Data Science and Machine Learning Essentials

Lab 3B – Feature Engineering and Modeling

By Stephen Elston and Graeme Malcolm

Overview

In this lab, you will learn how to use R or Python to engineer or construct new features and build a machine learning model. If you intend to work with R, complete the *Feature Engineering with R* exercise. If you plan to work with Python, complete the *Feature Engineering with Python* exercise. Unless you need to work in both languages, you do not need to try both of these exercises.

Note: This lab assumes you have completed lab 3A. If you have not completed lab 3A, you can copy the experiment from the Cortana Analytics Gallery.

In the previous lab, you noted the nonlinear relationship between the features in the building energy efficient dataset and the label. One possible approach to modeling these data is to compute and use polynomial features. In this lab you will compute just such features.

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.

Feature Engineering with R

Having visualized data to determine any apparent relationships between columns in a dataset, you can prepare for modeling by selecting only the data columns that you believe will be pertinent features for the label you need to predict. Additionally, you may decide to generate new feature columns based on calculations that combine or extrapolate existing columns.

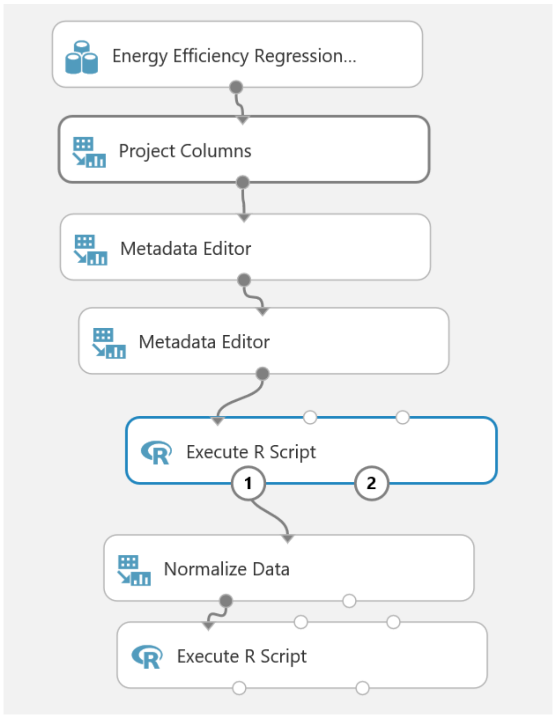
In this exercise, you will create an experiment that projects columns from the building energy efficiency dataset, and generate new columns that will help you build a model to predict the *heating load* of a building, which is a measure of its energy efficiency.

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

Add R to Generate New Columns

You will start by rearranging your experiment.

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. Open the **Visualize Data (R)** experiment you created in Lab 3A, and save a copy as **Modeling (R)**. If you did not complete Lab 3A, you can copy the **Visualize Data (R)** experiment from the collection for this course in the Cortana Analytics Gallery at http://gallery.cortanaanalytics.com/Collection/5bfa7c8023724a29a41a4098d3fc3df9.
3. Delete the connections between the **Metadata Editor** and **Normalize Data** modules.
4. Add another **Execute R Script** module to your experiment. Connect the output of the second Metadata Editor module to the Dataset1 input of the new **Execute R Script** module. Then connect the **Results dataset** output of the **Execute R Script** module to the input of the **Normalize Data** module. Your experiment should now look like the one shown here:



5. Select the new **Execute R Script** module, and in the **Properties** pane, replace the existing **Execute R Script** module code with the following code. You can copy this code from the **NewFeatures.R** file in the folder where you extracted the lab files for this lab.

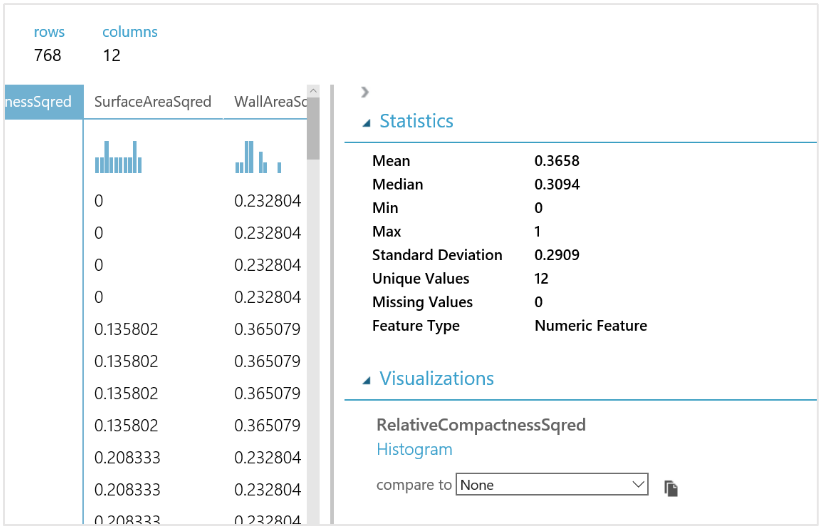
```
eeframe <- maml.mapInputPort(1)

library(dplyr)
eeframe = mutate(eeframe,
          RelativeCompactnessSqred = RelativeCompactness^2,
          SurfaceAreaSqred = SurfaceArea^2,
          WallAreaSqred = WallArea^2 ,
          RelativeCompactness3 = RelativeCompactness^3,
          SurfaceArea3 = SurfaceArea^3,
          WallArea3 = WallArea^3)

maml.mapOutputPort('eeframe')
```

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.

Review the code, and note that it creates polynomial columns named **RelativeCompactnessSqred**, **SurfaceAreaSqred**, and **WallAreaSqred** by squaring the values of the existing **RelativeCompactness**, **SurfaceArea**, and **WallArea** columns, and polynomial columns named **RelativeCompactness3**, **SurfaceArea3**, and **WallArea3** by cubing the values of the existing **RelativeCompactness**, **SurfaceArea**, and **WallArea** columns.

Note: In a later lab, you'll learn how to evaluate the effectiveness of these features in a predictive model. Remember that model creation is an iterative process, so you may go through several iterations of creating new features, adding them to models, and evaluating their effectiveness.

6. Save and run the experiment. When the experiment has finished, visualize the **Transformed Dataset** output port of the **Normalize Data** module, and verify that the dataset now includes columns named **RelativeCompactnessSqred**, **SurfaceAreaSqred**, **WallAreaSqred**, **RelativeCompactness3**, **SurfaceArea3**, and **WallArea3**, as shown here:



rows 768 columns 12

nessSqred	SurfaceAreaSqred	WallAreaSq
	0	0.232804
	0	0.232804
	0	0.232804
	0	0.232804
	0.135802	0.365079
	0.135802	0.365079
	0.135802	0.365079
	0.135802	0.365079
	0.208333	0.232804
	0.208333	0.232804
	0.208333	0.232804

Statistics

Mean	0.3658
Median	0.3094
Min	0
Max	1
Standard Deviation	0.2909
Unique Values	12
Missing Values	0
Feature Type	Numeric Feature

Visualize the New Columns

1. Select the last **Execute R Script** module in the experiment (after the **Normalize Data** module), and in the **Properties** pane, replace the existing code in the **Execute R Script** with the following code. You can copy this code from the **VisFeatures.R** file in the folder where you extracted the lab files for this lab.

```
## Use basic R graphics to create a pair-wise scatter plot
eeframe <- maml.mapInputPort(1)

## Scatterplot matrix
```

```r
pairs(~ ., data = eeframe)

## Use ggplot2 to create conditioned scatter plots
library(ggplot2)
plotCols <- c("RelativeCompactness",
              "SurfaceArea",
              "WallArea",
              "RelativeCompactnessSqred",
              "SurfaceAreaSqred",
              "WallAreaSqred",
              "RelativeCompactness3",
              "SurfaceArea3",
              "WallArea3",
              "RoofArea",
              "GlazingArea",
              "GlazingAreaDistribution")
plotEE <- function(x){
  title <- paste("Heating Load vs", x, "\n conditioned on OverallHeight
and Orientation")
  ggplot(eeframe, aes_string(x, "HeatingLoad")) +
    geom_point() +
    facet_grid(OverallHeight ~ Orientation) +
    ggtitle(title) +
    stat_smooth(method = "lm")
}
lapply(plotCols, plotEE)

## Create histograms
plotCols4 <- c("RelativeCompactness",
               "SurfaceArea",
               "WallArea",
               "RelativeCompactnessSqred",
               "SurfaceAreaSqred",
               "WallAreaSqred",
               "RelativeCompactness3",
               "SurfaceArea3",
               "WallArea3",
               "RoofArea",
               "GlazingArea",
               "GlazingAreaDistribution",
               "HeatingLoad")
library(gridExtra)
eeHist <- function(x) {
  title <- paste("Histogram of", x, "conditioned on OverallHeight")
  ggplot(eeframe, aes_string(x)) +
    geom_histogram(aes(y = ..density..)) +
    facet_grid(. ~ OverallHeight) +
    ggtitle(title) +
    geom_density()
}
lapply(plotCols4, eeHist)


## Create box plots
eebox <- function(x) {
  title <- paste("Box plot of", x, "by OverallHeight")
  ggplot(eeframe, aes_string('OverallHeight', x)) +
    geom_boxplot() +
    ggtitle(title)
}
lapply(plotCols4, eebox)
```

Review the code, and note that it creates visualizations of the columns in the datasets, including the new feature columns you added in the previous **Execute R Script** module.

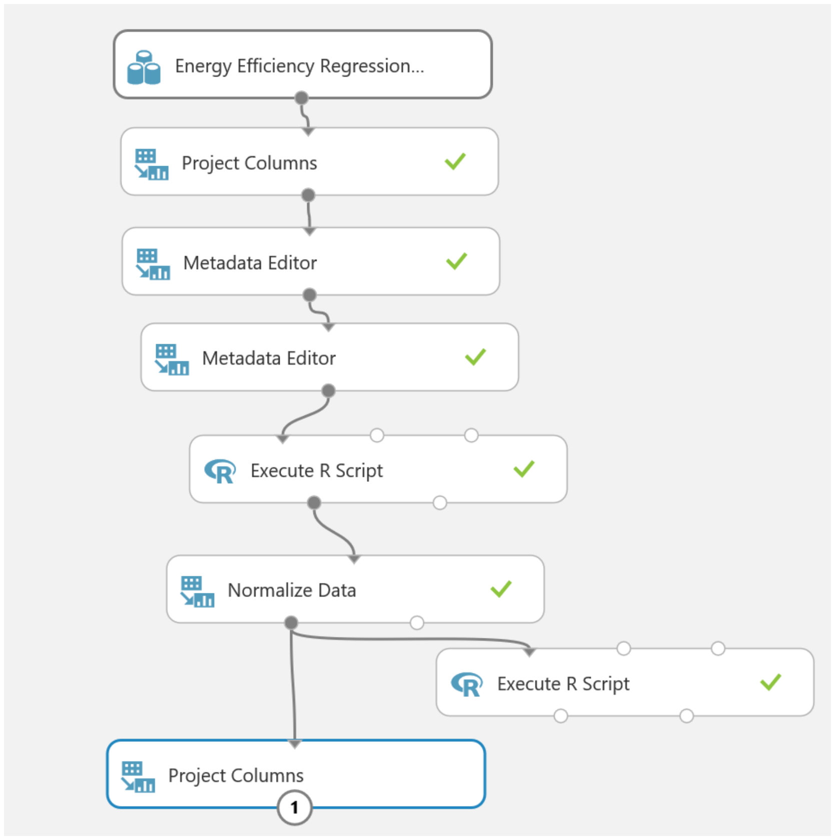
2. Save and run the experiment.
3. When the experiment has finished, visualize the **R Device Dataset** output port of the second **Execute R Script** module, and view the charts it has created.

   In particular, note that the conditioned scatter plots for **Heating Load vs Surface Area**, **Heating Load vs Surface Area Sqred**, **Heating Load vs Surface Area 3**, **Heating Load vs Wall Area**, **Heating Load vs Wall Area Sqred**, and **Heating Load vs Wall Area 3**. The shape of the curves on these plots is fairly similar for these similar features. There is some flattening of the curves with the higher order polynomials, the effect we are looking for. However, he effect is not dramatic in any event. Only by testing these features when you build machine learning models will you know which of these features are effective.

4. Close the R Device output.

Select Initial Features

1. Search for the **Project Columns** module and drag it to the canvas below the **Normalize Data** module. Then connect the **Transformed dataset** output port from the **Normalize Data** module (which is already connected to the second **Execute R Script** module) to the input port of the **Project Columns** module. You will use the **Project Columns** module to select the initial features for the model based on the data exploration and feature engineering you have performed so far.
2. Select the **Project Columns** module you just added, and in the **Properties** pane launch the column selector. Then configure the module to begin with all columns, and then exclude the following columns:
   - **Orientation**
   - **GlazingAreaDistribution**
3. Verify that your experiment looks like this, and then save the experiment and proceed to the *Creating a Model* exercise later in this lab.



Feature Engineering with Python

Having visualized data to determine any apparent relationships between columns in a dataset, you can prepare for modeling by selecting only the data columns that you believe will be pertinent features for the label you hope to predict. Additionally, you may decide to generate new feature columns based on calculations that combine or extrapolate existing columns.

In this exercise, you will create an experiment that projects columns from the building energy efficiency dataset, and generate new columns that will help you build a model to predict the *heating load* of a building, which is a measure of its energy efficiency.

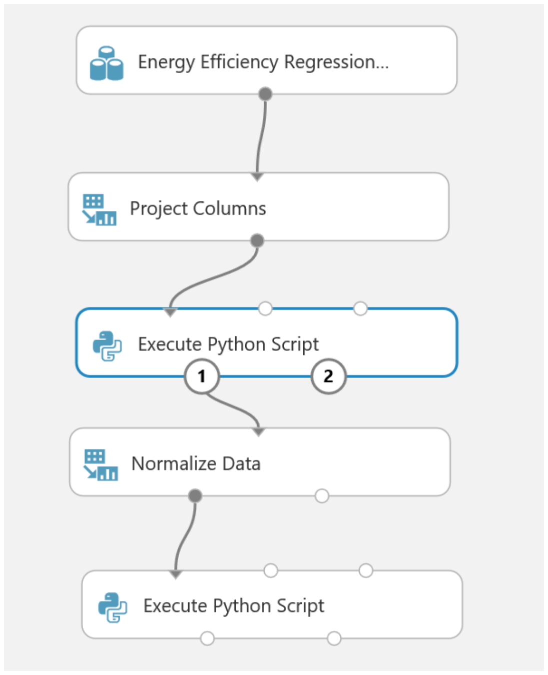
Note: If you prefer to work with R, skip this exercise and complete the previous exercise, *Feature Engineering with R*.

Add Python to Generate New Column

You will start by rearranging your experiment:

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. Open the **Visualize Data (Python)** experiment you created in Lab 3A, and save a copy as **Modeling (Python)**. If you did not complete Lab 3A, you can copy the **Visualize Data (Python)** experiment from the collection for this course in the Cortana Analytics Gallery at http://gallery.cortanaanalytics.com/Collection/5bfa7c8023724a29a41a4098d3fc3df9.
3. Add another **Execute Python Script** module to your experiment.
4. Delete connections between the **Project Columns** and **Normalize Data** modules. Connect the output of the **Project Columns** module to the **Dataset1** input port of the new **Execute Python**

Script module. Then connect the **Results dataset** output of the new **Execute Python Script** module to the input of the **Normalize Data** module. Your experiment should now resemble the figure below.



5. Select the new **Execute Python Script** module, and in the **Properties** pane, replace the existing code in the **Execute Python Script** module with the following code. You can copy this code from the **NewFeatures.py** file in the folder where you extracted the lab files for this lab.

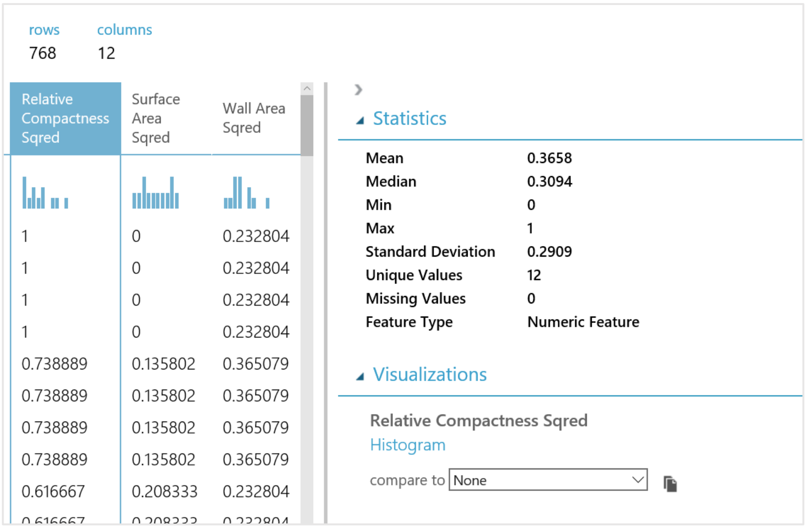
```
def azureml_main(frame1):
    sqrList = ["Relative Compactness", "Surface Area", "Wall Area"]
    sqredList = ["Relative Compactness Sqred", "Surface Area Sqred",
"Wall Area Sqred"]
    frame1[sqredList] = frame1[sqrList]**2
    cubeList = ["Relative Compactness 3", "Surface Area 3", "Wall Area
3"]
    frame1[cubeList] = frame1[sqrList]**3
    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.

Review the code, and note that it creates polynomial columns named **Relative Compactness Sqred**, **Surface Area Sqred**, and **Wall Area Sqred** by squaring the values of the existing **Relative Compactness**, **Surface Area**, and **Wall Area** columns, and polynomial columns named **Relative Compactness 3**, **Surface Area 3**, and **Wall Area 3** by cubing the values of the existing **Relative Compactness**, **Surface Area**, and **Wall Area** columns.

Note: In a later lab, you'll learn how to evaluate the effectiveness of these features in a predictive model. Remember that model creation is an iterative process, so you may go through several iterations of creating new features, adding them to models, and evaluating their effectiveness.

6. Save and run the experiment. When the experiment has finished, visualize the **Transformed Dataset** output port of the **Normalize Data** module, and verify that the dataset now includes columns named **Relative Compactness Sqred**, **Surface Area Sqred**, **Wall Area Sqred**, **Relative Compactness 3**, **Surface Area 3**, and **Wall Area 3**, as shown here:


rows 768 columns 12

Relative Compactness Sqred	Surface Area Sqred	Wall Area Sqred
1	0	0.232804
1	0	0.232804
1	0	0.232804
1	0	0.232804
0.738889	0.135802	0.365079
0.738889	0.135802	0.365079
0.738889	0.135802	0.365079
0.738889	0.135802	0.365079
0.616667	0.208333	0.232804

Statistics

Mean	0.3658
Median	0.3094
Min	0
Max	1
Standard Deviation	0.2909
Unique Values	12
Missing Values	0
Feature Type	Numeric Feature

Visualizations

Relative Compactness Sqred Histogram

compare to None

Visualize the New Columns

1. Select the last **Execute Python Script** module in the experiment (after the **Normalize Data** module), and in the **Properties** pane replace the existing code in the **Execute Python Script** module with the following code. You can copy this code from the **VisFeatures.py** file in the folder where you extracted the lab files for this lab.

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

    from pandas.tools.plotting import scatter_matrix
    import pandas.tools.rplot as rplot
    import matplotlib.pyplot as plt
    import numpy as np

## Create a pair-wise scatter plot
    fig1 = plt.figure(1, figsize=(10, 10))
    ax = fig1.gca()
    scatter_matrix(frame1, alpha=0.3,
                   diagonal='kde', ax = ax)
    plt.show()
    fig1.savefig('scatter1.png')

## Create conditioned scatter plots.
    col_list = ["Relative Compactness",
                "Surface Area",
                "Wall Area",
                "Relative Compactness Sqred",
                "Surface Area Sqred",
                "Wall Area Sqred",
                "Relative Compactness 3",
                "Surface Area 3",
                "Wall Area 3",
                "Roof Area",
                'Glazing Area',
                "Glazing Area Distribution"]

    indx = 0
    for col in col_list:
        if(frame1[col].dtype in [np.int64, np.int32,
np.float64]):
            indx += 1

            fig = plt.figure(figsize = (12,6))
            fig.clf()
            ax = fig.gca()
            plot = rplot.RPlot(frame1, x = col, y = 'Heating
Load')
            plot.add(rplot.TrellisGrid(['Overall
Height','Orientation']))
            plot.add(rplot.GeomScatter())
            plot.add(rplot.GeomPolyFit(degree=2))
            ax.set_xlabel(col)
            ax.set_ylabel('Heating Load')
            plot.render(plt.gcf())

            fig.savefig('scatter' + col + '.png')

## Histograms of Heating Load by Overall Height
    col_list = ["Relative Compactness",
                "Surface Area",
                "Wall Area",
                "Relative Compactness Sqred",
                "Surface Area Sqred",
                "Wall Area Sqred",
                "Relative Compactness 3",
                "Surface Area 3",
                "Wall Area 3",
                "Roof Area",
                'Glazing Area',
                "Glazing Area Distribution",
                "Heating Load"]
    for col in col_list:
        temp7 = frame1.ix[frame1['Overall Height'] == 7,
col].as_matrix()
        temp35 = frame1.ix[frame1['Overall Height'] == 3.5,
col].as_matrix()
        fig = plt.figure(figsize = (12,6))
        fig.clf()
        ax7 = fig.add_subplot(1, 2, 1)
        ax35 = fig.add_subplot(1, 2, 2)
        ax7.hist(temp7, bins = 20)
        ax7.set_title('Histogram of ' + col + '\n for for Overall
Height of 7')
        ax35.hist(temp35, bins = 20)
        ax35.set_title('Histogram of ' + col + '\n for for 
Overall Height of 3.5')
        fig.savefig('hists_' + col + '.png')


## Creat boxplots.
    for col in col_list:
        if(frame1[col].dtype in [np.int64, np.int32, 
np.float64]):
            fig = plt.figure(figsize = (6,6))
            fig.clf()
            ax = fig.gca()
            frame1[[col, 'Overall Height']].boxplot(column = 
[col], ax = ax, by = ['Overall Height'])
            ax.set_xlabel('')
            fig.savefig('box_' + col + '.png')

## Return the data frame
    return frame1
```

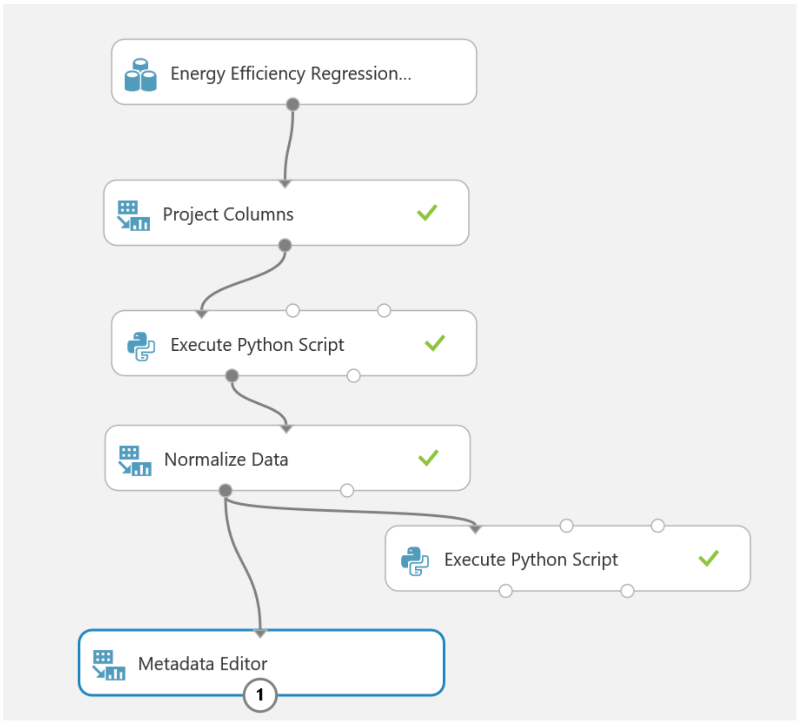
Review the code, and note that it creates visualizations of the columns in the datasets, including the new feature columns you added in the previous **Execute Python Script** module.

2. Save and run the experiment.
3. When the experiment has finished, visualize the **Python Device Dataset** output port of the second **Execute Python Script** module, and view the charts it has created.

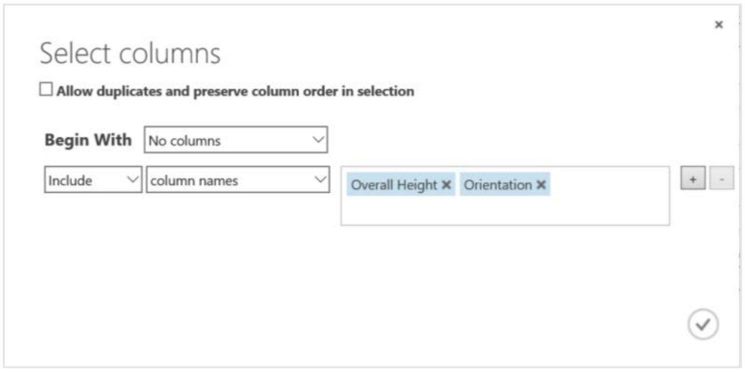
   In particular, note that the conditioned scatter plots for **Heating Load vs Surface Area**, **Heating Load vs Surface Area Sqred**, **Heating Load vs Surface Area 3**, **Heating Load vs Wall Area**, **Heating Load vs Wall Area Sqred**, and **Heating Load vs Wall Area 3**. The shape of the curves on these plots is fairly similar for these similar features. There is some flattening of the curves with the higher order polynomials, the effect we are looking for. However, he effect is not dramatic in any event. Only by testing these features when you build machine learning models will you know which of these features are effective.

Select Initial Features

1. Search for the **Metadata Editor** module, and drag it to the canvas. Then connect the **Results dataset** output port from the **Normalize Data** module (which is already connected to the second **Execute Python Script** module) to the input port of the **Metadata Editor** module so that your experiment looks like this:



2. Select the **Metadata Editor** module, and in the **Properties** pane, launch the column selector. Then configure the column selector to begin with no columns and include the **Overall Height** and **Orientation** column names as shown in the following image.

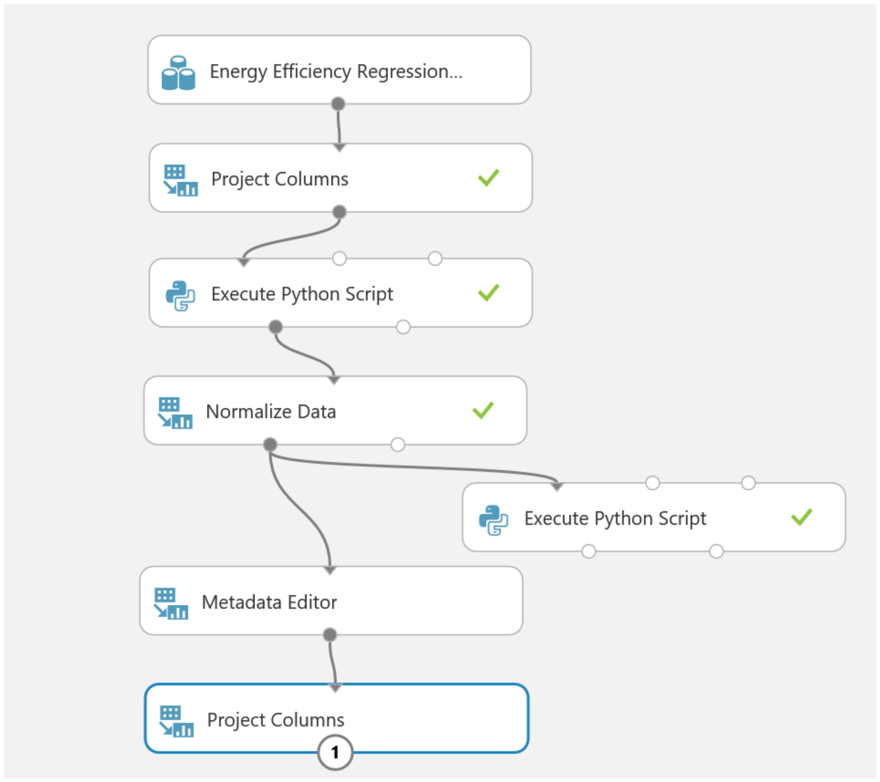

Select columns

☐ Allow duplicates and preserve column order in selection

Begin With: No columns

Include | column names | Overall Height ✕ Orientation ✕

3. With the **Metadata Editor** module selected, in the **Properties** pane, in the **Categorical** list, select **Make Categorical**.
4. Search for the **Project Columns** module and drag it to the canvas below the **Metadata Editor** module. Then connect the **Results dataset** output port from the **Metadata Editor** module to the input port of the **Project Columns** module. You will use the **Project Columns** module to select the initial features for the model based on the data exploration and feature engineering you have performed so far.
5. Select the **Project Columns** module you just added, and in the **Properties** pane launch the column selector. Then configure the module to begin with all columns, and then exclude the following columns:
   - **Orientation**
   - **Glazing Area Distribution**
2. Verify that your experiment looks like this, and then save the experiment and proceed to the *Creating a Model* exercise.



Creating a Model

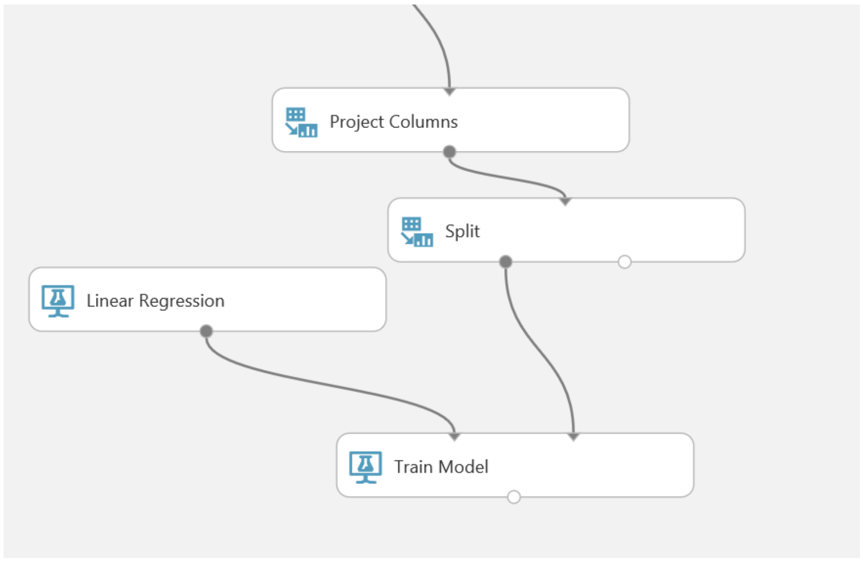
Having prepared for modeling by selecting or creating columns that you believe will be pertinent features for the label you hope to predict, you can start creating, training, and testing machine learning models for your particular predictive requirements.

Create and Train a Model

1. Search for the **Split** module, and drag it to the canvas below the **Project Columns** module you added at the end of the previous exercise. Then connect the **Results dataset** output port from the **Project Columns** module to the input port of the **Split** module. You will use the **Split** module to split the data into two sets; one to train the model, and another to test it.
2. Select the **Split** module, and in the **Properties** pane, set the following properties:
   - **Splitting mode**: Split Rows
   - **Fraction of rows in the first output dataset**: 0.6
   - **Randomized**: *Selected*
   - **Random seed**: 5416
   - **Stratified split**: False
3. Search for the **Linear Regression** module, and drag it to the canvas beside the **Split** module. Then select the **Linear Regression** module and in the **Properties** pane, set the following properties:
   - **Solution method**: Ordinary Least Squares
   - **L2 regularization weight**: 0.0001
   - **Include intercept term**: *Unselected*
   - **Random number seed**: 345689
   - **Allow unknown category levels**: *Selected*
4. Search for the **Train Model** module, and drag it to the canvas beneath the **Split** and **Linear Regression** modules. Then connect the output from the **Linear Regression** model to the left

input of the **Train Model** module, and connect the left output of the **Split** module (which represents the training data set) to the right input of the **Train Model** module.

5. Select the **Train Model** module, and in the **Properties** pane launch the column selector. Then configure the module to include only the **Heating Load** column. This trains the model to predict a value for the **Heating Load** label based on the other feature columns in the dataset.
6. Verify that your experiment from the second **Project Columns** module onwards resembles the following image, and then save and run the experiment.



7. When the experiment has finished, visualize the output for the **Train Model** module, and note that it describes the relative weight of each feature column. The feature weights represent the relative importance that the trained model applies to each feature column when used to predict the **Heating Load** label, based on the training data set. In the next exercise, you will use the set of test data to evaluate these features.
8. Close the output.

Testing and Scoring Models

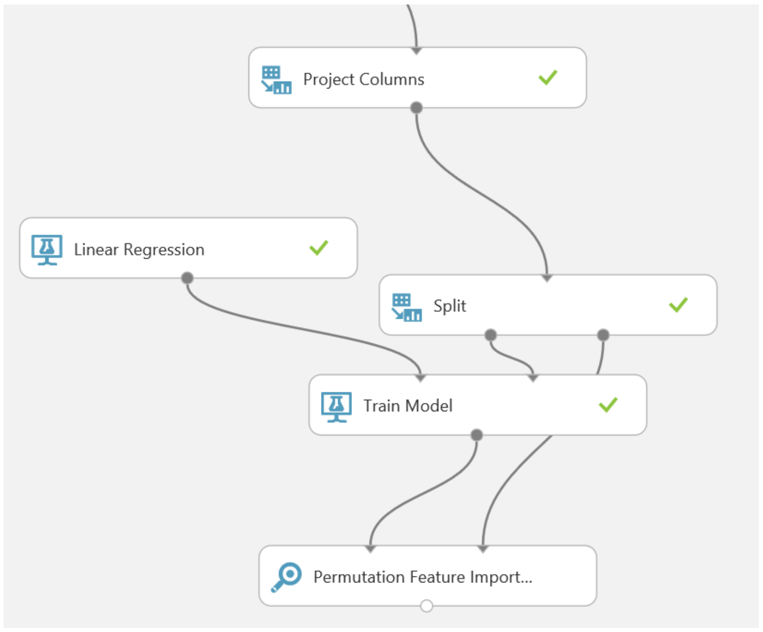
Having trained a model, you can examine it to evaluate its effectiveness at predicting label values. Predictive modeling is an iterative process that often involves creating and comparing multiple models, and refining these models until you find one that suits your requirements.

In this exercise, you will evaluate the effectiveness of the features used by your trained model against the test data set. Then you will create a second model, and score the models to compare them.

Evaluate Feature Importance

1. Search for the **Permutation Feature Importance** module, and drag it to the canvas beneath the **Train Model** module.
2. Connect the output of the **Train Model** module to the left input of the **Permutation Feature Importance** module. Then connect the right output of the **Split** module (which represents the training data set) to the right input of the **Permutation Feature Importance** module.
3. Select the **Permutation Feature Importance** module, and in the **Properties** pane, set the following properties:
    - **Random seed**: 1234
    - **Metric for measuring performance**: Regression – Root Mean Squared Error

4. Verify that your experiment from the **Project Columns** module onwards resembles the following image, and then save and run the experiment.

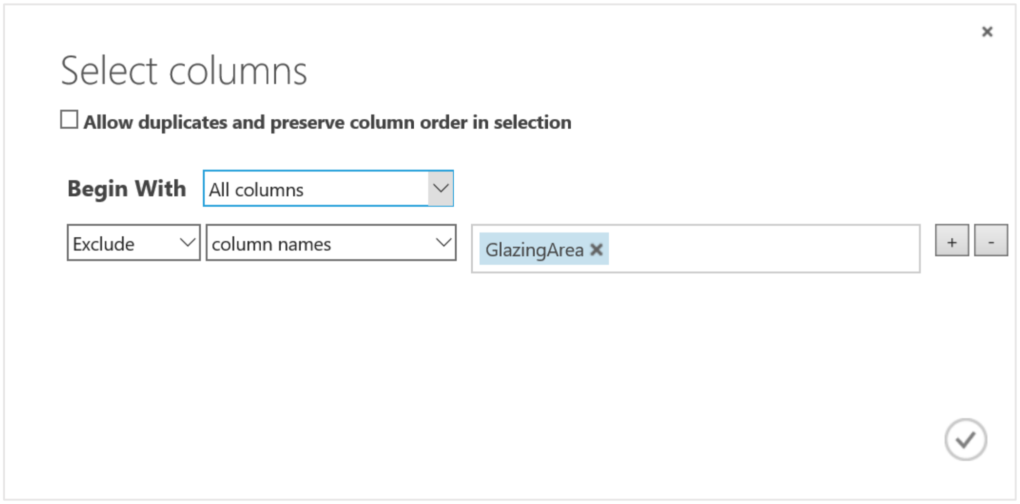


5. When the experiment has finished, visualize the output for the **Permutation Feature Importance** module, and note that it displays the feature columns in descending order of importance for predicting the label. Notice that the order of the features is different for the regression model weights and the feature importance. This is not too surprising since the first figures are regression weights and the second set of figures is computed using a permutation method on the features. The lowest valued column in both cases in **Glazing Area**.

Create a Second Model

In the previous exercise, you computed some new features and then determined which features were less important to the model. In this exercise you will compute a new model with a pruned feature set. Feature pruning is an important process in data science. Models with unnecessary features are said to be 'over parameterized'. Over parameterized models will generally not generalize well to the range of input values expected in production. However, one must proceed with caution. Removing too many features can lead to the loss of important information and therefore reduced model performance.

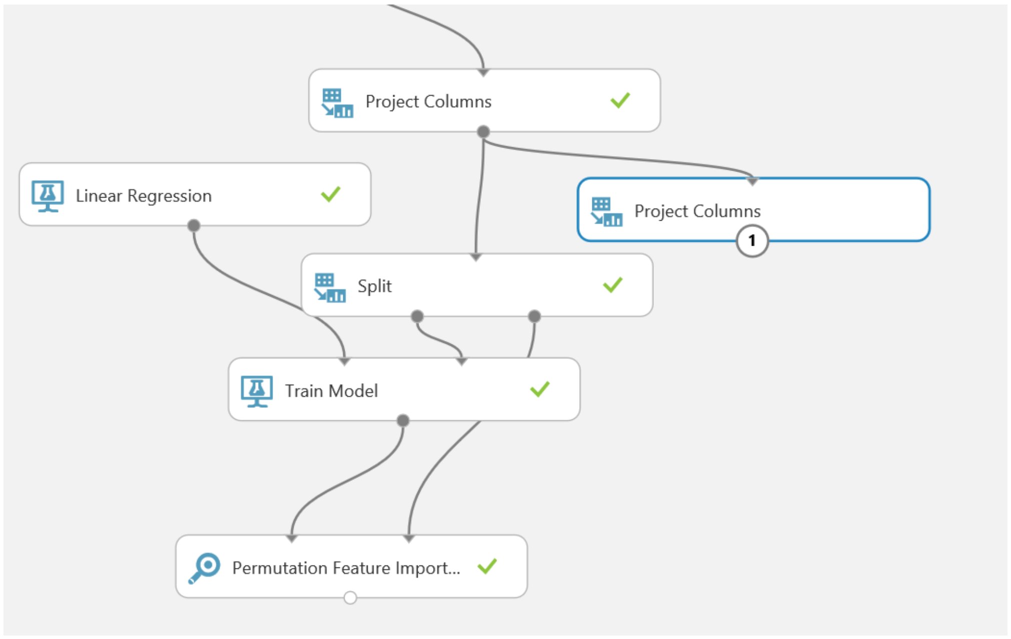
1. Search for the **Project Columns** module, and drag it to the canvas below the second existing **Project Columns** module. Then connect the output port from the existing **Project Columns** module to the input port of the **Project Columns** module you just added (in addition to the **Split** module to which it is already connected).
2. Select the **Project Columns** module you just added, and in the **Properties** pane launch the column selector. Then configure the module to begin with all columns and exclude the **Glazing Area** column; as shown here:



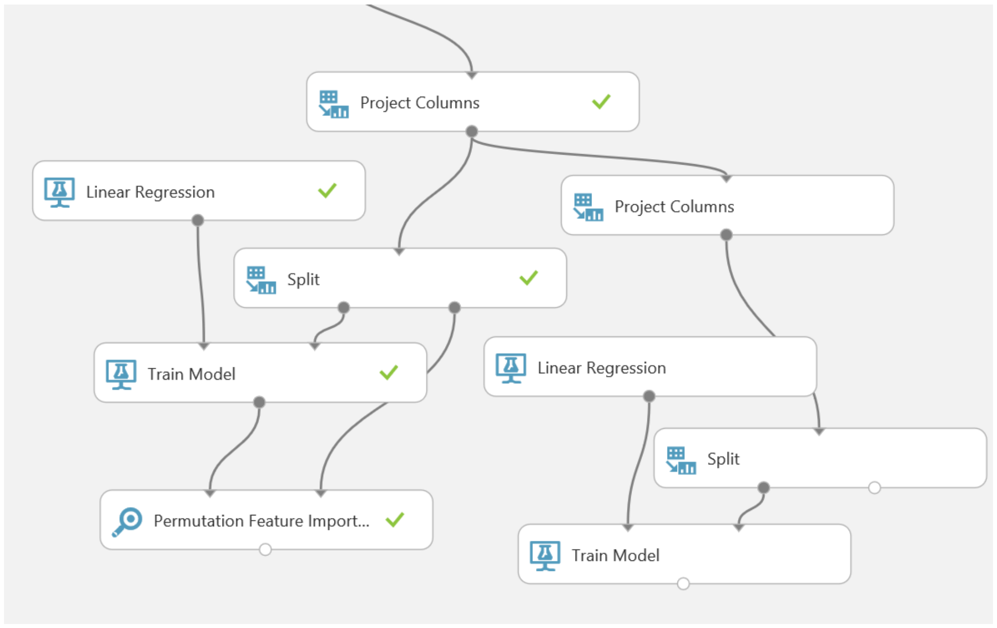
Begin With: All columns

Exclude | column names | GlazingArea

3. Verify that your experiment after the **Project Columns** module resembles this:



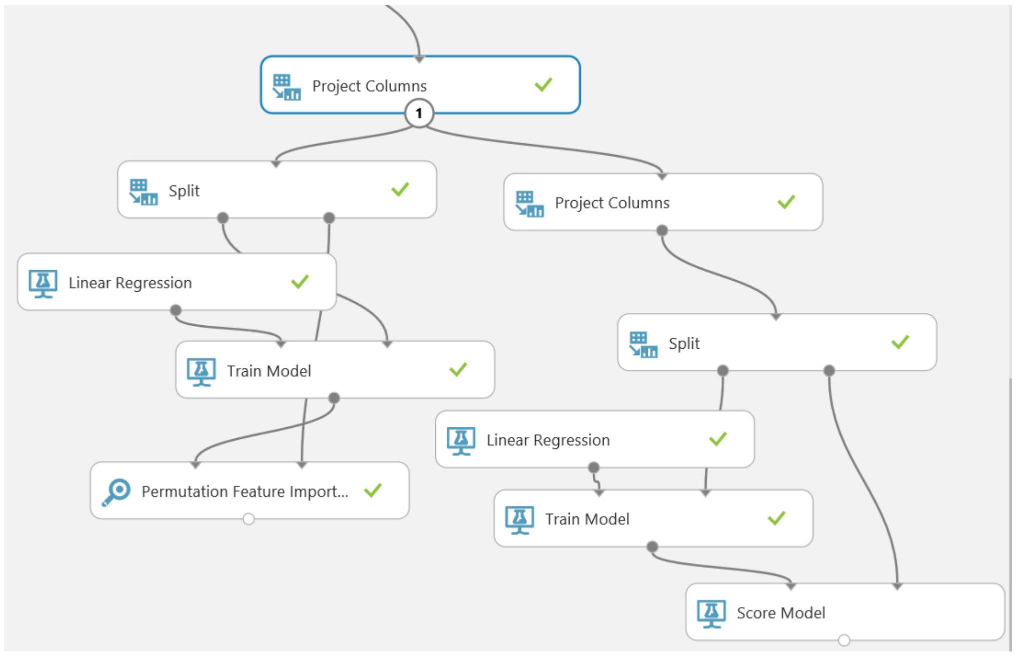
4. Click and drag around the **Linear Regression**, **Split**, and **Train Model** modules to select them, and then copy and paste them. Move the new copies of the modules so that they are under the **Project Columns** module that you added at the beginning of this procedure.
5. Connect the output of the **Project Columns** module to the input of the pasted **Split** module so that your experiment after the **Normalize Data** module resembles the following image. Then save and run the experiment.



6. When the experiment has finished, visualize the output for the second **Train Model** module (which uses the reduced set of columns from the **Project Columns** module), and note that the relative weighted feature columns do not include **Glazing Area**, which you removed from the model.

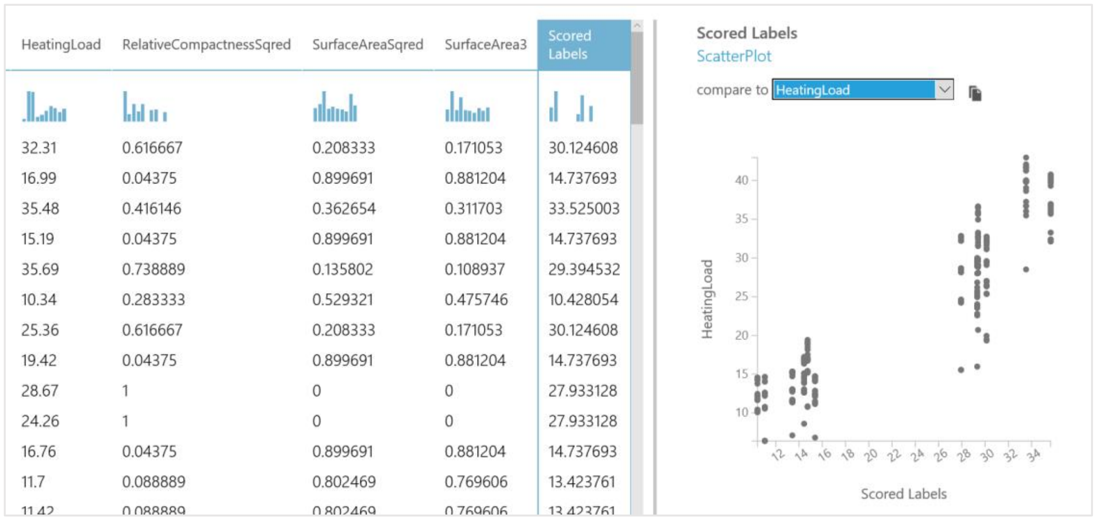
Score a Model

1. Search for the **Score Model** module, and drag it to the canvas below the second **Train Model** module. Then connect the output from second **Train Model** module to the left input of the **Score Model** module, and connect the right output from the copied **Split** module (which represents the test data set) to the right input of the **Score Model** module; as shown in the following image:



2. Save and run the experiment.

3. When the experiment has finished, visualize the output of the **Score Model** module.
4. Note that the output includes a **Scored Labels** column, which contains the predicted value for the **Heating Load** label (which is also included in the output).
5. Select the **Scored Label** column and in the **Visualizations** area, in the **compare to** list, select **Heating Load**. Note that the scatter plot shows an approximately linear correlation between the label predicted by the model and the actual label value in the test data, as shown here:



HeatingLoad	RelativeCompactnessSqred	SurfaceAreaSqred	SurfaceArea3	Scored Labels
32.31	0.616667	0.208333	0.171053	30.124608
16.99	0.04375	0.899691	0.881204	14.737693
35.48	0.416146	0.362654	0.311703	33.525003
15.19	0.04375	0.899691	0.881204	14.737693
35.69	0.738889	0.135802	0.108937	29.394532
10.34	0.283333	0.529321	0.475746	10.428054
25.36	0.616667	0.208333	0.171053	30.124608
19.42	0.04375	0.899691	0.881204	14.737693
28.67	1	0	0	27.933128
24.26	1	0	0	27.933128
16.76	0.04375	0.899691	0.881204	14.737693
11.7	0.088889	0.802469	0.769606	13.423761
11.42	0.088889	0.802469	0.769606	13.423761

Summary

In this lab, you used custom Python or R code to generate new feature columns. Then you created a model in Azure ML and examined the effectiveness of the features used to predict a label. You then created a new model with a pruned feature set, and scored it to evaluate its accuracy.

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