Exercise 1: Creating a Machine Learning Model

Machine Learning is a term used to describe the development of predictive models based on historic data. There are a variety of tools, languages, and frameworks you can use to create machine learning models; including R, the Sci-kit Learn package in Python, Apache Spark, and Azure Machine Learning.

In this lab, you will use Azure Machine Learning Studio, which provides an easy to use web-based interface for creating machine learning models. The principles used to develop the model in this tool apply to most other machine learning development platforms, but the graphical nature of the Azure Machine Learning Studio environment makes it easier to focus on learning these principles without getting distracted by the code required to manipulate data and train the model.

**Create an Azure Machine Learning Studio Workspace**

1. In your web browser, navigate to https://studio.azureml.net, and if you don’t already have a free Azure Machine Learning Studio workspace, click the option to sign up and choose the **Free Workspace** option and sign in using your Microsoft account.

2. After signing up, view the **EXPERIMENTS** tab in Azure Machine Learning Studio

Upload the Lemonade Sales Dataset

1. In Azure Machine Learning Studio, click **DATASETS**. You should have no datasets of your own (clicking **Samples** will display some built-in sample datasets).

2. At the bottom left, click **+ NEW**, and ensure that the **DATASET** tab is selected.

3. Click **FROM LOCAL FILE**. Then in the **Upload a new dataset** dialog box, browse to select the **Lemonade-Sales.csv** file and enter the following details, and then click the (✓) icon.

• **This is a new version of an existing dataset**: Unselected

• **Enter a name for the new dataset**: Lemonade.csv

• **Select a type for the new dataset**: Generic CSV file with a header (.csv)

• **Provide an optional description**: Lemonade sales data.

4. Wait for the upload of the dataset to be completed, then verify that it is listed under **MY DATASETS** and click the **OK** (✓) **button** to hide the notification.

The Lemonade-Sales.csv file contains the original lemonade sales data in comma-delimited format.

Create an Experiment and Explore the Data

1. In Azure Machine Learning Studio, click **EXPERIMENTS**. You should have no experiments in your workspace yet.

2. At the bottom left, click **+ NEW**, and ensure that the **EXPERIMENT** tab is selected. Then click the **Blank Experiment** tile to create a new blank experiment.

3. At the top of the experiment canvas, change the experiment name to **Lemonade Sales Training.**

The experiment interface consists of a pane on the left containing the various items you can add to an experiment, a canvas area where you can define the experiment workflow, and a **Properties** pane where you can view and edit the properties of the currently selected item. You can hide the experiment items pane and the **Properties** pane by clicking the **<** or **>** button to create more working space in the experiment canvas.

4. In the experiment items pane, expand **Saved Datasets** and **My Datasets**, and then drag the **Lemonade-Sales.csv** dataset onto the experiment canvas.

5. Right-click the dataset output of the **Lemonade-Sales.csv** dataset and click **Visualize**

1. In the data visualization, note that the dataset includes a record, often referred to as an *observation* or *case*, for each day, and each case has multiple characteristics, or *features* – in this example, the date, day of the week, temperature, rainfall, number of flyers distributed, and the price charged per lemonade that day. The dataset also includes the number of sales made that day – this is the *label* that ultimately you must train a machine learning model to predict based on the features.

2. Note the number of rows and columns in the dataset (which is very small – real-world datasets for machine learning are typically *much* larger), and then select the column heading for the **Temperature** column and note the statistics about that column that are displayed.

3. In the data visualization, scroll down if necessary, to see the histogram for **Temperature**. This shows the distribution of different temperatures in the dataset.

4. Click the **x** icon in the top right of the visualization window to close it and return to the experiment canvas.

Explore Data in a Jupyter Notebook

Jupyter Notebooks are often used by data scientists to explore data. They consist of an interactive browser-based environment in which you can add notes and run code to manipulate and visualize data. Azure Machine Learning Studio supports notebooks for two languages that are commonly used by data scientists: R and Python. Each language has its particular strengths, and both are prevalent among data scientists. In this lab, we are going to use R.

1. Right-click the **Lemonade-Sales.csv** dataset output, and in the **Open in a new Notebook** sub-menu, click **R**. This opens a new browser tab containing a Jupyter notebook with two cells, each containing some code. The first cell contains code that loads the CSV dataset into a data frame named **dat**, similar to this:

library("AzureML")

ws <- workspace()

dat <- download.datasets(ws, "Lemonade.csv")

The second cell contains the following code, which displays a summary of the data frame:

head(dat)

2. On the **Cell** menu, click **Run All** to run all of the cells in the workbook. As the code runs, the O symbol next to **R** at the top right of the page changes to a ⚫ symbol, and then returns to O when the code has finished running.

3. Observe the output from the second cell, which shows some rows of data from the dataset.

6. Click cell 2 (which contains the code head(dat)), and then on the **Insert** menu, click **Insert Cell Below**. This adds a new cell to the notebook, under the output generated by cell 2.

7. Add the following code to the new empty cell:

# Print statistics for Temperature and Sales

summary(dat[c('Temperature', 'Sales')])

print('Standard Deviations:')

apply(dat[c('Temperature', 'Sales')],2,sd)

# Print correlation for temperature vs Sales

print('Correlation:')

cor(dat[c('Temperature', 'Sales')])

# Plot Temperature vs Sales

plot(dat$Temperature, dat$Sales, xlab="Temperature", ylab="Sales")

8. With the cell containing the new code selected, on the **Cell** menu, click **Run Cells and Select Below** (or click the **|** button on the toolbar) to run the cell, creating a new cell beneath.

9. View the output from the code, which consists of descriptive statistics for the **Temperature** and **Sales** columns, the correlation matrix for **Temperature** and **Sales**, and a scatterplot chart of **Temperature** vs **Sales**

10. On the **File** menu, click **Close and Halt** to close the notebook and return to the experiment in Azure Machine Learning Studio.

Prepare Data for Model Training

1. In the **Lemonade Sales Training** experiment, visualize the output of the **Lemonade-Sales.csv** dataset and select the **Rainfall** column.

2. Under the statistics for this column, view the histogram and note that it is right-skewed.

3. In the **compare to** drop-down list, select **Sales** and view the resulting scatterplot

4. Note the curved nature of the relationship, and then select the **Rainfall** log scale checkbox and view the updated scatterplot

5. Note that this partially “straightens” the relationship to make it more linear; so converting **Rainfall** to its natural log may make it easier to define a linear function that relates these columns. Using the log scale for **Sales** would straighten it even more, but since **Sales** already has a linear relationship with other columns (as we saw with **Temperature** in the notebook visualizations), it may be best to leave that column as it is.

6. Close the visualization.

7. In the Search experiment items box, enter **Math**. Then drag the **Apply Math Operation** module onto the canvas, under the **Lemonade.csv** dataset, and connect the output of the **Lemonade-Sales.csv** dataset to the **Apply Math Operation** module

8. With the **Apply Math Operation** module selected, in the **Properties** pane, select the **Basic** category and the **Ln** basic function

9. In the **Properties** pane, click **Launch column selector**, and then in the column selector, on the **By Name** tab, move all columns other than **Rainfall** to the left so that only the Rainfall column is listed in the **Selected columns** list. Then click (✓).

10. In the **Properties** pane, in the **Output mode** list, select **Inplace**, so that the existing **Rainfall** column is replaced with its natural log.

11. At the bottom of the page, click the **Run** () button to run the experiment so far. Wait for the experiment to finish running. A green ✓icon in the **Apply Math Operation** module indicates that it has run.

12. Visualize the output of the **Apply Math Operation** module and select the **Rainfall** column. Then compare the visualization of this column with **Sales** as you did before, and verify that the default relationship is straighter without selecting the log scale.

13. In the **compare to** drop-down list, select **Temperature**, and view the relationship between rainfall and temperature.

Take a close look at the scale on each axis. Temperatures range from 0 to over 100, while the log of rainfall is fractional between 0 and 0.8. If you were to compare all of the features in the dataset, you’d find that there is some disparity between the scales of values – for example, the number of flyers distributed ranges from 9 to 80, but the price of a lemonade ranges from 0.3 to 0.5. When training a machine learning model, features with larger scales of value can dominate features on smaller scales; so, it’s generally useful to *normalize* the numeric features so that they are on a similar scale while maintaining the correct proportional distances between values for any given feature. We’ll do this next.

14. Close the visualization and return to the experiment canvas.

15. In the **Search experiment items** box, type **Normalize**, and then drag a **Normalize Data** module to the canvas and connect it to the output from the **Apply Math Operation** module

16. Configure the Normalize Data module properties as follows:

• Transformation method: ZScore

• Use 0 for constant columns when checked: Checked

• Selected columns: Temperature and Flyers

ZScore normalization works well for numeric features that have an approximately normal distribution.

17. Select the **Normalize Data** module and on the **Run** menu, click **Run Selected** to run the data flow.

18. After the experiment has been run, add a second **Normalize Data** module to the experiment, and connect the **Transformed dataset** (left) output of the first **Normalize Data** module to its input

* 1. 19. Configure the new **Normalize Data** module as follows
  2. • **Transformation method**: MinMax
  3. • **Use 0 for constant columns when checked**: Checked
  4. • **Selected columns**: Rainfall and Price
  5. MinMax normalization works well for features that are not normally distributed.

20. Run the experiment.

21. Visualize the **Transformed Dataset** (left) output of the last **Normalize Data** module and view the **Temperature**, **Rainfall**, **Flyers**, and **Price** columns. These have all been normalized so that the values are of a similar scale, while maintaining the proportional distributions within each feature

22. Close the visualization and return to the experiment canvas

Train a Regression Model

1. Search for the **Edit Metadata** module, add one to the experiment, and connect the **Transformed dataset** (left) output of the second **Normalize Data** module to its input

* 1. 2. Configure the properties of the **Edit Metadata** module as follows:
     1. • **Selected columns**: Date, Day, and Sales

• **Data type**: Unchanged

* 1. • **Categorical**: Unchanged
  2. • **Fields**: Clear feature
  3. • **New column names**: *leave blank*

The **Date** and **Day** columns aren’t likely to help predict sales volumes, and **Sales** column is the label the model will predict; so these fields should not be used as features to train the model.

3. Search for the **Split Data** module, add one to the canvas, and connect the **Results dataset** output of the **Edit Metadata** module to its input

* 1. 4. Configure the **Split Data** module properties as follows:
     1. • **Splitting mode**: Split Rows
  2. • **Fraction of rows in the first output dataset**: 0.7
  3. • **Randomized split:** Checked
  4. • **Random seed**: 0
  5. • **Stratified split**: False

You are going to train a regression model, which is a form of *supervised* learning that predicts numeric values. When training a supervised learning model, it is standard practice to split the data into a training dataset and a test dataset, so that you can validate the trained model using test data that contains the actual label values the model is being trained to predict. In this case, you are going to use 70% of the data to train the model while withholding 30% of the data with which to test it.

5. Select the **Split Data** module, and on the **Run** menu, click **Run selected**.

6. In the **Search experiment items** box, type **Linear Regression**, and then drag a **Linear Regression** module to the canvas, to the left of the **Split Data** module.

7. In the **Search experiment items** box, type **Train Model**, and then drag a **Train Model** module to the canvas, under the **Linear Regression** and **Split Data** modules.

8. Connect the **Untrained Model** output of the **Linear Regression** module to the **Untrained Model** (left) input of the **Train Model** module. Then connect the **Result dataset1** (left) output of the **Split Data** module to the **Dataset** (right) input of the **Train Model** module

9. Select the **Linear Regression** module and review its default properties. These parameters are used to *regularize* the training of the model – that is, minimize bias so that the model generalizes well when used with new data.

10. Select the **Train Model** module and use the column selector to select the **Sales** column – this is the label that the model will be trained to predict.

11. In the **Search experiment items** box, type **Score Model**, and then drag a **Score Model** module to the canvas, under the **Train Model** module.

12. Connect the **Trained model** output of the **Train Model** module to the **Trained model** (left) input of the **Score Model** module. Then connect the **Results dataset2** (right) output of the **Split Data** module to the **Dataset** (right) input of the **Score Model** module

The **Score Model** module applies the trained model to the withheld test dataset, predicting a scored label (in this case, the number of sales).

13. In the **Search experiment items** box, type **Evaluate Model**, and then drag an **Evaluate Model** module to the canvas, under the **Score Model** module. Then connect the **Scored dataset** output of the **Score Model** module to its **Scored dataset** (left) input

The **Evaluate Model** module compares the scored label prediction to the actual label value in the withheld test dataset (in this case **Sales**) and calculates a range of metrics that can be used to evaluate how accurately the model has predicted the labels.

14. Run the experiment and wait for it to complete.

15. When the experiment has completed, visualize the output of the **Scored Model** module and select the **Scored Labels** column header. This column contains the sales predicted by the model.

16. View the histogram for the **Scored Labels** column, and in the **compare to** list, select **Sales** to see a scatterplot of predicted sales against actual sales

17. The scatterplot shows a fairly straight diagonal line, indicating that the predicted sales match the actual sales fairly closely.

18. Close the visualization, and then visualize the output of the **Evaluate Model** module.

19. Review the metrics for the model

Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are metrics that measure the *residuals* (the variance between predicted and actual values) in the same units as the label itself – in this case the number of sales. Both of these metrics indicate that on average, the model is accurate within one sale.

Relative Absolute Error (RAE) and Relative Squared Error (RSE) are relative measures of error. The closer these values are to zero, the more accurately the model is predicting.

Coefficient of Determination, sometimes known as R-Squared, is another relative measure of accuracy; but this time, the closer it is to 1, the better the model is performing.

Overall, it looks like the model is performing well.

**Note**: In reality, most models are not immediately this accurate – it usually takes several iterations to determine the best features to use in the model. Additionally, just because the model performs well with the test data, that doesn’t mean it will generalize well with new data – it may be *overfitted* to the training dataset. There are techniques that data scientists use to validate models and avoid overfitting.

20. Close the visualization and return to the experiment canvas.