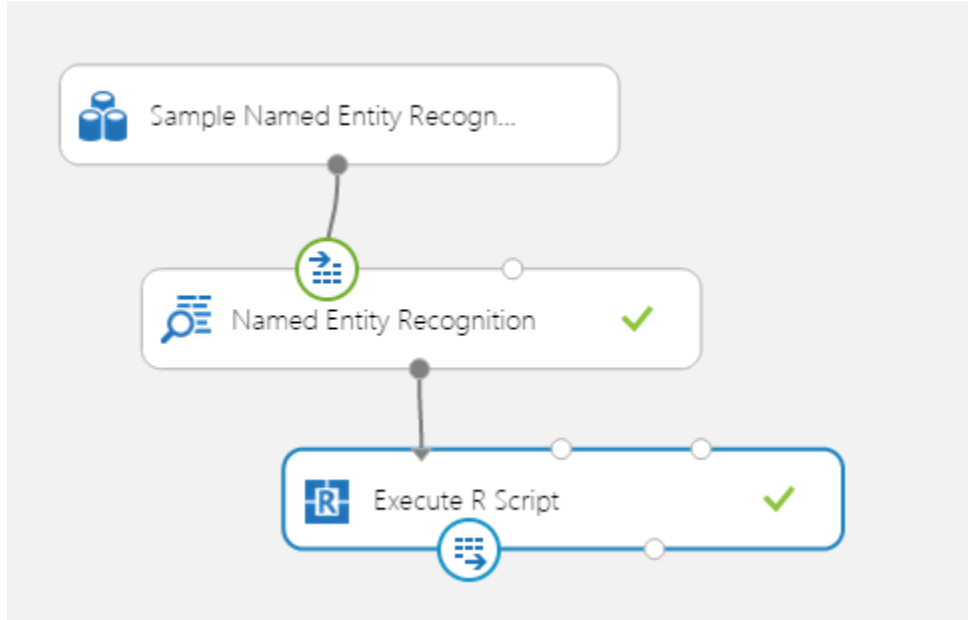
a <- paste(dataset1[,1], dataset1[,2], dataset1[,3], dataset1[,4],
dataset1[,5], sep =", ")
len <- length(a)
b <- ""
for(i in 1:len){
  if(nchar(b)>0){
    b <- paste(b, a[i], sep=", ")
  } else {
    b <- a[i]
  }
}
data.set <- data.frame(b)
# Select data.frame to be sent to the output Dataset port
mam1.mapOutputPort("data.set");
```

- a. This script will do the parsing of the dataset into a string separated by commas.

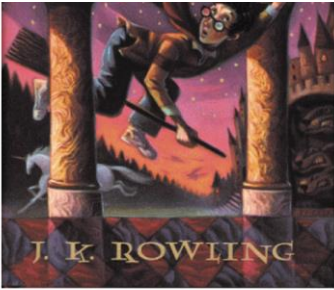
R Script

```
1 # Map 1-based optional input ports to variables
2 # class: data.frame
3 dataset1 <- mam1.mapInputPort(1)
4
5 a <- paste(dataset1[,1], dataset1[,2], dataset1[,3], dataset1[,4], dataset1[,5], sep =", ")
6 len <- length(a)
7 b <- ""
8 for(i in 1:len){
9   if(nchar(b)>0){
10     b <- paste(b, a[i], sep=", ")
11   } else {
12     b <- a[i]
13   }
14 }
15 data.set <- data.frame(b)
16 # Select data.frame to be sent to the output Dataset port
17 mam1.mapOutputPort("data.set");
```


2. Publish the web service and try to paste in various items such as CNN articles or book pages.
 - a. The test input form has a character limit, so it would prevent you from pasting in an entire book. However, if you were to make a "POST" call to your web service, there is no string limit, thus you can paste the entirety of books such as Harry Potter.



3. Try pasting in a few pages of Harry Potter into your entity extractor.
4. <http://www2.sdfi.edu.cn/netclass/jiaoan/englit/download/Harry%20Potter%20and%20the%20Sorcerer%27s%20Stone.pdf>



Test Lab 7 Entity Extractor Service

Enter data to predict

COL1

Harry Potter and the Sorcerer's Stone

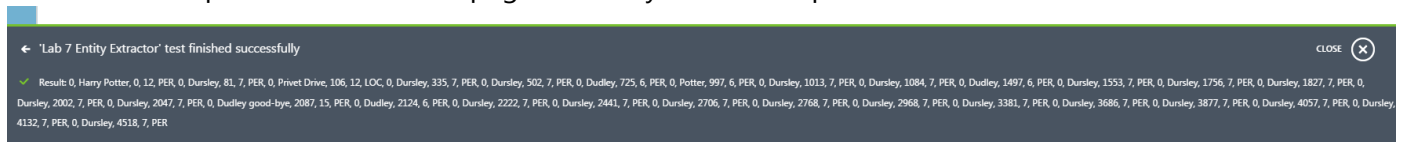
CHAPTER ONE

THE BOY WHO LIVED

Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense.

Mr. Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large mustache. Mrs. Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she

5. Here's an example where the first 2 pages of Harry Potter was pasted.



6. We have exported this Azure ML project and wrapped it into a Django web-app. In this App, you will be able to paste all of Harry Potter. Play around with the app and paste various things such as wikipedia articles or entire CNN articles.

Visit: <http://demos.datasciencedojo.com/demo/entityextractor>



Text Entity Extractor

Text Entity Extractor uses tags to classify elements in text into predefined categories. It can accurately find the names of people, places, and organizations in large bodies of text. It can also find product names, expressions of time, monetary values, and more. Text extracting is used to highlight important pieces of information so you can establish meaning in a body of text without even reading it.

Extract Entities from your Text

Insert text to analyze and extract here:

Sample Text:
John enjoyed his vacation in California. His personal favorite on the trip was Los Angeles. Microsoft announced upgrades to their line of products for information workers. The announcement was made at a partner conference at Boston.

Clear Table Reset Submit

Graph Details Table Summary Table

Show 10 entries

Search:

Entity	Type	Frequency
No data available in table		

Showing 0 to 0 of 0 entries

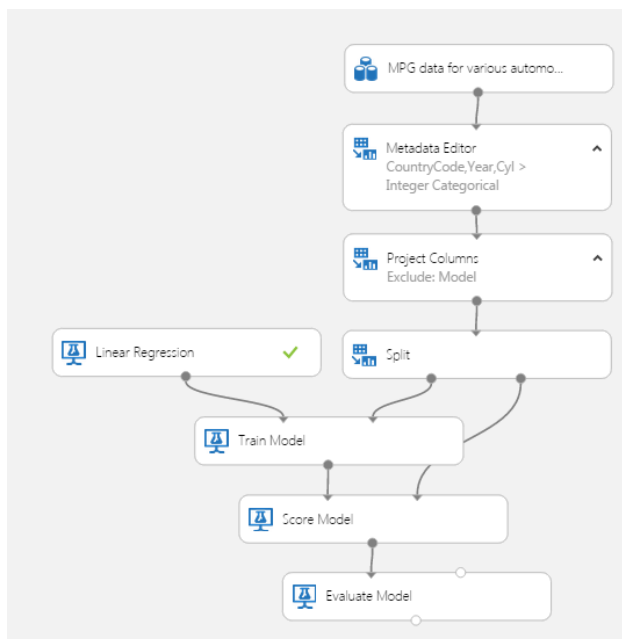
Previous Next

Extra Lab 3: Interpreting Attribute Significance Using Linear Regression

This lab shows you that you can use Azure ML to not only give predictions but also adapt it as a tool for data analysis.

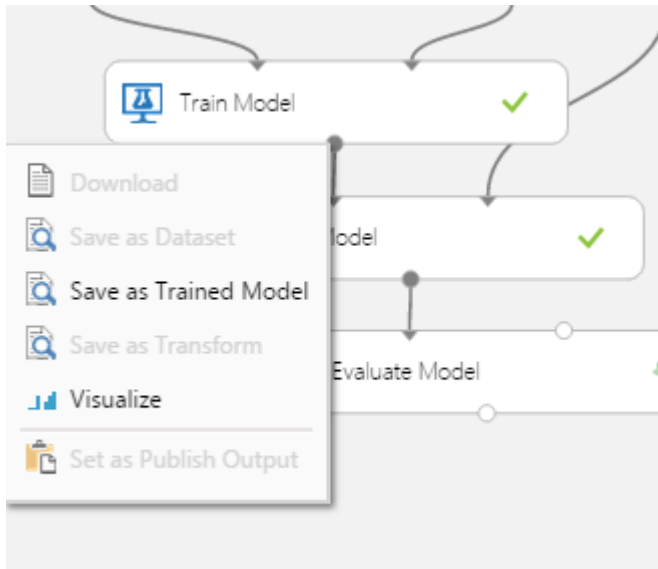
Exercise 1: Building a Modified Linear Regression Model for Data Analysis

1. Drag in the **"MPG data for various automobiles"** dataset.
2. Review the dataset's attributes here: <https://archive.ics.uci.edu/ml/datasets/Auto+MPG>
3. Cast "CountryCode", "Year", and Cyl" into "Integers" **and** "Categorical" using a metadata editor.
 - a. For predictive modeling, you normally would not convert "cyl" or "year" into categorical values because these are discrete values. However, since this is an analysis exercise and not a predictive exercise, we want to see the full significance that each extra engine cylinder would contribute to the MPG. Therefore, we want the model to treat each cylinder number independent from each other.
4. Drop the "Model" column using a project columns module.
5. Split your data into 70/30 partition with a Split module.
6. Train, Score, and Evaluate your data with a **Linear Regression** algorithm.
 - a. Perform fine tuning if necessary.
7. Final result:



Exercise 2: Obtaining Insight from Your Model

1. Visualize your “Train Model” module when you are happy with the performance of your model.
 - a. You should get a window called “Online Gradient Descent Regressor”:



Online Gradient Descent Regressor

Settings

Setting	Value
Normalize Features	true
Averaged	true
Learning Rate	0.1
Num Iterations	10
Decrease Learning Rate	true
L2 Regularizer Weight	0.001
Allow Unknown Levels	true

Feature Weights

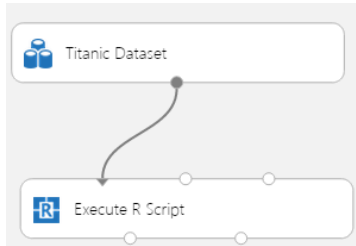
Feature	Weight
Bias	12.4003
Cyl_3_0	0.29717
Cyl_4_1	9.14634
Cyl_5_2	0.677511
Cyl_6_3	2.28852
Cyl_8_4	-0.162533
Cyl#unknown__6	0

2. Use the table called “**Feature Weights**” to interpret your data and answer the questions below.
 - a. Linear regression follows the linear regression equations such as $(a*x + b*y + c*z + e)$.
 - i. The weights represent the incremental contribution to the final result.
 - ii. For example:
 1. Horsepower -0.606535
 2. For every 1 increase in horse power, you lose 0.6 MPG from your vehicle.
 3. For every 1 decrease in horse power, you gain 0.6 MPG.
 4. $(a*x)$ in this case would be $-0.606535(\text{\# Horse Power})$
 - iii. Now apply this interpretation over all the attributes.
 - iv. Are there any relationships that surprise you? Inversely correlated or positively correlated?
 - b. How many cylinders do you want your car to have if you want it to have the highest possible MPG?
 - c. In general, which year did car manufacturers build the most fuel efficient cars?
 - d. As a car manufacturer and the given data, what three features would you pump the most R&D into in order to maximize a car’s MPG?
 - e. As a consumer buying a fuel efficient car, what kind of specs will you be looking at?
3. What specs would you identify as marketing “fluff” when buying a fuel efficient car?

Extra Lab 4: More R in Azure ML

Extra Exercise 1: Hello World

1. Drag in the Titanic dataset.
2. Connect the Titanic dataset to the left input node of the R Script module.



```
dataset1 <- maml.mapInputPort(1)
data.set <- data.frame(response=paste0("Hello ",dataset1$Name,"!"))
maml.mapOutputPort("data.set");
```

3. Insert the following code into the R Script Module:
 - a. Please note that the quotation marks MUST be vertical quotation marks, not slanted quotation marks.

Execute R Script

R Script

```
1 dataset1 <- maml.mapInputPort(1)
2 data.set <- data.frame(response=paste0("Hello ",dataset1$Name,"!"))
3 maml.mapOutputPort("data.set");
```

- b. Notice that dataset1 refers to the Titanic table because we assigned it to a variable called 'dataset1'.
 - c. Calling dataset1\$Name calls the 'Name' column from the Titanic dataset.
 - d. Paste0() is one form of string concatenation in R. Merging "Hello" and <PersonName> together.
4. Run the experiment and visualize the output.
 - a. It should now say hello to every person in the table!

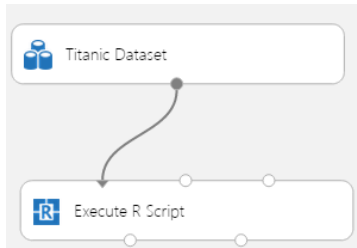
rows	columns
891	1


```
Alfred (Maria Mathilda
Gustafsson)!
Hello Ford, Mr. William
Neal!
Hello Slocovski, Mr.
Selman Francis!
Hello Fortune, Miss. Mabel
Helen!
Hello Celotti, Mr.
Francesco!
Hello Christmann, Mr. Emil!
Hello Andreasson, Mr. Paul
Edvin!
```

5. Deploy this model as a web service. Test the model in the API deployment screen to input in different strings and receive various "hello" <string> outputs, such as "Hello John".

Extra Exercise 2: Pie Charts

1. Drag in the Titanic dataset.
2. Connect the Titanic dataset to the left input node of the R Script module.



```
dataset1 <- maml.mapInputPort(1)
```

```
gender <- dataset1$Sex  
slices = table(gender)  
pie(slices)
```

```
maml.mapOutputPort("dataset1");
```

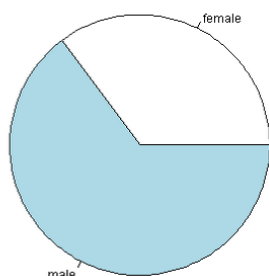
3. Insert the following code into the R Script Module:
 - a. `dataset1$Sex` retrieves the "Sex" column from the data.frame. Assign the single column row to a variable called 'gender'.
 - b. The object "gender" is currently a one-column wide data.frame, while `pie()` takes in a table parameter and not a data.frame. Let's convert gender from a data.frame to table.
 - c. `pie()` plots the data into a pie chart.

R Script

```
1 dataset1 <- maml.mapInputPort(1)  
2  
3 gender <- dataset1$Sex  
4 slices = table(gender)  
5 pie(slices)  
6  
7 maml.mapOutputPort("dataset1");  
8
```

4. Visualize the R device output, the right output node. Distribution of male to female.

Graphics Device



5. Spice up the table
 - a. Rename the labels
 - i. `lbs = c("Male", "Female")`
 - b. Add color
 - i. `colors = c("red", "blue")`
 - c. Add title
 - i. `title = "Gender Distribution"`
 - d. Put it all together.

```
dataset1 <- maml.mapInputPort(1)

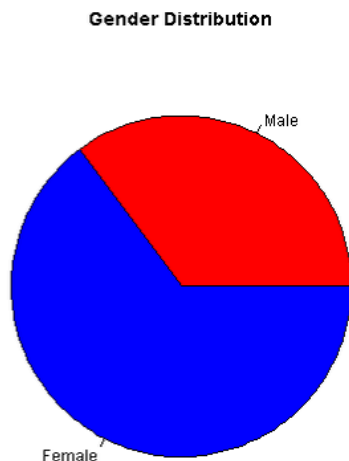
gender <- dataset1$Sex
slices = table(gender)
lbs = c("Male", "Female")
color = c("red", "blue")
title = "Gender Distribution"
pie(slices, label=lbs, col=color, main=title)

maml.mapOutputPort("dataset1");
```

R Script

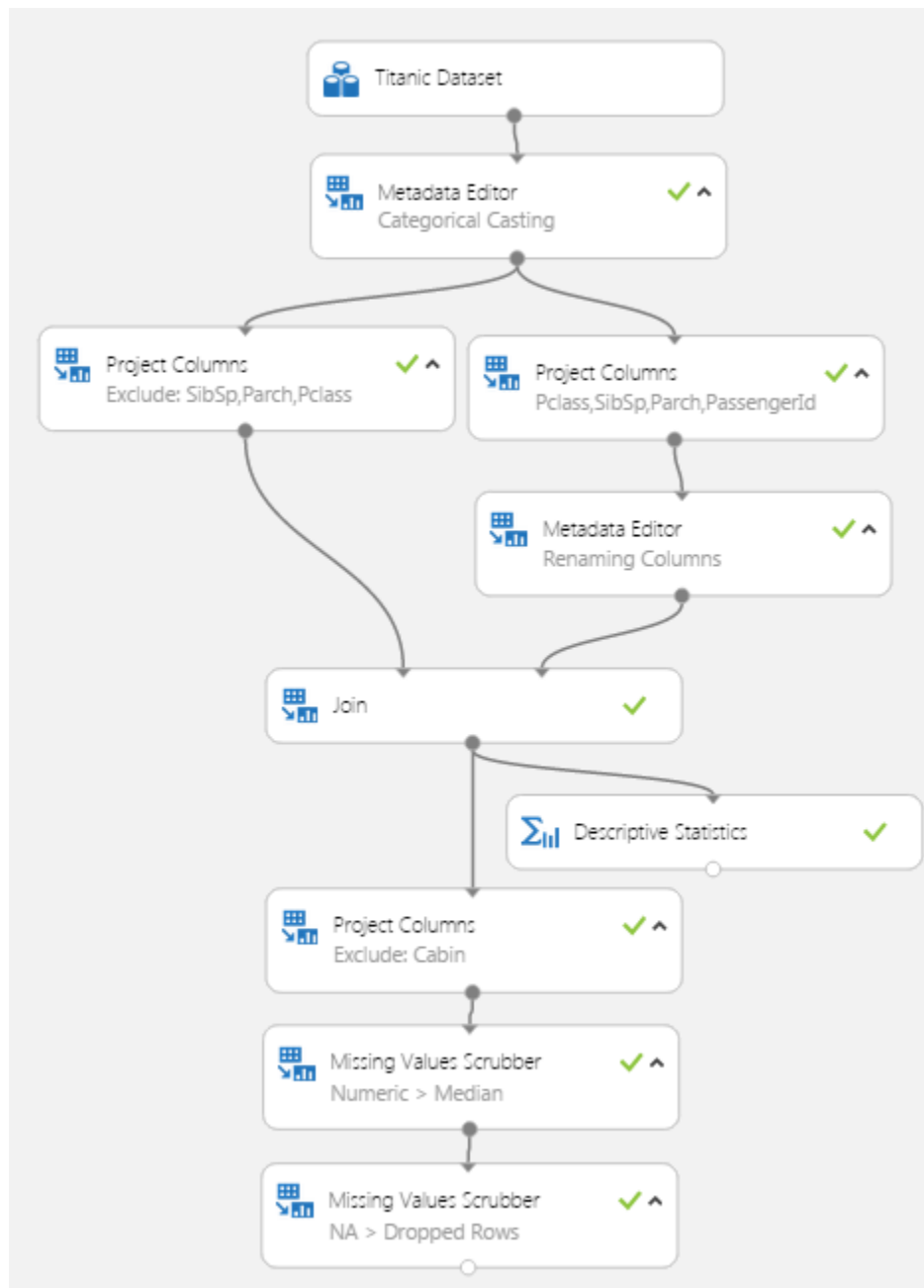
```
1 dataset1 <- maml.mapInputPort(1)
2
3 gender <- dataset1$Sex
4 slices = table(gender)
5 lbs = c("Male", "Female")
6 color = c("red", "blue")
7 title = "Gender Distribution"
8 pie(slices, label=lbs, col=color, main=title)
9
10 maml.mapOutputPort("dataset1");
11
```

Graphics Device



Extra Exercise 3: Feature Engineering with Azure ML & R

This lab will start where the lab titled “Visualizing, Exploring, Cleaning, and Manipulating Data” ended. This is what we had at the end of that lab:



Feature Engineering

Humidity + Temperature → Good/Bad Day

Age → Age Group: Child/Adult

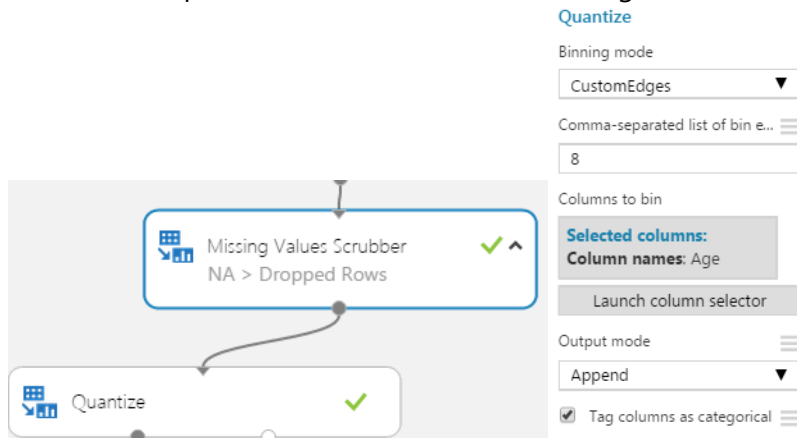
Age + Gender → IsDraftable / CanGoToWar

Years Worked → Senior/Junior Staff

Intro to Feature Engineering

Feature engineering is where data can be derived from your existing data. For example, consider the picture above. With humidity and temperature, you may decide that it's a good day or a bad day. With age, you can derive if a person is a child or an adult. In this exercise, you will be shown how to feature engineer age groups in Azure ML. In this example, we will define any person above 8 years of age to be an adult and anything equal or below as a child.

1. Differentiating Adults from Children
 - a. Drag in a "Quantize" module under Data Transformation > Scale and Reduce.
 - b. Connect the quantize module to the last missing values scrubber.



- c. Set "Binning Mode" to "CustomEdges"
 - d. Set "Comma-separated list of binned edges" to "8"
 - e. Launch column selector and include the column "Age"

f. Run the experiment and visualize the output.

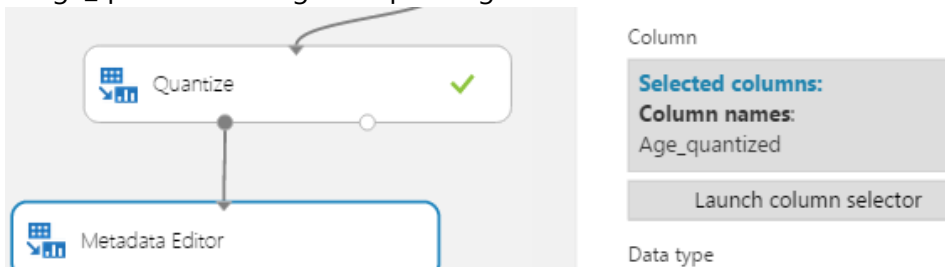
Survived	Sex	Age	Fare	Embarked	AccommodationClass	SiblingSpouse	ParentChild	Age_quantized
0	male	22	7.25	S	3	1	0	2
1	female	38	71.2833	C	1	1	0	2
1	female	26	7.925	S	3	0	0	2
1	female	35	53.1	S	1	1	0	2
0	male	35	8.05	S	3	0	0	2
0	male	28	8.4583	Q	3	0	0	2
0	male	54	51.8625	S	1	0	0	2
0	male	2	21.075	S	3	3	1	1
1	female	27	11.1333	S	3	0	2	2
1	female	14	30.0708	C	2	1	0	2
1	female	4	16.7	S	3	1	1	1
1	female	58	26.55	S	1	0	0	2
0	male	20	8.05	S	3	0	0	2

g. Notice how the 4 year old girl was casted as "1" under "Age_quantized"

h. What we did was the same thing as:

i. `if(age > 8){ return 2; } else { return 1 }`

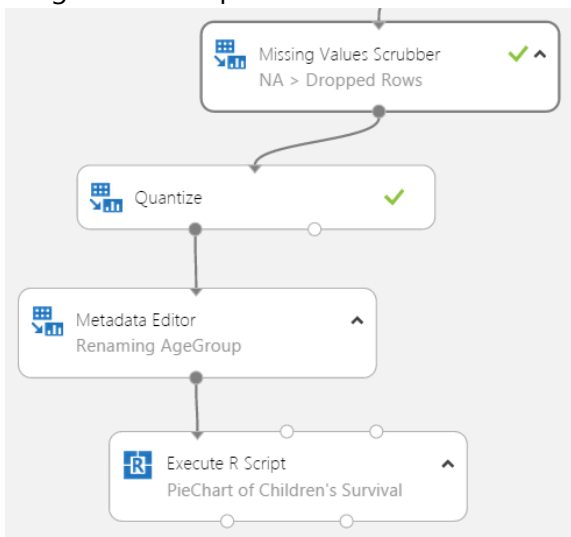
2. Rename "Age_quantize" to "AgeGroup" using the Metadata Editor



3. This dataset is now ready for machine learning!

4. Create a pie chart of children's survival in the Titanic disaster

a. Drag in an R Script module and connect it to the metadata editor

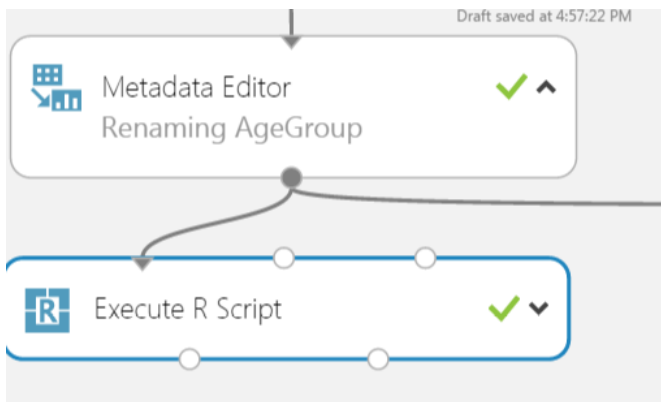


- b. Input the following code, which will create a pie chart of the new feature.

```
# Map 1-based optional input ports to variables
titanic.data <- maml.mapInputPort(1)
# Children below 8 and their survival information.
ChildrenSurvival<-titanic.data[titanic.data$AgeGroup == 1,]$Survived

# Labels for the pie chart.
childTbl <- table(ChildrenSurvival)
lbls <- cbind("Deceased", "Survived")
colors <- cbind("Red", "Green")
title <- "Children Below 8 Years Old"

# Plots and prints the survival rates of children.
childrenSurvivalPie <- pie(childTbl, lbls,col=colors,main=title)
print(childrenSurvivalPie)
```



```
R Script
1 # Map 1-based optional input ports to variables
2 titanic.data <- maml.mapInputPort(1)
3 # Children below 8 and their survival information.
4 ChildrenSurvival<-titanic.data[titanic.data$AgeGroup == 1,]$Survived
5
6 # Labels for the pie chart.
7 childTbl <- table(ChildrenSurvival)
8 lbls <- cbind("Deceased", "Survived")
9 colors <- cbind("Red", "Green")
10 title <- "Children Below 8 Years Old"
11
12 # Plots and prints the survival rates of children.
13 childrenSurvivalPie <- pie(childTbl, lbls,col=colors,main=title)
14 print(childrenSurvivalPie)
15
```

- c. Visualize the R device output from the R Script module

Graphics Device



- d. It looks like little children had a pretty high survival rate from this data.