Using Azure Machine Learning

Overview

Machine learning, which facilitates the analysis of large volumes of data by employing algorithms that iteratively learn from that data, is one of the hottest and fastest growing areas of computer science. Its uses range from credit-card fraud detection and self-driving cars to optical character recognition (OCR) and online shopping recommendations. It makes us smarter by making computers smarter. And its usefulness will only increase as more and more data comes available and our desire to perform predictive analysis from that data grows, too.

Azure Machine Learning is a cloud-based predictive-analysis service that offers a streamlined experience for data scientists of all skill levels. It's accompanied by the Azure Machine Learning Studio (ML Studio), which is a browser-based tool that allows you to build models using simple drag-and-drop gestures. It comes with a library of time-saving experiments and features best-in-class algorithms developed and tested in the real world by Microsoft businesses such as Bing. And its built-in support for R and Python means you can build custom scripts to further automate your model. Once you've built and trained your model in the ML Studio, a couple of button clicks exposes the model as a Web service that is consumable by your programming language of choice, or shares it with the community by placing it in the product gallery. On its own, Azure Machine Learning can ingest as much as 10 GB of input data, but if that's not enough, it can also read from Hive or Azure SQL Databases.

The Azure Machine Learning team have put together an outstanding "cheat sheet" that helps you decide which machine learning algoritm to use for many situations. You can download that cheat sheet at http://aka.ms/MLCheatSheet

In this lab, you will use Azure Machine Learning to model automobile prices and extract predictive data from the model. You will employ the following steps to build this experiment in Machine Learning Studio and to create, train, and score your model:

1. Create a model
    - Get the data
    - Preprocess the data
    - Define features
2. Train the model
    - Choose and apply a learning algorithm
3. Score and test the model
    - Predict new automobile prices

Objectives

In this hands-on lab, you will learn how to:

- Log in to Azure Machine Learning Studio
- Work with the Azure Machine Learning Studio
- Acquire and process data for machine-learning experiments
- Apply and test learning algorithms
- Deploy your model as a Web service so it can be accessed from code or scripts

Prerequisites

There are no prerequisites for this lab.

Exercises

This hands-on lab includes the following exercises:

1. Exercise 1: Log in to Azure Machine Learning Studio
2. Exercise 2: Get the data
3. Exercise 3: Preprocess the data
4. Exercise 4: Define the features
5. Exercise 5: Select and apply a learning algorithm
6. Exercise 6: Predict new automobile prices

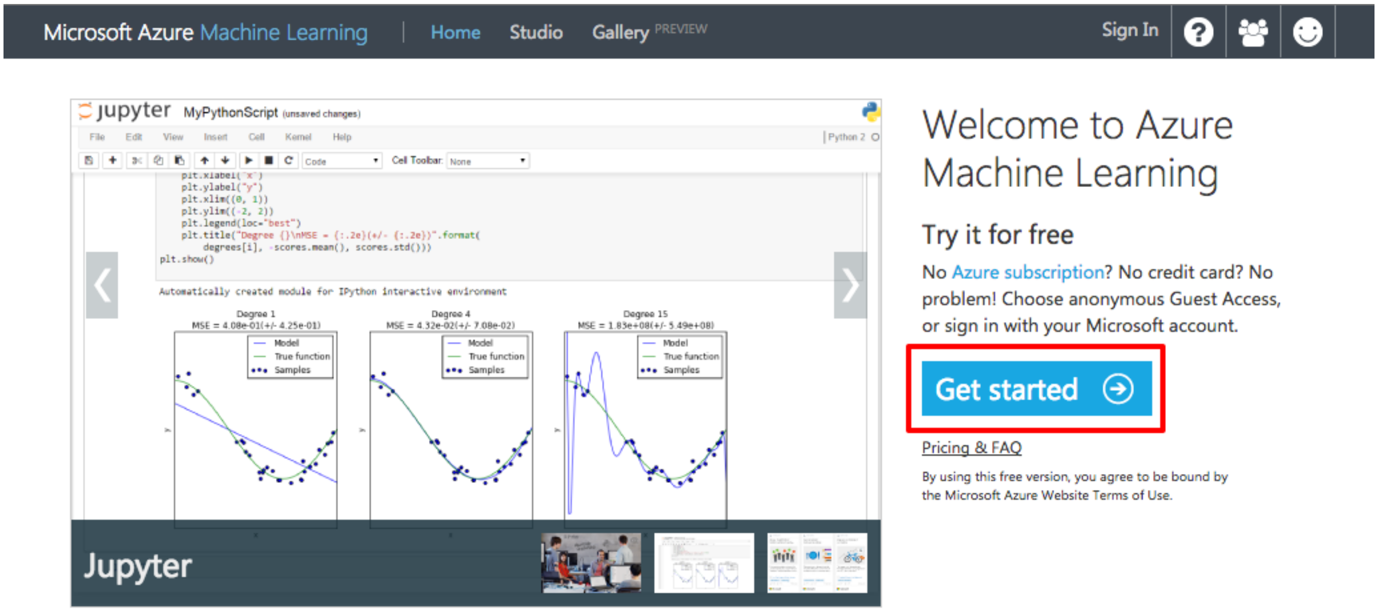
7. Exercise 7: Deploy as a Web service
8. Exercise 8 (Optional): Compare two models

Estimated time to complete this lab: **60** minutes.

Exercise1: Log in to Azure Machine Learning Studio

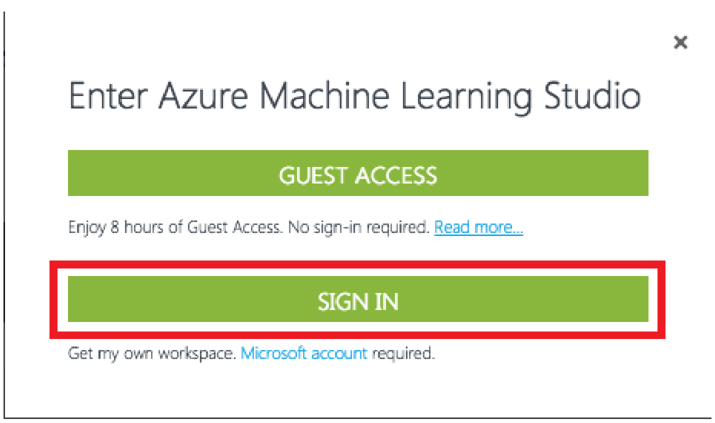
The first step in employing Azure Machine Learning is to log in.

1. In your web browser, navigate to http://studio.azureml.net and click the **Get started** button



Getting started with Azure ML

2. You can either use a guest account or sign in with your Azure subscription. You should do the latter because the guest account is both time-limited (8 hours) and size-limited (max data file size is 100 MB and experiments are limited to 50 or fewer modules), and it doesn't support R or Python. Therefore, click the **SIGN IN** button to sign in with your Microsoft Azure account.



Choosing the login type

3. When you first log into Azure ML, a pop up dialog offers you a chance to take a tour. If you have never used Azure ML, the seven to ten minute tour is quite good and shows you all aspects of the service. For this lab we will walk through the creating your own custom experiment so it is not necessary to do the tour first. Each time you log into Azure ML, you are prompted to take the tour so you can come back to it later. Do note that when you do the full tour, the use your Azure ML instance to create a new predictive experiment. When the demo pauses and shows "running" in the upper right corner, it is doing real ML processing. For this lab, just click the **Not now** button in the dialog to continue the lab.

Now that you're logged in, the next step is to import some data and begin building a model around it.

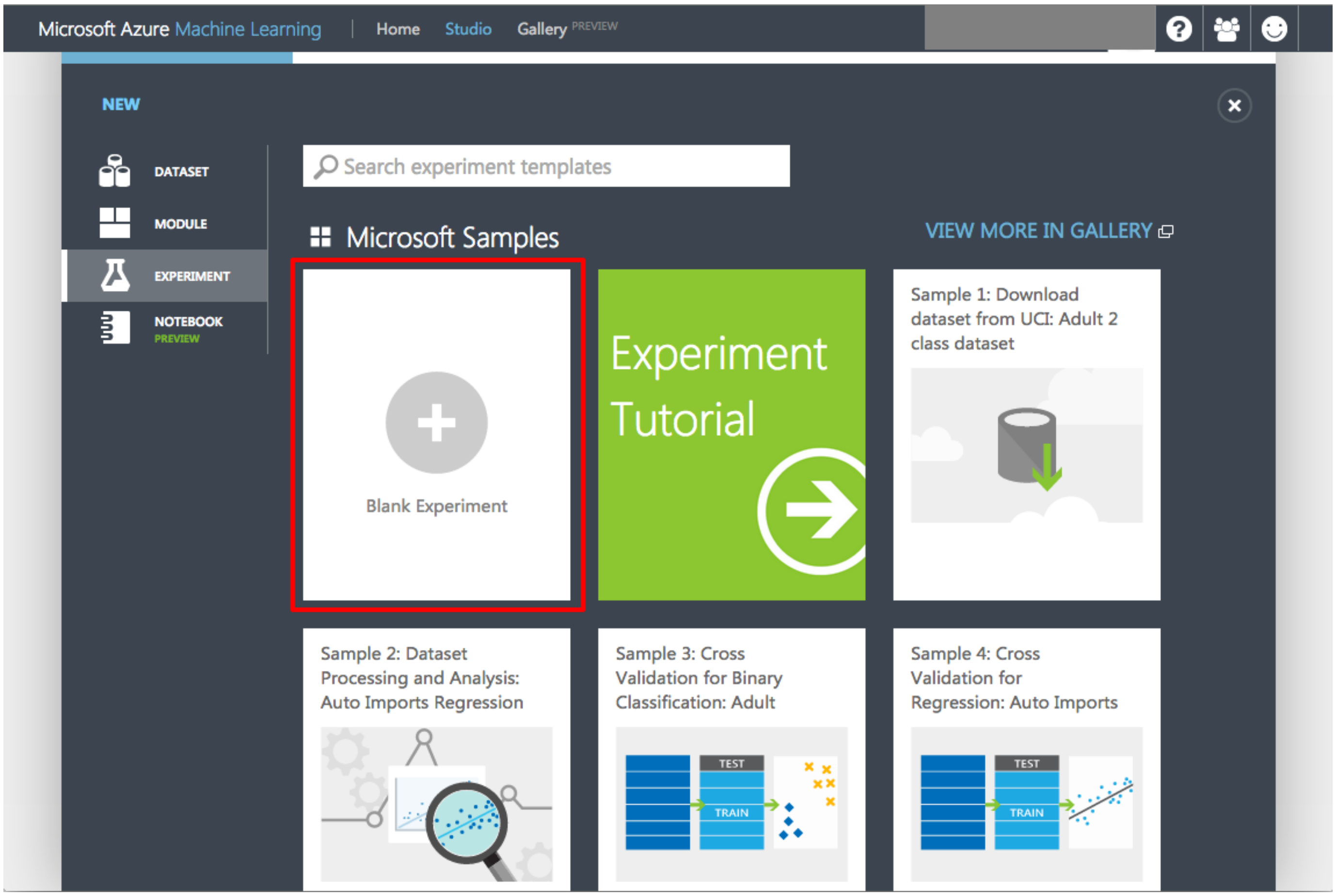
Exercise 2: Get the data

In this exercise, you will create an experiment and add a dataset to it.

> When working with the Azure Machine Learning Studio, get in the habit of saving your experiments often, as in after each step of this lab. That way, if you encounter a problem, you will not have to replicate steps to get caught up. Also, be aware that due to a known issue in ML Studio, you will **lose your work if you click the Back button without saving your experiment first.**

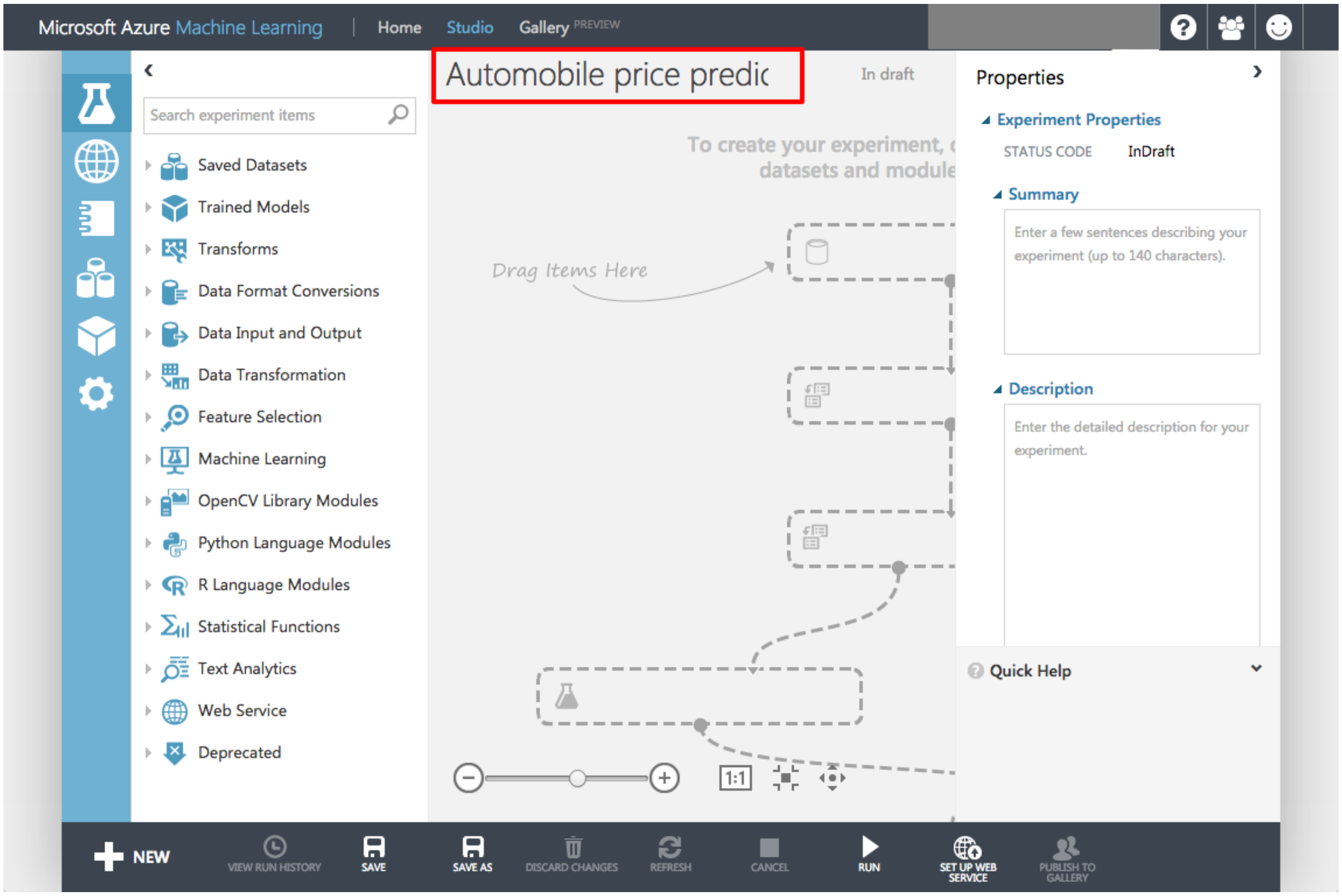
Azure Machine Learning Studio comes with several sample datasets. In this lab, you will utilize the sample dataset named "Automobile price data (Raw)." This dataset includes entries for a number of individual automobiles, including make, model, technical specifications, and price.

1. Start a new experiment by clicking **Blank Experiment**.



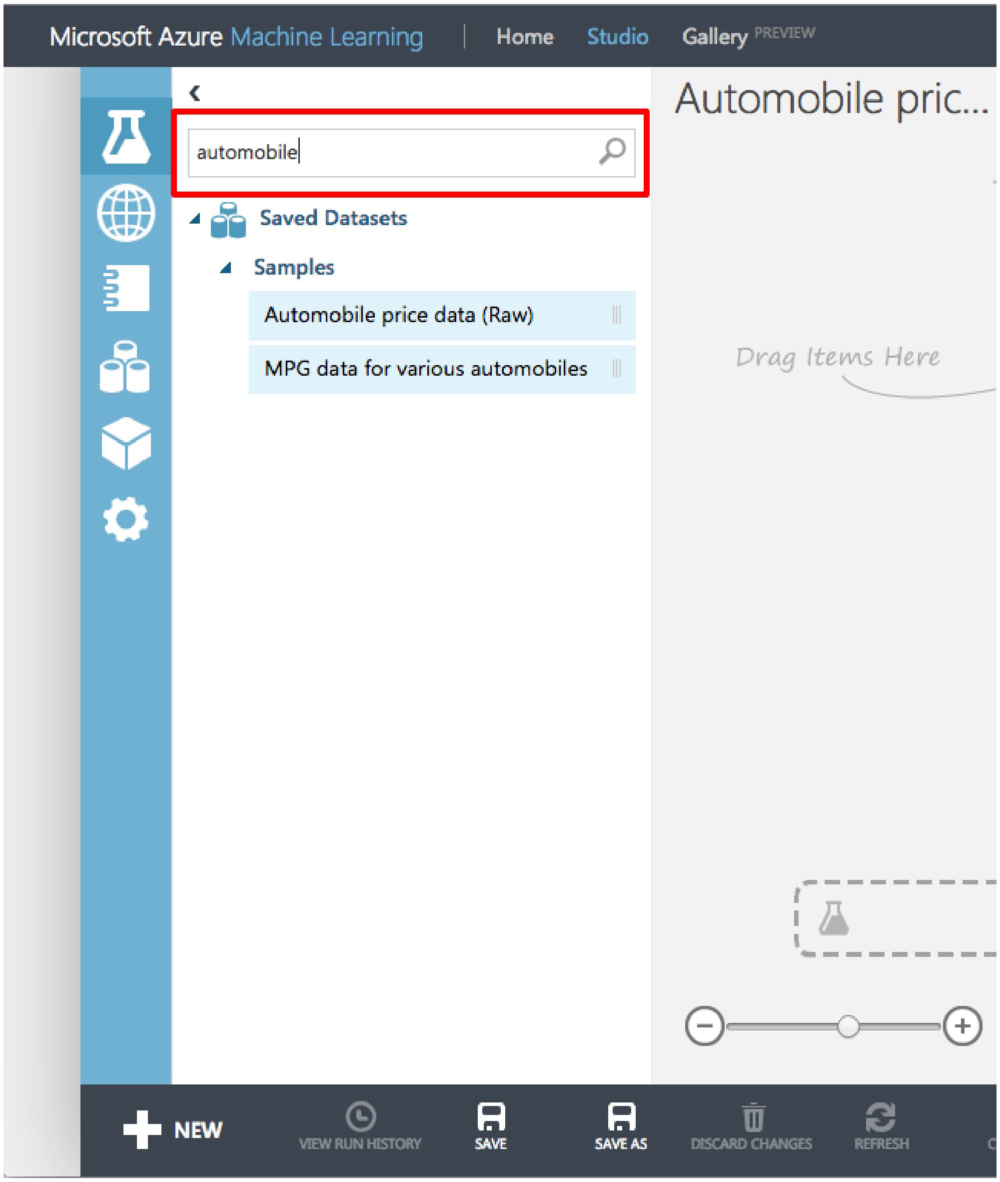
Creating a blank experiment

2. Click the default experiment name at the top of the canvas and change it to something more meaningful, such as "Automobile price prediction."



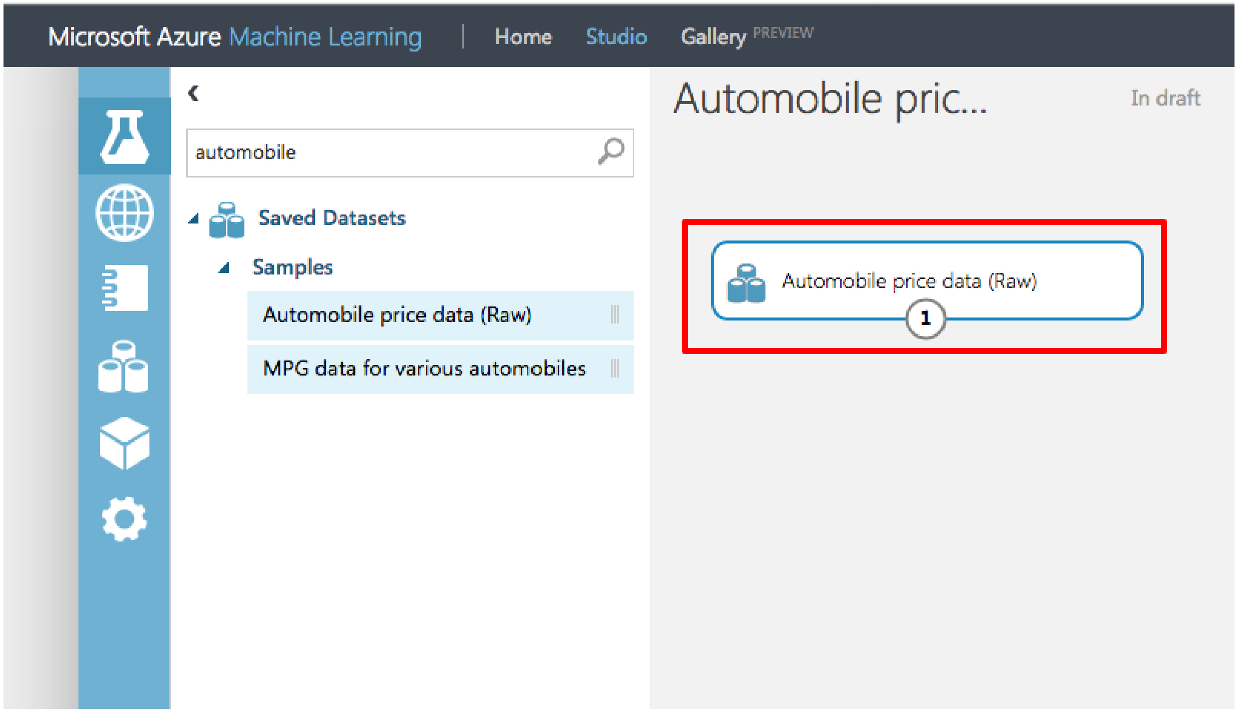
Renaming the experiment

3. To the left of the experiment canvas is a palette of datasets and modules. Type "automobile" in the search box at the top to find the dataset labeled "Automobile price data (Raw)."



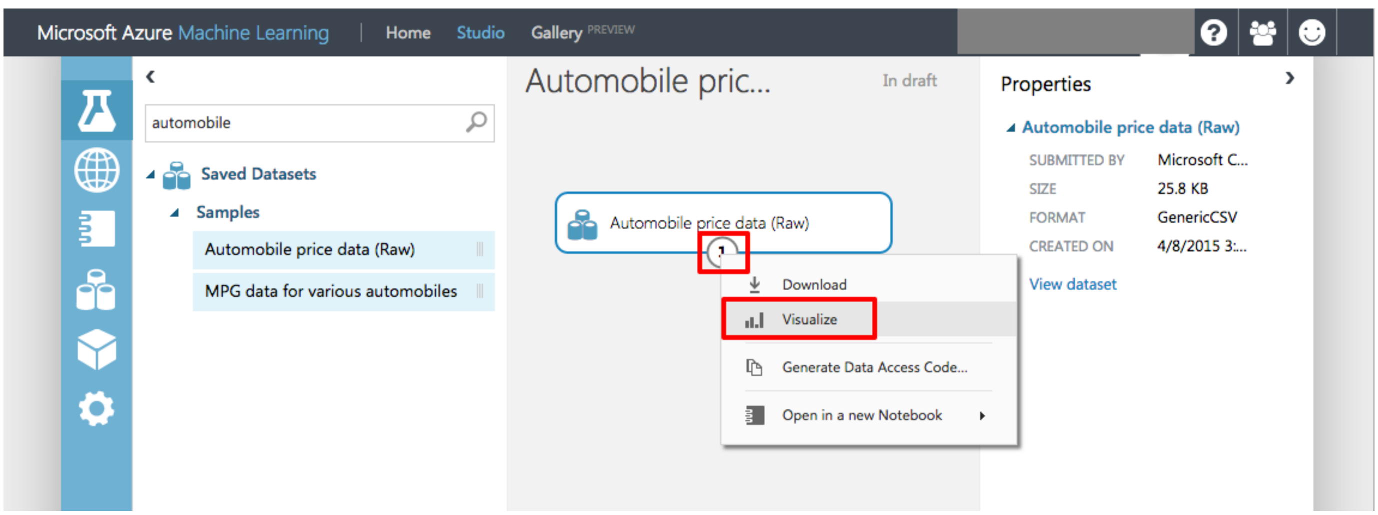
Finding a dataset

4. Drag the "Automobile price data (Raw)" dataset and drop it onto the experiment canvas.



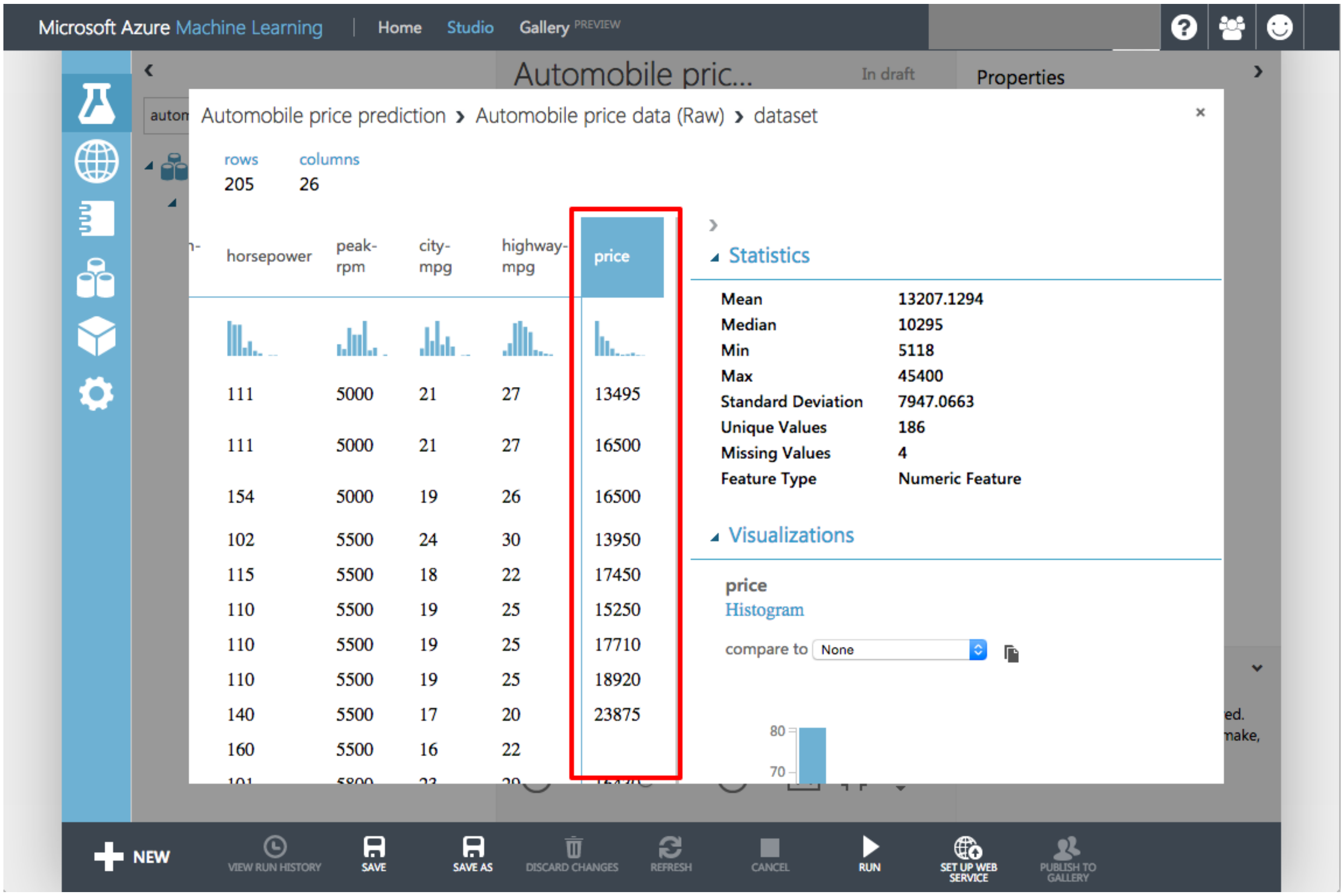
Adding a dataset

5. To see what this dataset looks like, click the output port (the circle with the "1" in it) at the bottom of the dataset and select **Visualize**.



Visualizing the dataset

6. The variables in the dataset appear as columns, and each instance of an automobile appears as a row. The far-right column (column 26, titled "price") is the target variable for your predictive analysis.



Viewing the raw data

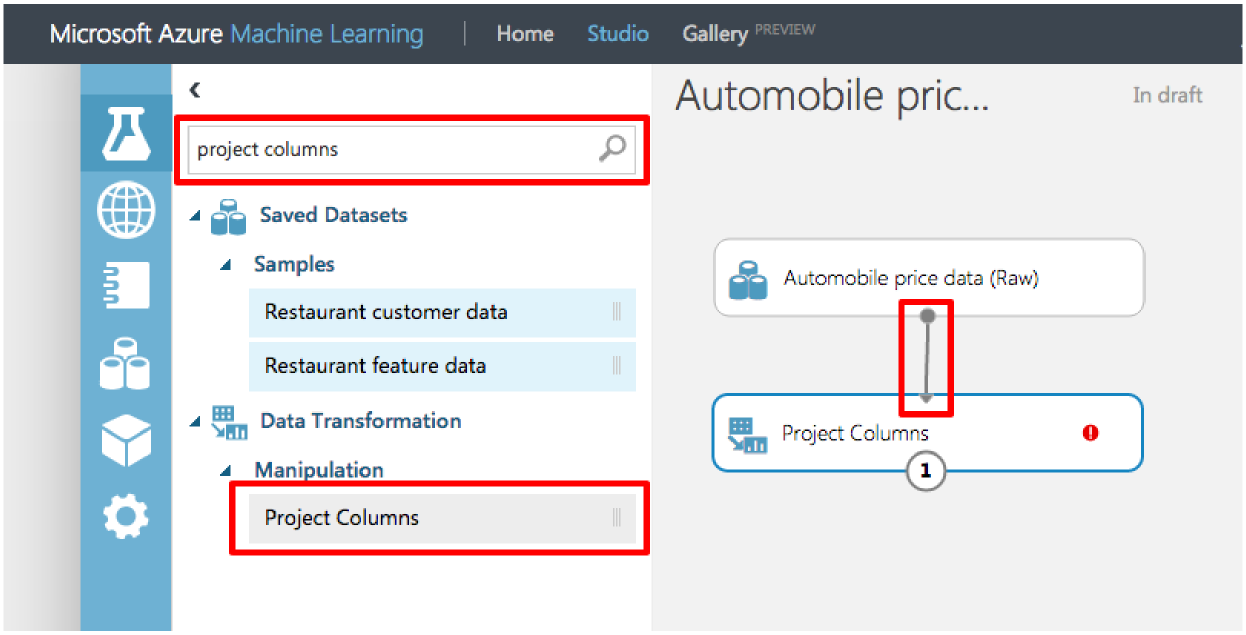
7. Close the data-visualization window by clicking the "**x**" in the upper right corner.

In this exercise, you learned how to create a new ML experiment and add a dataset to it. Next up: preparing the data for use.

Exercise 3: Preprocess the data

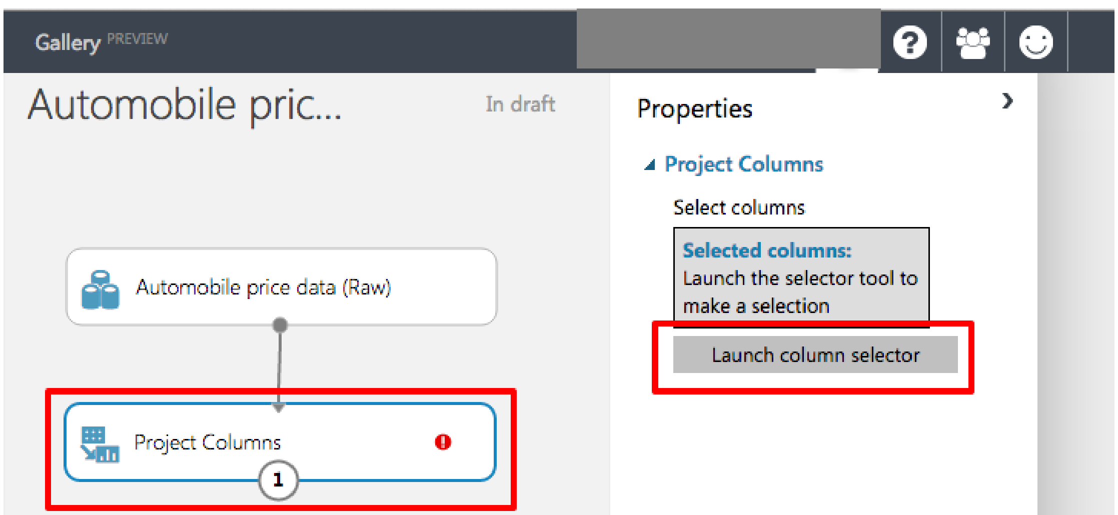
No dataset is perfect; most require some amount of preprocessing before they can be analyzed. When you visualized the data, you may have noticed that some rows of automobile data contained missing values. These missing values need to be cleaned up before analysis begins. In this example, you will simply remove any rows that have missing values. In addition, the "normalized-losses" column has a lot of missing values, so you'll exclude that column from the model.

1. At the top of the modules pallet, type "project columns" into the search box to find the Project Columns module. Drag that box over to the experiment canvas and connect it to the output port of the of the "Automobile price data (Raw)" dataset by dragging an arrow downward from the output port. The Project Columns module allows you to specify which columns of data to include or exclude in the model.



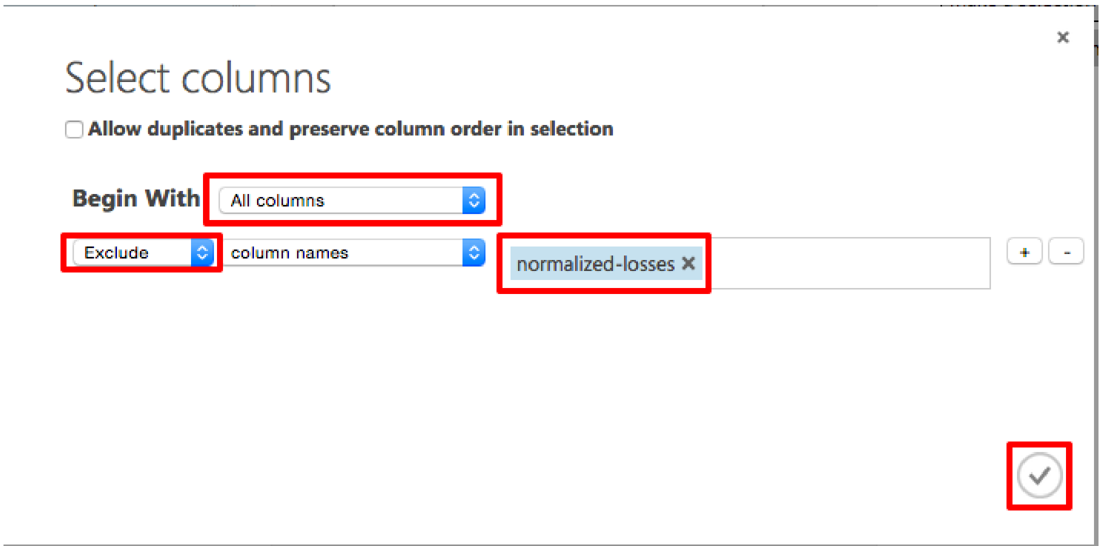
Connecting the dataset output to the Project Columns input

2. Select the Project Columns module on the experiment canvas and click the **Launch column selector** button in the Properties pane on the right.



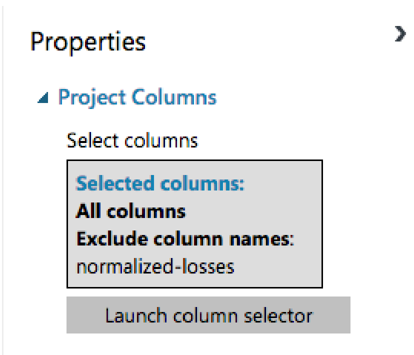
Launch column selector

3. Select **All columns** in the drop-down list labeled "Begin With." This tells the Project Columns modules to pass through all the columns (except those you're about to exclude). In the next row, select **Exclude** and **column names**, and then click inside the text box. A list of columns appears. Select **normalized-losses** to add that column to the text box. Now click the check mark to close the column selector.



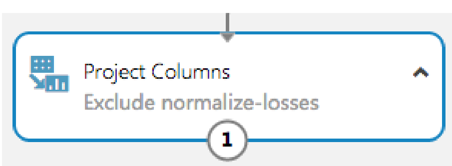
Selecting columns for the model

4. The Properties pane indicates that the Project Columns module will pass through all columns from the dataset except "normalized-losses."



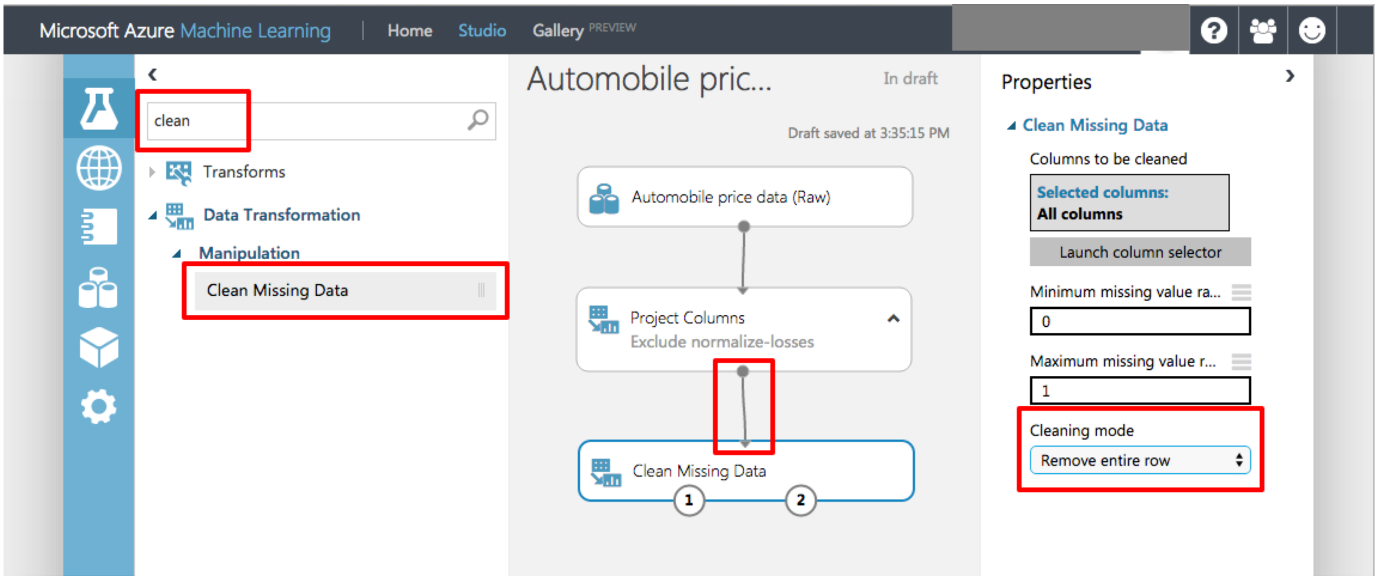
Final column selection

5. You are building a real model in this lab, but it is a relatively simple one. In larger and more complex experiments containing many modules, it's easy to lose track of what each module does and why. A nice feature of Azure Machine Learning Studio is that if you double-click a module on the experiment canvas, you can annotate it with comments. Double-click the Project Columns module and type the comment "Exclude normalized-losses" in the text box that pops up. To display the comment, click the down arrow in the Project Columns box. If you wish to change the comment, you can right-click the module and select **Edit Comment** from the menu that pops up.



Adding and viewing comments

6. Now it's time to remove rows containing blank values. Type "clean" into the search box and drag the Clean Missing Data module to the experiment canvas and connect it to the output of the Project Columns module. In the Properties pane, select **Remove entire row** to clean the data by removing rows that have at least one missing value.



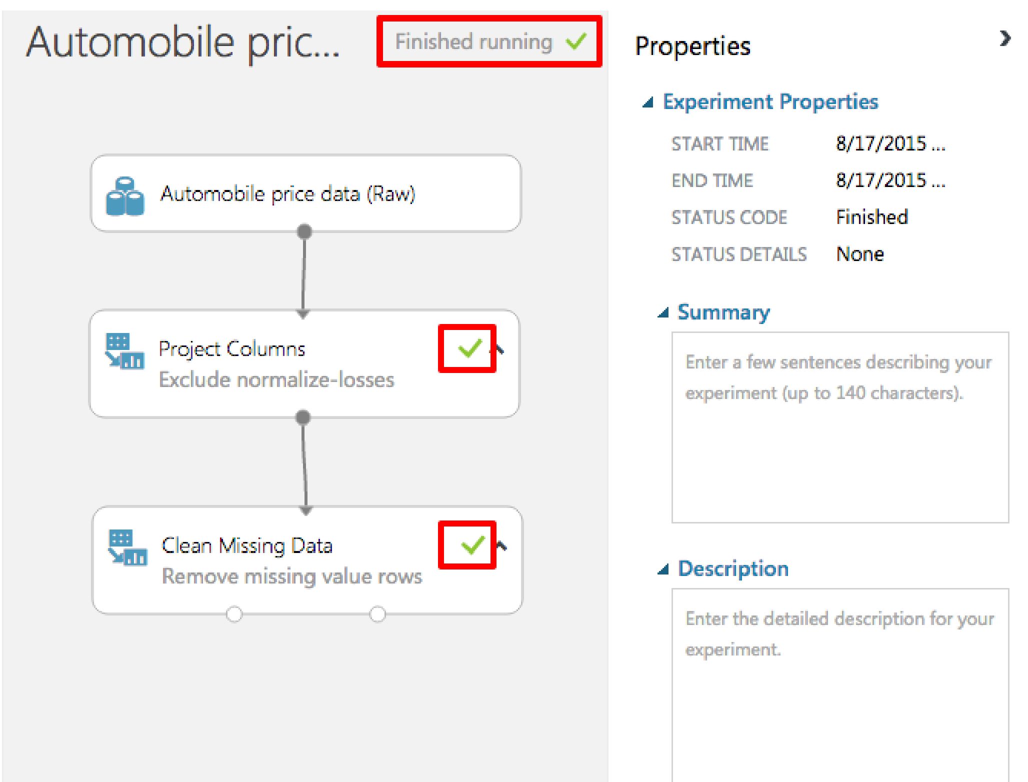
Removing rows with missing values

7. Double-click the Clean Missing Data module and enter the comment "Remove rows with missing values."

8. Click the **SAVE** button at the bottom of the canvas to save a draft of the experiment.

9. Click the **RUN** button at the bottom of the canvas to run the experiment.

10. When the experiment finishes running, both modules will be tagged with a green check mark to indicate that they performed successfully. In addition, the status in the upper-right corner will change to "Finished running".



Successful run

11. The output from the Clean Missing Data module is a dataset with the "normalized-losses" column removed and all rows with at least missing value removed. To view the cleaned dataset, click the left output port of the module ("Cleaned dataset") and select **Visualize**. Notice that the "normalized-losses" column is no longer present, and there are no missing values.

12. Close the data-visualization window by clicking the "**x**" in the upper-right corner.

In this exercise, you learned how to clean input to provide the best possible data to your experiments. The data is ready; now it's time to work on the model itself.

Exercise 4: Define the features

In machine learning, *features* are individually measurable properties of the data that you're analyzing. In the Automobile price data (Raw)" dataset, each row represents one automobile, and each column represents a feature of that automobile. Identifying features for a robust and accurate predictive model frequently requires experimentation and domain knowledge of the problem you're trying to solve. Some features are better for predicting target values than others. For example, it's likely that there is some correlation between engine size and price, because larger engines cost more. But intuition tells us that miles per gallon might not be a strong indicator of price. In addition, some features have a strong correlation with other features (for example, city-mpg versus highway-mpg), and can therefore be excluded since they add little to the model.

It is time to build a model that uses a subset of the features in the dataset. You'll start with the following features (columns), which include the "price" feature that the model will attempt to predict:

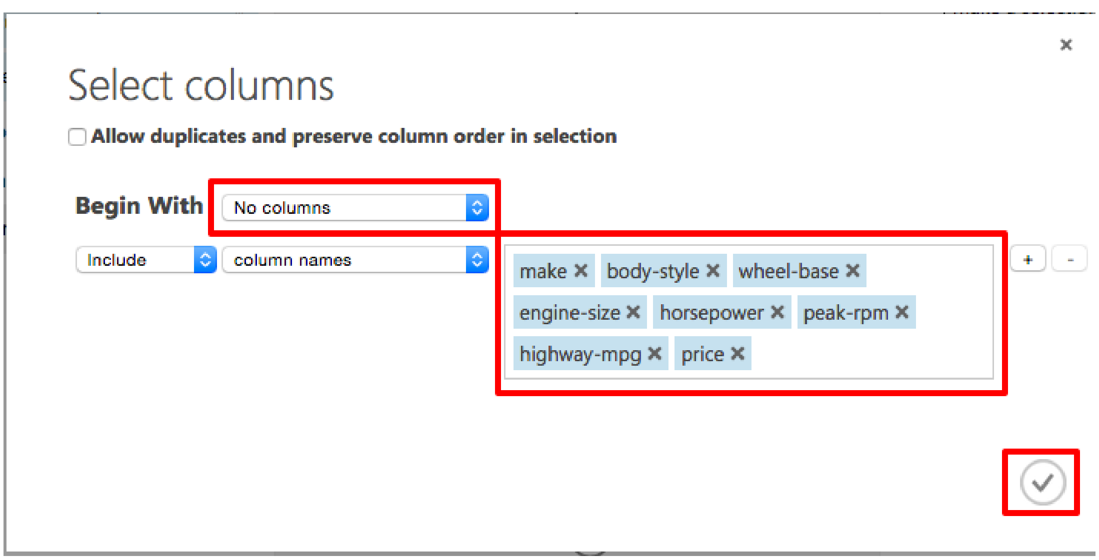
- make
- body-style
- wheel-base
- engine-size
- horsepower
- peak-rpm
- highway-mpg
- price

Later, you can always come back and refine the model by selecting different features.

1. Drag another Project Columns module to the experiment canvas and connect it to the left output port of the Clean Missing Data module. Double-click the new Project Columns module and type "Select features for prediction" as the comment.

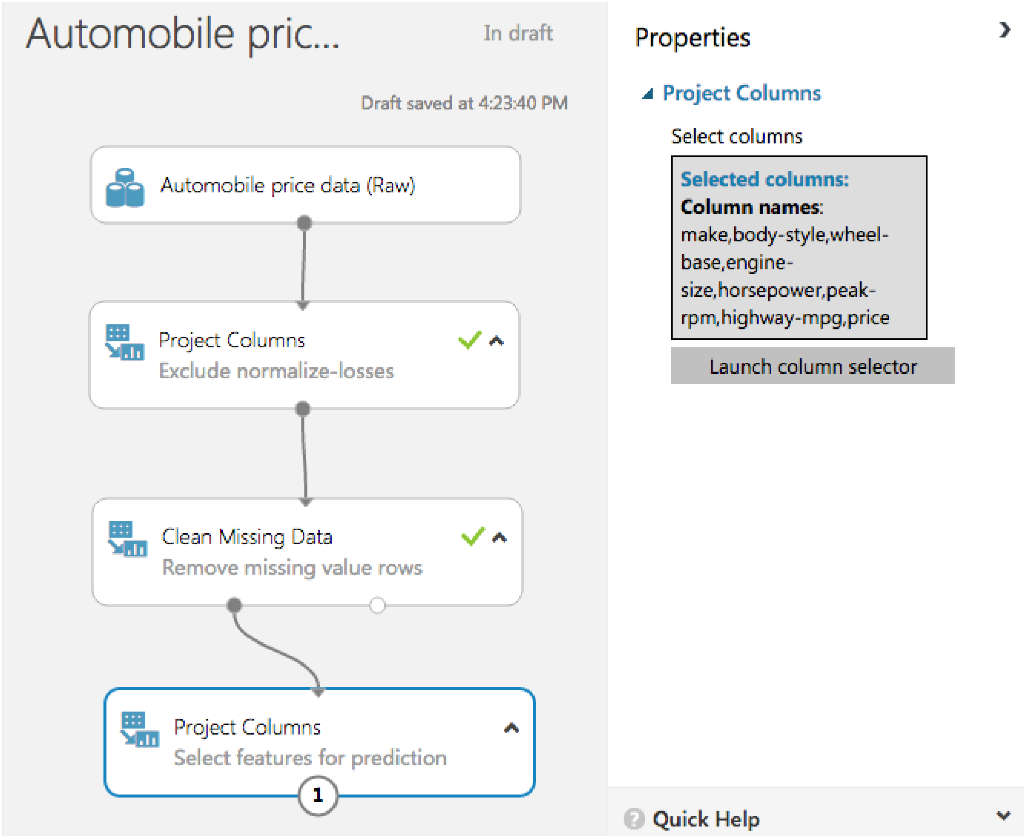
2. Select the module you just added and click **Launch column selector** in the Properties pane.

3. In the column selector, select **No columns**, and then select **Include** and **column names** in the filter row. This directs the module to pass through only the columns that are specified. Then click the box to the right of **column names** and select the columns pictured below.



Selecting columns for the experiment

4. Click the check mark to close the column selector. The Properties pane will show the selected columns.



Columns selected for the experiment

5. Click the **SAVE** button at the bottom of the canvas to save a draft of the experiment.

You're getting close! Now comes perhaps the most important part of the process: selecting a learning algorithm to use in the experiment.

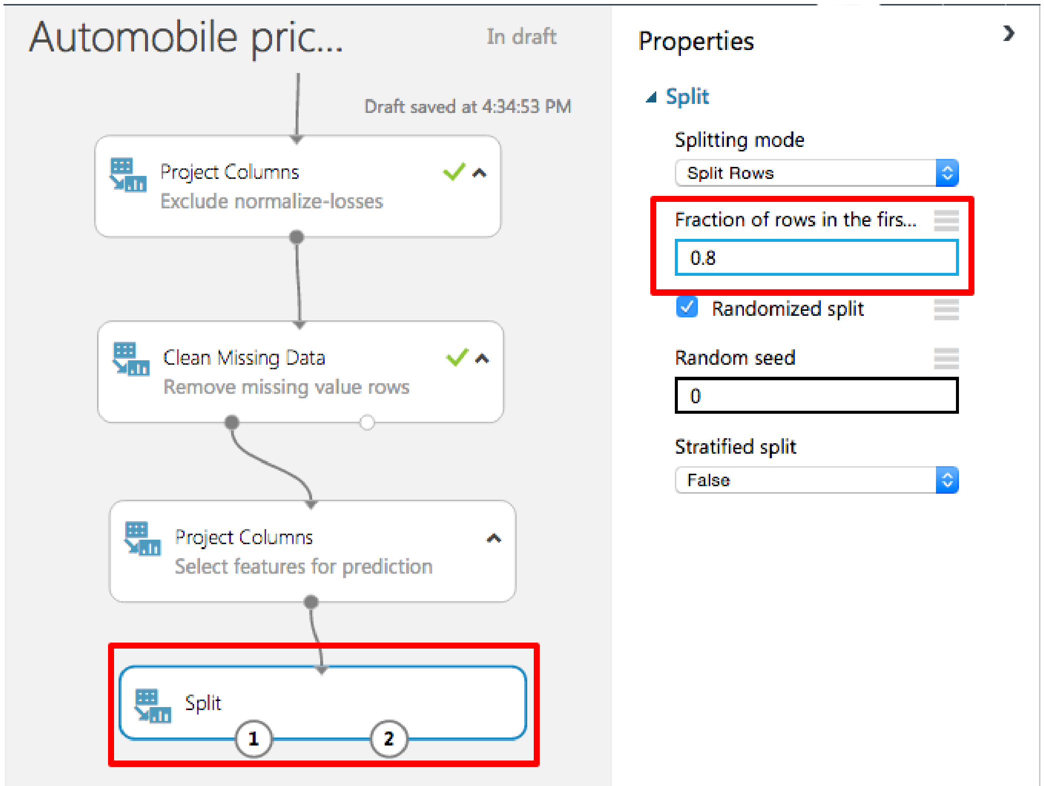
Exercise 5: Select and apply a learning algorithm

Now that the data is ready and the features are selected, constructing a robust predictive model requires training and testing. You will use the part of dataset to train the model, and another part of the dataset to ascertain how adept the model is at predicting automobile prices.

Before you can train and test the model, you must select a learning algorithm for it to use. *Classification* and *regression* are two types of supervised machine-learning algorithms. Classification is used to make a prediction from a defined set of values, such as the make of a car (for example, Honda or BMW). Regression is used to make a prediction from a continuous set of values, such as a person's age or the price of an automobile.

The goal is to predict the price of an automobile, so you will use a regression model. In this exercise, you will train a simple linear-regression model, and in the next exercise, you will test the results.

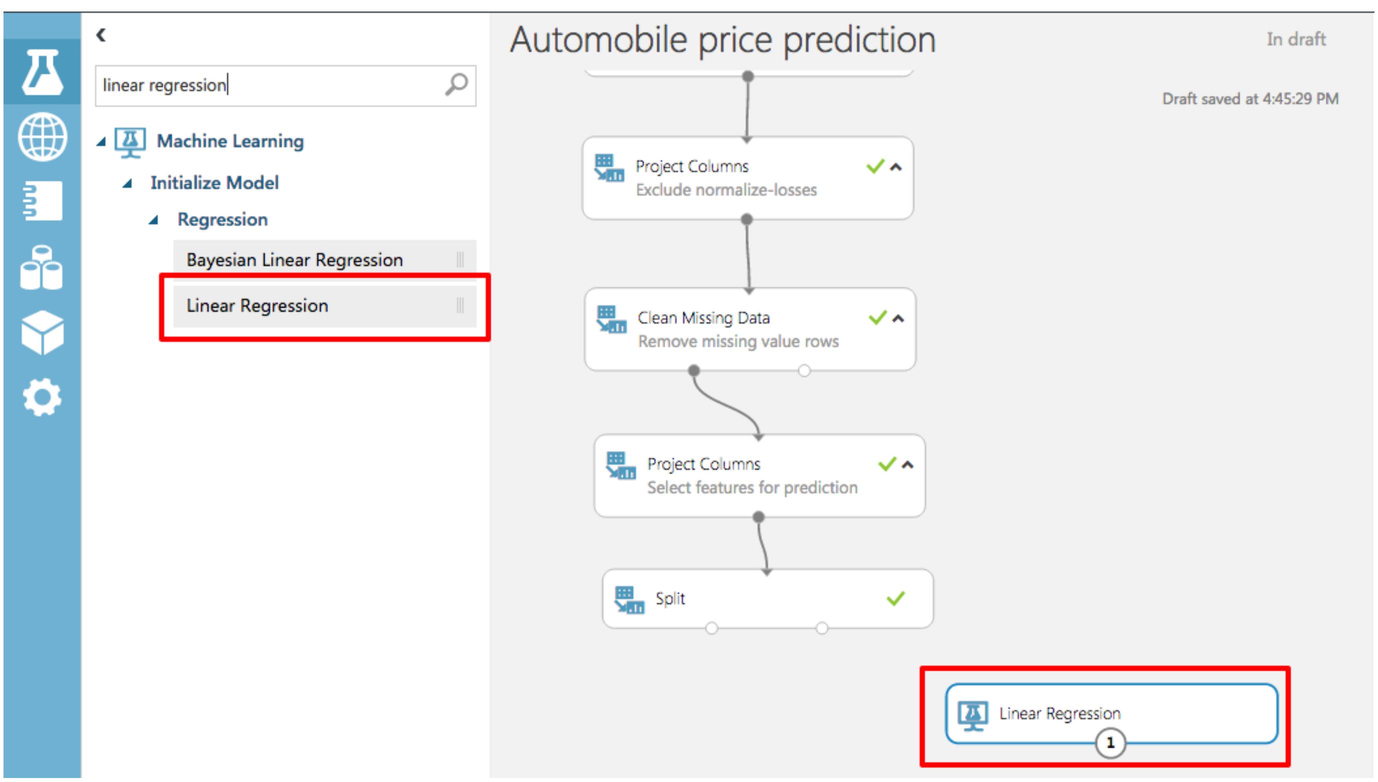
1. You can use one dataset for training and testing by splitting it into separate datasets. Select and drag the Split module to the canvas and connect it to the output of the last Project Columns module. Set "Fraction of rows in the first output dataset" to 0.8. This will use 80% of the data to train the model, and hold back 20% for testing. For this lab leave "Random seed" set to 0. This parameter controls the seeding of the pseudo-random number generator and allows you to produce different random samples by entering different values.



Splitting the data

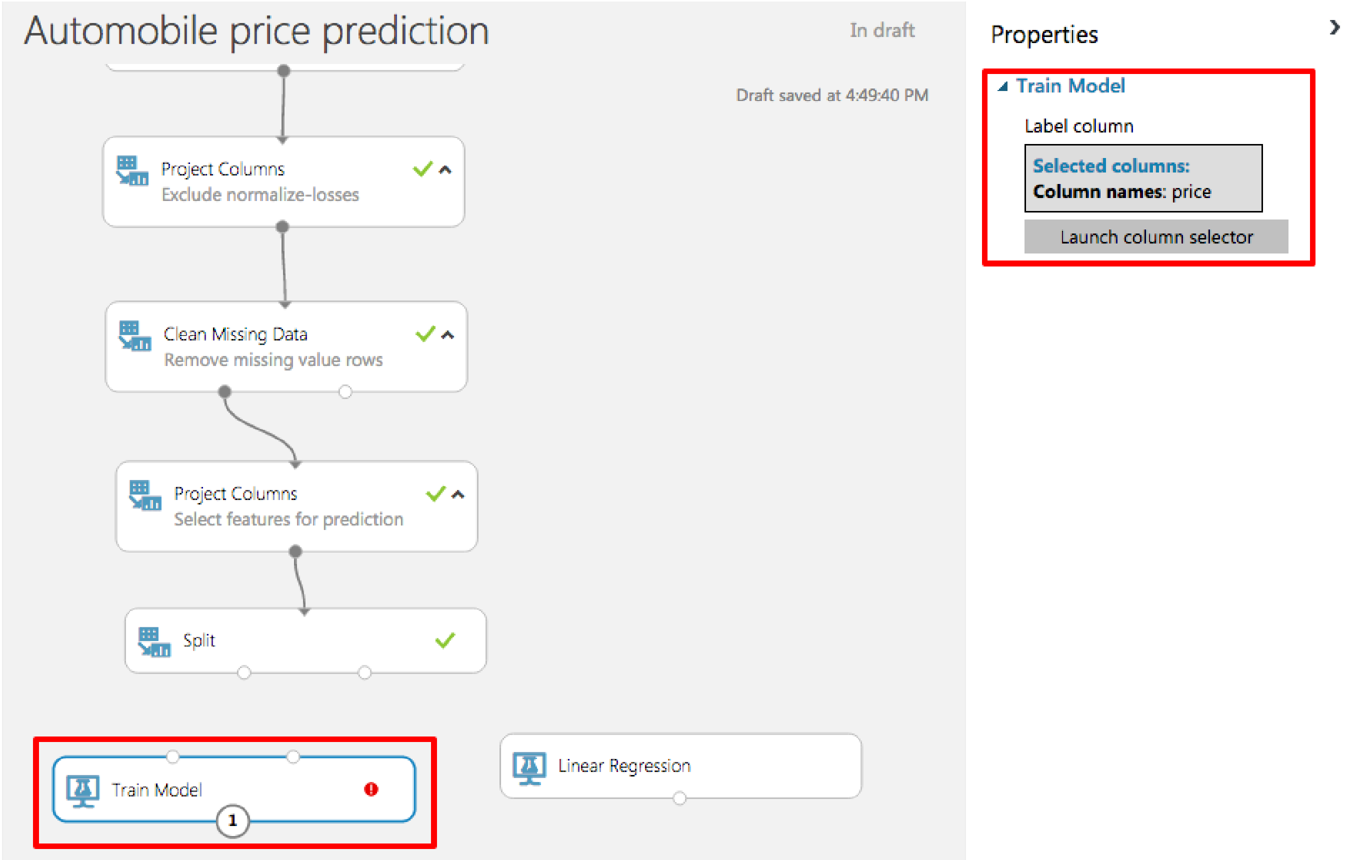
2. Click the **RUN** button to allow the Project Columns and Split modules to pass column definitions to the modules you will be adding next.

3. To specify a learning algorithm, type "linear regression" into the search box in the module palette. Then drag the Linear Regression module onto the canvas.



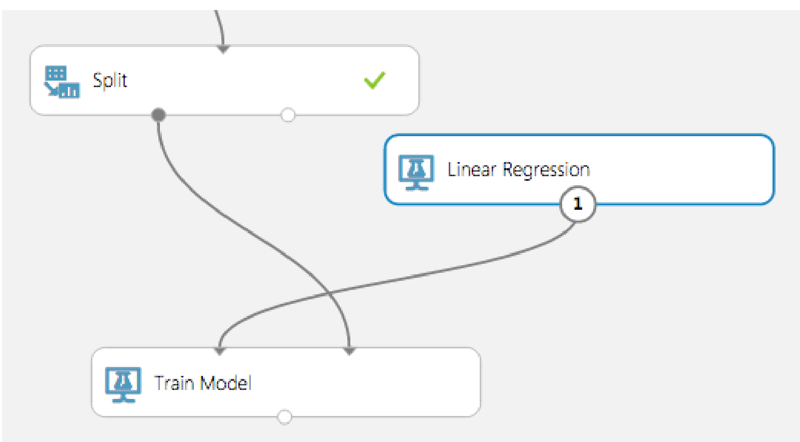
Adding the Linear Regression module

4. Find the Train Model module and add it to the experiment. Then select the module, click **Launch column selector** in the Properties pane, and select the "price" column. This is the value that your model is going to predict. Finish up by clicking the check mark.



Training the model on price

5. Connect the output port of the Linear Regression module to the left input port of the Train Model module. Then connect the left output port of the Split module to the right input port of the Train Model module.



Connecting the modules

6. Click the **SAVE** button to save a draft of the experiment.

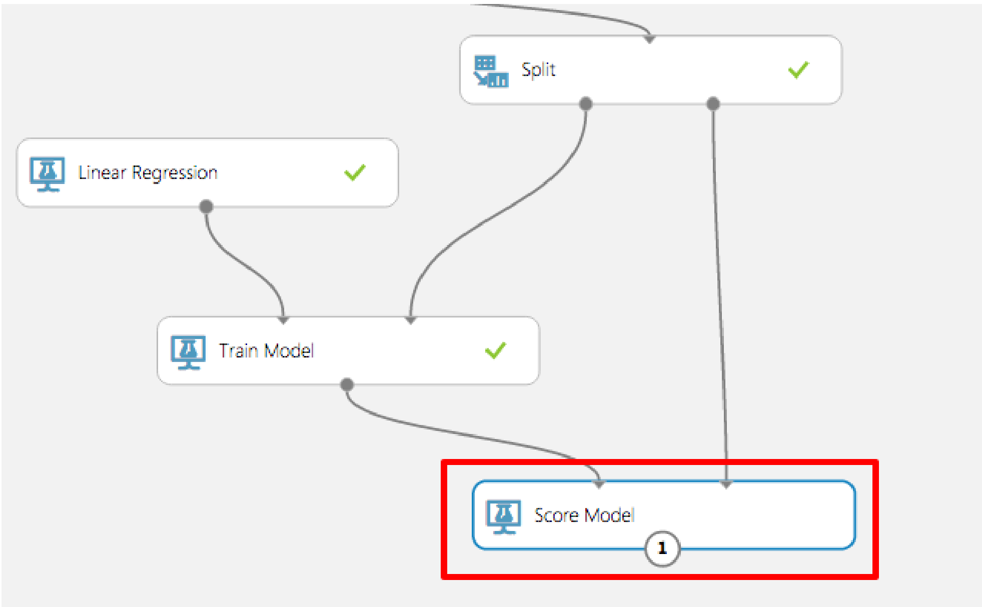
7. Click the **RUN** button to run the experiment. After the run finishes, you will have a trained regression model that can be used to score new samples to make predictions.

Now comes the fun part: testing the model to see how well it's able to predict automobile prices!

Exercise 6: Predict new automobile prices

With a trained model that uses 80% of the data, you can use it to score the other 20% of the data to judge how well the model functions.

1. Find and drag the Score Model module to the canvas and connect the left input port to the output of the Train Model module. Then connect the right input port to the test-data output (right) port of the Split module. That connection represents the 20% of the data that was not used for training.

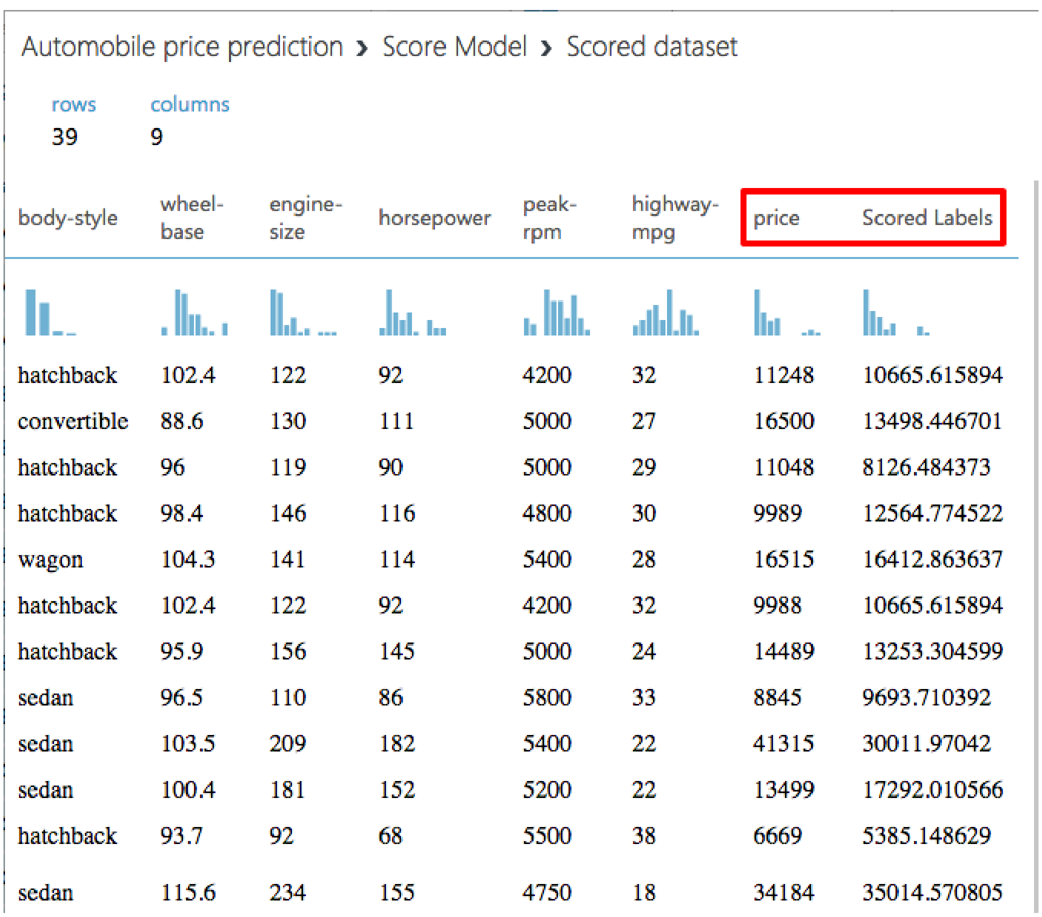


Adding the Score Model module

2. Click the **SAVE** button to save a draft of the experiment.

3. Click the **RUN** button to run the experiment.

4. After the run finishes, click the output port of Score Model and select **Visualize**. The output shows the predicted values for price and the known values from the test data. You may have to scroll the table to the right to see the "price" and "Scored Labeled" columns.



Automobile price prediction › Score Model › Scored dataset

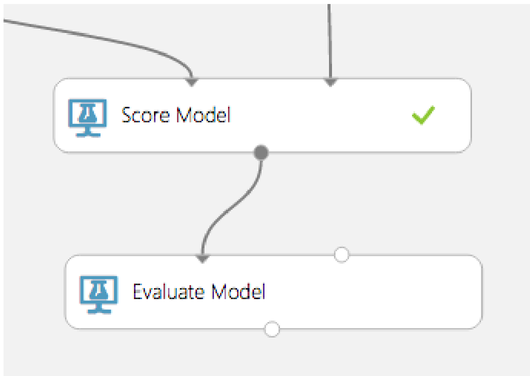
rows 39 columns 9

body-style	wheel-base	engine-size	horsepower	peak-rpm	highway-mpg	price	Scored Labels
hatchback	102.4	122	92	4200	32	11248	10665.615894
convertible	88.6	130	111	5000	27	16500	13498.446701
hatchback	96	119	90	5000	29	11048	8126.484373
hatchback	98.4	146	116	4800	30	9989	12564.774522
wagon	104.3	141	114	5400	28	16515	16412.863637
hatchback	102.4	122	92	4200	32	9988	10665.615894
hatchback	95.9	156	145	5000	24	14489	13253.304599
sedan	96.5	110	86	5800	33	8845	9693.710392
sedan	103.5	209	182	5400	22	41315	30011.97042
sedan	100.4	181	152	5200	22	13499	17292.010566
hatchback	93.7	92	68	5500	38	6669	5385.148629
sedan	115.6	234	155	4750	18	34184	35014.570805

The scored data

5. Close the data-visualization window by clicking the "**x**" in the upper right corner.

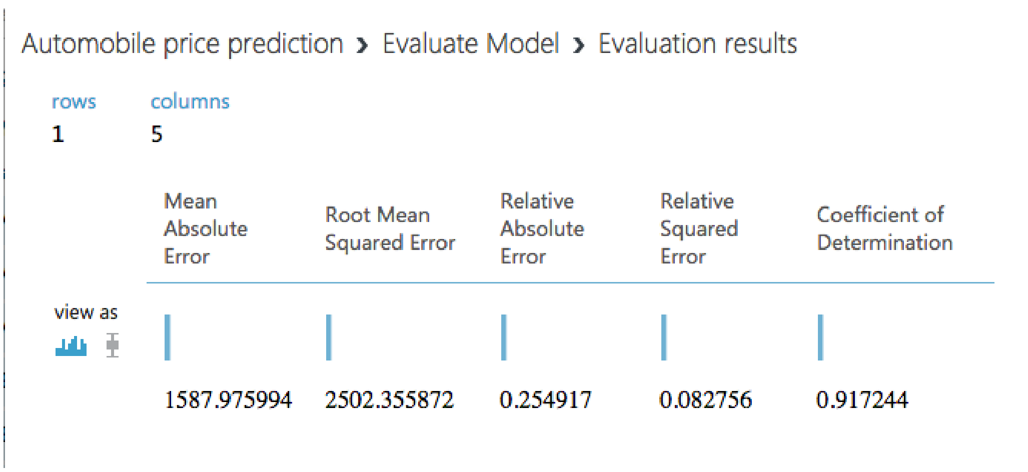
6. To evaluate the quality of the results, select and drag the Evaluate Model module to the experiment canvas, and connect the left input port to the output of the Score Model module. (There are two input ports because the Evaluate Model module can be used to compare two models.)



The Evaluate Model module

7. Run the experiment again by clicking the **RUN** button.

8. Click on the output port of the Evaluate Model and select Visualize from the popup menu.



The evaluation results

Here is a quick explanation of the results:

- **Mean Absolute Error** (MAE): The average of absolute errors (an error is the difference between the predicted value and the actual value).
- **Root Mean Squared Error** (RMSE): The square root of the average of squared errors of predictions made on the test dataset.
- **Relative Absolute Error**: The average of absolute errors relative to the absolute difference between actual values and the average of all actual values.
- **Relative Squared Error**: The average of squared errors relative to the squared difference between the actual values and the average of all actual values.
- **Coefficient of Determination**: Also known as the **R squared value**, this is a statistical metric indicating how well a model fits the data.

For each of the error statistics, smaller is better. A smaller value indicates that the predictions more closely match the actual values. For **Coefficient of Determination**, the closer its value is to 1.0, the better the predictions. In this case, the model was able to predict the price of a car from the test data with slightly more than 90% accuracy.

9. Close the data-visualization window by clicking the "**x**" in the upper right corner.

Now that the model is adequately refined (90% is indicative of a reasonably strong correlation between the input data and results), you might want to be able write programs that utilize the model. That is the subject of the next exercise.

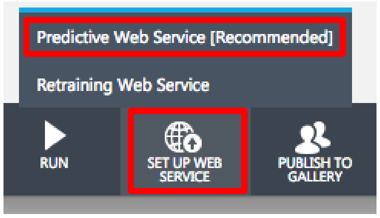
Exercise 7: Deploy as a Web service

Once you have a trained model, you can deploy it as a Web service in order to interact with it programmatically. That way, you can use your favorite programming language to pump data in and get results back.

Before deploying as a Web service, you need to streamline your experiment for scoring. This involves creating a *scoring experiment* from your trained model, removing unnecessary modules that were needed for training but are not needed for scoring, and adding Web-service input and output modules. Fortunately, Azure Machine Learning Studio can do this for you.

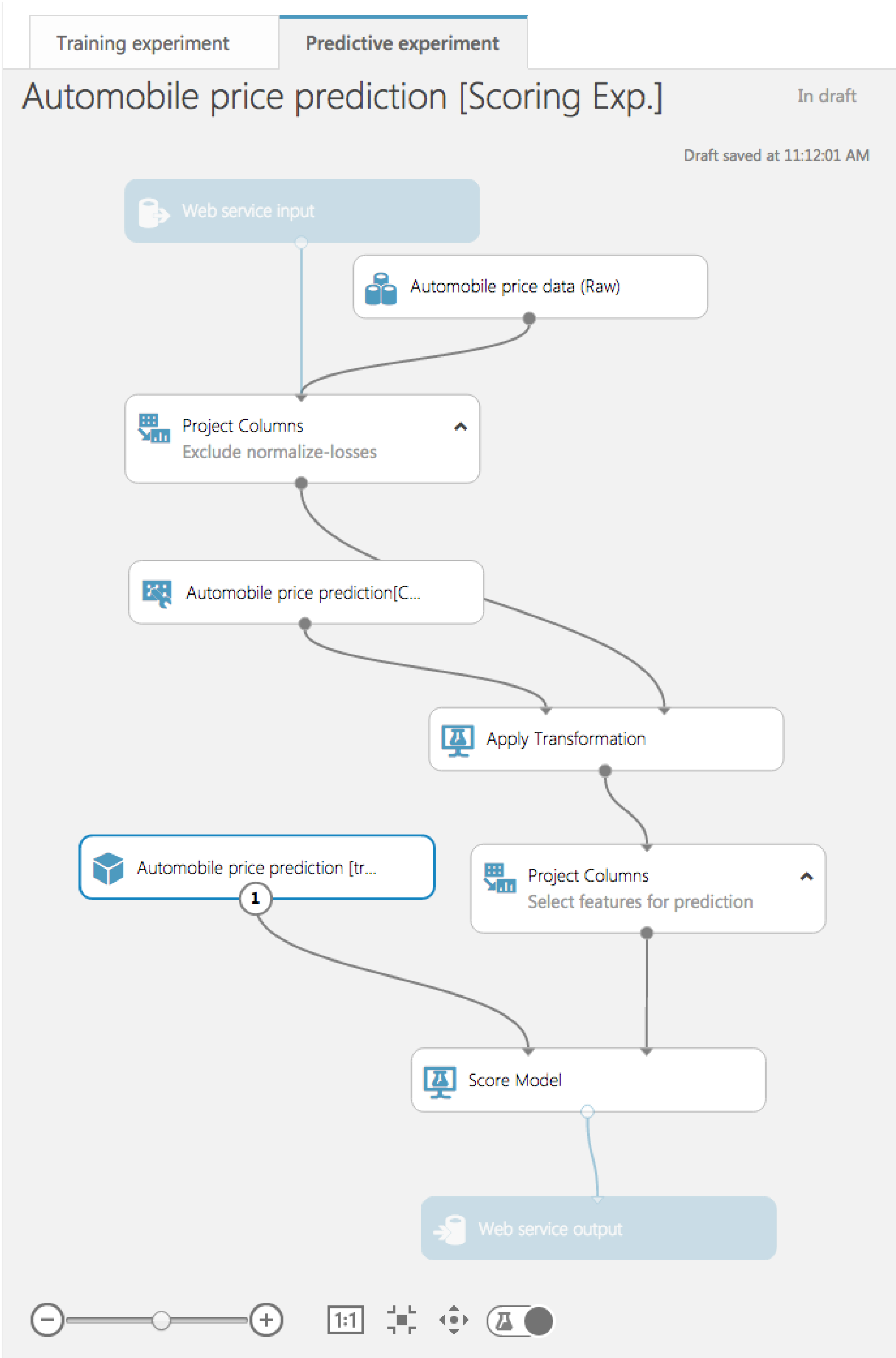
1. Even though you have already run your model, there's a bug in ML Studio whereby visualizing the data sometimes disables one of the Web-service menu items. To work around this, click the **RUN** button to rerun your model.

2. At the bottom of the screen, click the **SET UP WEB SERVICE** button and in the ensuing menu, select **Predictive Web Service [Recommended]**. If this option is grayed out, click the **RUN** button and try again.

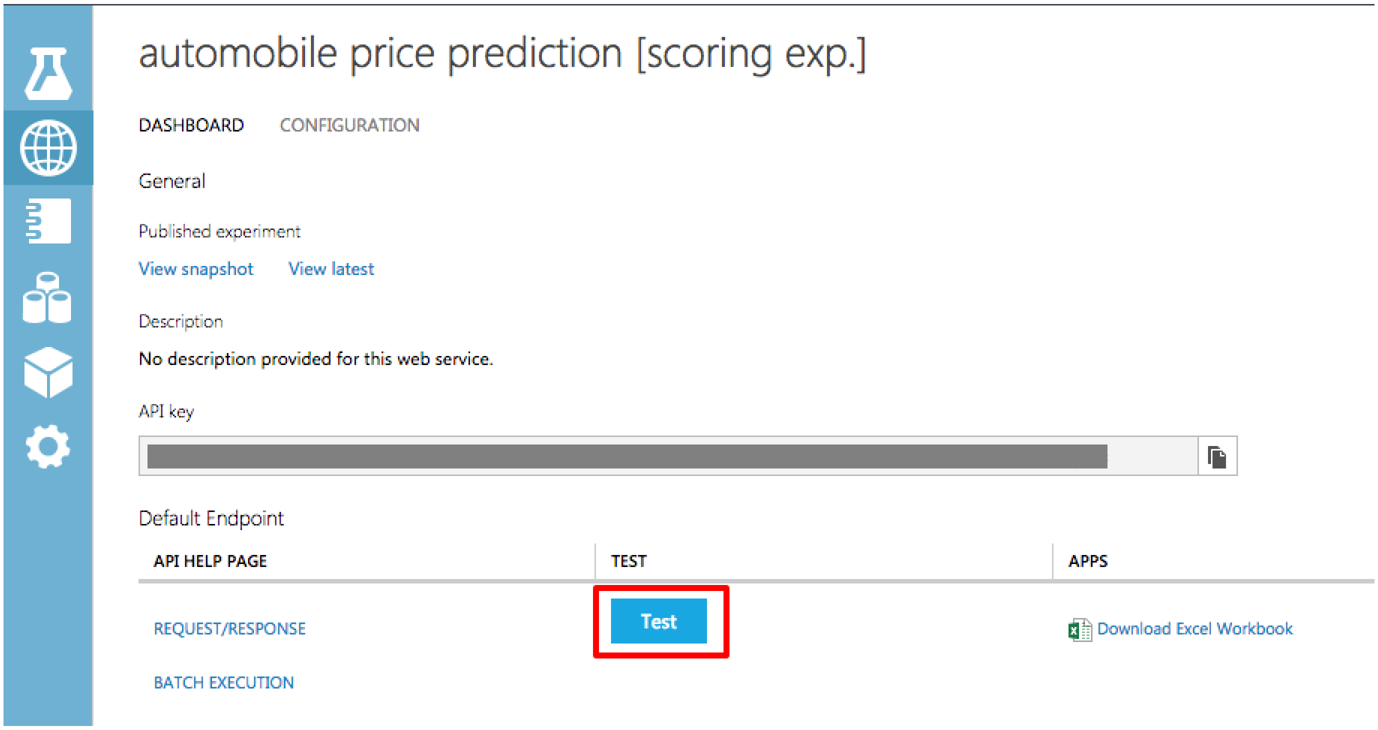
   Creating a predictive Web service

3. The predictive experiment setup will spin for a few seconds, after which you will see a set of tabs at the top of the canvas indicating which model is your training experiment and which is the predictive experiment. When you look at the predictive experiment, you will find that the Train Model module has been replaced by a module named "Automobile price prediction [trained model]," and that new modules were added at the top and bottom for Web-service input and output.



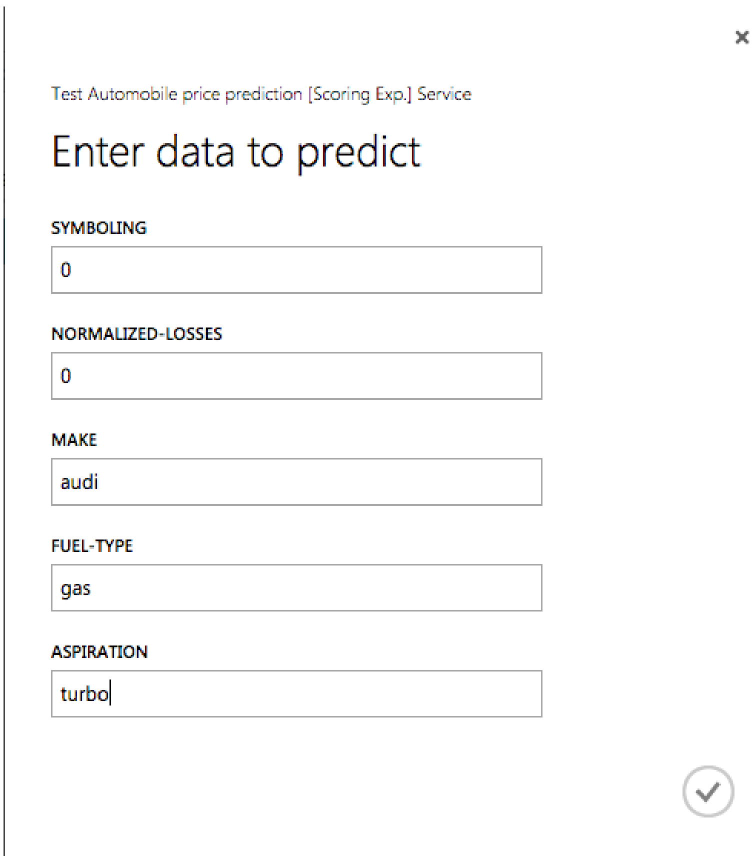
The predictive experiment

4. To create a Web service to use for your predictions, click **RUN** once more. After the run completes, click the **DEPLOY WEB SERVICE** button to deploy your price-predicting Web service. This will take you to the dashboard for the new Web service. Testing your Web service is easy: simply click the **Test** button in the middle of the screen. In the "APPS" column, you can download an Excel spreadsheet for working with your Web service. At the current time, that spreadsheet only runs on Windows-based operating systems.



Testing your Web service

5. Click the **Test** button to display a window for entering test parameters.



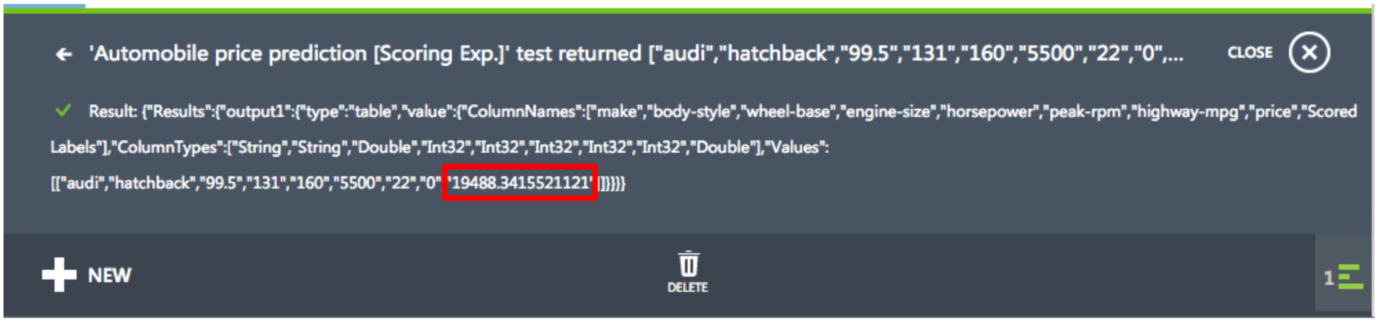
Entering data

6. Enter the following parameter values:

Field	Value

symboling	0
normalized-losses	0
make	audi
fuel-type	gas
aspiration	turbo
num-of-doors	two
body-style	hatchback
drive-wheels	4wd
engine-location	front
wheel-base	99.5
length	178.2
width	67.9
height	52
curb-weight	3053
engine-type	ohc
num-of-cylinders	five
engine-size	131
fuel-system	mpfi
bore	3.13
stroke	3.4
compression-ratio	7
horsepower	160
peak-rpm	5500
city-mpg	16
highway-mpg	22
price	0

7. Click the check mark to pass the data to your Web service.
8. After a short pause, a report will appear at the bottom of the screen. Click the **DETAILS** button to see the full results. The final number in the "Scored Labels" column is the projected price.

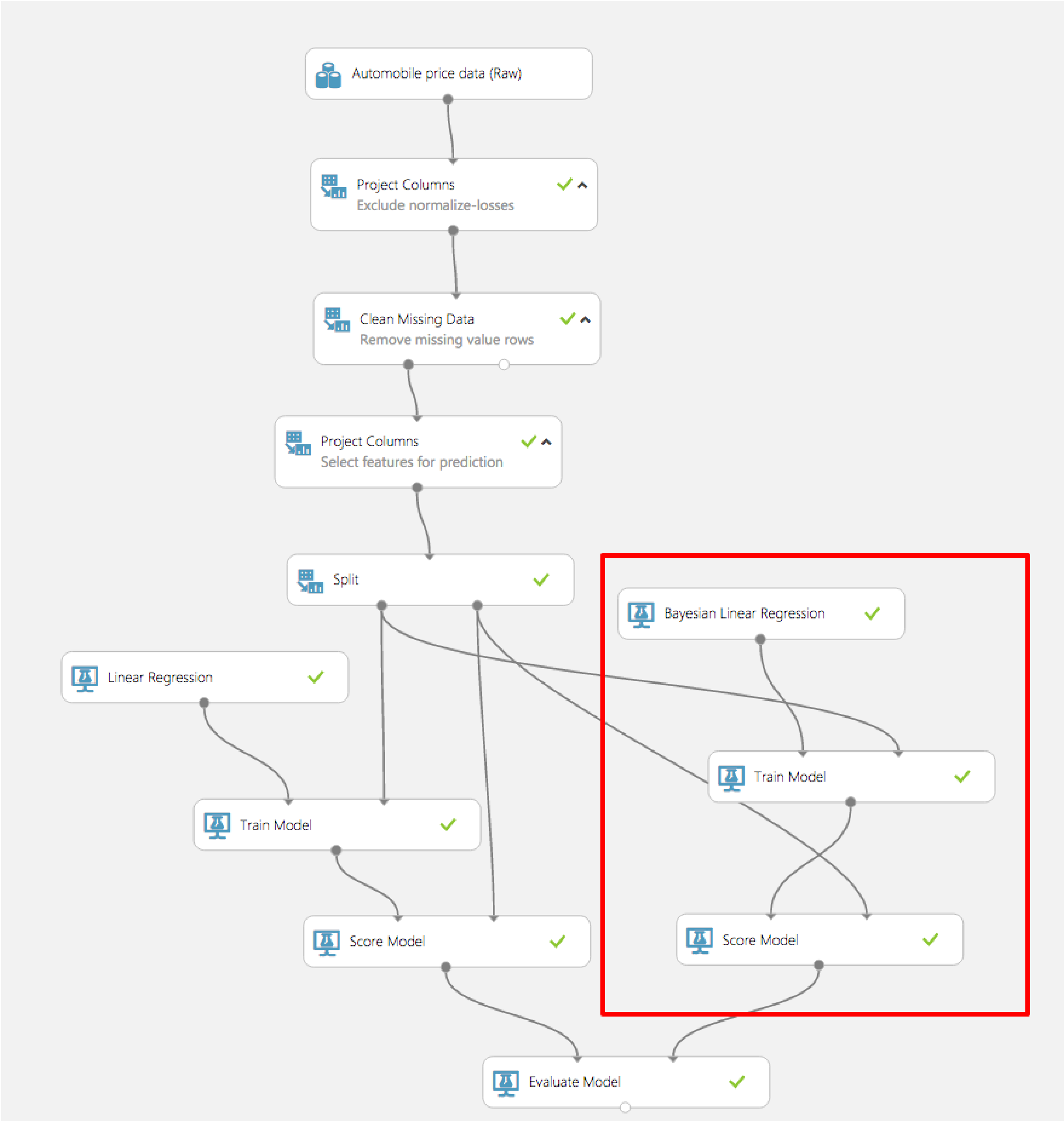


Projected price

Exercise 8 (Optional): Compare two models

When you build a predictive model, it is often useful to try different algorithms and compare the results to see which algorithm delivers the best results. The Evaluate Model module is very effective in comparing the metrics between two different algorithms.

1. If time permits and you're so inclined, compare the results of the linear-regression model to a different model. Which model performs better?



Comparing two models

Summary

In this hands-on lab, you learned how to:

- Log in to the Azure Machine Learning Studio
- Import a sample dataset and prepare it for analysis
- Define the features of a model and select an algorithm
- Train and test the model
- Deploy the model as a Web service

There's much more than you can do with Azure Machine Learning, but this is a start. Feel free to experiment with it on your own and explore the exciting world of predictive analysis with a tool that is not only productive, but fun!

Copyright 2015 Microsoft Corporation. All rights reserved. Except where otherwise noted, these materials are licensed under the terms of the Apache License, Version 2.0. You may use it according to the license as is most appropriate for your project on a case-by-case basis. The terms of this license can be found in http://www.apache.org/licenses/LICENSE-2.0.