Microsoft

Data Science and Machine Learning Essentials

Lab 2D – Cleaning, Transforming, and Integrating Data

By Stephen Elston

Overview

In this lab, you will develop skills for data cleaning, transformation and integration. Collectively this process is known among data scientists as 'data munging'. Data munging is often the most complex, time consuming and difficult part of a data science project.

Note: This lab builds on knowledge and skills developed in the previous labs. If you have little experience with using Python or R in Azure ML, and you did not complete the previous labs, you are advised to do so before attempting this lab.

What You'll Need

To complete this lab, you will need the following:

- An Azure ML account
- A web browser and Internet connection
- The lab files for this lab

Note: To set up the required environment for the lab, follow the instructions in the **Setup** document for this course. Then download and extract the lab files for this lab.

Cleaning Missing and Repeated Values

In this exercise, you will apply the **Clean Missing Data** and **Remove Duplicate Rows** modules to a data set that contains data on over 101,000 admissions and readmissions of diabetic patients at 130 US hospitals. Many algorithms will not accept missing values, so some action is often required when this is the case. Duplicated or repeated rows can cause an imbalance in the cases in a data set, which adversely affect the performance of many machine learning algorithms.

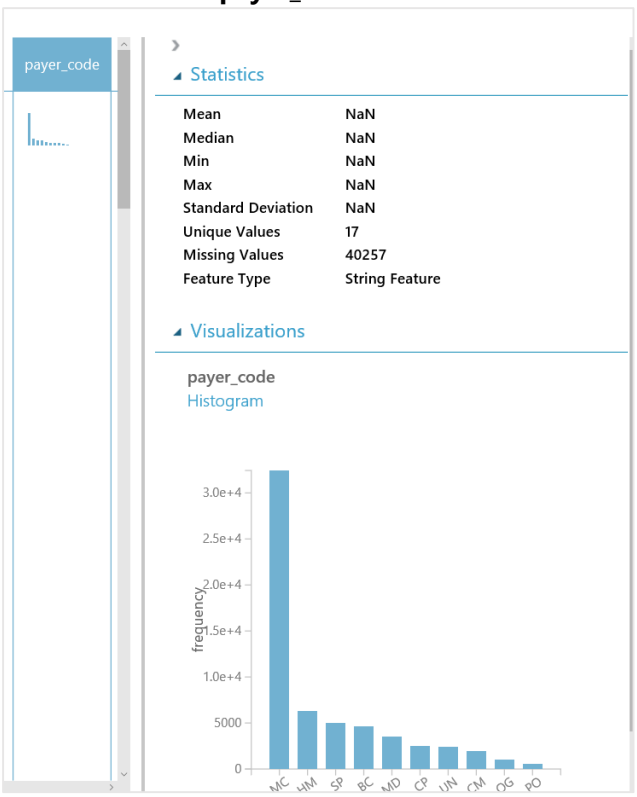
Note: The data used in this exercise was obtained from the University of California machine learning repository.
Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Create an Experiment

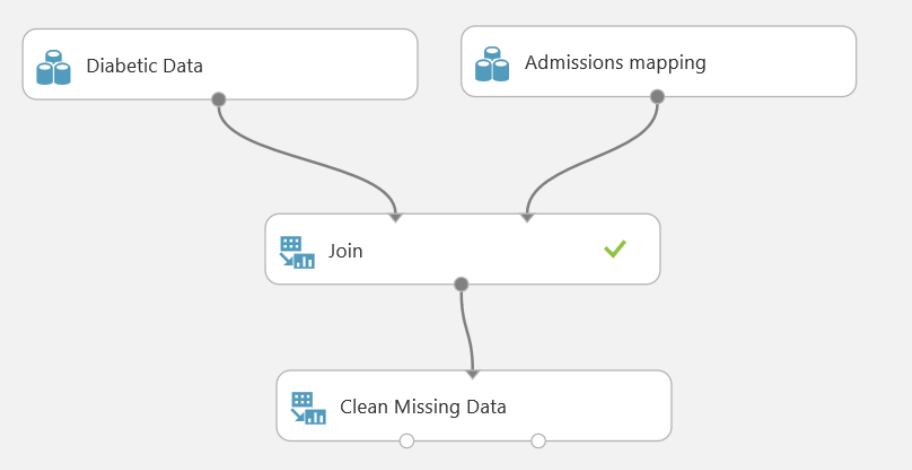
1. Open a browser and browse to https://studio.azureml.net. Then sign in using the Microsoft account associated with your Azure ML account.
2. Open the **Diabetes Data** experiment you created in Lab 2A. If you did not complete Lab 2A, you can copy the Diabetes data experiment from the collection for this course in the Cortana Analytic Library at http://gallery.cortanaanalytics.com/Collection/5bfa7c8023724a29a41a4098d3fc3df9.
3. Save a copy of the experiment as **Diabetes Data (Cleansed)**.

Configure and Run the Missing Data Module

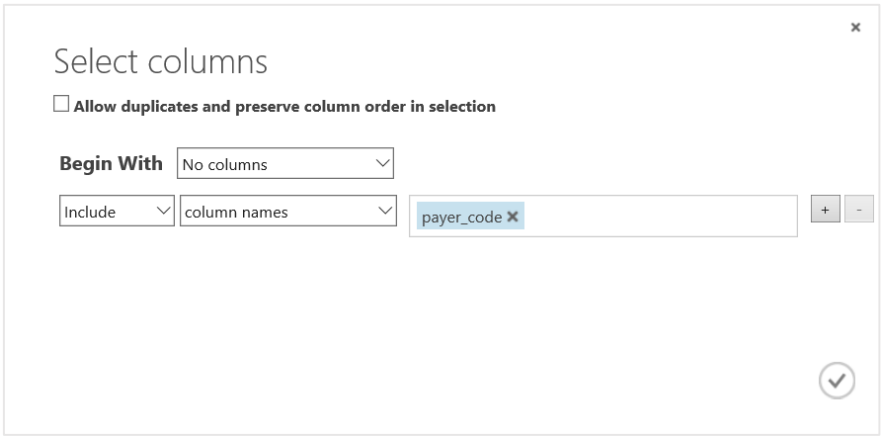
1. In the **Diabetes Data (Cleansed)** experiment, visualize the **Dataset** result dataset of the **Join** module, and view the statistics for the **payer_code** column as shown in the following image.



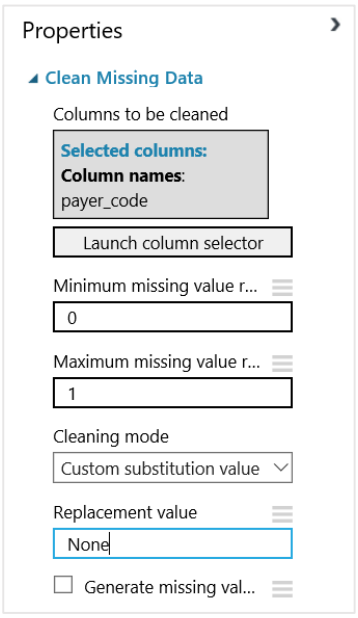
2. Note that the **payer_code** column has a large number of missing values, and then close the dataset.
3. In the **Diabetes Data (Cleansed)** experiment, search for the **Clean Missing Data** module and drag it to the canvas under the **Join** module. Then connect the output from the **Join** module to the input port of the **Clean Missing Data** module. Your experiment should look similar to the following image.



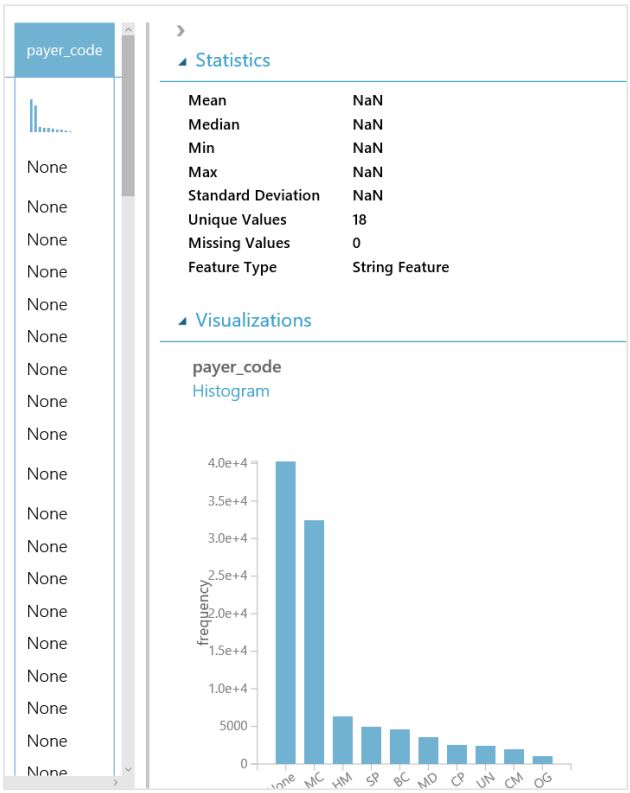
4. Select the **Clean Missing Data** module and in the **Properties** pane, launch the column selector. Then select the option to begin with no columns, and select only the **payer_code** column as shown in the following image.



5. In the **Properties** pane, in the **Cleaning mode** list, select **Custom substitution value**, and in the **Replacement value** textbox, type "None" as shown in the following image.



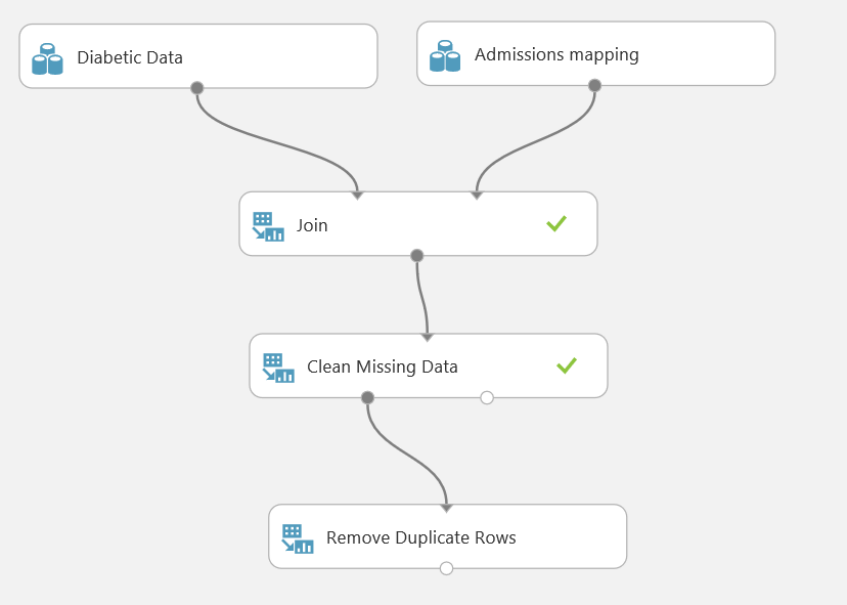
6. Save and run the experiment. Then, when the experiment has finished running, visualize the **Cleaned dataset** output of the **Clean Missing Data** module, and verify that the value "None" is now used to replace any missing **payer_code** values; as shown in the following image.



7. Note the total number of rows (indicated at the top left), and then close the cleaned dataset.

Remove Duplicate Rows

1. In the **Diabetes Data (Cleansed)** experiment, search for the **Remove Duplicate Rows** module and drag it to the canvas under the **Clean Missing Data** module. Then connect the **Cleaned dataset** output from the **Clean Missing Data** module dataset to the input port of the **Remove Duplicate Rows** module.
2. Select the **Remove Duplicate Rows** module, and in the **Properties** pane, launch the column selector to select the **encounter_id** column. This column should be unique for each row. After you have selected this column, your experiment should resemble the following image.



3. Save and run the experiment. Then, when the experiment has finished running, visualize the **Results dataset** output of the **Remove Duplicate Rows** module, and verify that the total number of rows has changed (because rows containing duplicate **encounter_id** values were removed).
4. Close the results dataset.

Filtering Outliers

In some cases, you may want to determine whether your data contains any values that deviate substantially from the norm, or *outliers*. Outliers can make the data difficult to interpret and make training a model less effective, so detecting the presence of outliers is an essential step in cleaning data for further analysis.

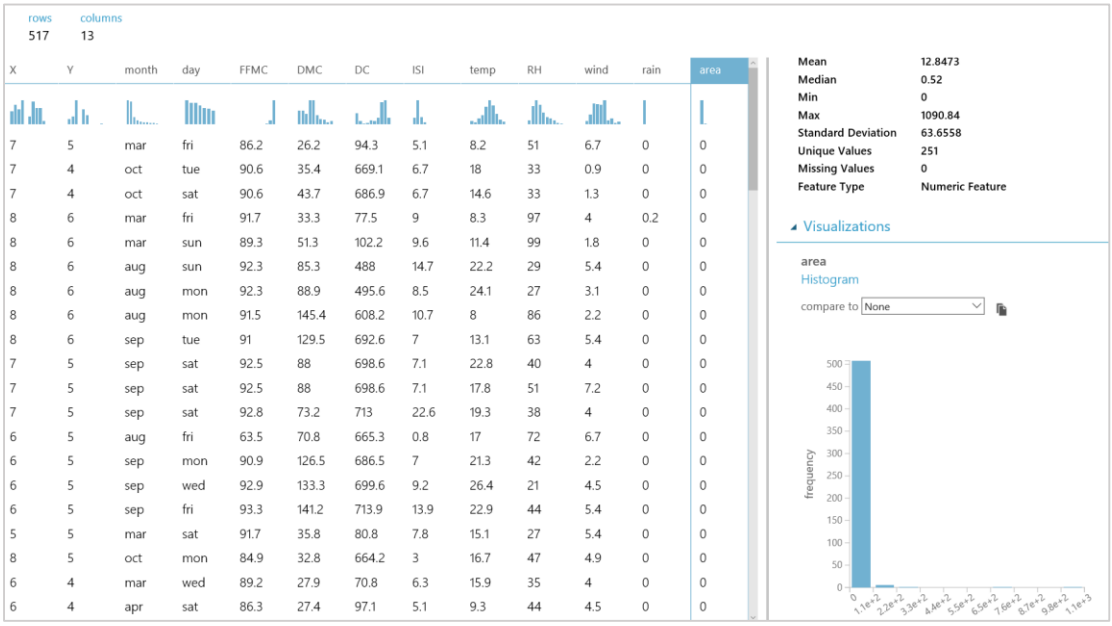
In this exercise you will examine a dataset containing information about forest fires in Portugal (which is provided as a sample dataset in Azure ML), detect the presence of outliers, and then use the **Clip Value** module in Azure ML to remove them.

Visualize Outliers

1. In your browser, in Azure ML Studio, create a new blank experiment and name it **Forest Fires Outliers**.
2. In the list of experiment items, search for the **Forest fires data** dataset, and drag it to the canvas.
3. Visualize the **dataset** output port of the **Forest fires data** dataset, and note the total number of rows it contains. Then note that the dataset contains the following columns:
    - **X** and **X**: The coordinates identifying the grid square location of the fire.
    - **month** and **day**: The month and day the fire occurred.
    - **FFMC**, **DMC**, **DC**, and **ISI**: The numerical values for *Fine Fuel Moisture Code*, *Duff Moisture Code*, *Drought Code*, and *Initial Spread Index*. These are specialist measurements used in the study of forest fires.
    - **temp**, **RH** (*relative humidity*), **wind**, and **rain**: Meteorological measurements at the time of the fire.
    - **area**: The size of the area affected by the fire.

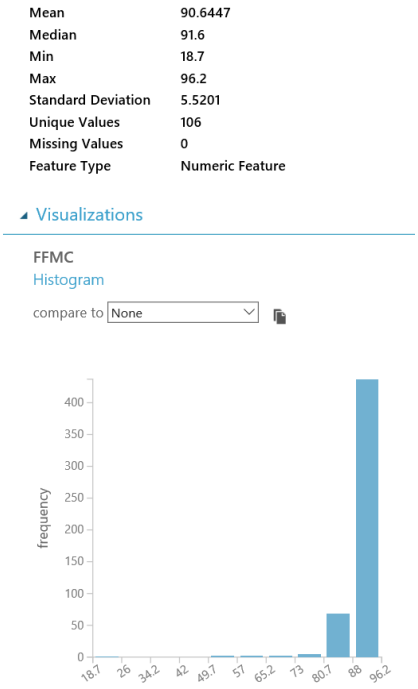
In this experiment, **area** represents the value that can be predicted by the other columns. However, before building a model, you should explore the relationships between this label column and the feature columns; and remove any outlier values that may skew the analysis.

4. Select the **area** column, and note the **Mean**, **Min**, and **Max** statistics for this column. Then look at the histogram, as shown here:



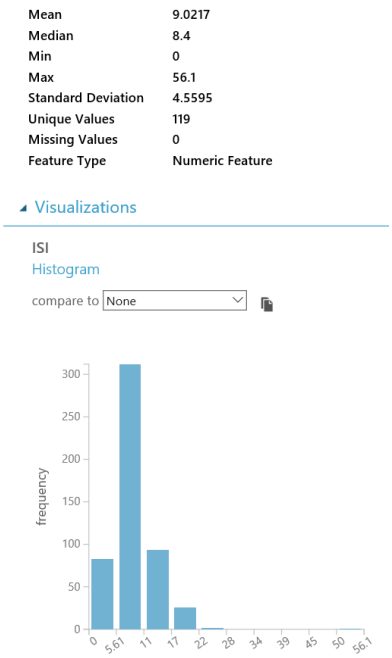
Note that the vast majority of fires are quite small – usually less than one acre; but there are also some very large fires.

5. In the list of columns, select the **FFMC** column and view the statistics for this column.

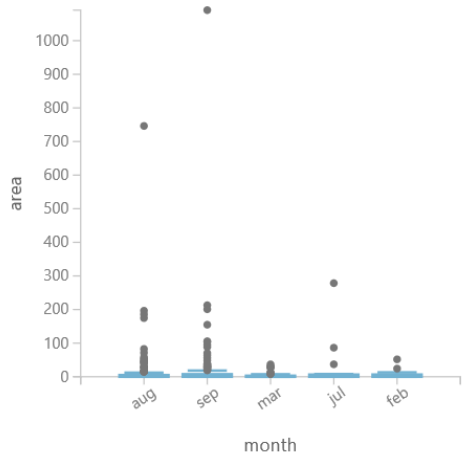


6. Note that the **Min** value for this column is significantly lower than the **Mean** and **Median** values, indicating that most **FFMC** values are high, but there are some low outliers.

7. In the list of columns, select the **ISI** column and view the statistics for this column.

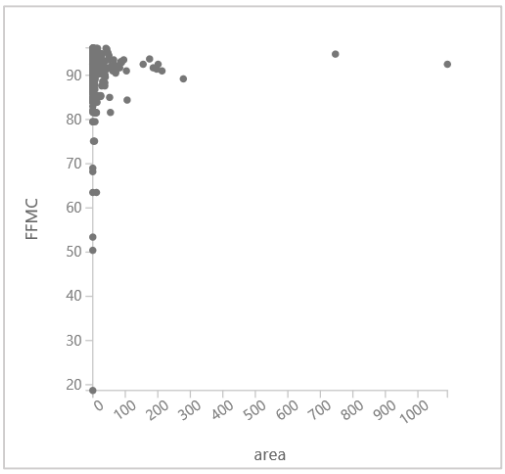


8. Note that the **Max** value for this column is significantly higher than the **Mean** and **Median** values, indicating that most **ISI** values are low, but there are some high outliers.
9. Select the **area** column again, and in the **Visualizations** area, in the **compare to** list, select **month**. This displays a scatter plot chart that shows **month** and **area** values as shown in the following image.

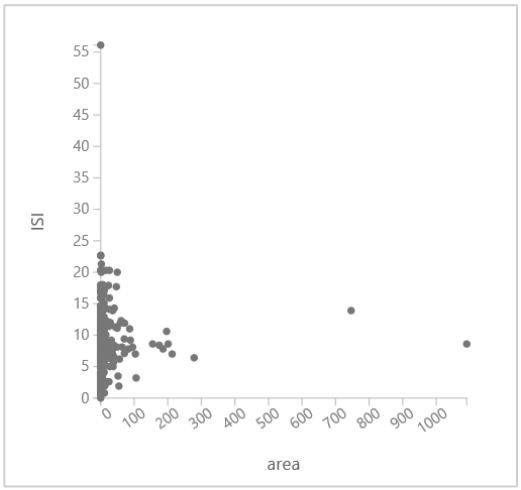


Note that while most plots are shown near the bottom of the chart, there are a few isolated plots that indicate extremely large fires during the late summer months of July, August, and September. This is not unexpected, so the outlier values for large area values have a reasonable explanation, and should remain in the dataset.

10. In the **compare to** list, select **FFMC** and view the resulting scatter plot:



11. The chart shows most plots clustered around the top left, which corresponds to high **FFMC** values and small fires. There are a few plots on the right that indicate large fires with high **FFMC** values; but because we already know that while most fires are small there are a few large ones, this seems to be reasonable. However, there are a few outlier plots that show low **FFMC** values for small fires; which could potentially indicate some erroneous data.
12. In the **compare to** list, select **ISI** and view the resulting scatter plot:



13. As with **FFMC**, the chart shows most **ISI** values are clustered together, and we can ignore the outliers to the right that indicate large fires. However, there is an outlier that show a high **ISI** value for a small **area**; which again seems unusual, and may indicate some issues with the **ISI** data.
14. Close the dataset.

Remove Outliers and Clean Missing Data

1. In **Forest Fires Outliers** experiment, search for the **Clip Values** module, drag it to the canvas under the **Forest fires data** dataset, and connect the output from the **Forest fires data** dataset to the input port of the **Clip Values** module.
2. Select the **Clip Values** module and in the **Properties** pane, set the following property values:
    - **Set of thresholds**: ClipSubpeaks
    - **Lower threshold**: Percentile
    - **Percentile number for lower threshold**: 1
    - **Lower substitute value**: Missing
    - **List of columns**: Launch the column selector and select only the **FFMC** column name.

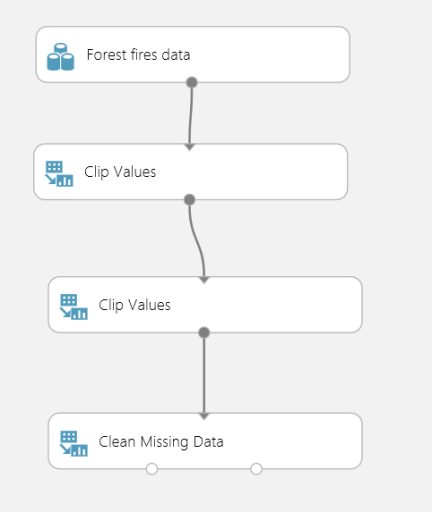
- **Overwrite flag**: Selected
- **Add indicator column**: Unselected

This removes the **FFMC** value for any rows where it is in the bottom one percentile.

3. Drag a second **Clip Values** module to the canvas under the first **Clip Values** module. Then connect the output from the first **Clip Values** module to the input port of the second **Clip Values** module.
4. Select the second **Clip Values** module and in the **Properties** pane, set the following property values:
    - **Set of thresholds**: ClipPeaks
    - **Upper threshold**: Percentile
    - **Percentile number for upper threshold**: 99
    - **Upper substitute value**: Missing
    - **List of columns**: Launch the column selector and select only the **ISI** column name.
    - **Overwrite flag**: Selected
    - **Add indicator column**: Unselected

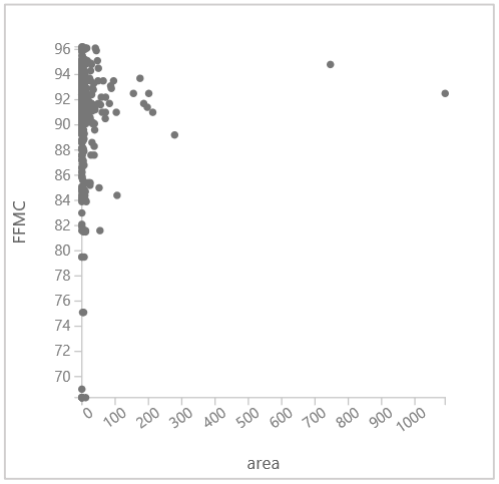
    This removes the **ISI** value for any rows where it is in the top one percentile.

5. In the list of experiment items, search for the **Clean Missing Data** module, and drag it to the canvas under the second **Clip Values** module. Then connect the output from the second **Clip Values** module to the import port of the **Clean Missing Data** module.
6. Select the **Clean Missing Data** module and in the **Properties** pane, set the following property values:
    - **Columns to be cleaned**: Launch the column selector and configure it to begin with no columns, and include only the **FFMC** and **ISI** column names.
    - **Minimum missing value ratio**: 0
    - **Maximum missing value ratio**: 1
    - **Cleaning mode**: Remove entire row
7. Verify that your experiment resembles the following image, and then save and run it.



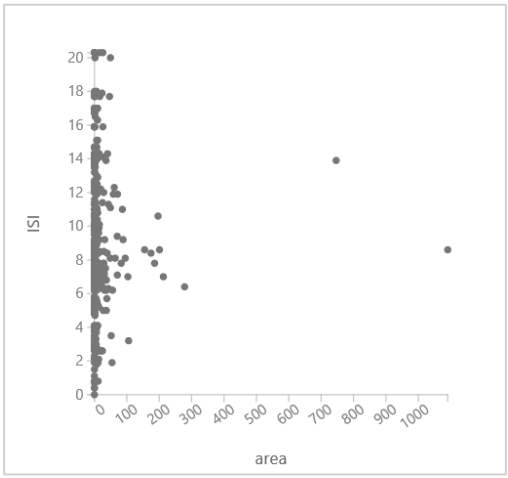
8. When the experiment has finished running, visualize the **Cleaned dataset** output of the **Clean Missing Data** module, and note the number of rows it contains. There should be fewer rows than in the source **Forest fires data** dataset because the rows containing outliers have been removed.
9. Select the **FFMC** column and note the **Mean**, **Min**, and **Max** statistics for this column. These should be higher than in the source data because low-valued outliers have been removed.

10. Select the **ISI** column and note the **Mean**, **Min**, and **Max** statistics for this column. These should be lower than in the source data because high-valued outliers have been removed.
11. Select the **area** column, and in the **Visualizations** area, in the **compare to** list, select **FFMC** and view the scatter plot chart:



Note that the very low outlier value that was previously present is no longer there.

12. With the **area** column selected, in the **Visualizations** area, in the **compare to** list, select **ISI**, and view the scatter plot chart:



Note that the outlier value that was previously present is no longer there.

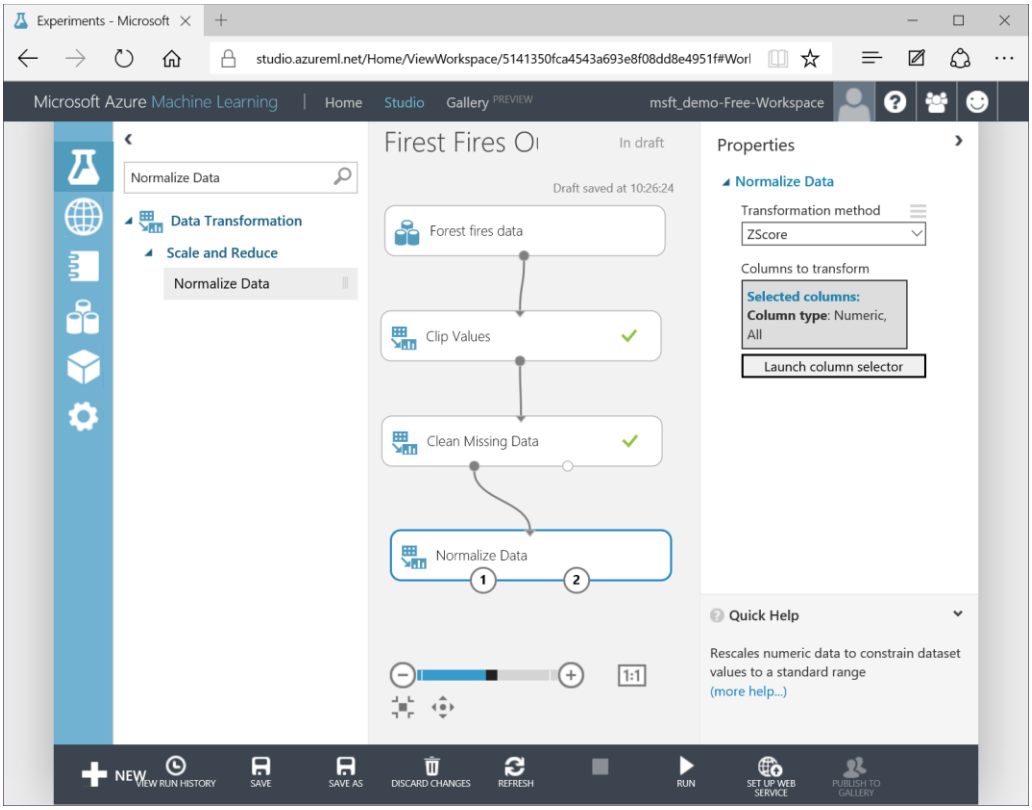
13. Close the cleaned dataset.

Scale the Data

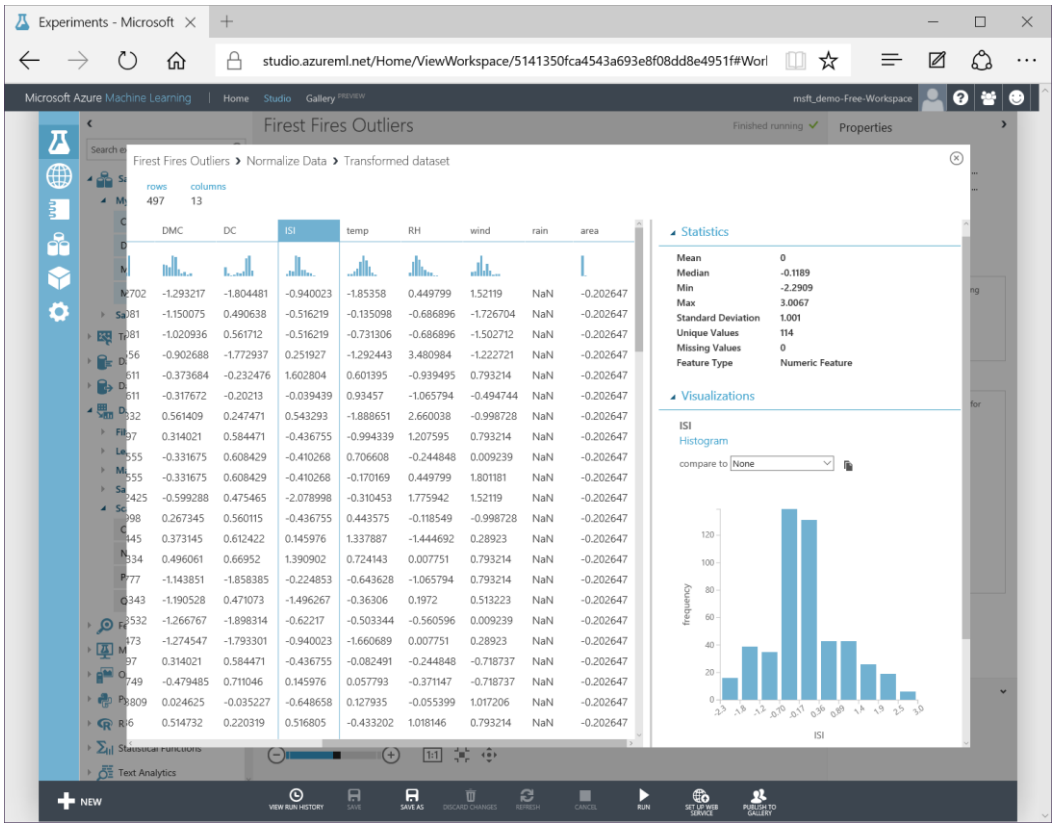
In this procedure you will use the **Normalize Data** module to scale columns of the forest fire data set. Scaling is often essential when training machine learning algorithms.

1. In the **Forest Fires Outliers** experiment, search for the **Normalize Data** module, and drag it to the canvas under the **Clean Missing Data** module. Then connect the **Cleaned dataset** output port of the **Clean Missing Data** module to the input port of the **Normalize Data** module.
2. Select the **Normalize Data** module, and in the **Properties** pane, ensure that the following properties are set:
    - **Transformation method**: ZScore (this is the default selection)
    - **Columns to transform**: All numeric columns (this is the default selection)

3. Verify that your experiment resembles the following image, and then save and run the experiment.



4. When the experiment has finished running, visualize the **Transformed dataset** output port of the **Normalize Data** module and select the **ISI** column.
5. View the statistics for the **ISI** column values and verify that they are zero-centered, with a **Mean** value of 0, as shown in the following image.



6. Close the transformed dataset.

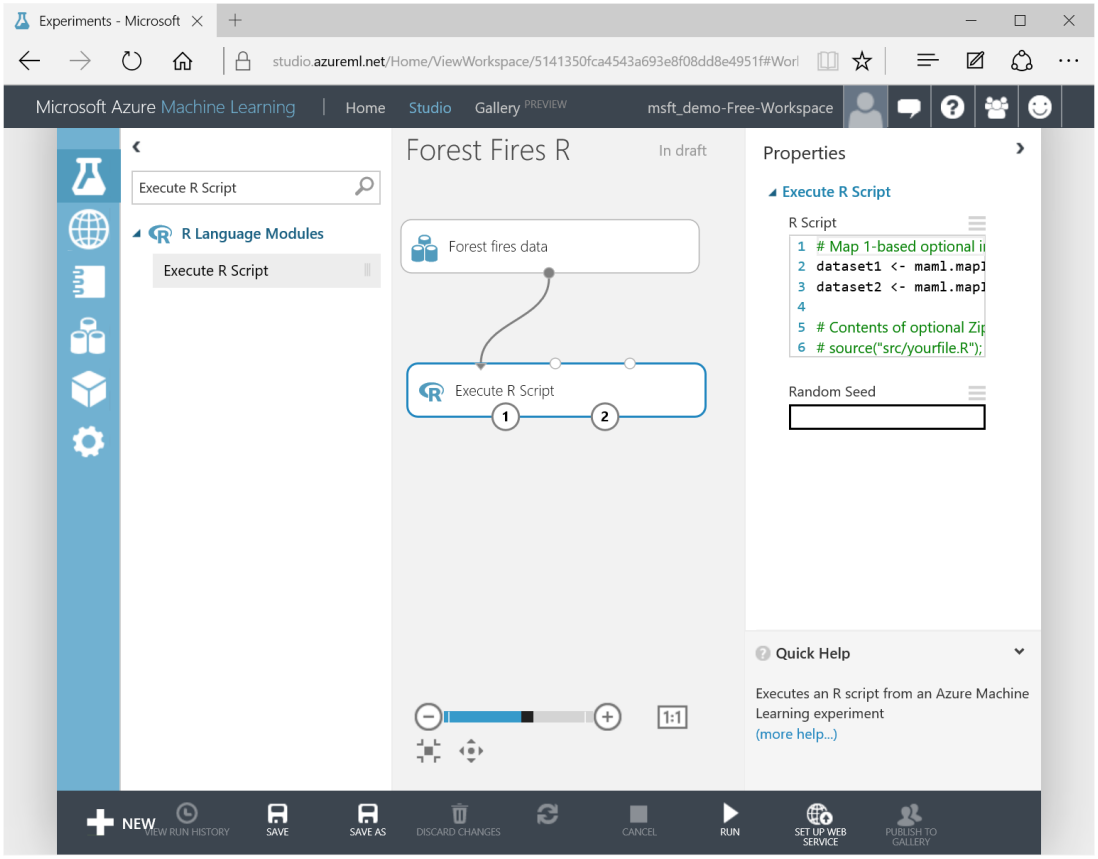
Removing Outliers with R

In this exercise, you will use a custom Python or R script to visualize outliers, and to clean and transform data.

Note: If you prefer to work with Python, skip this exercise and complete the next exercise *Removing Outliers with Python*.

Visualize Outliers with R

1. In your browser, in Azure ML Studio, create a new blank experiment and name it **Forest Fires (R)**.
2. In the list of experiment items, search for the **Forest fires data** dataset, and drag it to the canvas.
3. Visualize the **dataset** output port of the **Forest fires data** dataset, and note the total number of rows it contains. Then select the **FFMC** column, and note the **Mean**, **Min**, and **Max** statistics for this column.
4. View the **Mean**, **Min**, and **Max** statistics for the **ISI** and **rain** columns, before closing the dataset.
5. In the list of experiment items, search for the **Execute R Script** module and drag it to the canvas. Then connect the output port from the **Forest fires data** dataset to the **Dataset1** input port of the **Execute R Script** as shown in the following image.



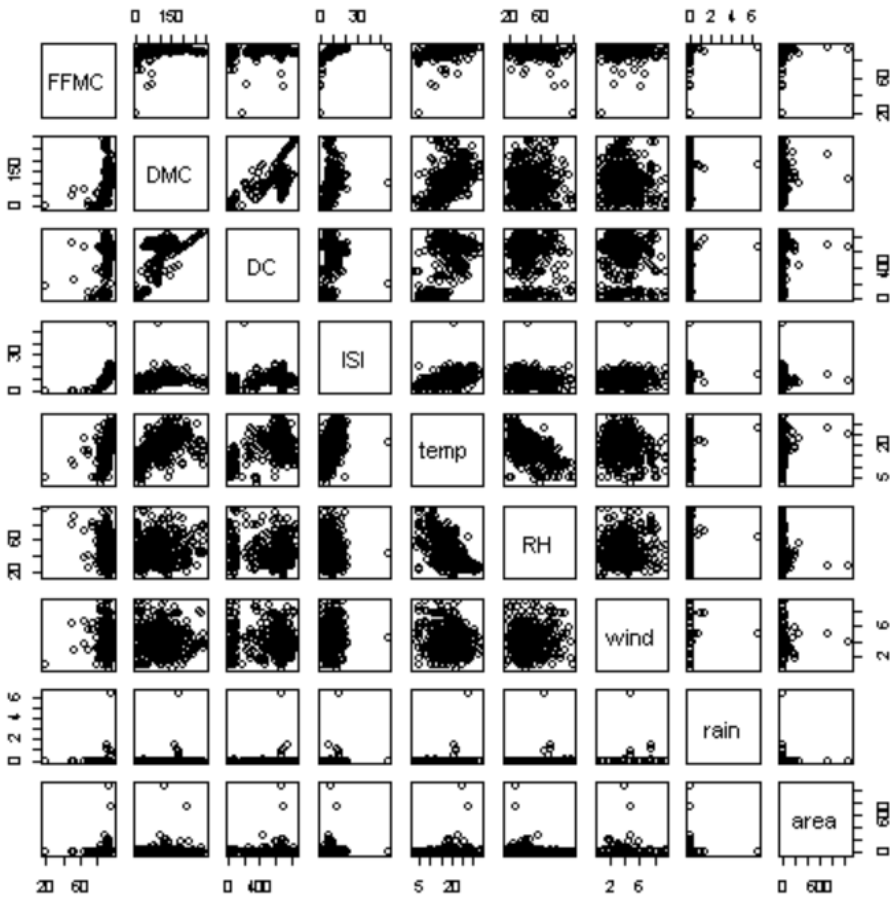
6. Select the **Execute R Script** module, and in the **Properties** pane, replace the default R script with the following code (which you can copy from the **VisualizeOutliers.R** Python script file in the folder where you extracted the lab files).

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

## remove columns
cols <- c("X", "Y", "month", "day")
dataset1 <- dataset1[, !(names(dataset1)) %in% cols]

## Create a pairs plot.
pairs(dataset1)
```

Tip: To copy code in a local code file to the clipboard, press **CTRL+A** to select all of the code, and then press **CTRL+C** to copy it. To paste copied code into the code editor in the Azure ML **Properties** pane, press **CTRL+A** to select the existing code, and then press **CTRL+V** to paste the code from the clipboard, replacing the existing code.

This code creates a scatter plot matrix (or pairs plot) visualization of the numeric columns in the dataset. You will learn more about using R to create visualizations in Module 3.

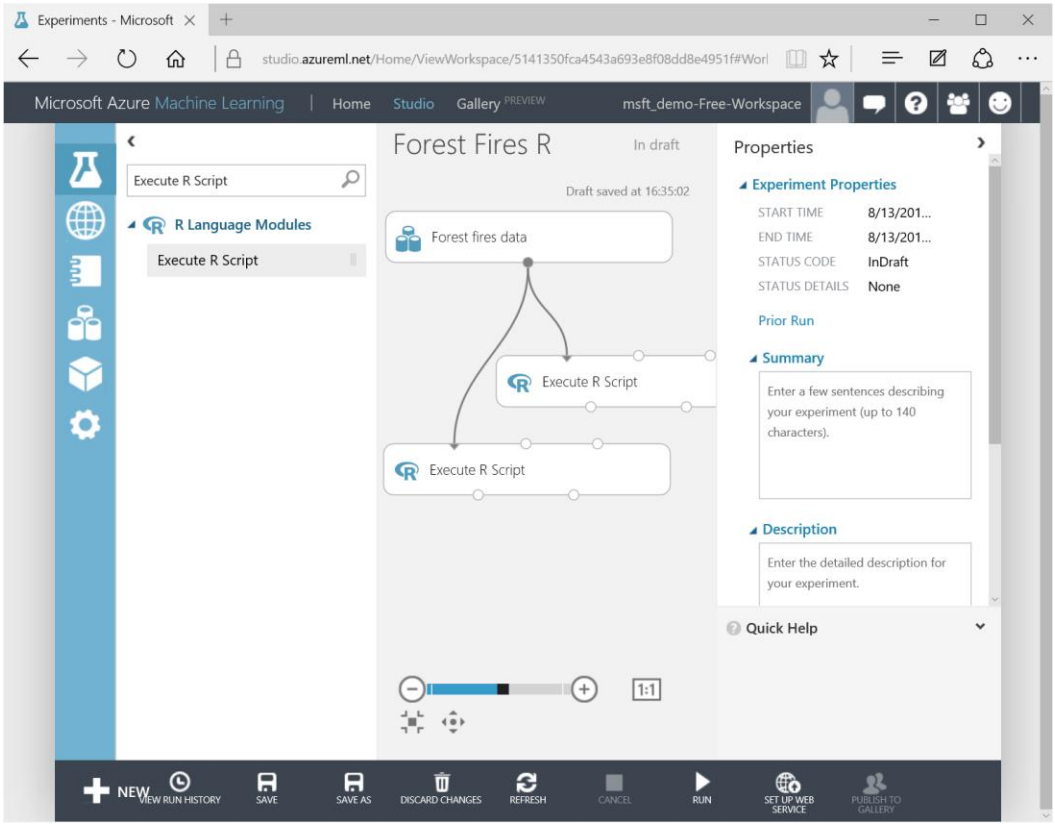
7. Save and run the experiment. Then, when the experiment has finished running, visualize the **R Device dataset** output port of the **Execute R Script** module (this is the right-most output port), and scroll down to the **Graphics Device** section to view the scatter plot chart that has been generated by the R script, which should resemble the following image.



8. Notice the significant outliers where **FFMC**, **ISI** and **rain** intersect **area**. These values do not correspond to large fires, and are clearly unusual in some way. By using R to visualize all of the columns together in a scatter plot matrix, it is easier to see outliers than when using the individual visualizations built into Azure ML datasets.
9. Close the dataset.

Remove Outliers with dplyr in R

1. In Azure ML Studio, in the **Forest Fires (R)** experiment, search for the **Execute R Script** module and drag it to the canvas next to the existing Execute R Script module. Then connect the output port from the **Forest fires data** dataset to the **Dataset1** input port of the **Execute R Script** as shown in the following image.



2. Select the new **Execute R Script** module, and in the **Properties** pane, replace the default R script with the following code (which you can copy from the **CleanForestFires.R** script file in the folder where you extracted the lab files).

```
frame1 <- maml.mapInputPort(1)

## Remove outliers
library(dplyr)
frame1 <- frame1 %>% filter(FFMC > 40) %>%
                     filter(ISI < 30) %>%
                     filter(rain < 3)

## Output the data frame
maml.mapOutputPort('frame1')
```

> **Note**: When using dplyr filtering in R, you can set finer-grained thresholds for trimming; than with the **Clip Values** module. It is also easier to trim the upper and lower values for multiple columns to different thresholds in a single R script than to add multiple **Clip Values** modules to the experiment.

3. Save and run the experiment. Then, when the experiment has finished running, visualize the **Results dataset** output port of the second **Execute R Script** module. Then view the **Mean**, **Min**, and **Max** statistics for the **FFMC**, **ISI**, and **rain** columns now that outliers have been removed.
4. Close the dataset.

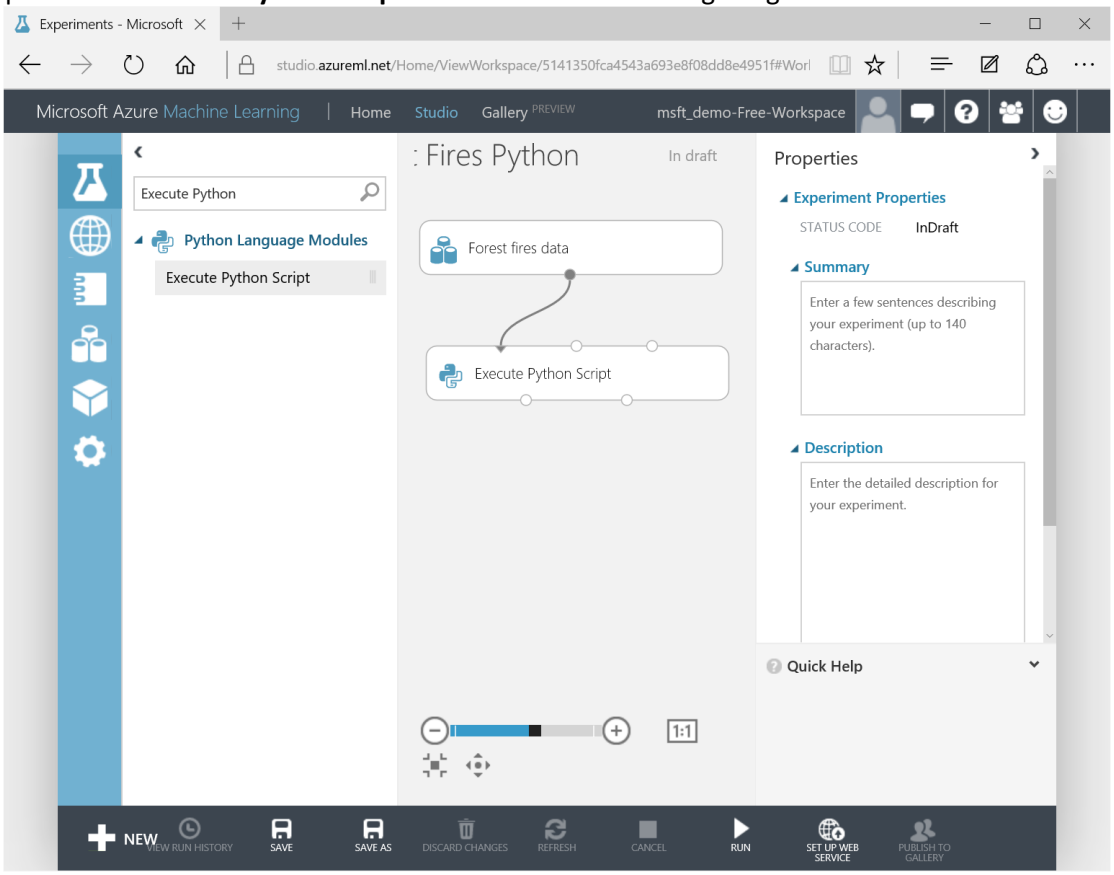
Removing Outliers with Python

In this exercise, you will use a custom Python or R script to visualize and remove outliers.

Note: If you prefer to work with R, skip this exercise and use the previous exercise *Removing Outliers with R*.

Visualize Outliers with Python

1. In your browser, in Azure ML Studio, create a new blank experiment and name it **Forest Fires (Python)**.
2. In the list of experiment items, search for the **Forest fires data** dataset, and drag it to the canvas.
3. Visualize the **dataset** output port of the **Forest fires data** dataset, and note the total number of rows it contains. Then select the **ISI** column, and note the **Mean**, **Min**, and **Max** statistics for this column before closing the dataset.
4. In the list of experiment items, search for the **Execute Python Script** module and drag it to the canvas. Then connect the output port from the **Forest fires data** dataset to the **Dataset1** input port of the **Execute Python Script** as shown in the following image.



5. Select the **Execute Python Script** module, and in the properties pane, replace the default Python script with the following code (which you can copy from the **VisualizeOutliers.py** Python script file in the folder where you extracted the lab files).

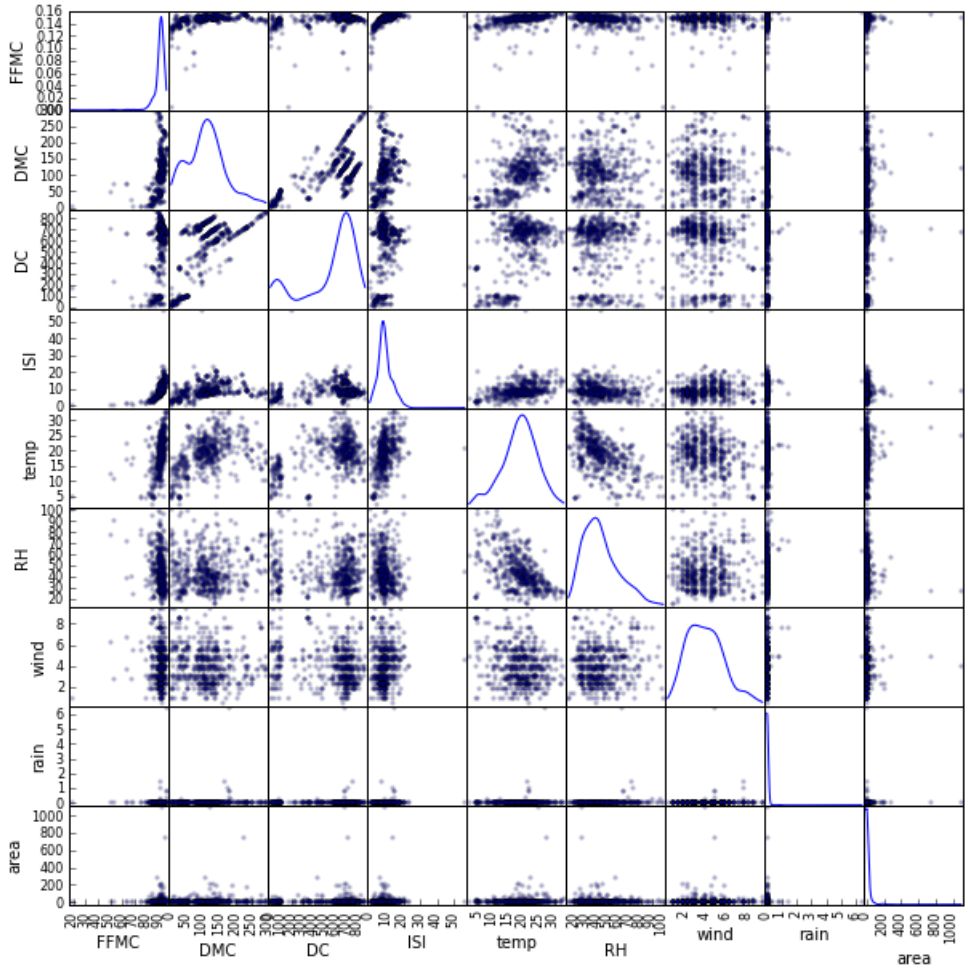
```
def azureml_main(frame1):
  import matplotlib
  matplotlib.use('agg')

  import pandas as pd
  import matplotlib.pyplot as plt
  from pandas.tools.plotting import scatter_matrix

## Remove unwanted columns
  frame1.drop(["X", "Y", "month", "day"], axis = 1, inplace = True)
```

```
## Create a scatter plot matrix
  fig1 = plt.figure(1, figsize = (12,9))
  ax = fig1.gca()
  scatter_matrix(frame1, alpha=0.2, figsize=(10, 10), diagonal='kde',
ax=ax)
  fig1.savefig('scatter2.png')

  return frame1
```

Tip: To copy code in a local code file to the clipboard, press **CTRL+A** to select all of the code, and then press **CTRL+C** to copy it. To paste copied code into the code editor in the Azure ML **Properties** pane, press **CTRL+A** to select the existing code, and then press **CTRL+V** to paste the code from the clipboard, replacing the existing code.

This code creates a scatter plot matrix visualization of the numeric columns in the dataset. You will learn more about using Python to create visualizations in Module 3.

6. Save and run the experiment. Then, when the experiment has finished running, visualize the **Python Device dataset** output port of the **Execute Python Script** module (this is the right-most output port), and scroll down to the **Graphics** section to view the scatter plot chart that has been generated by the Python script, which should resemble the following image.



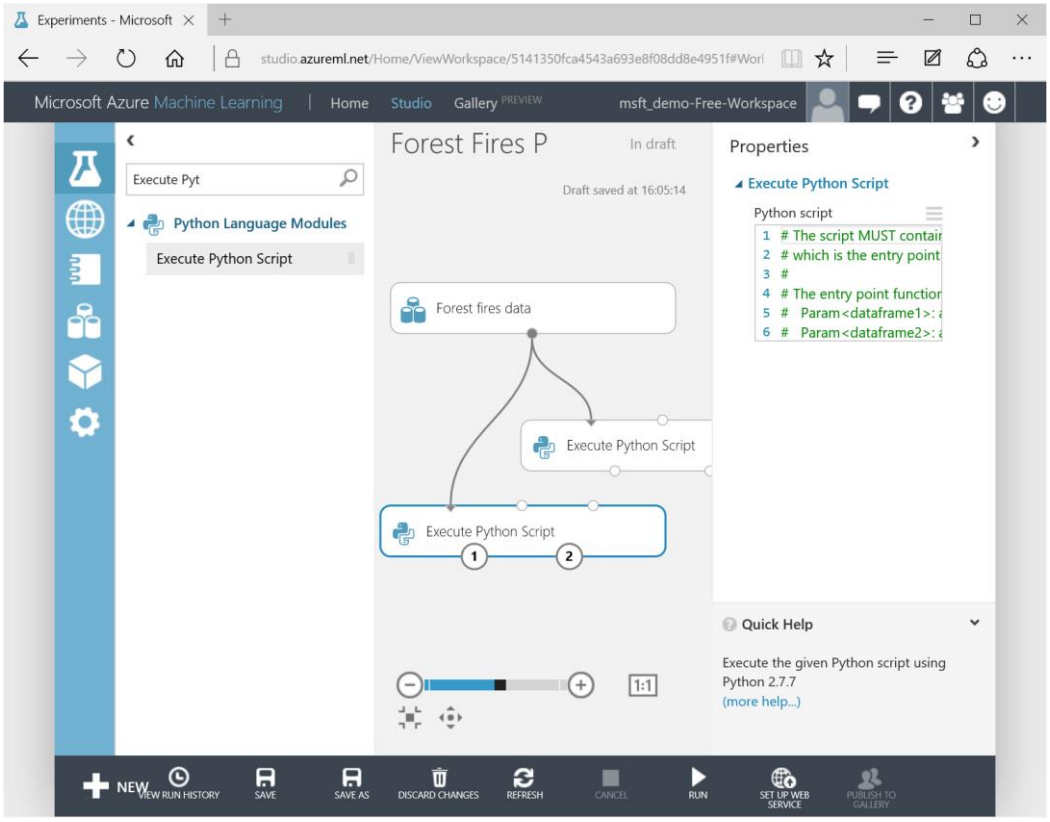
7. Notice the significant outliers where **FFMC**, **ISI** and **rain** intersect **area**. These values do not correspond to large fires, and are clearly unusual in some way. By using Python to visualize all of

the columns together in a scatter plot matrix, it is easier to see outliers than when using the individual visualizations built into Azure ML datasets.

8. Close the dataset.

Remove Outliers with pandas in Python

1. In Azure ML Studio, in the **Forest Fires (Python)** experiment, search for the **Execute Python Script** module and drag it to the canvas alongside the existing **Execute Python Script** module. Then connect the output port from the **Forest fires data** dataset to the **Dataset1** input port of the **Execute Python Script** as shown in the following image.



2. Select the new **Execute Python Script** module, and in the **Properties** pane, replace the default Python script with the following code (which you can copy from the **CleanForestFires.py** Python script file in the folder where you extracted the lab files).

```
def azureml_main(frame1):
  import pandas as pd
  import os.path

## Filter out outliers
  frame1 = frame1[(frame1["FFMC"] > 40.0) & \
                                (frame1["ISI"] < 30.0) & \
                                (frame1["rain"] < 3.0)]
  return frame1
```

Note: When using pandas filtering in Python, you can set finer-grained thresholds for trimming; than with the **Clip Values** module. It is also easier to trim the upper and lower values for multiple columns to different thresholds in a single Python script than to add multiple **Clip Values** modules to the experiment.

3. Save and run the experiment. Then, when the experiment has finished running, visualize the **Results dataset** output port of the second **Execute Python Script** module. Then view the **Mean**, **Min**, and **Max** statistics for the **FFMC**, **ISI**, and **rain** columns now that outliers have been removed.
4. Close the dataset.

Summary

In this lab, you have cleaned data to handle missing values, duplicate rows, and outliers. This kind of data cleaning is essential to build effective models that will predict labels accurately.

Note: The experiment created in this lab is available in the Cortana Analytics library at http://gallery.cortanaanalytics.com/Collection/5bfa7c8023724a29a41a4098d3fc3df9.