Data Science and Machine Learning Essentials

Lab 3C – Evaluating Models in Azure ML

By Stephen Elston and Graeme Malcolm

Overview

In this lab, you will learn how to evaluate and improve the performance of machine learning models.

Note: This lab assumes that you have completed the previous labs in this module. If you have not completed Lab 3B, you can copy the experiment from the Cortana Analytics Gallery.

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.

Having created and scored machine learning models you are ready to evaluate the performance of these models. You will evaluate your model using summary statistics produced by the Azure ML **Evaluate Model** module. Next, you will perform an in-depth evaluation of model errors using custom R or Python code.

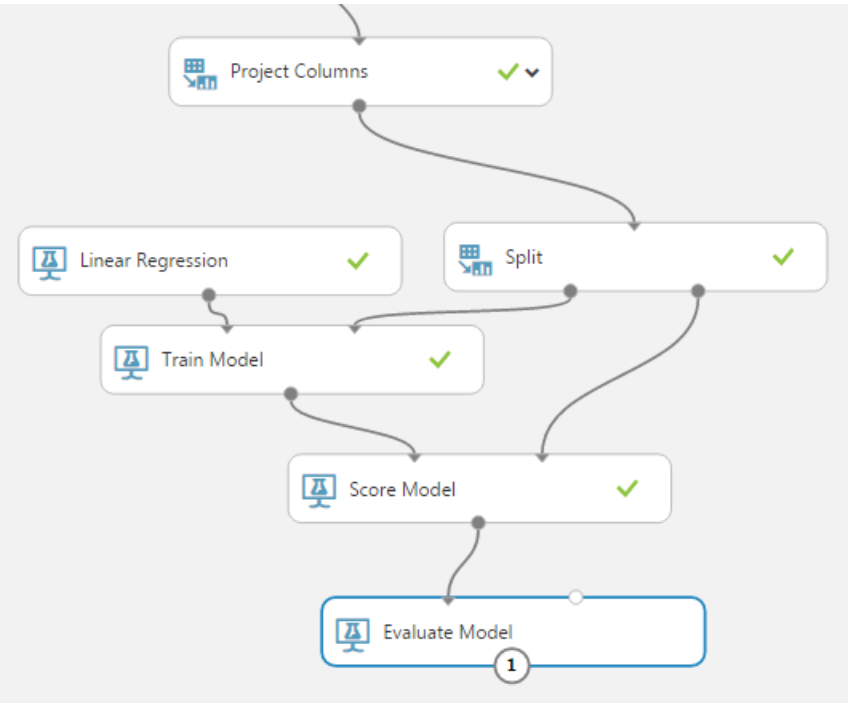
Evaluating Model Performance

Having created a machine learning model you are ready to evaluate the performance of the model. You will evaluate your model using summary statistics produced by the Azure ML Evaluate Model module. In subsequent exercises, you will perform an in-depth evaluation of model errors using custom R or Python code.

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

Evaluate a Machine Learning 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. If you prefer to work with R, save a copy of the **Modeling (R)** experiment you created in Lab 3B as **Evaluation (R)**. Otherwise, save a copy of the **Modeling (Python)** experiment as **Evaluation (Python)**. If you did not complete Lab 3B, you can copy the appropriate **Modeling** experiment from the collection for this course in the Cortana Analytics Gallery at http://gallery.cortanaanalytics.com/Collection/5bfa7c8023724a29a41a4098d3fc3df9.
3. Start with the model with the reduced feature set (filtered with a second **Project Columns** module). Search for the **Evaluate Model** module and drag it onto your canvas. Connect the **Scored Data Set** output port of the **Score Model** module to the left hand input of the **Evaluate Model** module. Your experiment should now look like this:



Tip. When evaluating a single model, always use the left input to the **Evaluate Model** module. The right input allows you to compare performance of another model to the performance of the first model.

4. Save and run the experiment. When the experiment is finished, visualize the **Evaluation Result** port of the **Evaluate Model** module and review the performance statistics for the model.

   Overall, these statistics should be promising, if not ideal. The **Coefficient of Determination,** is a measure of the reduction in the variance, between the raw label and the model error; squared error. This static is often referred to as R^2. A perfect model would have a **Coefficient of Determination** of 1.0, all the variance in the label is explained in the model. **Relative Squared Error** is the ratio of the variance or squared error of the model divided by the variance of the data. A perfect model would have a **Relative Squared Error** of 0.0, all model errors are zero. You should observe that the statistics for your model are some distance from ideal.

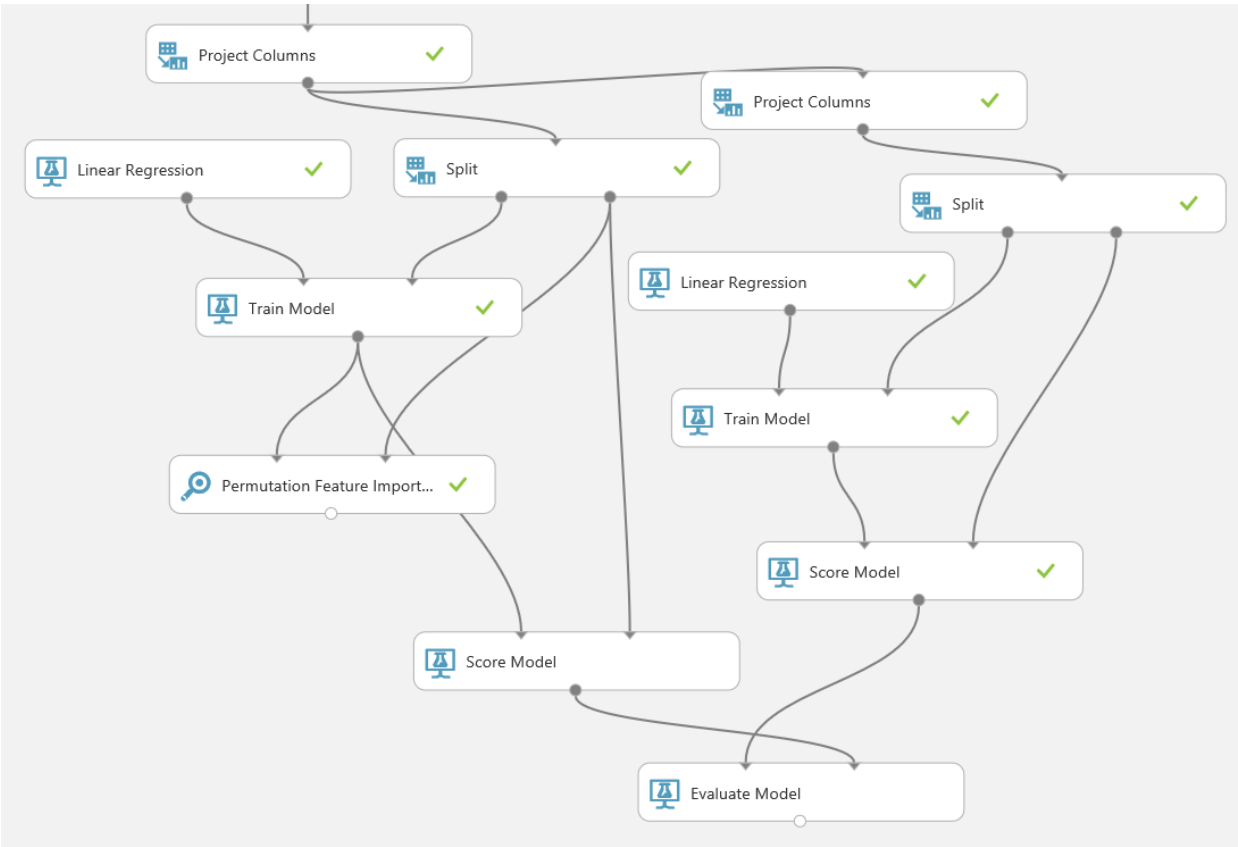
5. Close the evaluation results.

Compare Model Performance

Now that you have computed performance statistics for one model you can compare these figures to those from another model.

1. Search for the **Score Model** module, and drag it to the canvas below the **Train Model** module for the model that you have not so far evaluated. Then connect the output from unevaluated **Train Model** module to the left input of the **Score Model** module, and connect the right output from the **Split** module that defines the test data set for that model to the right input of the **Score Model** module. Then finally, connect the output from the new **Score Model** module to the right input port of the **Evaluate Model** module that is already connected to the **Score Model** module for the other model.

   Your experiment should now look similar to this:



2. Save and run the experiment. When the experiment is finished, visualize the **Evaluation Result** port of the **Evaluate Model** module and review the performance statistics for both models.
3. Compare the performance statistics for the models. For example, look at the **Relative Squared Error** and the **Coefficient of Determination**. You see that the model on the right performs better than the model on the left.

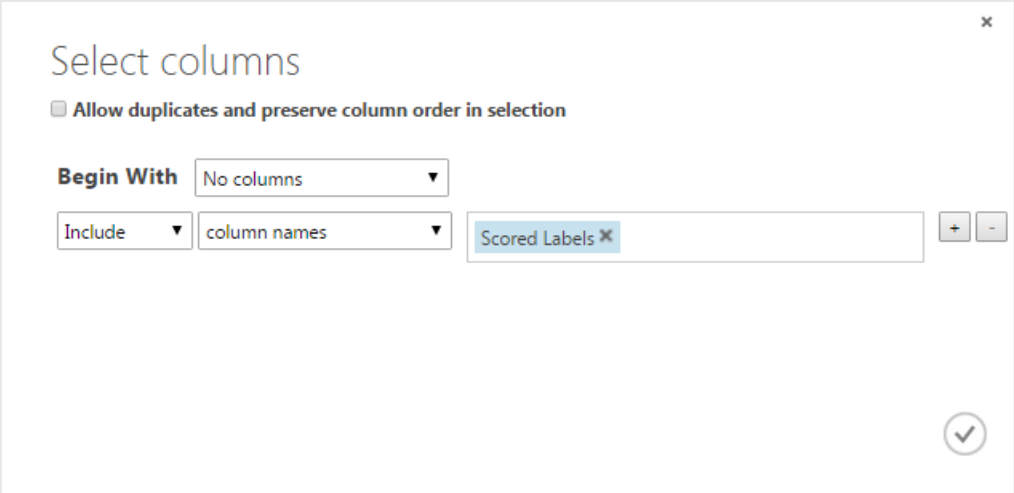
   Our feature pruning was not successful, and the **Glazing Area** feature contained useful information. This is likely an indication of some nonlinear behavior which our model has not properly captured. We will investigate this problem further with custom R or Python code.

Understanding Model Errors with R

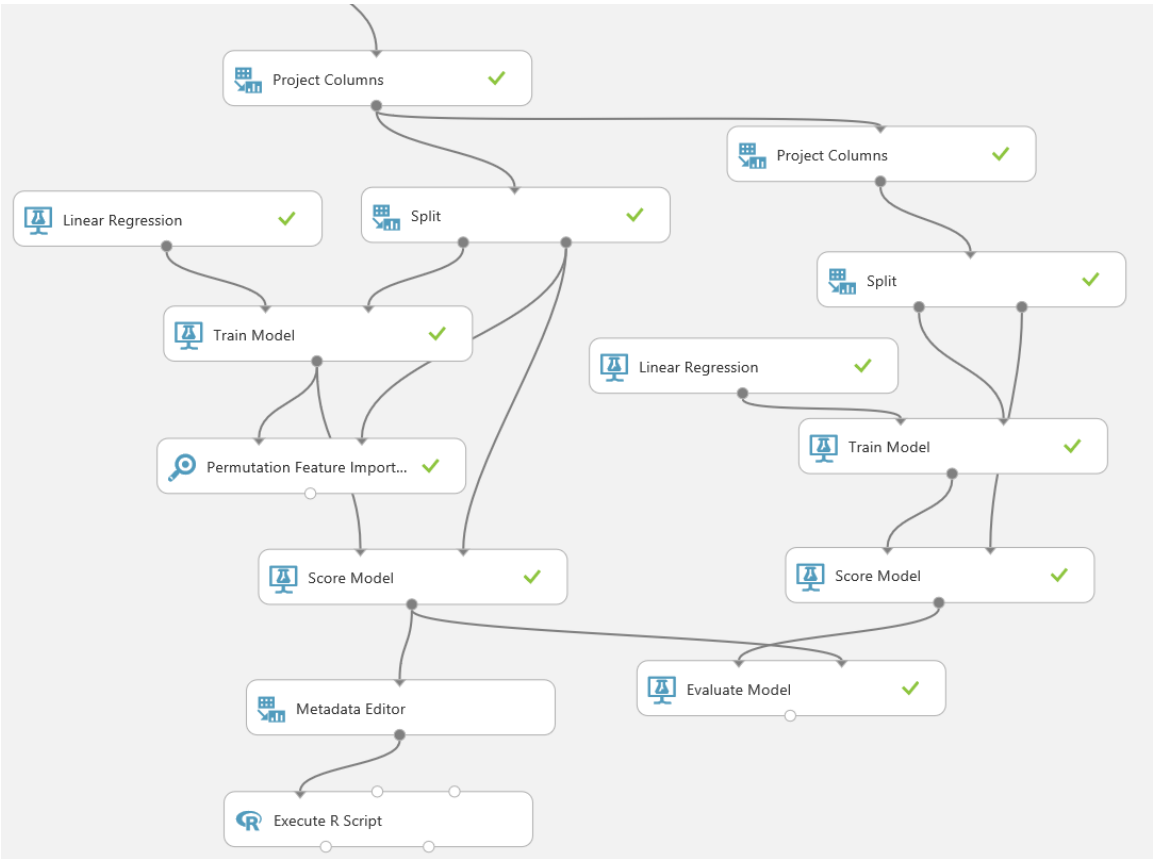
Note: If you prefer to work with Python, skip this exercise and complete the exercise *Evaluating Model Errors with Python*.

In the previous exercise you evaluated the performance of two models using the summary statistics provided by the **Evaluate Model** module. In this exercise you will evaluate the performance of a machine learning model in greater depth using custom R code.

1. In the **Evaluation (R)** experiment, search for and locate the **Metadata Editor** module. Drag this module onto your canvas. Connect the **Scored Dataset** output of the **Score Model** module for the first model you created (and which you evaluated to be performing best) to the input of the **Metadata Editor** module.
2. Select the **Metadata Editor** model and in the properties pane, launch the column selector and select the **Scored Labels** column as shown below.



3. With the **Metadata Editor** module selected, in the properties pane, in the **New column names** box type *ScoredLabels*. The output from this **Metadata Editor** model will now have a column name with no spaces, compatible with R data frame column names.
4. Search for and locate the **Execute R Script** module. Drag this module onto your canvas. Connect the **Results Dataset** output of the **Metadata Editor** module to the **Dataset1** (left hand) input of the **Execute R Script** module. Your experiment should resemble the following the figure below:



5. With the **Execute R Script** module selected, in the properties pane, replace the existing R script with the following code. You can copy this code from **VisResiduals.R** in the folder where you extracted the lab files:

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

## Compute the residuals
frame1$Resids <- frame1$HeatingLoad - frame1$ScoredLabels


## Plot of residuals vs HeatingLoad.
library(ggplot2)
ggplot(frame1, aes(x = HeatingLoad, y = Resids , by =
OverallHeight)) +
  geom_point(aes(color = OverallHeight)) +
  xlab("Heating Load") + ylab("Residuals") +
  ggtitle("Residuals vs Heating Load") +
  theme(text = element_text(size=20))

## create some conditioned plots of the residuals
plotCols <- c("SurfaceAreaSqred",
              "SurfaceArea",
              "RoofArea",
              "RelativeCompactnessSqred",
              "WallArea",
```

```
              "SurfaceArea3")

plotEERes <- function(x){
  title <- paste("Residuals vs Heating Load conditioned by", x)
  facFormula <- paste("OverallHeight ~", x)
  ggplot(frame1, aes(Resids, HeatingLoad)) +
    geom_point() +
    facet_grid(facFormula) +
    ggtitle(title)
}

lapply(plotCols, plotEERes)

## Conditioned histograms of the residuals
ggplot(frame1, aes(Resids)) +
  geom_histogram(binwidth = 0.5) +
  facet_grid(. ~ OverallHeight) +
  ggtitle('Histogram of residuals conditioned by Overall Height') +
  xlab('Residuals')

## Quantile-quantile normal plot of the residuals.
Resids35 <- frame1[frame1$OverallHeight == 3.5, ]$Resid
Resids7 <- frame1[frame1$OverallHeigh == 7, ]$Resid
par(mfrow = c(1,2))
qqnorm(Resids35)
qqnorm(Resids7)
par(mfrow = c(1,1))

## Compute the RMSE values of the residuals
## for both the overall and evaluation
## parts of the dataset.
rmse <- function(x){
  sqrt(sum(x^2)/length(x))
}

outFrame <- data.frame(
  rms_Overall = rmse(frame1$Resids),
  rms_35 = rmse(Resids35),
  rms_7 = rmse(Resids7))

## If in Azure output the data frame.
maml.mapOutputPort('outFrame')
```

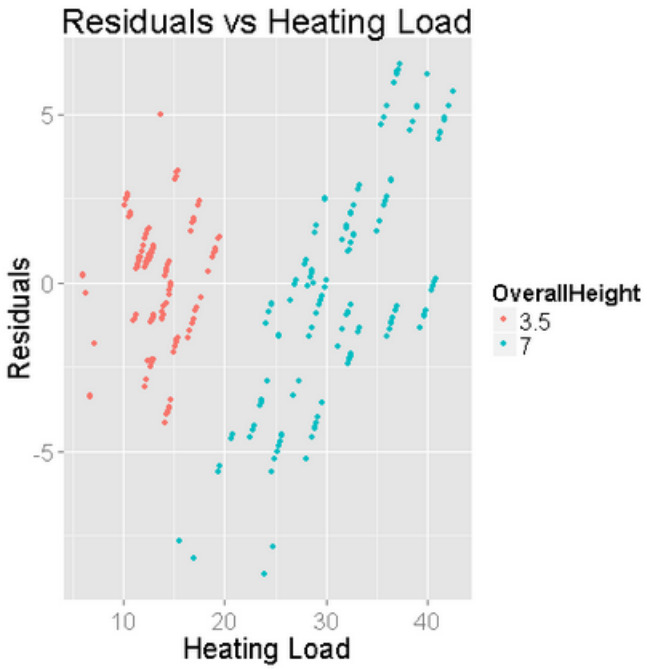
Tip: To copy code in a local code file to the clipboard, press **CTRL+A** to select all of the code, and then press **CTRL+C** to copy it. To paste copied code into the code editor in the Azure ML **Properties** pane, press **CTRL+A** to select the existing code, and then press **CTRL+V** to paste the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

This code performs the following tasks:

a. Creates a column named **Resids** containing the residuals (the differences between the original **Heating Load** label and the predicted **Scored Label** value)

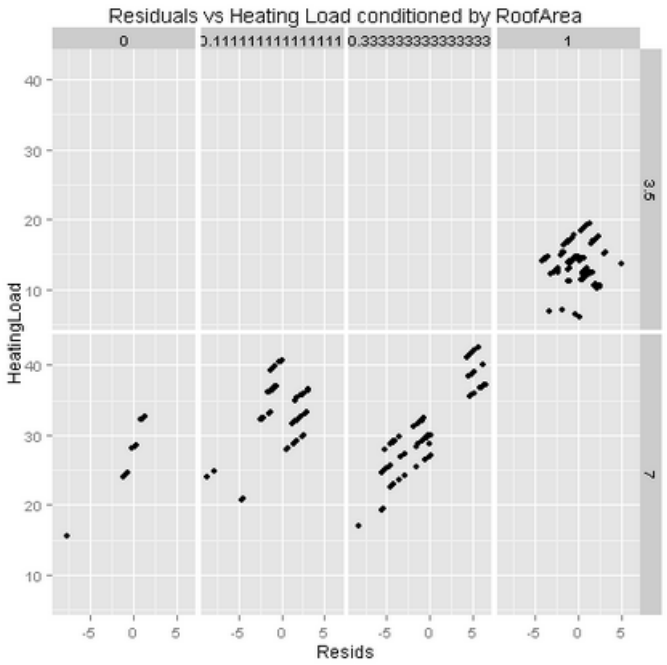
b. Creates a scatter plot of the heating load (x axis) and the residuals (y axis). These data are grouped by overall height. The points are given different colors depending on the overall height.
c. Creates scatter plots that show heating load against residuals, conditioned by overall height and other feature columns.
d. Creates a histogram that shows residuals conditioned by overall height.
e. Creates a Q-Q normal plot of the residuals.
f. Creates a function called **rmse** that returns root squared mean error. This function is called three times; once for all of the residuals, once for the residuals with an **OverallHeight** of 3.5 and once for the residuals with an **OverallHeight** of 7.

6. Save and run the experiment. Then, when the experiment is finished, visualize the **R Device port** of the **Execute R Script** module.
7. Examine the scatter plot that shows heating load against residuals conditioned by overall height, which should look similar to this:



Note the structure of these residuals with respect to the label, **HeatingLoad**. In an ideal case, the residuals should appear random with respect to the label (**HeatingLoad**). These results are not ideal. First, notice that the residuals are in two groups of clusters, based on **OverallHeight**. Second, notice that the residuals in each group have a linear structure.

8. Review the conditioned scatter plots have been created. For example, look in detail at the scatter plot conditioned on **RoofArea** and **OverallHeight**, as shown below.

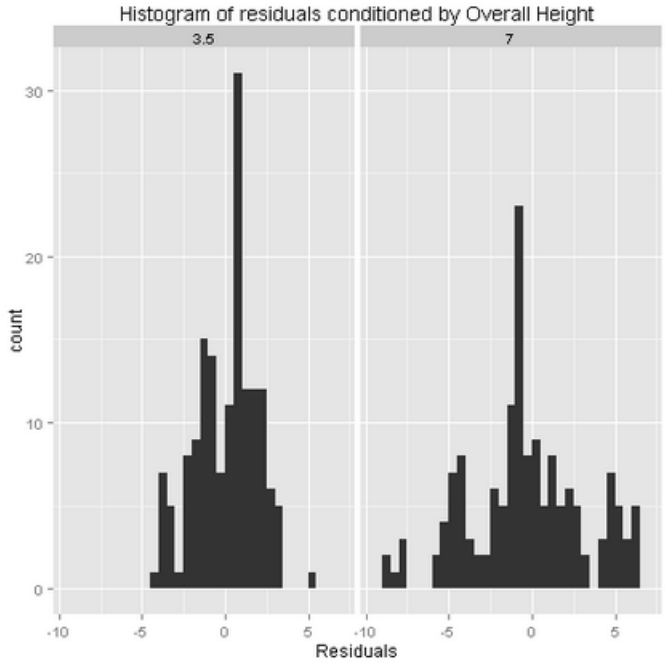


Note the shaded conditioning level tiles across the top and right side of this chart. The four tiles across the top (horizontal) show the four levels (unique values) of **RoofArea**. The two tiles on the right (vertical axis) show the two levels (unique values) of **OverallHeight**. Each scatter plot shows the data falling into the group by **RoofArea** and **OverallHeight**, with the label (Heating Load) on the vertical axis and the Residuals on the horizontal axis.

Examine this plot and notice that the residuals are not random across these plots. First, as previously noted, a linear structure is visible in several of these subplots. Second, the placement of the residuals is notably different from plot to plot across the bottom row (**OverallHeight** = 7). These observations confirm that there is nonlinear behavior not captured by this model.

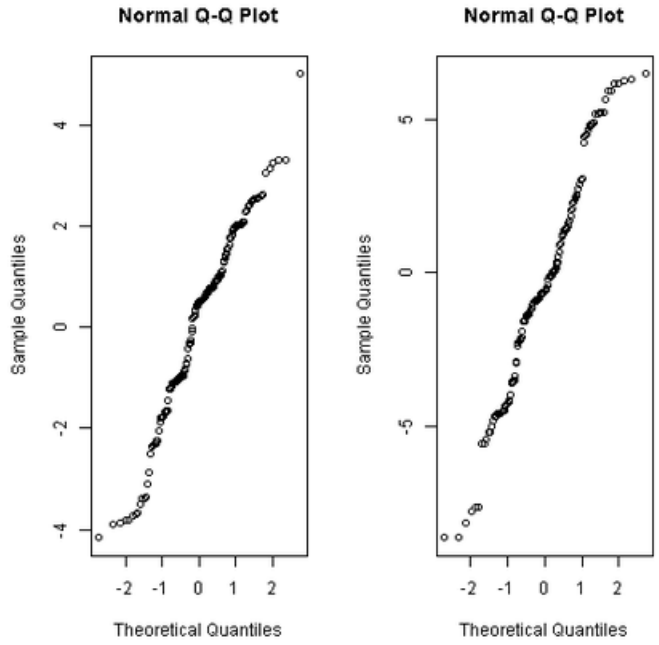
Examine the other conditioned scatter plots. You will see similar structure, further evidence that a linear model does not fit this data particularly well.

9. Examine the histogram, as shown below:



Examine these results, and note the differences in the histograms by **OverallHeight**. Further, there are apparent outliers for the **OverallHeight** of 7.

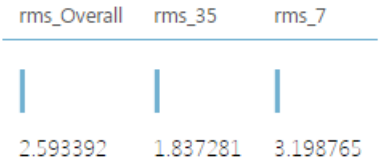
10. Review the pair of Q-Q normal plots, as shown below:



Note: A Q-Q normal plot uses the quantiles of a theoretical normal distribution on the horizontal axis vs. the quantiles of the residuals on the vertical axis. For an ideal linear model, the residuals will be normally distributed and fall close to a straight line.

You can see that pattern in neither of these plots resembles a straight line. This fact further confirms that our model fit in some distance from being idea.

11. Close the R Device output.
12. Visualize the **Result Dataset** output of the **Execute R Script** module, and review the root squared mean error results returned by the **rsme** function, as shown below.



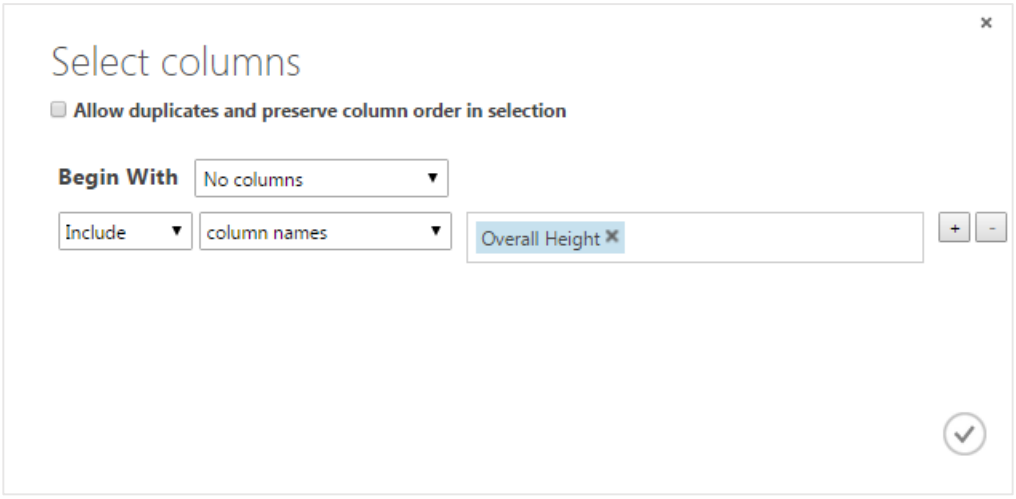
These results show significant variation between the root squared mean error calculations for the overall residuals and the residuals for **OverallHeight** of 3.5 and **OverallHeight** of 7. These results indicate that the residuals are not independent the **OverallHeight** feature.

Understanding Model Errors with Python

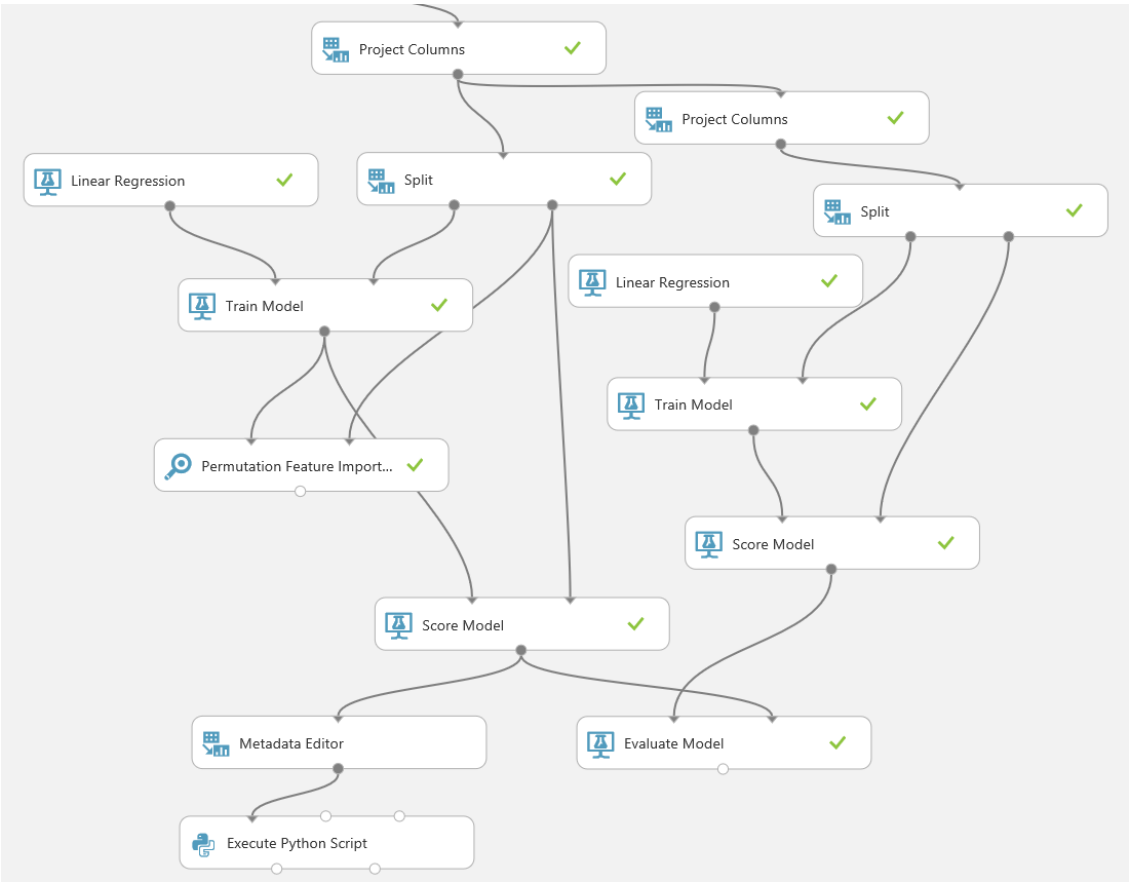
Note: If you prefer to work with R, complete the previous exercise, *Evaluating Model Errors with R*.

In the previous exercise you evaluated the performance of two models using the summary statistics provided by the **Evaluate Model** module. In this exercise you will evaluate the performance of a machine learning model in greater depth using custom Python code.

1. In the **Evaluation (Python)** experiment, search for and locate the **Metadata Editor** module. Drag this module onto your canvas. Connect the **Scored Dataset** output of the **Score Model** module for the first model you created (and which you evaluated to be performing best) to the input of the **Metadata Editor** module.
2. Select the **Metadata Editor** model and in the properties pane, launch the column selector and select the **Overall Height** column as shown below.



3. With the **Metadata Editor** module selected, in the properties pane, in the **Categorical** box select **Make non-categorical**. The output from this **Metadata Editor** model will show the **Overall Height** column as a string type which we can work with in Python.
4. Search for and locate the **Execute Python Script** module. Drag this module onto your canvas. Connect the **Results Dataset** output of the **Metadata Editor** module to the **Dataset1** (left hand) input of the **Execute Python Script** module. Your experiment should resemble the following the figure below:



5. With the **Execute Python Script** module selected, in the properties pane, replace the existing Python script with the following code. You can copy this code from **VisResiduals.py** in the folder where you extracted the lab files:

```
def rmse(Resid):
    import numpy as np
    resid = Resid.as_matrix()
    length = Resid.shape[0]
    return np.sqrt(np.sum(np.square(resid)) / length)


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

    import pandas as pd
```

```
    import pandas.tools.rplot as rplot
    import matplotlib.pyplot as plt
    import statsmodels.api as sm

## Compute the residuals
    frame1['Resids'] = frame1['Heating Load'] - frame1['Scored
Labels']

## Create data frames by Overall Height
    temp1 = frame1.ix[frame1['Overall Height'] == 7]
    temp2 = frame1.ix[frame1['Overall Height'] == 3.5]

## Create a scatter plot of residuals vs Heating Load.
    fig1 = plt.figure(1, figsize=(9, 9))
    ax = fig1.gca()
    temp1.plot(kind = 'scatter', x = 'Heating Load', \
                y = 'Resids', c = 'DarkBlue', alpha = 0.3, ax = ax)
    temp2.plot(kind = 'scatter', x = 'Heating Load', \
                y = 'Resids', c = 'Red', alpha = 0.3, ax = ax)
    ax.set_title('Heating load vs. model residuals')
    plt.show()
    fig1.savefig('plot1.png')

## Scatter plots of the residuals conditioned by
## several features
    col_list = ["Relative Compactness",
              "Surface Area",
              "Wall Area",
              "Roof Area",
              "Glazing Area"]

    for col in col_list:
        ## First plot one value of Overall Height.
        fig = plt.figure(figsize=(10, 4.5))
        fig.clf()
        ax = fig.gca()
        plot = rplot.RPlot(temp1, x = 'Heating Load', y = 'Resids')
        plot.add(rplot.GeomScatter(alpha = 0.3, colour =
'DarkBlue'))
        plot.add(rplot.TrellisGrid(['.', col]))
        ax.set_title('Residuals by Heating Load and height = 7
conditioned on ' + col + '\n')
        plot.render(plt.gcf())
        fig.savefig('scater_' + col + '7' + '.png')

        ## Now plot the other value of Overall Height.
        fig = plt.figure(figsize=(10, 4.5))
        fig.clf()
        ax = fig.gca()
        plot = rplot.RPlot(temp2, x = 'Heating Load', y = 'Resids')
```

```
        plot.add(rplot.GeomScatter(alpha = 0.3, colour = 'Red'))
        plot.add(rplot.TrellisGrid(['.', col]))
        ax.set_title('Residuals by Heating Load and height = 3.5
conditioned on ' + col + '\n')
        plot.render(plt.gcf())
        fig.savefig('scater_' + col + '3.5' + '.png')

## QQ Normal plot of residuals
    fig3 = plt.figure(figsize = (12,6))
    fig3.clf()
    ax1 = fig3.add_subplot(1, 2, 1)
    ax2 = fig3.add_subplot(1, 2, 2)
    sm.qqplot(temp1['Resids'], ax = ax1)
    ax1.set_title('QQ Normal residual plot \n with Overall Height =
3.5')
    sm.qqplot(temp2['Resids'], ax = ax2)
    ax2.set_title('QQ Normal residual plot \n with Overall Height =
7')
    fig3.savefig('plot3.png')

## Histograms of the residuals
    fig4 = plt.figure(figsize = (12,6))
    fig4.clf()
    ax1 = fig4.add_subplot(1, 2, 1)
    ax2 = fig4.add_subplot(1, 2, 2)
    ax1.hist(temp1['Resids'].as_matrix(), bins = 40)
    ax1.set_xlabel("Residuals for Overall Height = 3.5")
    ax1.set_ylabel("Density")
    ax1.set_title("Histogram of residuals")
    ax2.hist(temp2['Resids'].as_matrix(), bins = 40)
    ax2.set_xlabel("Residuals of model")
    ax2.set_ylabel("Density")
    ax2.set_title("Residuals for Overall Height = 7")
    fig4.savefig('plot4.png')


    out_frame = pd.DataFrame({ \
      'rmse_Overall' : [rmse(frame1['Resids'])], \
      'rmse_35Height' : [rmse(temp1['Resids'])], \
      'rmse_70Height' : [rmse(temp2['Resids'])] })

    return out_frame
```

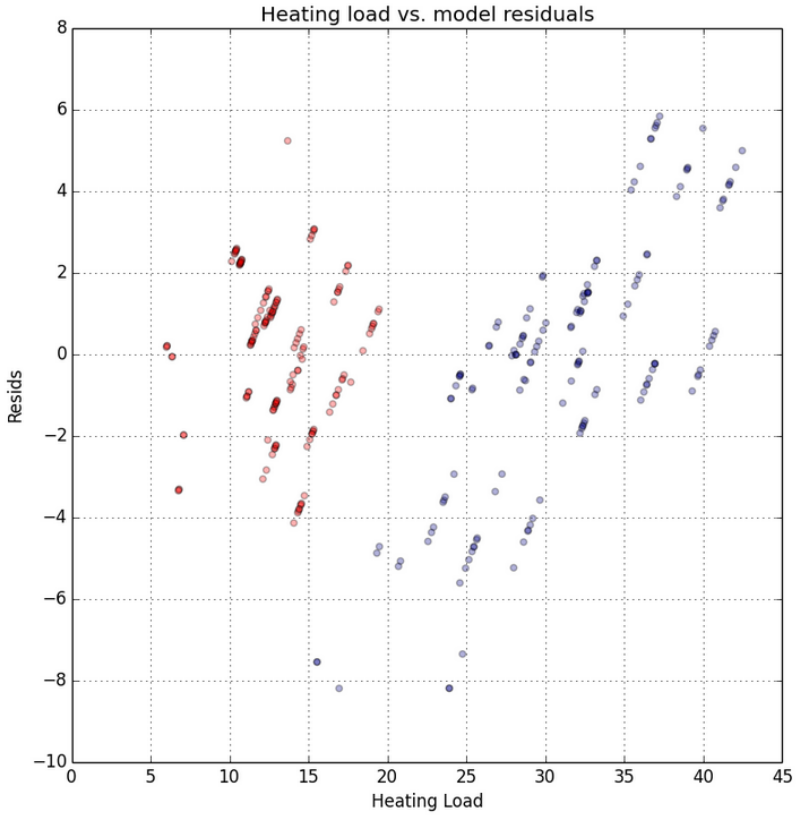
Tip: To copy code in a local code file to the clipboard, press **CTRL+A** to select all of the code, and then press **CTRL+C** to copy it. To paste copied code into the code editor in the Azure ML **Properties** pane, press **CTRL+A** to select the existing code, and then press **CTRL+V** to paste the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

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 following tasks:

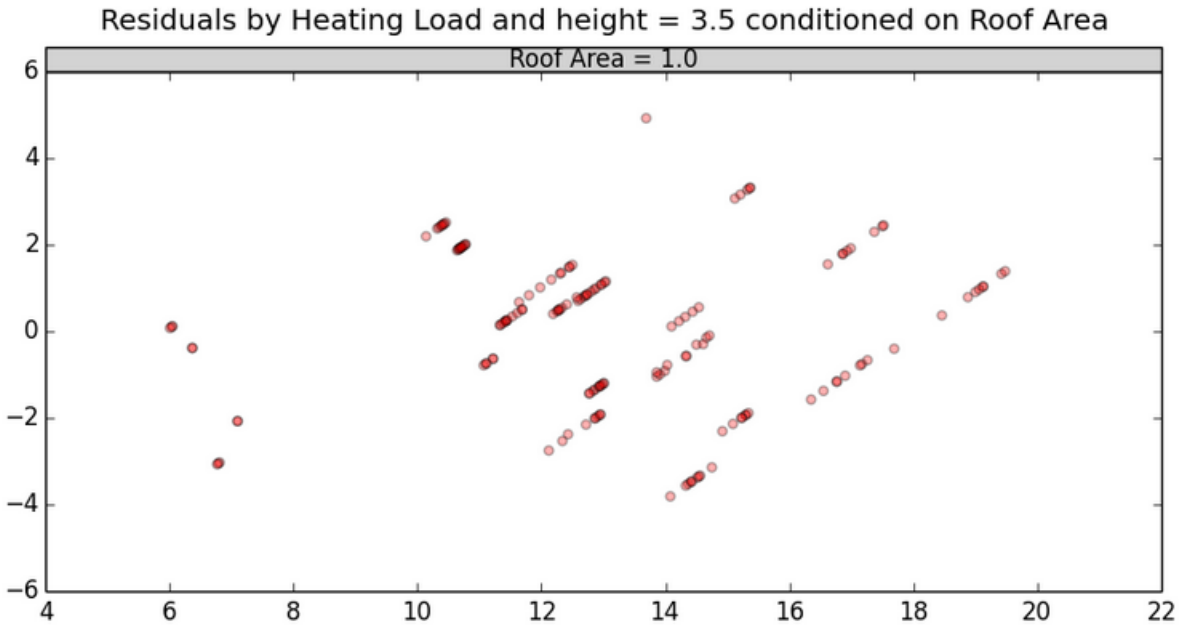
a. Creates a function called **rmse** that returns root squared mean error.
b. Creates a column named **Resids** containing the residuals (the differences between the original **Heating Load** label and the predicted **Scored Label** value)
c. Creates a data frame containing the data with an **Overall Height** value of 7, and a data frame containing the data with an **Overall Height** value of 3.5.
d. Creates a scatter plot of the heating load (x axis) and the residuals (y axis). These data are grouped by overall height. The points are given different colors depending on the overall height.
e. Creates scatter plots that show heating load against residuals, conditioned by overall height and other feature columns.
f. Creates a histogram that shows residuals conditioned by overall height.
g. Creates a Q-Q normal plot of the residuals.
h. Calls the **rmse** function three times; once for all of the residuals, once for the residuals with an **Overall Height** of 3.5 and once for the residuals with an **Overall Height** of 7. The results from these calls are returned as the output data frame for the function.

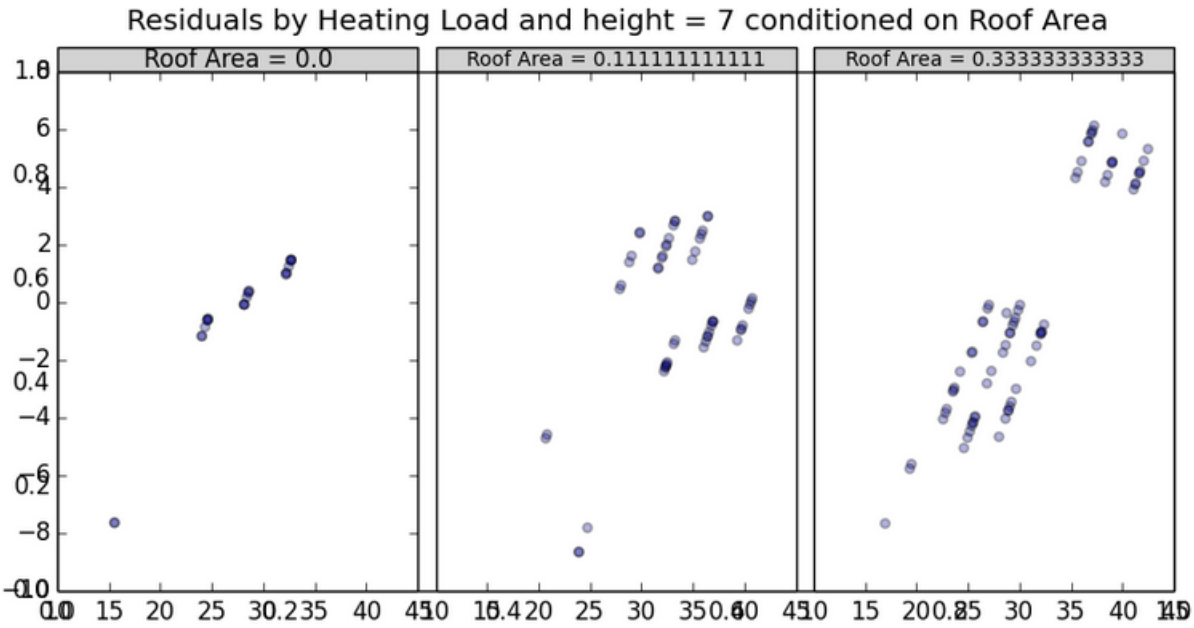
6. Save and run the experiment. Then, when the experiment is finished, visualize the **Python device port** of the **Execute Python Script** module.
7. Examine the scatter plot that shows heating load against residuals conditioned by overall height, which should look similar to this:



Examine the structure of these residuals with respect to the label, **Heating Load**. In an ideal case, the residuals should appear random with respect to the label (**Heating Load**). These results are not ideal. First, notice that the residuals are in two groups of clusters, based on **Overall Height**. Second, notice that the residuals in each group have a linear structure.

8. Review the conditioned scatter plots have been created. For example, look in detail at the scatter plots conditioned on **Roof Area** and **Overall Height**, as shown below.



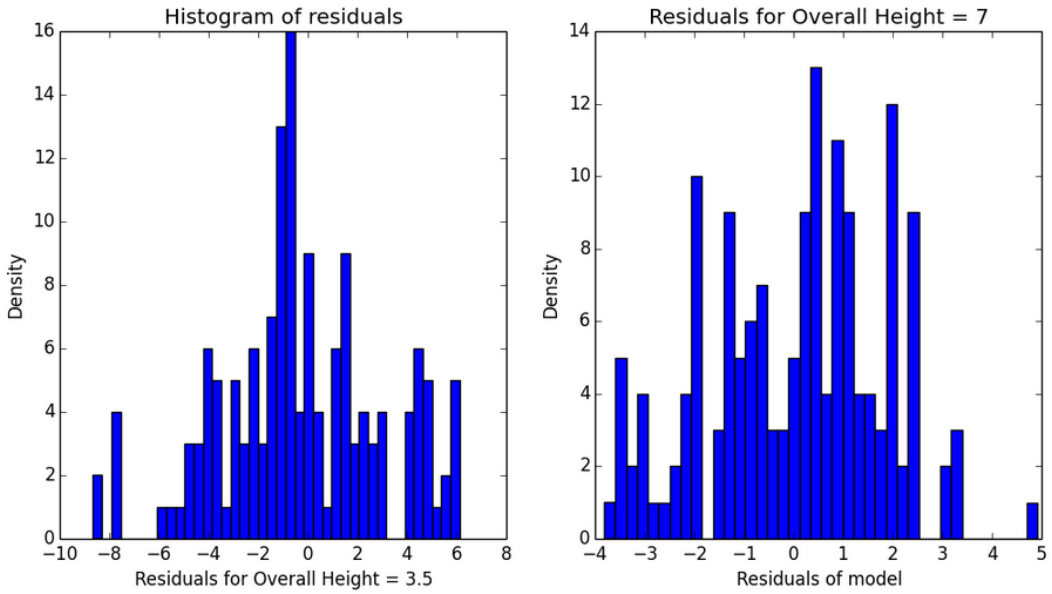


Note the shaded conditioning level tiles across the top of this charts. The first plot only has one value of **Roof Area**. The second chart has three tiles across the top (horizontal) showing the three levels (unique values) of **Roof Area**. Each scatter plot shows the data falling into the group by **Roof Area** and **Overall Height**.

Examine this plot and notice that the residuals are not random across these plots. First, as previously noted, a linear structure is visible in several of these subplots. Second, the placement of the residuals is notably different from plot to plot across the bottom row (Overall Height = 7). These observations confirm that there is nonlinear behavior not captured by this model.

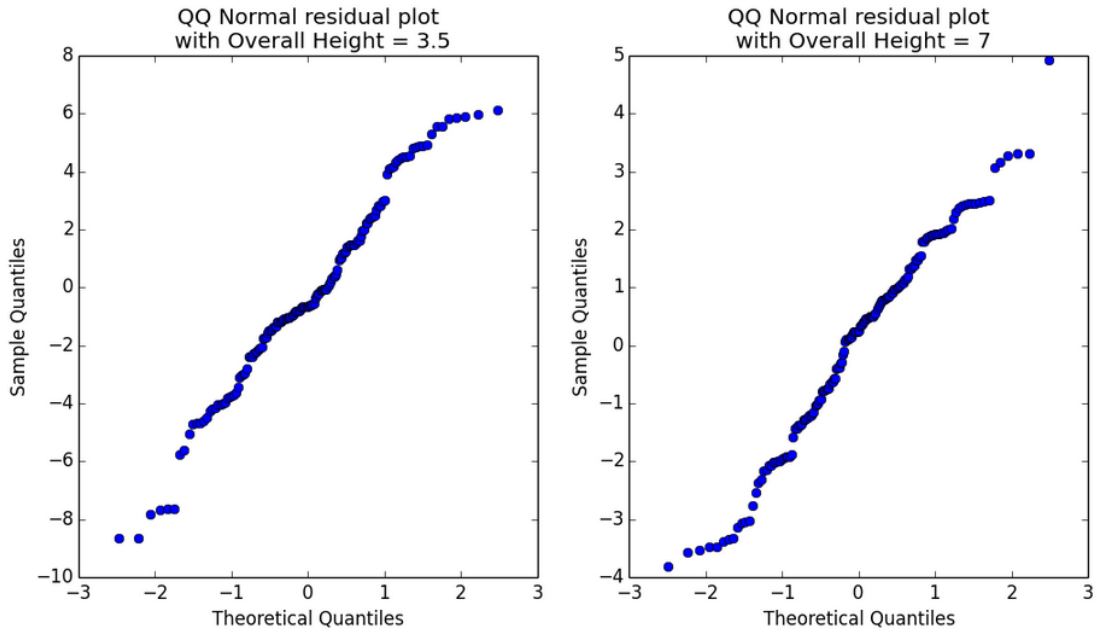
Examine the other conditioned scatter plots. You will see similar structure, further evidence that a linear model does not fit this data particularly well.

9. Examine the histogram, as shown below:



Examine these results, and note the differences in the histograms by **OverallHeight**. Further, there are apparent outliers for the **OverallHeight** of 7.

10. Review the pair of Q-Q normal plots, as shown below:



Note: A Q-Q normal plot uses the quantiles of a theoretical Normal distribution on the horizontal axis vs. the quantiles of the residuals on the vertical axis. For an ideal linear model, the residuals will be normally distributed and fall close to a straight line.

You can see that pattern in neither of these plots resembles a straight line. This fact further confirms that our model fit in some distance from being idea.

11. Close the Python device output.
12. Visualize the **Result Dataset** output of the **Execute R Script** module, and review the root squared mean error results returned by the **rsme** function, as shown below.

rmse_35Height	rmse_70Height	rmse_Overall
3.198765	1.837281	2.593392

These results show significant variation between the root squared mean error calculations for the overall residuals and the residuals for **Overall Height** of 3.5 and **Overall Height** of 7. These results indicate that the residuals are not independent the **Overall Height** feature.

Summary

In this lab, you evaluated a model in Azure ML and examined the errors in the scored or predicted values. Your understanding of model performance and errors can be used to improve model performance.

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