

Quickstart: Create your first data science experiment in Azure Machine Learning Studio (classic)

02/06/2019 • 11 minutes to read •  +7

In this article

[Get the data](#)

[Prepare the data](#)

[Define features](#)

[Choose and apply an algorithm](#)

[Predict new automobile prices](#)

[Clean up resources](#)

[Next steps](#)

Tip

Customers currently using or evaluating Machine Learning Studio (classic) are encouraged to try [Azure Machine Learning designer](#) (preview), which provides drag-n-drop ML modules **plus** scalability, version control, and enterprise security.

In this quickstart, you create a machine learning experiment in [Azure Machine Learning Studio \(classic\)](#) that predicts the price of a car based on different variables such as make and technical specifications.

If you're brand new to machine learning, the video series [Data Science for Beginners](#) is a great introduction to machine learning using everyday language and concepts.

This quickstart follows the default workflow for an experiment:

1. Create a model

- [Get the data](#)
- [Prepare the data](#)
- [Define features](#)

2. Train the model

- [Choose and apply an algorithm](#)

3. Score and test the model

- [Predict new automobile prices](#)

Get the data

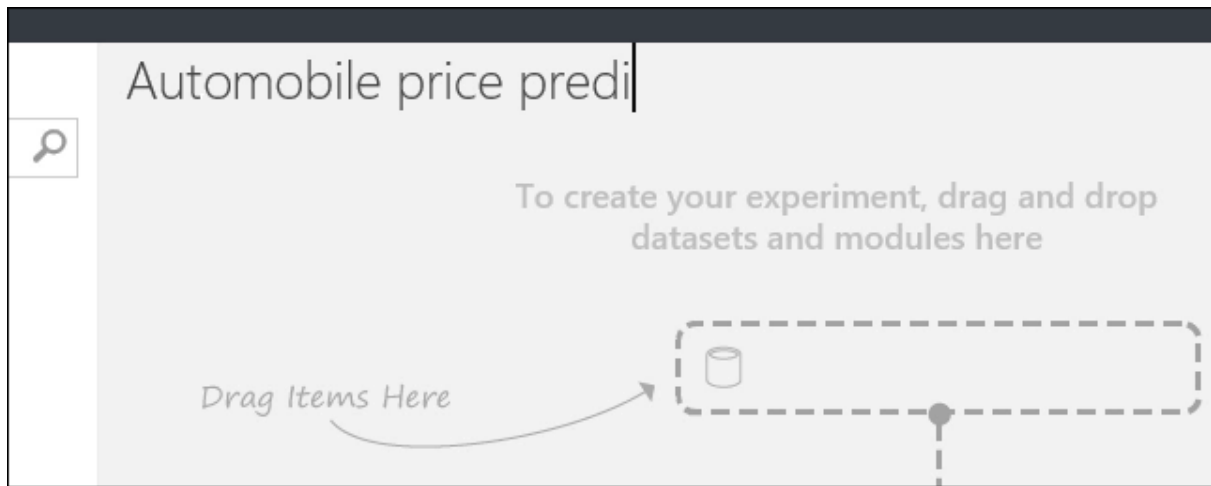
The first thing you need in machine learning is data. There are several sample datasets included with the classic version of Studio that you can use, or you can import data from many sources. For this example, we'll use the sample dataset, **Automobile price data (Raw)**, that's included in your workspace. This dataset includes entries for various individual automobiles, including information such as make, model, technical specifications, and price.

Tip

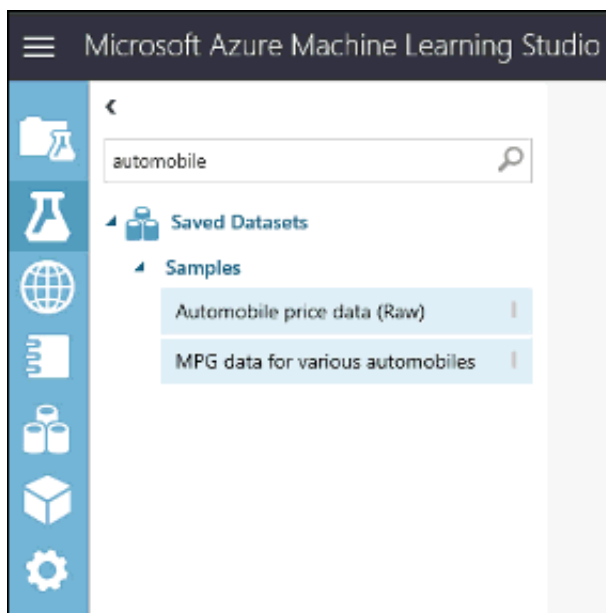
You can find a working copy of the following experiment in the [Azure AI Gallery](#). Go to [Your first data science experiment - Automobile price prediction](#) and click **Open in Studio** to download a copy of the experiment into your Machine Learning Studio (classic) workspace.

Here's how to get the dataset into your experiment.

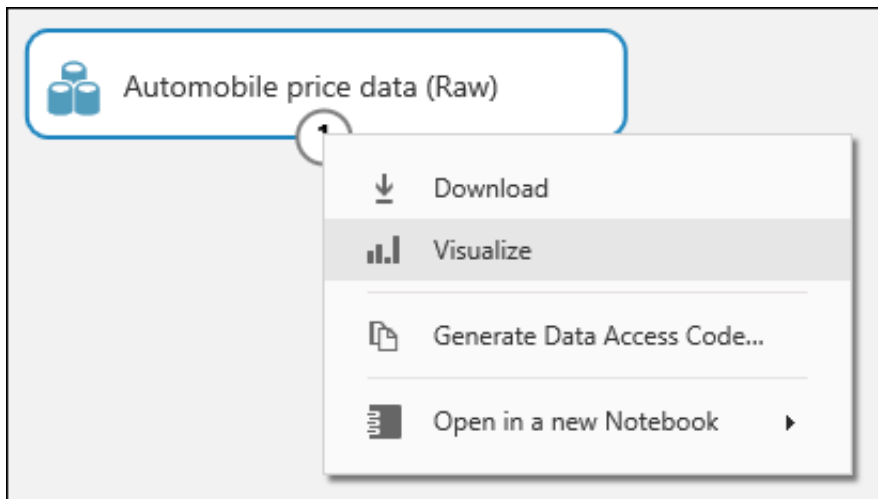
1. Create a new experiment by clicking **+NEW** at the bottom of the Machine Learning Studio (classic) window. Select **EXPERIMENT > Blank Experiment**.
2. The experiment is given a default name that you can see at the top of the canvas. Select this text and rename it to something meaningful, for example, **Automobile price prediction**. The name doesn't need to be unique.



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



To see what this data looks like, click the output port at the bottom of the automobile dataset then select **Visualize**.



Tip

Datasets and modules have input and output ports represented by small circles - input ports at the top, output ports at the bottom. To create a flow of data through your experiment, you'll connect an output port of one module to an input port of another. At any time, you can click the output port of a dataset or module to see what the data looks like at that point in the data flow.

In this dataset, each row represents an automobile, and the variables associated with each automobile appear as columns. We'll predict the price in far-right column (column 26, titled "price") using the variables for a specific automobile.

Automobile price prediction > Automobile price data (Raw) > dataset

rows: 205, columns: 26

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	engine	peak-rpm	city-mpg	highway-mpg	price
view as												
3			alfa-romero	gas	std	two	convertible	5000	21	27		13495
3			alfa-romero	gas	std	two	convertible	5000	21	27		16500
1			alfa-romero	gas	std	two	hatchback	54	5000	19	26	16500
2	164		audi	gas	std	four	sedan	102	5500	24	30	13950
2	164		audi	gas	std	four	sedan	115	5500	18	22	17450
2			audi	gas	std	two	sedan	110	5500	19	25	15250
1	158		audi	gas	std	four		110	5500	19	25	17710
1			audi	gas	std	four		110	5500	19	25	18920
1	158		audi	gas	turbo	four		140	5500	17	20	23875
0			audi	gas	turbo	two		160	5500	16	22	
2	192		bmw	gas	std	two		101	5800	23	29	16430
0	192		bmw	gas	std	four		101	5800	23	29	16925
0	188		bmw	gas	std	two		121	4250	21	28	20970
0	188		bmw	gas	std	four		121	4250	21	28	21105
1			bmw	gas	std	four		121	4250	20	25	24565

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

Prepare the data

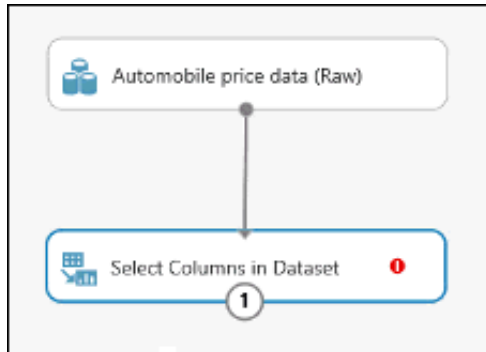
A dataset usually requires some preprocessing before it can be analyzed. You might have noticed the missing values present in the columns of various rows. These missing values need to be cleaned so the model can analyze the data correctly. We'll remove any rows that have missing values. Also, the **normalized-losses** column has a large proportion of missing values, so we'll exclude that column from the model altogether.

Tip

Cleaning the missing values from input data is a prerequisite for using most of the modules.

First, we add a module that removes the **normalized-losses** column completely. Then we add another module that removes any row that has missing data.

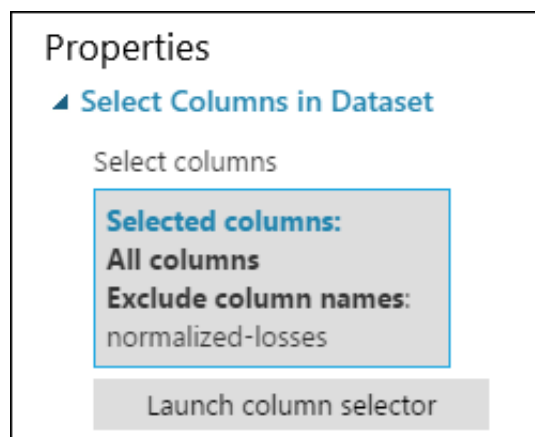
1. Type **select columns** in the search box at the top of the module palette to find the [Select Columns in Dataset](#) module. Then drag it to the experiment canvas. This module allows us to select which columns of data we want to include or exclude in the model.
2. Connect the output port of the **Automobile price data (Raw)** dataset to the input port of the Select Columns in Dataset.



3. Click the [Select Columns in Dataset](#) module and click **Launch column selector** in the **Properties** pane.
 - On the left, click **With rules**
 - Under **Begin With**, click **All columns**. These rules direct [Select Columns in Dataset](#) to pass through all the columns (except those columns we're about to exclude).
 - From the drop-downs, select **Exclude** and **column names**, and then click inside the text box. A list of columns is displayed. Select **normalized-losses**, and it's added to the text box.
 - Click the check mark (OK) button to close the column selector (on the lower right).

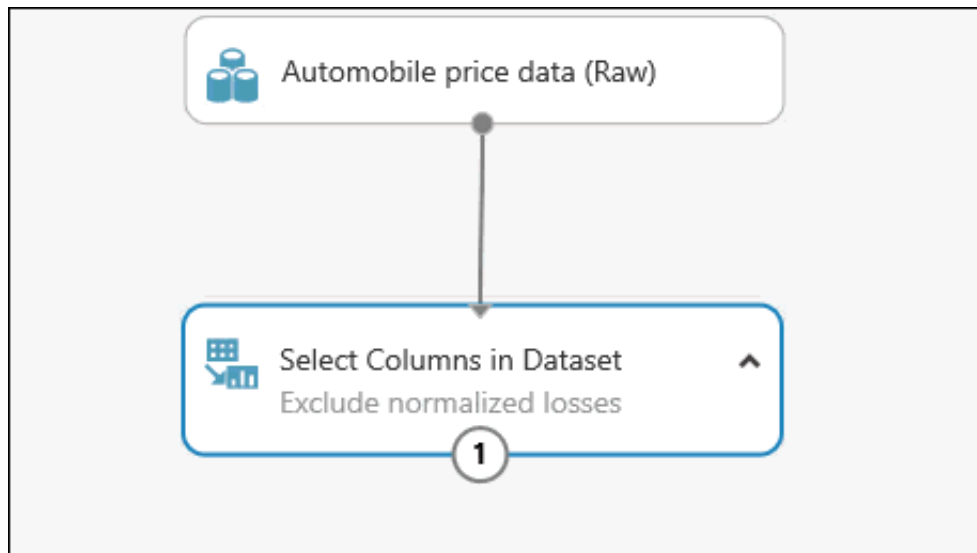


Now the properties pane for **Select Columns in Dataset** indicates that it will pass through all columns from the dataset except **normalized-losses**.



💡 Tip

You can add a comment to a module by double-clicking the module and entering text. This can help you see at a glance what the module is doing in your experiment. In this case double-click the **Select Columns in Dataset** module and type the comment "Exclude normalized losses."



4. Drag the [Clean Missing Data](#) module to the experiment canvas and connect it to the [Select Columns in Dataset](#) module. In the **Properties** pane, select **Remove entire row** under **Cleaning mode**. These options direct [Clean Missing Data](#) to clean the data by removing rows that have any missing values. Double-click the module and type the comment "Remove missing value rows."

Properties

▲ **Clean Missing Data**

Columns to be cleaned

Selected columns:
All columns

Launch column selector

Minimum missing value ratio

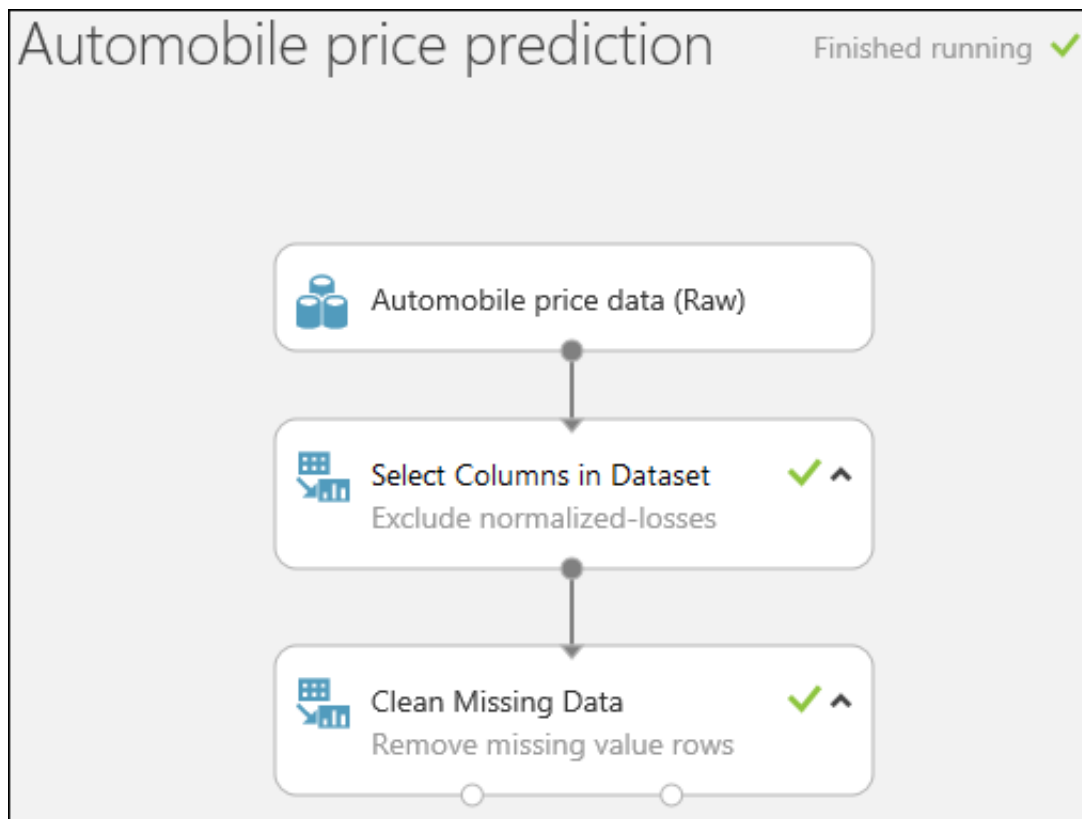
Maximum missing value ratio

Cleaning mode
Remove entire row ▼

5. Run the experiment by clicking **RUN** at the bottom of the page.

When the experiment has finished running, all the modules have a green check mark to indicate that they finished successfully. Notice also the **Finished**

running status in the upper-right corner.



💡 Tip

Why did we run the experiment now? By running the experiment, the column definitions for our data pass from the dataset, through the [Select Columns in Dataset](#) module, and through the [Clean Missing Data](#) module. This means that any modules we connect to [Clean Missing Data](#) will also have this same information.

Now we have clean data. If you want to view the cleaned dataset, click the left output port of the [Clean Missing Data](#) module and select **Visualize**. Notice that the **normalized-losses** column is no longer included, and there are no missing values.

Now that the data is clean, we're ready to specify what features we're going to use in the predictive model.

Define features

In machine learning, *features* are individual measurable properties of something

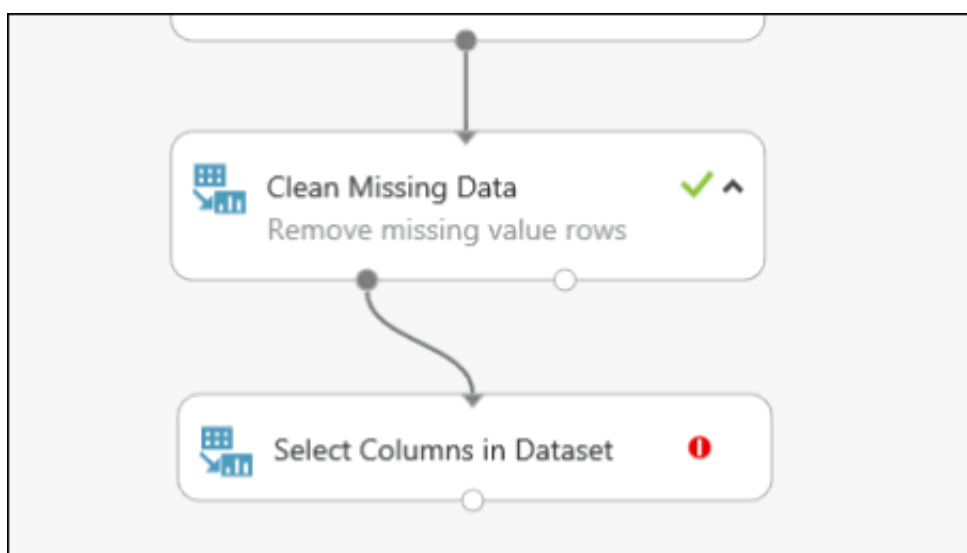
you're interested in. In our dataset, each row represents one automobile, and each column is a feature of that automobile.

Finding a good set of features for creating a predictive model requires experimentation and knowledge about the problem you want to solve. Some features are better for predicting the target than others. Some features have a strong correlation with other features and can be removed. For example, city-mpg and highway-mpg are closely related so we can keep one and remove the other without significantly affecting the prediction.

Let's build a model that uses a subset of the features in our dataset. You can come back later and select different features, run the experiment again, and see if you get better results. But to start, let's try the following features:

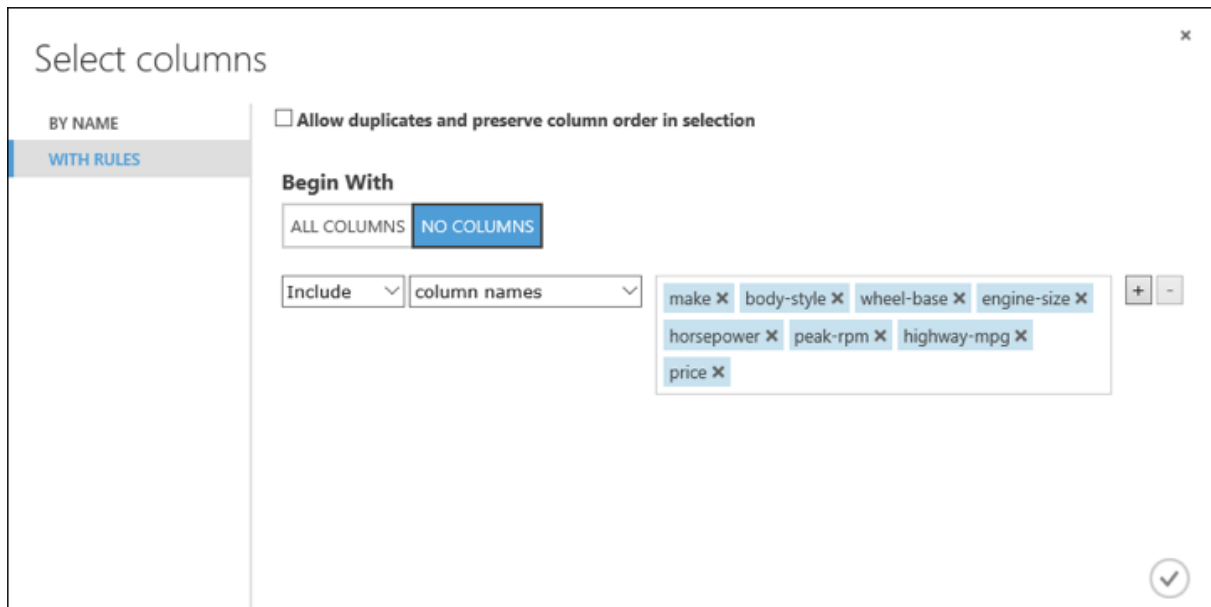
	Copy
make, body-style, wheel-base, engine-size, horsepower, peak-rpm, highway-mpg, price	

1. Drag another [Select Columns in Dataset](#) module to the experiment canvas. Connect the left output port of the [Clean Missing Data](#) module to the input of the [Select Columns in Dataset](#) module.



2. Double-click the module and type "Select features for prediction."
3. Click **Launch column selector** in the **Properties** pane.

4. Click **With rules**.
5. Under **Begin With**, click **No columns**. In the filter row, select **Include** and **column names** and select our list of column names in the text box. This filter directs the module to not pass through any columns (features) except the ones that we specify.
6. Click the check mark (OK) button.



This module produces a filtered dataset containing only the features we want to pass to the learning algorithm we'll use in the next step. Later, you can return and try again with a different selection of features.

Choose and apply an algorithm

Now that the data is ready, constructing a predictive model consists of training and testing. We'll use our data to train the model, and then we'll test the model to see how closely it's able to predict prices.

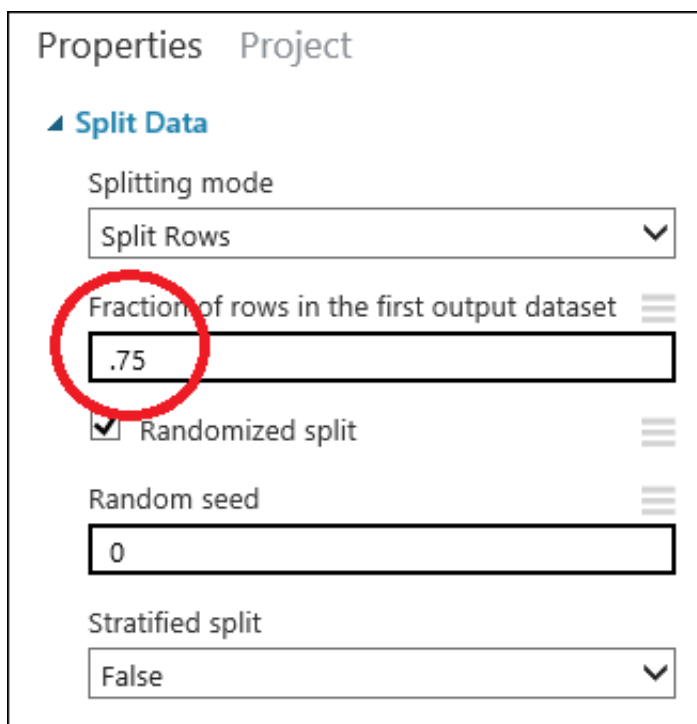
Classification and *regression* are two types of supervised machine learning algorithms. Classification predicts an answer from a defined set of categories, such as a color (red, blue, or green). Regression is used to predict a number.

Because we want to predict price, which is a number, we'll use a regression algorithm. For this example, we'll use a *linear regression* model.

We train the model by giving it a set of data that includes the price. The model scans the data and look for correlations between an automobile's features and its price. Then we'll test the model - we'll give it a set of features for automobiles we're familiar with and see how close the model comes to predicting the known price.

We'll use our data for both training the model and testing it by splitting the data into separate training and testing datasets.

1. Select and drag the [Split Data](#) module to the experiment canvas and connect it to the last [Select Columns in Dataset](#) module.
2. Click the [Split Data](#) module to select it. Find the **Fraction of rows in the first output dataset** (in the **Properties** pane to the right of the canvas) and set it to 0.75. This way, we'll use 75 percent of the data to train the model, and hold back 25 percent for testing.



The screenshot shows the 'Properties' pane for the 'Split Data' module. The 'Splitting mode' is set to 'Split Rows'. The 'Fraction of rows in the first output dataset' is set to '.75' and is circled in red. The 'Randomized split' checkbox is checked. The 'Random seed' is set to '0'. The 'Stratified split' is set to 'False'.

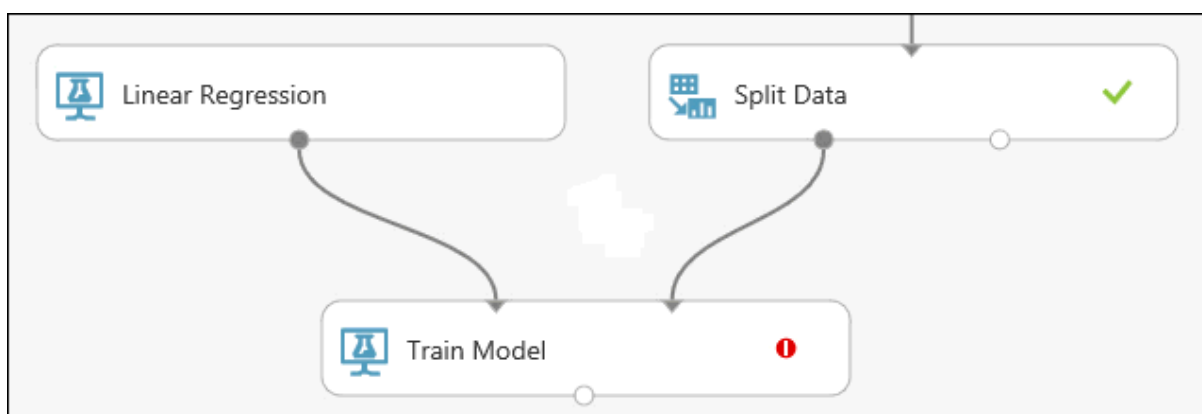
💡 Tip

By changing the **Random seed** parameter, you can produce different random samples for training and testing. This parameter controls the seeding of the pseudo-random number generator.

3. Run the experiment. When the experiment is run, the [Select Columns in Dataset](#)

and [Split Data](#) modules pass column definitions to the modules we'll be adding next.

4. To select the learning algorithm, expand the **Machine Learning** category in the module palette to the left of the canvas, and then expand **Initialize Model**. This displays several categories of modules that can be used to initialize machine learning algorithms. For this experiment, select the [Linear Regression](#) module under the **Regression** category, and drag it to the experiment canvas. (You can also find the module by typing "linear regression" in the palette Search box.)
5. Find and drag the [Train Model](#) module to the experiment canvas. Connect the output of the [Linear Regression](#) module to the left input of the [Train Model](#) module, and connect the training data output (left port) of the [Split Data](#) module to the right input of the [Train Model](#) module.



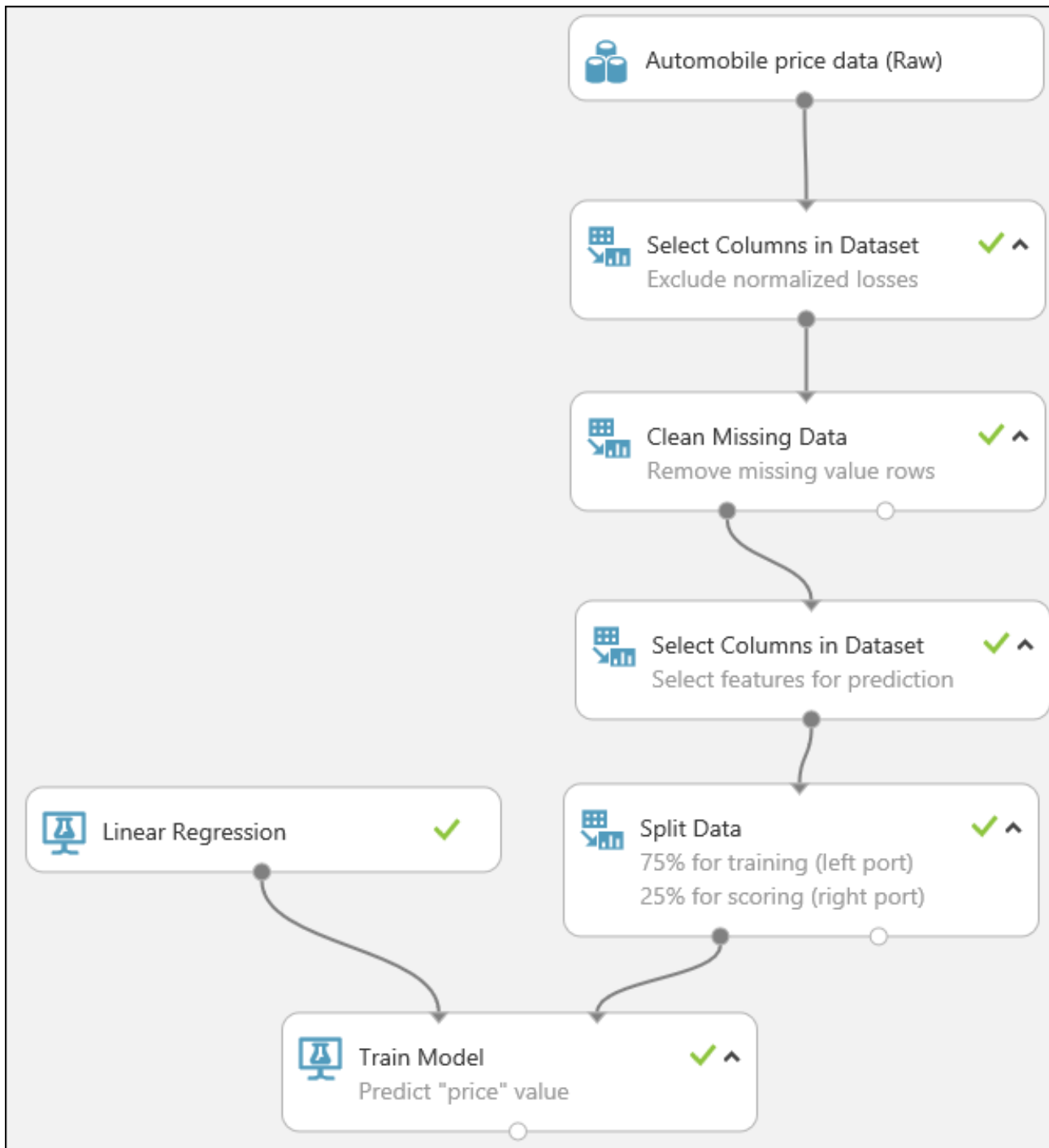
6. Click the [Train Model](#) module, click **Launch column selector** in the **Properties** pane, and then select the **price** column. **Price** is the value that our model is going to predict.

You select the **price** column in the column selector by moving it from the **Available columns** list to the **Selected columns** list.



7. Run the experiment.

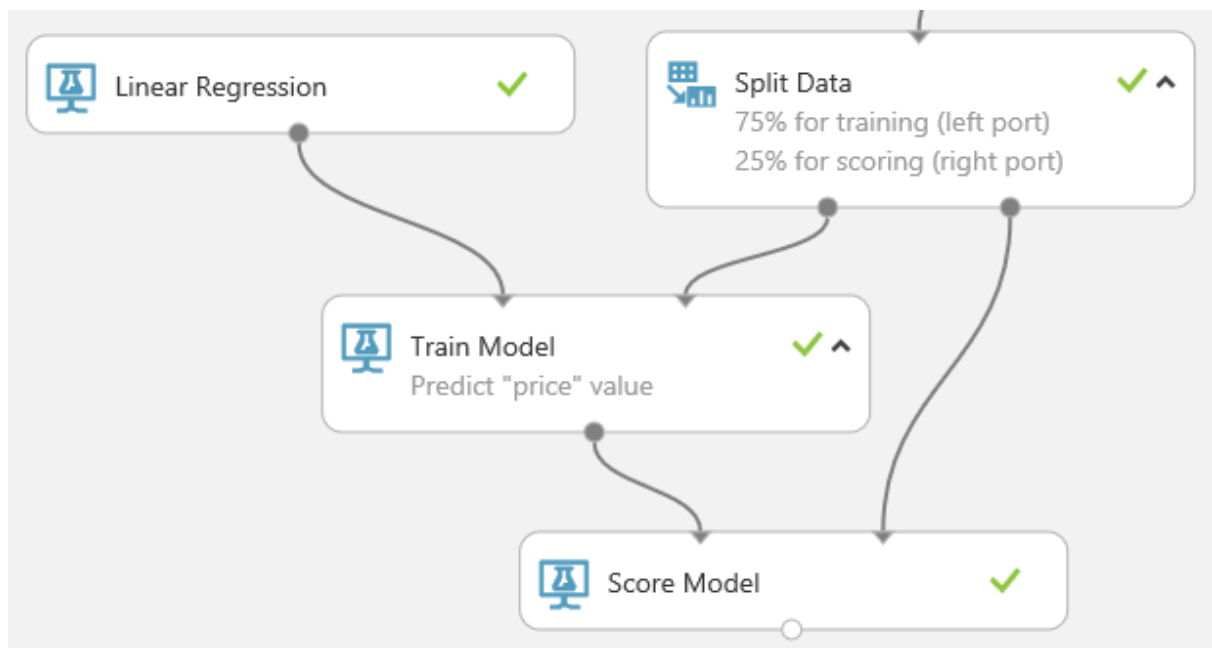
We now have a trained regression model that can be used to score new automobile data to make price predictions.



Predict new automobile prices

Now that we've trained the model using 75 percent of our data, we can use it to score the other 25 percent of the data to see how well our model functions.

1. Find and drag the [Score Model](#) module to the experiment canvas. Connect the output of the [Train Model](#) module to the left input port of [Score Model](#). Connect the test data output (right port) of the [Split Data](#) module to the right input port of [Score Model](#).



- Run the experiment and view the output from the [Score Model](#) module by clicking 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.

Simple test experiment - Copy > Score Model > Scored dataset

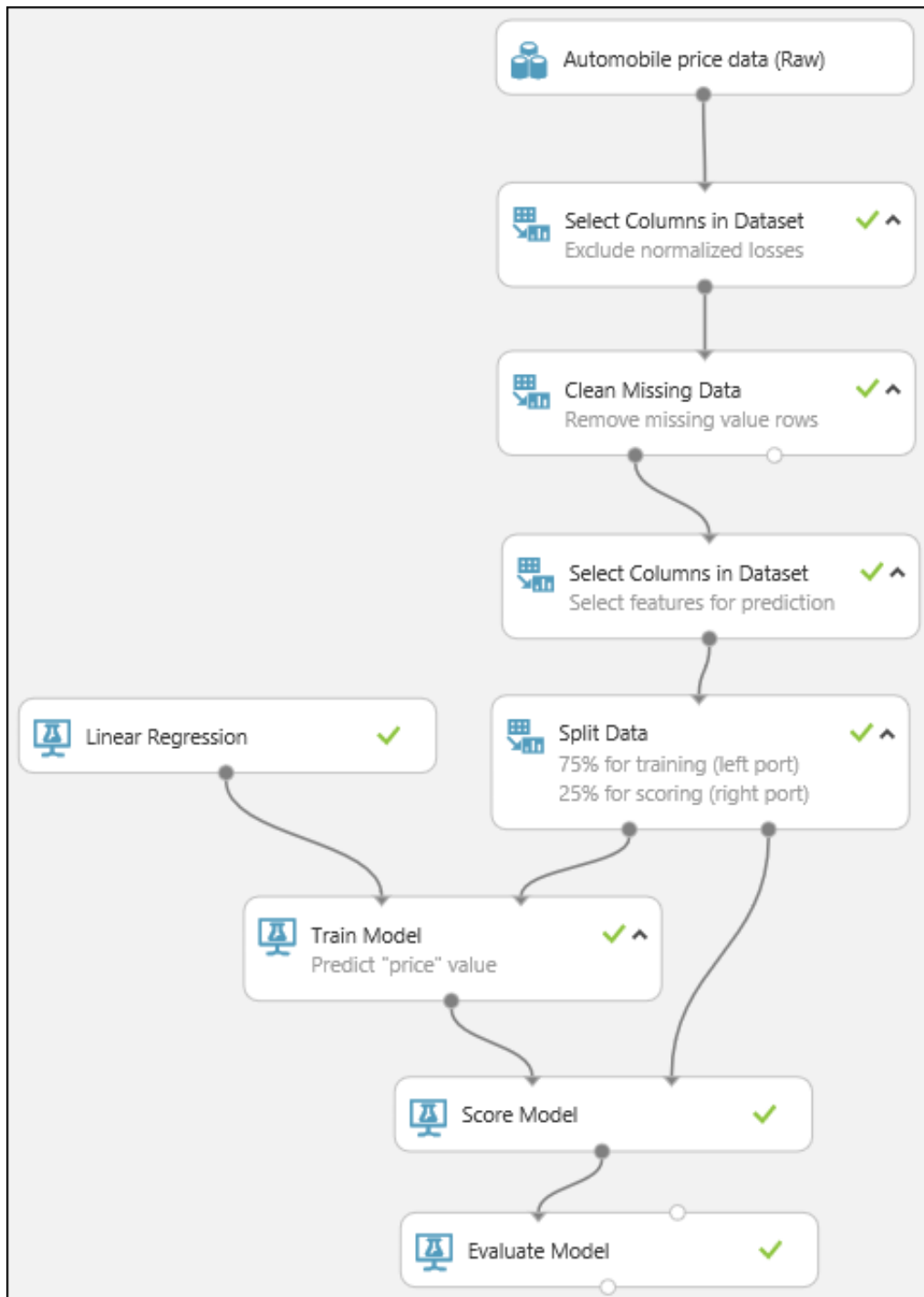
rows: 48, columns: 9

view as:

make	body-style	wheel-base	engine-size	horsepower	peak-rpm	highway-mpg	price	Scored Labels
subaru	sedan	97	108	111	4800	29	11259	10286.204819
mitsubishi	hatchback	93.7	92	68	5500	38	6669	5446.847864
dodge	hatchback	93.7	90	68	5500	38	6229	6344.800711
honda	hatchback	86.6	92	76	6000	38	6855	5528.302953
alfa-romero	convertible	88.6	130	111	5000	27	16500	13498.476233
volvo	wagon	104.3	141	114	5400	28	16515	16097.608038
isuzu	hatchback	96	119	90	5000	29	11048	8315.257218
dodge	hatchback	93.7	90	68	5500	41	5572	6630.154608
honda	sedan	101.2	108	101	5800	29	16420	10912.408695

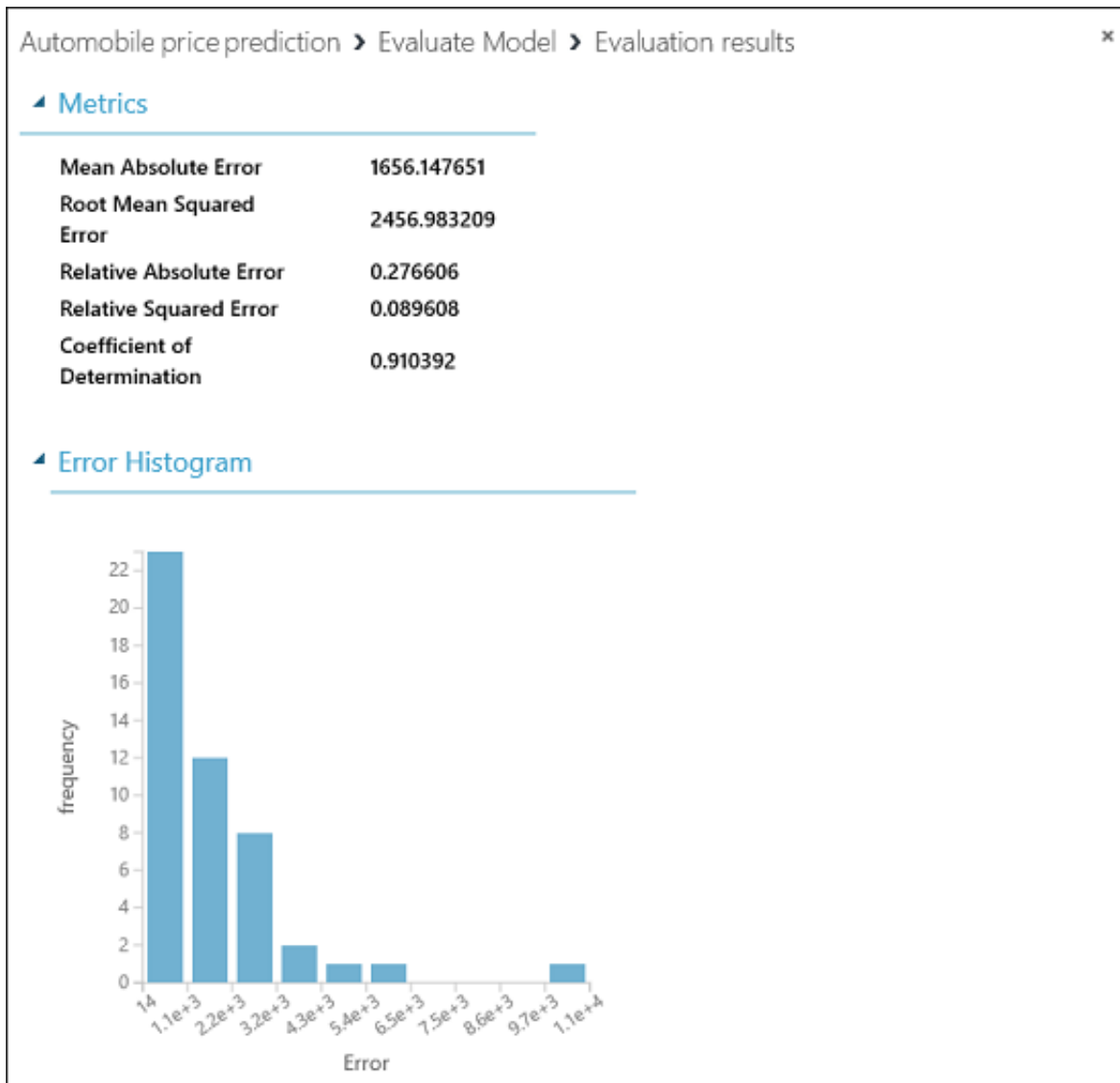
Known values (points to price column) and Predicted values (points to Scored Labels column) are indicated by red arrows.

- Finally, we test the quality of the results. Select and drag the [Evaluate Model](#) module to the experiment canvas, and connect the output of the [Score Model](#) module to the left input of [Evaluate Model](#). The final experiment should look something like this:



4. Run the experiment.

To view the output from the [Evaluate Model](#) module, click the output port, and then select **Visualize**.



The following statistics are shown for our model:

- **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 one (1.0), the better the predictions.

Clean up resources

If you no longer need the resources you created using this article, delete them to avoid incurring any charges. Learn how in the article, [Export and delete in-product user data](#).

Next steps

In this quickstart, you created a simple experiment using a sample dataset. To explore the process of creating and deploying a model in more depth, continue to the predictive solution tutorial.

[Tutorial: Develop a predictive solution in Studio \(classic\)](#)

Is this page helpful?

 Yes  No
