

Contents

Activity 1 – First Machine Learning with Azure.....	2
Activity 2 – Deploying your experiment as a Web Service.....	19
Activity 3 – Importing data.....	26
Activity 4 – Cleaning and Structuring Data.....	30
Activity 5 – Using Binary Classification Algorithms.....	35
Activity 6 – Evaluating a Regression Model with Cross Validation	42
Activity 7 – Hyperparameters tuning.....	45
Activity 8 – Car Damage Assessment Classification	48
Activity 9 – Book Genre Classifier.....	59
Activity 10 – Creating a Sentiment Analyser	70
Activity 11 – [Bonus] Anomaly Detection	80

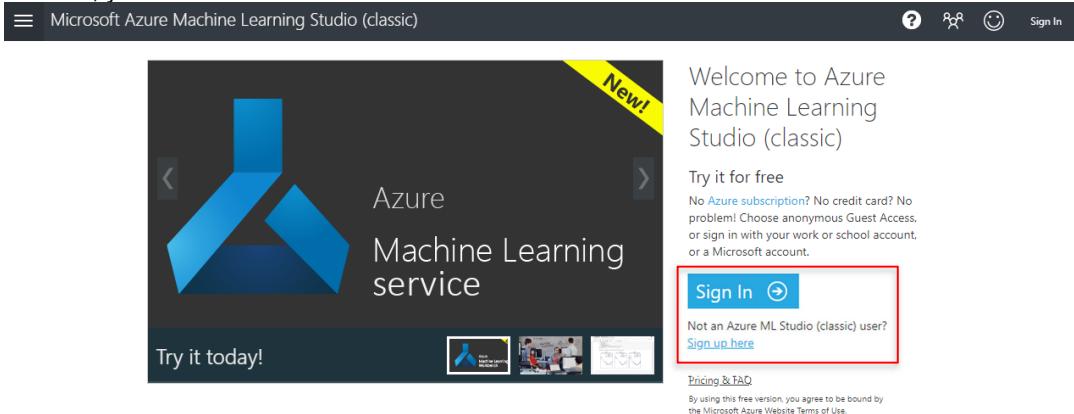
Activity 1 – First Machine Learning with Azure

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

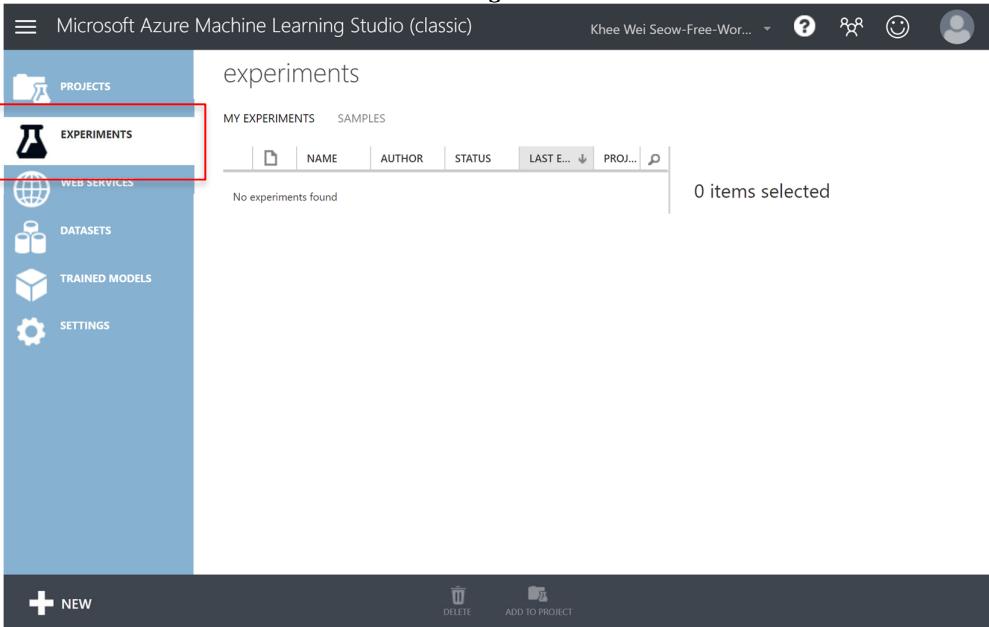
1) Setup account on Azure

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

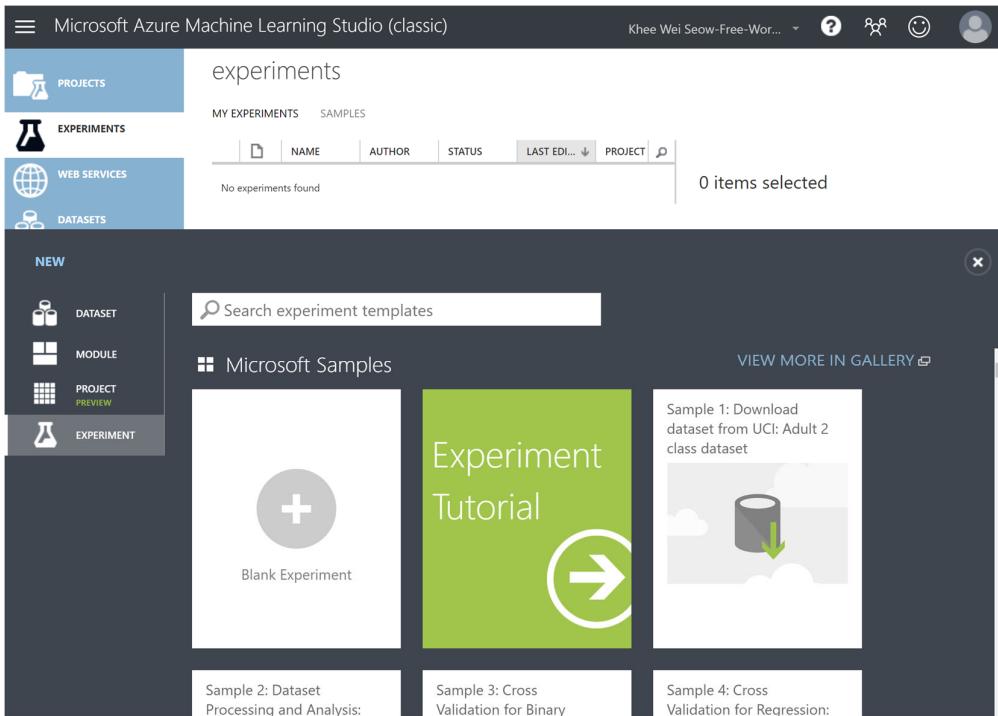


2) Create a new Experiment

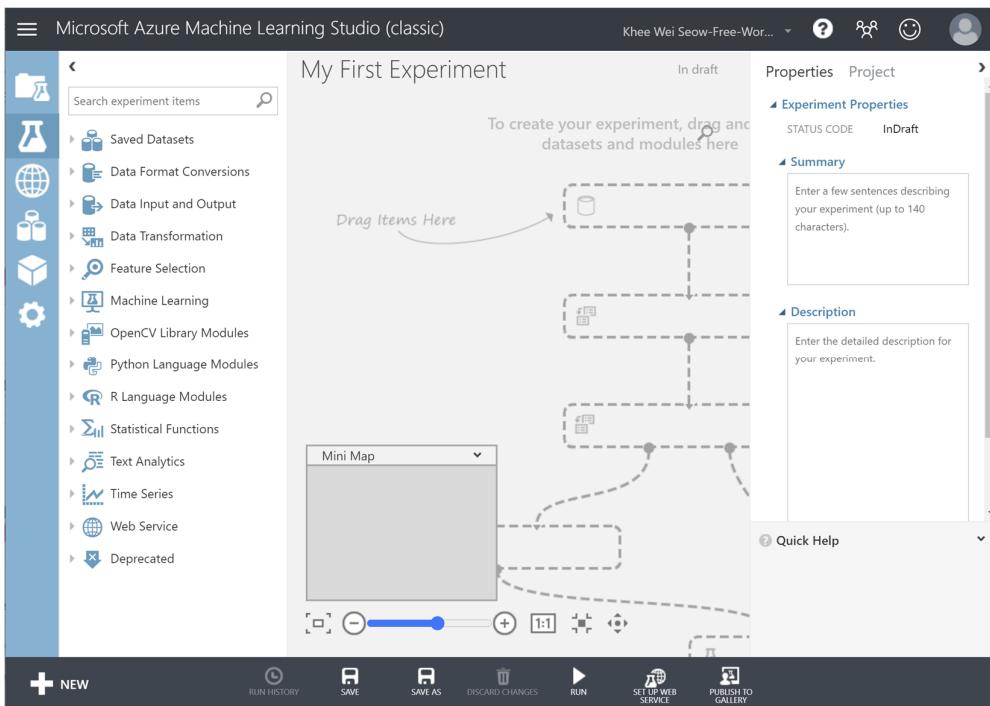
- 1) Click on **EXPERIMENTS** to see the following view



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

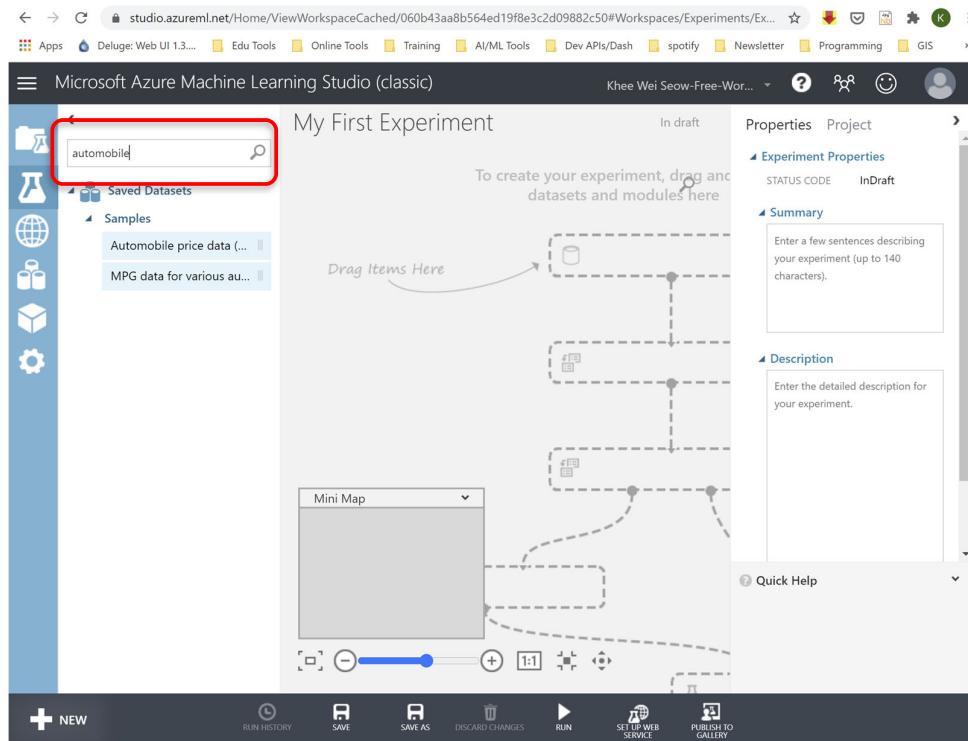


- 3) Name it “**My First Experiment**”. A screen similar to the follow will be presented.

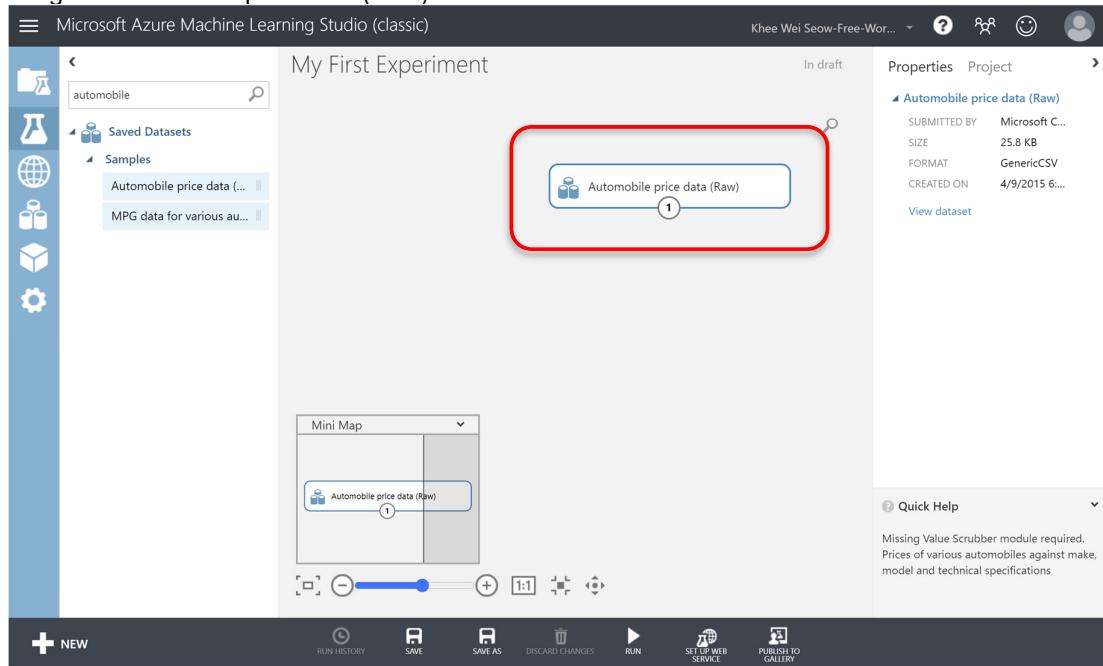


3) Setting up your data

- 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**

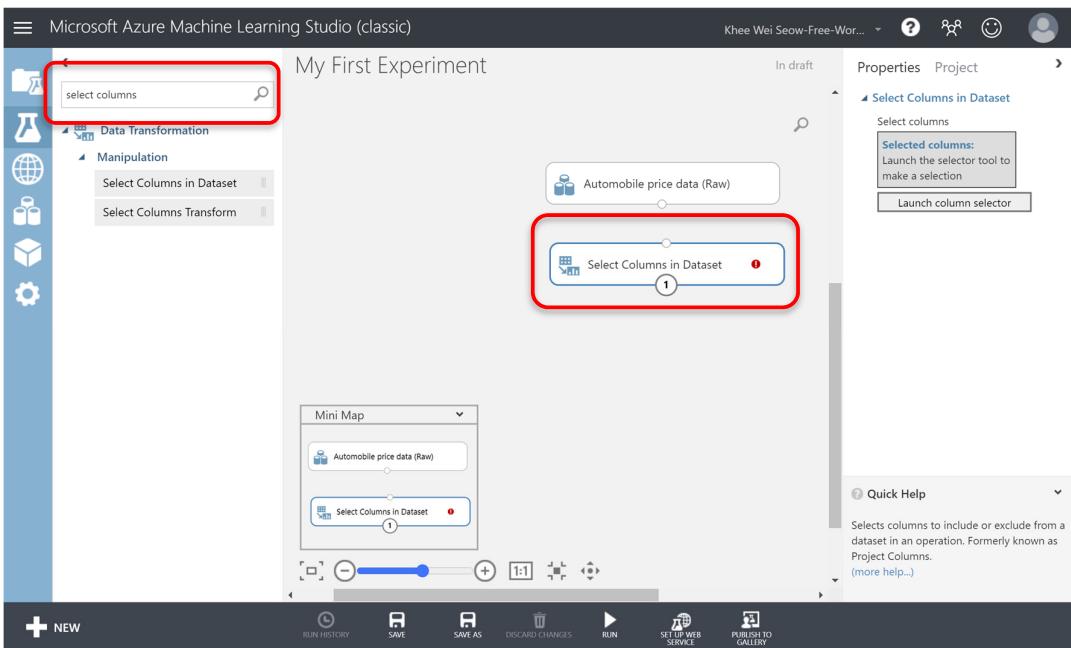
- 4) You should see the content of the dataset as shown below. Every column in the dataset is also known as a feature. Notice that some data (rows) have missing values.

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

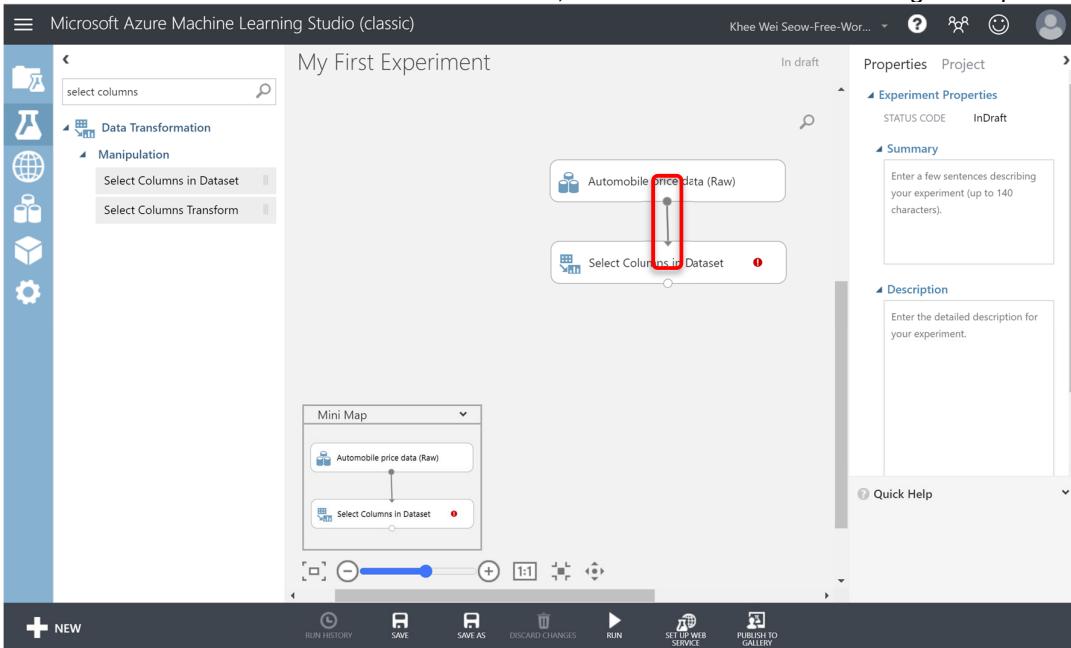
rows	columns
205	26
view as	
3	
1	
2	
2	
1	
1	
0	

4) Preparing your Dataset

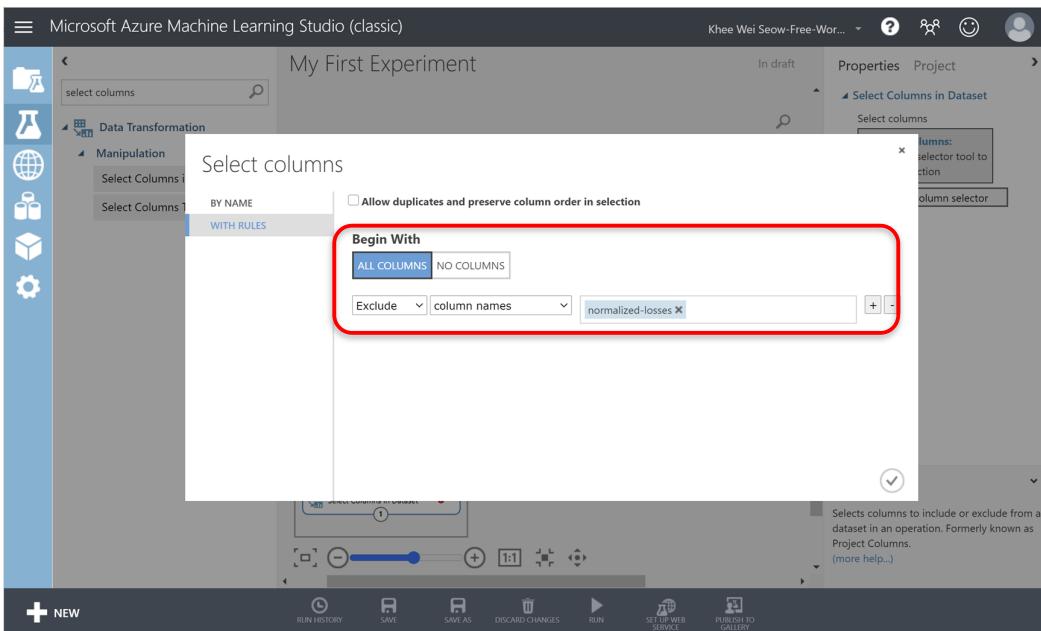
- 1) **Filter data.** In the search box, type **select column** and drag and drop the **Select Columns in Dataset** module onto the canvas. The **Select Columns in Dataset** module allows you to filter the dataset based on the specified column names.



- 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.

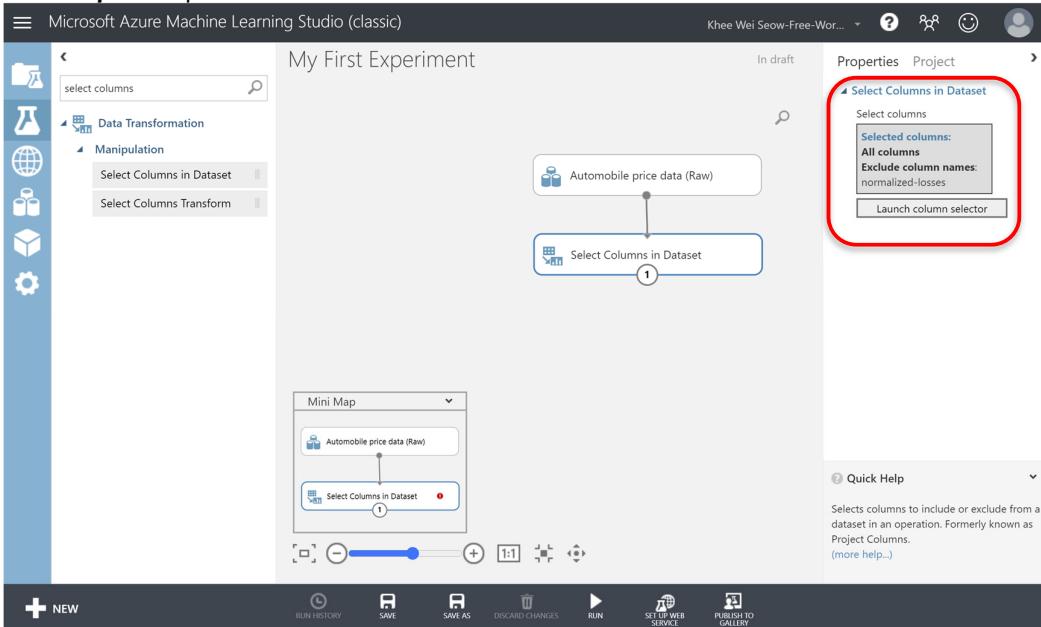


- 3) Select the **Select Columns in Dataset** module and on the Properties pane on the right, click the **Launch column selector** button.
- 4) Set the values as shown below. Click the check mark button when done.

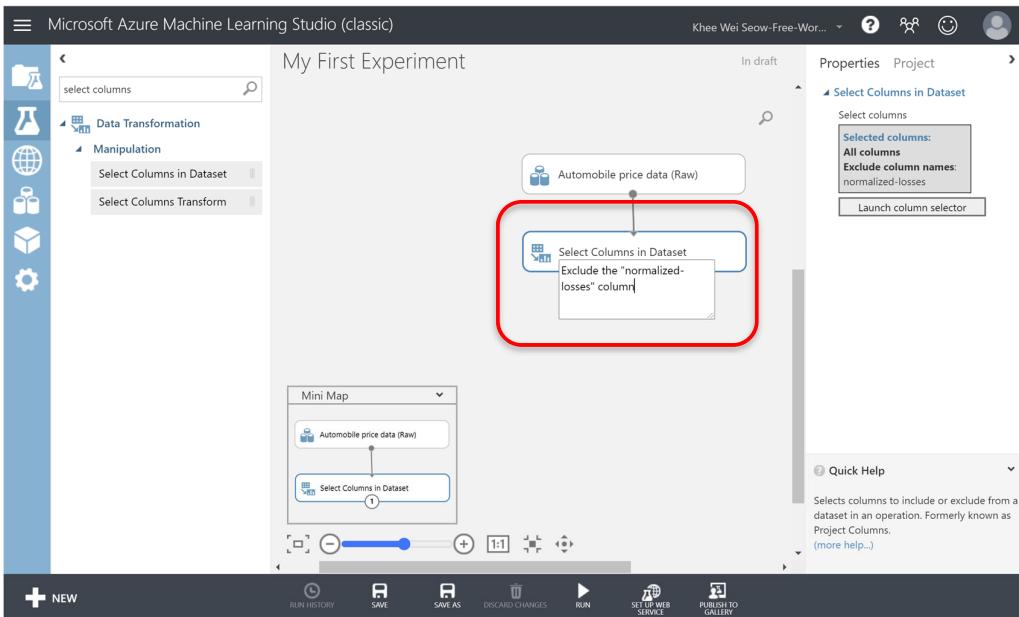


- This rule specifies that you want to exclude the normalised-losses column from the dataset.

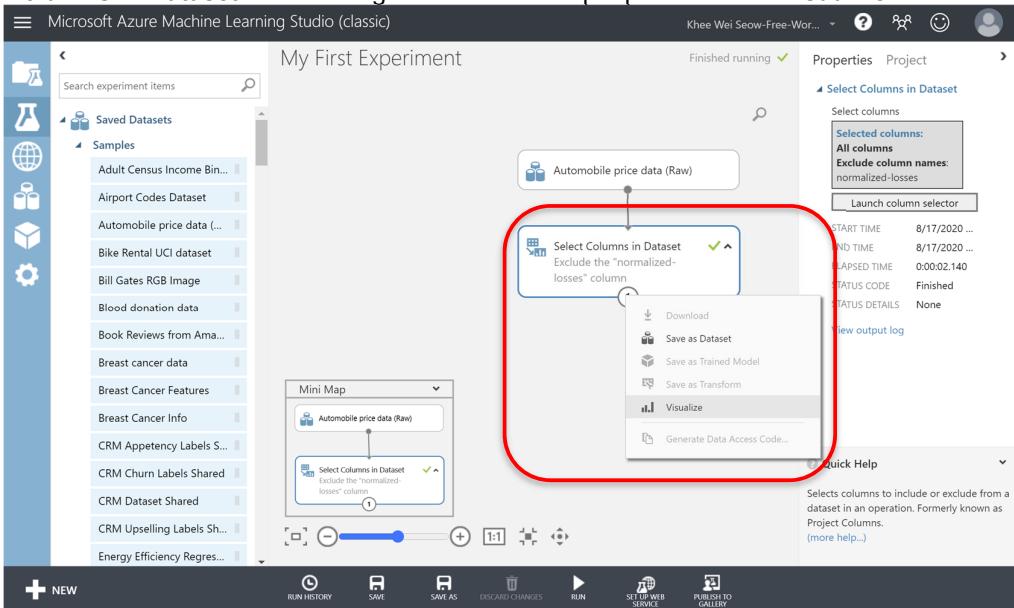
5) The **Properties** pane should now look like this:



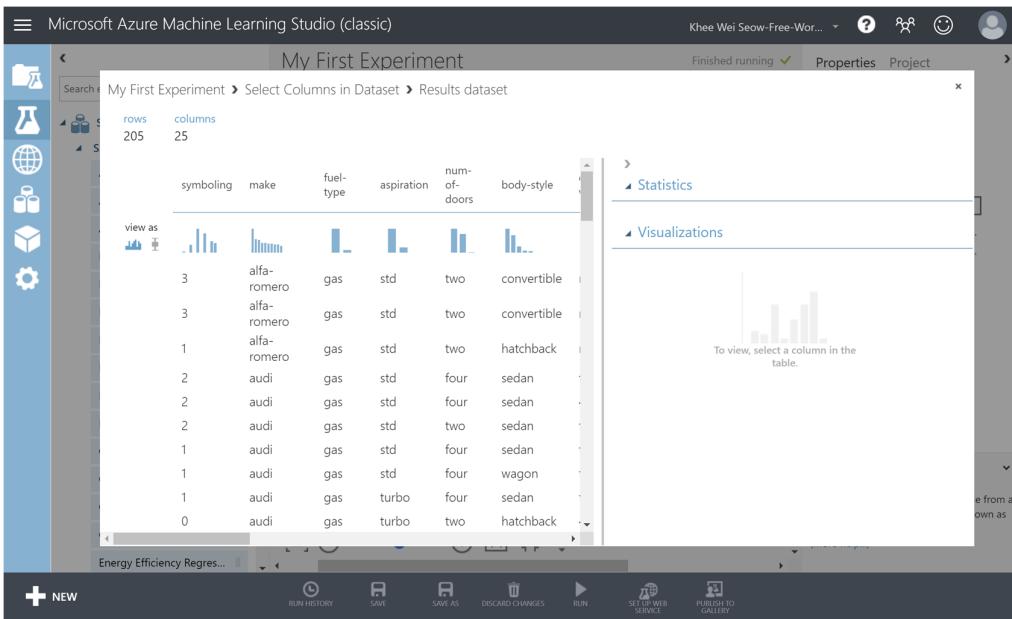
6) Double click on the **Select Columns in Dataset** module to add a comment.



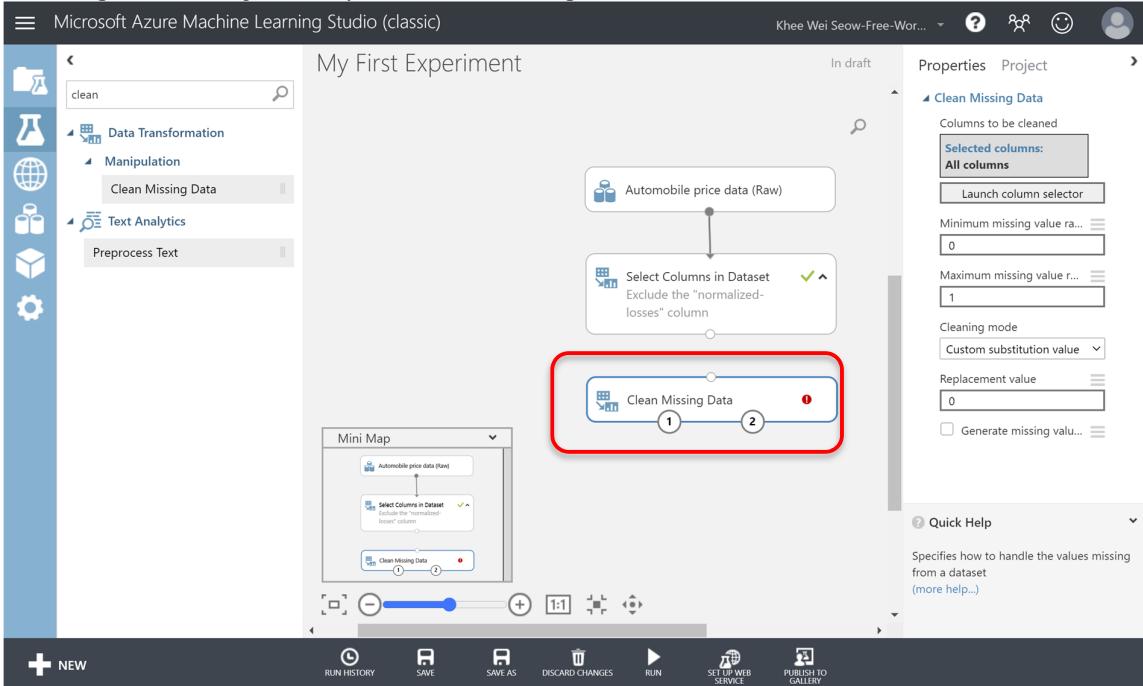
- 7) Click **Run** button located at the bottom of the screen. You will now see a green mark displayed in the **Select Columns in Dataset** module. Right-click on the output port and click **Visualize**.



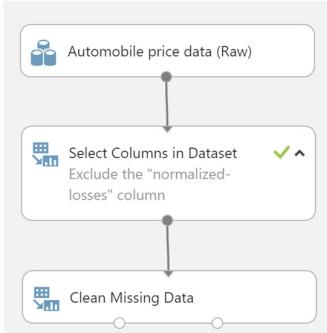
- 8) You should now see that the normalized-losses column is no longer in the dataset.



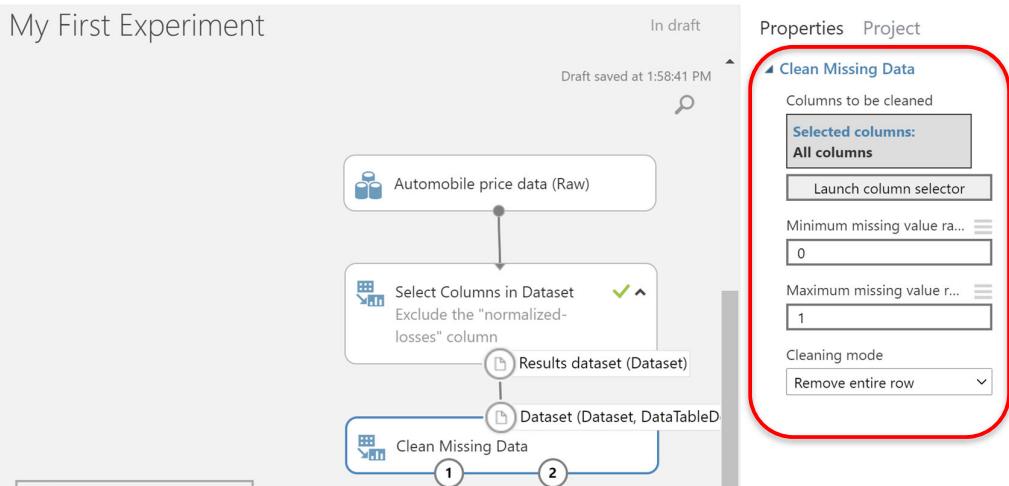
9) Cleaning Data – Drag and drop the **Clean Missing Data** module to the canvas.



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

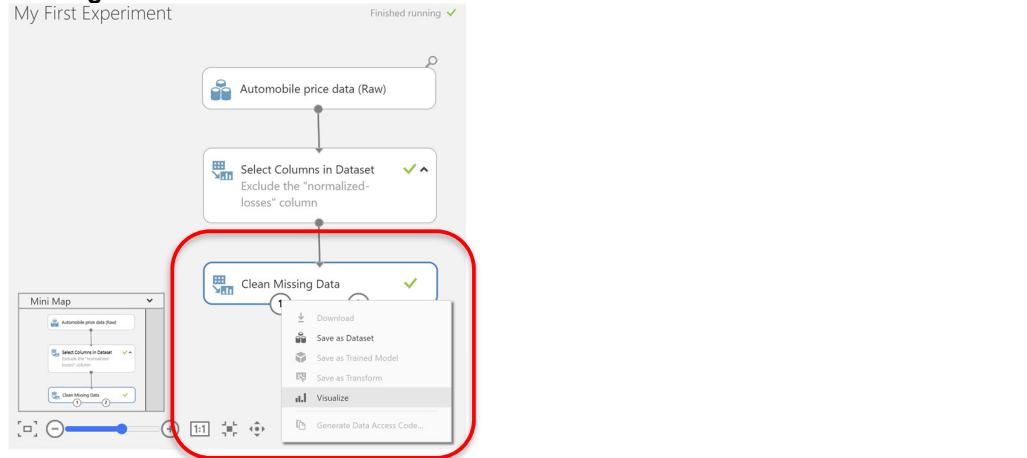


11) Select the **Select Columns in Dataset** module and set its **properties** as follows:



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>

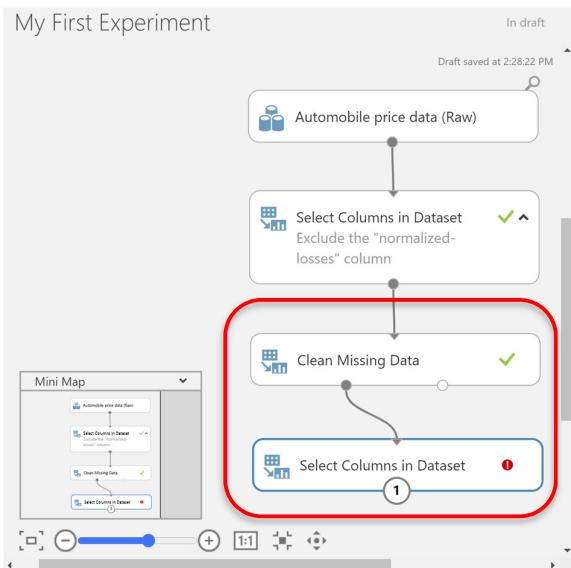
- 12) Click **Run**. Wait for a couple of seconds and you should see a green tick. Click on the left output of the **Clean Missing Data** module and click **Visualize**.



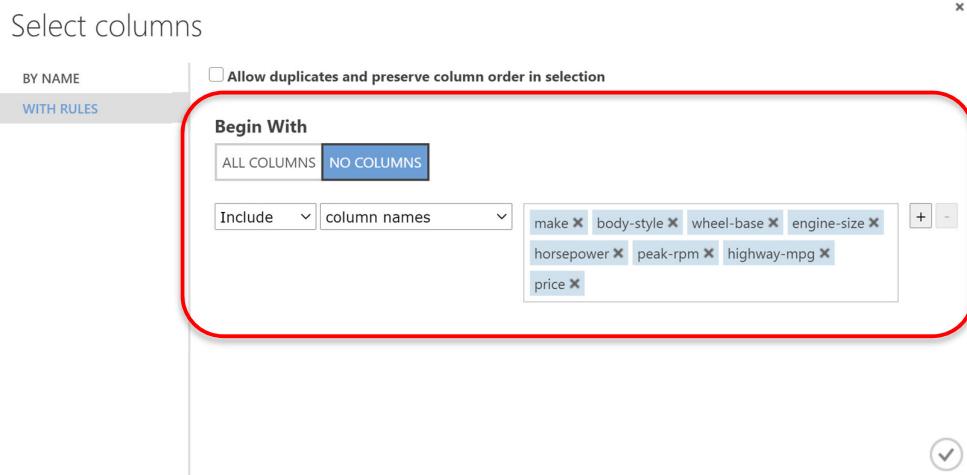
- 13) The dataset now has removed all rows with missing values:

	rows	columns
	193	25

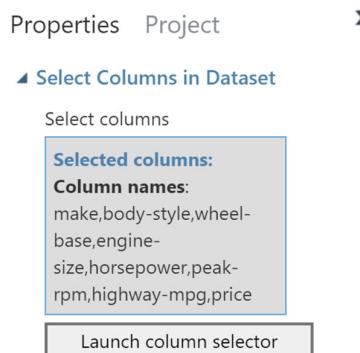
- 14) **Defining Features** – Add another **Select Columns in Dataset** module to the canvas and connect the **Clean Missing Data** module to it.



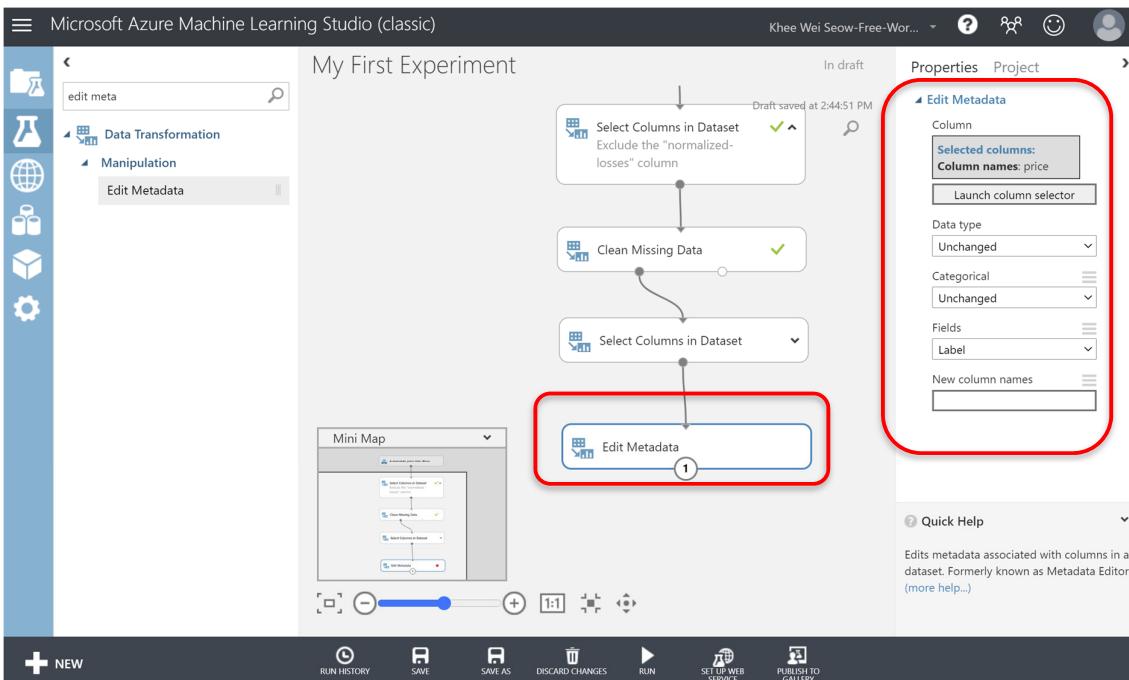
- 15) Double-click on the module and type **Select features for prediction**.
- 16) Select the module and in the **Properties** pane, click **Launch column selector**.
- 17) Set the values as shown below. Click on the check mark button when done.



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

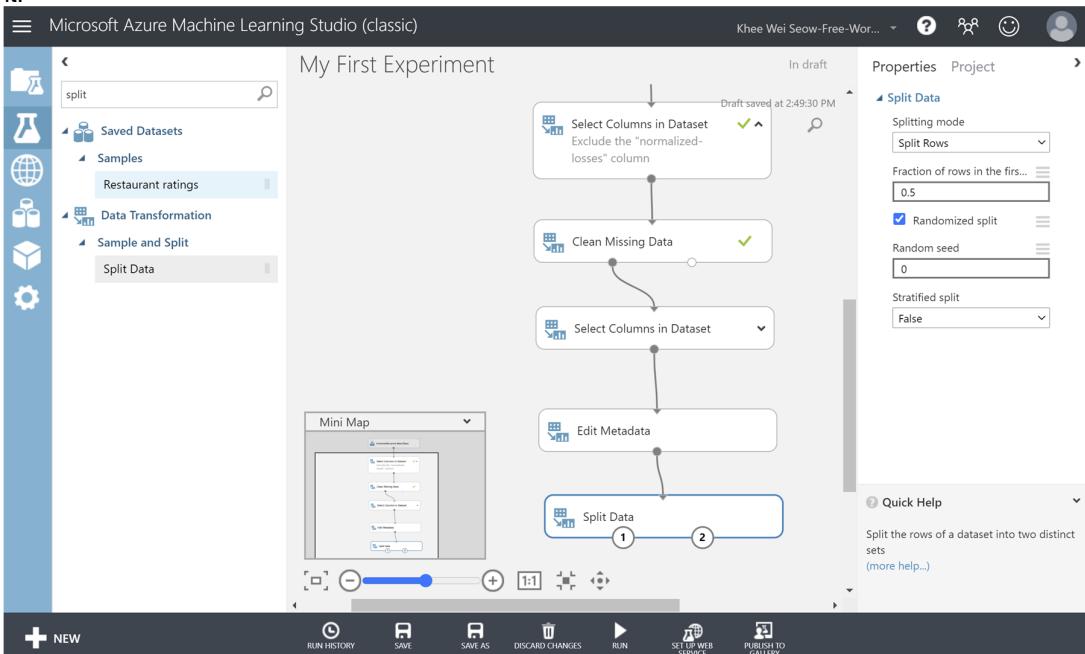


- 19) Labelling a Feature – Add the **Edit Metadata** module to the canvas and then connect and configure it as follows:



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.

- 20) 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.

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

Properties Project

Split Data

Splitting mode: Split Rows

Fraction of rows in the first part: 0.75

Randomized split

Random seed: 0

Stratified split: False

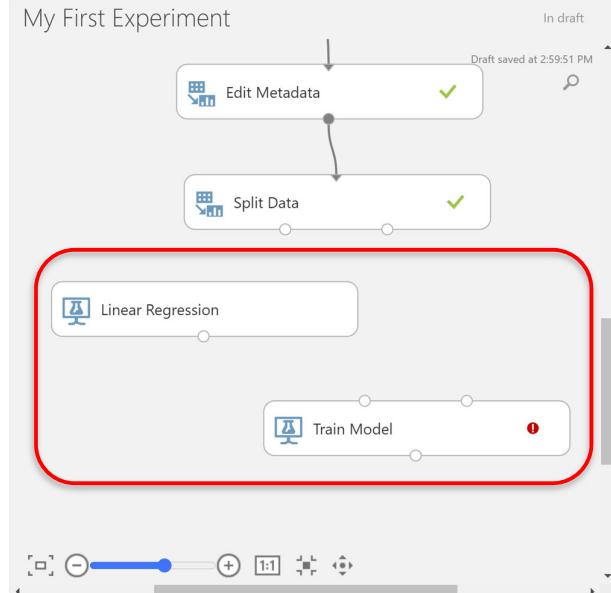
Ref: <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/split-data>

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

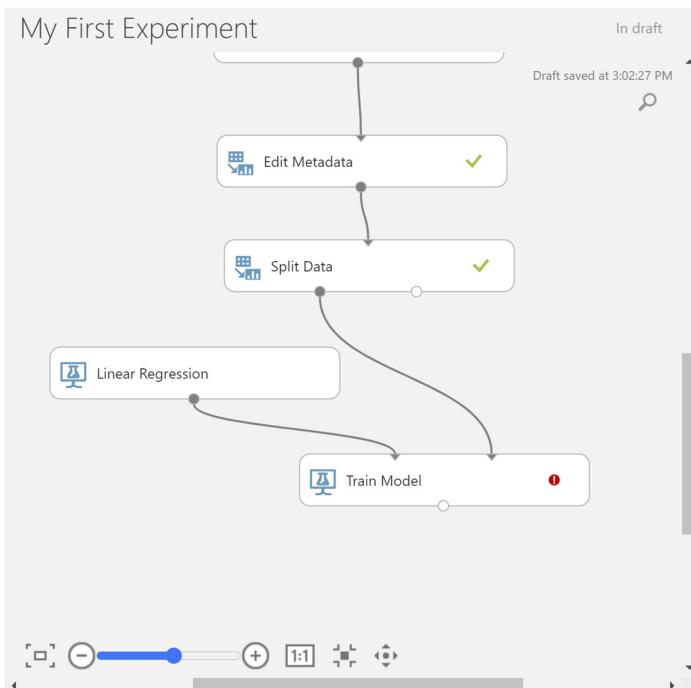
22) Click Run.

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

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

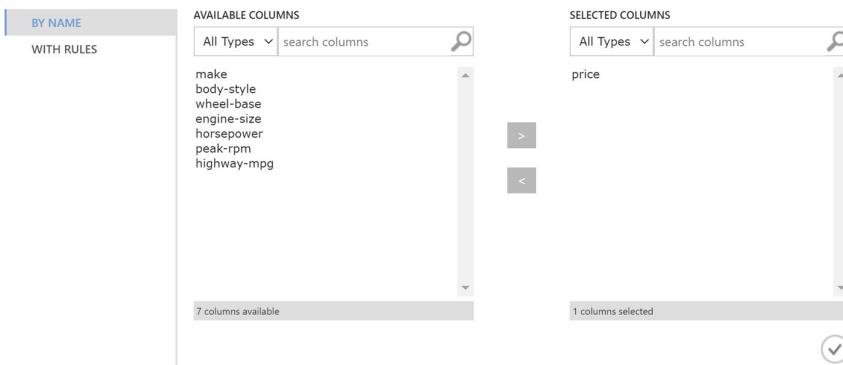


- Connect the **Linear Regression** module to the left input port of the **Train Model** module and the left output port of the **Split Data** module to the right input port of the **Train Model** module.



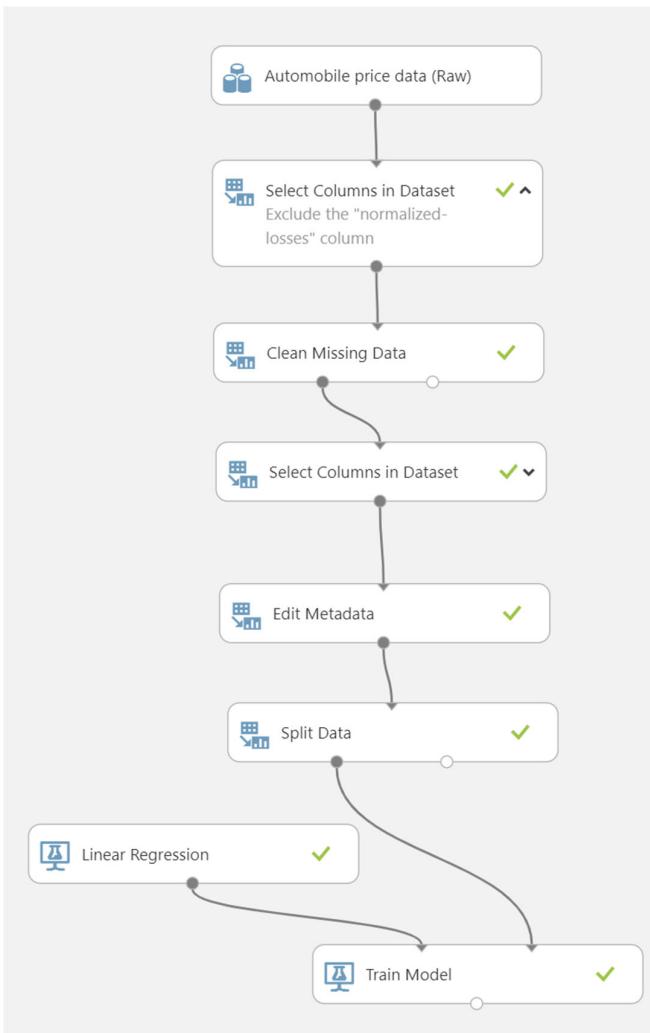
- 3) Select the **Train Model** module and in the **Properties** pane, click the **Launch column selector button**.

Select a single column

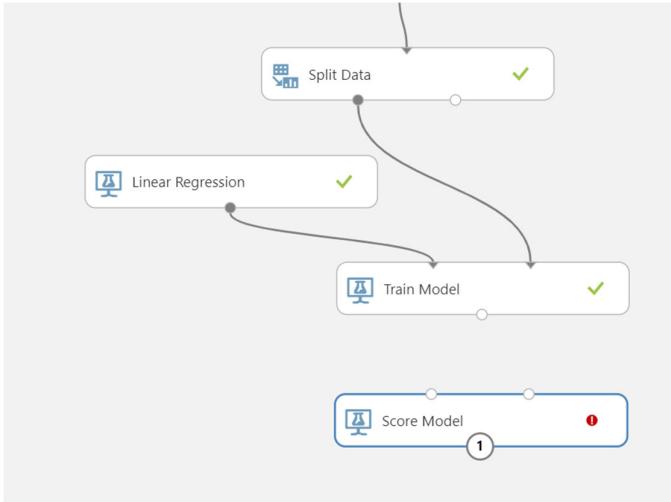


The above step indicates that you want the training model to predict the prices of vehicles.

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

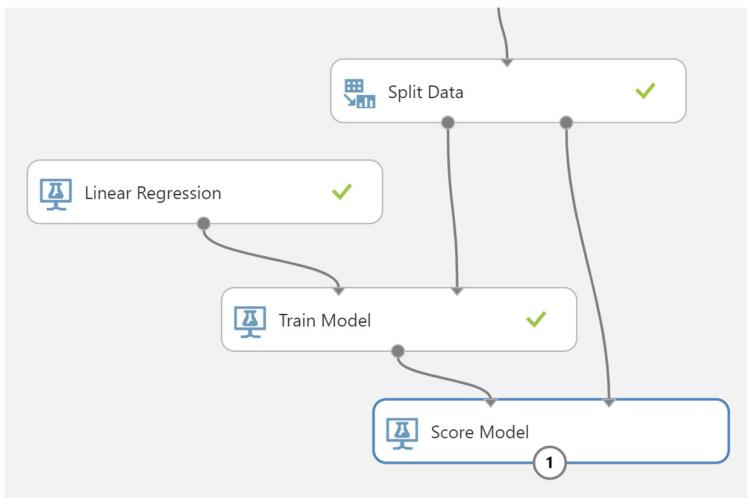


- 5) Add a **Score Model** module to the canvas:

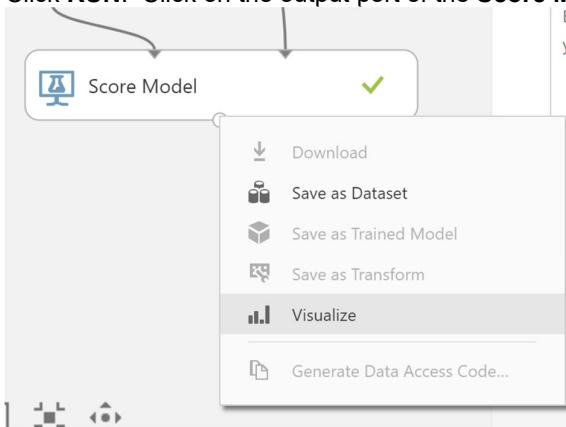


Ref: <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/score-model>

- 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 **Score Model** module to the right input port of the **Score Model** module:



- 7) Click **RUN**. Click on the output port of the **Score Model** and click **Visualize**:



- 8) You should see the following output.

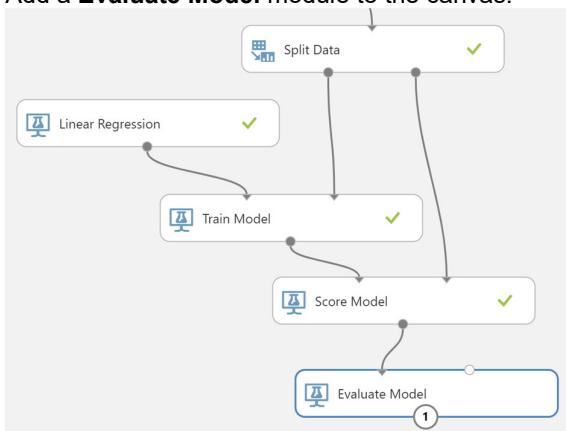
My First Experiment > Score Model > Scored dataset

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

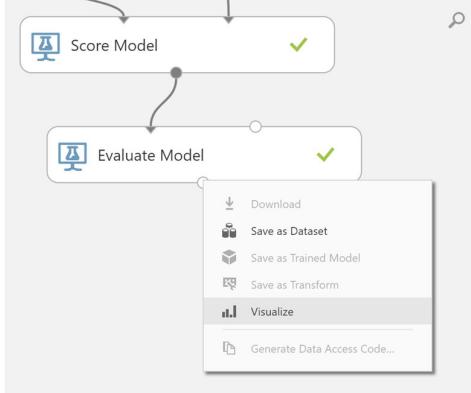
The price column shows the actual values and the Scored Labels column shows the predicted values.

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**:



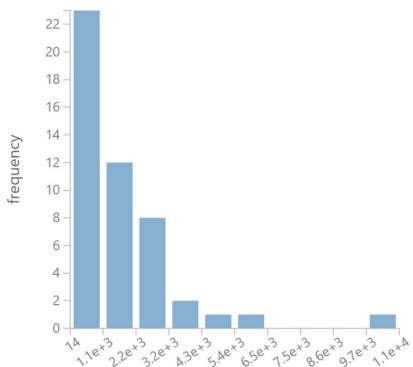
- 3) You should see the following:

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.

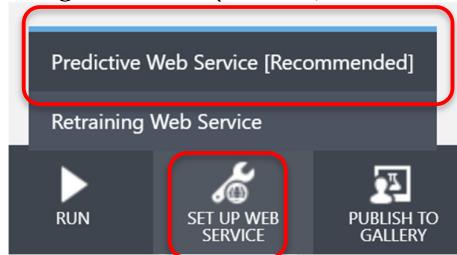
Activity 2 – Deploying your experiment as a Web Service

In this activity, we will learn:

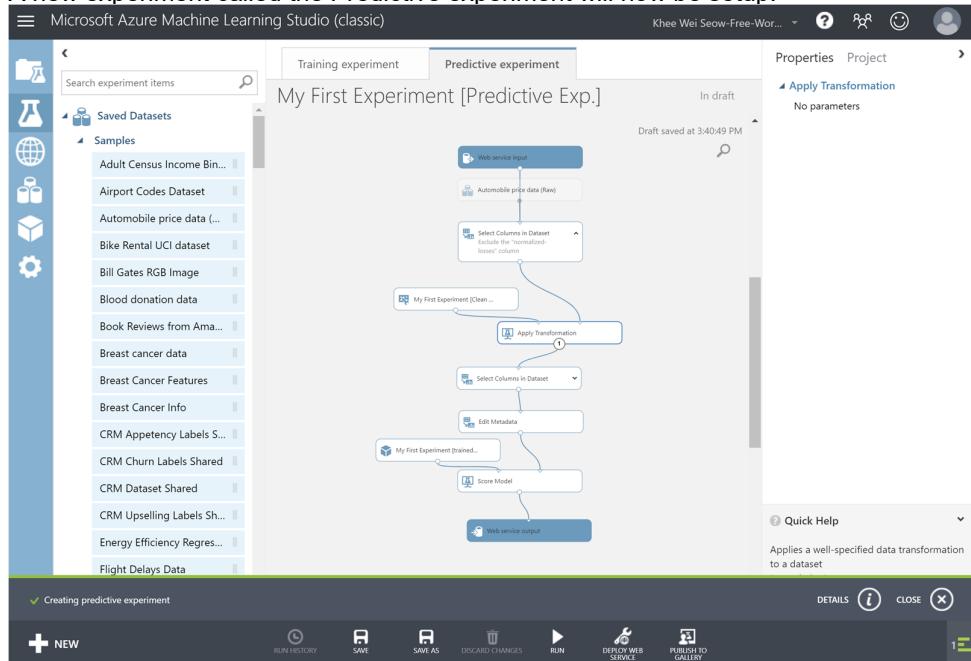
- Deploy a training model as a Web Service
- Test the webservice
- [Optional] access the web service using Python 3
- Use the web services via Excel

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.]

DASHBOARD CONFIGURATION

General New Web Services Experience [preview](#)

Published experiment

[View snapshot](#) [View latest](#)

Description

No description provided for this web service.

API key

IODK5GolyR97y0aMMWDtItgqamhDcE5JFevLLBTDbvYXiyO+zV3lXtwKampEMqa/pXX6xZPlZdHBVqjIHN6oQ==

Default Endpoint

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 Test preview	Excel 2013 or later workbook 8/17/2020 3:47:05 PM	

2) Testing the Web Service

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

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](#) [Test preview](#) Excel 2013 or later workbook 8/17/2020 3:47:05 PM

- 2) You should see the following:

Microsoft Azure Machine Learning Studio (classic) Web Services

Quickstart Dashboard Batch Request Log Configure Consume Test Swagger API

← My First Experiment [Predictive Exp.]

default

[View in Studio \(classic\)](#)

Request-Response Batch

Sample Data

Enable

input1 output1

Your prediction results will display here.

symboling: 1

normalized-losses: 1

make:

fuel-type:

aspiration:

- 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 Microsoft Azure Machine Learning Studio (classic) Web Services 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 of the Microsoft Azure Machine Learning Studio (classic) Web Services interface. The same form as the previous screenshot is shown, but the 'Test Request-Response' button has been clicked. On the right side of the screen, the 'Scored Labels' field is highlighted with a red box and contains the value '13498.4762334354'.

3) [Optional] Consuming the web service programmatically

- 1) Click the Consume link at the top of the page:

The screenshot shows the 'Consume' tab of the Microsoft Azure Machine Learning Studio (classic) Web Services interface. It displays 'My First Experiment [Predictive Exp.]' and the name 'default'. Below this, there is a section titled 'Web service consumption options' with three icons: 'Excel 2013 or later' (an icon of an X inside a building), 'Excel 2010 or earlier' (an icon of an X inside a document), and 'Request-Response Web App Template' (an icon of a circular network). To the right of these options is a green button labeled 'View in Studio (classic)'.

- 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.Net.Http.Formatting;
using System.Net.Http.Headers;
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>>() {

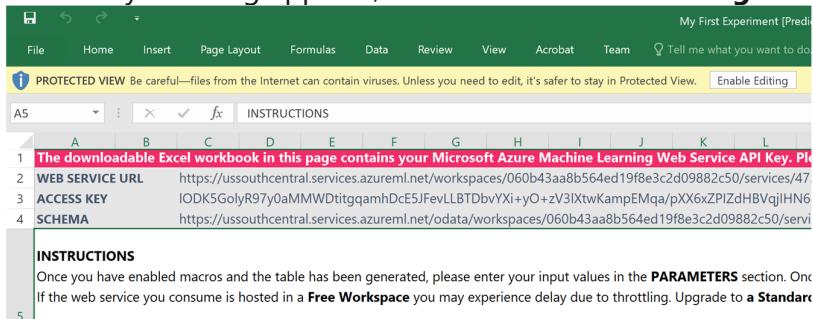
```

- 3) Click on Python 3+ tab and copy the code.
- 4) Make Prediction using Excel (before 2013)
 - 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.

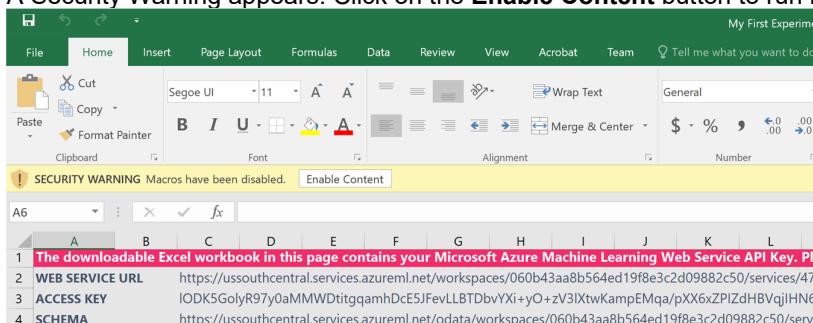
- 3) On the DASHBOARD tab for the web service is a row for the REQUEST/RESPONSE service. Click on **Excel 2010 or earlier workbook** the hyperlink to download the workbook in that row.

Default Endpoint	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

INSTRUCTIONS	
Once you have enabled macros and the table has been generated, please enter your input values in the PARAMETERS section. Once the web service you consume is hosted in a Free Workspace you may experience delay due to throttling. Upgrade to a Standard Workspace to avoid this issue.	
PARAMETERS	
Normalized-losses make-fuel-type aspiration num-of-doors body-style drive-wheels engine-location wheel-base length width height curb-weight engine-type num-of-cylinders	
symboling normalized-losses make fuel-type aspiration num-of-doors body-style drive-wheels engine-location wheel-base length width height curb-weight engine-type num-of-cylinders	
3 1 alfa-rgas std two convertible rwd front 88.6 168.8 64.1 48.8 2548 dohc four	
PREDICTED VALUES	
num-of-cylinders engine-size bore stroke compression-ratio horsepower peak-rpm city-mpg highway-mpg price make body-style wheel-base engine-size horsepower peak-rpm city-mpg highway-mpg price ScoredLabels	
four 130 mpgf 3.47 2.68 9 111 5000 21 27 13495 alfa-rc convertible 88.6 130 111 5000 27 13495 13498.4762 ScoredLabels	

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

PARAMETERS	
Normalized-losses	make-fuel-type aspiration num-of-doors body-style drive-wheels engine-location wheel-base length width height curb-weight engine-type num-of-cylinders
symboling normalized-losses make fuel-type aspiration num-of-doors body-style drive-wheels engine-location wheel-base length width height curb-weight engine-type num-of-cylinders	
3	1 alfa-rgas std two convertible rwd front 88.6 168.8 64.1 48.8 2548 dohc four
PREDICTED VALUES	
num-of-cylinders engine-size bore stroke compression-ratio horsepower peak-rpm city-mpg highway-mpg price make body-style wheel-base engine-size horsepower peak-rpm city-mpg highway-mpg price ScoredLabels	
four 130 mpgf 3.47 2.68 9 111 5000 21 27 13495 alfa-rc convertible 88.6 130 111 5000 27 13495 13498.4762 ScoredLabels	

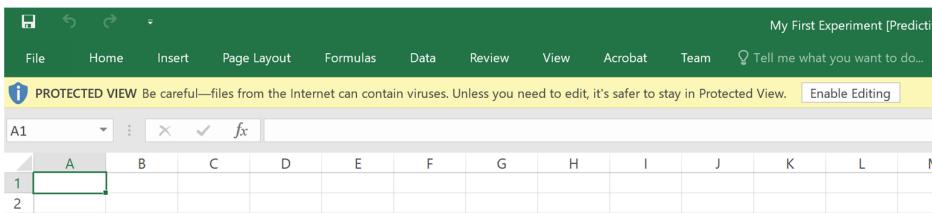
5) Make Prediction using Excel (after 2013)

- 1) 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 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

- 2) Open the sample Excel file, which contains the Excel add-in.
- 3) Click on **Enable Editing**.



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

Azure Machine Learning

Web Services

My First Experiment [Predictive Exp.]

+ Add Web Service

Auto-predict **Predict All**

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

← My First Experiment [Predictive Exp.]

1. VIEW SCHEMA

2. PREDICT

Input: input1

Type range or click button to select

My data has headers

Use sample data

Output: output1

Enter output cell (e.g. A20)

Include headers

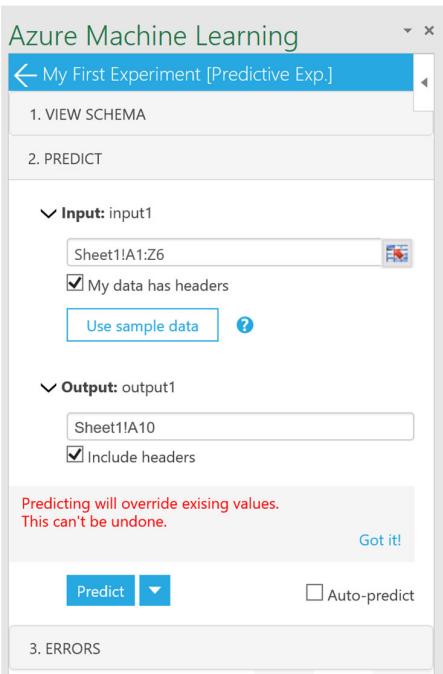
Predicting will override existing values.
This can't be undone.

Got it!

Predict Auto-predict

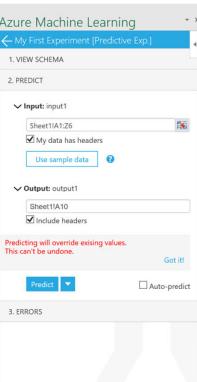
3. ERRORS

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



- 8) 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.
- 9) 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-wheel	engine-location	wheel-base	length
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											



Activity 3 – 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.

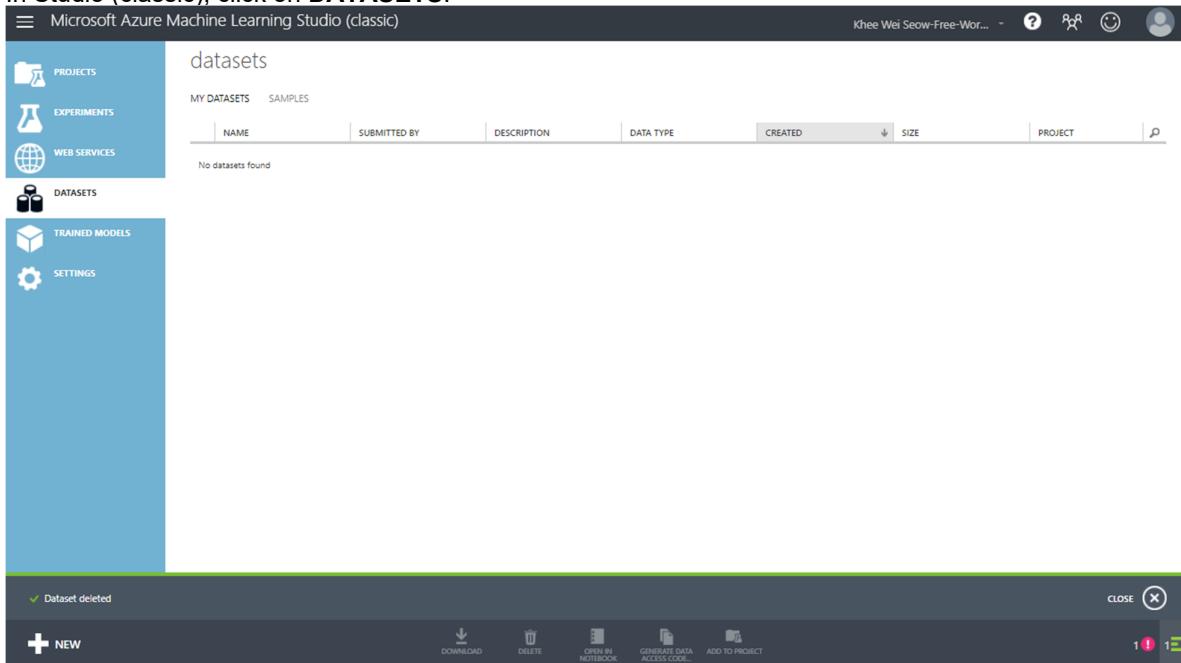
Online data sources - Use the Import Data module to access data from one of several online sources while your experiment is running

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>

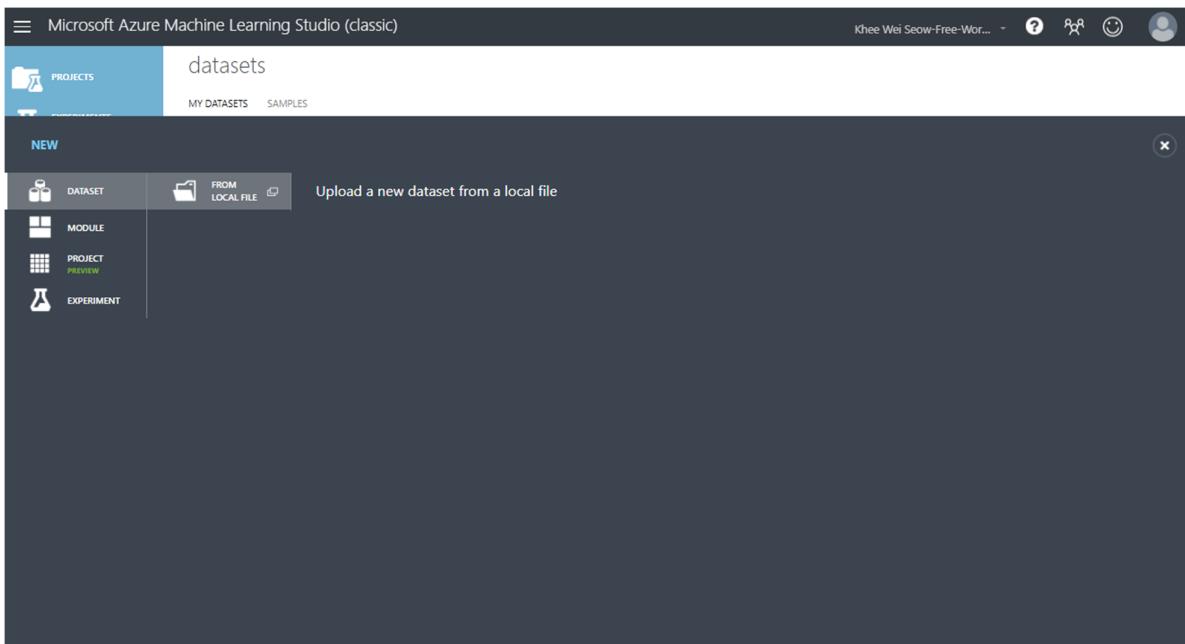
SQL Server database - Use data from a SQL Server database without having to copy data manually

- 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.

Upload a new dataset

SELECT THE DATA TO UPLOAD:
 Iris.csv

This is the new version of an existing dataset

ENTER A NAME FOR THE NEW DATASET:

SELECT A TYPE FOR THE NEW DATASET:

PROVIDE AN OPTIONAL DESCRIPTION:

(✓)

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

Microsoft Azure Machine Learning Studio (classic)

Khee Wei Seow-Free-Wor... ? ⚙️ 😊 🧑

datasets

MY DATASETS SAMPLES

NAME	SUBMITTED BY	DESCRIPTION	DATA TYPE	CREATED	SIZE	PROJECT
Iris.csv	kwseow		GenericCSV	8/18/2020 9:40:23 PM	4.99 KB	None

Upload of the dataset 'Iris.csv' has completed.

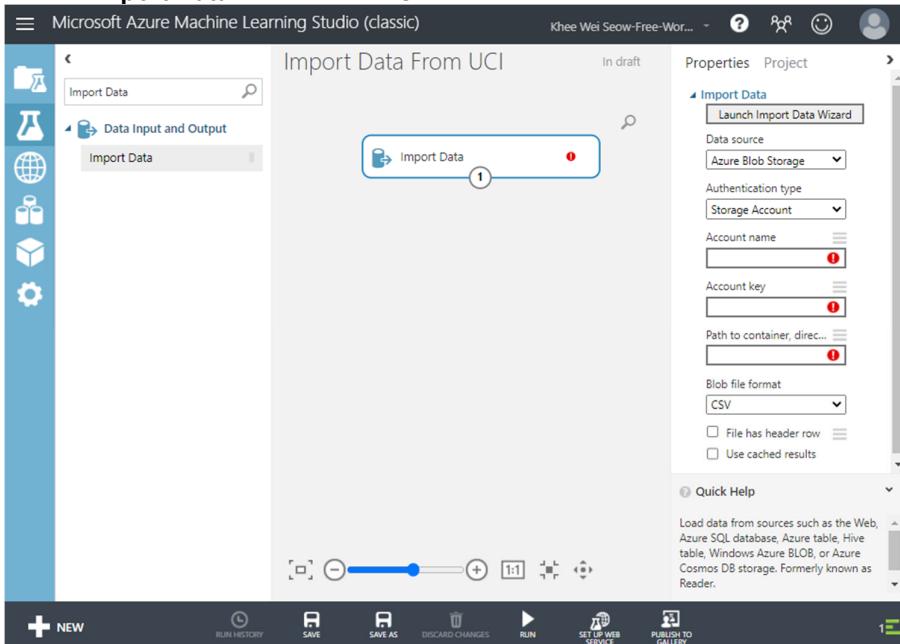
OK (✓)

NEW DOWNLOAD DELETE OPEN IN NOTEBOOK GENERATE DATA ACCESS CODE ADD TO PROJECT

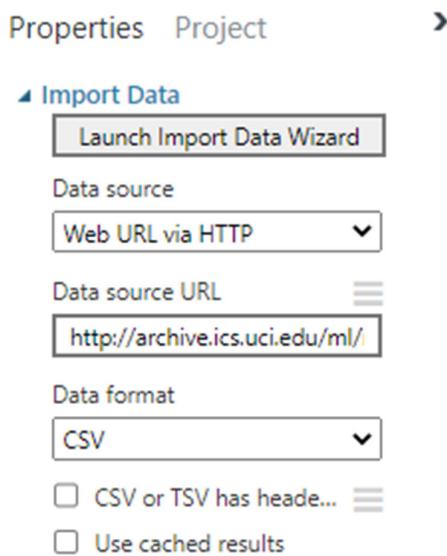
1 ! 12

3) Import directly from the web

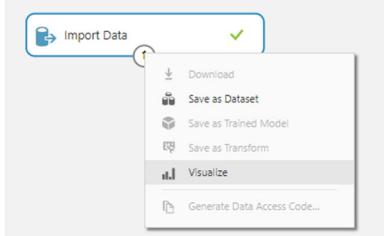
- 1) Create a new **Black Experiment** and name it Import dataset from UCI.
- 2) Add an **Import Data** module to the Canvas



- 3) Set the properties of the Import Data module as follows:



- 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	32562	columns	15								
	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11
view as											
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-	Other-service	Not-in-family	Black	Female	0	

x

Statistics and Visualizations

Activity 4 – Cleaning and Structuring Data

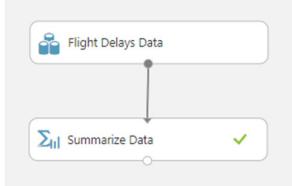
In this activity, we will learn:

- 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**.

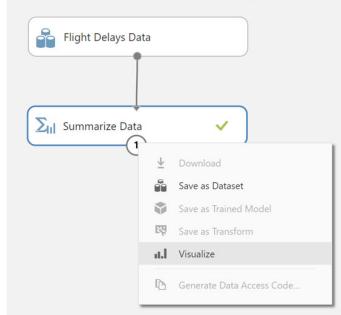
1) Summarizing Data

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

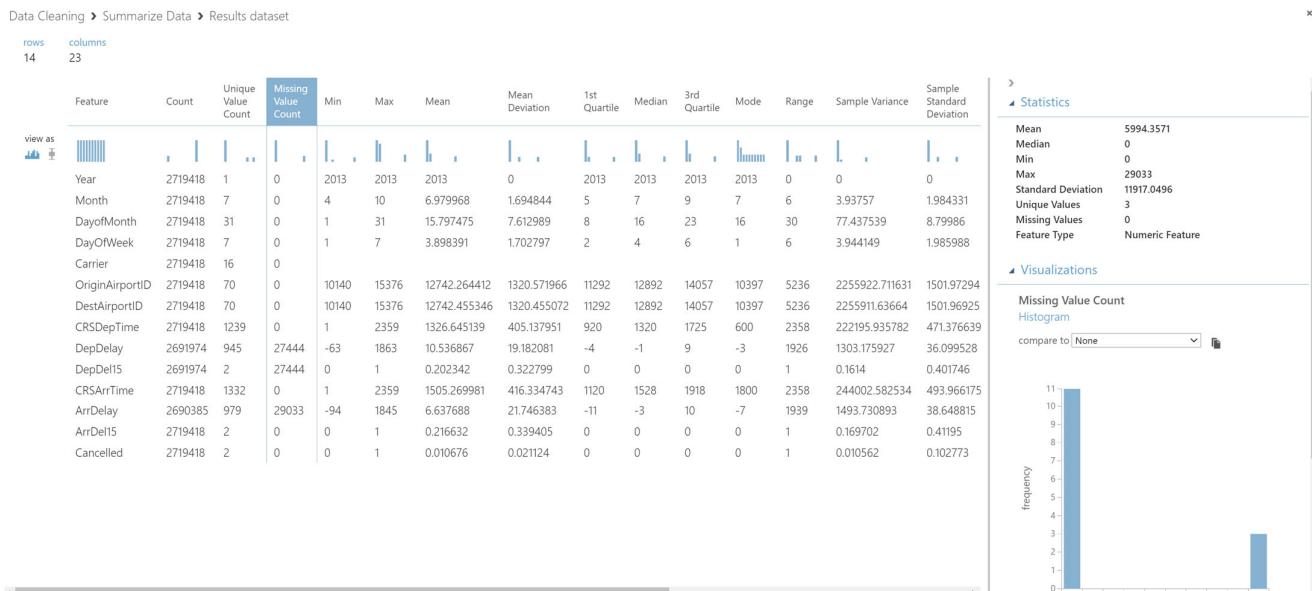


- 2) Click Run,

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

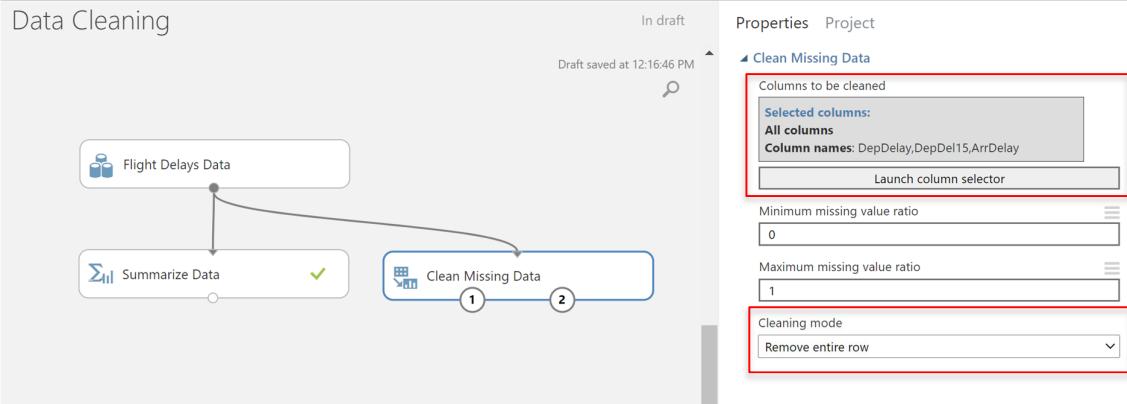


- 4) You should see a similar screen (below) that shows a summary of the dataset. The column “**Missing Value Count**” will tell you which feature(s) have missing data.

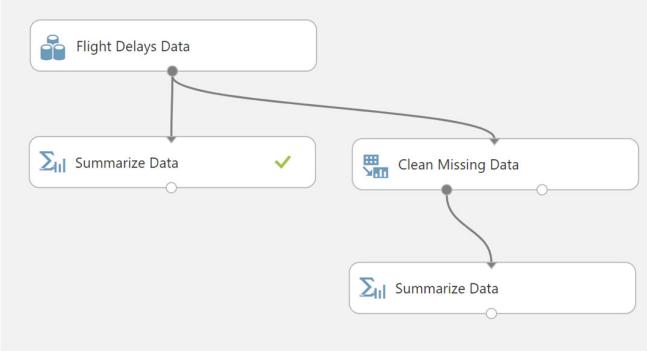


2) Missing Value

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



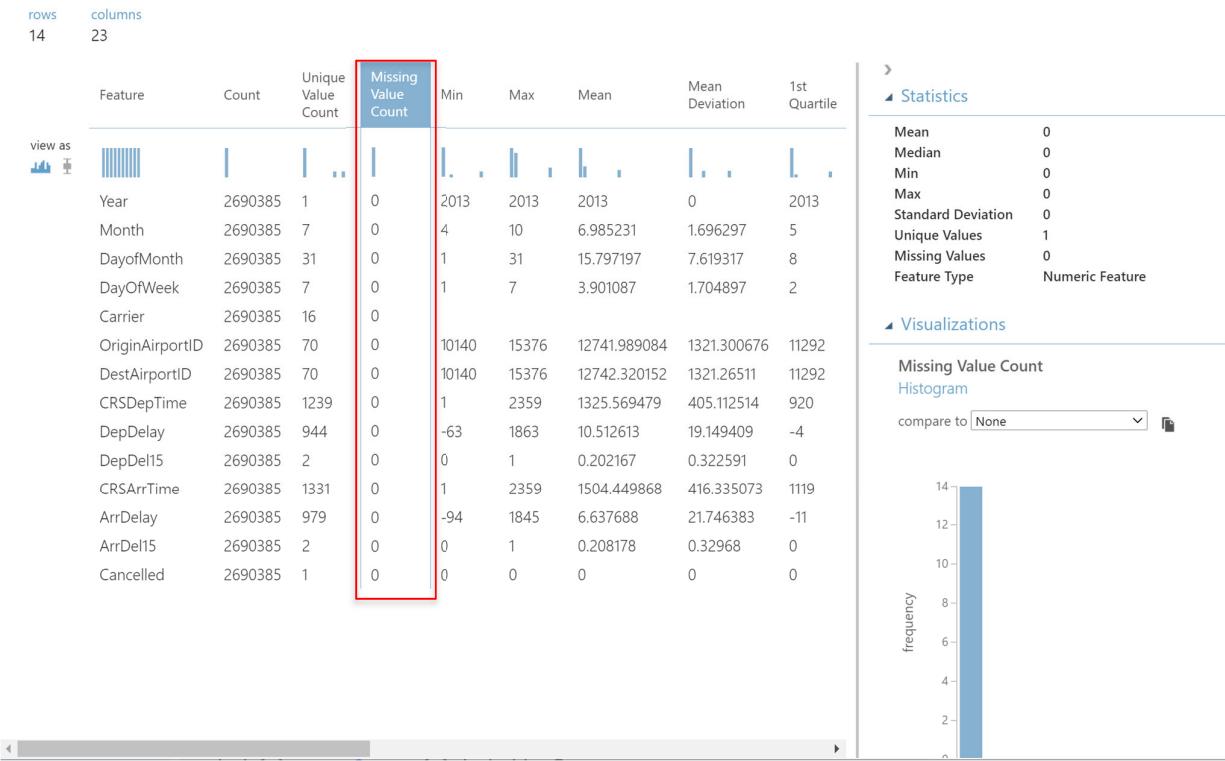
- 2) Add a **Summarize Data** module to the canvas as follows:



- 3) Click on **Run**

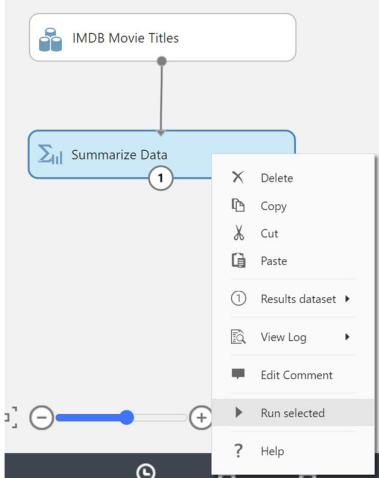
- 4) Visualise the output of the newly added **Summarize Data** module. There should be no missing values in the dataset now.

Data Cleaning > Summarize Data > Results dataset

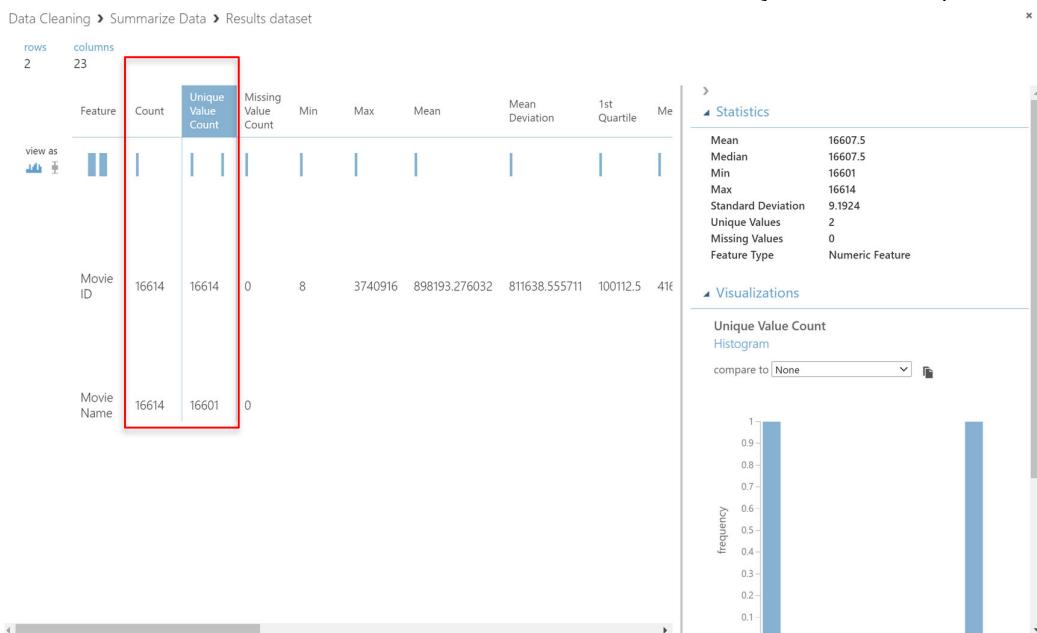


3) Duplicate Records

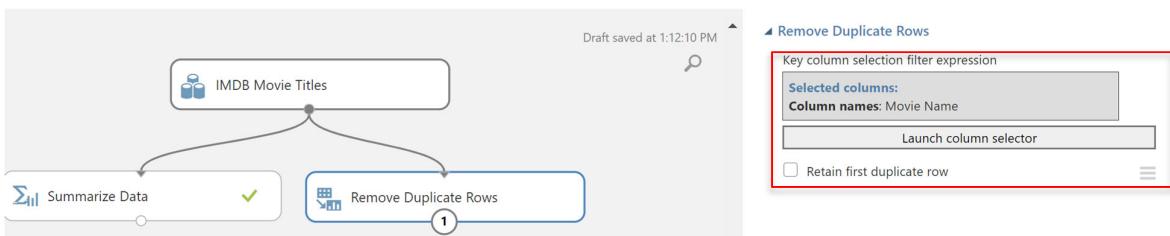
- 1) Add the **IMDB Movie Titles** and **Summarize Data** module to the canvas.
- 2) Right-click on the **Summarize Data** module and select **Run selected**. This action will only run the selected module rather than the whole experiment saving some time.



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

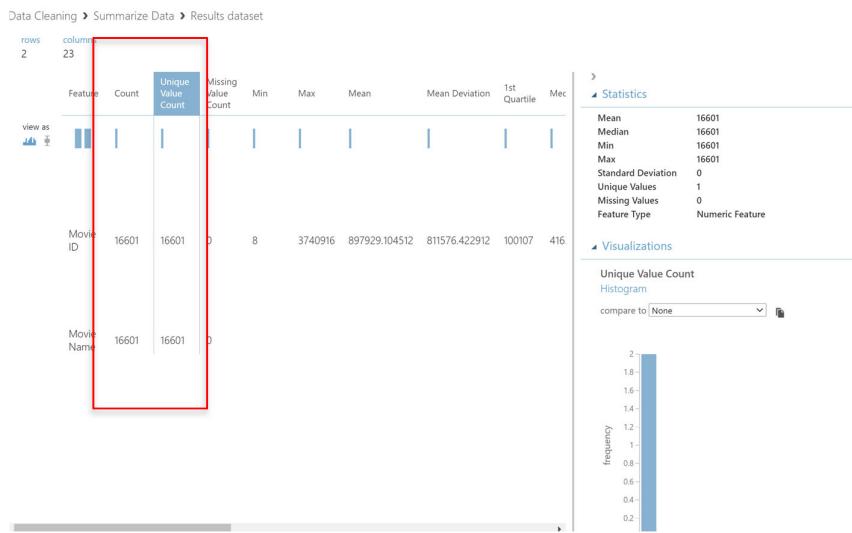


- 4) Add a **Remove Duplicate Rows** module to the canvas, connect and configure it as follows:



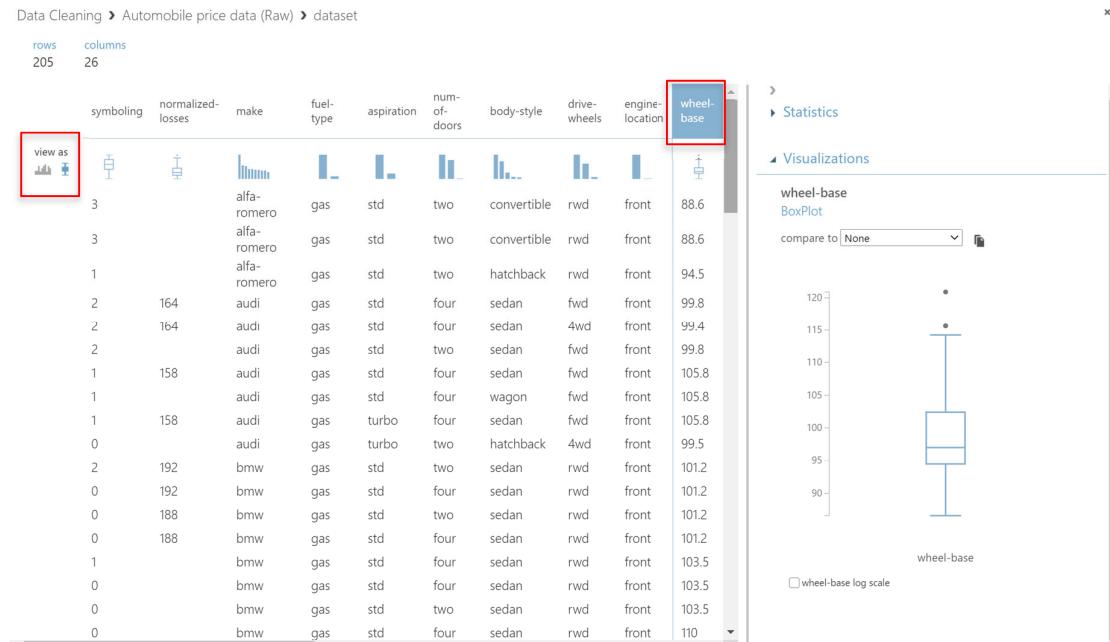
Note: Use the Retain first duplicate row checkbox to indicate which row to return when duplicates are found:
 If selected, the first row is returned and others discarded.
 If you uncheck this option, the last duplicate row is kept in the results, and others are discarded.

- 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.



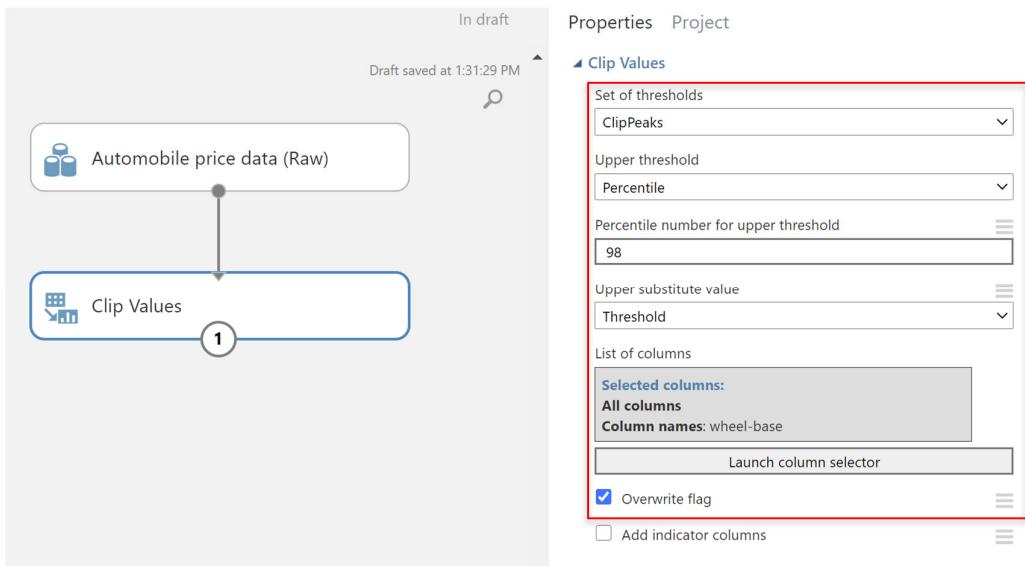
4) Removing Outliers

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



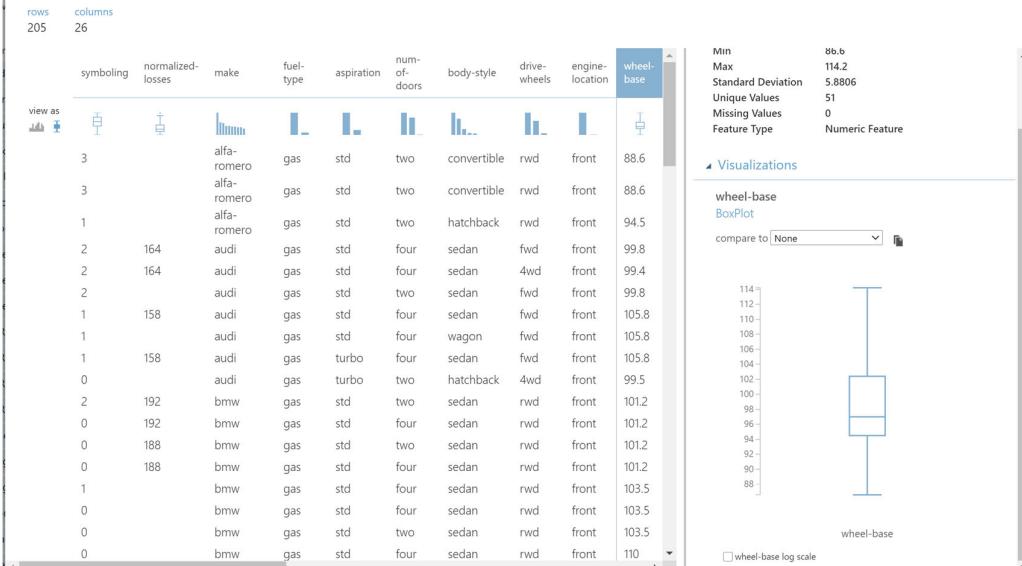
Note: The dots outside the boxplot are the outliers, and they should be removed from the dataset as they may affect the accuracy of the prediction.

- 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**.

Data Cleaning > Clip Values > Results dataset



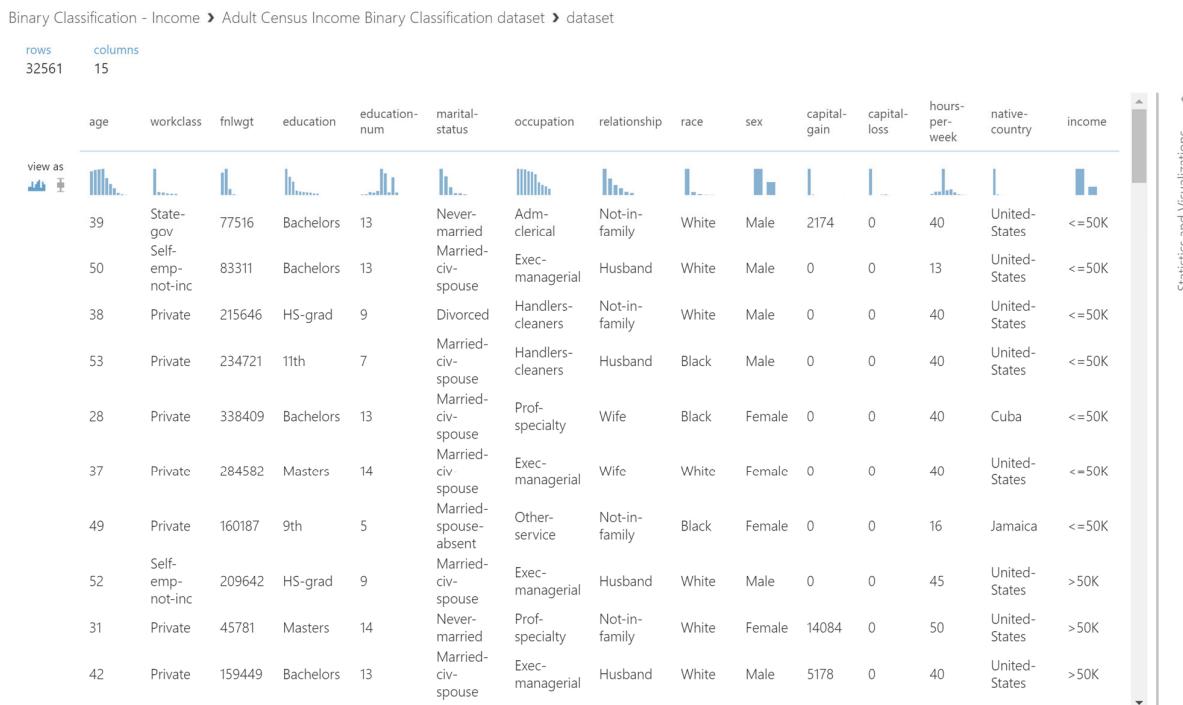
Note: If the outliers are still present, adjust the **Percentile number for upper threshold** property of the **clip-values** module

Activity 5 – Using Binary Classification Algorithms

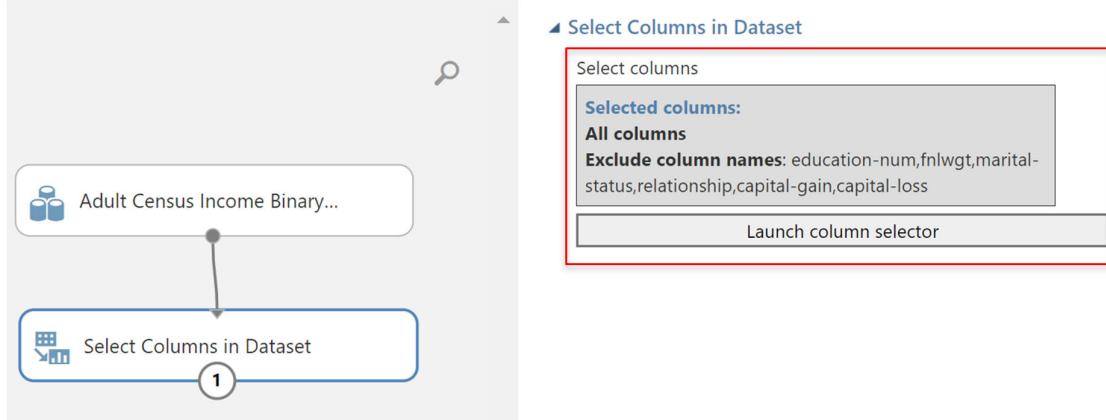
In this activity, we will learn:

- 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

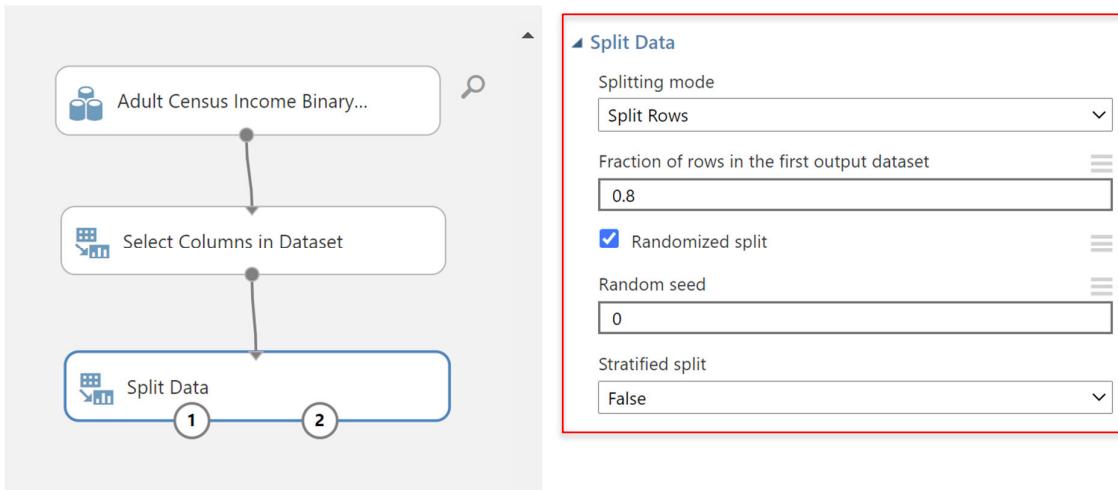
- 1) Create a new experiment and name it as **Binary Classification – Income**.
- 2) Add the **Adult Census Income Binary Classification** dataset to the canvas.
- 3) Right-click on the output port of the dataset and select Visualize. You should see the following:



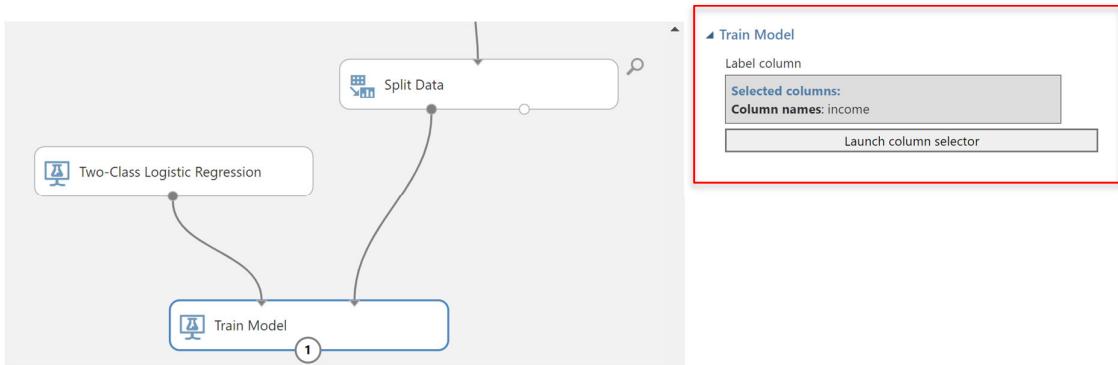
- 4) Add the **Select Columns in Dataset** module to the canvas and connect and configure it as follows:



- 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:

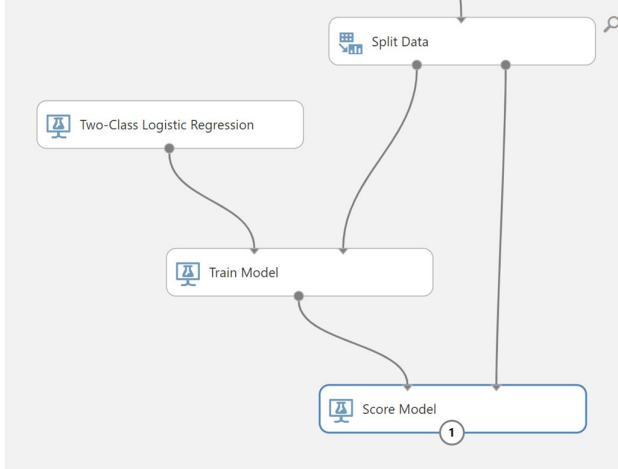


Ref:

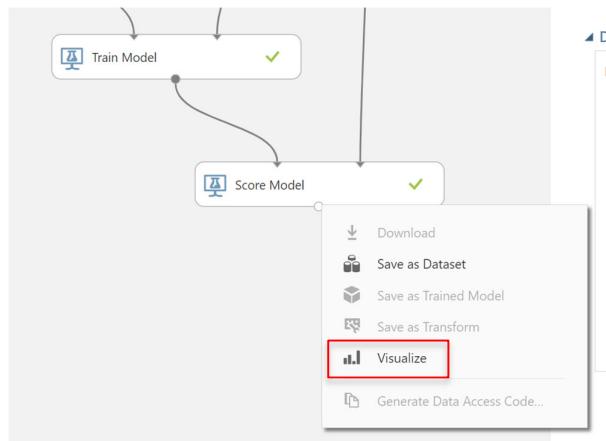
<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-logistic-regression>
<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/train-model>

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:

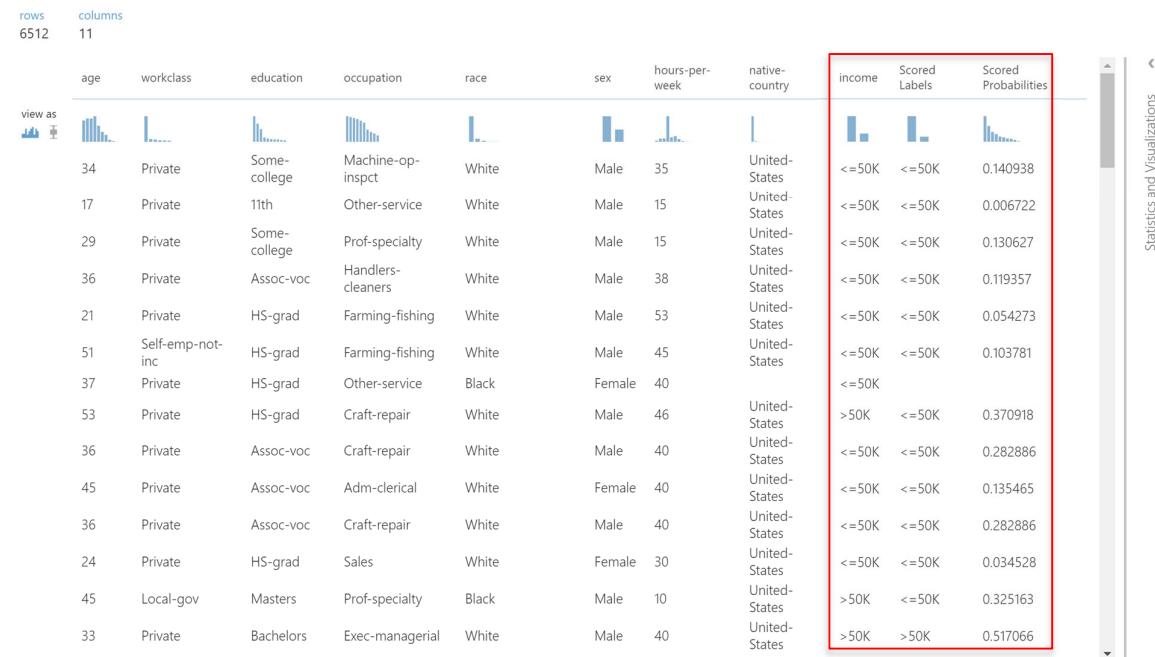


- 8) Click **Run**. When it is completed, right-click on the output port and select **Visualize**:



9) You should see the following:

Binary Classification - Income > Score Model > Scored dataset



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:



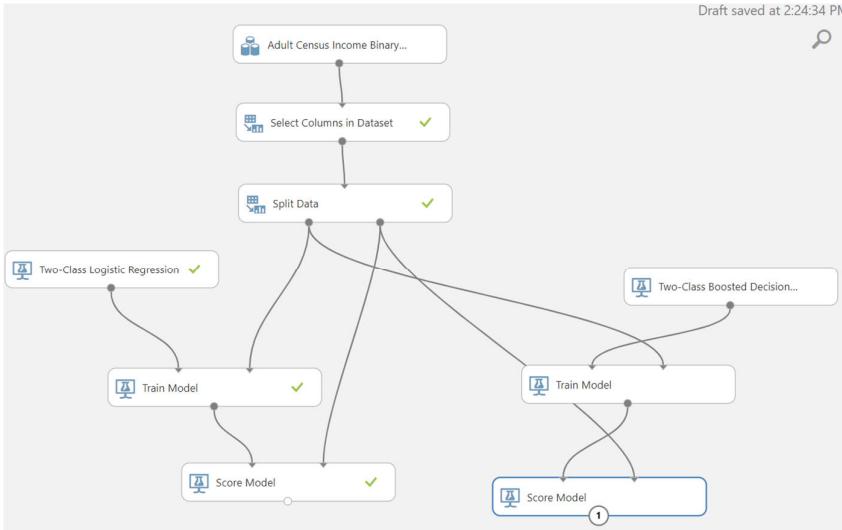
Ref:

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

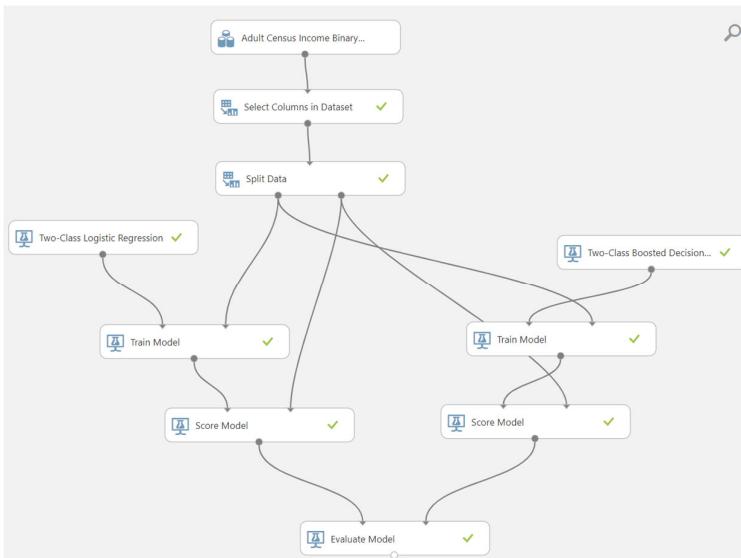
Note:

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:



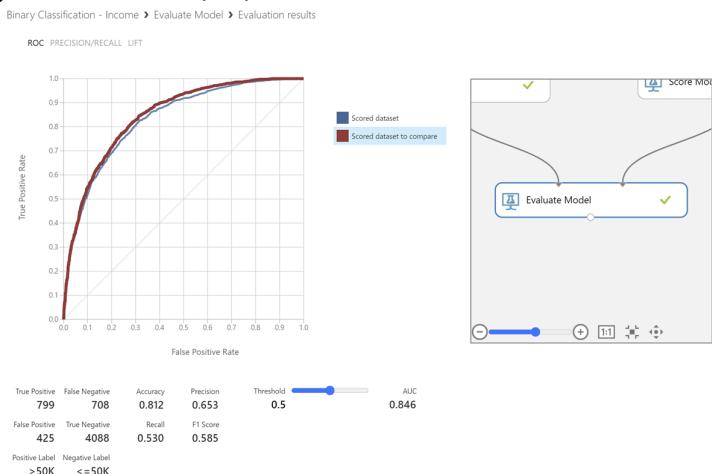
- 12) Add the **Evaluate Model** module to the canvas and connect it as follows:



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.

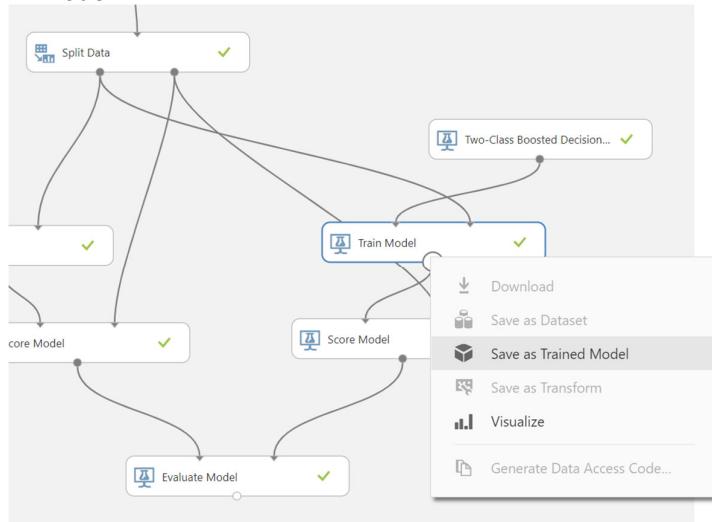
Evaluation metrics

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.

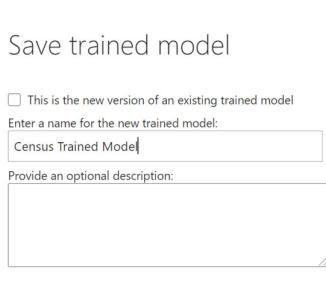
15) Saved a train model

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

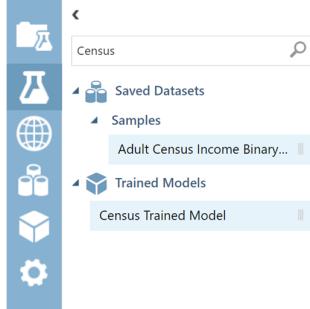


Note: Once a **Train Model** is trained, you can save it as a trained model so that you can later use it on the canvas without specify the algorithm.

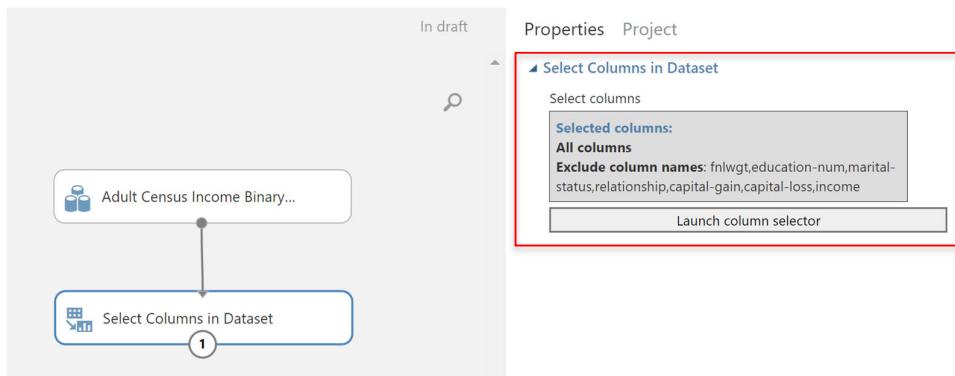
- Name the trained model as **Census Trained model**:



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

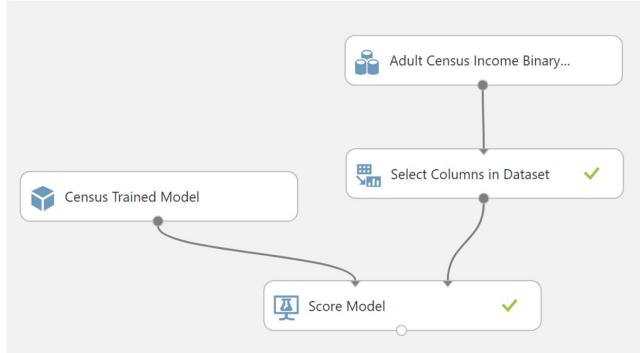


- Create a new experiment and name it as **Income Prediction**.
- Add the **Adult Census Income Binary Classification** dataset to the canvas.
- Add the **Select Columns in Dataset** module to the canvas and connect and configure it as follows:



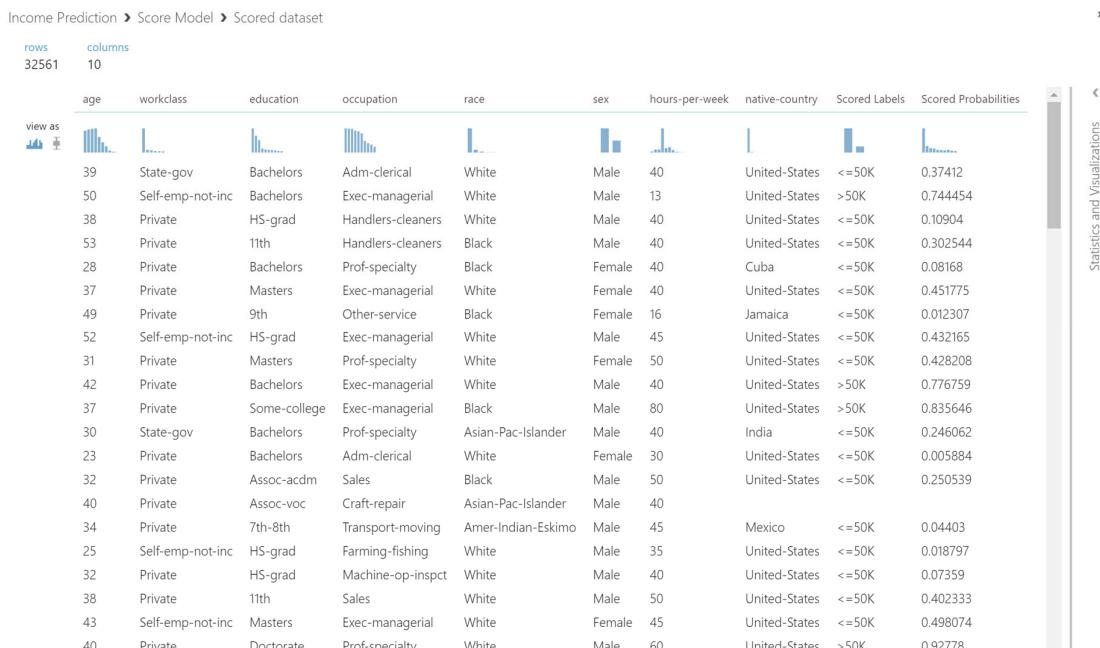
Note: We have excluded **income** column in this case. Our model is going to predict income, so there is no need to have the income as part of the inputs.

- 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**:



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.

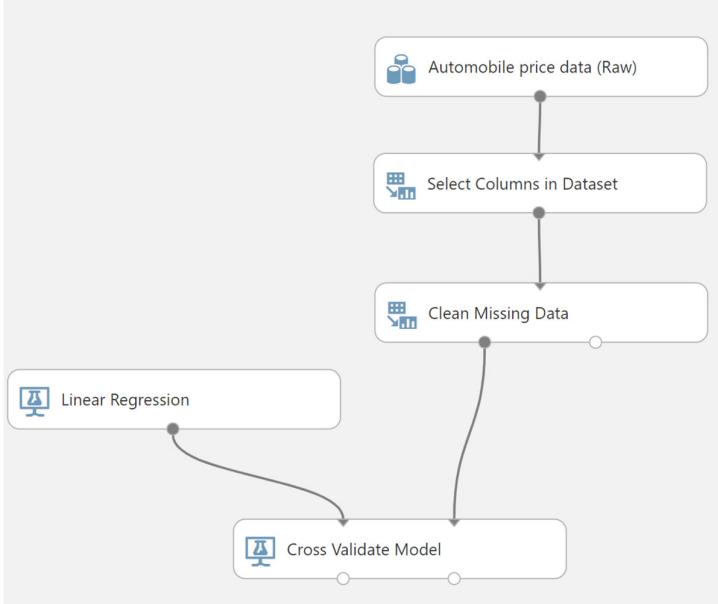
Activity 6 – Evaluating a Regression Model with Cross Validation

In this activity, we will learn:

- How to use the Cross Validate Model module
- How to read the output from the Cross Validate Model module

1) Create a new Experiment and name it as **Cross Validate**.

2) Add and connect the following modules:



Ref:

<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/cross-validate-model>
<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/linear-regression>

3) Set the properties for the **Select Columns in Dataset** module as follows:

▲ Select Columns in Dataset

Select columns

Selected columns:
All columns
Exclude column names: normalized-losses

Launch column selector

4) Set the properties for the **Clean Missing Data** module as follows:

▲ Clean Missing Data

Columns to be cleaned

Selected columns:
All columns

Launch column selector

Minimum missing value ratio

0

Maximum missing value ratio

1

Cleaning mode

Remove entire row

- 5) Set the properties for the **Cross Validate Model** module as follows:

▲ Cross Validate Model

Label column

Selected columns:
Column names: price

Launch column selector

Random seed

0

- 6) Click **Run**.

- 7) Click on the right output port of **Cross Validate Model** module and select **Visualize**. You should see the following:

Experiment created on 9/5/2020 > Cross Validate Model > Evaluation results by fold

rows	columns	Fold Number	Number of examples in fold	Model	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
12	8	0	19	Microsoft.Analytics.MachineLearning.Local.BatchLinearRegressor	2033.421462	2569.206167	0.628424	0.409588	0.590412
		1	19	Microsoft.Analytics.MachineLearning.Local.BatchLinearRegressor	1846.098776	3694.663157	0.389351	0.408822	0.591178
		2	19	Microsoft.Analytics.MachineLearning.Local.BatchLinearRegressor	1175.95891	1392.475547	0.223115	0.035692	0.964308
		3	19	Microsoft.Analytics.MachineLearning.Local.BatchLinearRegressor	2129.979458	2500.595686	0.313665	0.086848	0.913152
		4	19	Microsoft.Analytics.MachineLearning.Local.BatchLinearRegressor	1260.989438	1695.977419	0.267394	0.074843	0.925157
		5	19	Microsoft.Analytics.MachineLearning.Local.BatchLinearRegressor	1872.095966	2882.181942	0.30281	0.105757	0.894243
		6	20	Microsoft.Analytics.MachineLearning.Local.BatchLinearRegressor	1327.383343	1860.316954	0.221651	0.046855	0.953145
		7	20	Microsoft.Analytics.MachineLearning.Local.BatchLinearRegressor	2014.95775	2393.969016	0.290441	0.065669	0.934331
		8	19	Microsoft.Analytics.MachineLearning.Local.BatchLinearRegressor	1500.378613	1848.41663	0.208995	0.040137	0.959863

Note: The Cross Validate Model module takes in the input dataset and divide it into 10 parts (folds) by default. It will perform the training and validation 10 times, each time it will take a fold for testing and the remaining 9 folds for training. The results above shows the statistics for each fold. You can use the statistics to examine the sensitivity of any particular folds of your dataset when using a particular algorithm to train your model.

- 8) Click on the left output of the **Cross Validate** module and select **Visualize**. You will see the following:

Cross validation > Cross Validate Model > Scored results															x		
rows	columns	Fold Assignments	symboling	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	engine-type	r
193	27																
view as																	
6	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	f		
8	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	f		
7	1	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohcv	s		
6	2	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc	f		
1	2	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc	f		
0	2	audi	gas	std	two	sedan	fwd	front	99.8	177.3	66.3	53.1	2507	ohc	f		
5	1	audi	gas	std	four	sedan	fwd	front	105.8	192.7	71.4	55.7	2844	ohc	f		
7	1	audi	gas	std	four	wagon	fwd	front	105.8	192.7	71.4	55.7	2954	ohc	f		
5	1	audi	gas	turbo	four	sedan	fwd	front	105.8	192.7	71.4	55.9	3086	ohc	f		
5	2	bmw	gas	std	two	sedan	rwd	front	101.2	176.8	64.8	54.3	2395	ohc	f		
7	0	bmw	gas	std	four	sedan	rwd	front	101.2	176.8	64.8	54.3	2395	ohc	f		
3	0	bmw	gas	std	two	sedan	rwd	front	101.2	176.8	64.8	54.3	2710	ohc	s		
3	0	bmw	gas	std	four	sedan	rwd	front	101.2	176.8	64.8	54.3	2765	ohc	s		
1	1	bmw	gas	std	four	sedan	rwd	front	103.5	189	66.9	55.7	3055	ohc	s		
3	0	bmw	gas	std	four	sedan	rwd	front	103.5	189	66.9	55.7	3230	ohc	s		
9	0	bmw	gas	std	two	sedan	rwd	front	103.5	193.8	67.9	53.7	3380	ohc	s		
8	0	bmw	gas	std	four	sedan	rwd	front	110	197	70.9	56.3	3505	ohc	s		
6	2	chevrolet	gas	std	two	hatchback	fwd	front	88.4	141.1	60.3	53.2	1488	i	t		

The output above shows the results of taking each of the rows in the dataset and running against the model. **Cross validation does not simply measure the accuracy of a model, but also gives you some idea of how representative the dataset is and how sensitive the model might be to variations in the data.**

Note: The **Cross-Validate Model** module is useful for model optimization, but does not return a trained model. Instead, it provides metrics that you can use to determine the best model.

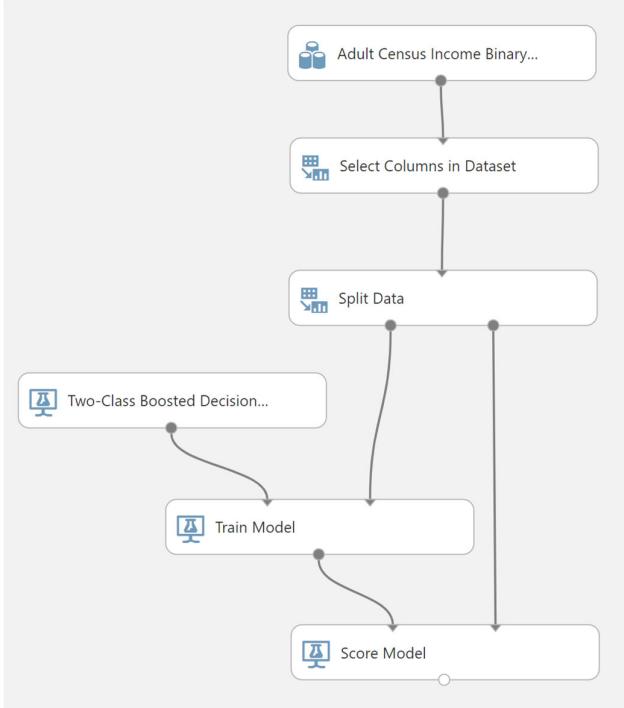
Activity 7 – Hyperparameters tuning

In this activity, we will learn:

- How to tune the hyperparameters used by a learning algorithm

1) Create a new experiment and name it **Hyperparameters Tuning**.

2) Add and connect the following modules as shown:



3) Set the properties for the **Select Columns in Dataset** module as follows:

▲ Select Columns in Dataset

Select columns

Selected columns:
All columns
Exclude column names: fnlwgt,education-num,marital-status,relationship,capital-gain,capital-loss

Launch column selector

4) Set the properties for the **Split Data** module as follows:

▲ Split Data

Splitting mode: Split Rows

Fraction of rows in the first output dataset: 0.8

Randomized split

Random seed: 0

Stratified split: False

5) Set the properties for the **Train Model** module as follows:

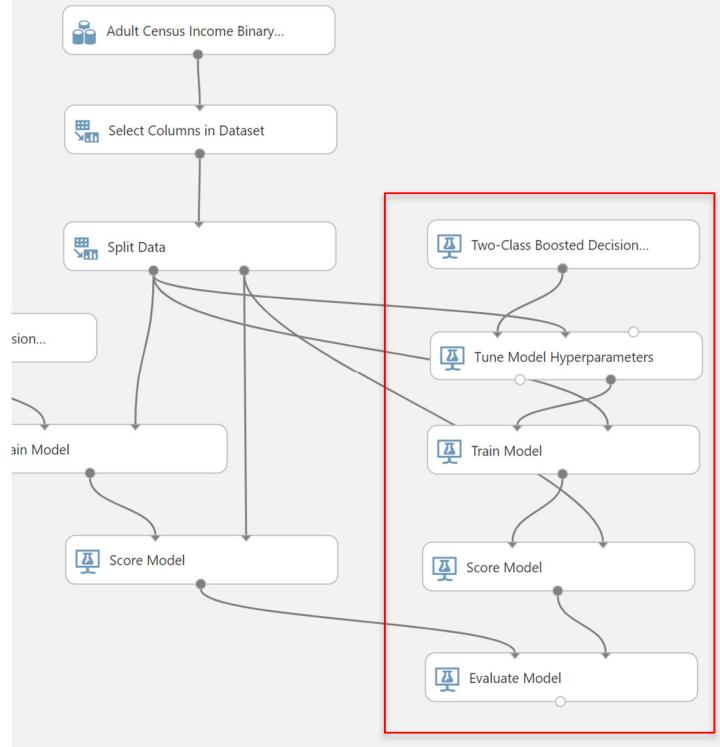
▲ Train Model

Label column

Selected columns:
Column names: income

Launch column selector

- 6) Add the following highlighted modules and connect them as shown below:



Ref:

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

- 7) Set the properties for the **Two-Class Boosted Decision Tree** module as follows:

▲ Two-Class Boosted Decision Tree

Create trainer mode

Parameter Range

Maximum number of leaves per tree

Use Range Builder
2, 8, 32, 128

Minimum number of samples per leaf node

Use Range Builder
1, 10, 50

Learning rate

Use Range Builder
0.025, 0.05, 0.1, 0.2, 0.4

Number of trees constructed

Use Range Builder
20, 100, 500

Random number seed

Allow unknown categorical levels

Note:

The Create Trainer Mode has the following options:

Single Parameter (default) – You fix all the values for the various hyperparameters of the learning model
Parameter Range – You provide a range of values for the various hyperparameters of the learning model

- 8) Set the properties for the **Tune Model Hyperparameters** module as follows:

▲ Tune Model Hyperparameters

Specify parameter sweeping mode

Maximum number of runs on random sweep

Random seed

Label column
Selected columns:

Metric for measuring performance for classification

Metric for measuring performance for regression

Note:

The Specify parameter sweeping mode has the following options:

Entire Grid – Calculates all possibilities. Perfect for testing a limited amount of parameter sets. All parameter combinations are covered (can take a lot of time)

Random sweep (default) – Tries x random guesses out of the possible parameter values you provided along with the model.

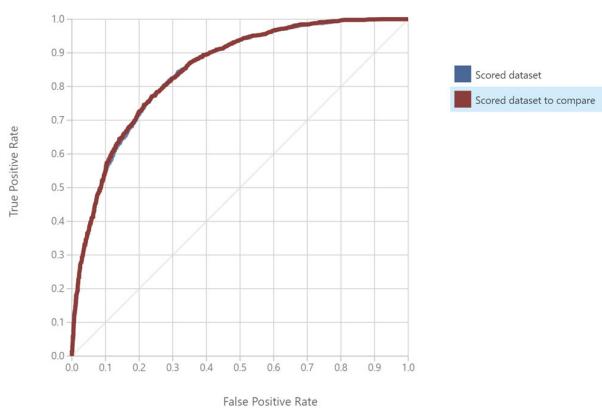
Random grid – Creates a grid of all possibilities, then samples a limited amount of random tries out of the grid. Great to get insight in how combinations of parameters perform.

9) Click **Run**.

10) Click on the output port of the **Evaluate Model** module and select **Visualize**:

Hyperparameters tuning > Evaluate Model > Evaluation results

ROC PRECISION/RECALL LIFT



True Positive	False Negative	Accuracy	Precision	Threshold	AUC		
776	731	0.812	0.659	0.5	0.848		
False Positive	True Negative	Recall	F1 Score				
402	4111	0.515	0.578				
Positive Label	Negative Label						
>50K	<=50K						

Compare the results of the two learning models.

Activity 8 – 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.

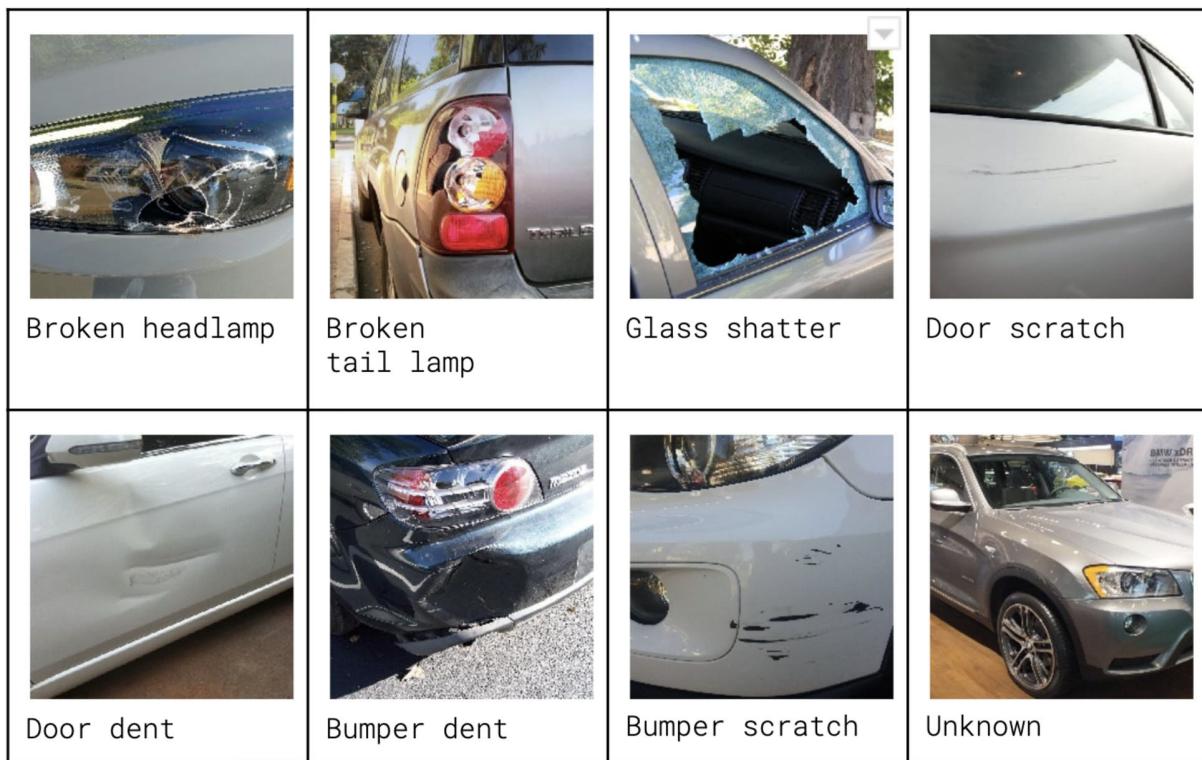
Classes

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.

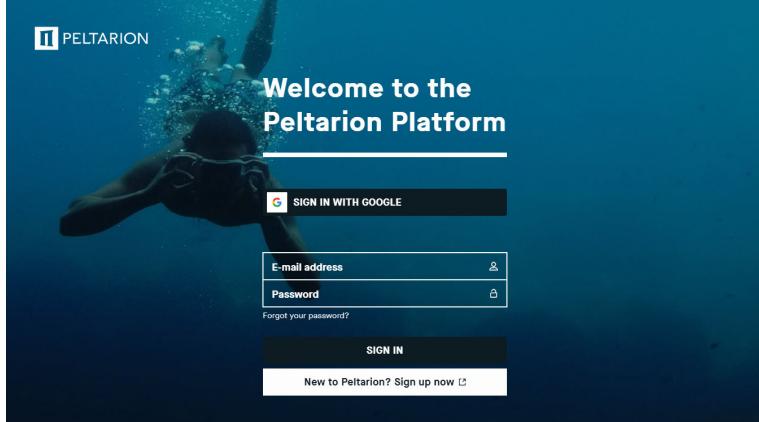
Each collected image represents one car with one specific type of damage. This means that the dataset can be used to solve a single-label classification problem.



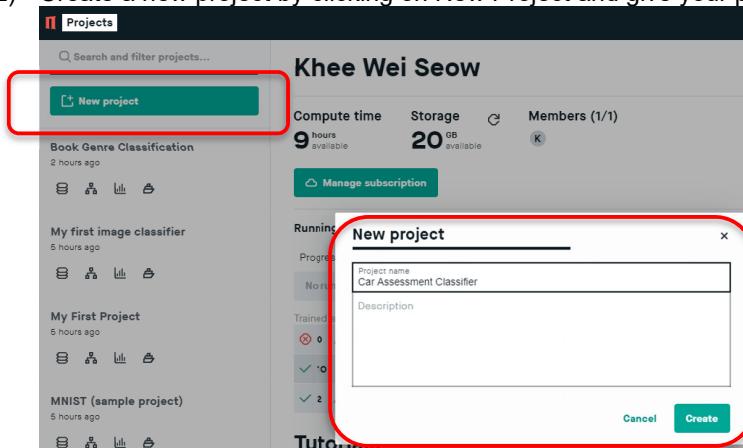
3) Create new Project

4)

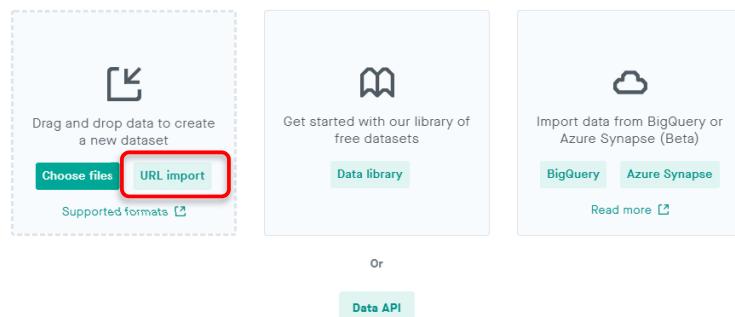
- 1) Sign up for an account and login at <https://platform.peltarion.com/login>



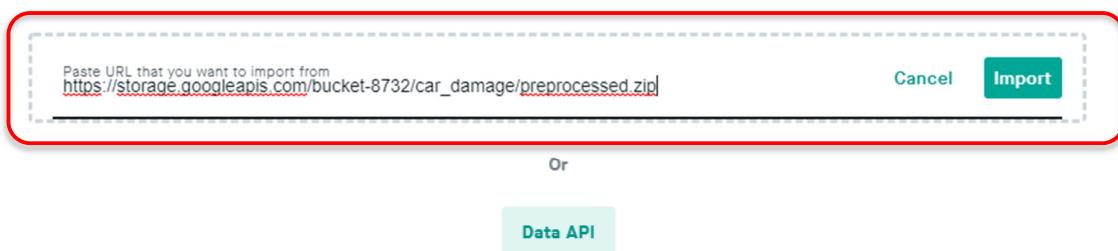
- 2) Create a new project by clicking on New Project and give your project a name. Click **Create** when done.



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

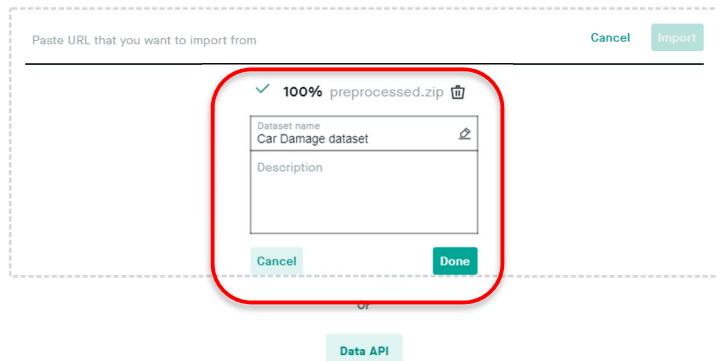


- 4) Copy the link and paste the link https://storage.googleapis.com/bucket-8732/car_damage/preprocessed.zip to the **Import data from URL** box. The zip includes the whole dataset.



- 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.

5) 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) Delete the default subsets **Training** and **Validation** by clicking the **Subsets options menu (...)** then **Delete**.

- 2) 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.

Create a subset

A subset enables you to define a set of examples in the dataset to be used for training and validation of your experiments.

Read more about how to create subsets [here](#).

Training

Filter(s)

1. Feature subset ▾ Operator Is equal to ▾ Value T

+ Add random % filter + Add conditional filter

Create

- 3) 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.

Create a subset

A subset enables you to define a set of examples in the dataset to be used for training and validation of your experiments.

Read more about how to create subsets [here](#).

Validation

Filter(s)

1. Feature subset ▾ Operator Is equal to ▾ Value V

+ Add random % filter + Add conditional filter

Create

- 4) Select the Training subset that you have created in **Normalize on subset**.

< Datasets overview

Car Damage dataset

Versions

#1 Draft

Subsets

+ New subset

Training

Select a feature to visualize

subset is equal to T

Validation

Select a feature to visualize

subset is equal to V

Normalize on subset

Training

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

The screenshot shows the 'Car Damage dataset' page. On the left, there's a sidebar with sections for Versions, Subsets (with a 'New subset' button), Training, and Validation. The main area displays three tables: 'image', 'class', and 'subset'. The 'image' table shows a histogram of distribution from 0 to 1. The 'class' table lists categories like 'door_dent' and 'bumper_scratch' with their respective percentages. The 'subset' table shows a binary distribution between T and V. A green 'Save version' button is located at the top right, with a red box drawn around it.

6) 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 **Define dataset** tab. Click **Next** to continue.

The screenshot shows the 'Experiment wizard' with the 'Define dataset' tab selected. It displays the 'Dataset' dropdown set to 'Car Damage dataset' and the 'Training subset' dropdown set to 'Training'. Below these are tabs for 'Choose snippet' and 'Initialize weights'. At the bottom are 'Create blank experiment' and 'Next' buttons. A red box highlights the 'Dataset' dropdown and its value.

- 4) Click on the **Choose snippet** tab and select the **EfficientNet B0** snippet. Click **Next** to continue.

Experiment wizard

Use the experiment wizard to create a ready-to-run experiment.
Or create a blank experiment and build your custom experiment.

Experiment name
Experiment 1

1. Define dataset 2. Choose snippet 3. Initialize weights

Input feature
image

Target feature
class

Problem type
Single-label image classification

Recommended snippets

EfficientNet B0

MobileNetV2 1.0

ResNetV2 large 101

Other suitable snippets

Inception v4

EfficientNet B5

VGG19

ResNetV2 medium 101

Previous

Create blank experiment

Next

- 5) Click on the **Initialize weights** tab. Select **ImageNet** for pretrained data. Click **Create** to continue.

Experiment wizard

Use the experiment wizard to create a ready-to-run experiment.
Or create a blank experiment and build your custom experiment.

Experiment name
Experiment 1

1. Define dataset 2. Choose snippet 3. Initialize weights

Choose how you want to initialize the weights. If using pretrained weights, choose the pretraining data that resembles your dataset the most.

Weight initialization
Pretrained from ImageNet

Weights trainable (all blocks) [Third-party terms apply](#)

Trained on the ImageNet classification dataset, with 1.2 million images from 1000 classes.

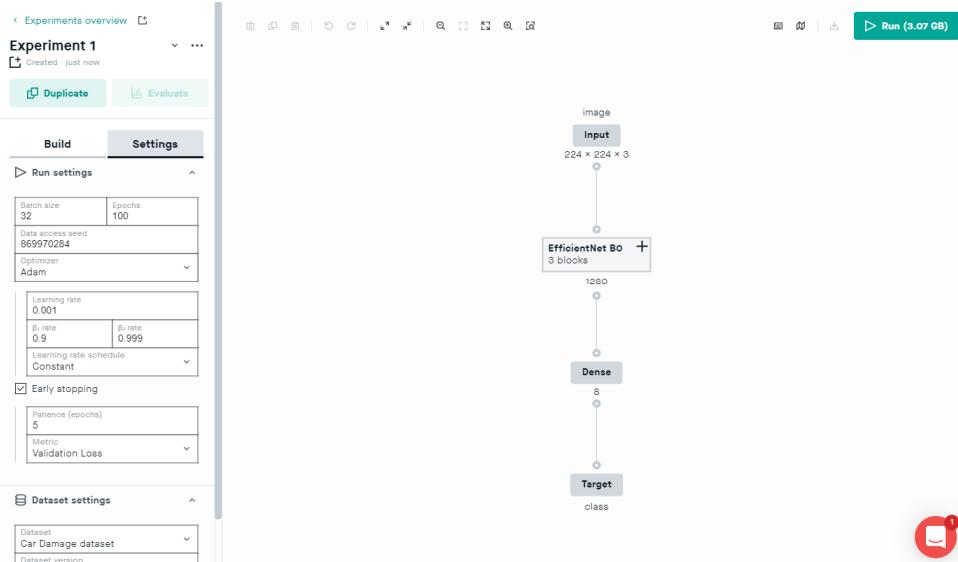
Browse the dataset at image-net.org

Previous

Create blank 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 -.



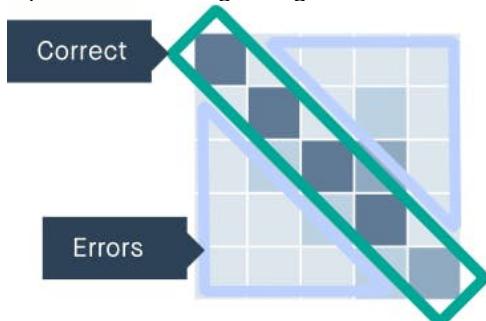
7) Run the experiment

- Click the **Settings** tab and change the **Learning rate** to 0.0005. Click **Run** to start the training.

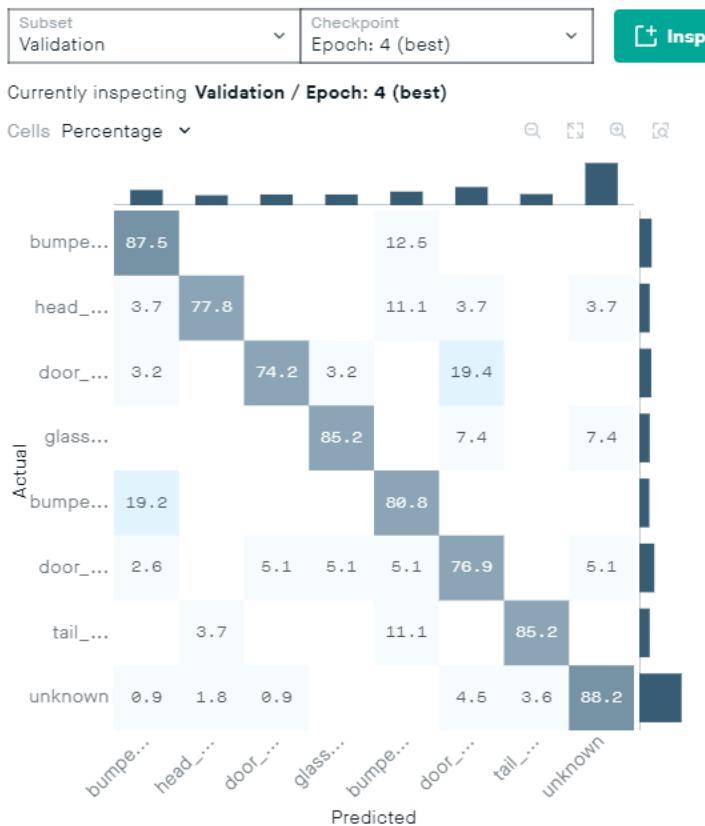
The screenshot shows the 'Settings' tab of 'Experiment 1'. The 'Run settings' section includes 'Batch size' (32), 'Epochs' (100), 'Optimizer' (Adam), and a 'Learning rate' field containing '0.0005'. This field is highlighted with a red rectangle. Other settings include 'Data access seed' (869970284), 'Early stopping' (Patience: 5, Metric: Validation Loss), and 'Dataset settings' (Dataset: Car Damage dataset).

8) Analyse the experiment

- Go to the **Evaluation** view. 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.



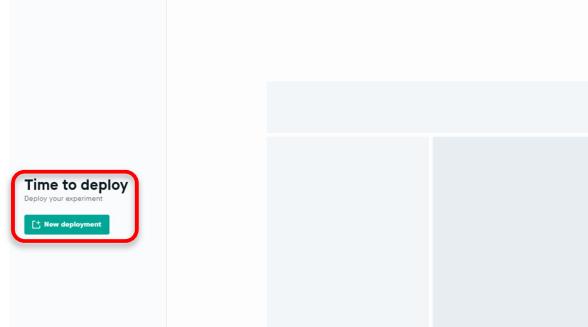
- 2) Note that metrics are based on the validation subset which only consists of 20% of the original dataset.
- 3) Click the dropdown next to **Cells** and select **Percentage**. The normalized values that are now displayed correspond to the recall for each class.



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

9) 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.

Experiment	Experiment 1
Checkpoint	Epoch: 4 (best)
Name	My deployment

Create

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

The screenshot shows the 'Deployment Info' section of the Peltarion Platform. It includes details like Experiment ID (Experiment 1), Checkpoint (Epoch: 4 (best)), Deployment ID (f51e9942-ffc4-40aa-9d5f-7fafba7b9686), and creation date (2020-09-17). Below this, the 'Parameters' section shows input and output configurations. A red box highlights the 'Test deployment' button, which is located in a light blue header bar above the API tester interface.

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

10) 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.

The screenshot shows the 'Image & Text Classifier API tester' interface. It has sections for Input datatype (Image selected), Output datatype (Categorical selected), and Setup (URL and Token fields filled with deployment details). Under 'Input parameter image', there are 'Width' (224) and 'Height' (224) fields, and 'Image processing' options (Grayscale, Tilt, Invert). On the left, there is a box for dragging images, and on the right, a 'Result' panel with a placeholder message 'Press ⏪ to get result...'. A red box highlights the 'Result' panel.

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



11) Next Steps

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

Activity 9 – 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 navigation bar with a red icon and the text 'Projects'. Below this is a search bar with the placeholder 'Search and filter projects...'. A large green button labeled 'New project' with a plus sign is prominently displayed. To the right, the project details for 'Khee Wei Seow' are shown, including compute time (9 hours available), storage (20 GB available), and members (1/1). Below the project details, a list item 'My first image classifier' is shown, created 3 hours ago.

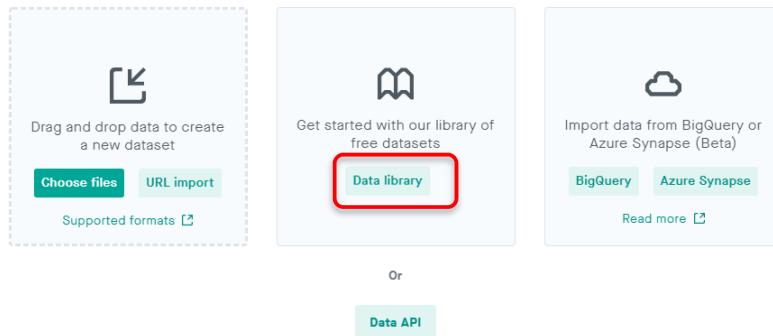
New project

Project name: Book Genre Classification

Description:

Cancel Create

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

The screenshot shows the Data library search results for 'book summaries'. The search bar at the top has 'book summaries' typed into it, with a red box around it. Below the search bar, the 'Book summaries' dataset is listed first, with its thumbnail, file size (10.7 MB), number of rows (12 835), and a brief description: '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) ...'. This dataset is also highlighted with a red box. Other datasets listed include 'Stanford Online Products', 'Fruits 360', 'Stack Overflow Tags', and 'Freesound Audio Tagging - tutorial data'.

5) Click on **Accept and import**



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

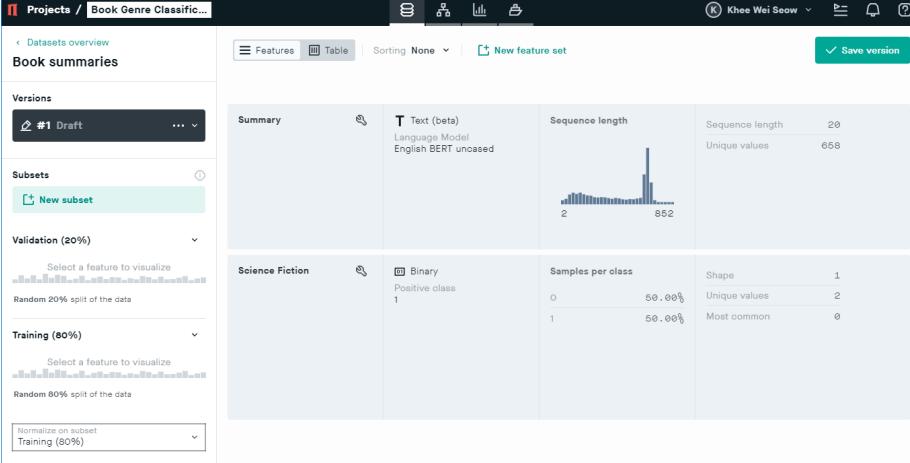
Build your sci-fi/not sci-fi classifier, based on the plot summary.

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.



Summary	Text (beta) Language Model English BERT uncased	Sequence length	Sequence length Unique values
			20 668

Science Fiction	Binary Positive class 1	Samples per class	Shape
		0 50.00% 1 50.00%	1 2 Most common

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 set the following in the **Feature Settings**.
- Encoding to Text (Beta)
 - Sequence length to 512
 - Language model to English BERT uncased

Shape 512

Summary

Encoding
Text (beta)
Language model
English BERT uncased

1 Old Major, the old boar on the Manor Farm calls the animals on the farm for a meeting, where he compares the humans to parasites and teaches the animals a revolutionary song, 'Beasts of England'. When Major dies, two young pigs, Snowball and Napoleon, assume command and turn his dream into a philosophy. The animals revolt and drive the drunken and irresponsible Mr. Jones from the farm, renaming it 'Animal Farm'. They adopt Seven Commandments of Animalism, the most important of which is, "All animals are equal". Snowball attempts to teach the animals reading and writing; food is plentiful, and the farm runs smoothly. The pigs elevate themselves to positions of leadership and set aside special food items, ostensibly for their personal health...

The animals revolt and drive the drunken and irresponsible Mr. Jones from the farm, renaming it 'Animal Farm'. They adopt Seven Commandments of Animalism, the most important of which is, "All animals are equal". Snowball attempts to teach the animals reading and writing; food is plentiful, and the farm runs smoothly. The pigs elevate themselves to positions of leadership and set aside special food items, ostensibly for their personal health...

Note:

Sequence length

Sequence length corresponds to the expected number of words in each summary sample. If the actual number of words in a sample are fewer than indicated by this parameter, the text will be padded. If it is longer, the text will be truncated, i.e., cut from the end to fit in the sequence.

The BERT block accepts any sequence length between 3 and 512. Smaller sequences compute faster, but they might cut some words from your text. You will want to set this value as low as possible to reduce training time and minimize memory use but still large enough to include enough information.

Language model

The Language model should match the language of the input data, in this case, English BERT uncased. Read <https://peltarion.com/knowledge-center/documentation/datasets-view/edit-an-imported-dataset-for-use-in-experiments/feature-encoding#Text> for more information about this parameter.

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.

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

Projects / Book Genre Classification

Go to draft

Use in new experiment

Datasets overview

Book summaries

Versions

#1

Subsets

New subset

Validation (20%)

Training (80%)

Shape 512

Shape 1

Summary

Encoding
Text (beta)
Language model
English BERT uncased

Science Fiction

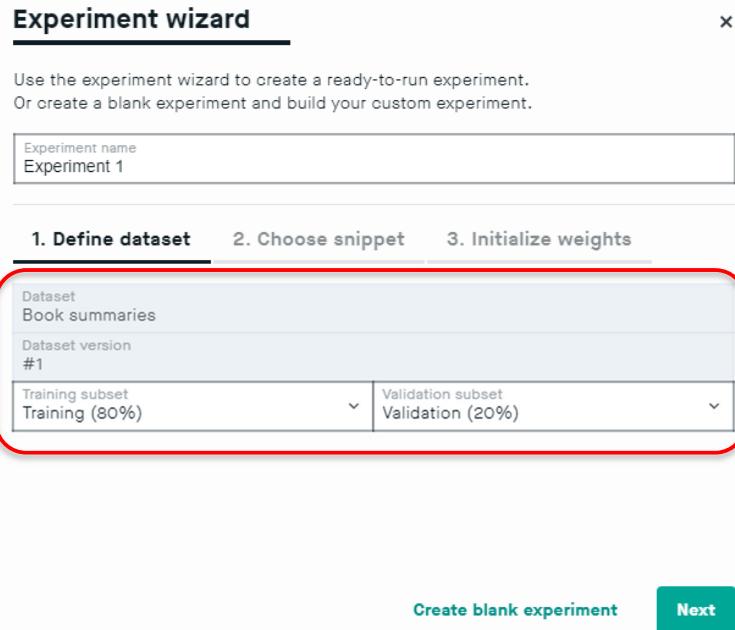
Encoding
Binary
Positive class
1

1 Old Major, the old boar on the Manor Farm calls the animals on the farm for a meeting, where he compares the humans to parasites and teaches the animals a revolutionary song, 'Beasts of England'. When Major dies, two young pigs, Snowball and Napoleon, assume command and turn his dream into a philosophy. The animals revolt and drive the drunken and irresponsible Mr. Jones from the farm, renaming it 'Animal Farm'. They adopt Seven Commandments of Animalism, the most important of which is, "All animals are equal". Snowball attempts to teach the animals reading and writing; food is plentiful, and the farm runs smoothly. The pigs elevate themselves to positions of leadership and set aside special food items, ostensibly for their personal health...

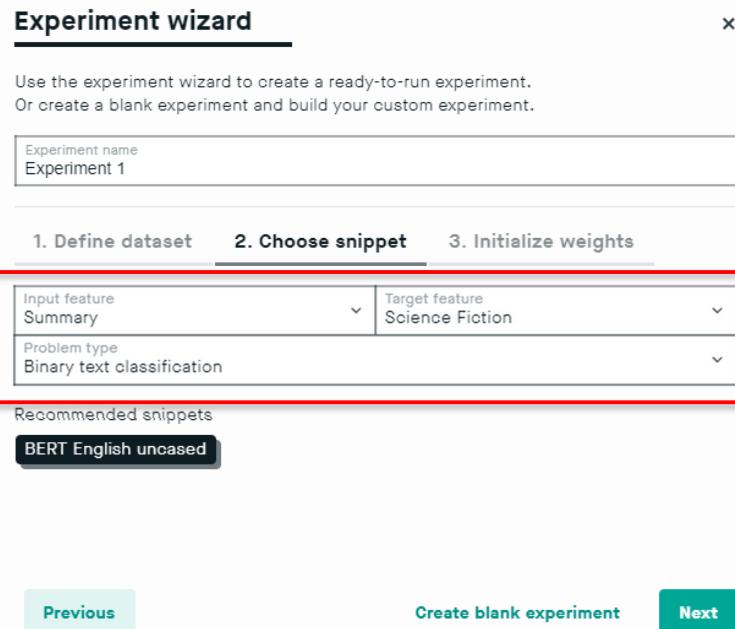
2 Alex, a teenager living in near-future England, leads his gang on nightly orgies of opportunistic, random "ultra-violence". Alex's friends ("droogs" in the novel's Anglo-Russian slang, Nadsat) are: Dim, a slow-witted bruiser who is the gang's muscle; Georgie, an ambitious second-in-command; and Pete, who mostly plays along as the droogs indulge their taste for ultra-violence. Characterized as a

5) Design a text classification model with the BERT model

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



- 2) Click **Next** to continue.
- 3) Set the input feature to **Summary** and the target feature to **Science Fiction** as shown below.



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 on the **Initialize weights** tab. Select the **Pretained from English Wikipedia and BookCorpus** and set all weights trainable by checking the Weights trainable (all blocks) box.

Experiment wizard

Use the experiment wizard to create a ready-to-run experiment.
 Or create a blank experiment and build your custom experiment.

Experiment name
 Experiment 1

1. Define dataset 2. Choose snippet 3. Initialize weights

Choose how you want to initialize the weights. If using pretrained weights, choose the pretraining data that resembles your dataset the most.

Weight initialization
 Pretrained from English Wikipedia and BookCorpus

Weights trainable (all blocks) Third-party terms apply

Trained on BookCorpus and English Wikipedia.

See Pre-training data section of the official implementation for further details.

Previous Create blank experiment Create

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

Projects / Book Genre Classification

Experiments overview

Experiment 1

Created: just now

Duplicate Evaluate

Build Settings

Run settings

Batch size	6	Epochs	2
Data access seed 1110274247			
Optimizer Adam			
Learning rate 0.00002			
β_1 rate	0.9	β_2 rate	0.999
Learning rate schedule Triangle schedule			
Warm-up epochs 0.5			
Decrement per epoch 0.000008			
Early stopping			

Dataset settings

Dataset	Book summaries
Dataset version	

Summary

Input
512

BERT Encoder
768

Dense
768

Dense
1

Target
Science Fiction

Run (417.70 MB)

- 7) Click the **Settings** tab and check that:
- **Batch Size** is 6. If you set a larger batch size you will run out of memory since we've set sequence length to 512.
 - **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 0.000001 (5 zeros). To avoid catastrophic forgetting.

Experiment 1

Created · just now

Duplicate **Evaluate**

Build **Settings**

Run settings

Batch size	6	Epochs	2
Data access seed 1110274247			
Optimizer Adam			
Learning rate 0.000001			
Beta1 rate	0.9	Beta2 rate	0.999
Learning rate schedule Triangle schedule			
Warm-up epochs 0.5			
Decrement per epoch 0.000008			
<input type="checkbox"/> Early stopping			

- 5) Click **Run** to start training the model.

Projects / Book Genre Classification

Experiment 1

Created · just now

Duplicate **Evaluate**

Build **Settings**

Run settings

Batch size	6	Epochs	2
Data access seed 1110274247			
Optimizer Adam			
Learning rate 0.000001			
Beta1 rate	0.9	Beta2 rate	0.999
Learning rate schedule Triangle schedule			
Warm-up epochs 0.5			
Decrement per epoch 0.000008			
<input type="checkbox"/> Early stopping			

Dataset settings

Dataset
Book summaries

Summary

```

graph TD
    Input[Input  
512] --> BERT[BERT Encoder  
768]
    BERT --> Dense1[Dense  
768]
    Dense1 --> Dense2[Dense  
1]
    Dense2 --> Target[Target  
Science Fiction]
  
```

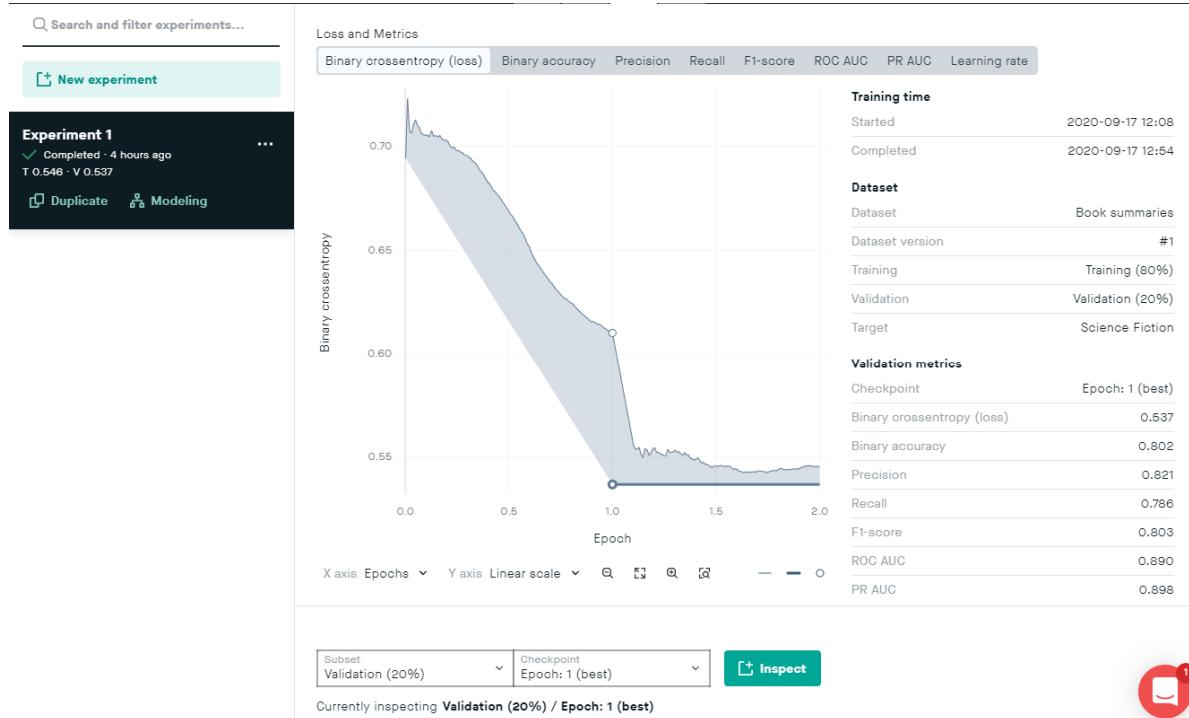
Info, warnings and errors

⚠ Run settings: The selected decrement per epoch of 0.000008 is greater than the initial learning rate. This lets the learning rate go to zero within one epoch or less. Try using a smaller decrement value.

Run (412.70 MB)

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.

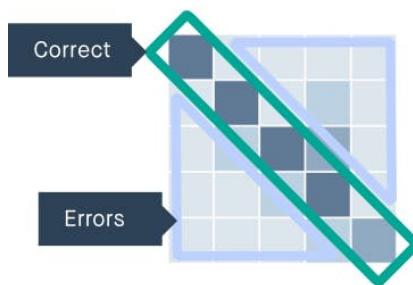
You can read more about the loss metrics here: <https://peltarion.com/knowledge-center/documentation/evaluation-view/classification-loss-metrics>

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

- In the Deployment view click New deployment.

Deployment Info

- Experiment: Experiment 1
- Checkpoint: Epoch: 1 (best)
- Date created: 2020-09-17
- Deployment ID: 80be3998-6499-4734-af32-bed66e73cd4f

Parameters

Input	Feature	Type	Name
Summary	text (512)	Summary	

Output	Feature	Type	Name
Science Fiction	binary (1)	Science Fiction	

Test deployment

Format: CURL

```
curl -X POST https://a.azure-eu-west.platform.peltarion.com/deployment/80be3998-6499-4734-af32-bed66e73cd4f/forward
```

URL: https://a.azure-eu-west.platform.peltarion.com/deployment/80be3998-6499-4734-af32-bed66e73cd4f/forward

Token: (redacted)

Input example:

```
curl -X POST https://a.azure-eu-west.platform.peltarion.com/deployment/80be3998-6499-4734-af32-bed66e73cd4f/forward
-F "Summary=VALUE"
```

Output example: (redacted)

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

8) 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.

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 forgettable.



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 forgettable.



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 10 – 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 interface. At the top, there is a dark header bar with the word "Projects". Below it is a search bar labeled "Search and filter projects...". A teal button labeled "+ New project" is prominently displayed. On the left, a sidebar shows a project titled "My first image classifier" created 3 hours ago. The main area is titled "Khee Wei Seow" and displays resource details: Compute time (9 hours available), Storage (20 GB available), and Members (1/1). Below this, a modal window titled "New project" is open. It contains a form with a "Project name" field containing "Movie Review Sentiments", a "Description" field (empty), and two buttons at the bottom: "Cancel" and "Create".

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. At the top, there is a navigation bar with "Projects / Movie Review Sentiments" and a user profile icon. Below the navigation bar are three main import options: 1) A dashed box for "Drag and drop data to create a new dataset" with "Choose files" and "URL import" buttons. 2) A box for "Get started with our library of free datasets" with a "Data library" button. 3) A box for "Import data from BigQuery or Azure Synapse (Beta)" with "BigQuery" and "Azure Synapse" buttons. A "Read more" link is also present. Below these options is 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...

The Cali House - tutorial data
The Cali House dataset combines data from two sources. The tabular data from the California 1990 Census, which give...
210 MB 20 640 rows

Most downloaded books - tutor...
Over 100000 text passages from public domain books, with extra...
27 MB 102 775 rows

IMDB - tutorial data
This dataset contains textual movie reviews from IMDB users, together...
13 MB 25 000 rows

Car Damage - tutorial data 16 MB 1 594 rows
Photos of damaged cars, with the type of damage (glass shatter, bumper dent, door dent, etc.)

Bank marketing - tutorial data 5.4 MB 45 rows

- 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

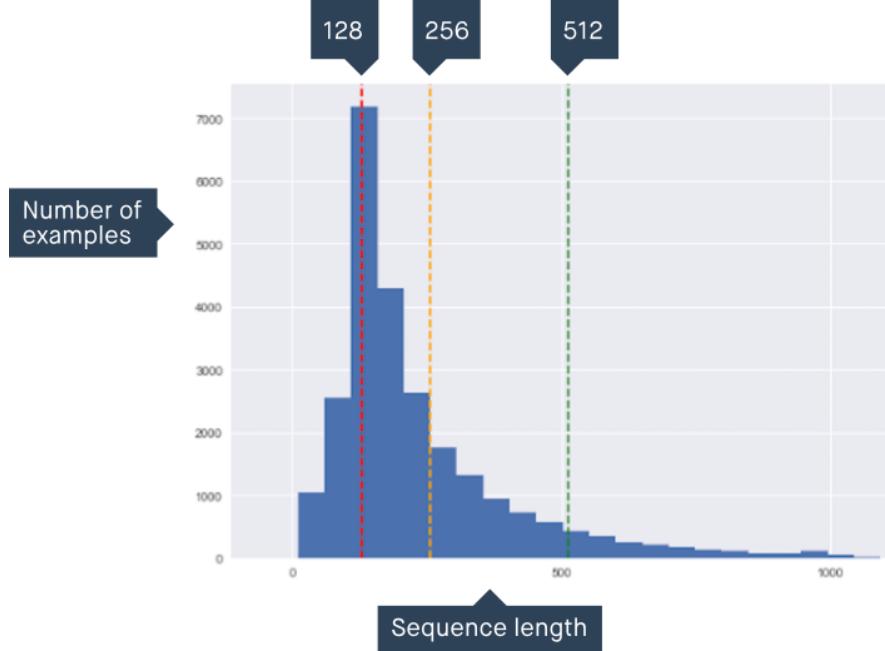
- 1) Click on the **Table** button, click the **Review** column and set the following in the **Feature Settings**.
- Encoding to Text (Beta)
 - Sequence length to 512
 - Language model to English BERT uncased

Note:

Sequence length

Sequence length corresponds to the expected number of words in each summary sample. If the actual number of words in a sample are fewer than indicated by this parameter, the text will be padded. If it is longer, the text will be truncated, i.e., cut from the end to fit in the sequence.

The BERT block accepts any sequence length between 3 and 512. Smaller sequences compute faster, but they might cut some words from your text. You will want to set this value as low as possible to reduce training time and minimize memory use but still large enough to include enough information. In our dataset a Sequence length of 512 will cut 8% of the review samples, a length of 256 will cut 30%, and 128 will cut 74%.



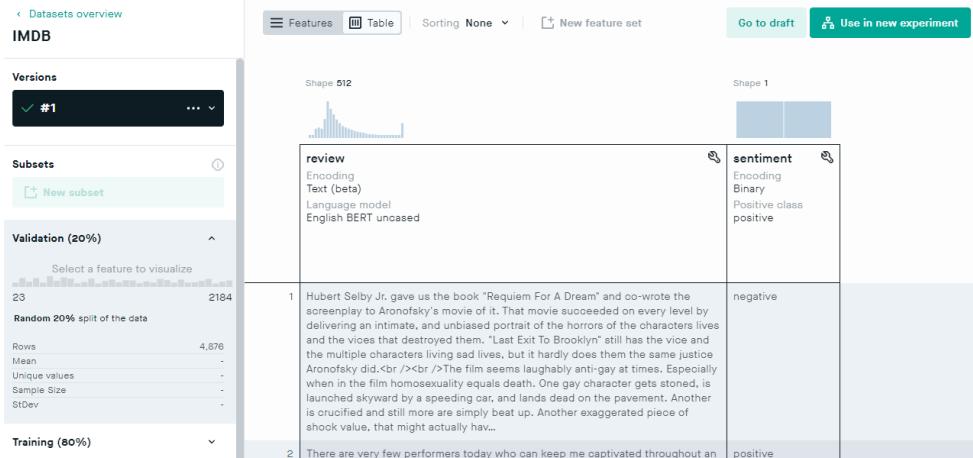
Language model

The Language model should match the language of the input data, in this case, English BERT uncased. Read <https://peltarion.com/knowledge-center/documentation/datasets-view/edit-an-imported-dataset-for-use-in-experiments/feature-encoding#Text> for more information about this parameter.

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**.



6) Design a text classification model with the BERT model

- 1) Make sure that the **IMDB** dataset is selected in the **Experiment wizard**. Click **Next**.

The 'Experiment wizard' step 1: Define dataset. It shows the 'Dataset' dropdown set to 'IMDB' and 'Dataset version' set to '#1'. Below it, 'Training subset' is 'Training (80%)' and 'Validation subset' is 'Validation (20%)'. A red box highlights the 'Define dataset' section.

- 2) Set the input feature to **review** and the target feature to **sentiment** as shown below.

The 'Experiment wizard' step 2: Choose snippet. It shows the 'Input feature' dropdown set to 'review' and 'Target feature' dropdown set to 'sentiment'. Below it, 'Problem type' is 'Binary text classification'. A red box highlights the 'Choose snippet' section. At the bottom, a button for 'BERT English uncased' is shown.

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.
- 3) Click on the **Initialize weights** tab. Select the pretrained weights **English Wikipedia and BookCorpus** and set all weights trainable by checking the **Weights trainable (all blocks)** box.

Experiment wizard

Use the experiment wizard to create a ready-to-run experiment.
Or create a blank experiment and build your custom experiment.

Experiment name
Experiment 1

1. Define dataset 2. Choose snippet **3. Initialize weights**

Choose how you want to initialize the weights. If using pretrained weights, choose the pretraining data that resembles your dataset the most.

Weight initialization
Pretrained from English Wikipedia and BookCorpus

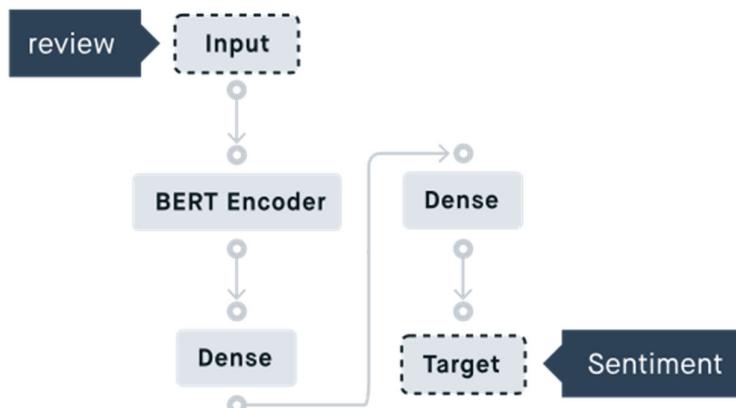
Weights trainable (all blocks) Third-party terms apply

Trained on BookCorpus and English Wikipedia.

See Pre-training data section of the official implementation for further details.

Previous Create blank experiment **Create**

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



- 5) Click the Settings tab and check that:
- a. **Batch Size** is 6. If you set a larger batch size you will run out of memory since we've set sequence length to 512.
 - 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 0.000001 (5 zeros). To avoid catastrophic forgetting.

< Experiments overview ⌂

Experiment 1

Created · 1 min ago

Duplicate Evaluate

Build Settings

Run settings

Batch size	Epochs
6	2

Data access seed: -202080887

Optimizer: Adam

Learning rate	0.000001
---------------	----------

beta1 rate: 0.9 beta2 rate: 0.999

Learning rate schedule: Triangle schedule

Warm-up epochs	0.5
Decrement per epoch	0.000008

Early stopping

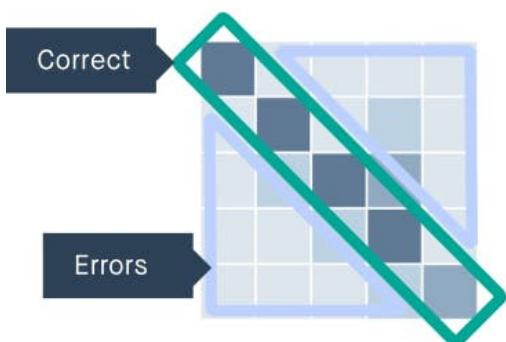
- 6) Click **Run** to start training the model. **Note: This experiment will take hours to complete.**
- 7) **Evaluation**
- 1) Navigate to the **Evaluation** view and watch the model train. The training will take quite a long time since BERT is a very large and complex model. Currently expect more than four hours for 2 epochs when you use Sequence length 512.

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.

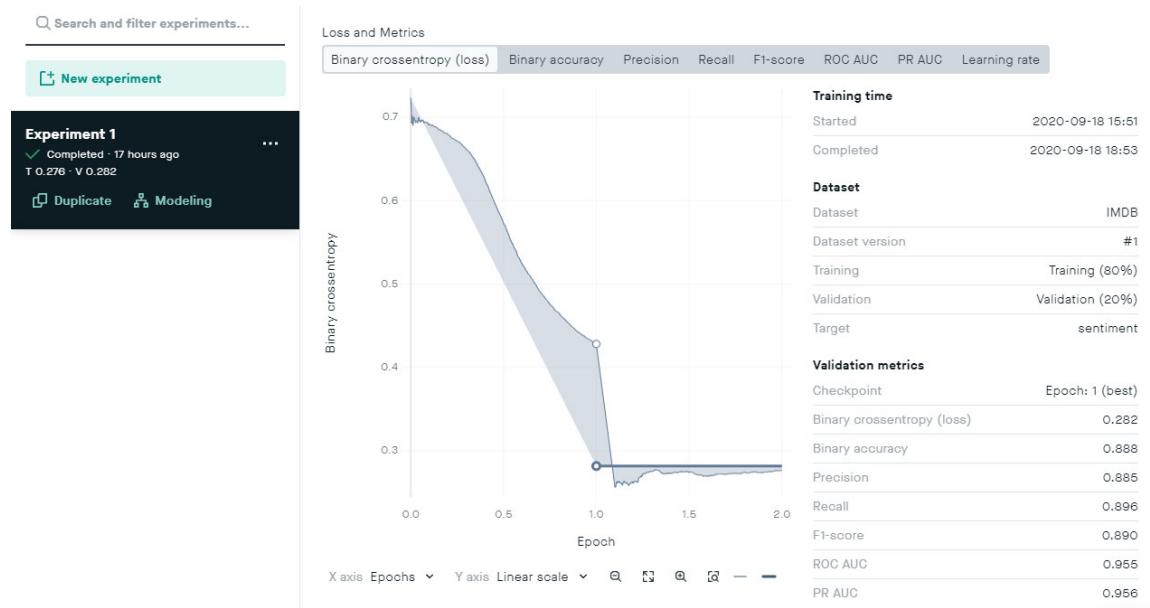
Confusion matrix

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.



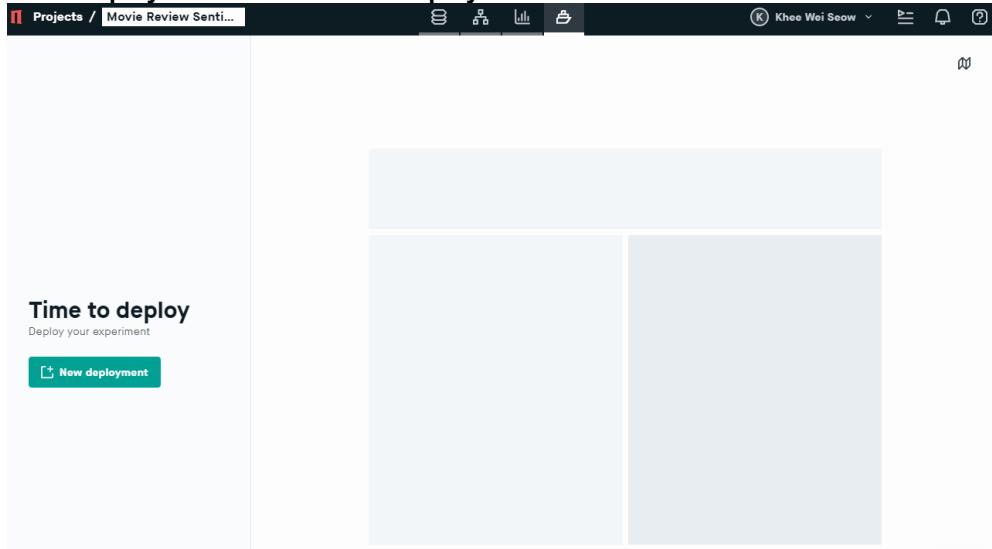
Recall

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.



8) Create new deployment

- In the Deployment view click New deployment.



- 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.

The screenshot shows the 'Create deployment' dialog. It has three input fields: 'Experiment' (set to 'Experiment 1'), 'Checkpoint' (set to 'Epoch: 1 (best)'), and 'Name' (set to 'Experiment 1'). At the bottom is a green 'Create' button.

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

The screenshot shows the Pelatirion platform's deployment interface. On the left, there's a sidebar with 'Projects / Movie Review Senti...' and a 'New deployment' button. The main area is titled 'Experiment 1' with a status of 'Created - just now'. It has buttons for 'Enable', 'Rename', and 'Evaluation'. The 'Parameters' section has two tabs: 'Input' and 'Output'. Under 'Input', there's a table with one row: Feature 'review' (Type 'text (512)'), Name 'review'. Under 'Output', there's a table with one row: Feature 'sentiment' (Type 'binary (1)'), Name 'sentiment'. To the right, there's a 'Test deployment' panel with a 'Format' dropdown set to 'CURL'. It contains a 'curl' command and examples for both input and output.

9) Test the text classifier in a browser

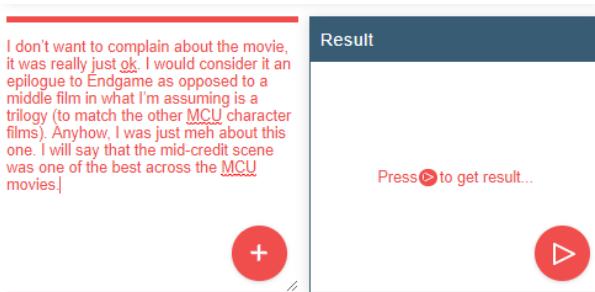
- Let's test your model. Click the **Test deployment** button, and you'll open the **Text classifier API tester** with all relevant data copied from your deployment.

The screenshot shows the 'Text classifier API tester' tool. It has several configuration sections: 'Input datatype' (Image or Text, Text is selected), 'Output datatype' (Categorical or Binary, Binary is selected), 'Setup' (URL: https://a.azure-eu-west.platform.peltarion.com/deployment/0f32946d-75b6-434c-b7dd-, Token: 67a78989-1f3b-4417-b46d-34df74e9f33a, Threshold: 0.5, Input parameter: review). Below these, there's a text input field 'Insert text here...' and a 'Result' panel with the message 'Press ⏪ to get result...'. There are also red circular buttons with '+' and '▶' symbols.

- 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.



3) Click **Play**.

4) [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 "67a78989-1f3b-4417-b46d-34df74e9f33a" ^
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 "67a78989-1f3b-4417-b46d-34df74e9f33a" ^
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

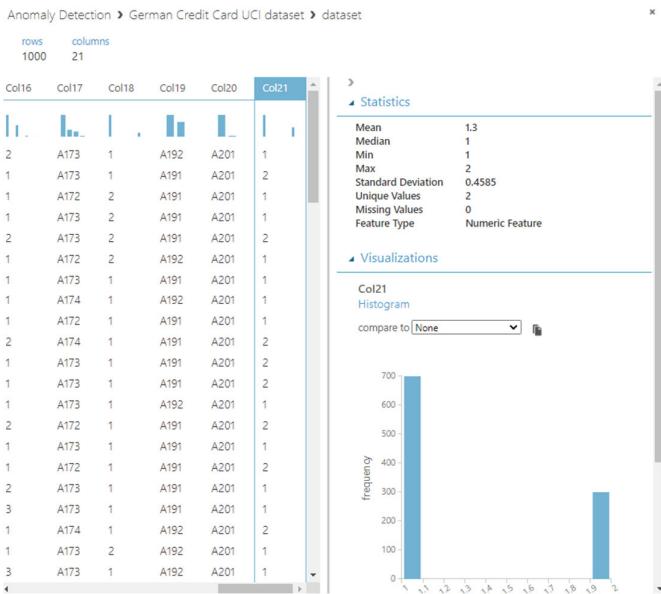
The next steps could be to try to run the project for different sequence length and see if that improves the result or maybe change the learning rate

Activity 11 – [Bonus] Anomaly Detection

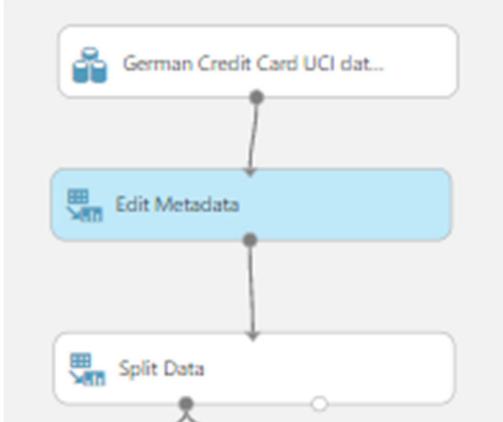
In this activity, we will learn:



- 1) Create a new Experiment and name it as **Anomaly Detection**.
- 2) Add the **German Credit Card UCI dataset** into the canvas. This dataset classifies people described by a set of attributes as good or bad credit risks. Comes in two formats (one all numeric).
Ref: [https://archive.ics.uci.edu/ml/datasets/statlog+\(german+credit+data\)](https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data))



- 3) Add and connect the following modules:



- 4) Set the properties for the **Edit Metadata** module as follows:

▲ Edit Metadata

Column

Selected columns:
Column names: Col21

Launch column selector

Data type

Unchanged

Categorical

Unchanged

Fields

Label

New column names

Label

This will rename and mark the target column Col21 as "Label".

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

- 5) Set the properties for the **Split Data** module as follows:

▲ Split Data

Splitting mode

Split Rows

Fraction of rows in the first set

0.75

Randomized split

Random seed

0

Stratified split

True

Stratification key column

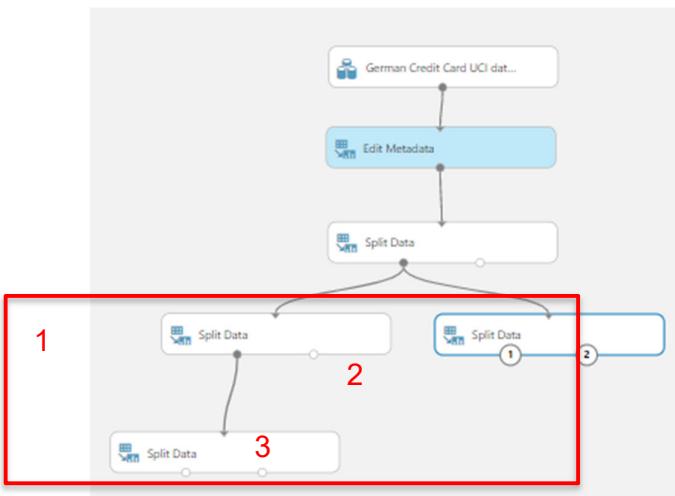
Selected columns:
Column names: Label

Launch column selector

This will split the dataset into 75% for training and 25% for testing. Remember to set the **Stratified split** to True and select the **Label** column.

Note: Some datasets do not have a balanced number of examples for each class label like this case. As such, it is desirable to split the dataset into train and test sets in a way that preserves the same proportions of examples in each class as observed in the original dataset. This is called stratified split.

- 6) Add three **Split Data** into the canvas and connect them as shown



- 7) Set the properties for the **Split Data** module marked (1) as follows:

▲ Split Data

Splitting mode: Split Rows

Fraction of rows in the first split: 0.75

Randomized split

Random seed: 0

Stratified split: True

Stratification key column: Selected columns: Column names: Label

Launch column selector

This is further split our training dataset to 75%/25%. We will use this train-test set for our hyperparameter tuning in the workflow which we will set up shortly.

- 8) Set the properties for the **Split Data** module marked (2) as follows:

▲ Split Data

Splitting mode: Regular Expression

Regular expression: \\"Label" ^1

In this split, we are separating the “good” and the “bad” records.

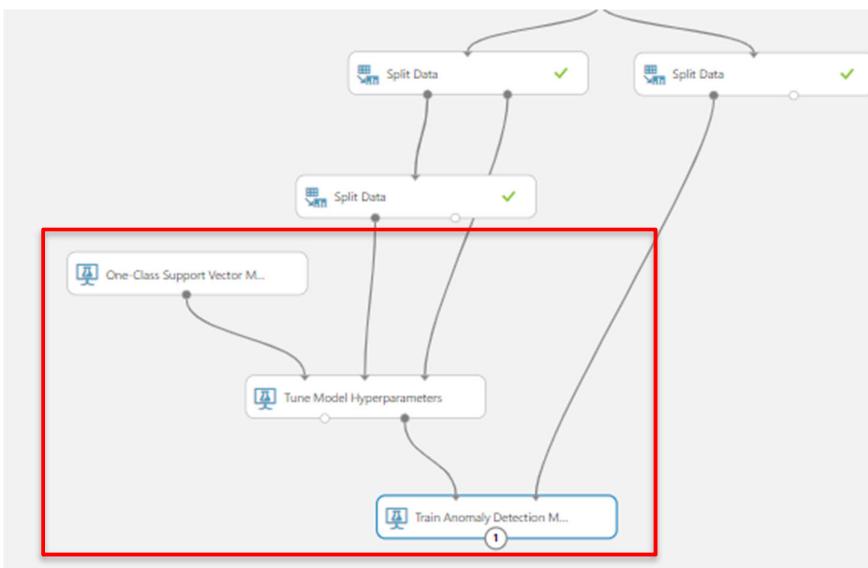
- 9) Set the properties for the **Split Data** module marked (3) as follows (Similar to step (8) above) :

▲ Split Data

Splitting mode: Regular Expression

Regular expression: \\"Label" ^1

- 10) Add and connect the following modules as shown:



Ref:

<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/one-class-support-vector-machine>

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

- 11) Set the properties for the **One-Class Support Vector Machine** as follows:

▲ **One-Class Support Vector Machine**

Create trainer mode

Parameter Range

η

Use Range Builder
0.001, 0.01, 0.1, 1

ε

Use Range Builder
0.00001, 0.001, 0.01

- 12) Set the properties for the **Tune Model Hyperparameters** module as follows:

▲ **Tune Model Hyperparameters**

Specify parameter sweeping mode

Entire grid

Label column

Selected columns:
All labels

Launch column selector

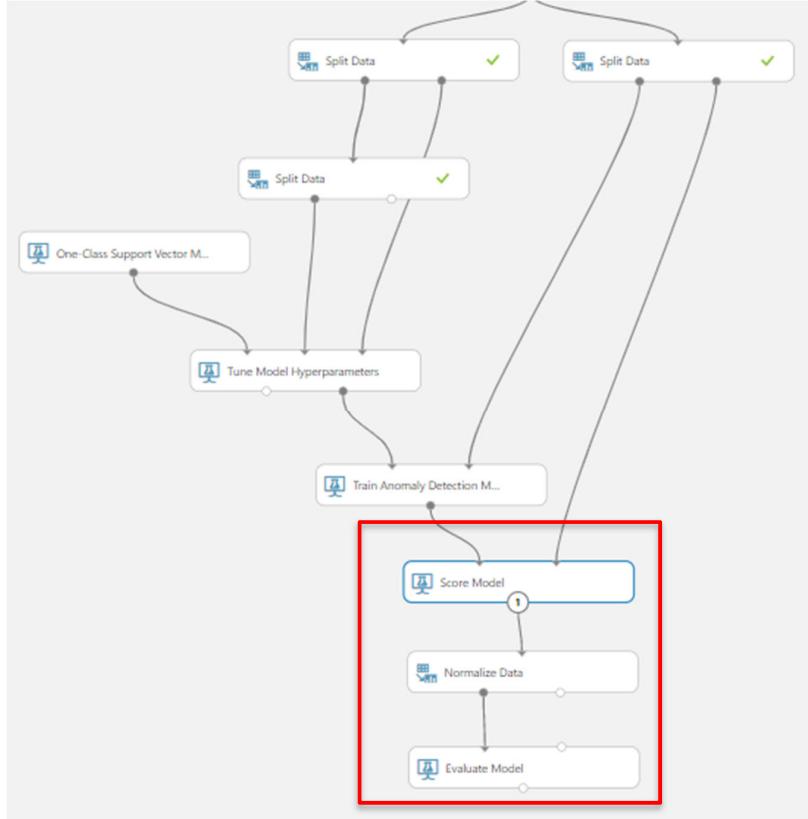
Metric for measuring performance

F-score

Metric for measuring performance

Mean absolute error

- 13) Add and connect the following modules as shown:



Ref:

Normalize Data : <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/normalize-data>

- 14) Set the properties for the **Normalize Data** module as follows

▲ Normalize Data

Transformation method

Logistic

Columns to transform

Selected columns:

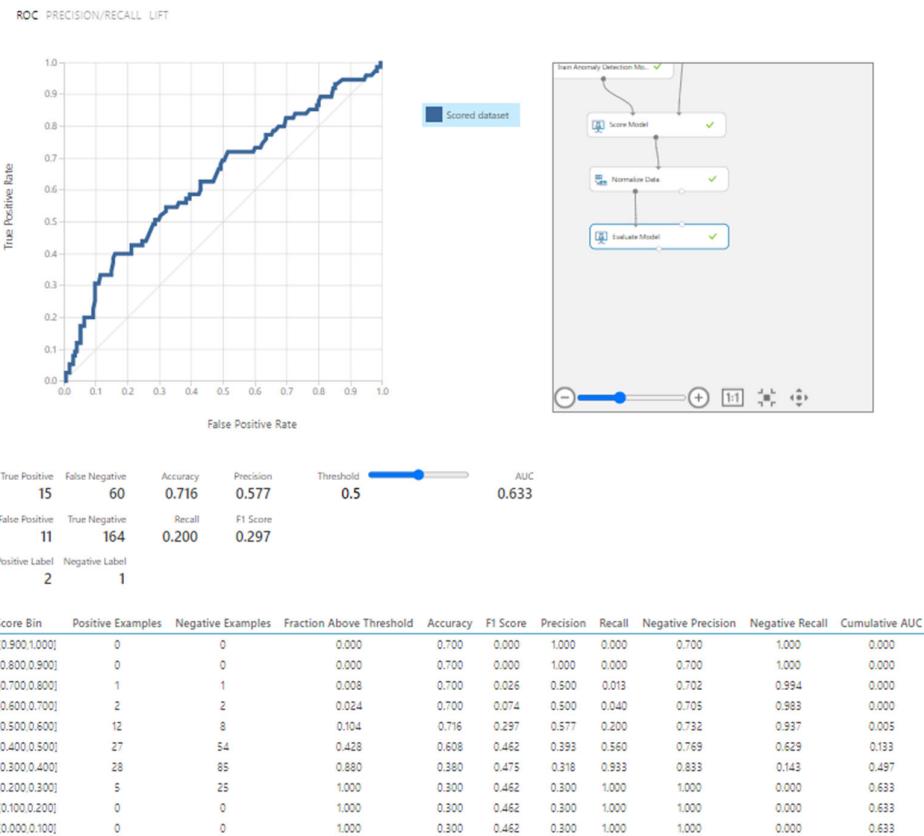
Column names: Scored
Probabilities

Launch column selector

- 15) Click Run.

- 16) Once it finished running. Right-click on the output port of **Visualize** module. You should see the following.

Anomaly Detection > Evaluate Model > Evaluation results



We are getting an AUC of 0.633. Not too bad.

- 17) **Exercise.** Use the **PCA-Based Anomaly Detection** module to build another model for the same dataset and compare which gives you better performance.