

Contents

Activity 1 – First Machine Learning with Azure	2
Activity 2 – Deploying your experiment as a Web Service.....	19
Activity 3 – Car Damage Assessment Classification	28
Activity 4 – Creating a Sentiment Analyser	39
Activity 5 – [Bonus] Book Genre Classifier.....	51
Activity 6 – [Bonus] Importing data.....	62
Activity 7 – [Bonus] Cleaning and Structuring Data.....	66
Activity 8 – [Bonus] Using Binary Classification Algorithms	71

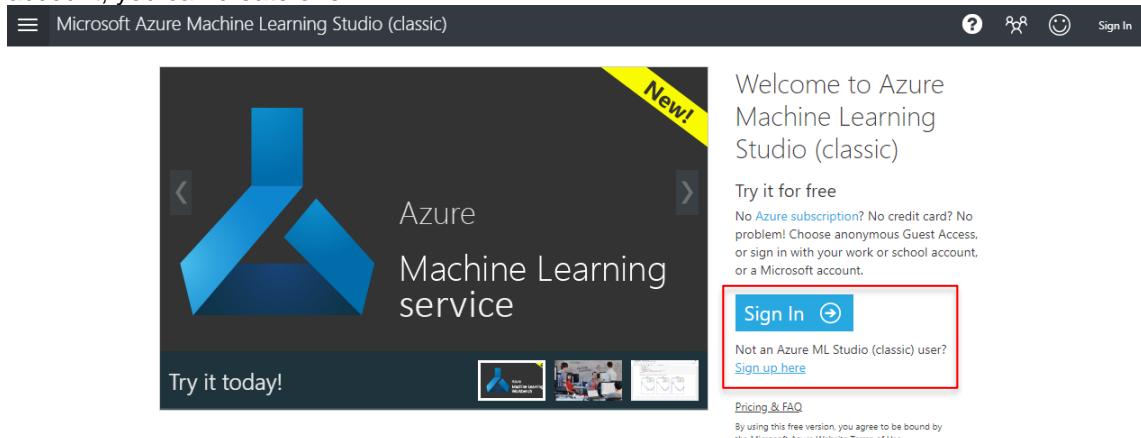
Activity 1 – First Machine Learning with Azure

In this activity, we will:

- Create a new experiment in Azure Machine Learning Studio (Classic)
- Use various dataset modules
- Perform data filtering
- Clean missing data
- Define features for training
- Apply a learning algorithm
- Score a training model
- Evaluate a training model

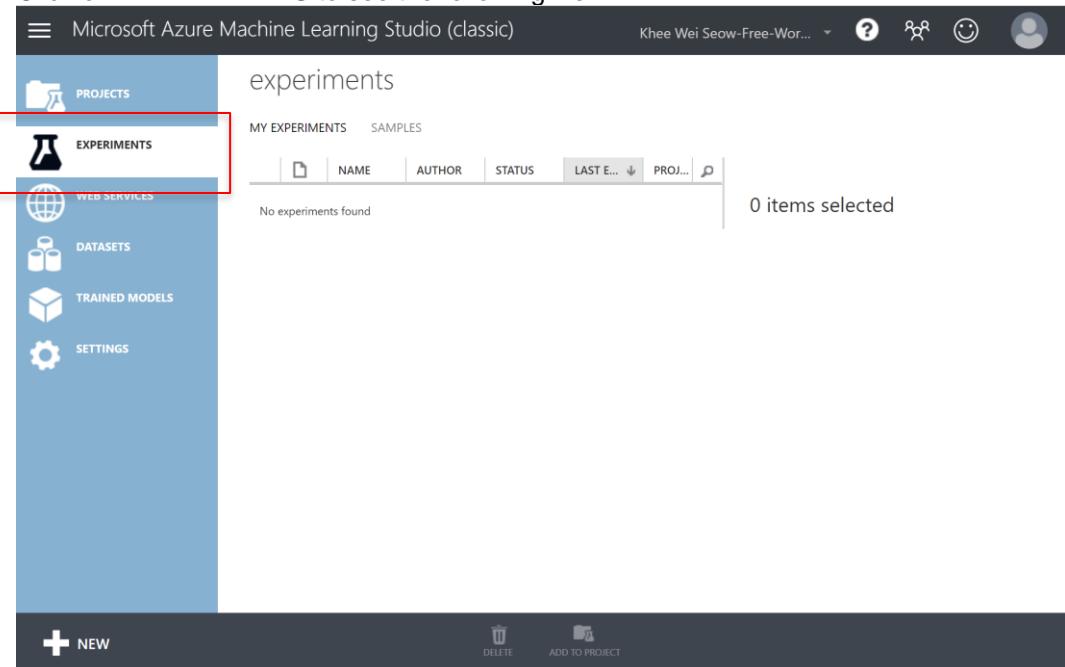
1) Setup account on Azure

- 1) Launch your web browser, navigate to <https://studio.azureml.net/> and sign in. If you have not created an account, you can create one.

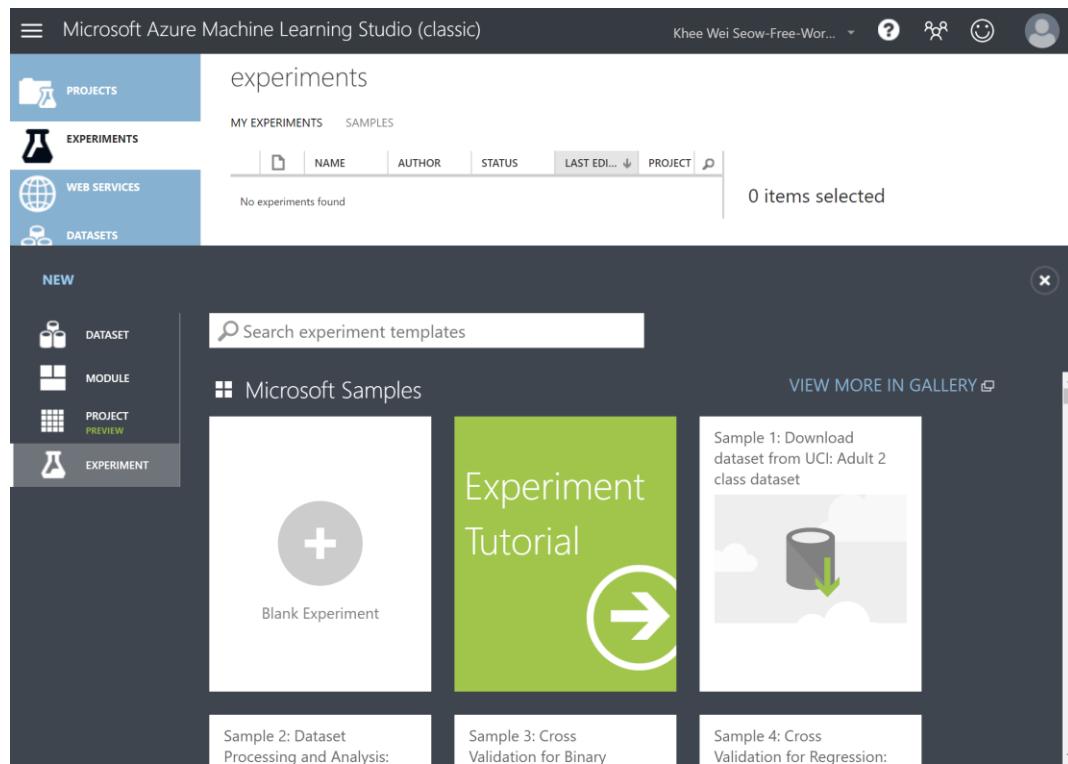


2) Create a new Experiment

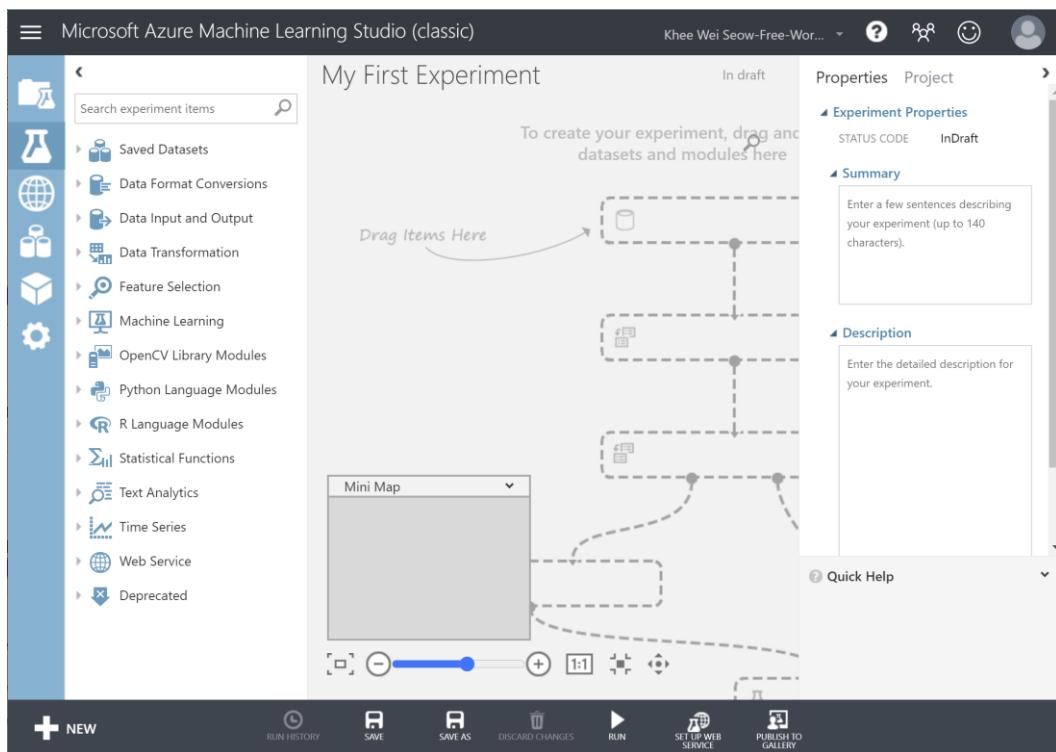
- 1) Click on **EXPERIMENTS** to see the following view



- 2) Click on **+New** at the bottom left of the view and click on **Blank Experiment**.

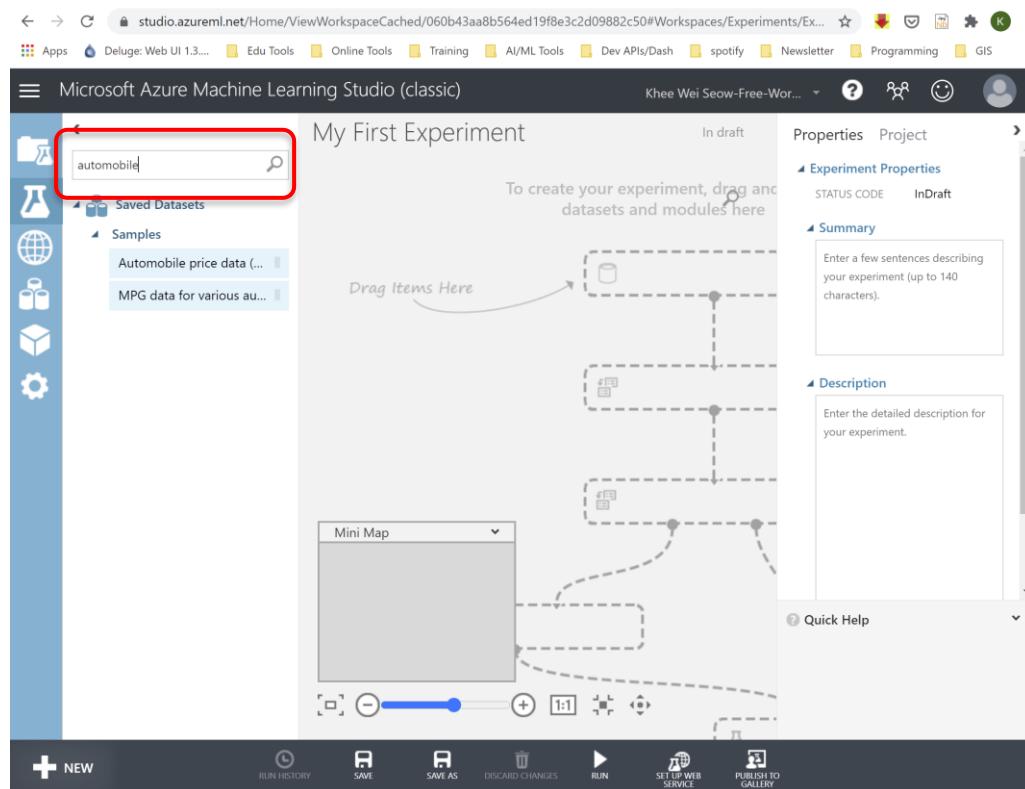


- 3) Name it “**My First Experiment**”. A screen similar to the follow will be presented.

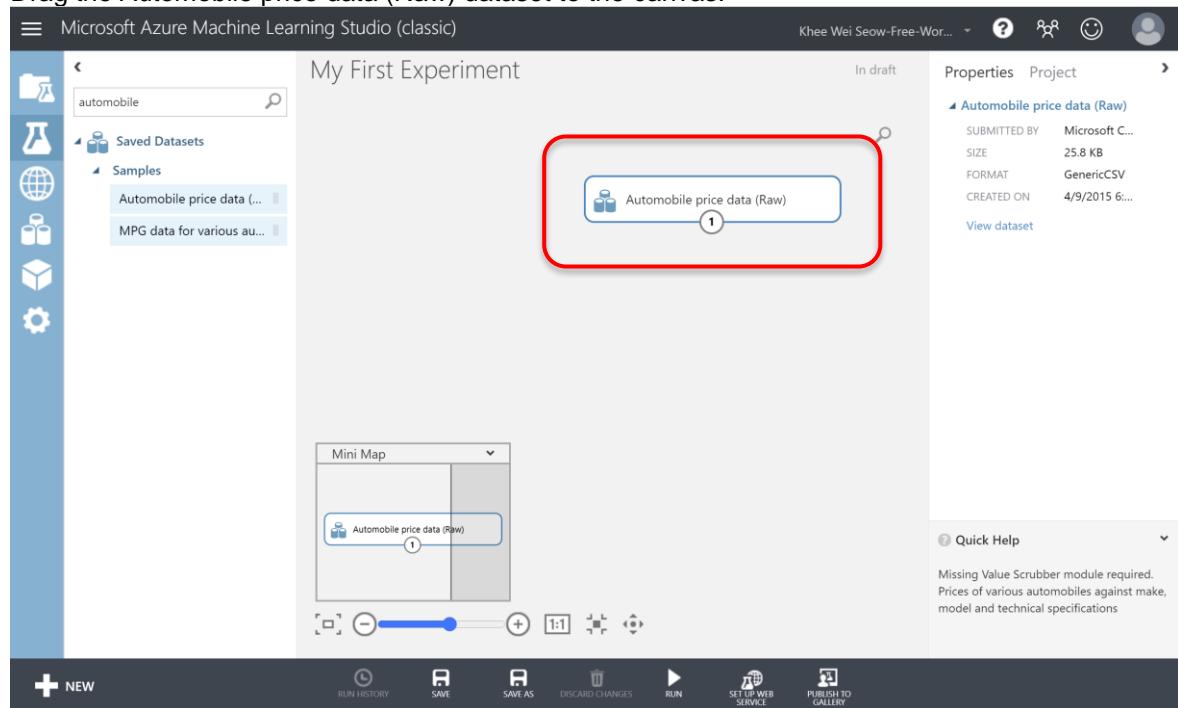


3) Setting up your data

- 1) In the left search box, type Automobile.



- 2) Drag the Automobile price data (Raw) dataset to the canvas.



- 3) Right-click on the output port and click **Visualize**

The screenshot shows the Microsoft Azure Machine Learning Studio interface. On the left, there's a sidebar with icons for datasets, samples, and other project components. The main workspace is titled 'My First Experiment'. In the center, there's a 'Mini Map' canvas where the 'Automobile price data (Raw)' dataset is represented by a small icon. To the right of the canvas, there are several buttons: RUN HISTORY, SAVE, SAVE AS, DISCARD CHANGES, RUN, SET UP WEB SERVICE, and PUBLISH TO GALLERY. Above the canvas, there's a search bar and a properties panel for the selected dataset. The properties panel shows details like 'SUBMITTED BY Microsoft C...', 'SIZE 25.8 KB', 'FORMAT GenericCSV', and 'CREATED ON 4/9/2015 6:...'. A 'Quick Help' section provides information about missing value scrubbing.

- 4) You should see the content of the dataset as shown below. Every column in the dataset is also known as a feature. Notice that some data (rows) have missing values.

My First Experiment > Automobile price data (Raw) > dataset

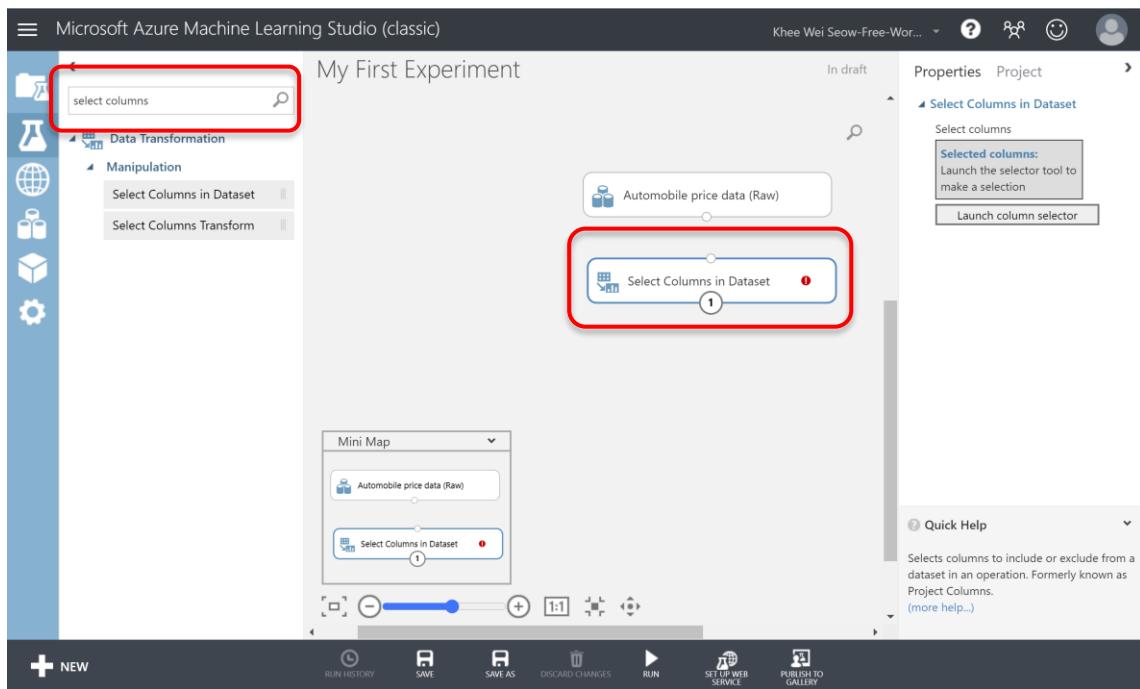
rows 205 columns 26

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors
3			alfa-romero	gas	std	two
1			alfa-romero	gas	std	two
2	164		audi	gas	std	four
2	164		audi	gas	std	four
2			audi	gas	std	two
1	158		audi	gas	std	four
1			audi	gas	std	four
1	158		audi	gas	turbo	four
0			audi	gas	turbo	two

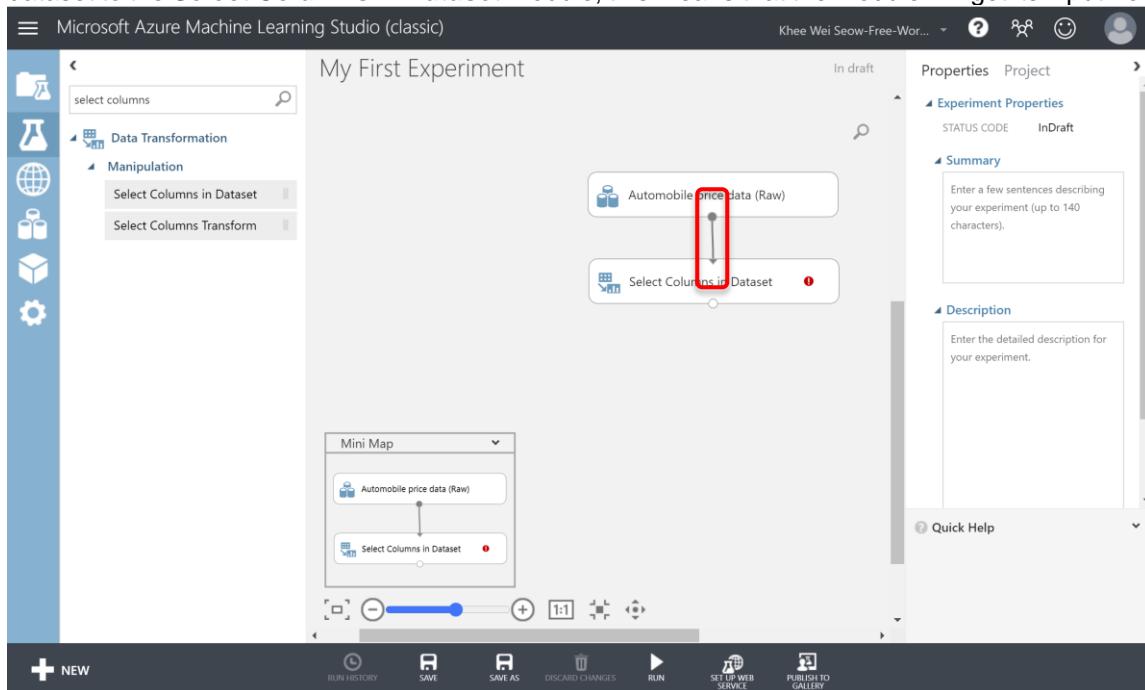
To view, select a column in the table.

4) Preparing your Dataset

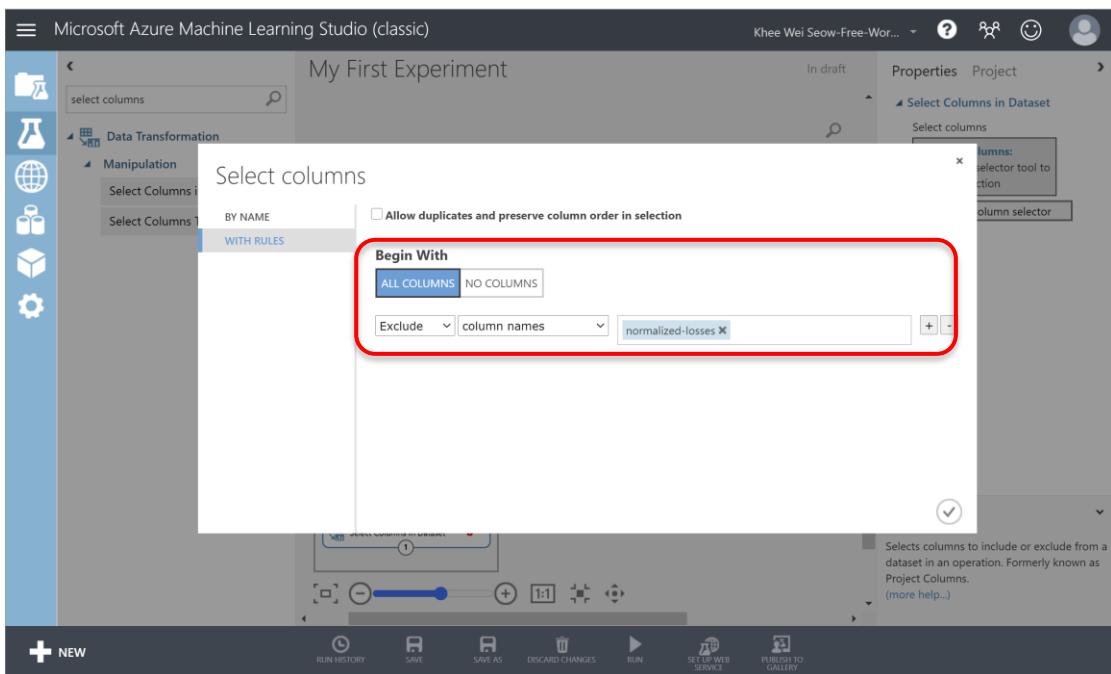
- 1) **Filter data.** In the search box, type **select column** and drag and drop the **Select Columns in Dataset** module onto the canvas. The **Select Columns in Dataset** module allows you to filter the dataset based on the specified column names.



- 2) Connect the output port of the dataset to the input port of the module as shown below. By connecting the dataset to the **Select Columns in Dataset** module, this means that the module will get its input from the dataset.

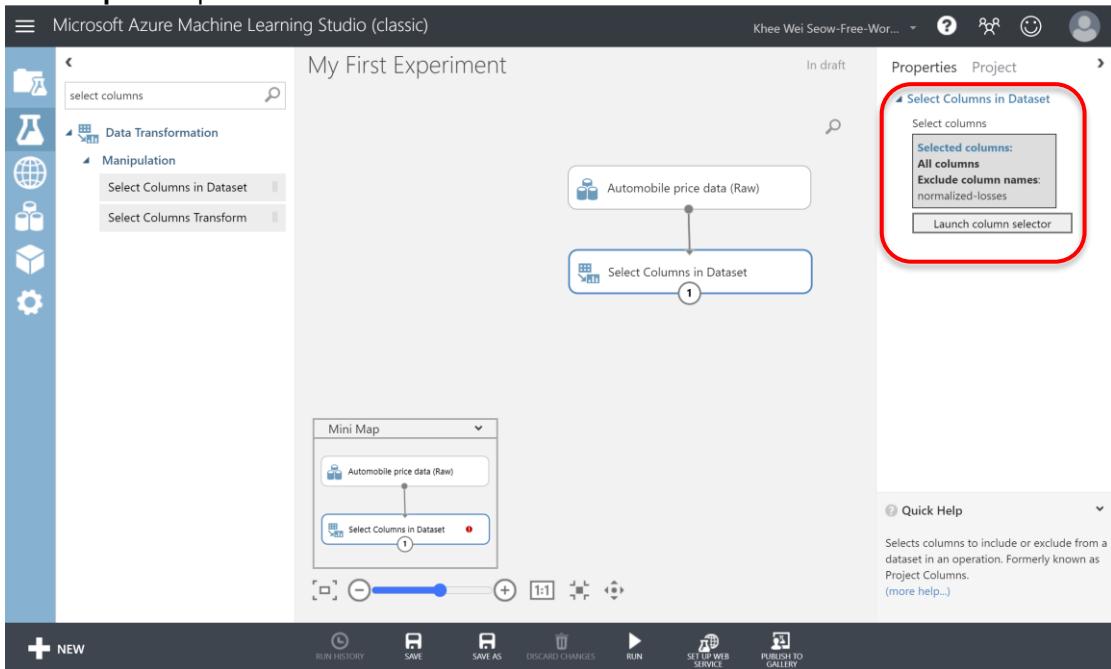


- 3) Select the **Select Columns in Dataset** module and on the Properties pane on the right, click the **Launch column selector** button.
- 4) Set the values as shown below. Click the check mark button when done.

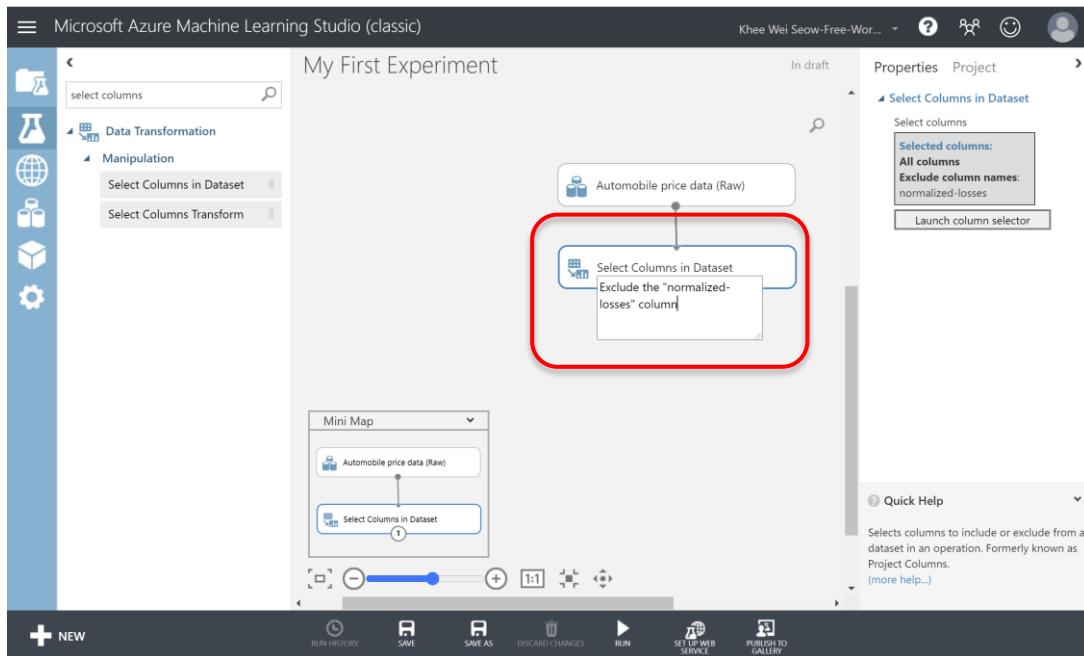


- This rule specifies that you want to exclude the **normalized-losses** column from the dataset.

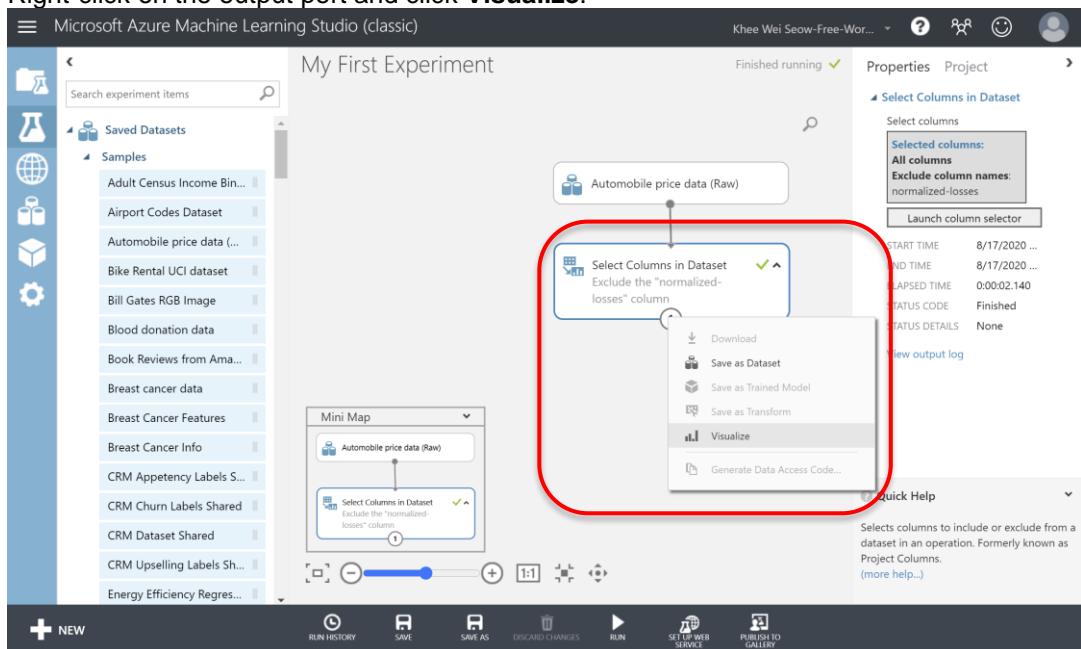
5) The **Properties** pane should now look like this:



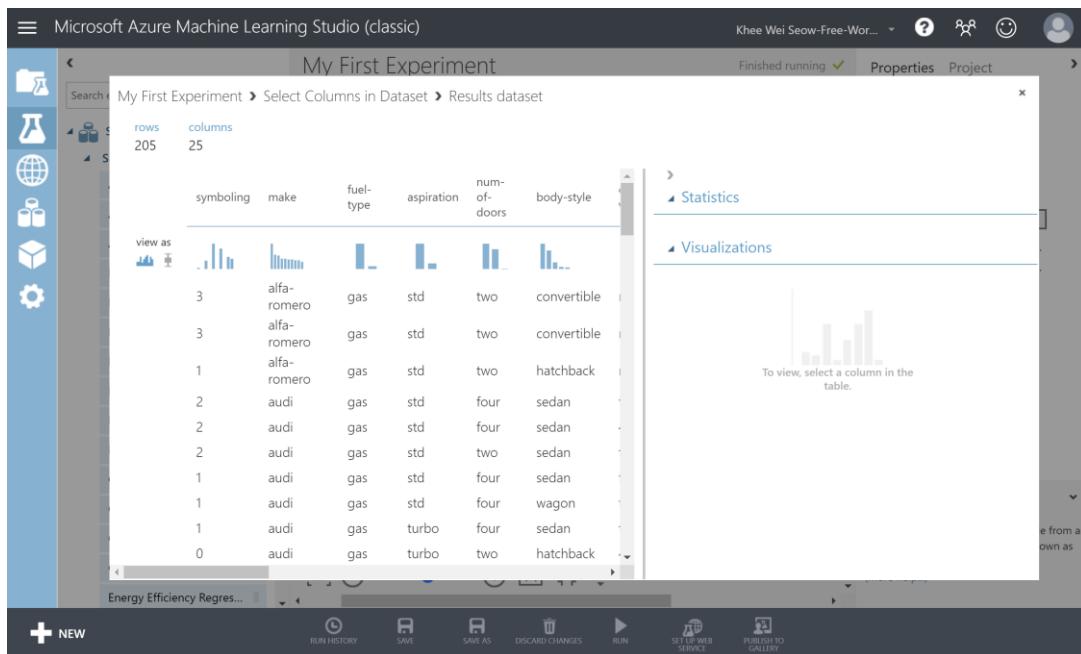
6) Double click on the **Select Columns in Dataset** module to add a comment.



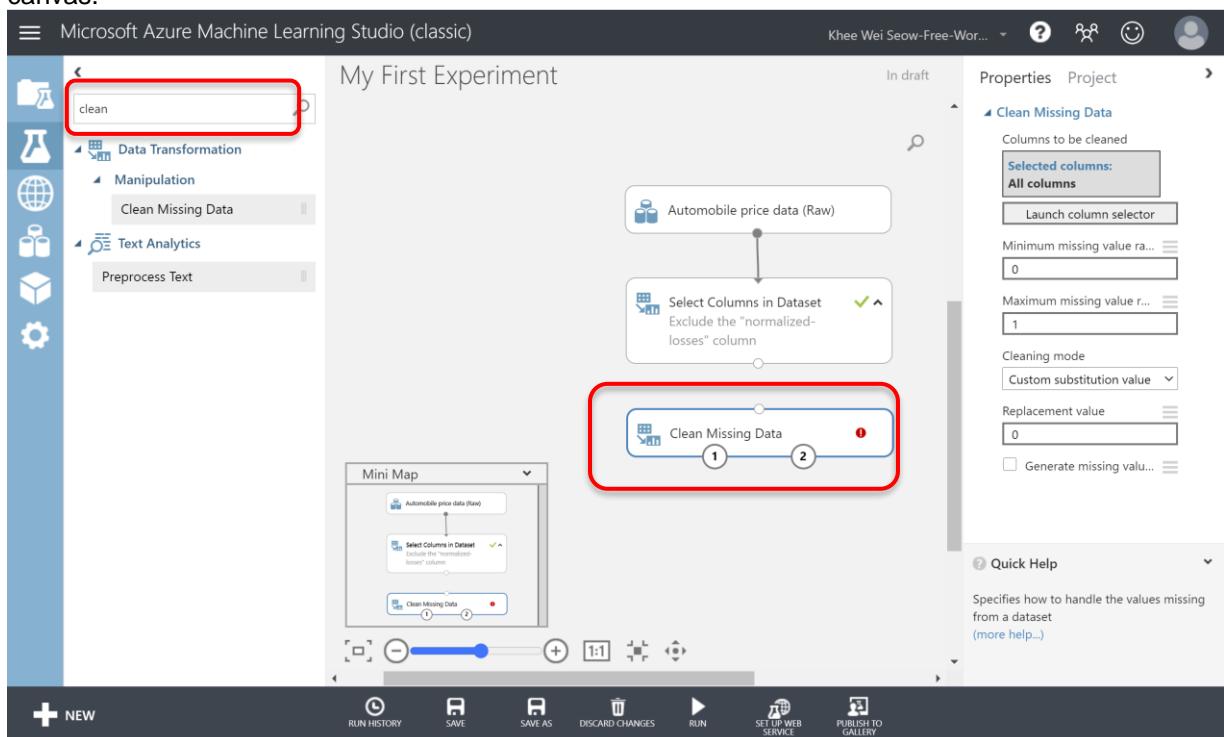
- 7) Click **Run** button located at the bottom of the screen. You will now see a green mark displayed in the **Select Columns in Dataset** module.
- 8) Right-click on the output port and click **Visualize**.



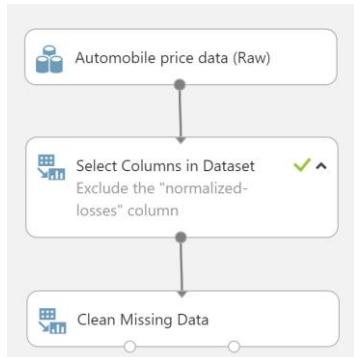
- 9) You should now see that the normalized-losses column is no longer in the dataset.



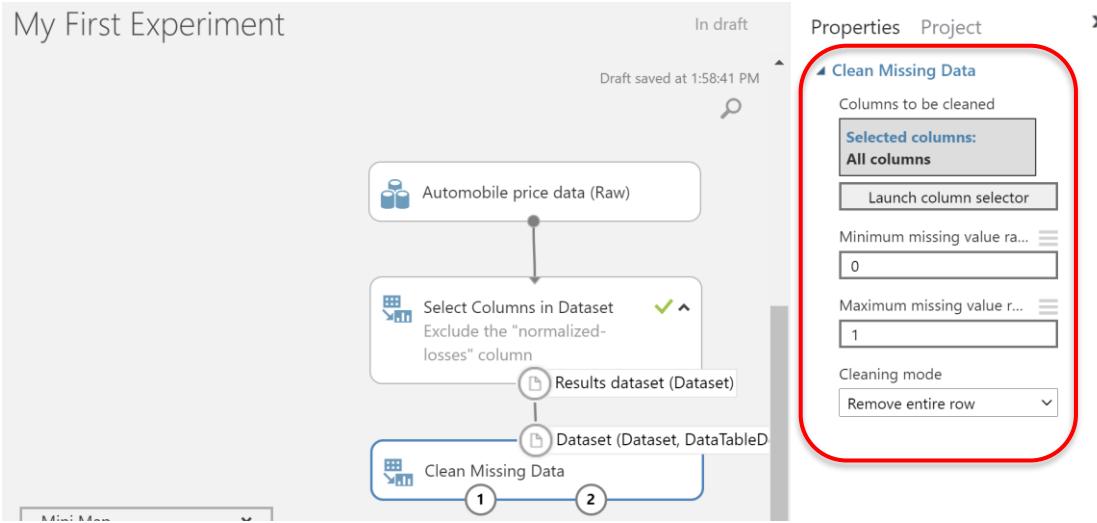
- 10) **Cleaning Data** – In the search box, type Clean missing and Drag the **Clean Missing Data** module to the canvas.



- 11) Connect the **Select Columns in Dataset** module to the **Clean Missing Data** module.



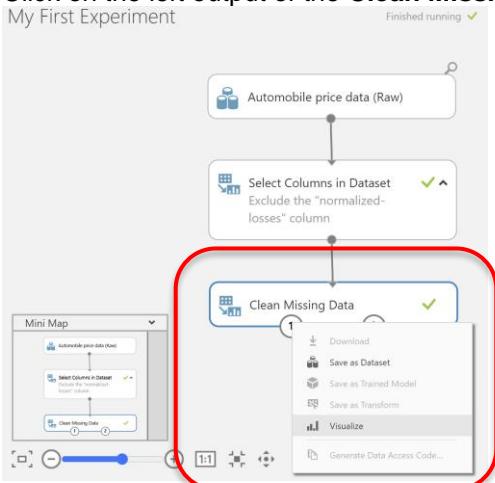
- 12) Select the **Clean Missing Data** module and set its **properties** as follows:



This property removes all rows with missing values. For other options available, you may want to refer to <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/clean-missing-data>

13) Click **Run**. Wait for a couple of seconds and you should see a green tick.

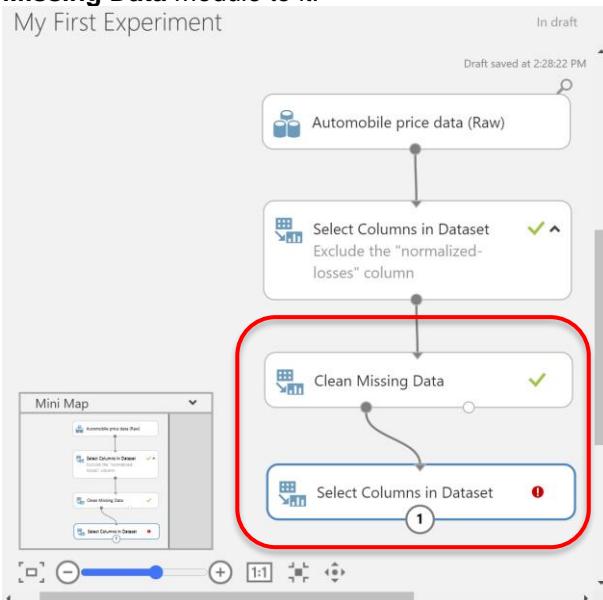
14) Click on the left output of the **Clean Missing Data** module and click **Visualize**.



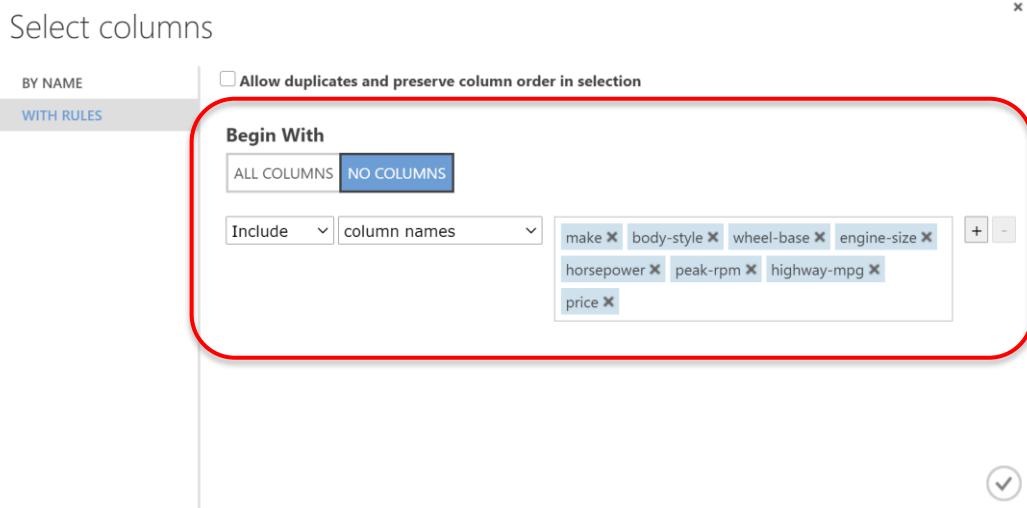
15) The dataset now has removed all rows with missing values:

The screenshot shows the 'Cleaned dataset' visualization. The 'rows' count is 193 and 'columns' count is 25. The table includes columns: symboling, make, fuel-type, aspiration, num-of-doors, body-style, and others. A red box highlights the 'view as' dropdown menu, which includes options like 'Table', 'Scatter', 'Line', etc. To the right, there are sections for 'Statistics' and 'Visualizations'.

- 16) Defining Features – Add another **Select Columns in Dataset** module to the canvas and connect the **Clean Missing Data** module to it.

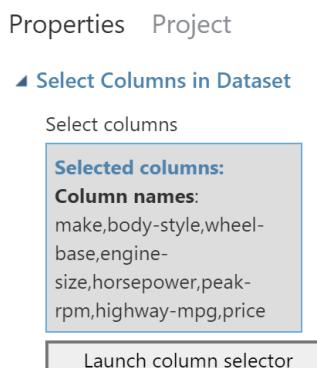


- 17) Double-click on the module and type **Select features for prediction**. (For comment purpose)
 18) Select the module and in the **Properties** pane, click **Launch column selector**.
 19) Set the values as shown below. Click on the check mark button when done.

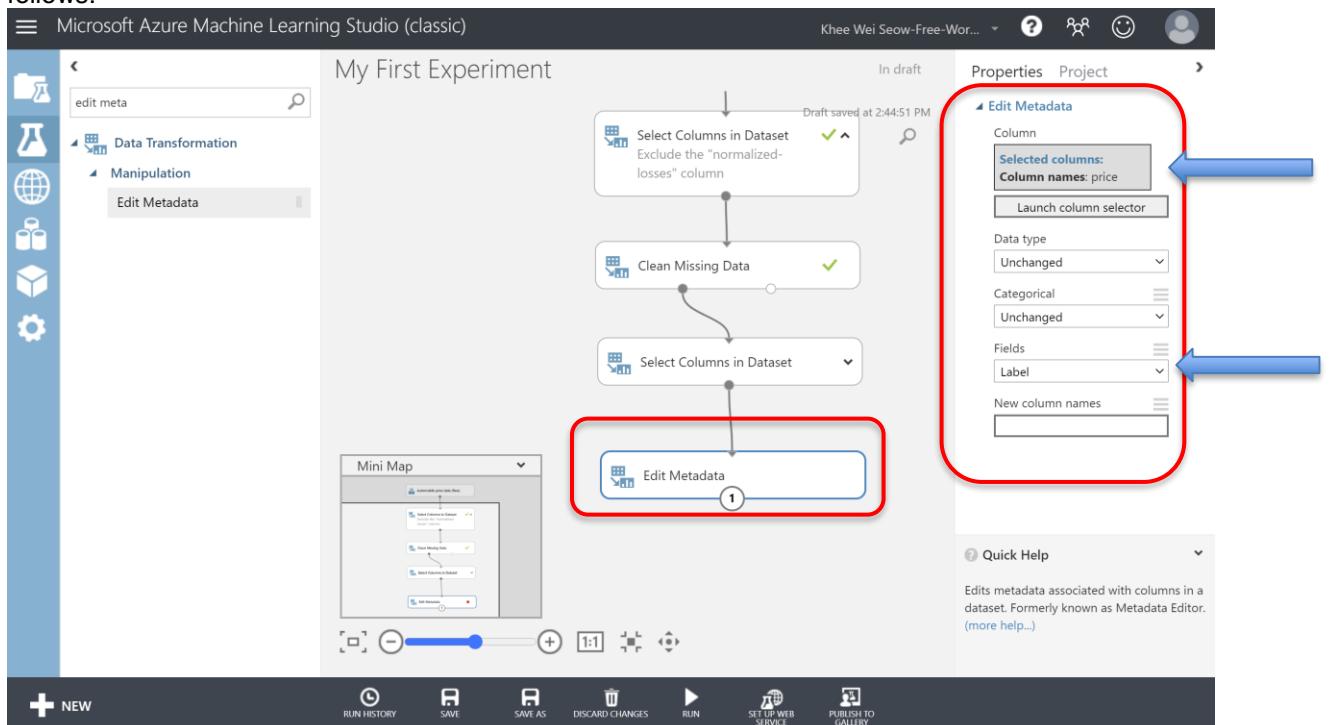


- We are effectively including the following columns: **make, body-style, wheel-base, engine-size, horsepower, peak-rpm, highway-mpg, price**

- 20) The **Properties** pane should now look like this.

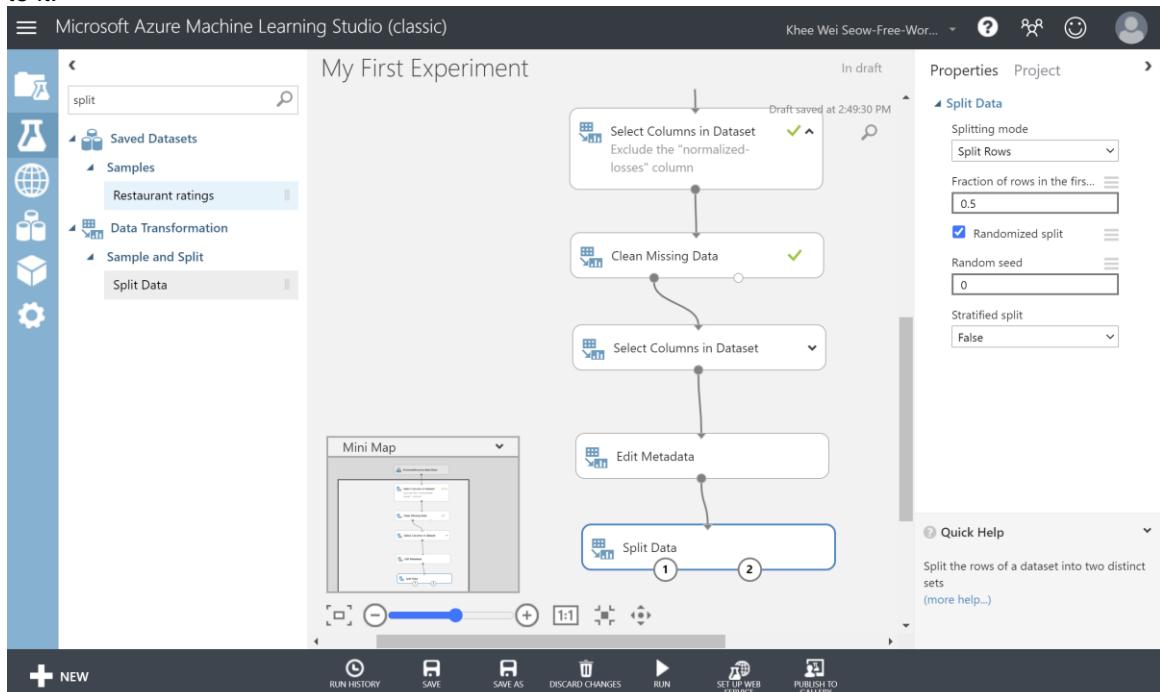


- 21) **Labelling a Feature** – Add the **Edit Metadata** module to the canvas and then connect and configure it as follows:



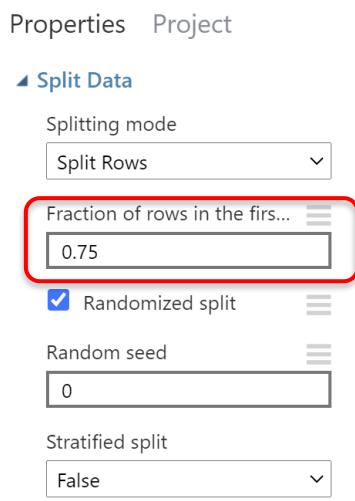
In this case, we are specifying the **price** column as the label. **A label defines the output of your learning model.** I.e. What we are going to predict.

- 22) **Splitting the dataset** – Add the **split data** module to the canvas and then connect the **Edit Metadata** module to it:



The **Split Data** module allows you to split the dataset into 2 groups – one for training the model and the other to use for testing the accuracies/performance of the prediction.

- 23) Select the **Split Data** module and set its properties as follows:



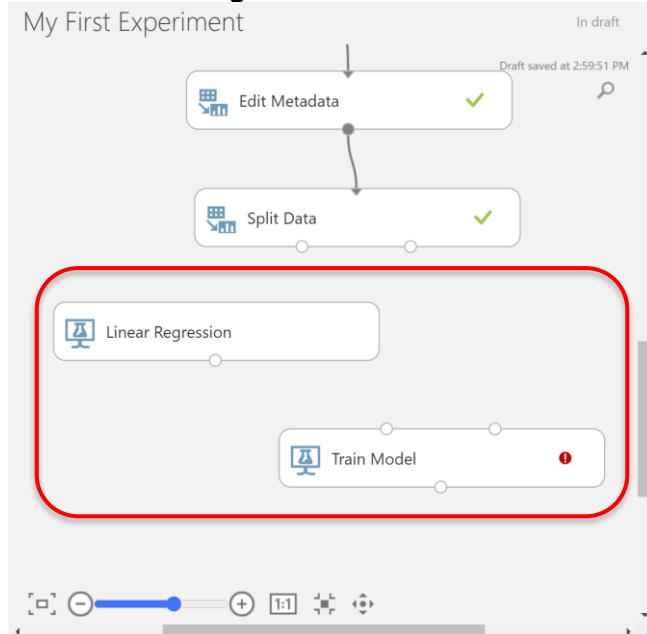
Ref: <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/split-data>

The above setting splits the dataset into 2 parts – 75% of it for training the model and the rest, 25%, for testing.

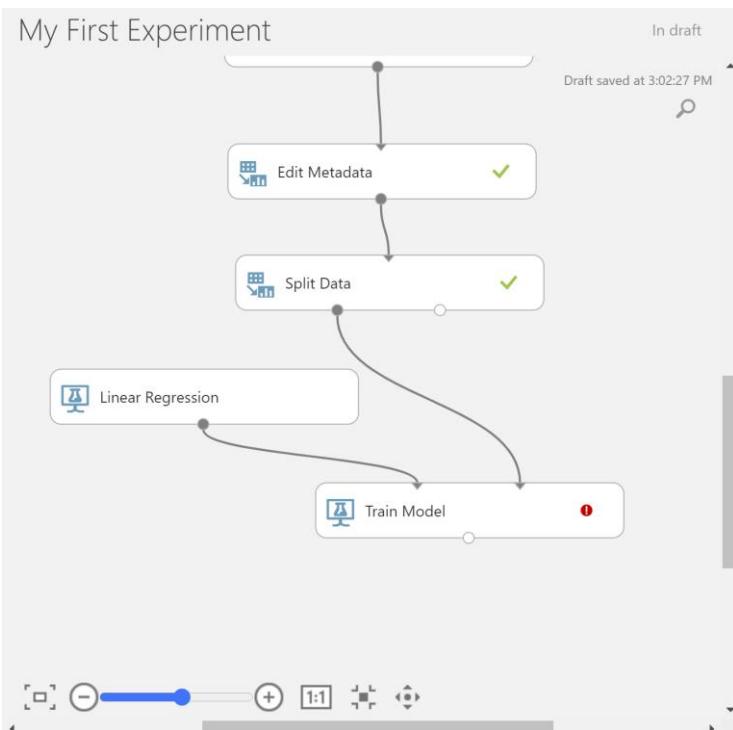
24) Click **Run**.

5) Select and score a model (or learning algorithm)

- 1) Add the **Linear Regression** and **Train Model** modules to the canvas.

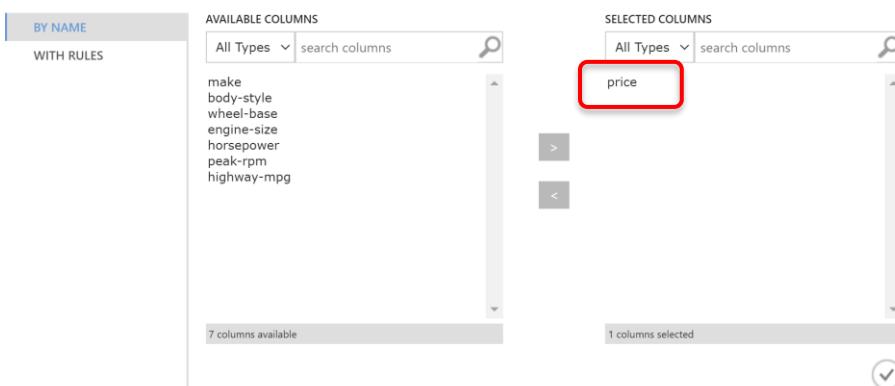


- 2) Connect the **Linear Regression** module to the left input port of the **Train Model** module and the left output port of the **Split Data** module to the right input port of the **Train Model** module.



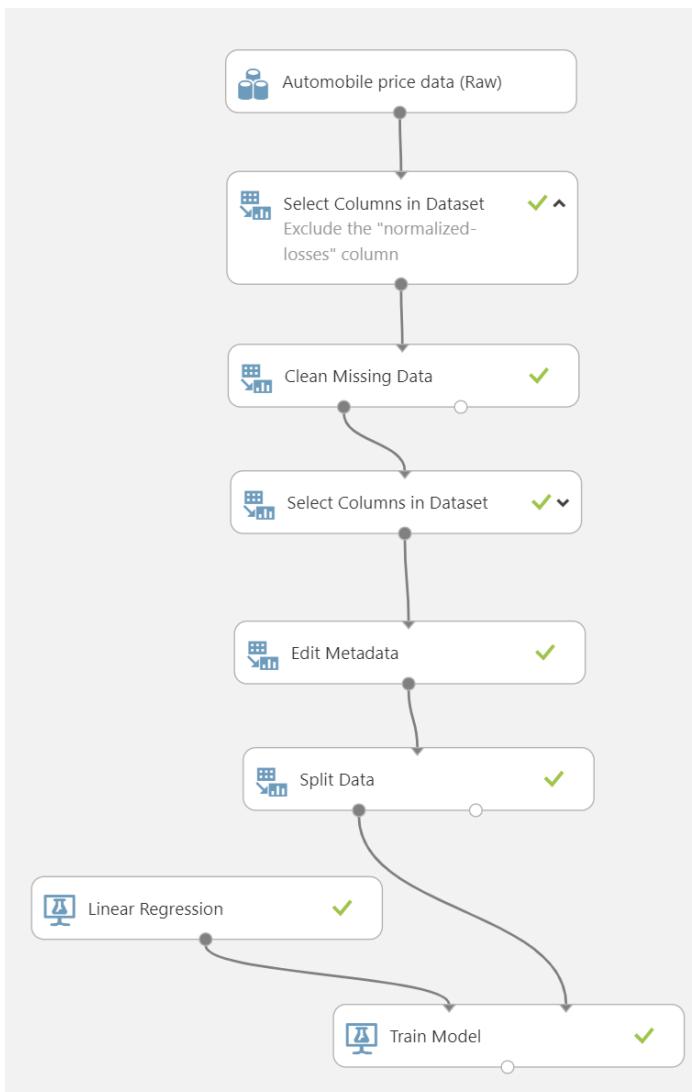
- 3) Select the **Train Model** module and in the **Properties** pane, click the **Launch column** selector button. Select **price** in the AVAILABLE COLUMNS pane and click on the **>** button to move it to the SELECTED COLUMNS pane. Click on the tick button to finish this step.

Select a single column

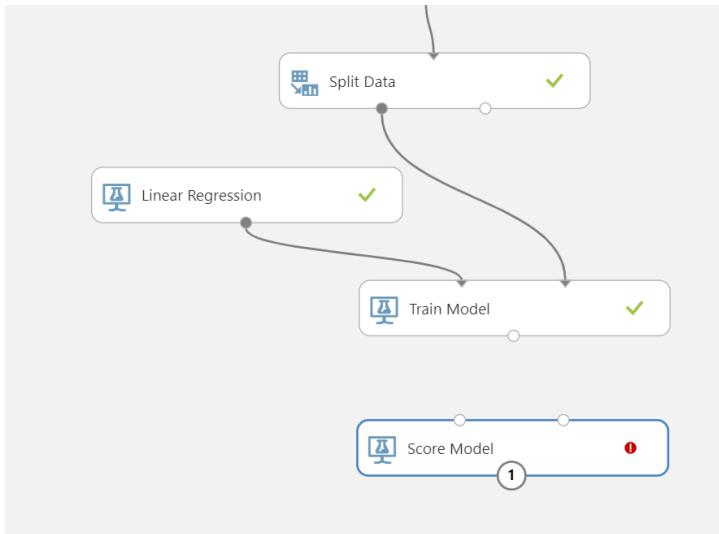


The above step indicates that you want the training model to predict the prices of vehicles.

- 4) Click **Run**. The canvas should now look like this:

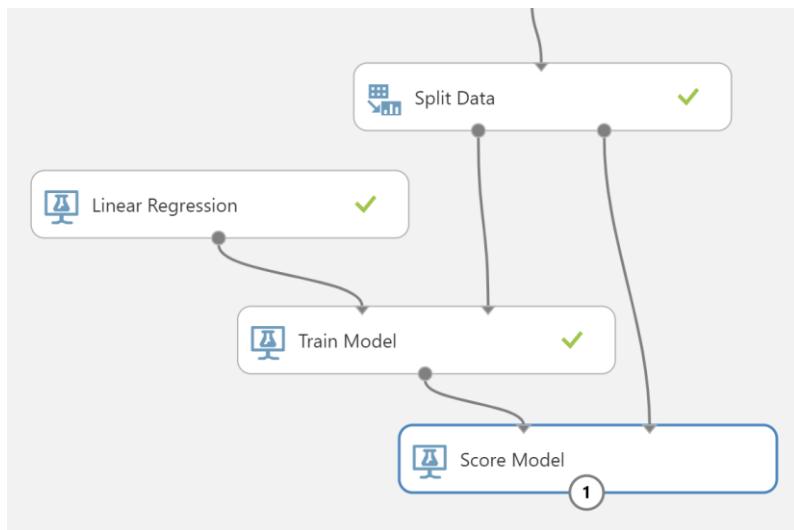


- 5) Add a **Score Model** module to the canvas:

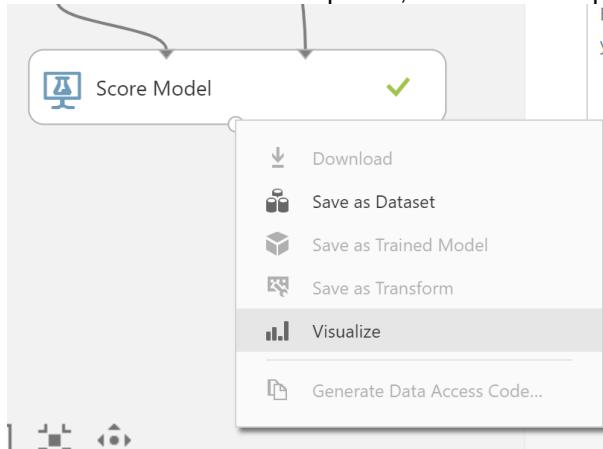


Ref: <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/score-model>

- 6) Connect the output port of the **Train Model** to the left input port of the **Score Model** module and the right output port of the **Split Data** module to the right input port of the **Score Model** module:



- 7) Click **RUN**. When it is completed, click on the output port of the **Score Model** and click **Visualize**:



- 8) You should see the following output.

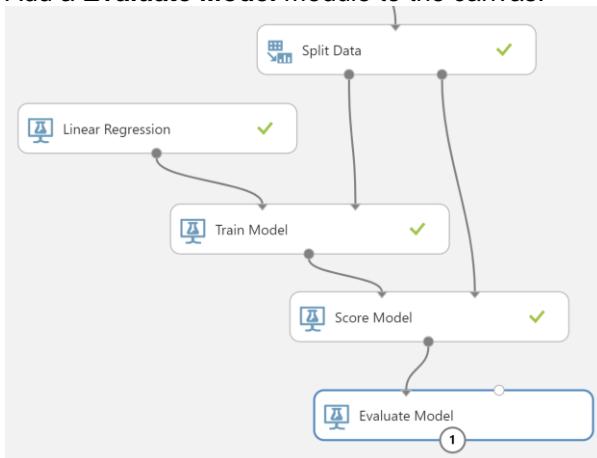
My First Experiment > Score Model > Scored dataset

rows	columns	make	body-style	wheel-base	engine-size	horsepower	peak-rpm	highway-mpg	price	Scored Labels
48	9	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
		bmw	sedan	101.2	108	101	5800	29	16430	19913.408695
		mitsubishi	hatchback	93.7	92	68	5500	41	5389	5732.201761
		bmw	sedan	103.5	209	182	5400	22	41315	30548.819502
		jaguar	sedan	113	258	176	4750	19	35550	30863.486076
		plymouth	hatchback	93.7	90	68	5500	38	6229	5806.676601
		toyota	hatchback	102.9	171	161	5200	24	16558	17388.014192
		mitsubishi	hatchback	95.9	156	145	5000	24	14489	13094.447938
		plymouth	hatchback	93.7	90	68	5500	41	5572	6092.030497

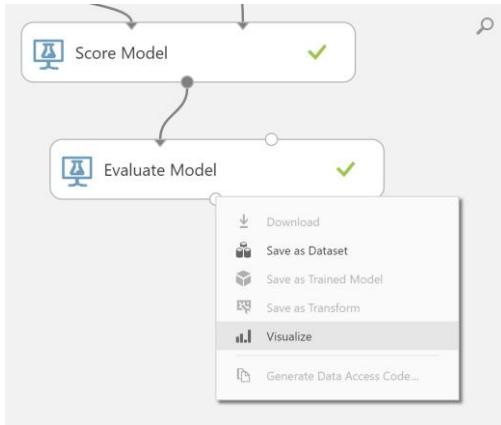
The **price** column shows the actual values and the **Scored Labels** column shows the predicted values.

6) Evaluating the model

- 1) Add a **Evaluate Model** module to the canvas.



- 2) Click **RUN**. Click on the **Evaluate Model** module output port and click **Visualize**:



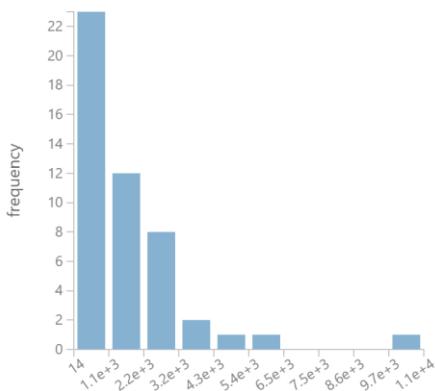
- 3) You should see the following:

My First Experiment > Evaluate Model > Evaluation results

Metrics

Mean Absolute Error	1656.147651
Root Mean Squared Error	2456.983209
Relative Absolute Error	0.276606
Relative Squared Error	0.089608
Coefficient of Determination	0.910392

Error Histogram



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.

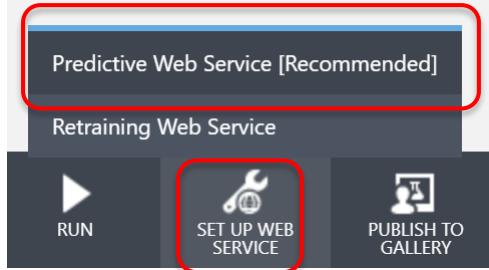
Activity 2 – Deploying your experiment as a Web Service

In this activity, we will learn:

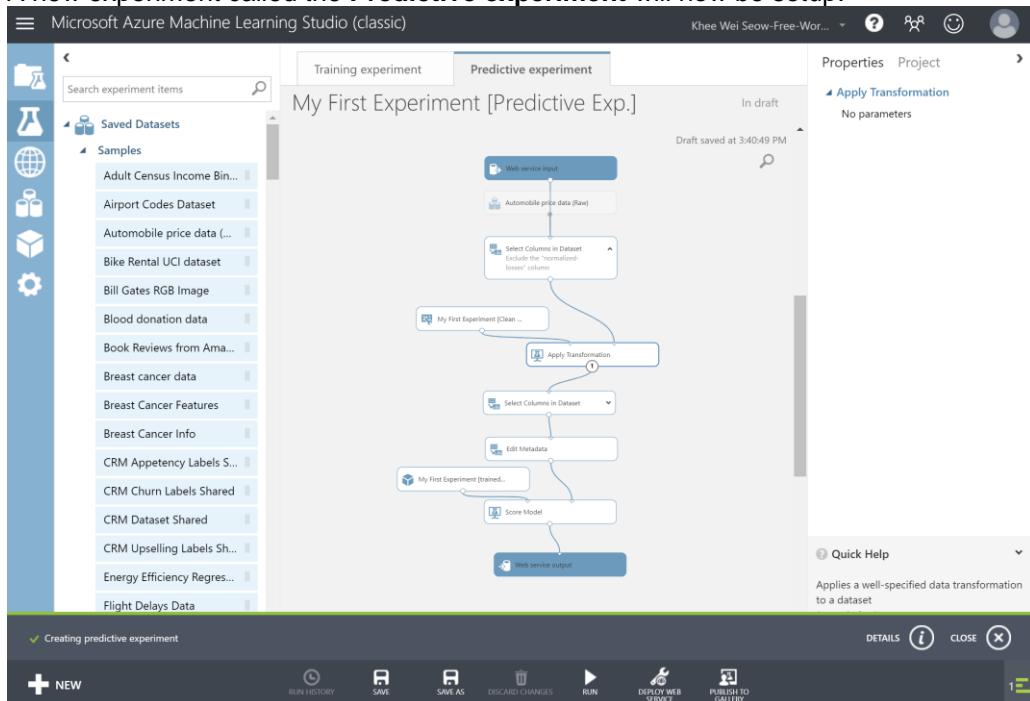
- Deploy a training model as a Web Service
- Test the webservice
- [Optional] access the web service using Python 3
- Use the web services via Excel

1) Prediction using a web Service

- 1) Using the same experiment, click on **SETUP WEB SERVICE, Predictive Web Service**



- 2) A new experiment called the **Predictive experiment** will now be setup:



- 3) Click **RUN**. Then click **DEPLOY WEB SERVICE**.

- 4) You should see the following after a few seconds.

my first experiment [predictive exp.]

The screenshot shows the 'TEST' tab selected in the navigation bar. Below it, there are two sections: 'REQUEST/RESPONSE' and 'BATCH EXECUTION'. Under 'REQUEST/RESPONSE', there is a 'Test preview' button. Under 'BATCH EXECUTION', there is also a 'Test preview' button. The interface includes a header with 'API key' and 'Default Endpoint' fields, and a 'LAST UPDATED' section.

2) Testing the Web Service

- 1) Click on the **Test** hyperlink.

This screenshot is similar to the previous one, but the 'Test preview' button under 'REQUEST/RESPONSE' is highlighted with a red box. The rest of the interface remains the same.

- 2) You should see the following:

This screenshot shows the 'Request-Response' tab selected. At the top, there is a 'Sample Data' section with an 'Enable' button highlighted by a red box. Below this, there are input fields for 'input1' and 'output1', and a prediction result area. A 'View in Studio (classic)' button is visible at the bottom right.

- 3) Click the **Enable** button to populate the various fields with sample data from your dataset.
- 4) At the bottom of the page, click the **Test Request-Response** button to test the web service.

The screenshot shows the 'Test' tab of the Azure Machine Learning Studio interface. A form is displayed with various input fields for a car dataset. The fields include: engine-type (dohc), num-of-cylinders (four), engine-size (130), fuel-system (mpfi), bore (3.47), stroke (2.68), compression-ratio (9), horsepower (111), peak-rpm (5000), city-mpg (21), highway-mpg (27), and price (13495). Below the form is a green button labeled 'Test Request-Response' which is highlighted with a red box.

- 5) The prediction will now be shown.

The screenshot shows the 'Test' tab results. The input fields are the same as in the previous screenshot. To the right of the input fields, the output is displayed: wheel-base (88.6), engine-size (130), horsepower (111), peak-rpm (5000), highway-mpg (27), and price (13495). Below these, a box highlights the 'Scored Labels' field, which contains the value 13498.4762334354.

3) [Optional] Consuming the web service programmatically

- 1) Click the **Consume** tap at the top of the page:

The screenshot shows the 'Consume' tab selected. At the top, it says 'default'. Below that is a section titled 'Web service consumption options' with three icons: 'Excel 2013 or later' (an Excel icon), 'Excel 2010 or earlier' (an Excel icon), and 'Request-Response Web App Template' (a globe icon). To the right is a green button labeled 'View in Studio (classic)' with a small icon. At the bottom, there is a section titled 'Basic consumption info' with a note: 'Want to see how to consume this information? Check out this easy tutorial.'

- 2) Scroll down the page and you should see the following:

```
// This code requires the Nuget package Microsoft.AspNet.WebApi.Client to be installed.
// Instructions for doing this in Visual Studio:
// Tools -> Nuget Package Manager -> Package Manager Console
// Install-Package Microsoft.AspNet.WebApi.Client

using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;

namespace CallRequestResponseService
{
    class Program
    {
        static void Main(string[] args)
        {
            InvokeRequestResponseService().Wait();
        }

        static async Task InvokeRequestResponseService()
        {
            using (var client = new HttpClient())
            {
                var scoreRequest = new
                {
                    Inputs = new Dictionary<string, List<Dictionary<string, string>> () {
```

3) Click on **Python 3+** tab and copy the code.

4) Make Prediction using Excel (2010 or earlier)

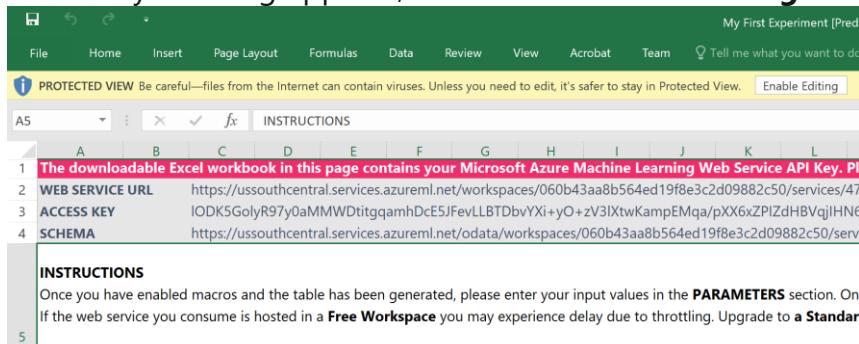
- 1) Azure Machine Learning Studio (classic) makes it easy to call web services directly from Excel without the need to write any code. If you are using Excel 2013 (or later) or Excel Online, then we recommend that you use the Excel Excel add-in (next section).
- 2) In Microsoft Azure Machine Learning Studio (classic), click on **WEB SERVICES** on the left pane. Then click on “My First Experiment [Predictive Exp.] shown below.

NAME	CREATED ON	PROJECT
My First Experiment [Predictive Exp.]	8/17/2020 3:47:03 PM	None

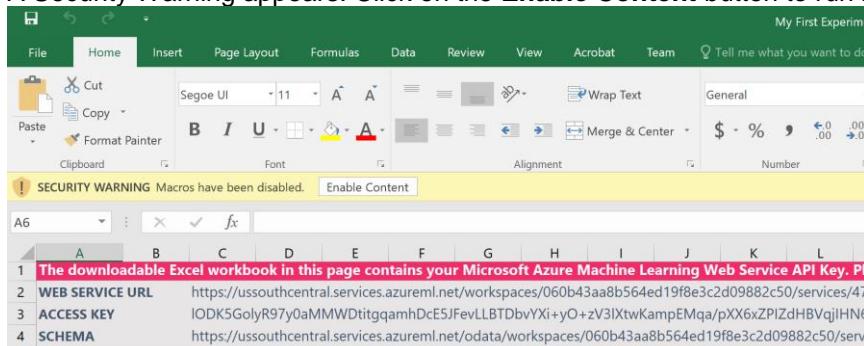
- 3) On the DASHBOARD tab for the web service is a row for the REQUEST/RESPONSE service. Click on **Excel 2010 or earlier workbook** the hyperlink to download the workbook in that row.

API HELP PAGE	TEST	APPS	LAST UPDATED
REQUEST/RESPONSE	Test Test preview	Excel 2013 or later Excel 2010 or earlier workbook	8/17/2020 3:47:05 PM
BATCH EXECUTION	Test preview	Excel 2013 or later workbook	8/17/2020 3:47:05 PM

- 4) Open the workbook.
- 5) A Security Warning appears; click on the **Enable Editing** button.



- 6) A Security Warning appears. Click on the **Enable Content** button to run macros on your spreadsheet.



- 7) Once macros are enabled, a table is generated. Columns in blue are required as input into the RRS web service, or PARAMETERS. Note the output of the service, PREDICTED VALUES in green. When all columns for a given row are filled, the workbook automatically calls the scoring API, and displays the scored results

PARAMETERS		PREDICTED VALUES	
symboling	normalized-losses	make	fuel-type
3	1 alfa-rcgas	std	aspiration

- 8) Key in some data for the blue columns and you will see that the green columns are automatically populated.

PARAMETERS											
PREDICTED VALUES											
symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width
3	1 alfa-rcgas	std	two	convertible	rwd	front		88.6	168.8	64.1	48.8
four	130 mpfi	3.47	2.68	9	111	5000	21	27	13495 alfa-rcconvertible	88.6	130
										5000	27 13495 13498.4762

5) Make Prediction using Excel (after 2013)

- 1) In Microsoft Azure Machine Learning Studio (classic), click on **WEB SERVICES** on the left pane. Then click on "My First Experiment [Predictive Exp.]" shown below.

Microsoft Azure Machine Learning Studio (classic)

web services

NAME	CREATED ON	PROJECT
My First Experiment [Predictive Exp.]	8/17/2020 3:47:03 PM	None

Creating predictive experiment

DETAILS CLOSE

+ NEW ADD TO PROJECT

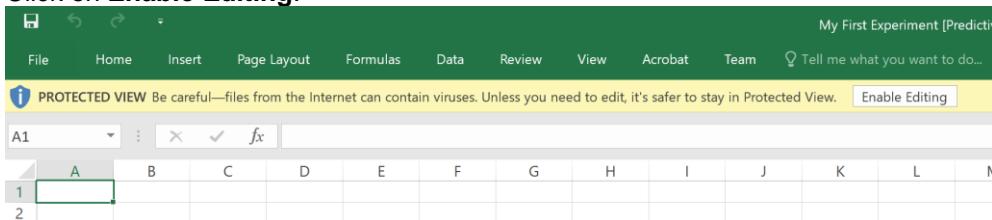
- 2) On the DASHBOARD tab for the web service is a row for the REQUEST/RESPONSE service. Click on **Excel 2013 or later workbook** hyperlink to download the workbook in that row.

Default Endpoint

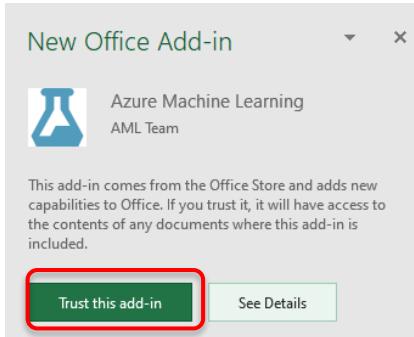
API HELP PAGE	TEST	APPS	LAST UPDATED
REQUEST/RESPONSE	Test	Excel 2013 or later Excel 2010 or earlier workbook	8/17/2020 3:47:05 PM
BATCH EXECUTION	Test	Excel 2013 or later workbook	8/17/2020 3:47:05 PM

- 3) Open the sample Excel file, which contains the Excel add-in.

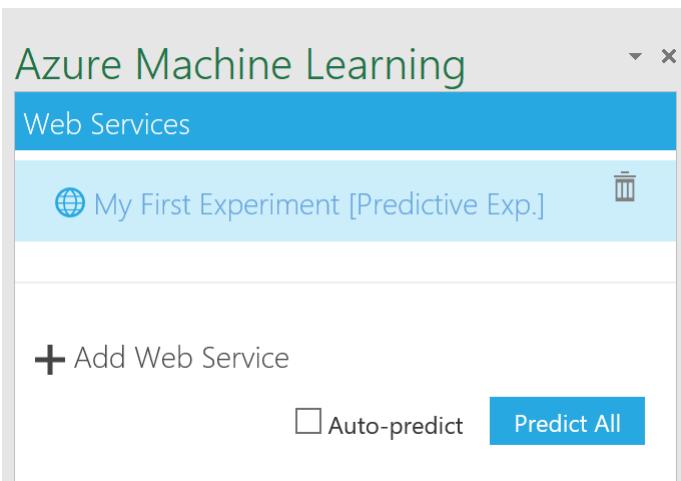
- 4) Click on **Enable Editing**.



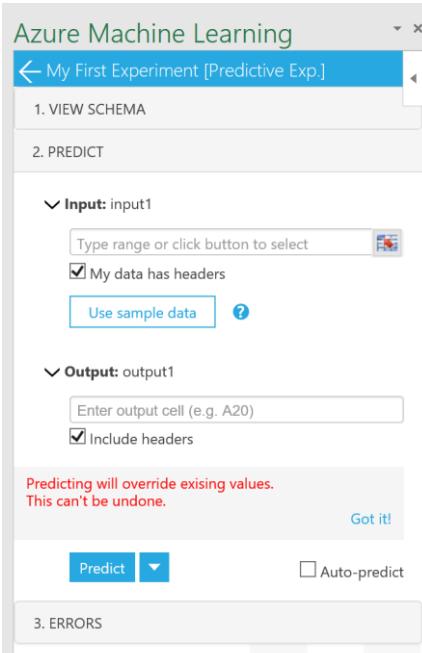
- 5) Click on **Trust this add-in**



- 6) Choose the web service by clicking it – “My First Experiment [Predictive Exp.]” in this activity.



- 7) This takes you to the **Predict** section. For a blank workbook you can select a cell in Excel and click **Use sample data** to data.



- 8) Select the data with headers and click the input data range icon. Make sure the "My data has headers" box is checked.
9) Under Output, enter the cell number where you want the output to be, for example "A10".

- 10) Click **Predict**. If you select the "auto-predict" checkbox any change on the selected areas (the ones specified as input) will trigger a request and an update of the output cells without the need for you to press the predict button.

- 11) You will see the predicted values as follow:

	A	B	C	D	E	F	G	H	I	J
1	symboling	normalized	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
2	3	1	alfa-romero	gas	std	two	convertible	rwd	front	88.6
3	3	1	alfa-romero	gas	std	two	convertible	rwd	front	88.6
4	1	1	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
5	2	164	audi	gas	std	four	sedan	fwd	front	99.8
6	2	164	audi	gas	std	four	sedan	4wd	front	99.4
7										
8										
9										
10	make	body-style	wheel-base	engine-size	horsepower	peak-rpm	highway-mpg	price	Scored Labels	
11	alfa-romero	convertible	88.6	130	111	5000	27	13495	13498.476	
12	alfa-romero	convertible	88.6	130	111	5000	27	16500	13498.476	
13	alfa-romero	hatchback	94.5	152	154	5000	26	16500	14329.816	
14	audi	sedan	99.8	109	102	5500	30	13950	15696.502	
15	audi	sedan	99.4	136	115	5500	22	17450	17161.153	
16										

6) Make Prediction using Excel on the web (E.g. you are using MacOS)

- 1) Since Microsoft Azure Machine Learning Excel add-in only works on windows and Excel on the web, if you are using MacOS, the workaround is to use Excel on the web instead. The easiest way is to copy the downloaded excel file in step (5) to your one drive (via OneDrive agent or Onedrive.com)
- 2) Open your web browser and go to <https://onedrive.live.com>. Upload the excel file to onedrive.
- 3) On the web browser, double click on the xls file.
- 4) In the loading process, you may be redirected to Microsoft website to install the Azure Machine Learning Excel add-in. When prompted, click install to add the add-in. Once complete, you will see a windows as shown below:

Apps > Azure Machine Learning > Launch



Get started with the add-in:

Open in Excel Online

This add-in works in: Excel 2013 Service Pack 1 or later on Windows, Excel 2016 or later on Windows, Excel on the web.



How to start add-ins directly within Office Online

You can launch any add-in within Office Online.

[Click here for step by step instructions](#)

- 5) If your xls does not open in a new browser window, return to your onedrive on the web browser and double click on the xls again. This will open the xls in another browser window with Excel on the web. The following will be shown. Click on the web service shown to load your web service.

- 6) Once loaded, you should see the details of the web service as follows:

- 7) Follow step (5) Make Prediction using Excel (after 2013), 7) and later to define input and output cells and make a prediction.

Activity 3 – Car Damage Assessment Classification

In this activity, we will learn:

- Save training time and create well-performing models with small datasets using transfer learning

Ref: A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning.

<https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>

1) The Problem

In this activity, you will use a pretrained snippet in a classification model designed to detect different types of car damage. The number of images in the input data is small relative to the number of classes and has a significant amount of variation. This makes it challenging to create a well-performing model based on this dataset alone.

2) The Data

The dataset that we will use in this activity contains approximately 1,500 unique RGB images with the dimensions 224 x 224 pixels, and is split into a training- and a validation subset.

Unbalanced dataset

The underrepresented classes in the training subset have been upsampled in the pre-processing stage in order to reduce **bias**. This means that the index file (index.csv) has duplicate entries that are linking to the same image file. The total number of entries in the index file is approximately 3,800.

Classes

Each image belongs to one of the following classes:

- Broken headlamp
- Broken tail lamp
- Glass shatter
- Door scratch
- Door dent
- Bumper dent
- Bumper scratch
- Unknown

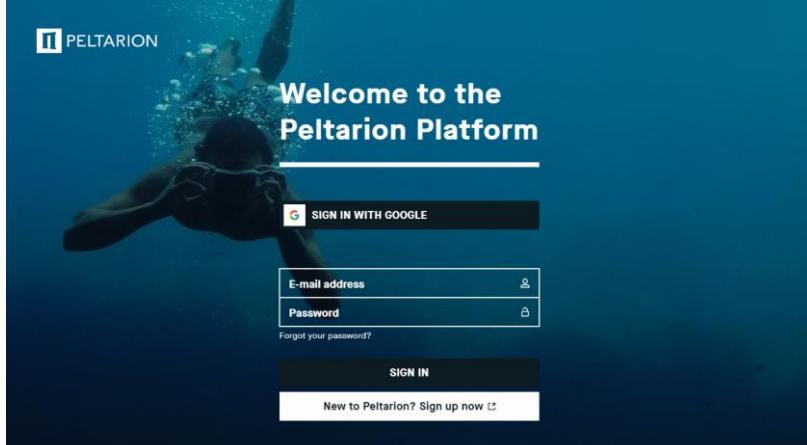
Below are sample images from the various classes in the dataset. Note that the unknown class contains images of cars that are in either a pristine or wrecked condition.

Each collected image represents one car with one specific type of damage. This means that the dataset can be used to solve a single-label classification problem.



3) Create new Project

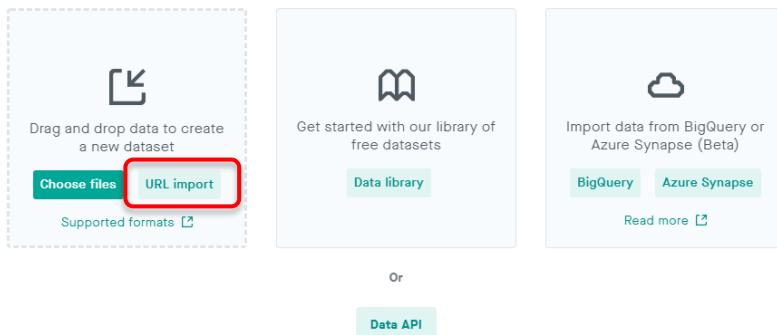
- 1) Sign up for an account and login at <https://platform.peltarion.com/login>



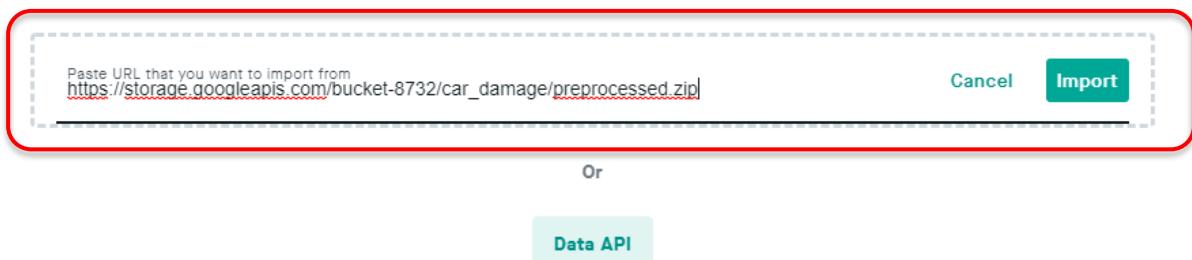
- 2) Create a new project by clicking on New Project and give your project a new. Click **Create** when done.

A screenshot of the Peltarion Platform interface. At the top, there's a navigation bar with 'Projects' selected. Below it is a search bar with placeholder text 'Search and filter projects...'. A red box highlights the 'New project' button, which is located in a teal-colored bar. The main area shows a list of existing projects: 'Book Genre Classification' (2 hours ago), 'My first image classifier' (5 hours ago), 'My First Project' (5 hours ago), and 'MNIST (sample project)' (5 hours ago). To the right of the project list is a 'Khee Wei Seow' profile card. Below the profile card is a 'Running' section with a progress bar and a 'New project' dialog box. The dialog box has a red border and contains fields for 'Project name' (set to 'Car Assessment Classifier') and 'Description'. At the bottom of the dialog are 'Cancel' and 'Create' buttons, with the 'Create' button also highlighted by a red box.

- 3) Navigate to the **Datasets** canvas if you are not automatically brought there. Click on **URL import**.

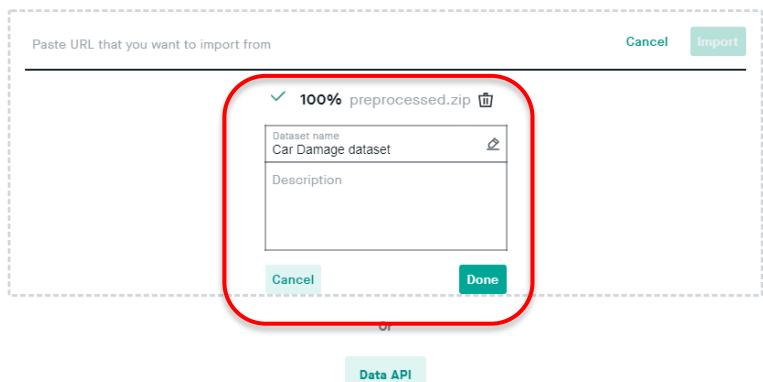


- 4) Copy the link and paste the link https://storage.googleapis.com/bucket-8732/car_damage/preprocessed.zip to the **Import data from URL** box. The zip includes the whole dataset.

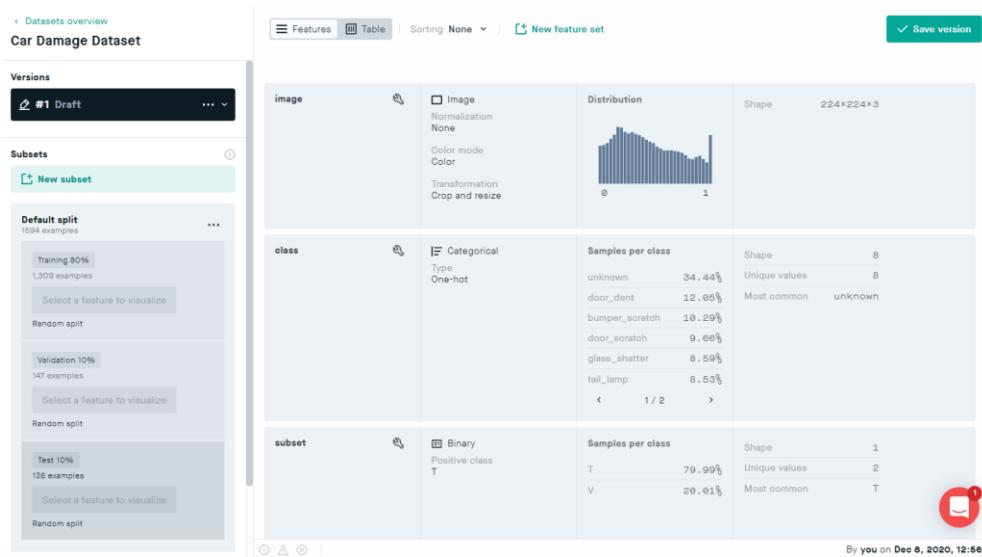


- 5) Click on **Import**.

- 6) When done, name the dataset **Car damage dataset** and click **Done**.



- 7) You will see the summaries of the dataset displayed as shown below.



4) Create subsets of the car damage dataset

The subset column, containing a T or a V, indicates if the row should be used for training or validation. The split between training and validation data is approximately 80% and 20%. This column was created during the pre-processing of the raw data.

Even though it is possible to use the default subsets created by the platform when you upload the data, it is more advantageous to create a conditional split based on the subset column. For this dataset, there is no separate labelled test subset, in case you want to analyse the performance of the deployed model outside the platform. Instead, you can compare the model predictions with the ground truth provided with the predefined validation subset.

- 1) Click **New subset** and name the training subset **Training**. Then click **Add conditional filter** and set **Feature to subset**, **Operator** to **is equal to**, and **Value** to **T**. The details are shown below. Click **Create subset** to create this subset.

New subset

Create a subset by applying filters on the dataset. Subsets contains some, or all, of the data in the dataset. Limiting the amount of data used can make experimentation faster or help you better understand how well your model generalize.

In step 2 (optional), split the subset into two or three non-overlapping subsets. We recommend using splits to create training, validation and test subsets. [Read more...](#)

Subset name: Training

1. Filter data **2. Split subset**

Data in use: 79.99%

1. Feature subset: Is equal to: Value: T

+ Add random % filter + Add conditional filter

Previous Create subset Next

- 2) Repeat the procedure for a new **Validation** subset and set **Feature to subset**, **Operator** to **is equal to**, and **Value** to **V**. The details are shown below.

New subset

Create a subset by applying filters on the dataset. Subsets contains some, or all, of the data in the dataset. Limiting the amount of data used can make experimentation faster or help you better understand how well your model generalizes.

In step 2 (optional), split the subset into two or three non-overlapping subsets. We recommend using splits to create training, validation and test subsets. [Read more.](#)

Subset name: Validation

1. Filter data

Data in use: 20.01%

1. Feature subset: Is equal to V

+ Add random % filter + Add conditional filter

2. Split subset

Previous Create subset Next

- 3) We have now created a dataset ready to be used in the platform. Click **Save version**.

[Datasets overview](#)

Car Damage Dataset

Validation 10%
147 examples
Select a feature to visualize
Random split

Test 10%
136 examples
Select a feature to visualize
Random split

Training 1,275 examples
Filters: subset is equal to T
Select a feature to visualize

Validation 310 examples
Filters: subset is equal to V
Select a feature to visualize

Features Table Sorting None New feature set

Filtered on subset Training 80% - 1309 examples [Clear filter](#)

image	Image Normalization: None Color mode: Color Transformation: Crop and resize	Distribution	Shape: 224x224x3

class	Categorical Type: One-hot	Samples per class	Shape: 8 Unique values: 8 Most common: unknown
		unknown: 33.23%, door_dent: 12.15%, bumper_scratch: 10.62%, door_scratch: 10.16%, glass_shatter: 8.56%, bumper_dent: 8.48%	

subset	Binary Positive class: T	Samples per class	Shape: 1 Unique values: 2 Most common: T
		T: 80.60%, V: 19.40%	

By you on Dec 8, 2020, 12:56

✓ Save version

5) Create a new experiment

- 1) Click on **Use in new experiment** to create a new experiment using this dataset.
- 2) Name the experiment in the **Experiment wizard**.
- 3) Make sure that the **Car damage dataset** is selected in the **Dataset** tab along with standalone subsets, Training subset and Validation subset as shown below. Click **Next** to continue.

Experiment wizard

Experiment name
Experiment 1

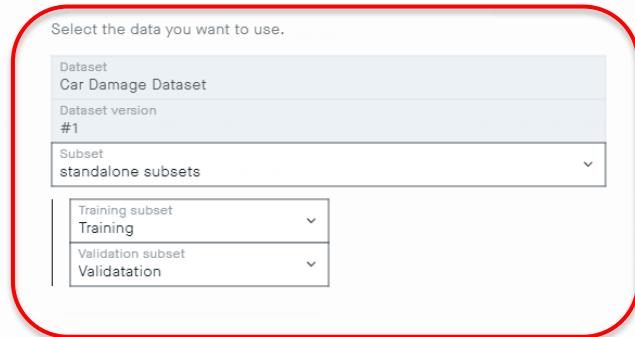
1. Dataset **2. Input(s) / target** **3. Snippet** **4. Weights**

Select the data you want to use.

Dataset
Car Damage Dataset
Dataset version
#1
Subset
standalone subsets

Training subset
Training
Validation subset
Validation

Create custom experiment **Next**



- 4) In the Input(s) / target tab, check that the following inputs are pre-populated. Click **Next** to move to the next step.

Experiment wizard

Experiment name
Experiment 1

1. Dataset **2. Input(s) / target** **3. Snippet** **4. Weights**

Select the input(s) and target features you want to use.

Input(s)

- Search feature
- image (224x224x3)
- class (8)
- subset (1)

Target

- image (224x224x3)
- class (8)
- subset (1)

Previous **Create custom experiment** **Next**

- 5) On the **Snippet** tab and select the **EfficientNet B0** snippet. Click **Next** to continue.

Experiment wizard

Experiment name
Experiment 1

1. Dataset **2. Input(s) / target** **3. Snippet** **4. Weights**

Select problem type and snippet.

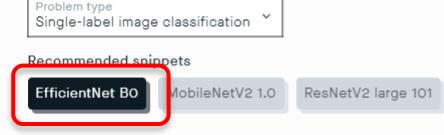
Problem type
Single-label image classification

Recommended snippets

EfficientNet B0 MobileNetV2 1.0 ResNetV2 large 101

▼ Other snippets

Previous **Create custom experiment** **Next**



- 6) On the **Weights** tab. Select **ImageNet** for pretrained data. Click **Create** to continue.

Experiment wizard

Experiment name
Experiment 1

1. Dataset 2. Input(s) / target 3. Snippet **4. Weights**

Choose how you want to initialize the weights.

Weights
ImageNet

Weights trainable (all blocks)

Trained on the ImageNet classification dataset, with 1.2 million images from 1000 classes.
 Browse the dataset at image-net.org

[Third-party terms apply](#)

[Previous](#) [Create custom experiment](#) **Create**

The EfficientNet B0 blocks will be added to the Modeling canvas. You can expand and collapse the EfficientNet B0 group at any time by clicking + or -.

Experiments overview

Experiment 1

Created - just now

[Duplicate](#) [Evaluate](#)

Build **Settings**

Run settings

- Batch size: 32 Epochs: 100
- Data access seed: 8689970284
- Optimizer: Adam
- Learning rate: 0.001
- β_1 rate: 0.9 β_2 rate: 0.999
- Learning rate schedule: Constant
- Early stopping
- Patience (epochs): 5
- Metric: Validation Loss

Dataset settings

Dataset: Car Damage dataset Dataset version:

Modeling canvas:

```

graph TD
    Input[Input: 224 x 224 x 3] --> EfficientNet[EfficientNet B0  
3 blocks]
    EfficientNet --> Dense[Dense: 8]
    Dense --> Target[Target: class]
    
```

Run (3.07 GB)

6) Run the experiment

- 1) Click the **Settings** tab and change the **Learning rate** to 0.0005 and **Epoch** to 5. Click **Run** to start the training. The training may take a long time depending on complexity and the number of epochs. You can grab a coffee or tea at this point in time. For this case, we should take around 8 mins.

Experiment 1

Created -1 secs

Duplicate **Evaluate**

Build **Settings**

Run settings

Batch size	32	Epochs	5
Data access seed	-184292982		
Optimizer	Adam		

Learning rate	0.0005
lrn rate	0.9
lrn rate schedule	Constant

Early stopping

Patience (epochs)	5
Metric	Validation Loss

7) Analyse the experiment

- 1) Go to the **Evaluation** view. Click on **Predictions inspection**.

Projects / Demo car assessment

Search and filter experiments...

New experiment

Experiment 1 Completed 11 mins ago Modeling

Subset Validation **Checkpoint** Epoch: 3 (best) **Inspect**

Currently inspecting Validation / Epoch: 3 (best)

Cells Percentage

Actual	bump...	head...	door...	glass...	bump...	door...	tall...	unknown
Actual	bumper...	75.6	25.0	7.4	3.7	3.7	7.4	
Actual	head...	77.8	7.4	22.6				
Actual	door...	77.4	77.8	3.7	11.1			
Actual	glass...	3.7	3.7	77.8	3.7			
Actual	bump...	7.7	3.8	88.6	7.7			
Actual	door...	2.6	7.7	2.6	79.5	7.7		
Actual	tall...	3.7	3.7	11.1	74.1	7.4		
Actual	unknown	1.8		5.5	2.7	98.8		
Predicted	bump...	75.6	25.0	7.4	3.7	3.7	7.4	
Predicted	head...	77.8	7.4	22.6				
Predicted	door...	77.4	77.8	3.7	11.1			
Predicted	glass...	3.7	3.7	77.8	3.7			
Predicted	bump...	7.7	3.8	88.6	7.7			
Predicted	door...	2.6	7.7	2.6	79.5	7.7		
Predicted	tall...	3.7	3.7	11.1	74.1	7.4		
Predicted	unknown	1.8		5.5	2.7	98.8		

Actual

Predicted

Actual

Image

Encoding

Normalisation

None

bumper_dent

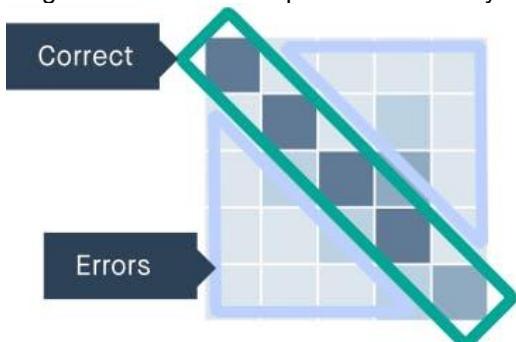
unknown

unknown

unknown

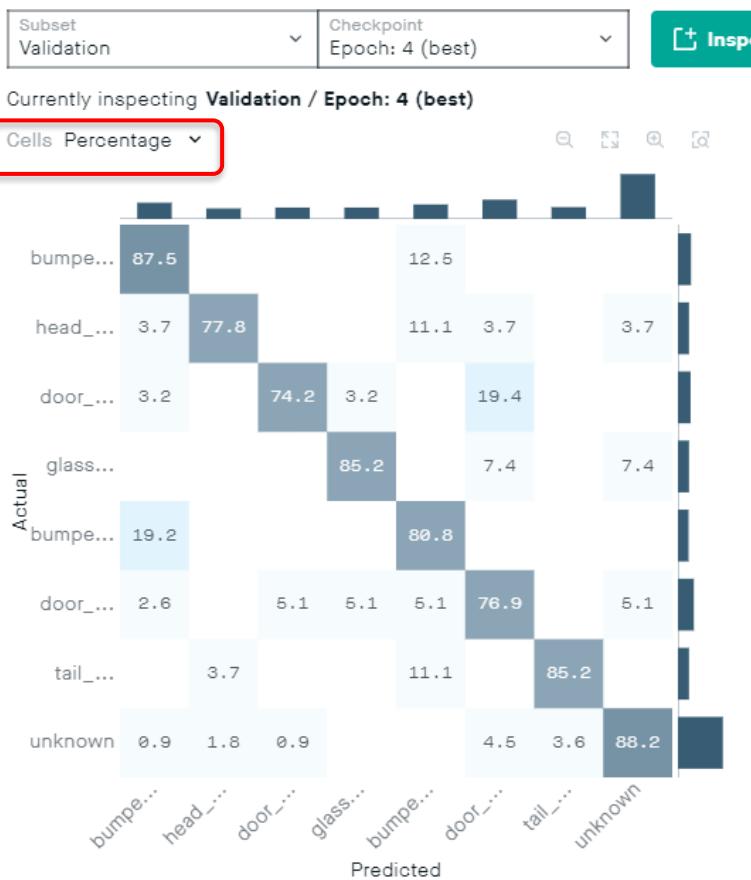
door_scratch

- 2) Since the model solves a classification problem, a confusion matrix is displayed. The top-left to bottom-right diagonal shows correct predictions. Everything outside this diagonal are errors.



- 3) Note that metrics are based on the validation subset which only consists of 20% of the original dataset.

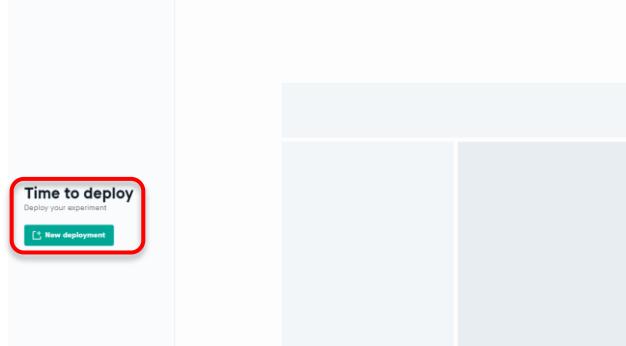
- 4) Click the dropdown next to **Cells** and select **Percentage**. The normalized values that are now displayed correspond to the recall for each class.



The recall values clearly indicate that the model has learned the features in the images.
 Ref: <https://peltarion.com/knowledge-center/documentation/glossary#R>

8) Deploy the trained model

- 1) In the **Deployment** view click **New deployment** as shown below.



- 2) In the **Create deployment** window, select the experiment, **Checkpoint** model and set a **name** for this deployment. Click **Create** to continue.

- 3) Once the deployment is ready, you will see a summary as follow.

Deployment Info

Experiment: Experiment 1
Checkpoint: Epoch: 4 (best)
Date deployed: 2020-09-17
Deployment ID: f51e9942-fc4-40aa-9d5f-7fafba7b9686

Parameters

Input
Feature: image, Type: image (224x224x3), Name: image

Output
Feature: class, Type: categorical (8), Name: class

Test deployment

Format: CURL
URL: https://a.azure-eu-west.platform.peltarion.com/deployment/f51e9942-fc4-40aa-9d5f-7fafba7b9686/forward
Token: 71a8b28e-0dc8-4b23-8069-e0442bfc716b

Input example:
curl -X POST -F "image=VALUE" -u "71a8b28e-0dc8-4b23-8069-e0442bfc716b" https://a.azure-eu-west.platform.peltarion.com/deployment/f51e9942-fc4-40aa-9d5f-7fafba7b9686/forward

Output example:

- Click the **Enable** switch to deploy the experiment.

9) Test the classifier in a browser

- Let's test your model. Click the **Test deployment** button, and you'll open the **Image & Text classifier API tester** with all relevant data copied from your deployment.



Image & Text Classifier

API tester

Input datatype
 Image Text

Output datatype
 Categorical Binary

Setup

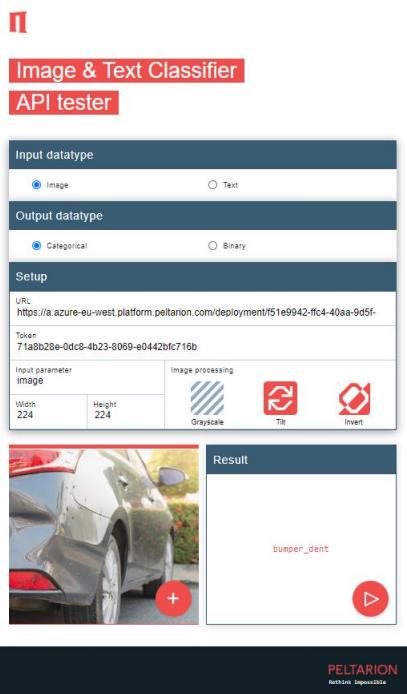
URL: https://a.azure-eu-west.platform.peltarion.com/deployment/f51e9942-fc4-40aa-9d5f-7fafba7b9686/forward
Token: 71a8b28e-0dc8-4b23-8069-e0442bfc716b

Input parameter: image
Width: 224 Height: 224
Image processing: Grayscale, Fit, Invert

Result
Press to get result...

PELTARION

- Drag a test image onto the image box on the left and click **Play** to get a prediction.



10) Next Steps

The next steps could be to try to run the project using different models to see if that improves the result or maybe change the learning rate or training epochs.

Activity 4 – Creating a Sentiment Analyser

In this activity, we will learn:

- We will solve a text classification problem using BERT (Bidirectional Encoder Representations from Transformers). The input is an IMDB dataset consisting of movie reviews, tagged with either positive or negative sentiment – i.e., how a user or customer feels about the movie.

Ref:

- 1) Deploy an operational AI model (<https://peltarion.com/knowledge-center/documentation/tutorials/deploy-an-operational-ai-model>)
 - 2) Embeddings - If you want it could be a good idea to read about word embeddings, which is an important concept in NLP (Natural Language Processing). For an introduction and overview of different types of word embeddings, check out the links below:
 - 3) Get Busy with Word Embeddings — An Introduction (<https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction>)
 - 4) Introduction to Embedding in Natural Language Processing (<https://www.datascience.com/blog/embedding-in-natural-language-processing>)
- Text embedding block (<https://peltarion.com/knowledge>)

1) The Problem

Text classification aims to assign text, e.g., tweets, messages, or reviews, to one or multiple categories. Such categories can be the author's mood: is a review positive or negative?

We will learn how to build and deploy a model based on BERT.

BERT pushed the state of the art in Natural Language Processing (NLP) by combining two powerful technologies:

- a. It is based on a deep Transformer network. A type of network that can process efficiently long texts by using attention.
- b. It is bidirectional. Meaning that it takes into account the whole text passage to understand the meaning of each word.

2) Dataset – The Large Movie Review Dataset v1.0

The raw dataset contains movie reviews along with their associated binary category: positive or negative. The dataset is intended to serve as a benchmark for sentiment classification

The core dataset contains 50,000 reviews split evenly into a training and test subset. The overall distribution of labels is balanced, i.e., there are 25,000 positive and 25,000 negative reviews.

The raw dataset also includes 50,000 unlabelled reviews for unsupervised learning, these will not be used in this tutorial.

In the entire collection, no more than 30 reviews are allowed for any given movie because reviews for the same movie tend to have correlated ratings.

In the labelled train/test sets, a negative review has a score that is less or equal to 4 out of 10, and a positive review has a score that is higher than 7. Reviews with more neutral ratings are not included in the dataset.

Each review is stored in a separate text file, located in a folder named either "positive" or "negative."

Note: For more information about the raw dataset, see the ACL 2011 paper "Learning Word Vectors for Sentiment Analysis".

Written by Maas, A., Daly, R., Pham, P., Huang, D., Ng, A. and Potts, C. (2011). Learning Word Vectors for Sentiment Analysis: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. [online] Portland, Oregon, USA: Association for Computational Linguistics, pp.142–150. Available at: <http://www.aclweb.org/anthology/P11-1015>.

For this activity, the dataset that we will upload in this activity has been preprocessed so that all the reviews and their respective sentiments are stored in a single CSV file with two fields, “review” and “sentiment.”

The review text may include commas, which will be interpreted as a field delimiter on the platform. To escape these commas, the text is surrounded by double-quotes.

The processed dataset only includes the training data.

3) Create a new project

- 1) Sign up for an account and login at <https://platform.peltarion.com/login>
- 2) Create a new project by clicking on New Experiment and give your project a name. In this case, we can name it Movie Review Sentiments.

The screenshot shows the Peltarion Platform's 'Projects' interface. At the top, there is a search bar labeled 'Search and filter projects...' and a teal button labeled '+ New project'. Below this, a project titled 'My first image classifier' is listed, created 3 hours ago. The main area displays the details for a project named 'Khee Wei Seow': Compute time (9 hours available), Storage (20 GB available), and Members (1/1). Below this, a modal window titled 'New project' is open, showing the project name 'Movie Review Sentiments' in the input field. The modal also has a 'Description' section and 'Cancel'/'Create' buttons at the bottom.

4) Add a new dataset

- 1) After creating the project, you will be taken to the **Datasets** view, where you can import data.

The screenshot shows the 'Datasets' view of the Peltarion Platform. It features three main sections for importing data: 1) 'Drag and drop data to create a new dataset' with buttons for 'Choose files' and 'URL import'. 2) 'Get started with our library of free datasets' with a 'Data library' button highlighted with a red box. 3) 'Import data from BigQuery or Azure Synapse (Beta)' with buttons for 'BigQuery' and 'Azure Synapse'. A 'Read more' link is also present. Below these sections, there is an 'Or' label and a 'Data API' button.

- 2) Click the **Data library** button and look for the **IMDB - tutorial data** dataset in the list. Click on it to get more information.

Data library

Search and filter...

Cali House - tutorial data
 The Cali House dataset combines data from two sources. The tabular data from the California 1990 Census, which give...

Most downloaded books - tutor...
 Over 100000 text passages from public domain books, with extra...

IMDB - tutorial data
 This dataset contains textual movie reviews from IMDB users, together...

Car Damage - tutorial data
 Photos of damaged cars, with the type of damage (glass shatter, bumper dent, door dent, etc.)

Bank marketing - tutorial data

- 3) If you agree with the license, click **Accept and import**. This will import the dataset in your project, and you will be taken to the dataset's details where you can edit features and subsets.

IMDB

About this dataset
 This dataset contains textual movie reviews from IMDB users, together with the rating (simplified as positive or negative) that the user gave to the movie.

Inspiration
 Use this dataset to predict a simple positive or negative category from paragraph-sized text data.

Information	
Creator	
Features	Review, Sentiment
Rows	25 000
Size	13 MB
Categories	Text, Classification
Date added	2020-01-28
License	License
Tutorial	Knowledge center

Accept and import

5) Text Encoding

- 1) Click on the **Table** button, click the **Review** column and check the following in the **Feature Settings**.
- Encoding to Text

The screenshot shows a machine learning interface with two main sections: 'Shape -' and 'Shape 1'. In the 'Shape -' section, there is a histogram-like visualization. Below it, a table row for 'review' is shown with the following details: Label 'review', Encoding 'Text', and Type 'One-hot'. A red box highlights a modal window titled 'Feature Settings' for 'review'. The modal shows the 'Label' as 'review' and the 'Encoding' as 'Text'. In the 'Shape 1' section, there is another table row for 'sentiment' with the following details: Label 'sentiment', Encoding 'Binary', and Positive class 'positive'. The 'negative' column contains a sample text from the IMDB dataset.

Note:

Subsets of the dataset

On the left, you will see the subsets. All samples in the dataset are by default split into 10% testing, 10% validation and 80% training subsets. Let us change these values to 20% validation and keep 80% for training.

- 2) Click on ..., Edit to start editing the subset

The screenshot shows the 'IMDB' dataset overview. On the left, there is a sidebar for 'Subsets' with a 'New subset' button. Below it, three splitting options are listed: 'Default split' (250,000 examples), 'Validation 10%' (2,469 examples), and 'Test 10%' (2,407 examples). The 'Edit' button for the 'Default split' is highlighted with a red box. The main area shows the dataset table with two rows of text samples. The first row is labeled 'negative' and the second row is labeled 'positive'. A red circle with a white exclamation mark is in the bottom right corner of the main area.

- 3) Click on 2. Split subset, change the distribution as shown below.

Edit subset

Create a subset by applying filters on the dataset. Subsets contains some, or all, of the data in the dataset. Limiting the amount of data used can make experimentation faster or help you better understand how well your model generalizes.

In step 2 (optional), split the subset into two or three non-overlapping subsets. We recommend using splits to create training, validation and test subsets. [Read more.](#)

Subset name: Default split

1. Filter data **2. Split subset**

Type of split: Stratified

Data in use: [Progress bar]

● Size (percentage) 80	▲ Name Training 80%
● Size (percentage) 20	▲ Name Validation 20%

+ Add test set

Previous Edit

4) Save the dataset by clicking on **Save Version**. The view will then change to **Use in new experiment**.

Datasets overview

IMDB

Versions

#1

Subsets

New subset

Default split

25,000 examples

Training 80%
20,124 examples

Select a feature to visualize

Random split

Validation 20%
4,876 examples

Select a feature to visualize

Random split

Features Table Sorting None New feature set Go to draft Use in new experiment

Shape 1

review Encoding Text

sentiment Encoding Binary Positive class positive

1 Hubert Selby Jr. gave us the book "Requiem For A Dream" and co-wrote the screenplay to Aronofsky's movie of it. That movie succeeded on every level by delivering an intimate, and unbiased portrait of the horrors of the characters lives and their addictions. It's a movie that is both deeply moving and horrific. It follows the multiple characters living sad lives, but it hardly does them the same justice Aronofsky did. The film seems laughably anti-gay at times. Especially when in the film homosexuality equals death. One gay character gets stoned, is launched skyward by a speeding car, and lands dead on the pavement. Another is crucified and still more are simply beat up. Another exaggerated piece of shock value, that might actually have...

negative

2 There are very few performers today who can keep me captivated throughout an entire film just by their presence. One of those few is Judy Davis, who has built a successful career out of creating characters that are headstrong in attitude but very vulnerable at heart. She takes roles that most other performers would treat melodramatically and adds a fiery, deeply emotional intensity that pulls attention away from everything else on the screen. Her skills are well displayed in "High Tide," a film that matches her up a second time with director Gillian Armstrong, who gave Davis her first major success with "My Brilliant Career." In that film, Davis played a young woman who was determined to make it in the world, despite the scoffs...

positive

6) Design a text classification model with the BERT model

- 1) Click on **Use in new experiment**.
- 2) Make sure that the **IMDB** dataset is selected in the **Experiment wizard** with the Subset, training and validation selected as shown below. Click **Next**.

Experiment wizard

Experiment name
Experiment 1

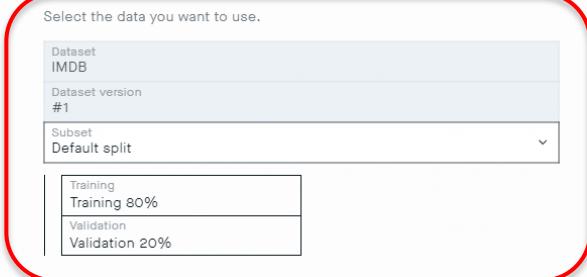
1. Dataset 2. Input(s) / target 3. Snippet 4. Weights

Select the data you want to use.

Dataset
IMDB
Dataset version
#1
Subset
Default split

Training
Training 80%
Validation
Validation 20%

[Create custom experiment](#) **Next**



- 3) In the **Inputs(s) / target** tab, check that the **Input(s)** and **Target** are set as follows:

Experiment wizard

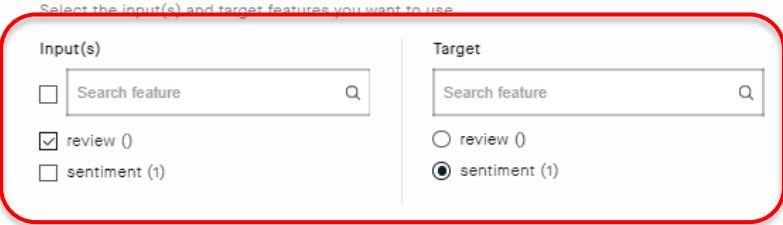
Experiment name
Experiment 1

1. Dataset **2. Input(s) / target** 3. Snippet 4. Weights

Select the input(s) and target features you want to use.

Input(s)	Target
<input type="checkbox"/> Search feature	<input type="checkbox"/> Search feature
<input checked="" type="checkbox"/> review (0)	<input type="radio"/> review (0)
<input type="checkbox"/> sentiment (1)	<input checked="" type="radio"/> sentiment (1)

Previous [Create custom experiment](#) **Next**



- 4) In the Snippet tab, set the Problem type to Binary text classification since we are predicting positive or negative review. Use English BERT uncased snippet.

Experiment wizard

Experiment name
Experiment 1

1. Dataset 2. Input(s) / target **3. Snippet** 4. Weights

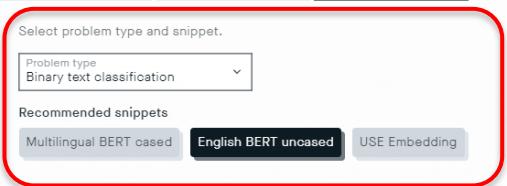
Select problem type and snippet.

Problem type
Binary text classification

Recommended snippets

Multilingual BERT cased **English BERT uncased** USE Embedding

Previous [Create custom experiment](#) **Create**

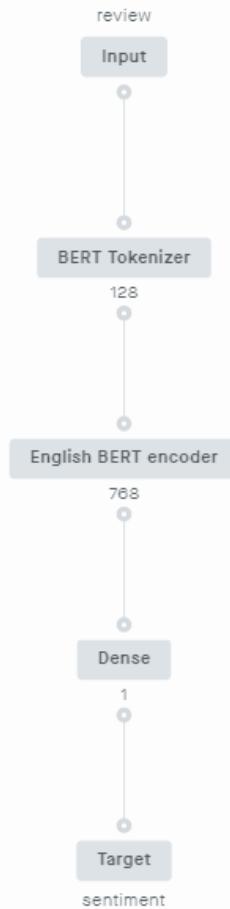


The BERT English uncased snippet includes the whole BERT network. The BERT block implements the base version of the BERT network. It is composed of 12 encoding layers from a Transformer network, each layer having 12 attention heads. The total number of parameters is 110 million. The snippet allows you to use this massive network with weights pre-trained to understand the text.

The BERT snippet includes:

- a. An Input block.
- b. A BERT Encoder block with pre-trained weights which gives BERT a general understanding of English. The BERT encoder block looks at the input sequence as a whole, producing an output that contains an understanding of the sentence. The block outputs a single vector of 768 size.
- c. A label block with pre-trained weights.
- d. A Dense block that is untrained.
- e. A Target block.

5) Click **Create** to create the experiment and the prepopulated BERT model will appear in the **Modeling** canvas.



6) Click the Settings tab and check that:

- a. **Batch Size** is 2. If you set a larger batch size you will run out of memory.
- b. **Epochs** is 2. Training takes a long time, so don't train for too long the first time when you check if your model is good.
- c. **Learning rate** is 0.00001 (4 zeros). To avoid catastrophic forgetting.

The screenshot shows the 'Experiment 2' settings page. The 'Settings' tab is active. In the 'Run settings' section, the 'Batch size' and 'Epochs' fields are both set to 2. The 'Learning rate' field is set to 0.00001. The 'Optimizer' dropdown is set to Adam. The 'Learning rate schedule' is set to Triangle schedule. The 'Warm-up epochs' is set to 0.5 and 'Decrement per epoch' is set to 0.000008. There is a checkbox for 'Early stopping'.

7) Click **Run** to start training the model. **Note: This experiment will take 2 hours to complete.**

7) Evaluation

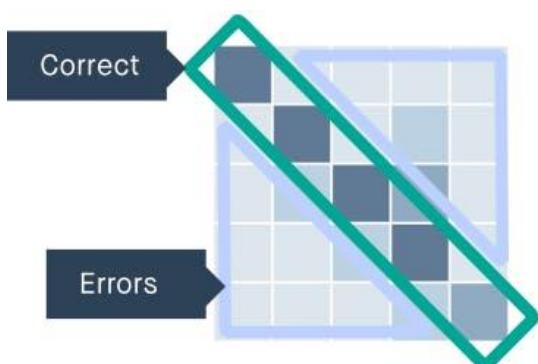
- 1) Navigate to the **Evaluation** view and watch the model train. The training will take quite a long time since BERT is a very large and complex model.

Accuracy

To evaluate the performance of the model, you can look at overall accuracy, which is displayed in the Experiment info section to the right. It should be approximately 85-90%. For comparison, a classifier that would predict the class randomly would have a 50% accuracy, since 50% of reviews in the dataset are positive and 50% are negative.

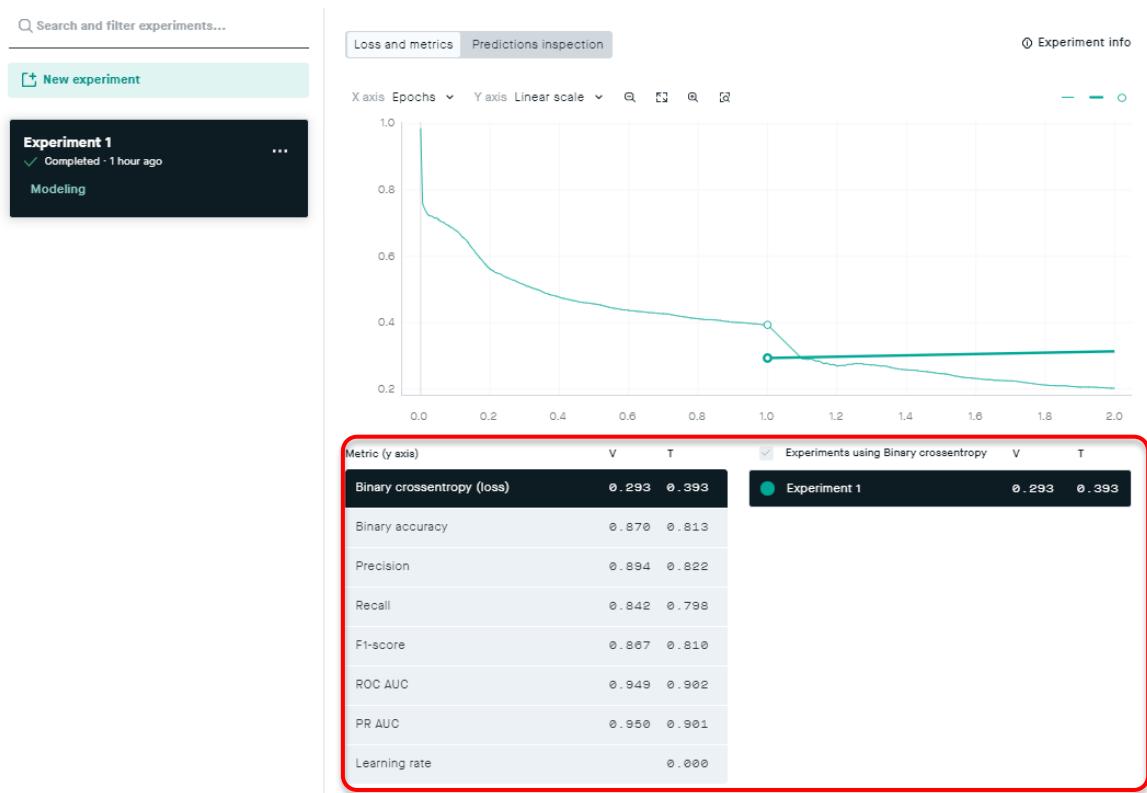
Confusion matrix

Since the model solves a classification problem, a confusion matrix is displayed. The top-left to bottom-right diagonal shows correct predictions. Everything outside this diagonal are errors.

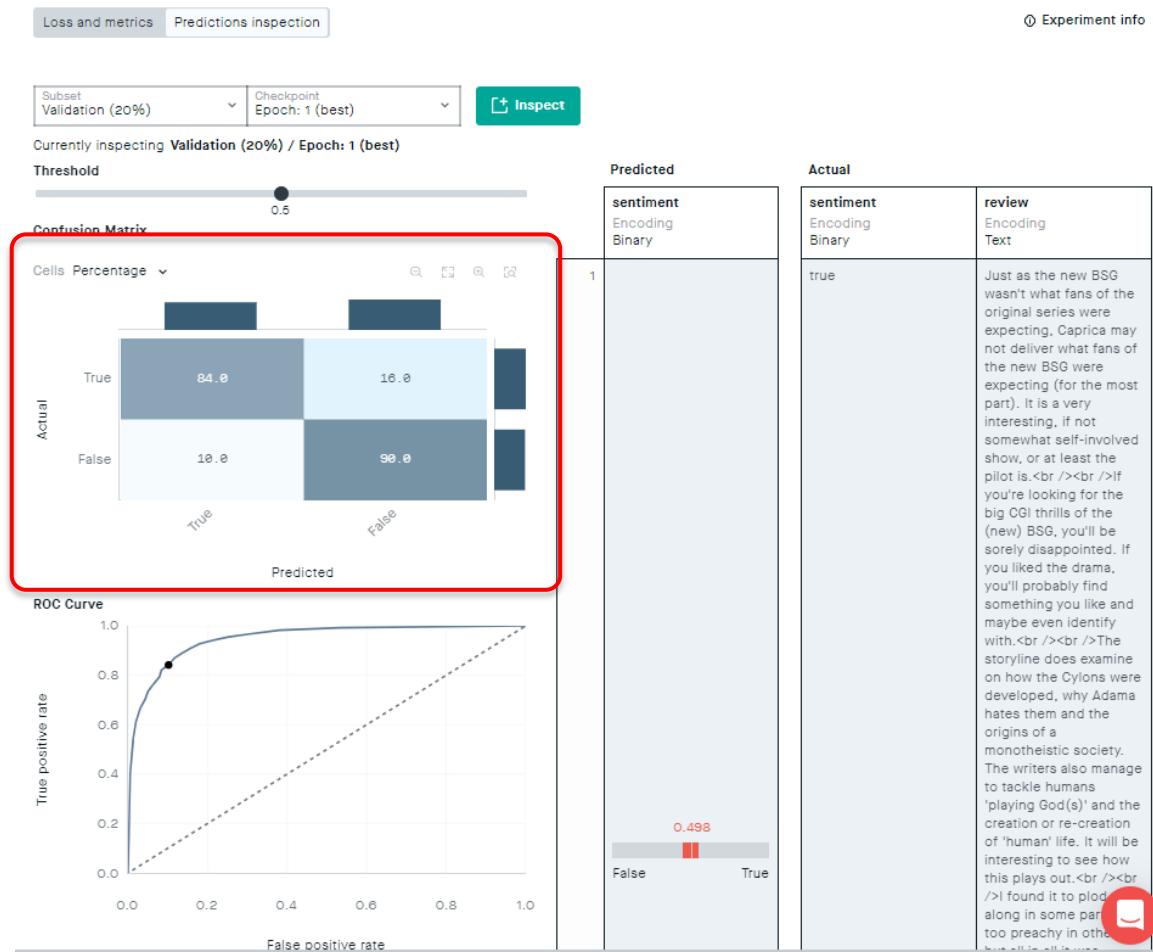


Recall

The recall (<https://peltarion.com/knowledge-center/documentation/glossary>) per class corresponds to the percentage values in the confusion matrix diagonal. You can display the same metric by hovering over the horizontal bars to the right of the confusion matrix. You can also view the precision per class by hovering over the vertical bars on top of the confusion matrix.

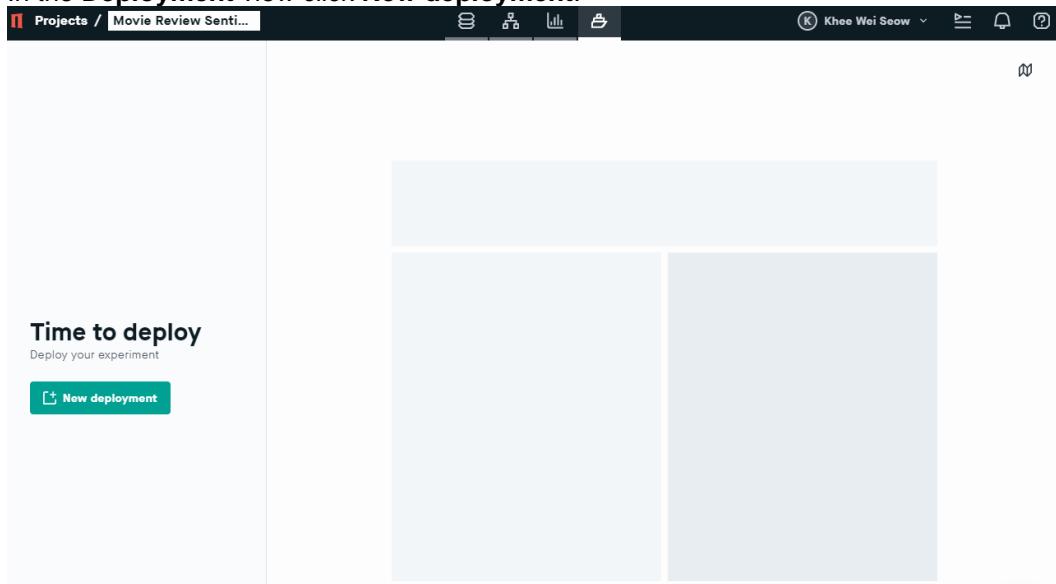


- 2) Navigate to the **Predictions Inspection** view. It will take awhile to create the confusion matrix. When all the examples where processed, you should see the following:



8) Create new deployment

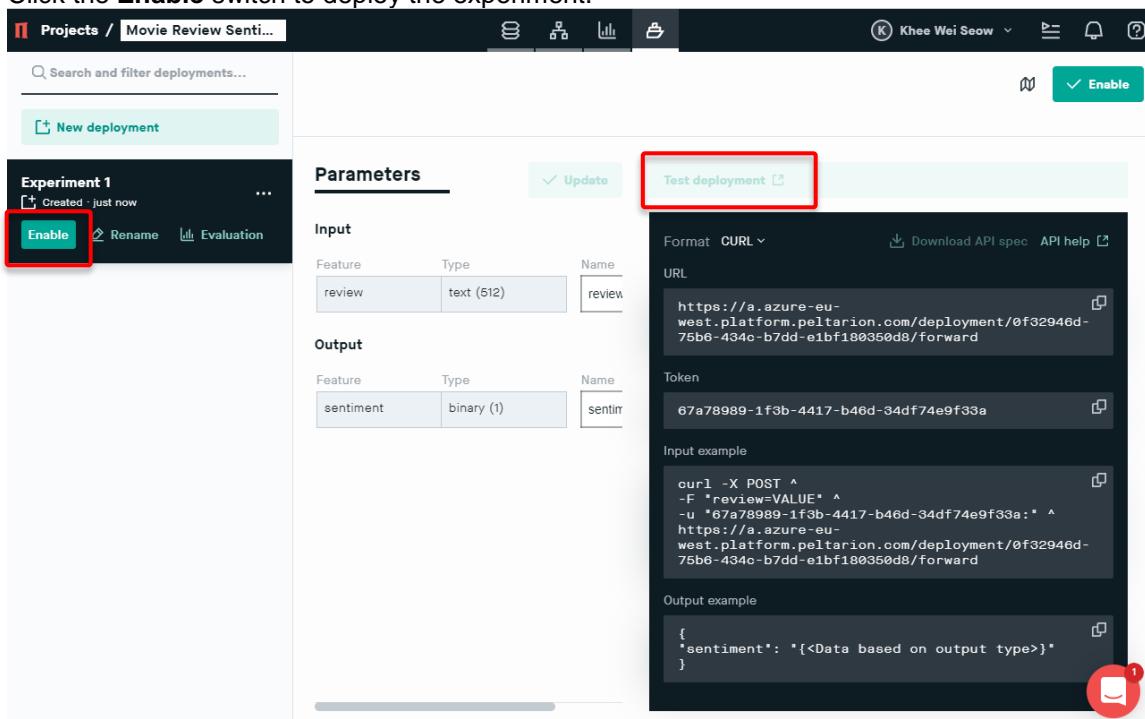
- 1) In the Deployment view click New deployment.



- 2) Select experiment and checkpoint of your trained model to test it for predictions, or enable for business product calls. Both best epoch and last epoch for each trained experiment are available for deployment. Click **Create** to continue.

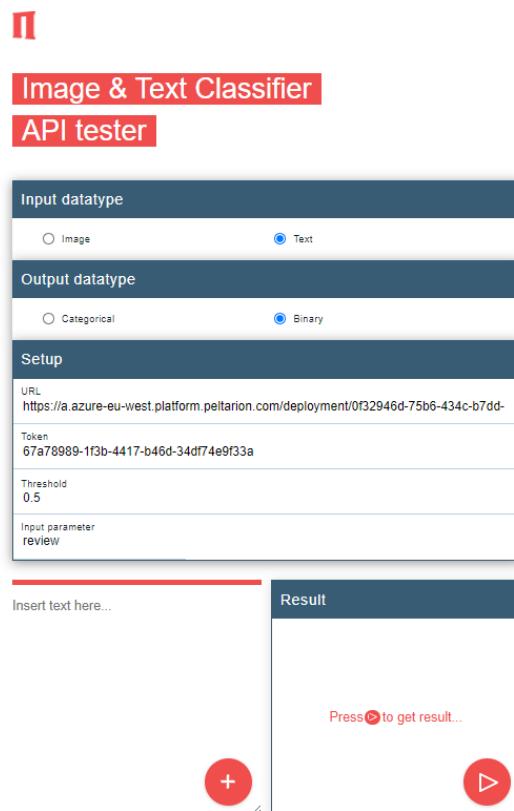


- 3) Click the **Enable** switch to deploy the experiment.



9) Test the text classifier in a browser

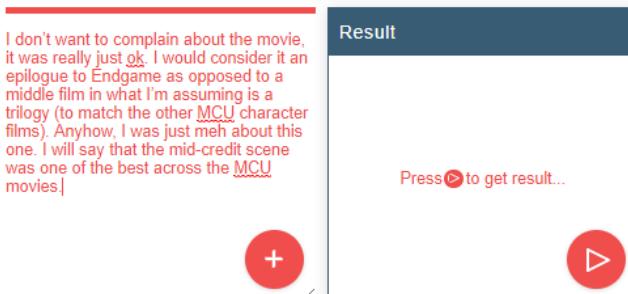
- Let's test your model. Click the **Test deployment** button, and you'll open the **Text classifier API tester** with all relevant data copied from your deployment.



- Now, write your own review, copy the example below or simply copy a recent review from, e.g., IMDB.

Example:

I don't want to complain about the movie, it was really just ok. I would consider it an epilogue to Endgame as opposed to a middle film in what I'm assuming is a trilogy (to match the other MCU character films). Anyhow, I was just meh about this one. I will say that the mid-credit scene was one of the best across the MCU movies.



- Click **Play**.
- [Optional] To see what an actual request from the application and the response from the model may look like, you can run the example CURL command that is provided in the Code examples section of the Deployment view. Replace the VALUE parameter with review text and run the command in a terminal.

Parameters

Input

Feature	Type	Name
review	text (512)	review

Output

Feature	Type	Name
sentiment	binary (1)	sentir

Test deployment

Format: CURL ▾ [Download API spec](#) [API help](#) ▾

URL: <https://a.azure-eu-west.platform.peltarion.com/deployment/0f32946d-75b6-434c-b7dd-e1bf180350d8/forward>

Token: 67a78989-1f3b-4417-b46d-34df74e9f33a

Input example:

```
curl -X POST ^  
-F "review=VALUE" ^  
-u "":^  
https://a.azure-eu-west.platform.peltarion.com/deployment/0f32946d-75b6-434c-b7dd-e1bf180350d8/forward
```

Output example:

```
{  
  "sentiment": "{<Data based on output type>}"  
}
```



In a cmd windows:

```
C:\Users\seow_khee_wei>curl -X POST ^  
More? -F "review=A fun brain candy movie...good action...fun dialog. A genuinely good day" ^  
More? -u "":^  
More? https://a.azure-eu-west.platform.peltarion.com/deployment/0f32946d-75b6-434c-b7dd-e1bf180350d8/forward  
{"sentiment":0.95231044}  
C:\Users\seow_khee_wei>
```

10) Next Step

The next steps could be to try to run the experiment with increased epochs and see if that improves the result or maybe change the learning rate.

Activity 5 – [Bonus] Book Genre Classifier

In this activity, we will learn:

- In this activity, we will use the Peltarion Platform to build a model to classify books.

Ref:

- Deploy an operational AI model (<https://peltarion.com/knowledge-center/documentation/tutorials/deploy-an-operational-ai-model>)
- Embeddings - If you want it could be a good idea to read about word embeddings, which is an important concept in NLP (Natural Language Processing). For an introduction and overview of different types of word embeddings, check out the links below:
 - Get Busy with Word Embeddings — An Introduction (<https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction>)
 - Introduction to Embedding in Natural Language Processing (<https://www.datascience.com/blog/embedding-in-natural-language-processing>)
 - Text embedding block (<https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/text-embedding>)

1) The Problem

Text classification aims to assign text, e.g., tweets, messages, or reviews, to one or multiple categories. Such categories can be whether or not a book is considered as science fiction.

We will learn how to build and deploy a model based on BERT.

BERT pushed the state of the art in Natural Language Processing (NLP) by combining two powerful technologies:

- It is based on a deep Transformer network. A type of network that can process efficiently long texts by using attention.
- It is bidirectional. Meaning that it takes into account the whole text passage to understand the meaning of each word.

2) Dataset - CMU book summary dataset

The data comes from the CMU Book Summary Dataset (<http://www.cs.cmu.edu/~dbamman/booksummaries.html>), a dataset of over 16 000 book summaries. For this activity, we wanted a dataset with science fiction book summaries, so we chose to preprocess the data to our task, so it contains book summaries along with their associated binary category: Science Fiction or not.

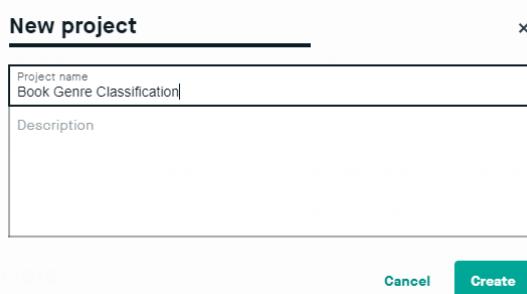
The dataset is intended to serve as a benchmark for sentiment classification. The overall distribution of labels is balanced, i.e., there are approximately 2 500 science fiction and 2 500 non-science fiction book summaries. Each summary is stored in a column, with a science fiction classification of either "yes" or "no".

3) Create a new Project

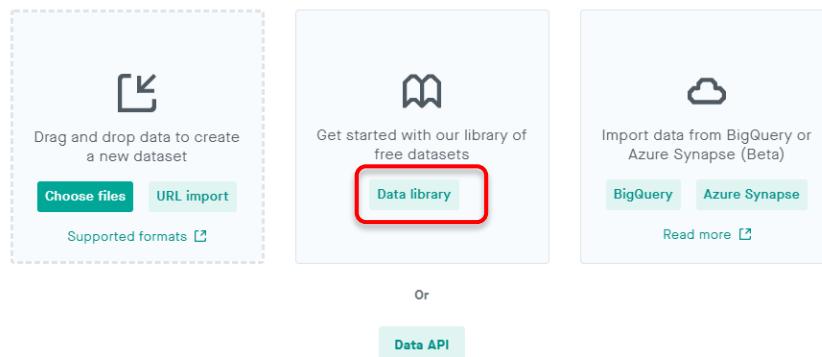
- 1) Sign up for an account and login at <https://platform.peltarion.com/login>

- 2) Create a new project by clicking on **New Experiment** and give your project a name.

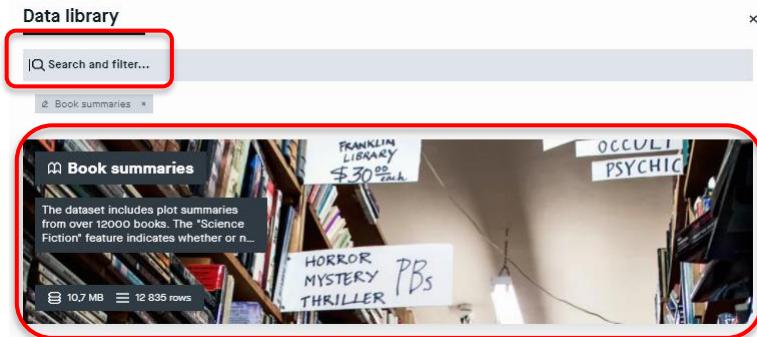
The screenshot shows the Peltarion Platform interface. At the top, there is a dark header bar with a 'Projects' button and a search bar labeled 'Search and filter projects...'. Below the header, a teal button says '+ New project'. To the left, there is a sidebar with a project titled 'My first image classifier' created '3 hours ago'. The main area is titled 'Khee Wei Seow' and displays project details: 'Compute time' (9 hours available), 'Storage' (20 GB available), and 'Members (1/1)' with a user icon. There is also a small circular progress bar icon.



- 3) Navigate to the Datasets if you are not automatically brought there. Click on **Data Library**.



- 4) Type **Book Summaries** in the search box and select the Book Summaries dataset



- 5) Click on **Accept and import**

Book summaries

About this dataset
 The dataset includes plot summaries from over 12000 books. The "Science Fiction" feature indicates whether or not the book is classified as sci-fi (1) or not(0)

Inspiration
 Build your sci-fi/not sci-fi classifier, based on the plot summary.

Information	
Creator	David Bamman
Features	1 Text, 1 Categorical
Rows	12 835
Size	10,7 MB
Categories	Text, Classification
Date added	2020-01-28
License	License

Accept and import

- 6) You will see the summaries of the dataset displayed as shown below.

Book summaries

Versions
 #1 Draft

Subsets
 New subset

Validation (20%)
 Select a feature to visualize
 Random 20% split of the data

Training (80%)
 Select a feature to visualize
 Random 80% split of the data

Summary
T Text (berts)
 Language Model English BERT uncased

Sequence length
 Sequence length 20
 Unique values 658

Science Fiction
Binary
 Positive class 1

Samples per class
 0 50.00%
 1 50.00%

Shape
 1
 Unique values 2
 Most common 0

The dataset is labelled binary, that is, 1 indicates that the book is classified as a science fiction book and 0 is not.

4) Text encoding

- 1) Click on the **Table** button, click the **Summary** column and check the following in the **Feature Settings**.
- Encoding to Text

The screenshot shows a dataset editor interface. At the top, there are tabs for 'Features' and 'Table', with 'Table' being the active tab and highlighted by a red box. Below the tabs, there are two sections: 'Shape -' and 'Shape 1'. The 'Shape -' section contains a histogram and a summary table for 'Summary' with encoding 'Text' and type 'One-hot'. The 'Shape 1' section contains a summary table for 'Science Fiction' with encoding 'Binary' and positive class '1'. In the main content area, there are two rows of data samples. The first sample (row 1) has its details expanded in a modal window titled 'Feature Settings' with the label 'Summary' and encoding 'Text'. The second sample (row 2) is collapsed.

Note:

Subsets of the dataset

On the left, you will see the subsets. All samples in the dataset are by default split into 20% validation and 80% training subsets. Keep these default values in this project.

- 2) Save the dataset by clicking on **Save Version**. The view will then change to **Use in new experiment**.

The screenshot shows the dataset editor after saving a version. The left sidebar displays 'Book summaries' with a 'Versions' section showing '#1' selected and a 'Subsets' section with a 'New subset' button. The main content area shows the dataset details and two samples. The 'Use in new experiment' button at the top right is highlighted with a red box. The first sample (row 1) has its details expanded in a modal window titled 'Feature Settings' with the label 'Summary' and encoding 'Text (berts)'. The second sample (row 2) is collapsed.

5) Design a text classification model

- 1) Make sure that the **Book Summaries** dataset is selected in the Experiment wizard.

Experiment wizard

Experiment name
Experiment 1

1. Dataset **2. Input(s) / target** **3. Snippet** **4. Weights**

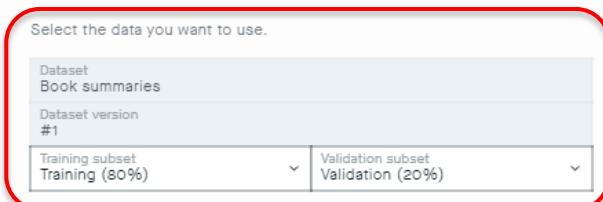
Select the data you want to use.

Dataset
Book summaries
Dataset version
#1

Training subset
Training (80%)

Validation subset
Validation (20%)

Create custom experiment **Next**



- 2) Click **Next** to continue.
- 3) Check that the Inputs and Target are set as follows:

Experiment wizard

Experiment name
Experiment 1

1. Dataset **2. Input(s) / target** **3. Snippet** **4. Weights**

Select the input(s) and target features you want to use.

Input(s)

Search feature

Summary ()

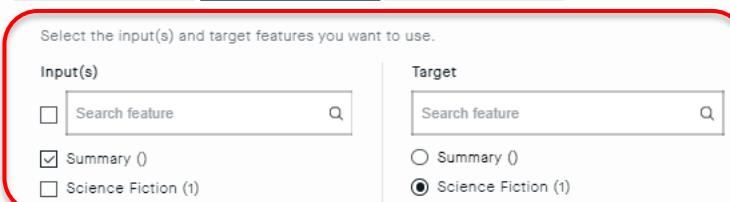
Science Fiction (1)

Target

Summary ()

Science Fiction (1)

Previous **Create custom experiment** **Next**



Click **Next** to continue.

- 4) In the **Snippet** tab, set the Problem type to **Binary text classification** since we are predicting Science fiction or not. Use **English BERT uncased** snippet.

Experiment wizard

Experiment name
Experiment 1

1. Dataset **2. Input(s) / target** **3. Snippet** **4. Weights**

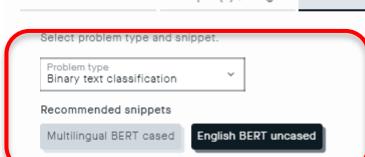
Select problem type and snippet.

Problem type
Binary text classification

Recommended snippets

Multilingual BERT cased **English BERT uncased**

Previous **Create custom experiment** **Create**



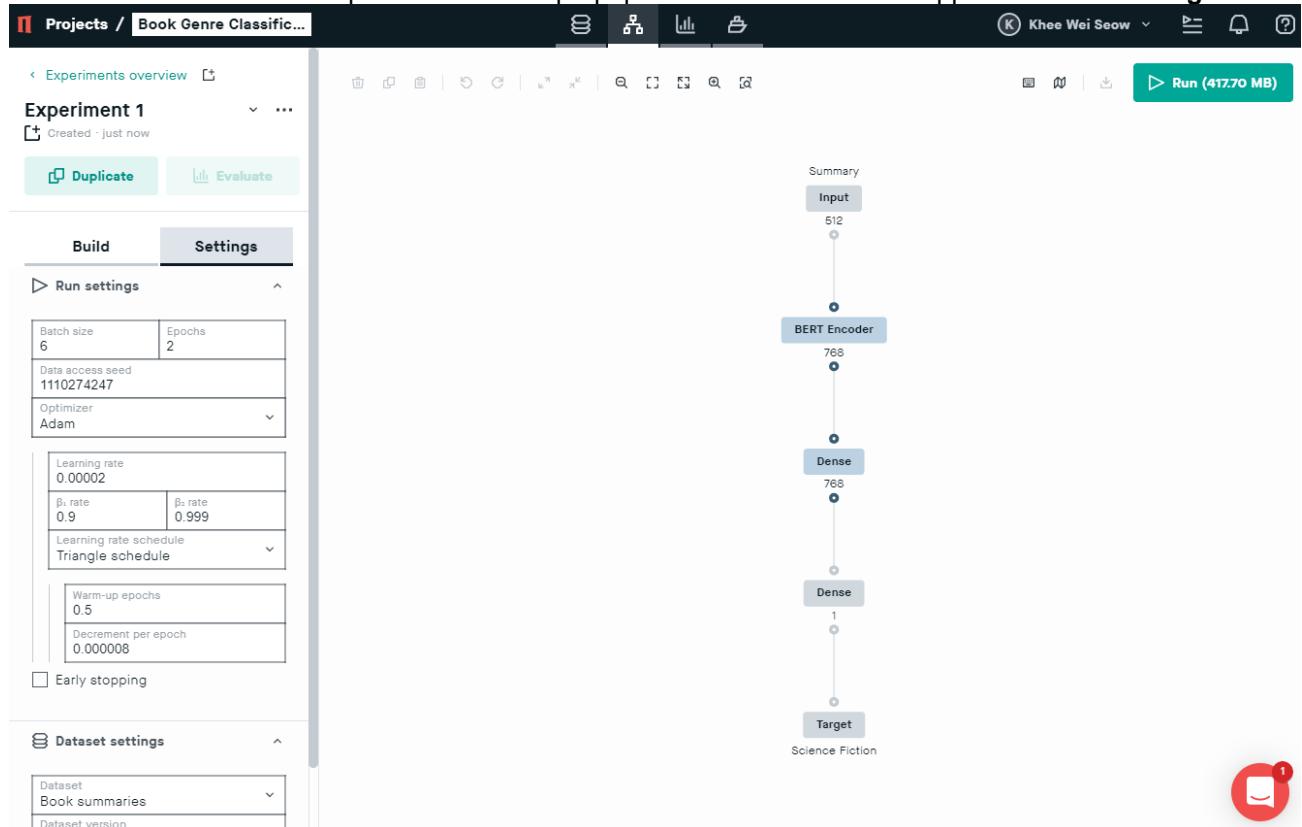
The BERT English uncased snippet includes the whole BERT network. The BERT block implements the base version of the BERT network. It is composed of 12 encoding layers from a Transformer network, each layer having

12 attention heads. The total number of parameters is 110 million. The snippet allows you to use this massive network with weights pre-trained to understand the text.

The BERT snippet includes:

- An Input block.
- A BERT Encoder block with pre-trained weights which gives BERT a general understanding of English. The BERT encoder block looks at the input sequence as a whole, producing an output that contains an understanding of the sentence. The block outputs a single vector of 768 size.
- A label block with pre-trained weights.
- A Dense block that is untrained.
- A Target block.

- 5) Click **Create** to create the experiment and the prepopulated BERT model will appear in the **Modeling** canvas.



- 6) Click the **Settings** tab and check that:

- **Batch Size** is 2. If you set a larger batch size you may run out of memory.
- **Epochs** is 2. Training takes a long time, so don't train for too long the first time when you check if your model is good.
- **Learning rate** is 0.00001 (4 zeros). To avoid catastrophic forgetting.

Experiment 1

Created - 1 secs

Duplicate **Evaluate**

Build **Settings**

Run settings

Batch size	Epochs
2	2

Date created: 1478377733

Optimizer: Adam

Learning rate	0.00001
momentum	0.9
beta_1	0.999

Learning rate schedule: Triangle schedule

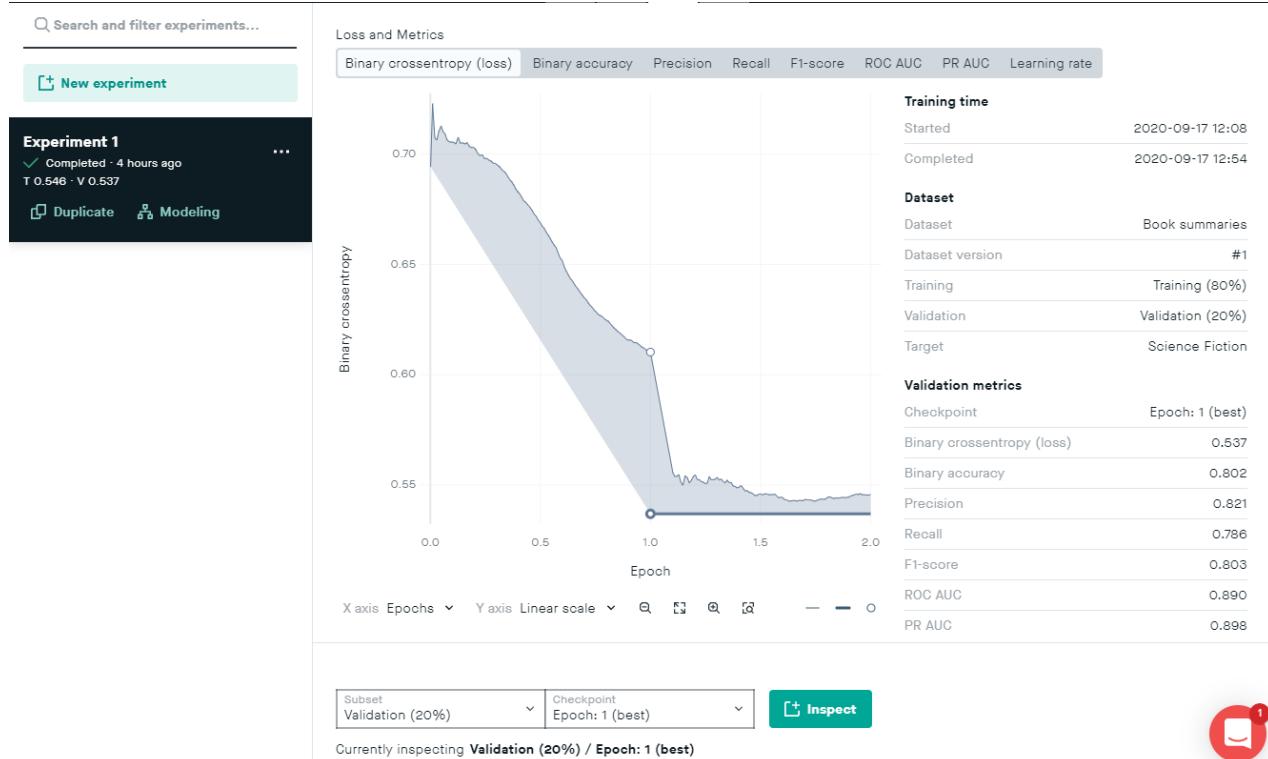
Warm-up epochs	0.5
Decrement per epoch	0.000008

Early stopping

5) Click **Run** to start training the model. With our settings, it took about

6) Evaluation

Navigate to the **Evaluation** view and watch the model train. The training will take quite a long time since BERT is a very large and complex model.



The training loss will decrease for each epoch, but the evaluation loss may start to increase. This means that the model is starting to overfit to the training data.

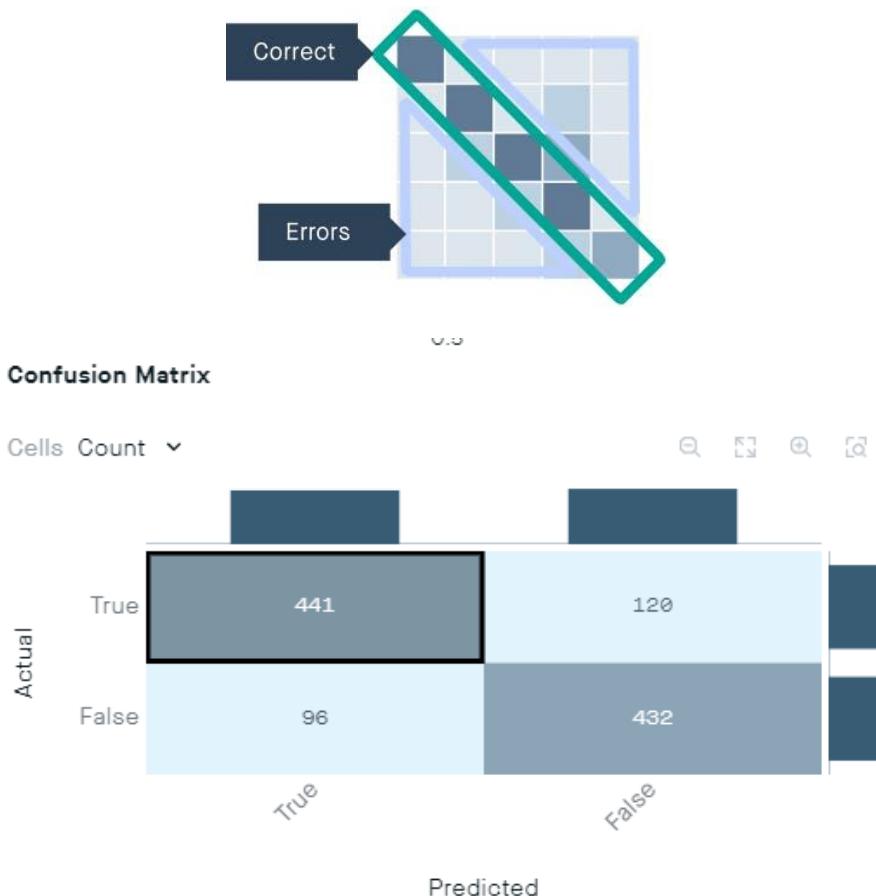
You can read more about the loss metrics here: <https://peltarion.com/knowledge-center/documentation/evaluation-view/classification-loss-metrics>

Accuracy

To evaluate the performance of the model, you can look at overall accuracy, which is displayed in the Experiment info section to the right. It should be approximately 85-90%.

Confusion matrix

Since the model solves a classification problem, a confusion matrix is displayed. The top-left to bottom-right diagonal shows correct predictions. Everything outside this diagonal are errors.



7) Create new deployment

- 1) In the **Deployment** view click **New deployment**.

The screenshot shows the Peltonion deployment interface. On the left, there's a sidebar with 'Projects / My First Project' and a search bar. Below it, a card for 'Experiment 1' shows it is disabled (3 hours ago). It has buttons for 'Enable', 'Rename', and 'Evaluation'. The main area is titled 'Deployment Info' and lists experiment details: Experiment 1, Checkpoint Epoch: 1 (best), Date created 2020-09-17, Date deployed 2020-09-17, and Deployment ID 80be3998-6499-4734-af32-bed66e73cd4f. Below this is a 'Parameters' section with 'Input' and 'Output' fields. The 'Input' field shows 'Summary' as a text (512) feature with the name 'Summary'. The 'Output' field shows 'Science Fiction' as a binary (1) feature with the name 'Science Fiction'. To the right, a 'Test deployment' API tester window is open, showing a CURL command to test the deployment. The URL is https://a.azure-eu-west.platform.peltarion.com/deployment/80be3998-6499-4734-af32-bed66e73cd4f/forward. The token field is empty. The input example shows a curl command to post 'Summary=VALUE' to the URL. A red circle highlights the 'curl' icon in the input example.

- 2) Select experiment and checkpoint of your trained model to test it for predictions, or enable for business product calls. Both best epoch and last epoch for each trained experiment are available for deployment.
- 3) Click the **Enable** switch to deploy the experiment.

8) Test the text classifier in a browser

- 1) Let's test your model. Click the **Test deployment** button, and you'll open the **Text classifier API tester** with all relevant data copied from your deployment.

II

Image & Text Classifier

API tester

Input datatype

Image Text

Output datatype

Categorical Binary

Setup

URL
<https://a.azure-eu-west.platform.peltarion.com/deployment/80be3998-6499-4734-af32-9aa11d1f-5f7d-4cdd-8403-909599eaedfd>

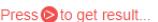
Token
9aa11d1f-5f7d-4cdd-8403-909599eaedfd

Threshold
0.5

Input parameter
Summary

Insert text here...

Result

Press  to get result...

- 2) Now, write your own summary, copy the example below or simply copy a recent summary from:

Example:

Harry Potter has never been the star of a Quidditch team, scoring points while riding a broom far above the ground. He knows no spells, has never helped to hatch a dragon, and has never worn a cloak of invisibility.

All he knows is a miserable life with the Dursleys, his horrible aunt and uncle, and their abominable son, Dudley -- a great big swollen spoiled bully. Harry's room is a tiny closet at the foot of the stairs, and he hasn't had a birthday party in eleven years.

But all that is about to change when a mysterious letter arrives by owl messenger: a letter with an invitation to an incredible place that Harry — and anyone who reads about him — will find unforgettable.

Now, write your own summary, copy the example below or simply copy a recent summary from: Now, write your own summary, copy the example below or simply copy a recent summary from:

All he knows is a miserable life with the Dursleys, his horrible aunt and uncle, and their abominable son, Dudley -- a great big swollen spoiled bully. Harry's room is a tiny closet at the foot of the stairs, and he hasn't had a birthday party in eleven years.

But all that is about to change when a mysterious letter arrives by owl messenger: a letter with an invitation to an incredible place that Harry — and anyone who reads about him — will never forget.



Result

Press to get result...



- 3) Click Play to get a result.

All he knows is a miserable life with the Dursleys, his horrible aunt and uncle, and their abominable son, Dudley -- a great big swollen spoiled bully. Harry's room is a tiny closet at the foot of the stairs, and he hasn't had a birthday party in eleven years.

But all that is about to change when a mysterious letter arrives by owl messenger: a letter with an invitation to an incredible place that Harry — and anyone who reads about him — will never forget.



Result

Science Fiction true



9) Next Steps

The next steps could be to try to run the project for more epochs and see if that improves the result or maybe change the learning rate

Activity 6 – [Bonus] Importing data

In this activity, we will learn:

- How to upload CSV file into Azure Machine Learning Studio (Classic)
- How to import a CSV file from the web

- 1) To use your own data in Machine Learning Studio (classic) to develop and train a predictive analytics solution, you can use data from:

Local file - Load local data ahead of time from your hard drive to create a dataset module in your experiment.

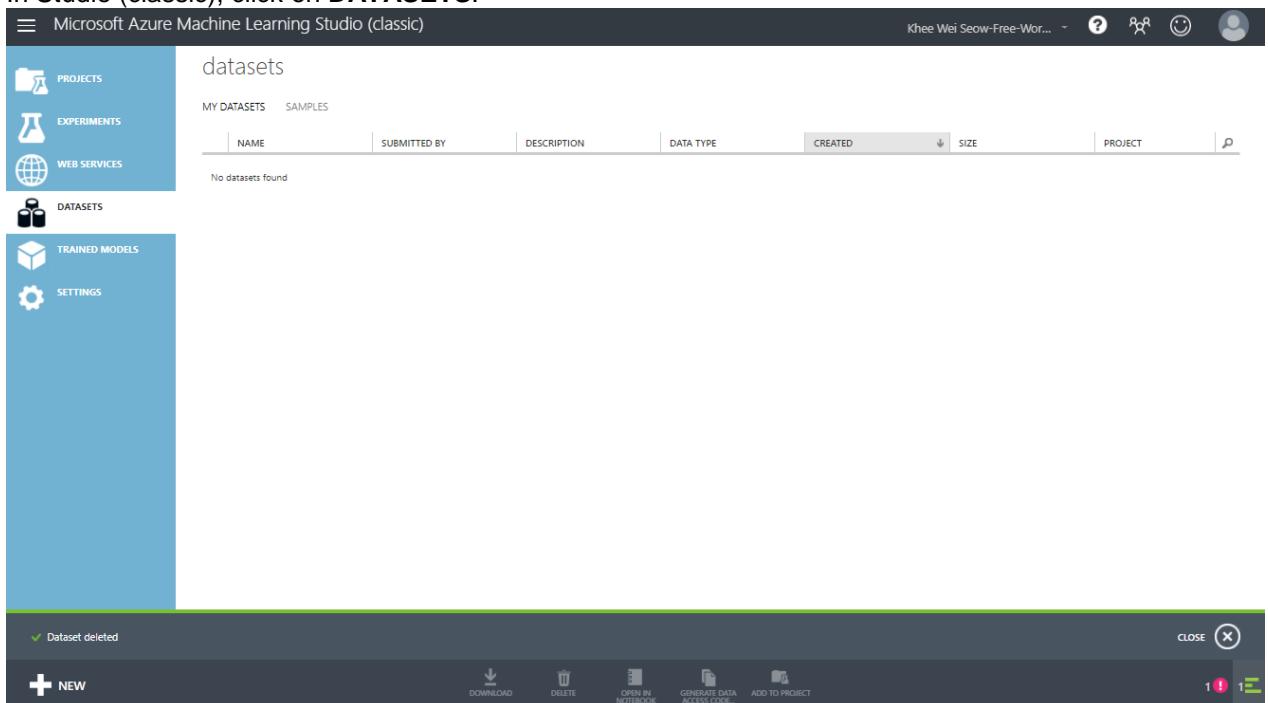
Online data sources - Use the Import Data module to access data from one of several online sources while your experiment is running

Machine Learning Studio (classic) experiment - Use data that was saved as a dataset in Machine Learning Studio (classic). For a list of datasets available in Studio (classic), you may want to refer to <https://docs.microsoft.com/en-us/azure/machine-learning/studio/use-sample-datasets>

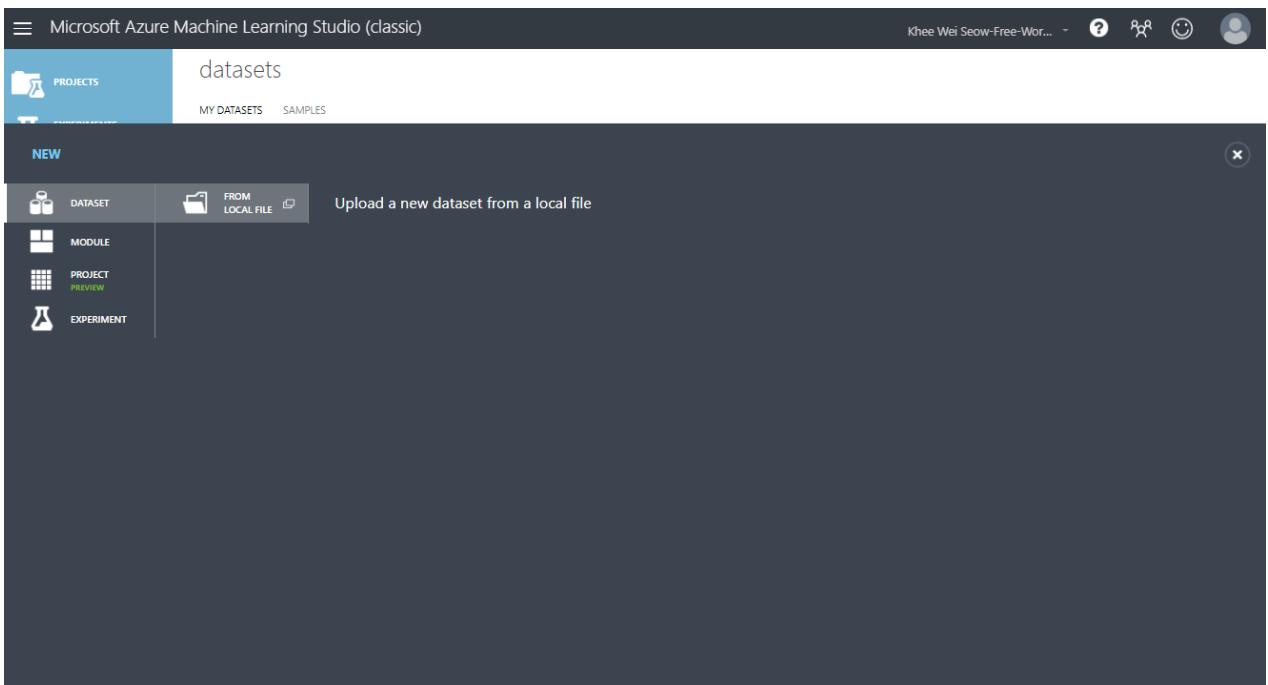
SQL Server database - Use data from a SQL Server database without having to copy data manually

2) Import from a local file

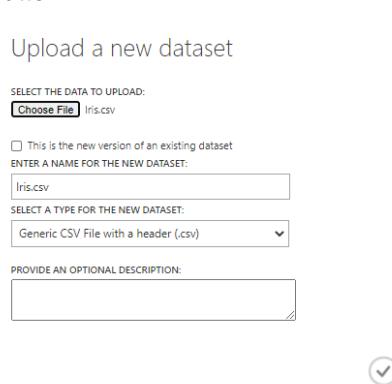
- 1) In Studio (classic), click on DATASETS.



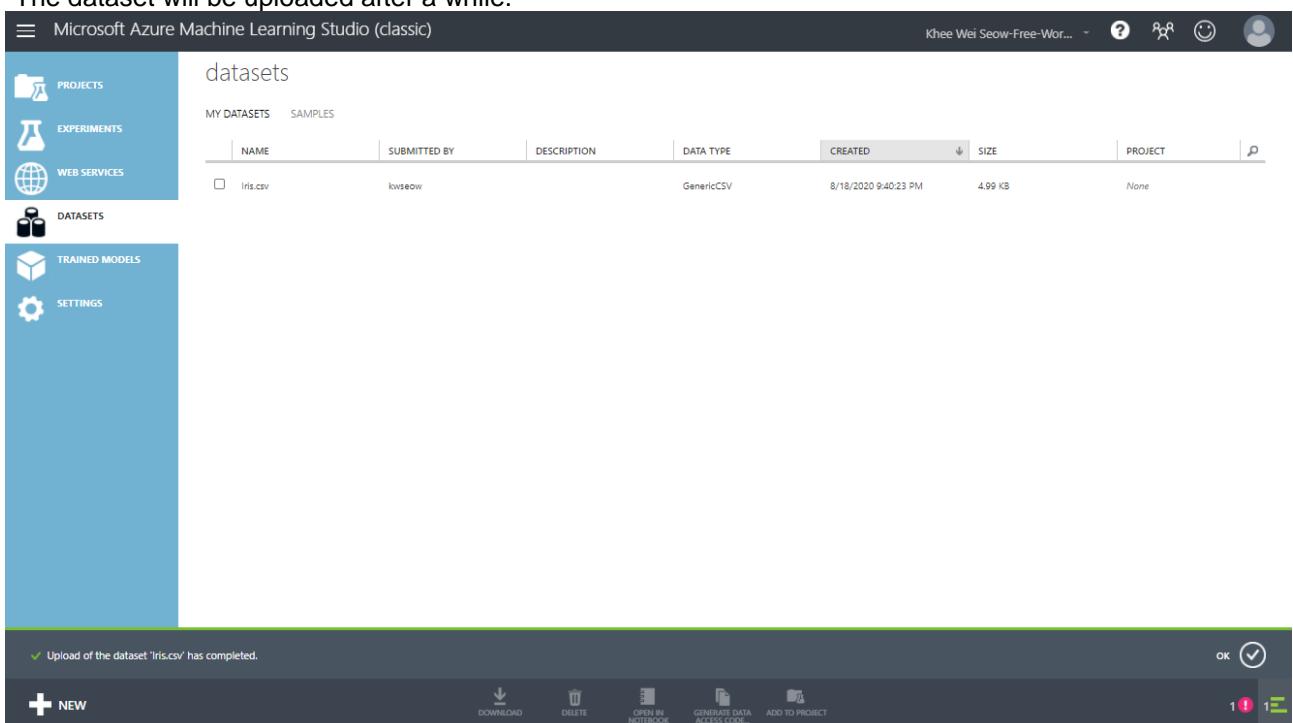
- 2) Click +NEW and then FROM LOCAL FILE.



- 3) In the Upload a new dataset dialog, click **Choose File** button and locate the iris.csv file provided. Click the tick button.

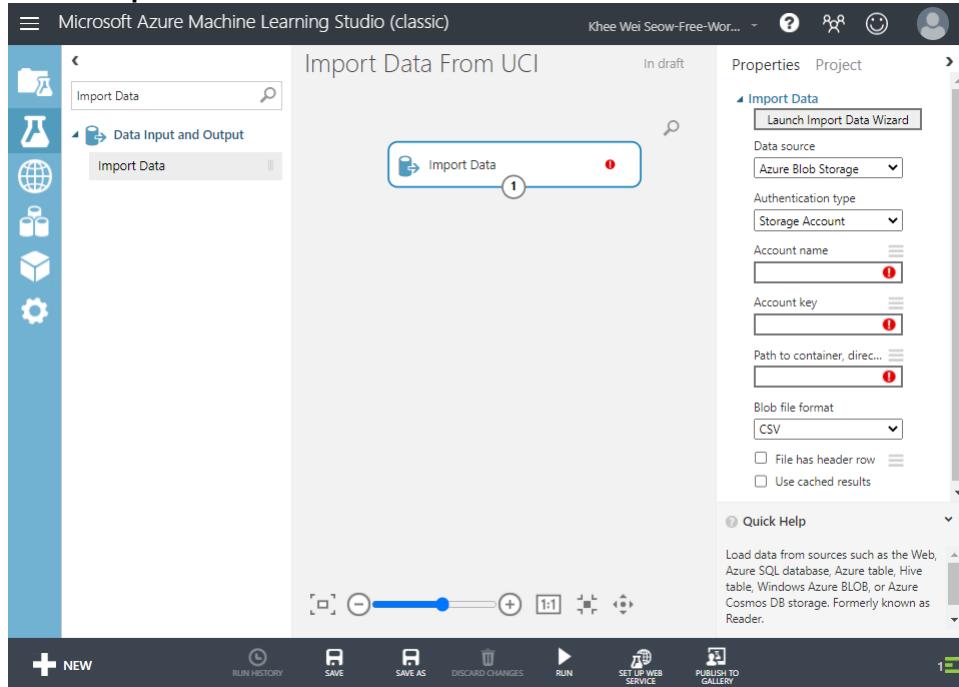


- 4) The dataset will be uploaded after a while.

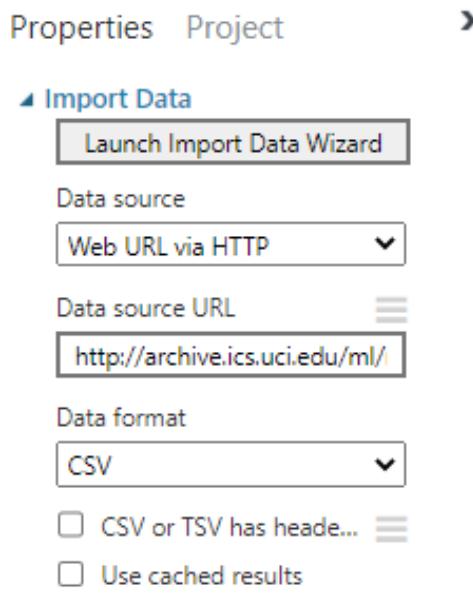


3) Import directly from the web

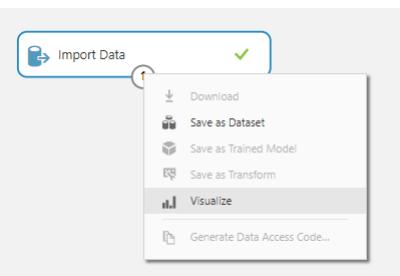
- 1) Create a new **Black Experiment** and name it Import dataset from UCI.
- 2) Add an **Import Data** module to the Canvas



- 3) Set the properties of the Import Data module as follows (Data Source URL: <http://archive.ics.uci.edu/ml/index.php>) :



- 4) Click **RUN**. Wait for the green tick to appear before proceeding to the next step.
- 5) Right click the output port of the Import Data module and select Visualise.



6) You should see the following:

Import Data From UCI > Import Data > Results dataset

rows	columns	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	
32562	15												
		view as	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174			
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0			
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0			
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0			
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0			
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0			
49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0			

x

Statistics and Visualizations

Activity 7 – [Bonus] Cleaning and Structuring Data

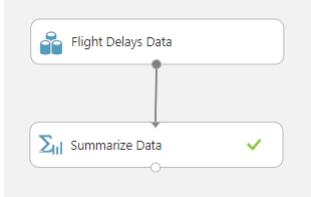
In this activity, we will learn:

- How to summarise data
- How to remove missing values
- How to remove duplicate records
- How to remove outliers
- How to read a boxplot

In Studio (classic), create a new Blank Experiment using **+ New** and call it **Data Cleaning**.

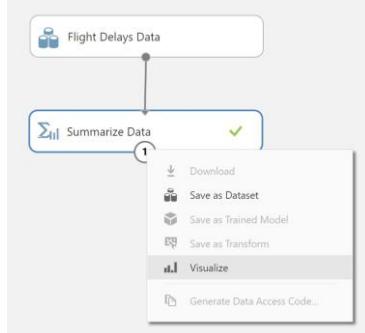
1) Summarizing Data

- 1) Add the following dataset and module onto the canvas.

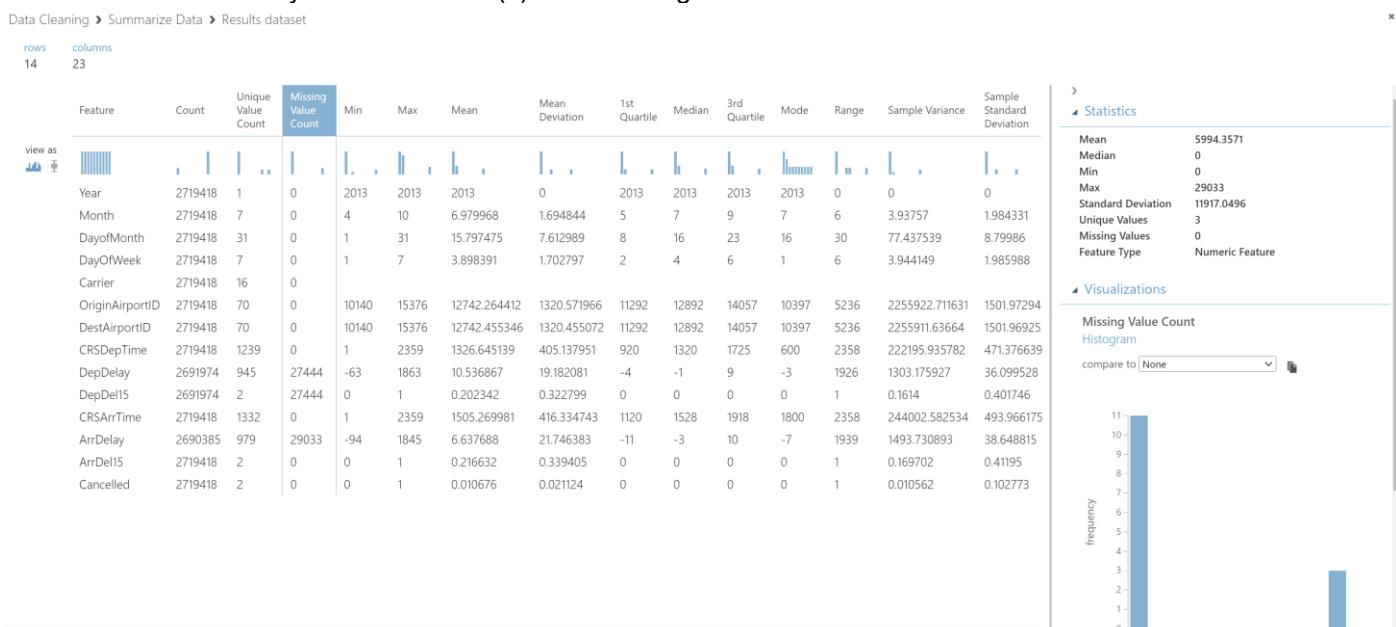


- 2) Click **Run**,

- 3) Right-click on the output port of the **Summarize Data** module and select **Visualize**:



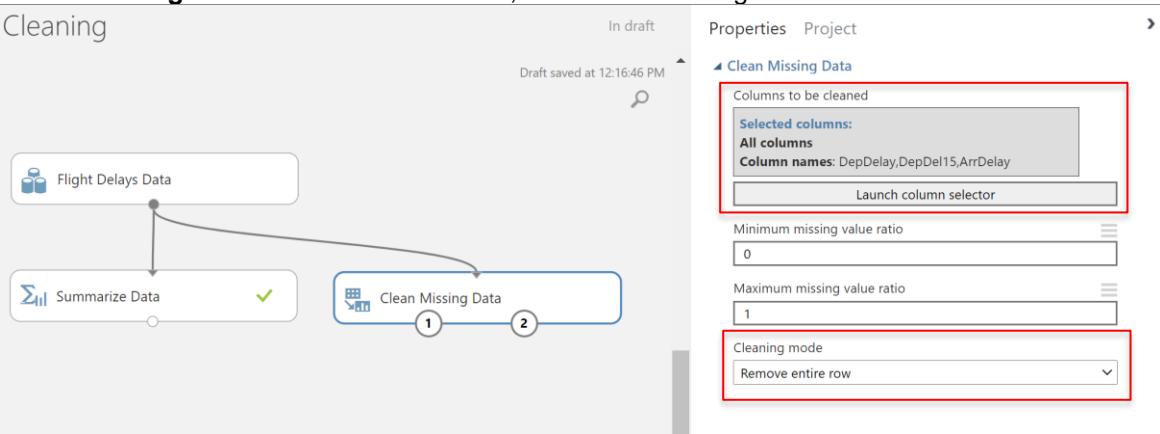
- 4) You should see a similar screen (below) that shows a summary of the dataset. The column “**Missing Value Count**” will tell you which feature(s) have missing data.



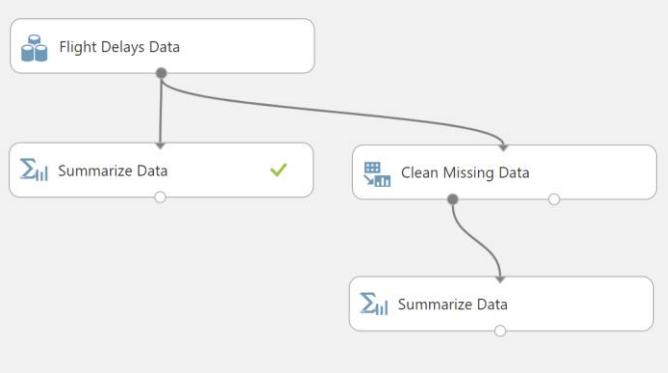
2) Missing Value

- Add a **Clean Missing Data** module to the canvas, connect and configure it as follows.

Data Cleaning



- Add a **Summarize Data** module to the canvas as follows:



- Click on **Run**

- Visualise the output of the newly added **Summarize Data** module. There should be no missing values in the dataset now.

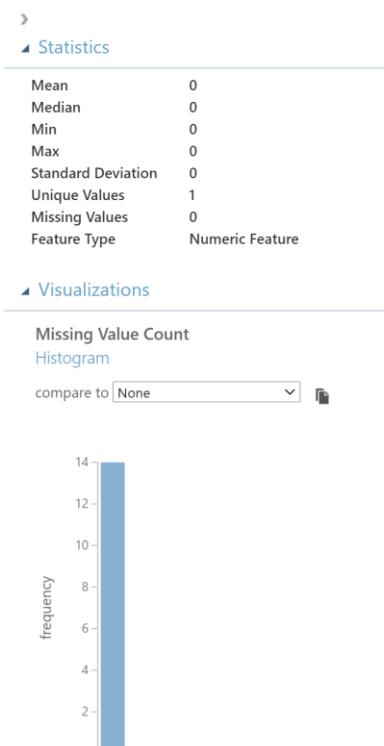
Data Cleaning > Summarize Data > Results dataset

rows
14

columns
23

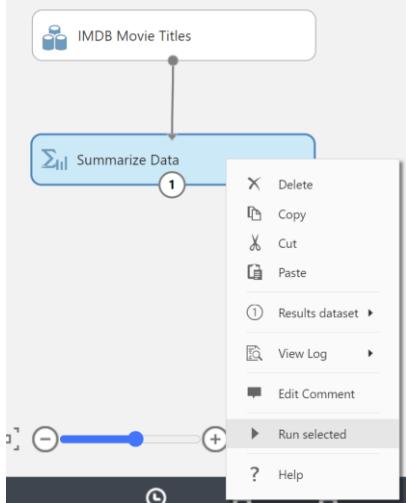
view as

Feature	Count	Unique Value Count	Missing Value Count	Min	Max	Mean	Mean Deviation	1st Quartile
Year	2690385	1	0	2013	2013	2013	0	2013
Month	2690385	7	0	4	10	6.985231	1.696297	5
DayofMonth	2690385	31	0	1	31	15.797197	7.619317	8
DayOfWeek	2690385	7	0	1	7	3.901087	1.704897	2
Carrier	2690385	16	0					
OriginAirportID	2690385	70	0	10140	15376	12741.989084	1321.300676	11292
DestAirportID	2690385	70	0	10140	15376	12742.320152	1321.26511	11292
CRSDepTime	2690385	1239	0	1	2359	1325.569479	405.112514	920
DepDelay	2690385	944	0	-63	1863	10.512613	19.149409	-4
DepDel15	2690385	2	0	0	1	0.202167	0.322591	0
CRSArrTime	2690385	1331	0	1	2359	1504.449868	416.335073	1119
ArrDelay	2690385	979	0	-94	1845	6.637688	21.746383	-11
ArrDel15	2690385	2	0	0	1	0.208178	0.32968	0
Cancelled	2690385	1	0	0	0	0	0	0

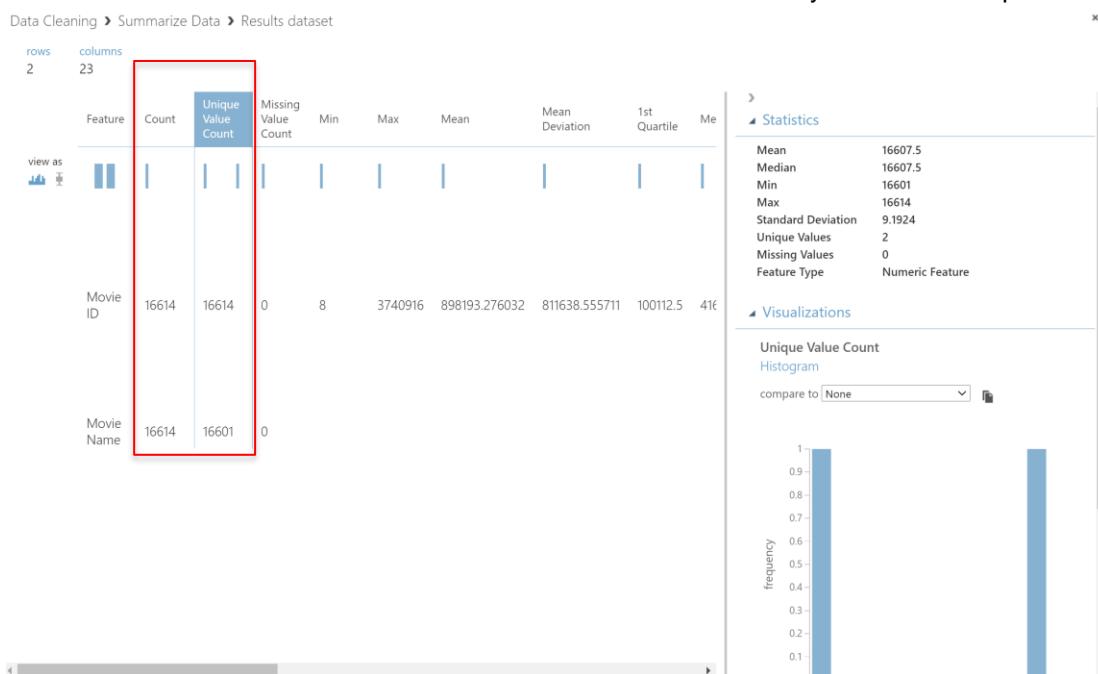


3) Duplicate Records

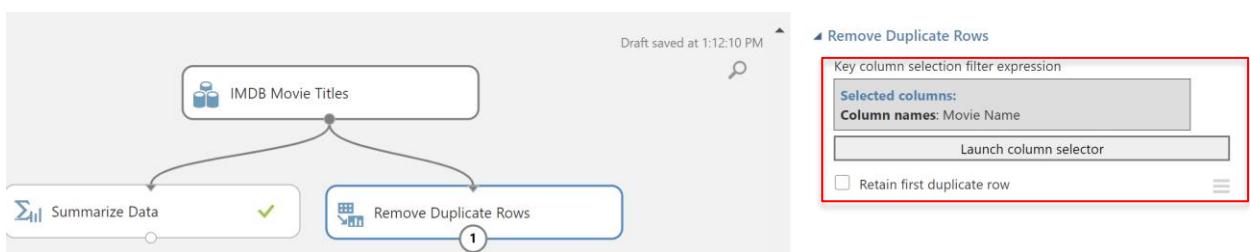
- 1) Add the **IMDB Movie Titles** and **Summarize Data** module to the canvas.
- 2) Right-click on the **Summarize Data** module and select **Run selected**. This action will only run the selected module rather than the whole experiment saving some time.



- 3) Visualize the dataset. There should be a total of 16614 rows but only 16601 are unique.

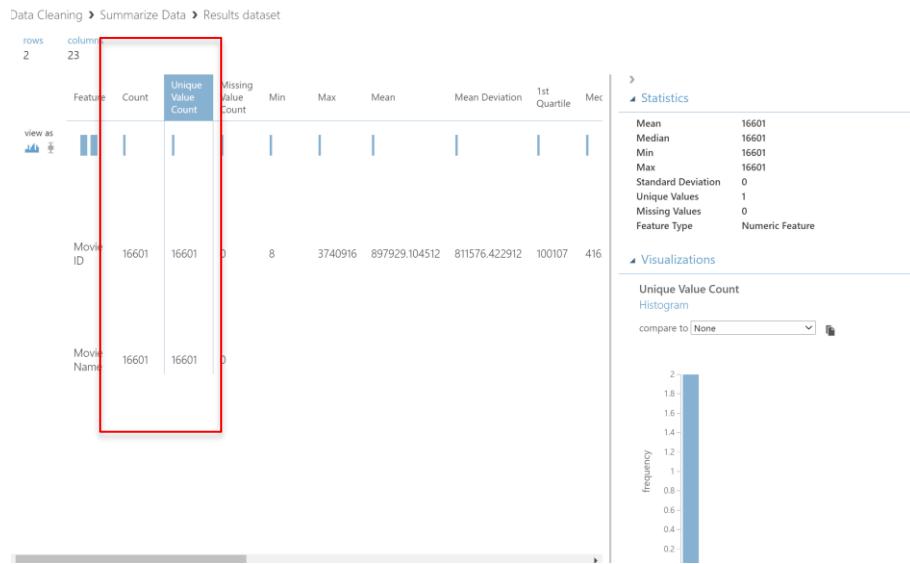


- 4) Add a **Remove Duplicate Rows** module to the canvas, connect and configure it as follows:



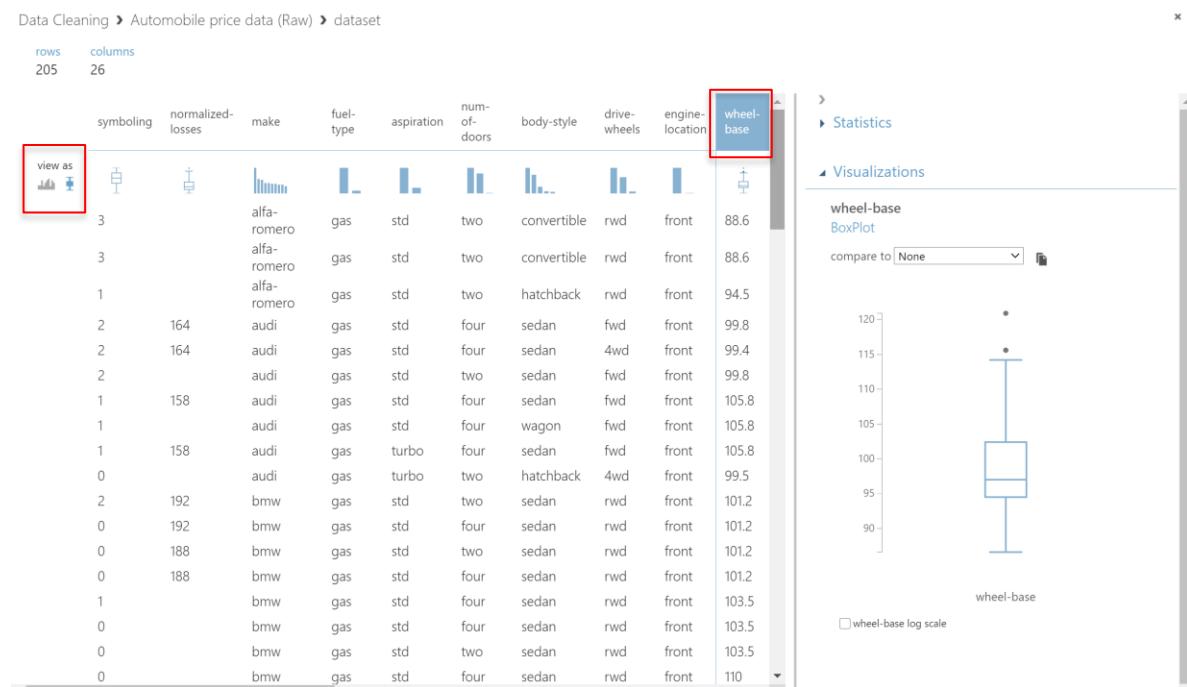
Note: Use the Retain first duplicate row checkbox to indicate which row to return when duplicates are found:
 If selected, the first row is returned and others discarded.
 If you uncheck this option, the last duplicate row is kept in the results, and others are discarded.

- 5) Add a **Summarize Data** module and **Visualize** the output. There should be a total of 16601 rows and the same number of unique rows.



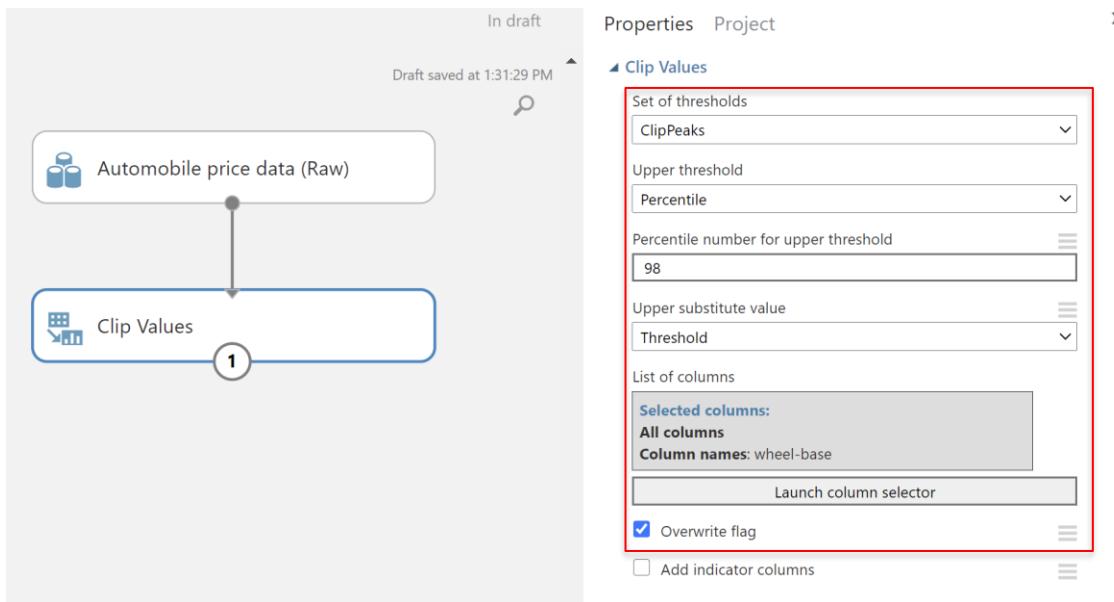
4) Removing Outliers

- 1) Add the **Automobile price data (Raw)** to the canvas.
- 2) **Visualize** and view as **Box-Plot** the wheel-base feature.

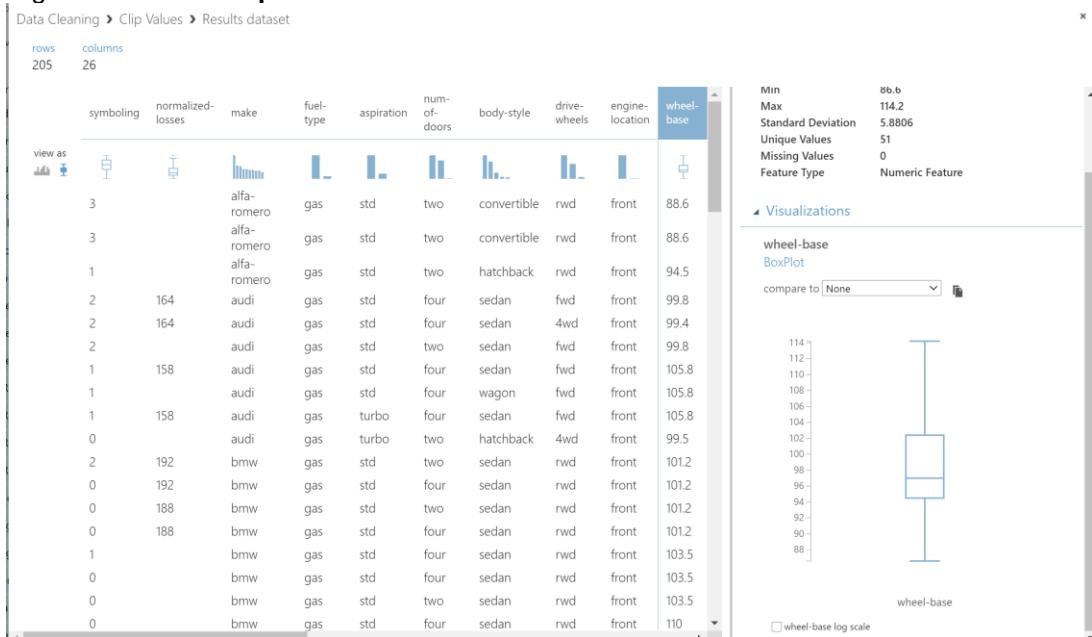


Note: The dots outside the boxplot are the outliers, and they should be removed from the dataset as they may affect the accuracy of the prediction.

- 3) Add a **Clip-Values** module to the canvas, connect and configure it as follows:
Ref: <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/clip-values>



- 4) Right-Click on the **Clip-Values** and select **Run selected**.



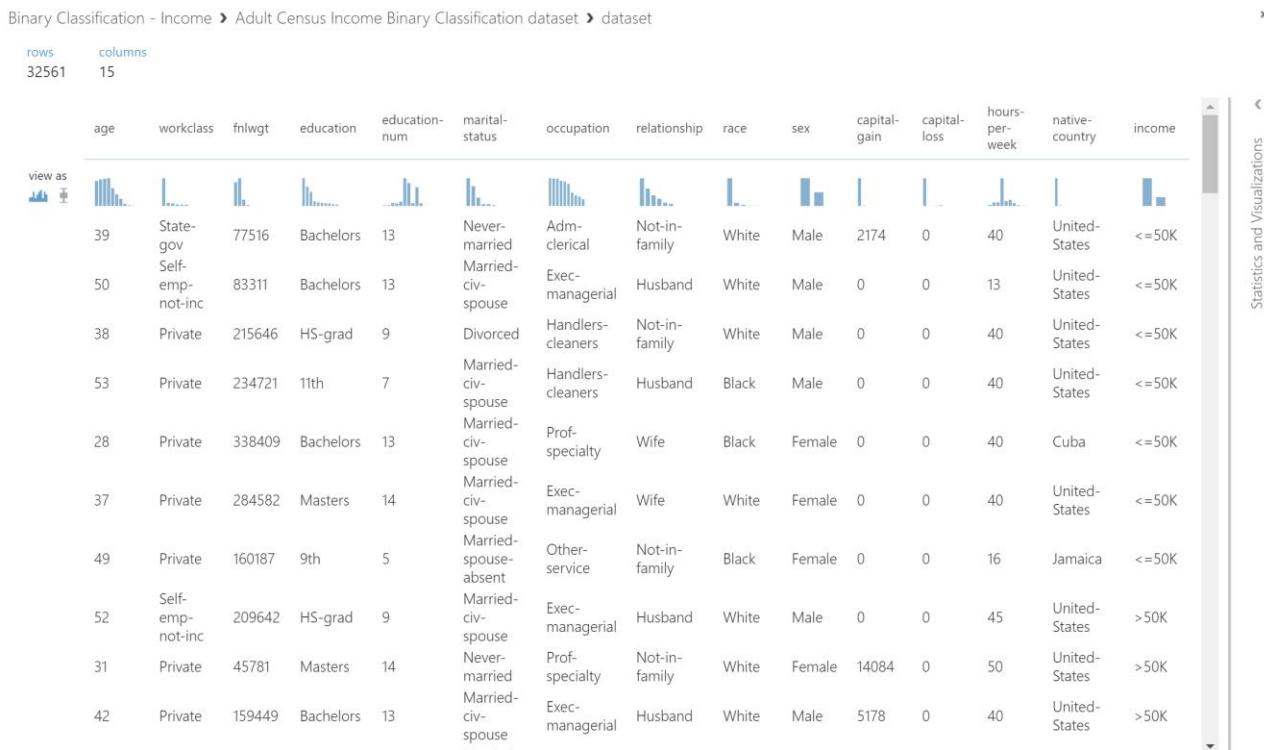
Note: If the outliers are still present, adjust the **Percentile number for upper threshold** property of the **clip-values** module

Activity 8 – [Bonus] Using Binary Classification Algorithms

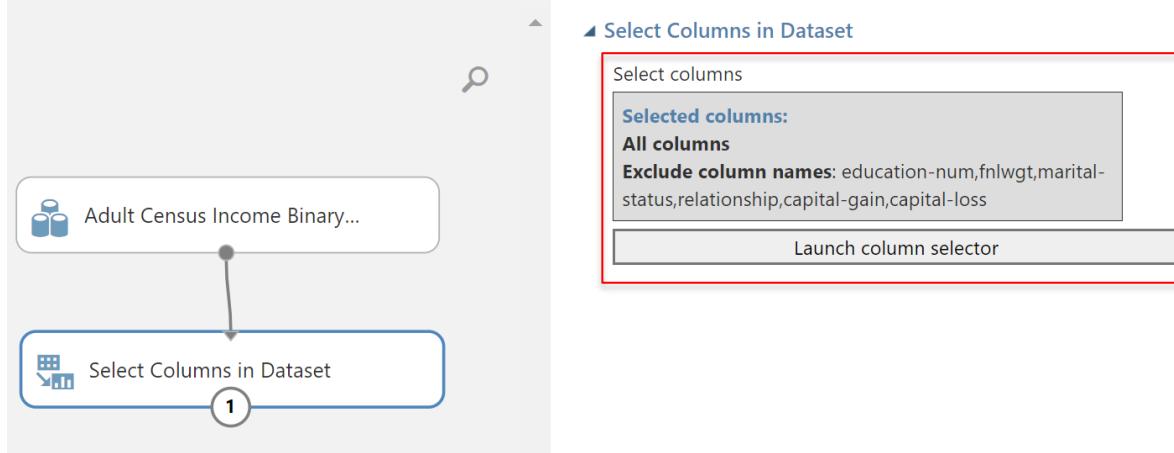
In this activity, we will learn:

- How to use the Two-Class Logistic Regression algorithm for training
- How to use the Two-Class Boosted Decision Tree algorithm for training
- How to evaluate two learning algorithms
- How to save a trained model

- 1) Create a new experiment and name it as **Binary Classification – Income**.
- 2) Add the **Adult Census Income Binary Classification** dataset to the canvas.
- 3) Right-click on the output port of the dataset and select Visualize. You should see the following:

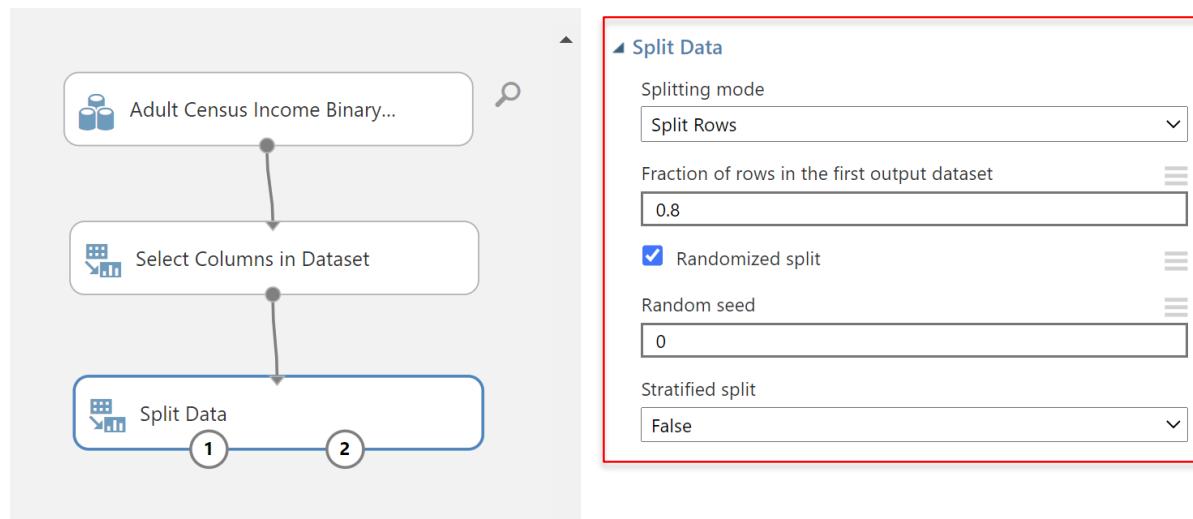


- 4) Add the **Select Columns in Dataset** module to the canvas and connect and configure it as follows:

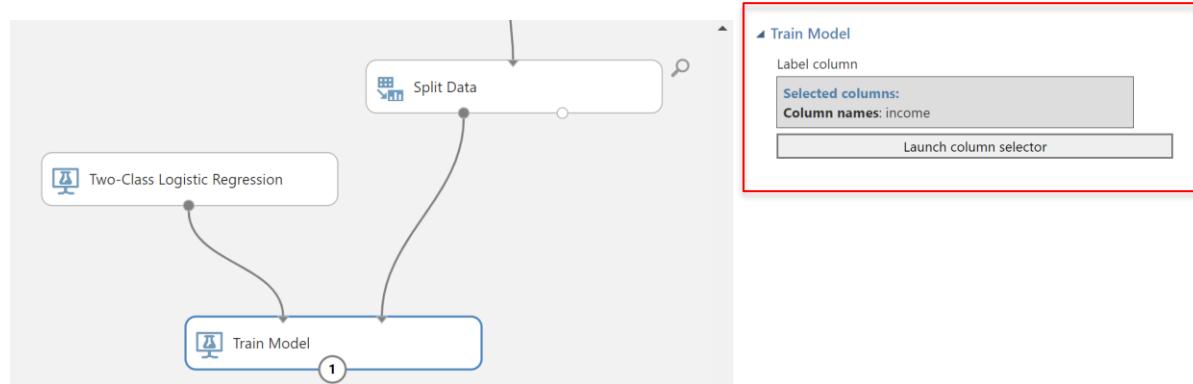


- 5) Add **Split Data** module to the canvas, connect and configure it as follows:

Ref: <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/split-data>



- 6) Add a **Two-Class Logistic Regression** and **Train Model** modules to the canvas, connect and configure them as follows:

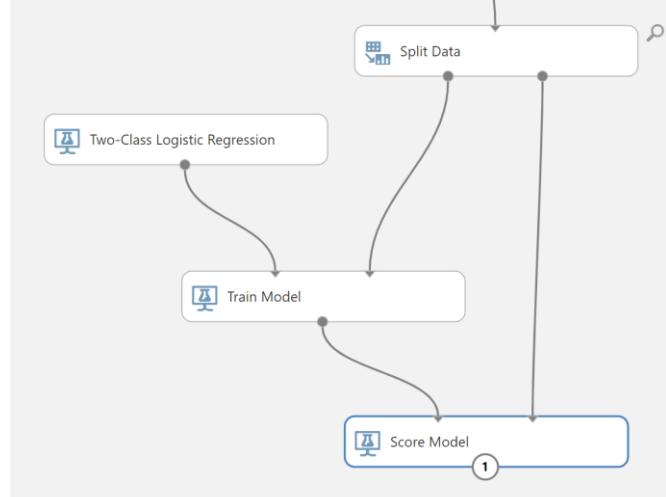


Ref:

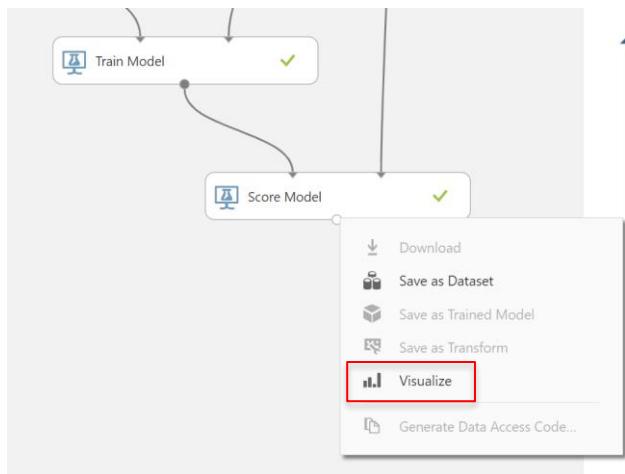
<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-logistic-regression>
<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/train-model>

Note: You use the **Two-Class Logistic Regression** module to create a logistic regression model that can be used to predict one of two states/classes of the target variable (label).

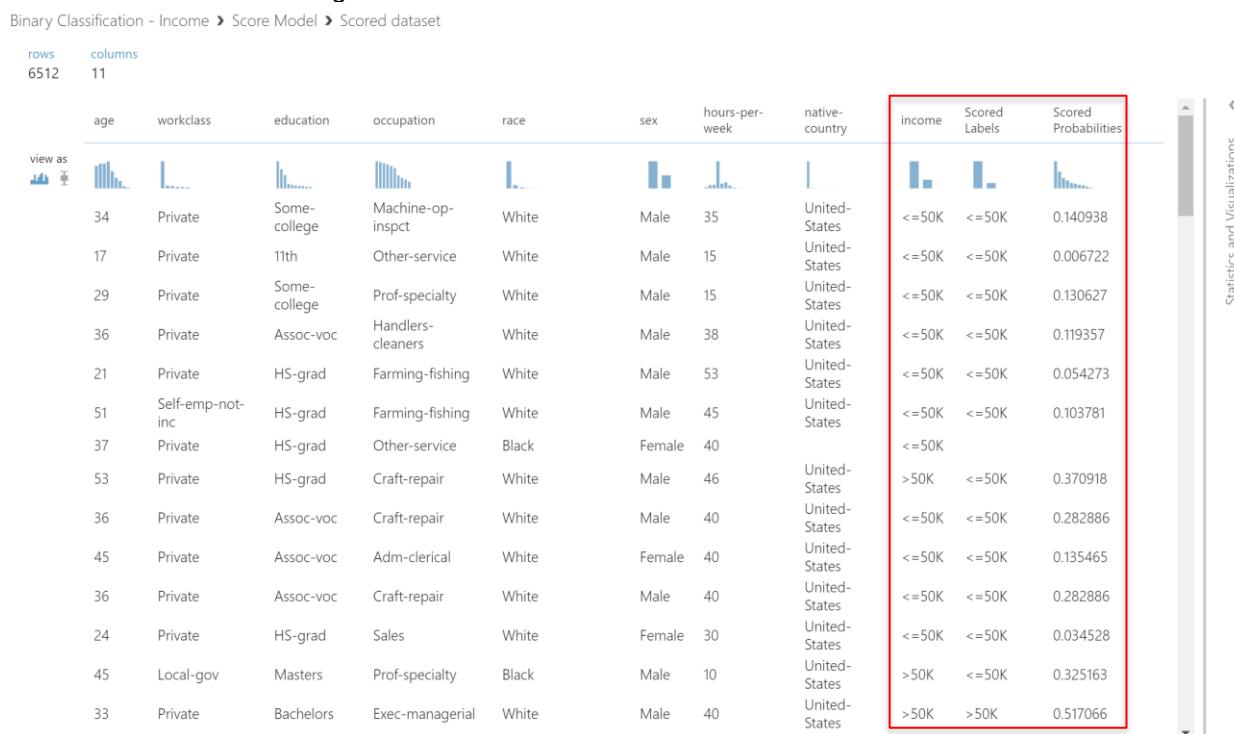
- 7) Add the **Score Model** module to the canvas and connect it as follows:



- 8) Click **Run**. When it is completed, right-click on the output port and select **Visualize**:



9) You should see the following:



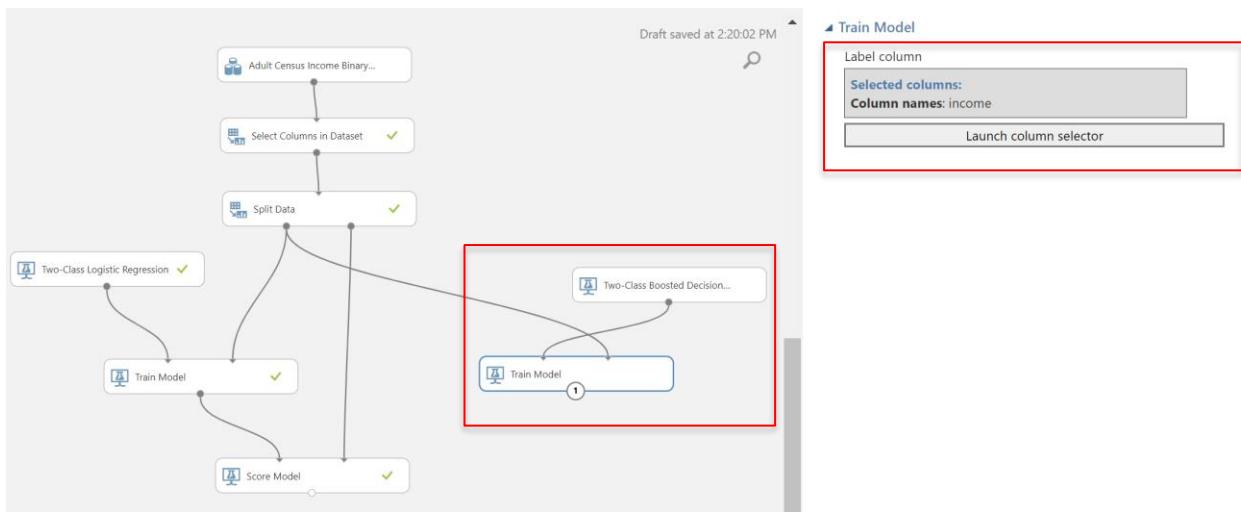
Note:

The **Scored Labels** column shows the predicted income group.

The **Scored Probabilities** column shows the confidence levels of the scored labels

If your test dataset is missing the dependent values, your **Scored Labels** may be empty.

- 10) Add the **Two-Class Boosted Decision Tree** and the **Train Model** modules to the canvas and connect and configure them as follows:



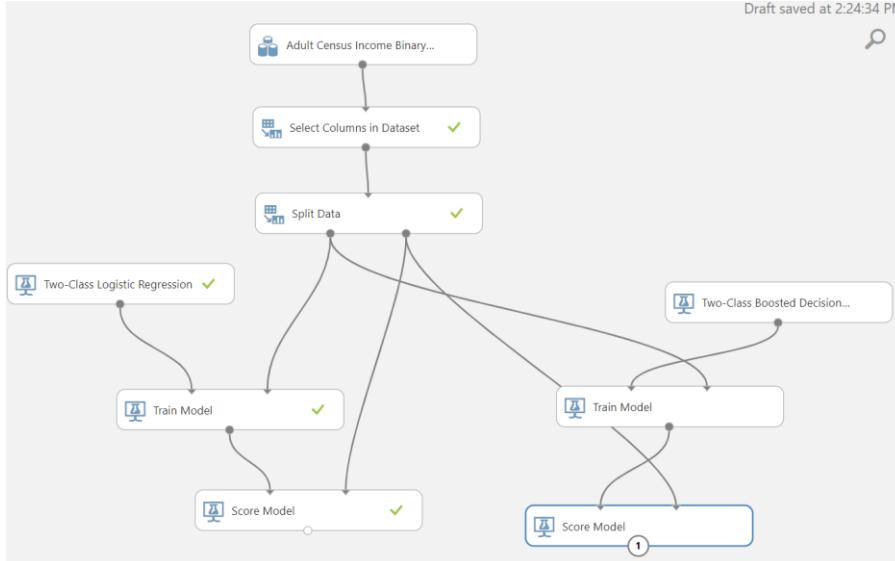
Ref:

<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-boosted-decision-tree>

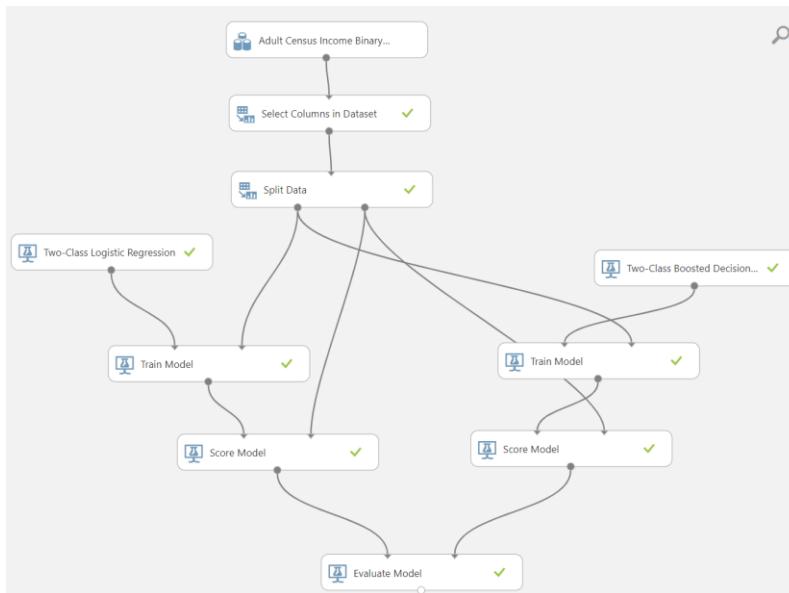
Note:

You use the **Two-Class Boosted Decision Tree** module to create a machine learning model that is based on the boosted decision trees algorithm. A boosted decision tree is an ensemble learning method in which the second tree corrects for the errors of the first tree, the third tree corrects for the errors of the first and second tree, and so forth. Predictions are based on the entire ensemble of trees together that makes the prediction.

- 11) Add a **Score Model** module to the canvas and connect it as follows:



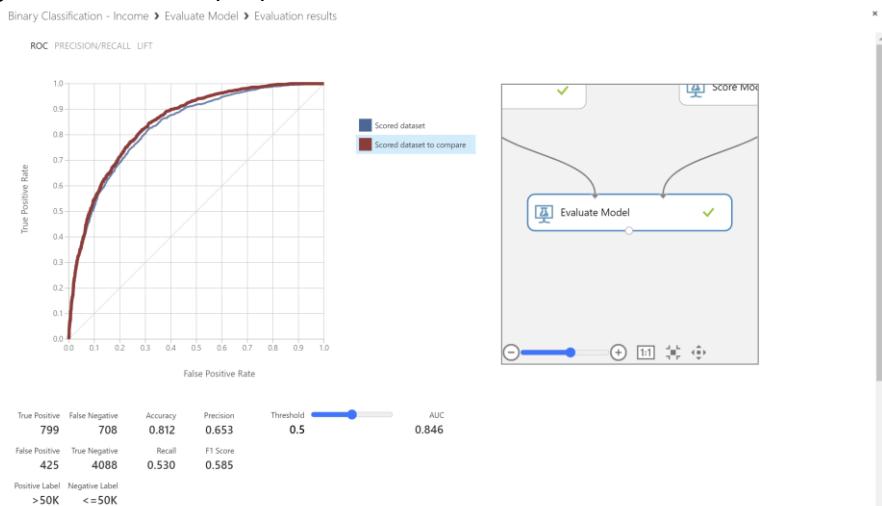
- 12) Add the **Evaluate Model** module to the canvas and connect it as follows:



Ref: <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/evaluate-model>

13) Click Run

14) Right-click on the output port of the **Evaluate Model** module and select **Visualize**:



You should see two curves on the output. Click on either the blue or red box. The curve that is closer to the vertical axis is the better one.

Evaluation metrics

Mean Absolute Error (MAE)	The average of absolute errors (an error is the difference between the 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

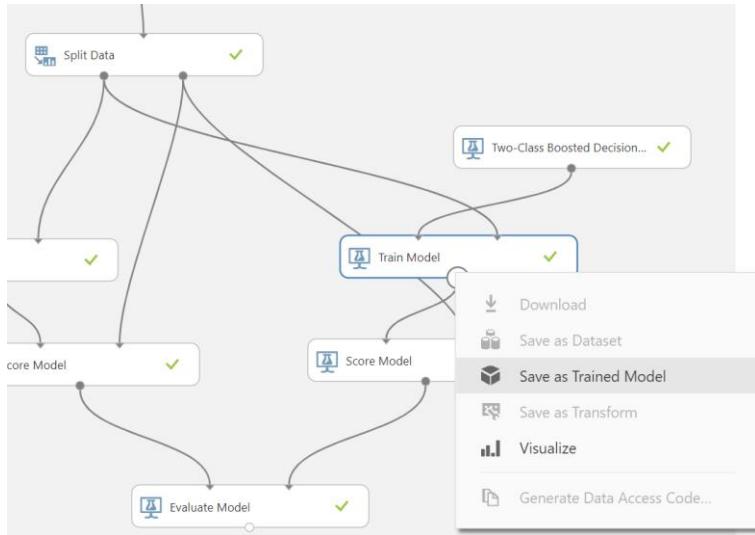
For each of the metrics, smaller is better.

A smaller value indicates that the predictions more closely match the actual values.

For Coefficient of Determination, the closer its value to one (1.0), the better the predictions.

15) Saved a train model

- Right-click on the output port of the **Train Model** module (on the right of the canvas) and select **Save as Trained Model**:



Note: Once a **Train Model** is trained, you can save it as a trained model so that you can later use it on the canvas without specify the algorithm.

- Name the trained model as **Census Trained model**:

Save trained model

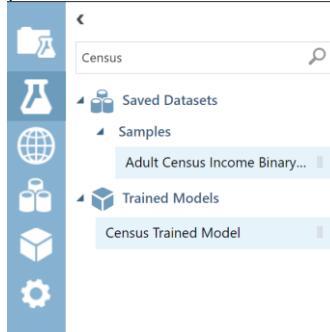
This is the new version of an existing trained model

Enter a name for the new trained model:
Census Trained Model

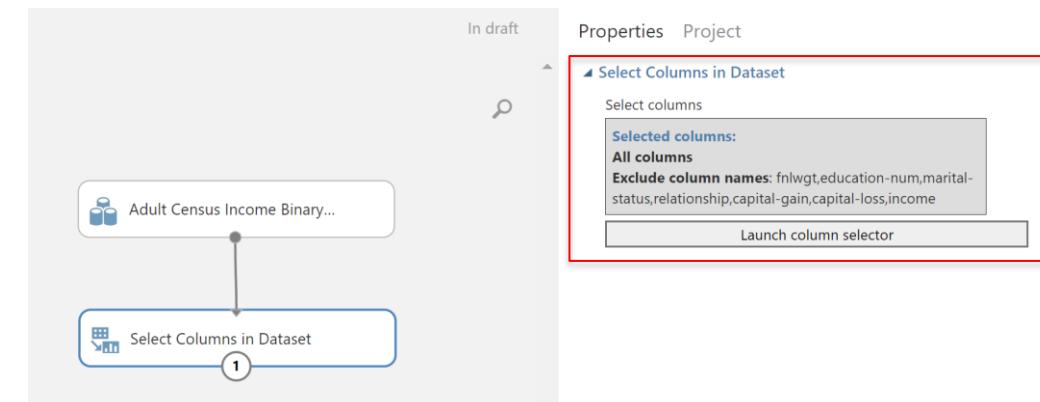
Provide an optional description:



- You can now find the newly saved trained model in the left panel.

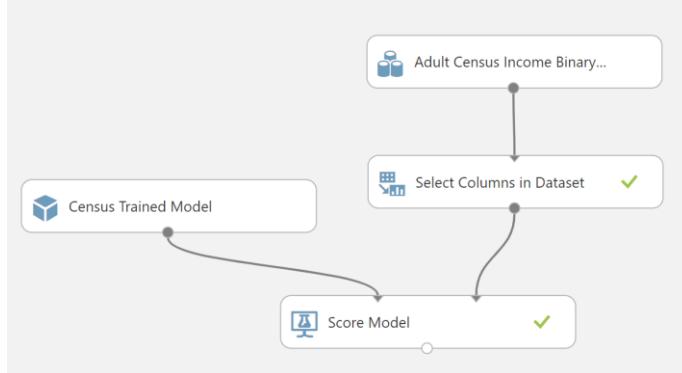


- Create a new experiment and name it as **Income Prediction**.
- Add the **Adult Census Income Binary Classification** dataset to the canvas.
- Add the **Select Columns in Dataset** module to the canvas and connect and configure it as follows:



Note: We have excluded **income** column in this case. Our model is going to predict income, so there is no need to have the income as part of the inputs.

- 7) Drag and drop the saved trained model onto the canvas, and a **Score Model** module and connect as follows:



- 8) Click **Run**.

- 9) Right-click on the output port of the **Score Model** module and select **Visualize**:

Income Prediction > Score Model > Scored dataset											
rows	columns										
32561	10	age	workclass	education	occupation	race	sex	hours-per-week	native-country	Scored Labels	Scored Probabilities
view as		histogram	histogram	histogram	histogram	histogram	histogram	histogram	histogram	histogram	
39	State-gov	Bachelors	Adm-clerical	White	Male	40	United-States	<=50K	0.37412		
50	Self-emp-not-inc	Bachelors	Exec-managerial	White	Male	13	United-States	>50K	0.744454		
38	Private	HS-grad	Handlers-cleaners	White	Male	40	United-States	<=50K	0.10904		
53	Private	11th	Handlers-cleaners	Black	Male	40	United-States	<=50K	0.302544		
28	Private	Bachelors	Prof-specialty	Black	Female	40	Cuba	<=50K	0.08168		
37	Private	Masters	Exec-managerial	White	Female	40	United-States	<=50K	0.451775		
49	Private	9th	Other-service	Black	Female	16	Jamaica	<=50K	0.012307		
52	Self-emp-not-inc	HS-grad	Exec-managerial	White	Male	45	United-States	<=50K	0.432165		
31	Private	Masters	Prof-specialty	White	Female	50	United-States	<=50K	0.428208		
42	Private	Bachelors	Exec-managerial	White	Male	40	United-States	>50K	0.776759		
37	Private	Some-college	Exec-managerial	Black	Male	80	United-States	>50K	0.835646		
30	State-gov	Bachelors	Prof-specialty	Asian-Pac-Islander	Male	40	India	<=50K	0.246062		
23	Private	Bachelors	Adm-clerical	White	Female	30	United-States	<=50K	0.005884		
32	Private	Assoc-acdm	Sales	Black	Male	50	United-States	<=50K	0.250539		
40	Private	Assoc-voc	Craft-repair	Asian-Pac-Islander	Male	40					
34	Private	7th-8th	Transport-moving	Amer-Indian-Eskimo	Male	45	Mexico	<=50K	0.04403		
25	Self-emp-not-inc	HS-grad	Farming-fishing	White	Male	35	United-States	<=50K	0.018797		
32	Private	HS-grad	Machine-op-inspect	White	Male	40	United-States	<=50K	0.07359		
38	Private	11th	Sales	White	Male	50	United-States	<=50K	0.402333		
43	Self-emp-not-inc	Masters	Exec-managerial	White	Female	45	United-States	<=50K	0.498074		
40	Private	Doctorate	Prof-specialty	White	Male	60	United-States	>50K	0.92778		

Note:

Observe that **Income** column is no longer visible.

Scored Labels and **Scored Probabilities** have been added. **Scored Labels** is the predicted income.

Exercise: Use the **Select Columns in Data** after the score model to show only the predicted salary.

