# Predictive Maintenance Hand-on Workshop

## Hands-On Pre-Lab A – Prepare the Training Data

#### Summary

In most machine learning endeavors, you’re likely first analyze/visualize applicable data prior to building a predictive model. To familiarize yourself with the source data, open the PredictiveMaintenance\_VisualInspection.pbix file using the Power BI Desktop application (www.powerbi.com). We will explore the visualizations together.

**Note (October 2020)** – Since this workshop was originally created, the original Power BI Desktop file has been updated/enhanced. Both R and Python examples have been added - along with a page demonstrating the Key Influencers visualization. If you want to explore the R and Python pages, you may receive errors that point to package dependencies. Refer to the R/Python documentation to learn more about how to install the necessary packages.

#### Before you get started

Find the PredictiveMaint\_Workshop folder on your computer.

## ~~Hands-On Pre-Lab B – Working in R (Time Permitting)~~

#### ~~Summary~~

~~Many data scientists use one or more open-source languages to build machine learning models. Two of the more popular languages, R and Python, are supported in Azure Machine Learning as custom modules. These modules provide a “bridge” between Azure ML’s built-in capabilities and what can be accomplished with a scripting language. The predictive maintenance templates use a fair amount of R scripts, so we are going to briefly familiarize ourselves with R.~~

#### ~~Before you get started~~

~~Find the PredictiveMaint\_Workshop folder on your laptop. Open the RParseInputEgv2.R file in RStudio.~~

## Hands-On Lab 1 – Prepare the Training Data

#### Summary

In this lab, we recreate a portion of the sample Azure template experiment **Predictive Maintenance: Step 1 of 3, data preparation and feature engineering.** This particular experiment is primarily focused on cleaning/augmenting the source files – an unglamorous, but very important, step in any machine learning process.

#### Before you get started

Open up the Machine Learning Studio in your browser - <https://studio.azureml.net>. Use the credentials provided by the instructor.

**Note (October 2020) –** Since this workshop was created, quite a bit has changed in the area of Machine Learning! You may want to try recreating these experiments in the new Azure Machine Learning Designer (<https://docs.microsoft.com/en-us/azure/machine-learning/overview-what-is-machine-learning-studio>) and/or via automated machine learning (<https://docs.microsoft.com/en-us/azure/machine-learning/concept-automated-ml>).

|  |  |
| --- | --- |
| Step | Directions/Notes |
| 1. For reference purposes, go ahead and first import the sample experiment from the Azure Gallery. We will be creating a new experiment from scratch, but it may be helpful to refer to the complete sample as you work through this tutorial.   Note – All seven experiments – along with the documentation – may be found here - <https://gallery.cortanaintelligence.com/Collection/Predictive-Maintenance-Template-3>. | 1. From the Azure Machine Learning Studio, click the **Gallery** link (located near the top of your browser) 2. In the Search bar, enter **Predictive Maintenance.** Find and then click the applicable experiment, and then click the **Open in Studio** link to add this experiment to your workspace   <https://gallery.cortanaintelligence.com/Experiment/Predictive-Maintenance-Step-1-of-3-data-preparation-and-feature-engineering-2> |
| 1. In the Azure Studio, click the New Icon (bottom of the browser) and click **Blank Experiment**  * Rename your experiment to something like **HOL 1 - Prepare Training Data** * In the Search box (which contains the text **Search experiment items**), enter **Reader** (update – this is now called **Import Data** – but using Reader will still work as a filter) to filter the module toolbox. Drag the Reader module to the Experiment pane. * In the Properties window for the Reader module, Select/Enter the following values * Data Source – Web URL via HTTP * URL - <http://azuremlsamples.azureml.net/templatedata/PM_train.txt> * Data format – TSV (don’t select the has headers checkbox)   At this point, you can Save your experiment (click the **Save** button near the bottom on your browser).  You can Run your experiment at any time (click the Run button near the bottom on your browser) – and then view the output of a particular step – we will do this shortly. |  |
| 1. Now we’ll use R to transform this file into a set of columns  * In the Search box, enter **R Script**. Drag the **Execute R Script module** to the Experiment pane (place it a bit underneath the Reader module). * Connect the Reader module to the Execute R Script module by clicking on the Circle (bottom of the Reader module) and then, while still holding your left mouse button down, dragging the arrow to the left top Circle of the Execute R Script module.      * In the Properties window for the Execute R Script module, delete the existing script, and then paste in the following R Code   # This module parse the input data into multiple column data frame  # with appropriate column names  # Map 1-based optional input ports to variables  dataset <- maml.mapInputPort(1) # class: data.frame  names(dataset) <- "V1"  # delete the extra space at the end of the lines  dataset$V1 <- gsub(" +$","",dataset$V1)  # convert to character  dataset$V1 <- as.character(dataset$V1)  # parse the input into multiple columns and generate the data frame  m <- sapply(dataset$V1,strsplit, split=" ")  df <- data.frame(matrix(unlist(m), nrow=length(m), byrow=T), stringsAsFactors=FALSE)  # generate column names for the data frame  colnames <- c("id","cycle","setting1","setting2","setting3","s1","s2","s3","s4","s5","s6","s7",  "s8","s9","s10","s11","s12","s13","s14","s15","s16","s17","s18","s19","s20","s21")  names(df) <- colnames  # genearte outpout data  data.set <- df  # Select data.frame to be sent to the output Dataset port  maml.mapOutputPort("data.set"); |  |
| 1. Let’s run the Experiment. Click the Run button at the bottom of the Broswer. After the experiment is finished, select the Execute R Script module, click the circle labeled **1**, and then select the Visualize menu item.   You will then see a Result Dataset window; you can select columns to view statistics and Visualizations. We still have some cleanup to perform, so the statistics/visualizations are not terrible useful at this point. |  |
| 1. Next, add a Metadata Editor module to the experiment.  * Click the **Launch column selector** button, and select **all features.** * Change the data type to **Floating point.**   Tip - If you the run experiment again, and Visualize the result (select the Metadata Editor module), you’ll now see meaningful results in the Statistics/Visualizations. Note that you can Visualize results from any step in an experiment (not just the last step). |  |
| 1. Add another R Script (append to Metadata Editor) and paste in the following text to add additional labels (RUL, Label1, and Label2) to the dataset.   # This module generates the labels for the training data.  # RUL for regression models, label1 for binary classification models,  # label2 for multi-class classification models  # Map 1-based optional input ports to variables  dataset <- maml.mapInputPort(1) # class: data.frame  # user defined variables to set the windows for classifcation  w1 <- 30  w0 <- 15  # generate the column RUL (remaining useful life)  library(plyr)  # get the maximum cycle number for each id  d1 <- ddply(dataset,~id,summarise,max=max(cycle))  d2 <- merge(dataset,d1,by=c("id"))  # generate the column RUL based on the values of columns "max" and "cycle"  d2$RUL <- d2$max - d2$cycle  # exclude column "max" from the data frame  d2 <- d2[,-which(names(d2) == "max")]  # genearte label1 and label2  dataset <- d2  dataset$label1 <- ifelse(d2$RUL <= w1, 1, 0)  dataset$label2 <- ifelse(d2$RUL <= w0, 2, ifelse(d2$RUL <= w1,1,0))  # generate output data  data.set <- dataset  # Select data.frame to be sent to the output Dataset port  maml.mapOutputPort("data.set"); | sd |
| 1. We will stop here – switch over to the completed Experiment (Predictive Maintenance: Step 1 of 3, data preparation) that we imported from the Gallery. Take note of a few important points –  * In the Normalize Data step, we rescale all columns (except for the RUL, Label1, and Label2) to a 0-1 scale   + Notice how we re-use this normalization against the testing data (via an Apply Transformation module) * We remove columns with all NAs from the data set |  |
|  |  |

## Hands-On Lab 2A – Build a Regression Model

#### Summary

In this lab, we recreate a portion of the sample Azure template experiment **Predictive Maintenance: Step 2A of 3, train and evaluate regression models.** This experiment predicts time to failure i.e. how many more cycles the equipment will run prior to failure

#### Before you get started

Open the Machine Learning Studio in your browser - <https://studio.azureml.net>. Use the credentials provided by the instructor.

|  |  |
| --- | --- |
| Step | Directions/Notes |
| 1. From the Azure ML Studio, click New Experiment-Blank Experiment. Add a **Reader** (**Import Data**) module in the experiment, and in the properties section, set the following properties  * Data Source - **Web URL via HTTP.** * URL - <http://azuremlsamples.azureml.net/templatedata/PM_step1output_train.csv> * Data format – CSV * Check the **CSV or TSV had headers** checkbox |  |
| 1. Add a **Project Columns** module (connect to the Reader); in the properties pane, launch the column selector. Set as shown 2. Note – Project Columns let you exclude columns that are not useful for building a model (e.g., a unique identifier) |  |
| 1. Next, add a **Filter Based Feature Selection** module. Set the properties as follows:     **Note**: Is this step necessary? Not necessarily, but it can cut down on the time to run the experiment – and helps you understand which features have the most impact on the dependent variable. To illustrate, run your experiment, and then right-click on the 2 output of the Filter Based Feature Selection. You will see a result set as shown to the right.  Tip – To calculate the Pearson correlation value in Excel, you can use either the CORREL or PEARSON function (which takes 2 arrays as arguments. I’ve shown an example of this in the PM\_step1output\_train\_CorrelationsExample.xlsx workbook. | Pearson correlation values. In this example, RUL and a4 have a correlation value of .736241. Note this is an absolute value, as these two arrays are actually negatively correlated. |
| 1. Add another **Reader** module to pull in the testing data (you don’t need to connect this module to the other modules; we will use this a bit later).   URL - http://azuremlsamples.azureml.net/templatedata/PM\_step1output\_test.csv |  |
| 1. Now we are ready to train a regression model. We are going to add 2 modules 2. **Decision Forrest Regression –** we will not modify any of the property default values 3. **Train Model** – After hooking up the modules as shown to the right, push the launch column selector button, and select the RUL column. This specifies that RUL (remaining useful life) is the value to predict.   Tip: this layout may appear a bit confusing; think of Train Model as a generic component that accepts two modules as parameters – a specific model (e.g., Decision Forest Regression) and a training dataset. |  |
| 1. We can now score/evaluate a model. Add a **Score Model** and **Evaluate Model** module to the experiment.  * For the Score Model module, connect the Test Data (not the Training Data).   After you run the experiment, you can right click the bottom node of the Score Model to see the predicted results    Right click the bottom of the Evaluate Module to see a set of aggregate evaluation metrics (we will discuss and look at these metrics in our next lab). |  |
| 1. We will stop here – switch over to the completed Experiment (Predictive Maintenance: Step 2A of 3, train and evaluate regression models) that we imported from the Gallery. Take note of a few important points –  * The Evaluate model can be used to score/evaluate one or two models (in this experiment, there are 4 regression models, and hence 2 Evaluate Model modules). Another possible use of this capability is to score a single model two times – once with the same data used to train the model, and then another time with the holdback (testing data). This can be a useful technique when trying to understand the quality/strength of the training data * Not all regression models return the same metrics. We therefore use a bit of R Script to combine all results together |  |

## Hands-On Lab 2B – Evaluate Regression Metrics (Results)

#### Summary

In this lab, we are going to take a brief look at an Excel workbook that will help us better understand the five metrics shared across all 4 regression algorithms (Mean Absolute Error, Root Mean Squared Error, Relative Absolute Error, Relative Squared Error, and the Coefficient of Determination (aka R², pronounced “R Squared”)

#### Before you get started

Find the PredictiveMaint\_Workshop folder on your laptop. Open the workbook Demand\_Metrics\_Excel\_Template\_Tyler.xlsx in Excel.

|  |  |
| --- | --- |
| Step | Directions/Notes |
| 1. In the Demand\_Metrics\_Excel\_Template\_Tyler.xlsx workbook, navigate to the **Data** worksheet.   Note: I found this workbook from an online search. Most of the worksheets have not been changed. I did add an additional forecast (Forecast2) to the Data worksheet – along with the “Actual and Forecast” Chart - and additional cells (Accuracy Method). Some of what I added in the Accuracy Method area can also be found in the Demand Metrics worksheet – I’ve tried to simplify things a bit to match the metrics available in Azure ML. |  |
| 1. Let’s look at the first metric – Mean Absolute Error (MAE). This metric takes the average (mean) of the forecast errors (expressed as positive i.e. absolute values). A forecast “error” sometimes called a residual) is simply the difference between the actual and forecasted value. |  |
| 1. The second metric, Root Mean Squared Error, is similar to MAE, but instead it squares each forecast error, computes an average of these squared values, and takes the square root of the average. The net result is that larger errors receive a higher “weight” in the final “score”. |  |
| 1. The third metric, Relative Absolute Error, is a bit more complex. It involved the following steps:  * Absolute Deviation (from Mean Demand) – using the average of all actuals, it calculates (for each row) the absolute value of (actual – actuals average) * Cumulative Absolute Deviation – a running (cumulative) total of the prior step for each row. * Cumulative Abs Forecast. Error – a running total of the absolute value of the forecast error for each row * Relative Absolute Error - Cumulative Abs Forecast. Error / Cumulative Absolute Deviation. Calculated for each row |  |
| 1. The 4th metric, Relative Squared Error is discussed here - <http://www.saedsayad.com/model_evaluation_r.htm> |  |
| 1. The final metric, Coefficient of Determination, is also referred to a R² (R Square) or the least squared. R² is the square root of r – which is the correlation coefficient (i.e. the Pearson correlation). R² can be calculated as:   1 – ( Sum of Forecast Errors squared / Sum of Squared Deviations from the Mean Squared)  As notation….  1 – ( ² / )  For another way to understand R², see my time series forecasting article - <http://sqlmag.com/sql-server-analysis-services/understanding-time-series-forecasting-concepts>. | See the Demand Metrics worksheet for the R² calculation |
| Decent (easy to understand) discussion on common forecast evaluation methods. <http://www.forecastpro.com/Trends/forecasting101August2011.html> |  |

## Hands-On Lab 2C – Build/Evaluate a Binary Classification Model

#### Summary

In this lab, we are going to import and the run the sample Azure template experiment **Predictive Maintenance: Step 2B of 3, train and evaluate binary classification models.** This experiment predicts if an asset will fail within a certain time frame (in this case, within 30 cycles).

#### Before you get started

Open up the Machine Learning Studio in your browser - <https://studio.azureml.net>. Use the credentials provided by the instructor.

|  |  |
| --- | --- |
| Step | Directions/Notes |
| 1. Import the experiment **Predictive Maintenance: Step 2B of 3, train and evaluate binary classification models**, from the Gallery. Run the experiment |  |
| 1. Right click on the bottom node of the **Score model** module (for the Two-class logistic regression model) and select Visualize. Scroll to the far right side of the table where you will see –  * label1 (what we are trying to predict) * Scored Label – the prediction * Scored Probabilities. Anything less than .5 will correspond to a prediction (Scored Label) of 0; .5 or higher will correspond to 1. |  |
| 1. Right click on the bottom node of the leftmost **Evaluate model** module and select Visualize.   The chart show the results for two models (Two-Class Logistic Regression, and Two-Class Boosted Decision Tree). The first model (Scored dataset) is highlighted in the chart legend; if you select the other legend entry (Scored dataset to compare), the numeric results below the chart will be updated accordingly.   * Accuracy. This is an evaluation of the accuracy of the guesses (true positive + true negative) / total number of cases. For example, of the 100 cases, the two-class boosted decision tree scored (18+74) / 100 = 92%. * Precision. True Positives / (True Positives + False Positives). A measure of how often you guessed positives correctly. Example: 18/(18+1) = 94.7% * Recall – see equation below * F-Score - see equation below   Accuracy =  Precision =  Recall = = |  |

## Hands-On Lab 3A – Deploy Web Service with a Regression Model

#### Summary

In this lab, we are going deploy the regression model (Step 2A) as a web service

#### Before you get started

Open up the Machine Learning Studio in your browser - <https://studio.azureml.net>. Use the credentials provided by the instructor.

|  |  |
| --- | --- |
| Step | Directions/Notes |
| 1. In Azure ML Studio, open up the experiment Predictive Maintenance: Step 2A of 3, train and evaluate regression models; Click the **Save-Save As** button to create a copy of this experiment.  * For our Web Service, we will use the Boosted Decision Tree Regression Model – **remove the steps associated with other three models**. Your experiment should look similar to what is shown. * Run your experiment. |  |
| 1. Click the **Set Up Web Service – Predictive Web Service (recommended)** button. Click the Next buttons (several times) to see the Predictive Experiment. Note the steps Azure ML does on our behalf -  * A copy of the experiment is made (same name – ending with [Scoring Exp.]   + A Tabbed view is created (allowing you to toggle between the training and predictive experiment * The Boosted Decision Tree Regression model is saved as a Trained Model (it is given the same name as the experiment) – in the predictive   + The model now shows up under **Trained Models –**you can drag/drop into new experiments.   + Note – you can manually save a Model by right-clicking the bottom node of a model (make sure you’ve first run the experiment) and selecting the **Save as Trained Model** menu-item. With this approach, you can customize the name. You can also re-save a model if you perform further training * A Web service input is created   + Notice that the Web Service input is hooked directly to the Score Model – it therefore has the same schema as the testing data. * A Web service output is created. Output includes the original inputs and a scored value. |  |
| 1. Click the Web Services link in the browser (the Globe) – you will now see a Web Service.   You can click the Web Service to bring up additional information about the service. From the browser, you can also test the service interactively – or you can download a sample Workbook (with Macro Code) that will call the service.  Note – the workbook has some usability issues - we will discuss workarounds/alternatives together. |  |