Data Science and Machine Learning Essentials

Lab 4C – Working with Unsupervised Learning Models

By Stephen Elston and Graeme Malcolm

Overview

The previous labs in this course have used supervised machine learning methods. In this lab, you will train and evaluate an unsupervised machine learning model. Specifically, you will construct and evaluate a k-means clustering model.

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
- Python Anaconda or R and RStudio

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.

Creating K-Means Cluster Models

You will train and evaluate a k-means clustering model for the **Forest Fire** data set. The ability to classify the potential forest fires would be of great use to fire managers. For example, a fire manager could determine the fire is in a cluster of serious fires or not.

Create a Two-Cluster Model

1. If you have not already done so, open a browser and browse to https://studio.azureml.net. Then sign in using the Microsoft account associated with your Azure ML account.
2. Create a new blank experiment, and give it the title **Fire Clustering**.
3. Search for the **Forest fires data** dataset, and drag it onto the blank canvas.

   Note: When applying k-means clustering models it is important to clean outliers from the data. The presence of outliers will distort the covariance structure of the data leading to poor clustering results.

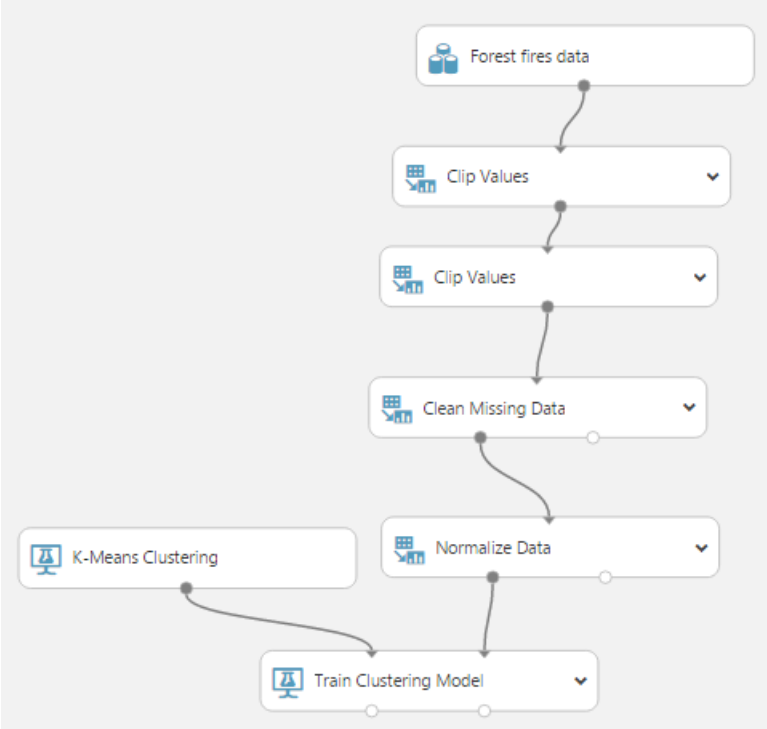
4. Search for the **Clip Values** module and drag it onto the canvas. Connect the output of the **Forest fire data** dataset to the input of the **Clip Values** module.
5. Select the **Clip Values** module and set its properties as follows:
    - **Set of thresholds**: ClipSubpeaks
    - **Lower threshold**: Percentile
    - **Percentile number of lower threshold**: 1
    - **Lower substitute values**: Missing
    - **Column selector**: column names FFMC
    - **Overwrite flag**: checked
    - **Add indicator columns**: unchecked
6. Drag another **Clip Values** module onto the canvas. Connect the output of the first **Clip Values** module to the input of the new **Clip Values** module.
7. Select the new **Clip Values** and set the properties as follows:
    - **Set of thresholds**: ClipPeaks
    - **Upper threshold**: Percentile
    - **Percentile number of upper threshold**: 99
    - **Upper substitute values**: Missing
    - **Column selector**: column names ISI, rain
    - **Overwrite flag**: checked
    - **Add indicator columns**: unchecked
8. Search for the **Clean Missing Data** module and drag it onto the canvas. Connect the output of the second **Clip Values** module to the input of the **Clean Missing Data** module.
9. Click on the **Clean Missing Data** module and set the parameters as follows:
    - **Column Selector**: All columns
    - **Minimum missing value ratio**: 0
    - **Maximum missing value ratio**: 1
    - **Cleaning mode**: Remove entire row

    Note: Before computing clusters with the k-means method it is important to normalize the columns in the data. Since k-means clustering depends on covariance structure ZScore normalization is typically used. If data is not properly normalized, the columns with the larges range of numeric values will dominate the solution.

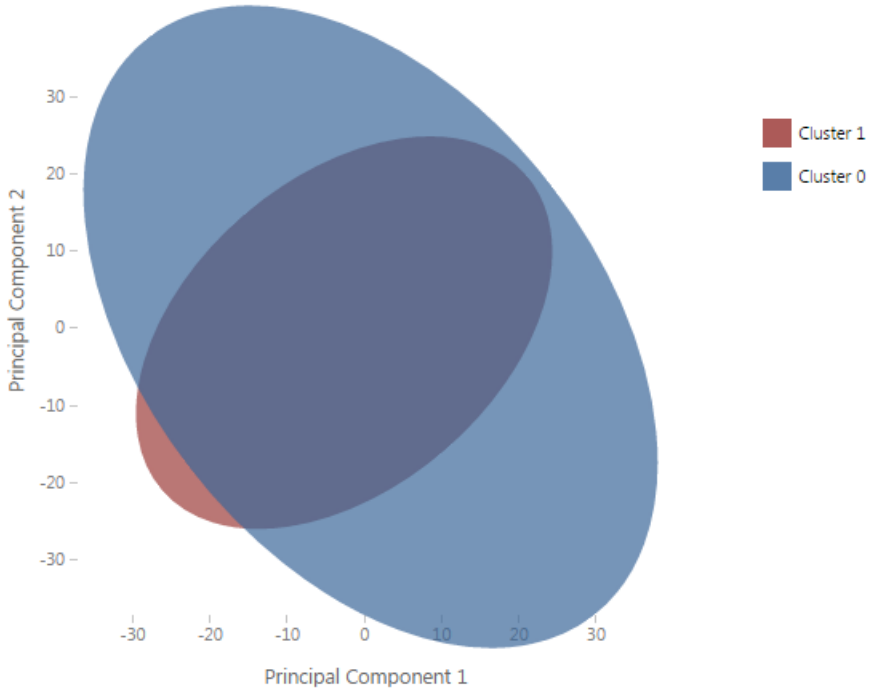
10. Search for the **Normalize Data** module and drag it onto the canvas. Connect the **Cleaned dataset** (left) output of the **Clean Missing Data** model to the input to the **Normalize Data** module.
11. Select the **Normalize Data** module and set the properties as follows:
    - **Transformation method**: ZScore
    - **Use 0 for constant columns when checked**: Unchecked
    - **Column selector**: column type Numeric

12. Search for the **K-Means Clustering** module and drag it onto the canvas.
13. Search for the **Train Clustering Model** module and drag it onto the canvas.
14. Connect the **Untrained model** output of the **K-Means Clustering** module to the **Untrained model** (left) input of the **Train Clustering Model** module; and connect the **Transformed dataset** (left) output of the **Normalize Data** module to the **Dataset** (right) input of the **Train Clustering Model** module.

15. Select the **Train Clustering Model** module, and in the Properties pane, launch the column selector and select the following columns:
    - **FFMC**
    - **DMC**
    - **DC**
    - **ISI**
    - **temp**
    - **RH**
    - **wind**
    - **area**
16. Select the **K-Means Clustering** module and configure the following properties:
    - **Create trainer mode**: Single Parameter
    - **Number of Centroids**: 2
    - **Initialization**: Random
    - **Random number seed**: 2345
    - **Metric**: Euclidian
    - **Iterations**: 100
    - **Assign Label Mode**: Fill missing values
17. Verify that your experiment resembles the following figure:



18. Save and run your experiment.
19. When the experiment has finished visualize the output of the **Results dataset** (right) output of the **Train Clustering Model** module. Your visualization should resemble the figure below:



Note that there are two distinct ellipses shown on this projection of the first two principle components. The major axes (long dimension) of each ellipse are in a distinct, nearly orthogonal, indicating the two clusters have good separation.

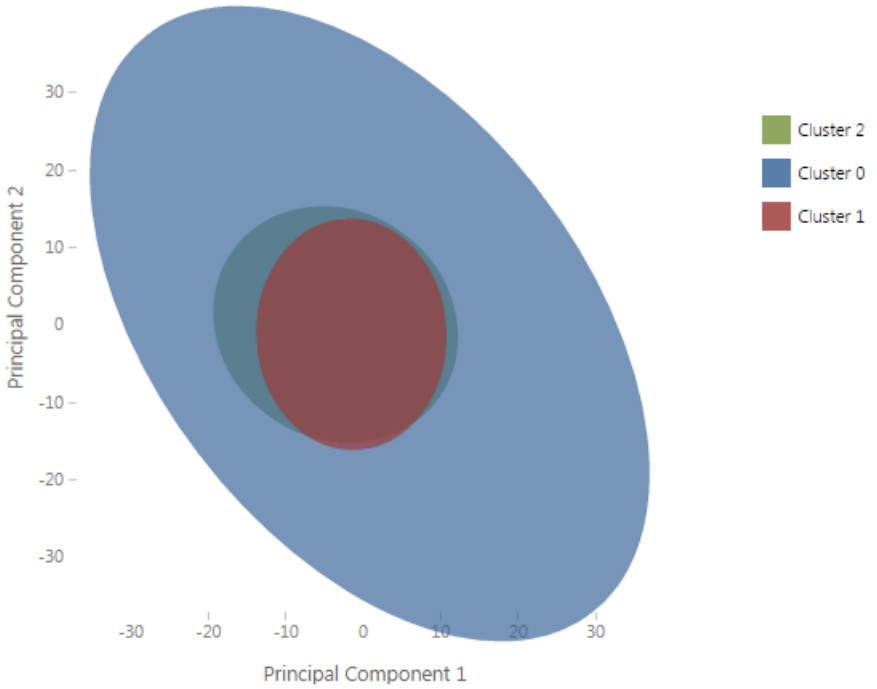
20. Close the results dataset.

Create a Three-Cluster Model

Creating two clusters appears to have worked fairly well. You will now create a model with three clusters and compare it to the model with two clusters by following these steps:

1. Copy the **K-Means Clustering** module and the **Train Clustering Model** module. Paste the copied modules onto the canvas and drag them to one side.
2. Connect the **Transformed dataset** (left) output of the **Normalize Data** module to the **Dataset** (right) input of the new **Train Clustering Model** module.
3. Ensure the output of the copied **K-Means Clustering** module is still connected to the **Untrained model** input of the copied **Train Clustering Model** module., and that your experiment resembles the figure below:

4. Select the new **K-Means Clustering** module and configure the following parameters, so that 3 clusters will be computed:
    - **Create trainer mode**: Single Parameter
    - **Number of Centroids**: 3
    - **Initialization**: Random
    - **Random number seed**: 2345
    - **Metric**: Euclidian
    - **Iterations**: 100
    - **Assign Label Mode**: Fill missing values
4. Save and run your experiment.
5. When the experiment has finished visualize the output of the **Results dataset** output of the new **Train Clustering Model** module. Your visualization should resemble the figure below:



Examine this diagram and compare it to the result obtained for the two cluster case. The ellipse for the first cluster (**Cluster 0**) looks about the same. However, the ellipses for the other two clusters (**Cluster 1** and **Cluster 2**) are nearly circular. Further the direction of the major axes of the ellipses for **Cluster 1** and **Cluster 2** are close to that of **Cluster 0**. The separation of the clusters in this projection is poor and it appears that three clusters are too many for this dataset.

6. Close the visualization.

Visualizing the Clusters with R

Note: If you prefer to work with Python, complete the exercise *Visualizing the Clusters with Python.*

You have created and performed an initial evaluation of two clustering models for the **Forest Fires** dataset. The two cluster model seemed to work well for these data. But what are these data telling you? Could a forest manager take advantage of these results to better deploy firefighting resources on to fires with a tendency to grow large as opposed to minor fires which might actually improve forest health?

In this exercise you will evaluate using custom R visualizations of the results of the two cluster model to determine how to interpret these clusters. You will plot cluster members (assignments) in a number of views to include the following:

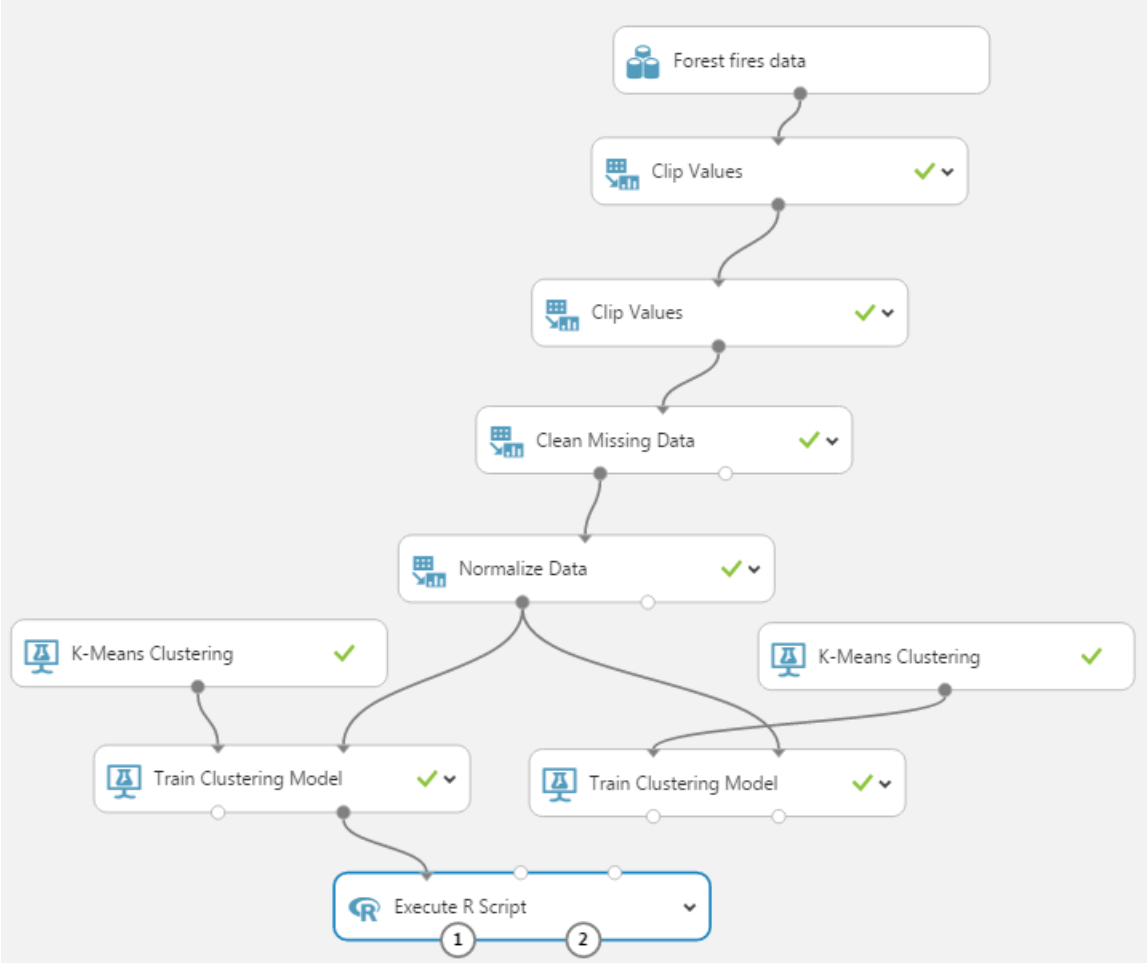
- The principle component projections of the members (assignments) of the two clusters.
- Scatter plots of the cluster assignments by the numeric columns used to compute the cluster model and fire area.
- Scatter plots of the cluster assignments columns used to compute the cluster model vs. the values of FFMC.

Add an R Script to Visualize Clusters

Follow these steps to visualize and investigate this data set:

1. Search for an **Execute R Script** module and drag it onto the canvas.

2. Connect the **Results dataset** (right) output of the first **Train Clustering Model** module (for the model that generates 2 clusters) to the **Dataset1** (left) input port of the **Execute R Script** module. Your experiment should now resemble the figure below:



3. Select the **Execute R Script** module, and in the Properties pane, replace the existing R script with the following code. You can copy and paste this code from the **VisClusters.R** file in the folder where you extracted the lab files:

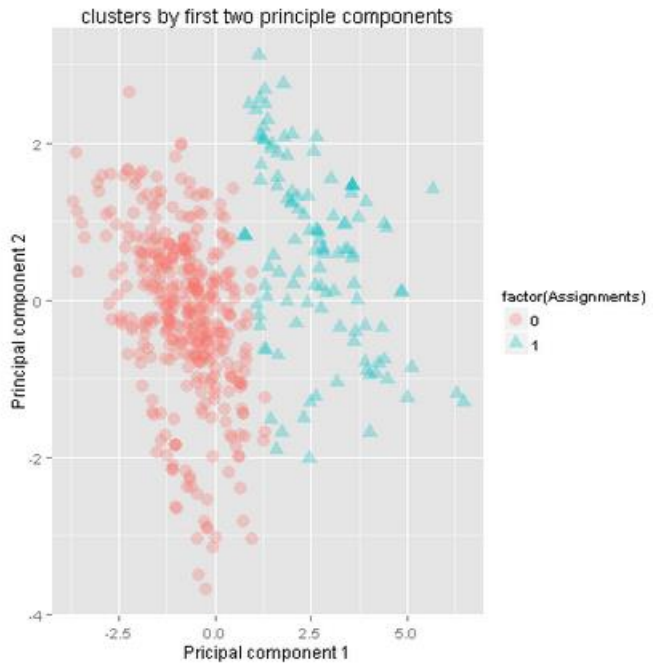
```
frame1 <- maml.mapInputPort(1)

library(ggplot2)

## Compute principal components and
## plot the clusters projected by the first two.
numCols <- c("FFMC", "DMC", "DC", "ISI", "temp",
             "RH", "wind", "area")
pcfit <- princomp(frame1[, numCols])
pcframe <- data.frame( as.matrix(frame1[, numCols])
                       %*% pcfit$loadings[, 1:2],
                       Assignments = frame1$Assignments)

ggplot(pcframe, aes(Comp.1, Comp.2)) +
  geom_point(aes(shape = factor(Assignments),
```

```
                color = factor(Assignments)),
            alpha = 0.3, size = 4) +
  ggtitle(paste("clusters by first two principle components")) +
  xlab("Pricipal component 1") + ylab("Principal component 2")

## Create scatter plots of certain numeric columns vs. area.
numCols <- c("FFMC", "DMC", "DC", "ISI", "temp",
             "RH", "wind")
plot.clusts <- function(x){
  ggplot(frame1, aes_string(x, 'area')) +
          geom_point(aes(shape = factor(Assignments),
                         color = factor(Assignments)),
                     alpha = 0.3, size = 4) +
          ggtitle(paste("clusters for", x, "vs. area"))
}
lapply(numCols, plot.clusts)

## Look at scatter plots of the clusters by FFMC.
numCols <- c("DC", "DMC", "ISI", "temp",
             "RH", "wind", "area")

plot.clusts2 <- function(x){
  ggplot(frame1, aes_string(x, 'FFMC')) +
    geom_point(aes(shape = factor(Assignments),
                   color = factor(Assignments)),
               alpha = 0.3, size = 4) +
    ggtitle(paste("clusters for", x, "vs. FFMC"))
}
lapply(numCols, plot.clusts2)
```

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.

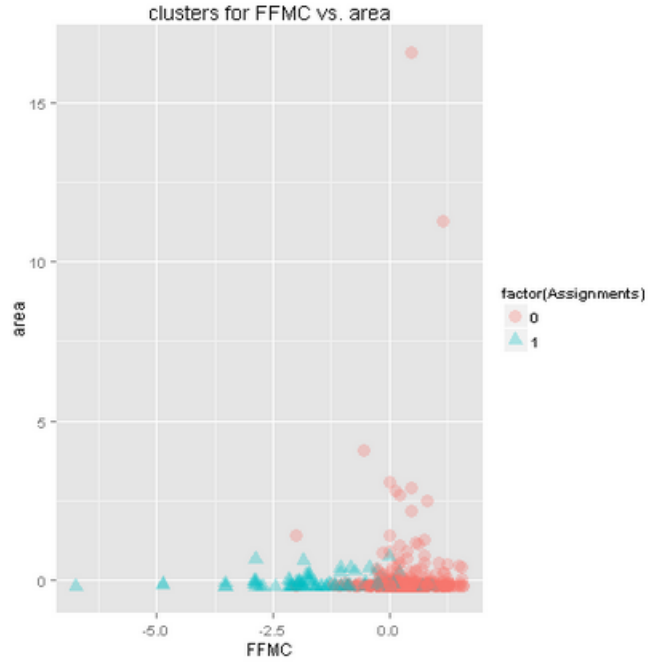
This code performs the functions already mentioned in the introduction to this section.

4. Save and run the experiment.
5. When the experiment has finished running visualize the output of the **R Device** output of the **Execute R Script** module.
6. Examine the scatter plot of the cluster assignments projected by the first two principle components:



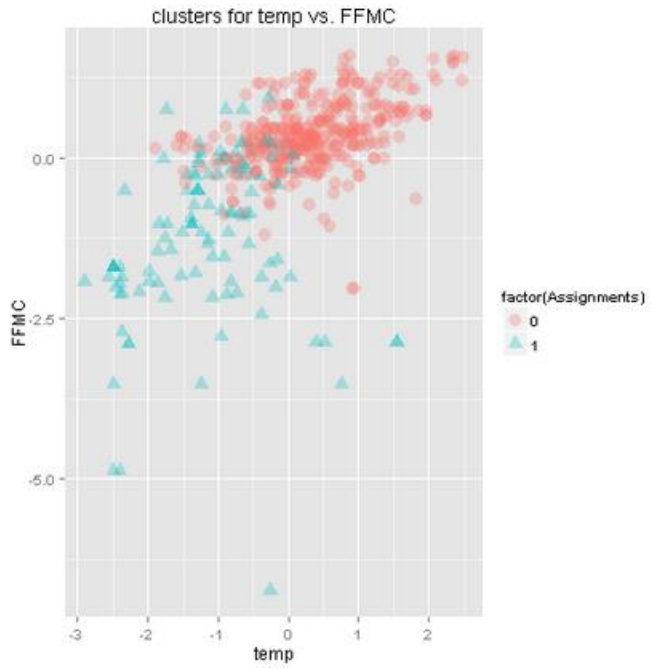
As you examine this plot, note that the two clusters are well separated in this projection, confirming that a two cluster model works well for these data.

7. Examine the scatter plot of the cluster assignments by **FFMC** and **area**:



Notice that nearly all the fire which grow to significant size are in cluster 0. This might be useful information for a forest manager.

8. Examine the plot of cluster assignments by **temp** and **FFMC**:



Notice, that once again, the two clusters exhibit separation by these two factors. This confirms that the two cluster model appears to work well with these data.

9. Close the R Device output.

Visualizing the Clusters with Python

Note: If you prefer to work with R, complete the exercise *Visualizing the Clusters with R*.

You have created and performed an initial evaluation of two clustering models for the Forest Fires dataset. The two cluster model seemed to work well for these data. But what are these data telling you? Could a forest manager take advantage of these results to better deploy firefighting resources on to fires with a tendency to grow large as opposed to minor fires which might actually improve forest health?

In this exercise you will evaluate using custom Python visualizations of the results of the two cluster model to determine how to interpret these clusters. You will plot cluster members (assignments) in a number of views to include the following:

- Plot the principle component projections of the members (assignments) of the two clusters.
- Create scatter plots of the cluster assignments by the numeric columns used to compute the cluster model and fire area.
- Create scatter plots of the cluster assignments columns used to compute the cluster model vs. the values of FFMC.

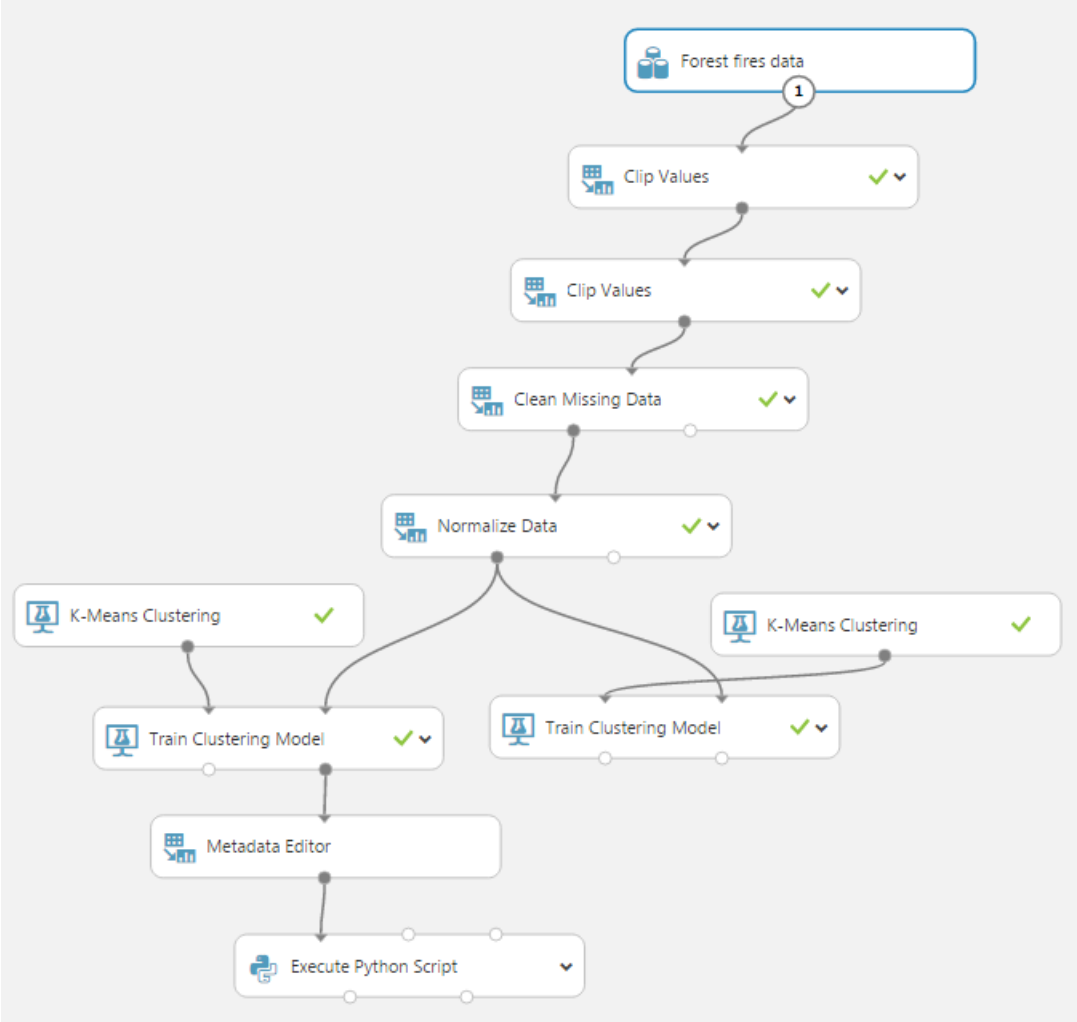
Add a Python Script to Visualize Clusters

Follow these steps to visualize and investigate this data set:

1. Search for the **Metadata Editor** module and drag it onto the canvas.
2. Connect the **Results dataset** (right) output of the first (two cluster) **Train Cluster Model** module to the input of the **Metadata Editor**.

3. Configure the properties of the **Metadata Editor** as follows:
    - **Column selector**: Assignments
    - **Data type**: Unchanged
    - **Categorical**: Make non-categorical
    - **Fields**: Unchanged
    - **New column names**: blank

4. Search for the **Execute Python Script** module and drag it onto the canvas.
5. Connect the **Results dataset** output of the **Metadata Editor** module to the **Dataset1** (left) input port of the **Execute Python Script** module. Your experiment should now resemble the figure below:



6. Select the **Execute Python Script** module, and in the Properties pane, replace the existing Python script with the following code. You can copy and paste this code from the **VisClusters.py** file in the folder where you extracted the lab files:

```
def azureml_main(frame1):
# Set graphics backend
    import matplotlib
    matplotlib.use('agg')
```

```
    import matplotlib.pyplot as plt
    import sklearn.decomposition as de
    import pandas as pd

    Azure = True

## Compute and plot the clusters by first two principal
components
    num_cols = ['FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH', 'area']
    pca = de.PCA(n_components = 2)
    pca.fit(frame1[num_cols].as_matrix())
    pca_frame =
pd.DataFrame(pca.transform(frame1[num_cols].as_matrix()))
    pca_frame['Assignments'] = frame1.Assignments

    temp0 = pca_frame.ix[pca_frame['Assignments'] == 0, :]
    temp1 = pca_frame.ix[pca_frame['Assignments'] == 1, :]
    temp0.columns = ['PC1', 'PC2', 'Assignments']
    temp1.columns = ['PC1', 'PC2', 'Assignments']

    fig = plt.figure(figsize = (12,6))
    fig.clf()
    ax = fig.gca()
    temp0.plot(kind = 'scatter', x = 'PC1', y = 'PC2',
color='DarkBlue', label='Group 0', alpha = 0.3, ax = ax)
    temp1.plot(kind = 'scatter', x = 'PC1', y = 'PC2',
color='Red', label='Group 1', alpha = 0.3, ax = ax)
    ax.set_title('Clusters by principal component projections')
    ax.set_xlabel('First principal component')
    ax.set_ylabel('Second principal component')
    if(Azure == True): fig.savefig('PCA.png')

## Create data frames for each cluster
    temp0 = frame1.ix[frame1['Assignments'] == 0, :]
    temp1 = frame1.ix[frame1['Assignments'] == 1, :]

## Scatter plots of area vs other numeric variables
    num_cols = ['FFMC', 'DC', 'ISI', 'temp', 'RH', 'rain']
    fig = plt.figure(figsize = (12, 24))
    fig.clf()
    i = 1
    for col in num_cols:
        ax = fig.add_subplot(6, 1, i)
        title = 'Scatter plot of ' + col + ' vs. fire area'
        temp0.plot(kind = 'scatter', x = col, y = 'area',
color='DarkBlue', label='Group 0', alpha = 0.3, ax = ax)
        temp1.plot(kind = 'scatter', x = col, y = 'area',
color='Red', label='Group 1', alpha = 0.3, ax = ax)
        ax.set_title(title)
        ax.set_xlabel('')
```

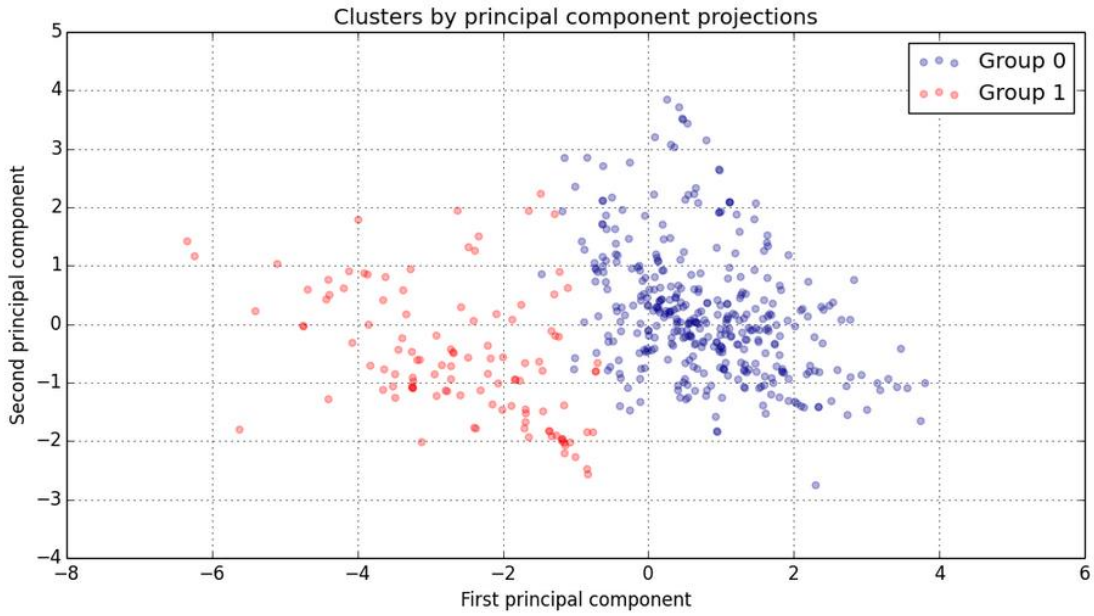
```
        i += 1
    if(Azure == True): fig.savefig('Scatter_vs_area.png')

## Scatter plots of FFMC vs the other numeric variables.
    num_cols = ['DC', 'ISI', 'temp', 'RH', 'rain', 'area']
    fig = plt.figure(figsize = (12, 24))
    fig.clf()
    i = 1
    for col in num_cols:
        ax = fig.add_subplot(6, 1, i)
        title = 'Scatter plot of ' + col + ' vs. FFMC'
        temp0.plot(kind = 'scatter', x = col, y = 'FFMC',
color='DarkBlue', label='Group 0', alpha = 0.3, ax = ax)
        temp1.plot(kind = 'scatter', x = col, y = 'FFMC',
color='Red', label='Group 1', alpha = 0.3, ax = ax)
        ax.set_title(title)
        ax.set_xlabel('')
        i += 1
    if(Azure == True): fig.savefig('Scatter_vs_FFMC.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.

WARNING!: Ensure you have a Python `return` statement at the end of your `azureml_main` function; for example, `return frame1`. Failure to include a return statement will prevent your code from running and may produce an inconsistent error message.

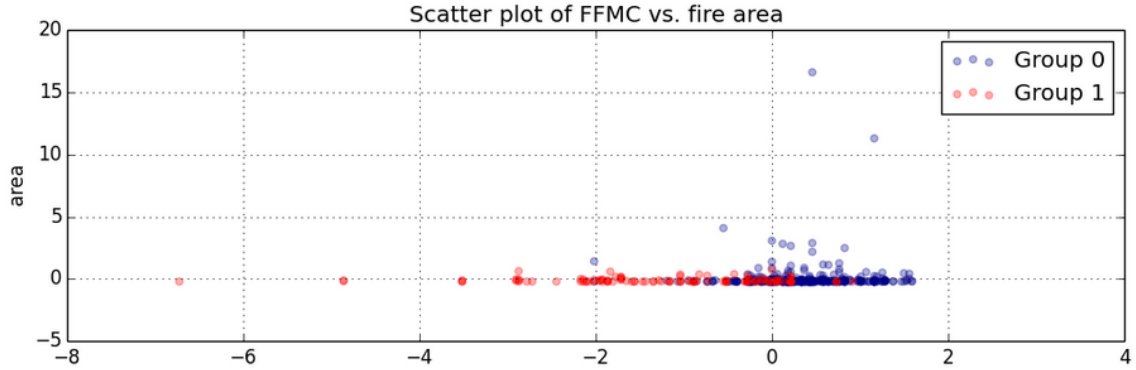
This code performs the functions already mentioned in the introduction to this section.

7. Save and run the experiment.
8. When the experiment has finished running visualize the output of the **Python device** output of the **Execute Python Script** module.
9. Examine the scatter plot of the cluster assignments projected by the first two principle components:



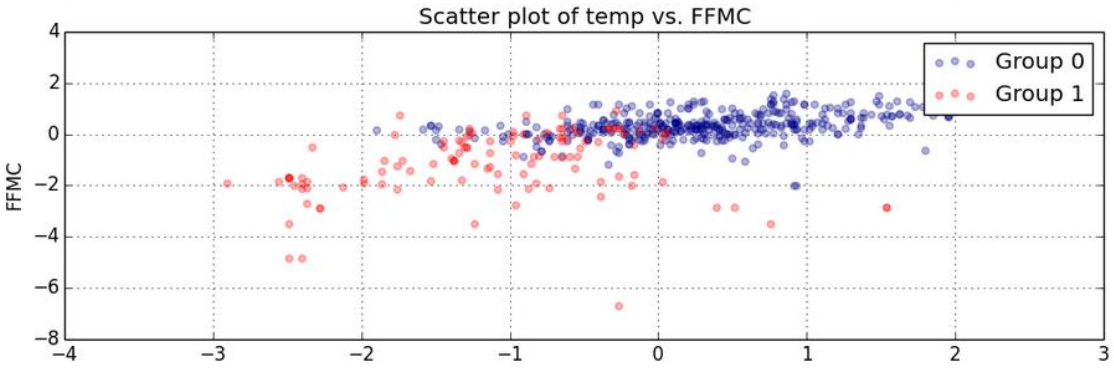
As you examine this plot, note that the two clusters are well separated in this projection, confirming that a two cluster model works well for these data.

10. Examine the scatter plot of the cluster assignments by **FFMC** and **area**:



Notice that nearly all the fire which grow to significant size are in cluster 0. This might be useful information for a forest manager.

11. Examine the plot of cluster assignments by **temp** and **FFMC**:



Notice, that once again, the two clusters exhibit separation by these two factors. This confirms that the two cluster model appears to work well with these data.

12. Close the Python Device output.

Summary

In this lab you performed k-means clustering analysis of the **Forest Fires** dataset. In particular, you:

- Computed and evaluated a two cluster model for this dataset.
- Tested a three cluster model on this dataset.
- Visualized the results of the clustering model with R or Python code. You confirmed that the two cluster model fits these data well. Further, you noted that fires which grown to significant size have a **Group 0** assignment.

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