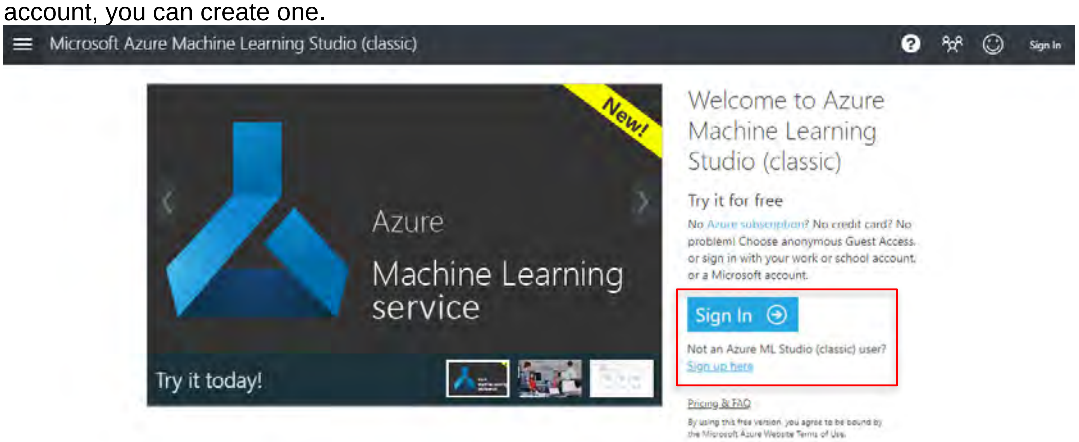
Contents

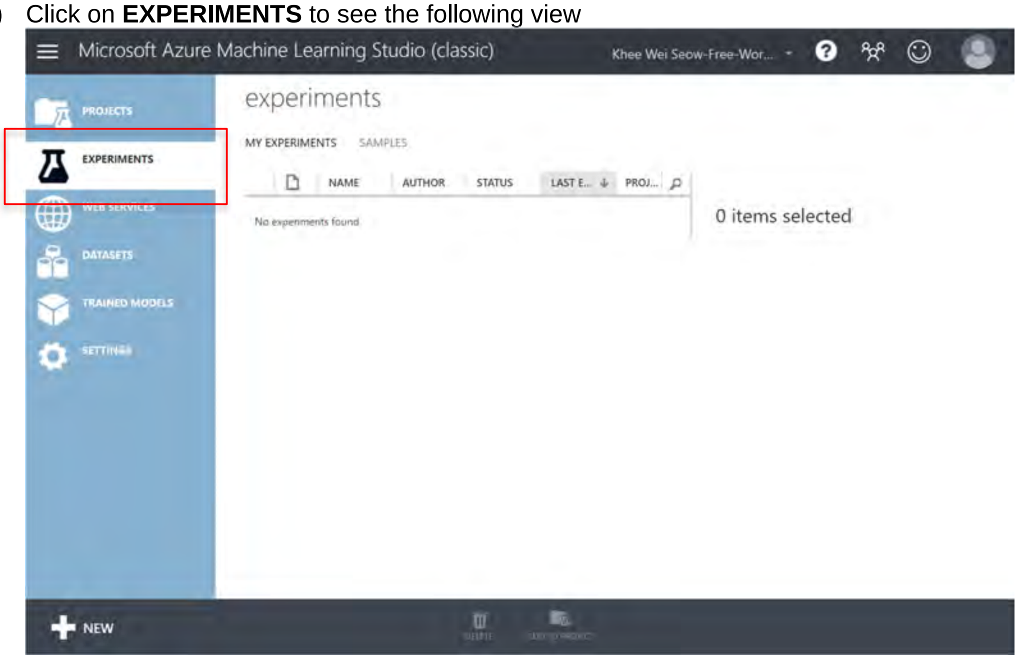
Activity 1 – First Machine Learning with Azure

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

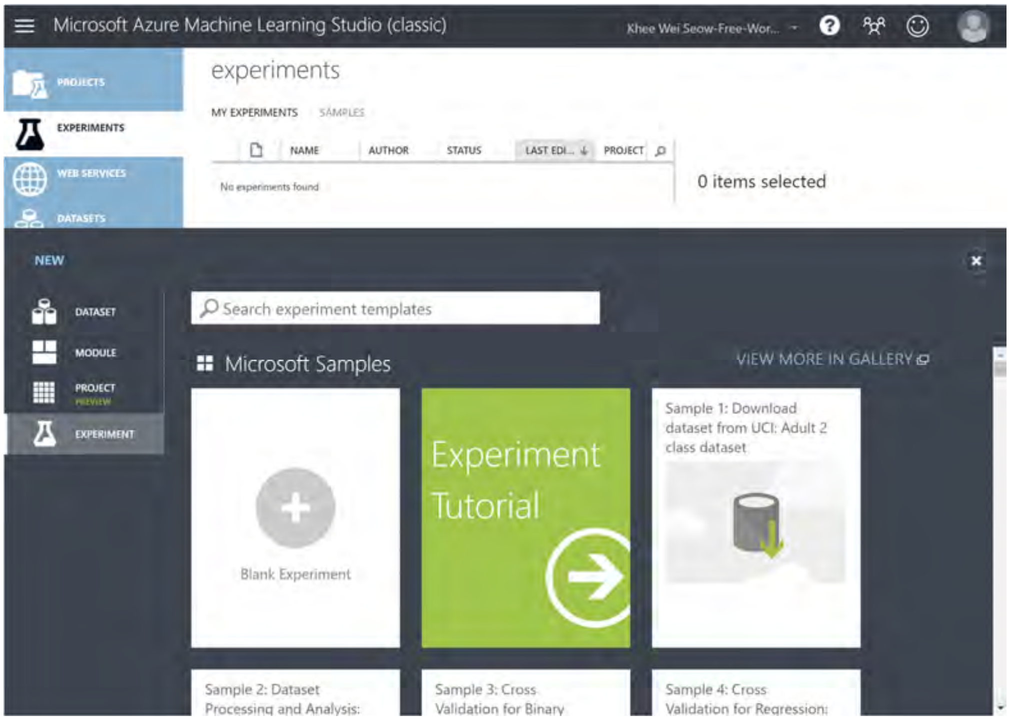
1) **Setup account on Azure**
   1) Launch your web browser, navigate to https://studio.azureml.net/ and sign in. If you have not created an account, you can create one.



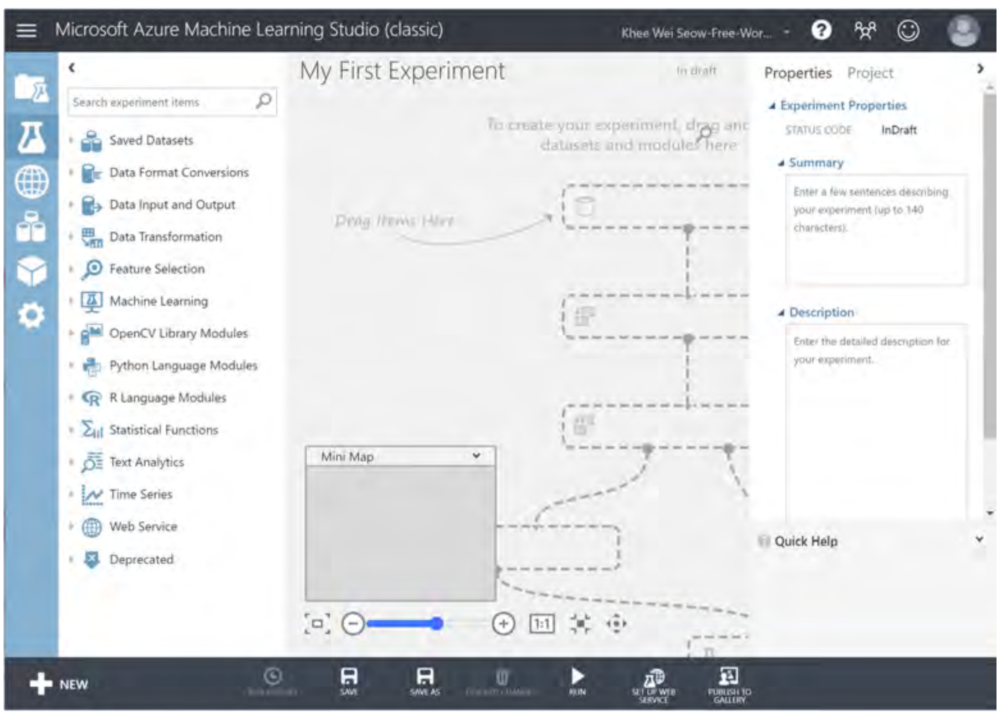
2) **Create a new Experiment**
   1) Click on **EXPERIMENTS** to see the following view



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

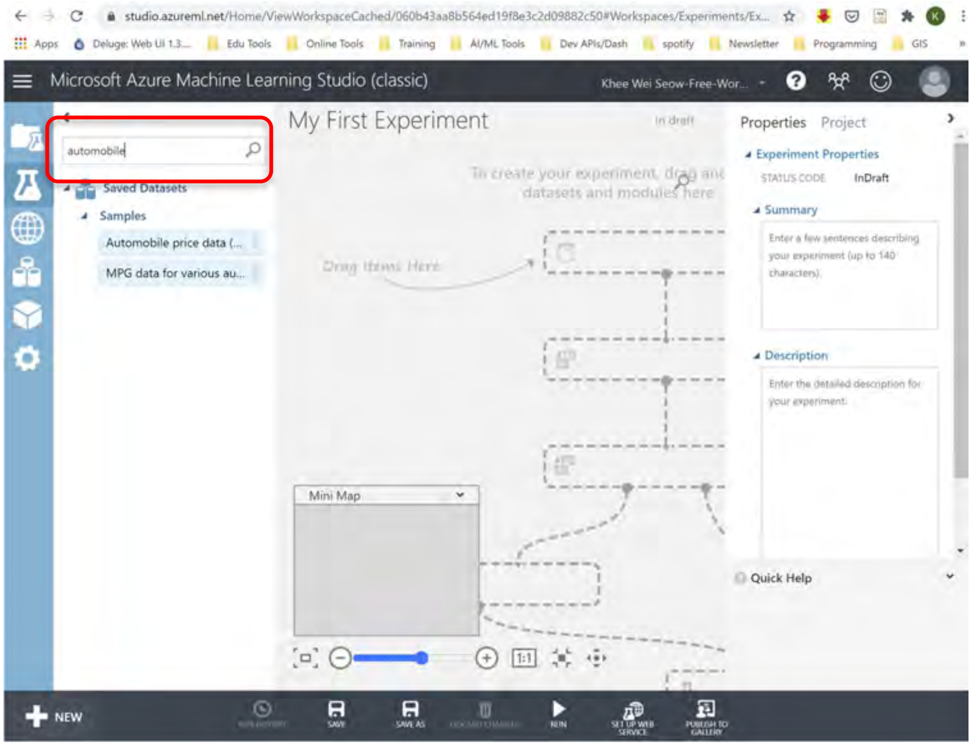


3) Name it "**My First Experiment**". A screen similar to the follow will be presented.

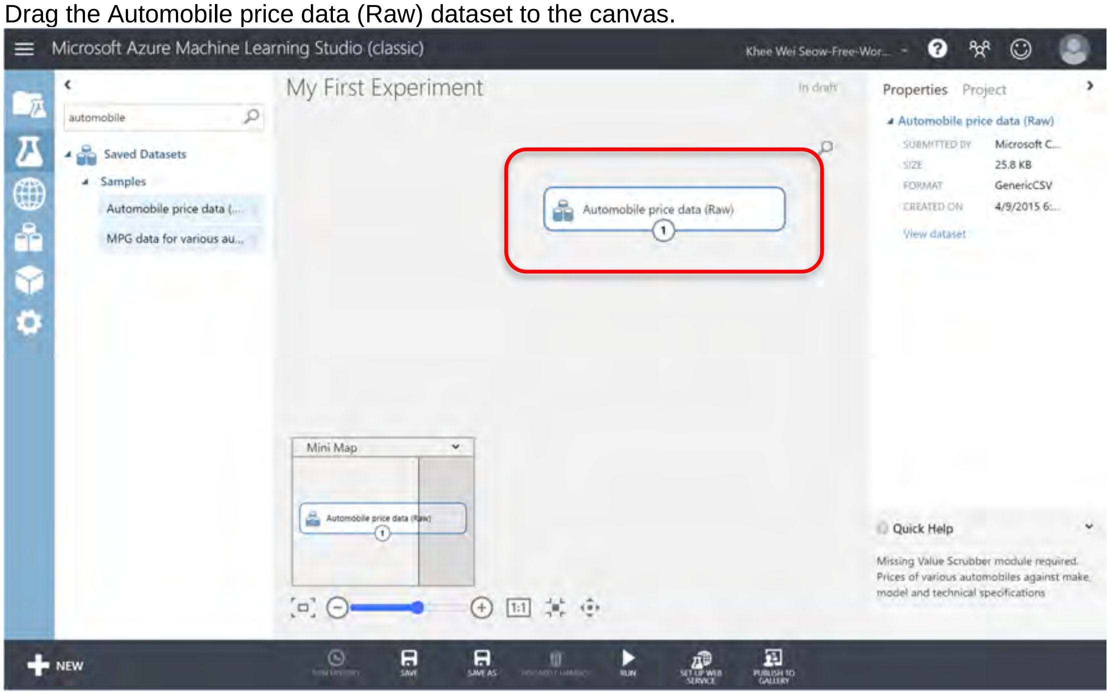


3) Setting up your data

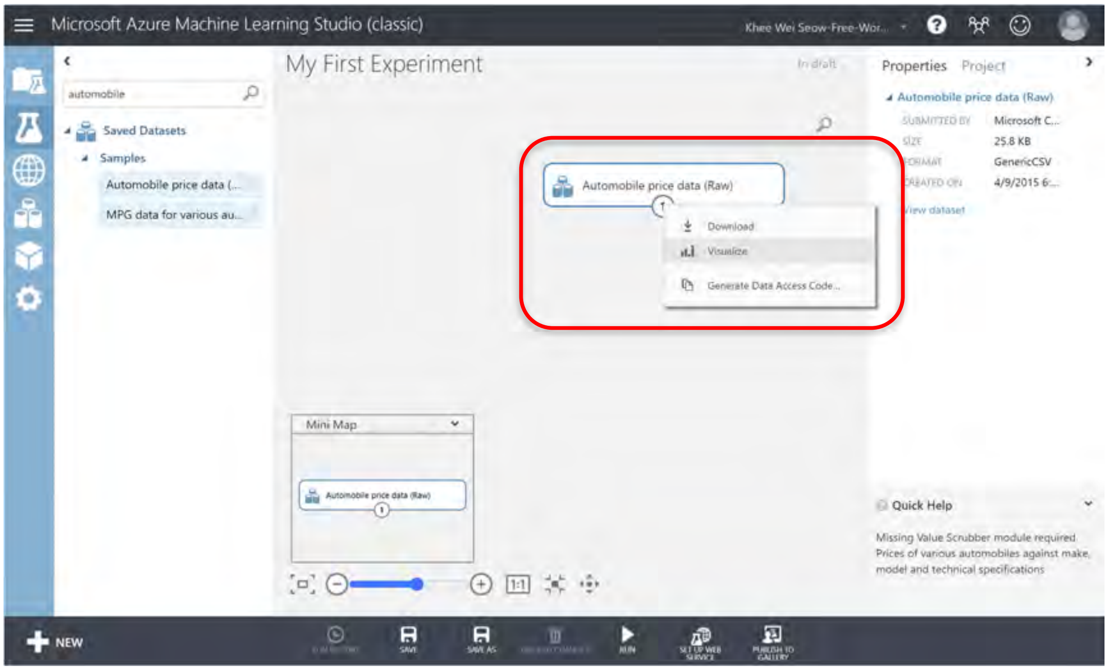
1) In the left search box, type Automobile.



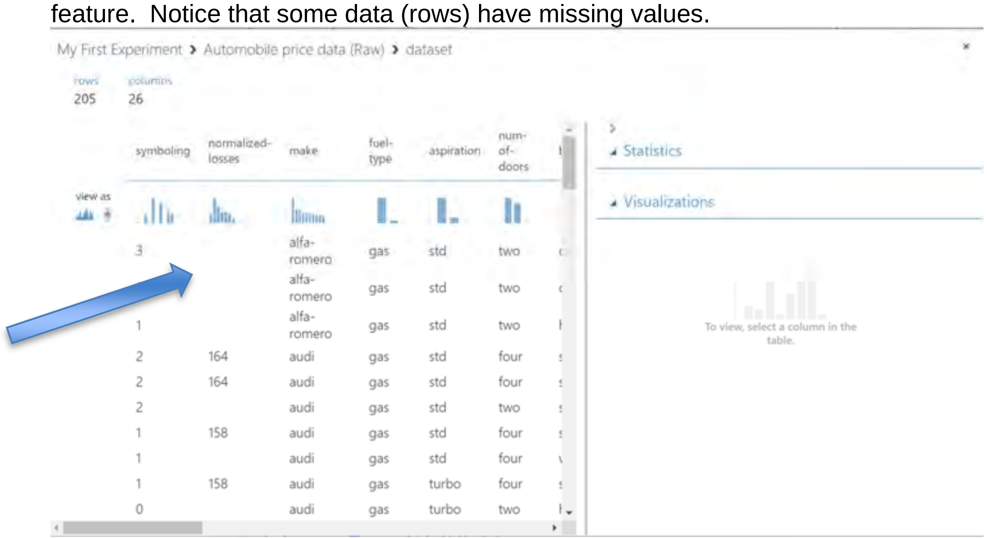
2) Drag the Automobile price data (Raw) dataset to the canvas.



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

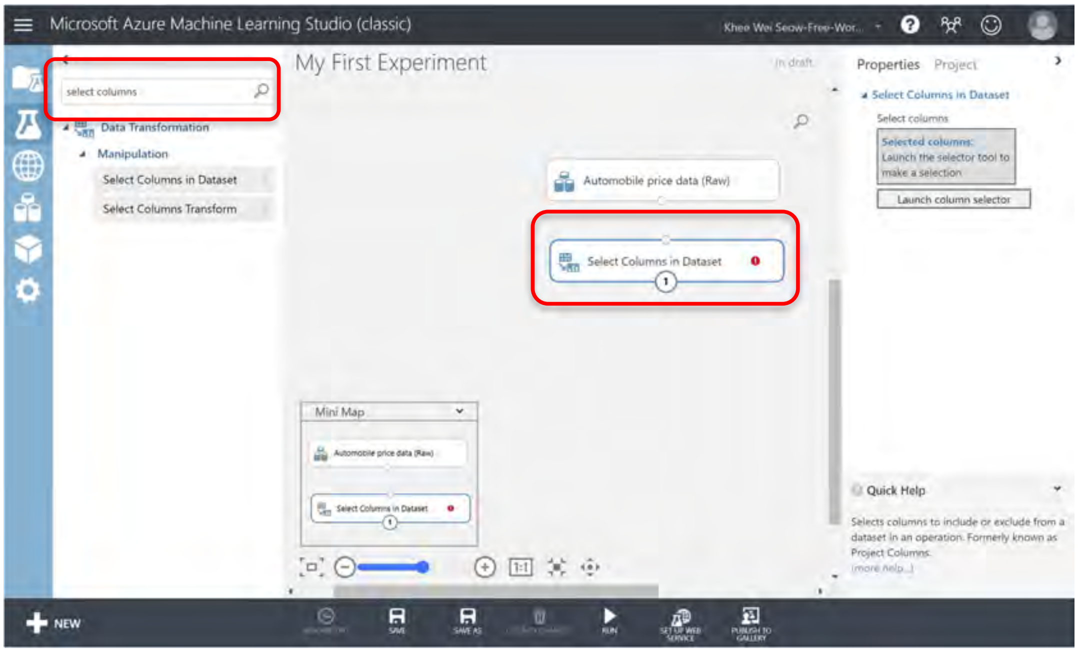


4) You should see the content of the dataset as shown below. Every column in the dataset is also known as a feature. Notice that some data (rows) have missing values.

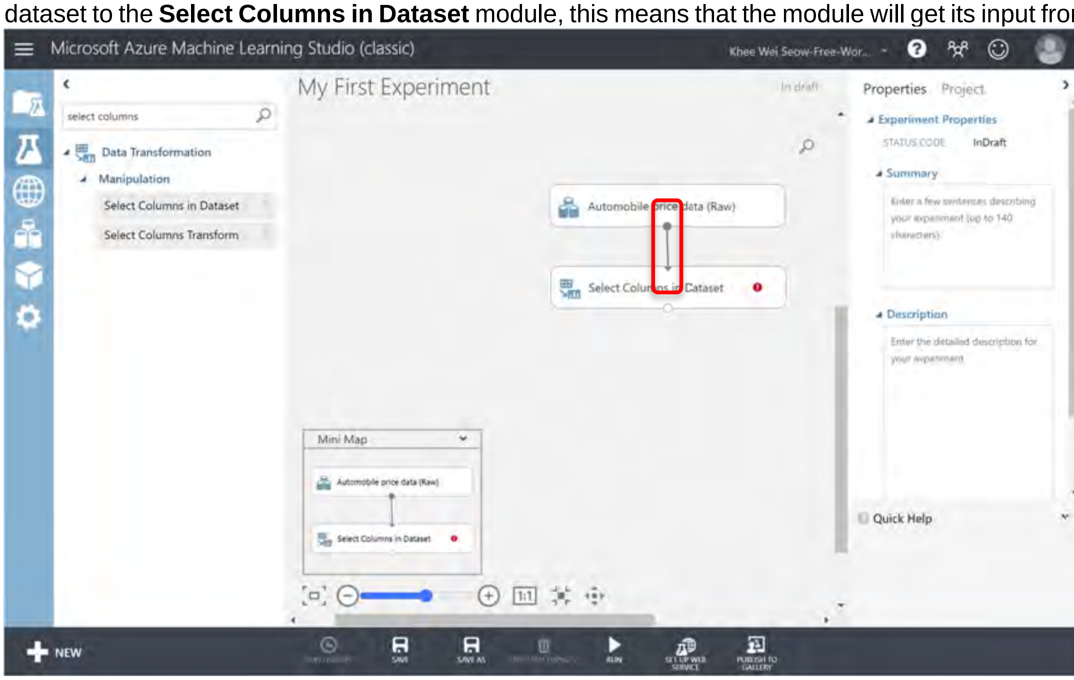


4) Preparing your Dataset

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.

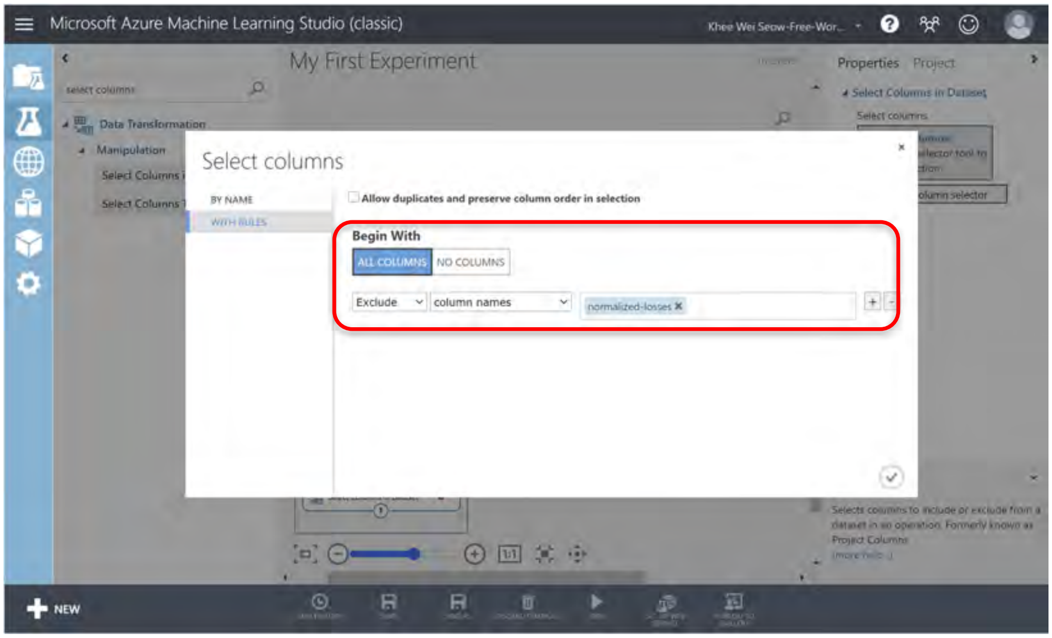


2) Connect the output port of the dataset to the input port of the module as shown below. By connecting the dataset to the **Select Columns in Dataset** module, this means that the module will get its input from the dataset.



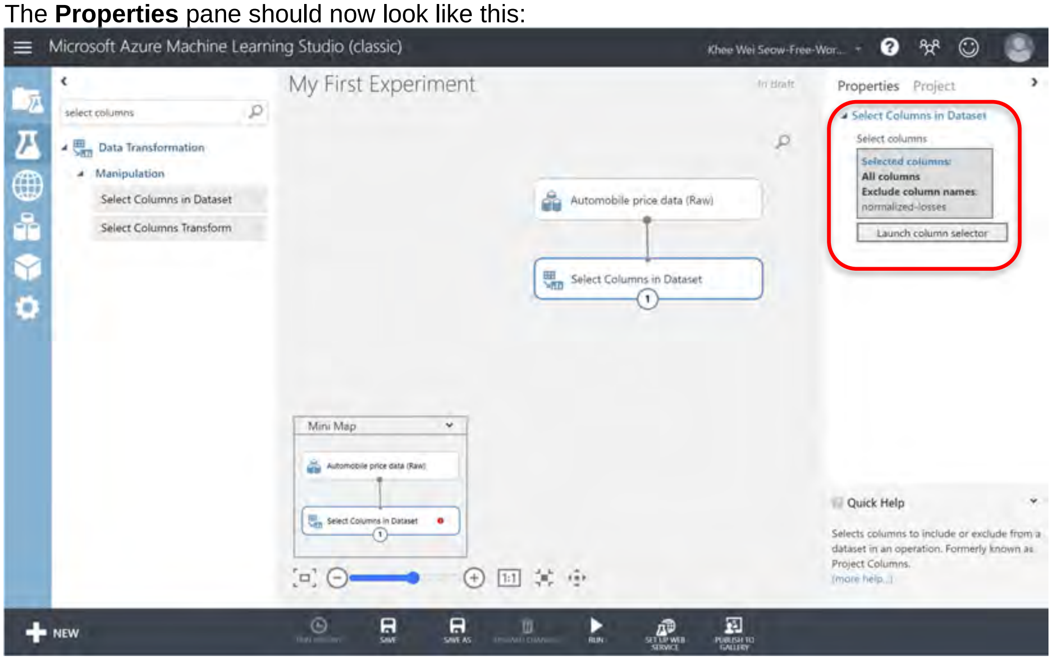
3) Select the **Select Columns in Dataset** module and on the Properties pane on the right, click the **Launch column selector** button.

4) Set the values as shown below. Click the check mark button when done.

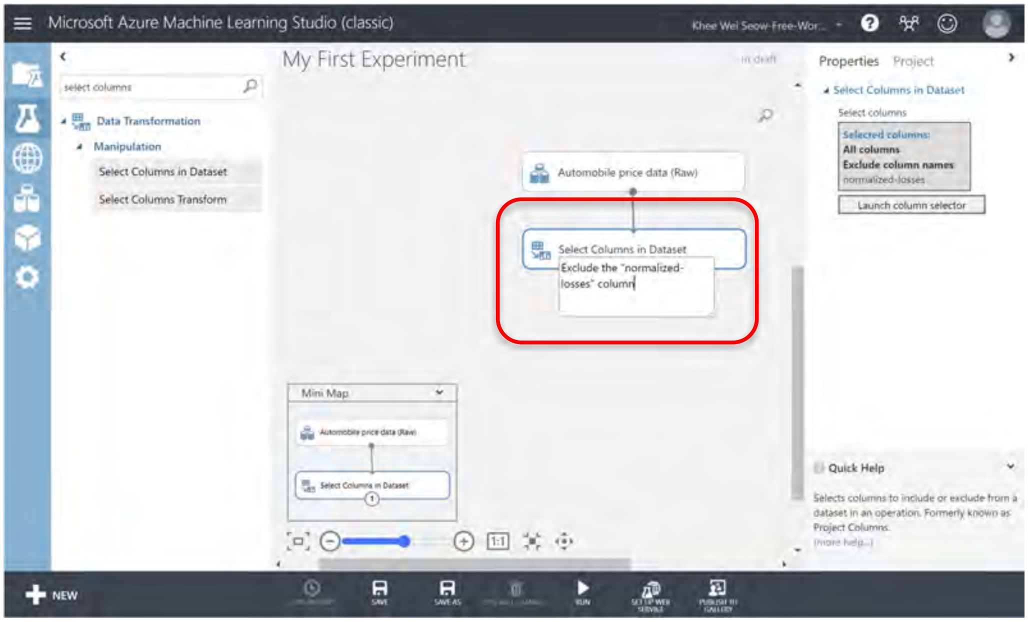


- This rule specifies that you want to exclude the ***normalised-losses*** column from the dataset.

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

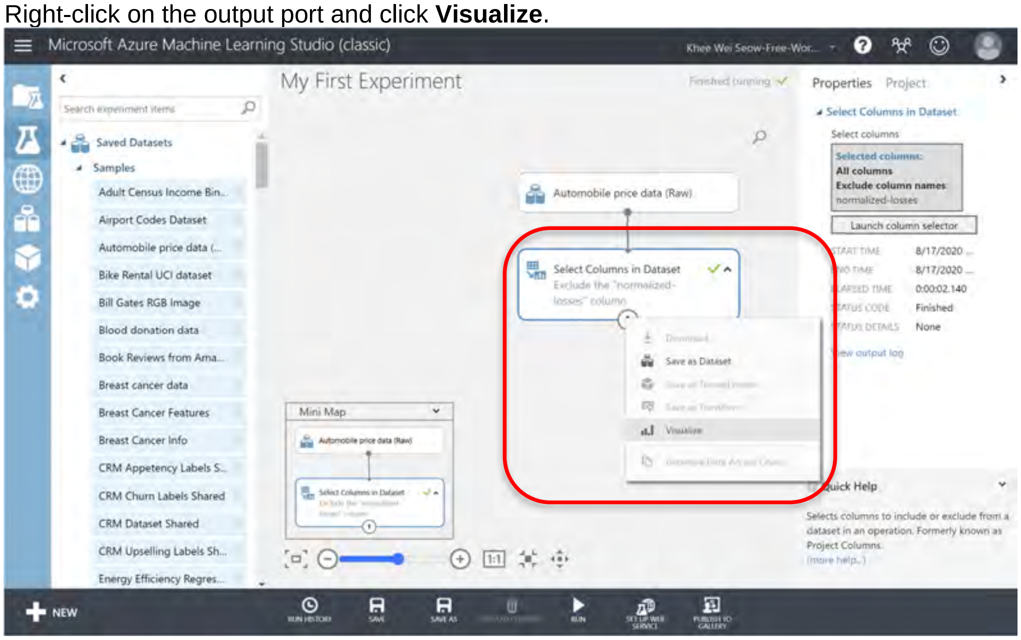


6) Double click on the **Select Columns in Dataset** module to add a comment.

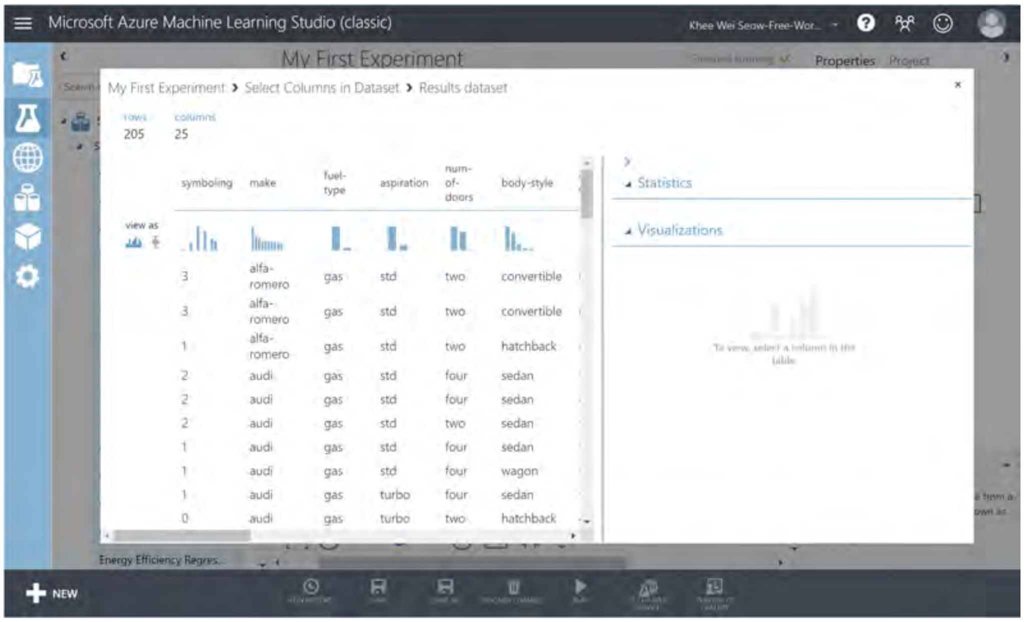


7) Click **Run** button located at the bottom of the screen. You will now see a green mark displayed in the **Select Columns in Dataset** module.

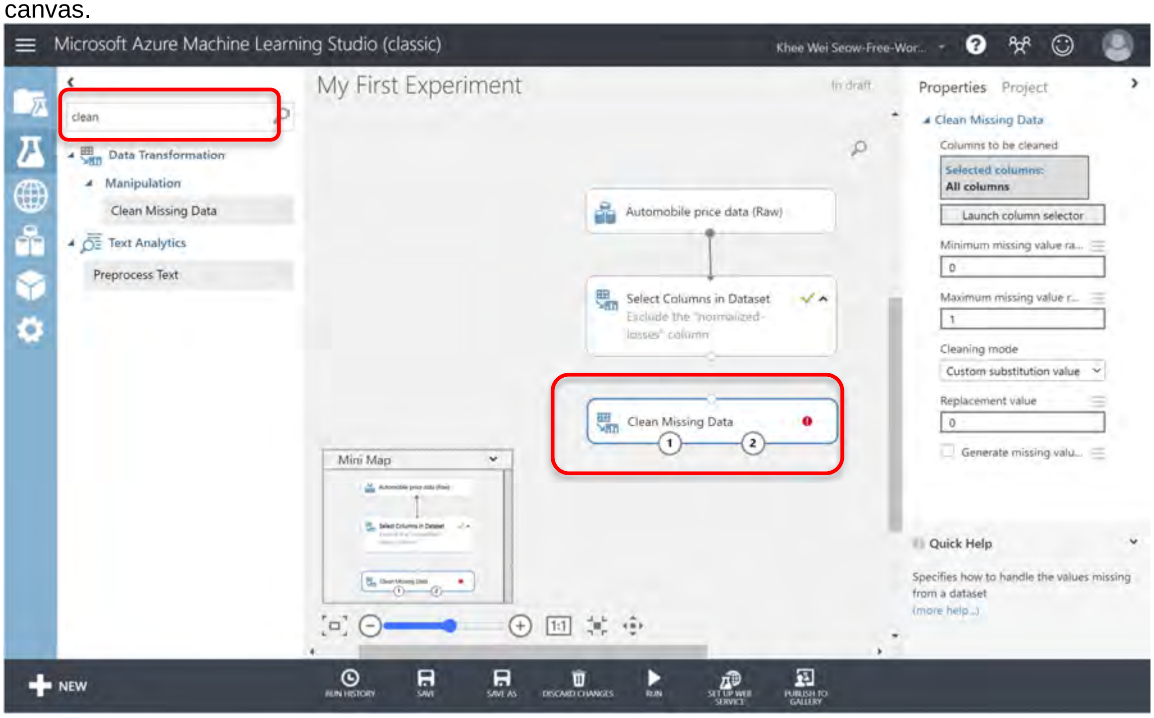
8) Right-click on the output port and click **Visualize**.



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

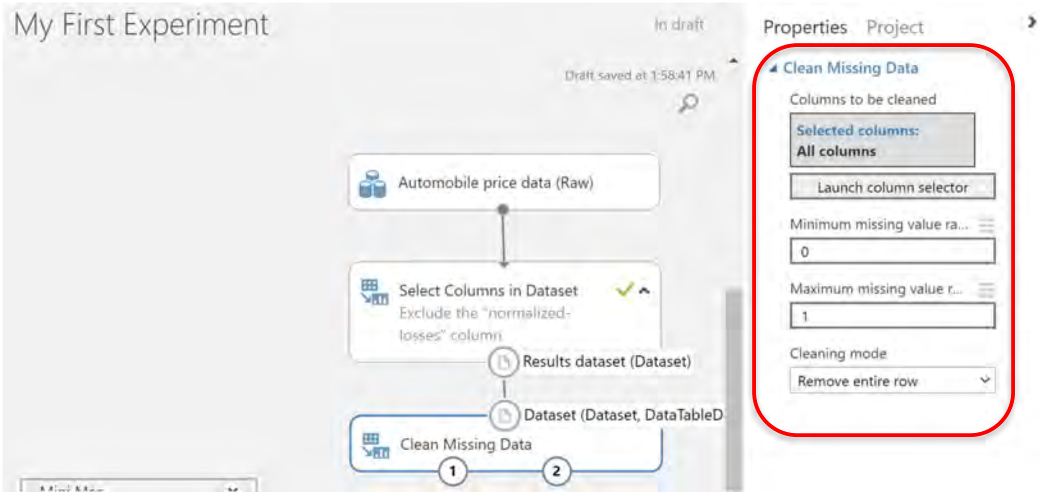


10) **Cleaning Data** – In the search box, type Clean missing and Drag the **Clean Missing Data** module to the canvas.



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

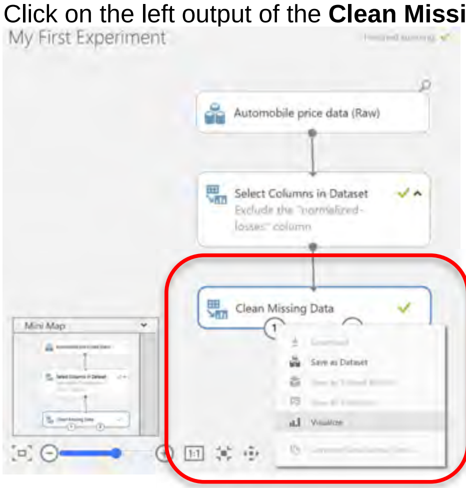
12) Select the **Clean Missing Data** module and set its **properties** as follows:



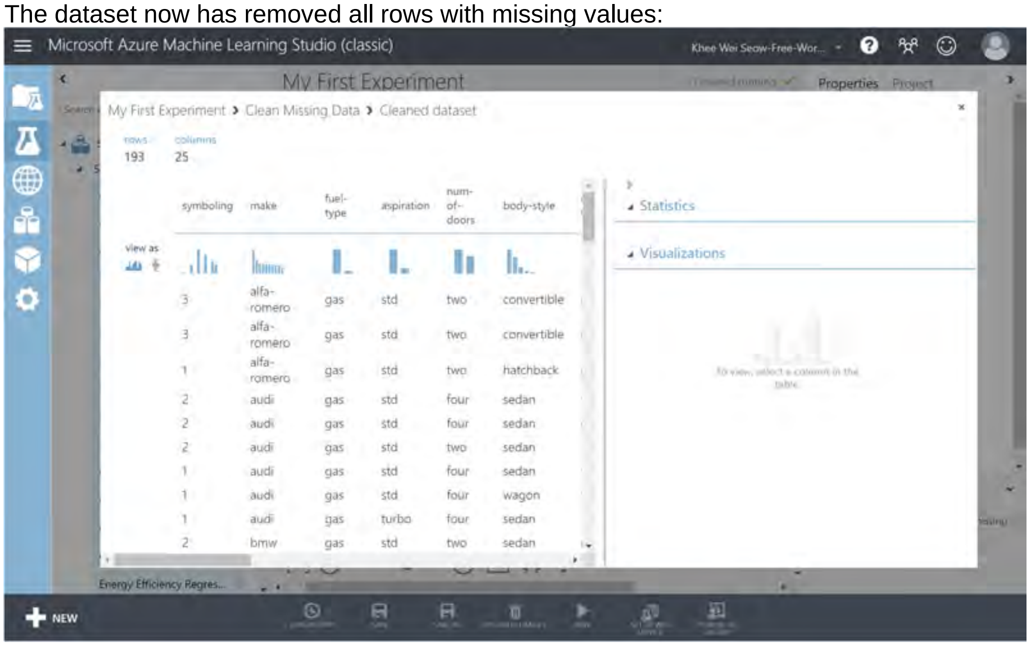
This property removes all rows with missing values. For other options available, you may want to refer to https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/clean-missing-data

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

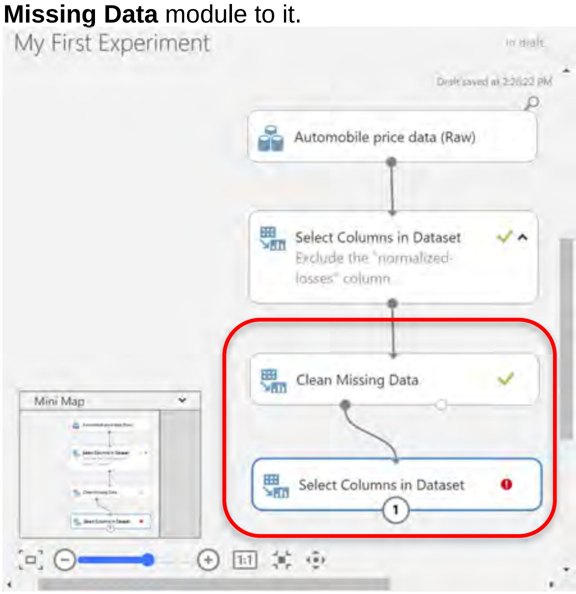
14) Click on the left output of the **Clean Missing Data** module and click **Visualize**.



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



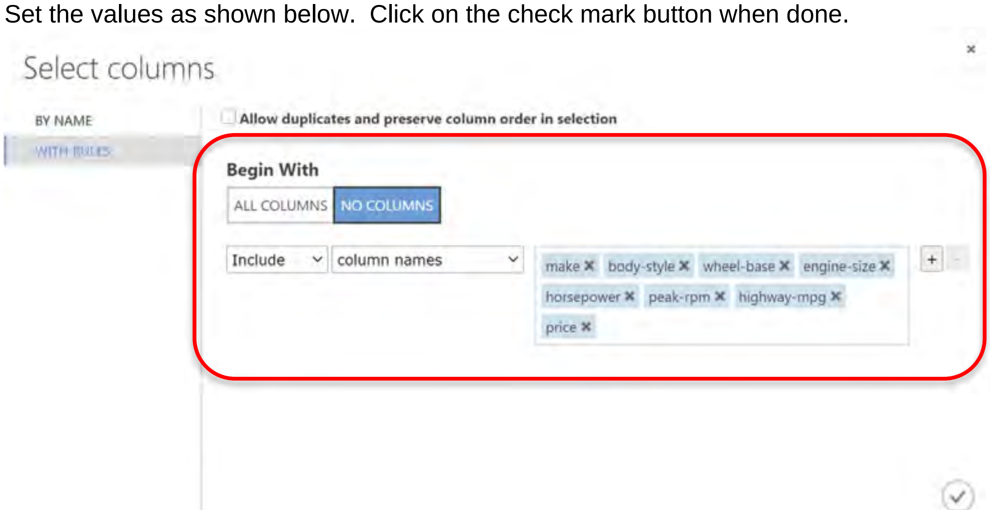
16) **Defining Features** – Add another **Select Columns in Dataset** module to the canvas and connect the **Clean Missing Data** module to it.



17) Double-click on the module and type **Select features for prediction**. (For comment purpose)

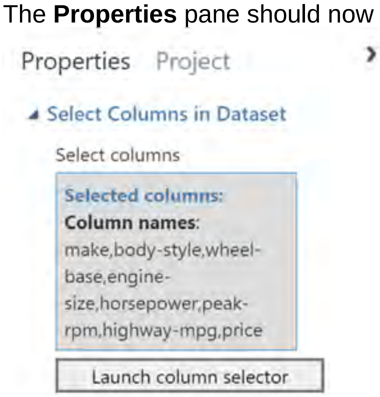
18) Select the module and in the **Properties** pane, click **Launch column selector**.

19) Set the values as shown below. Click on the check mark button when done.

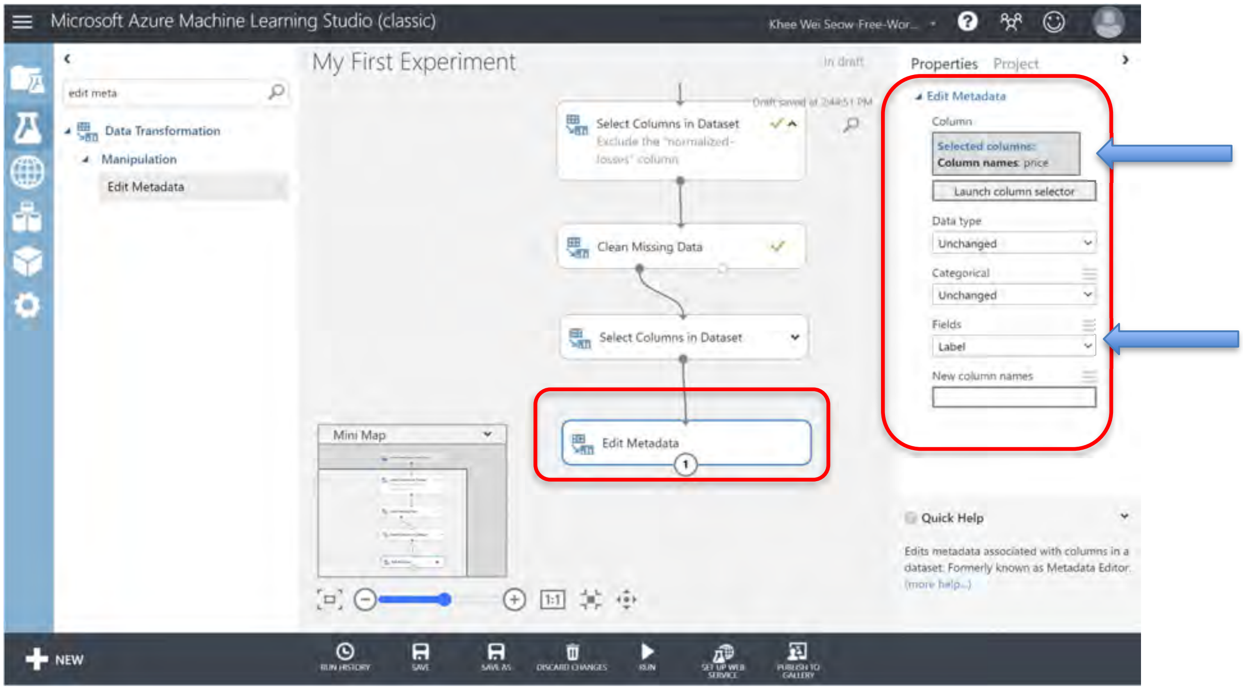


- We are effectively including the following columns: ***make, body-style, wheel-base, engine-size, horsepower, peak-rpm, highway-mpg, price***

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

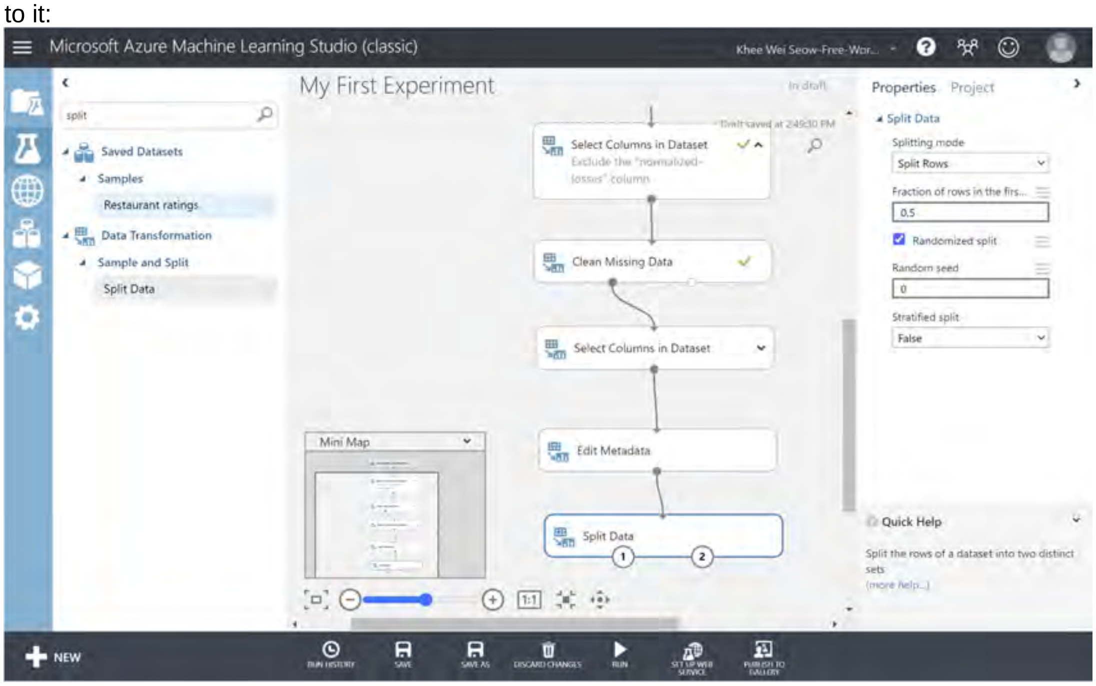


21) **Labelling a Feature** – Add the **Edit Metadata** module to the canvas and then connect and configure it as follows:



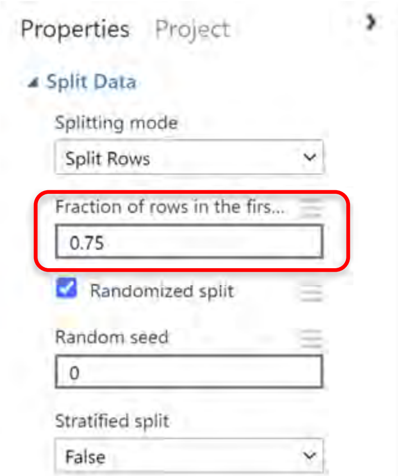
In this case, we are specifying the **price** column as the label. **A label defines the output of your learning model.** I.e. What we are going to predict.

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



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

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



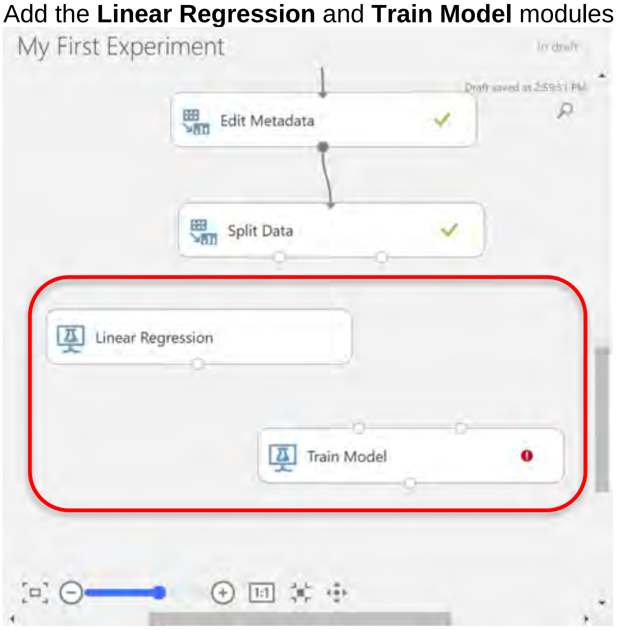
Ref: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/split-data

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

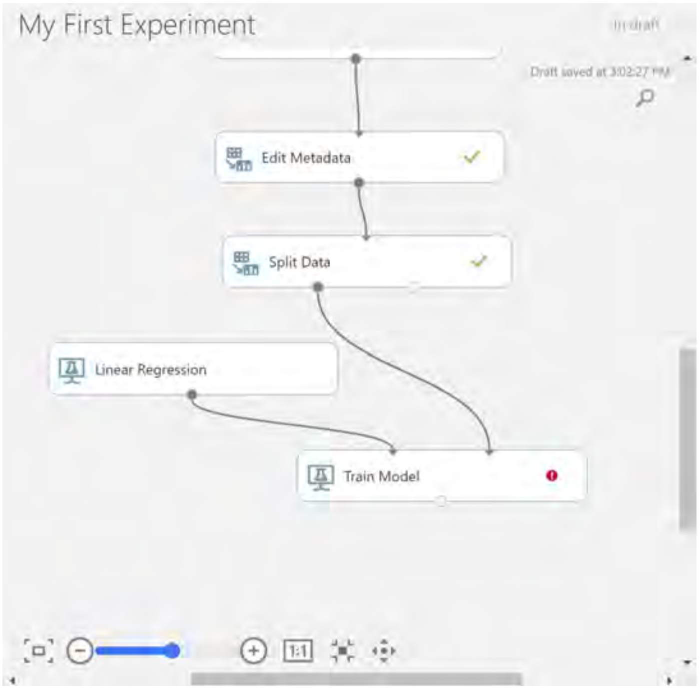
24) Click **Run**.

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

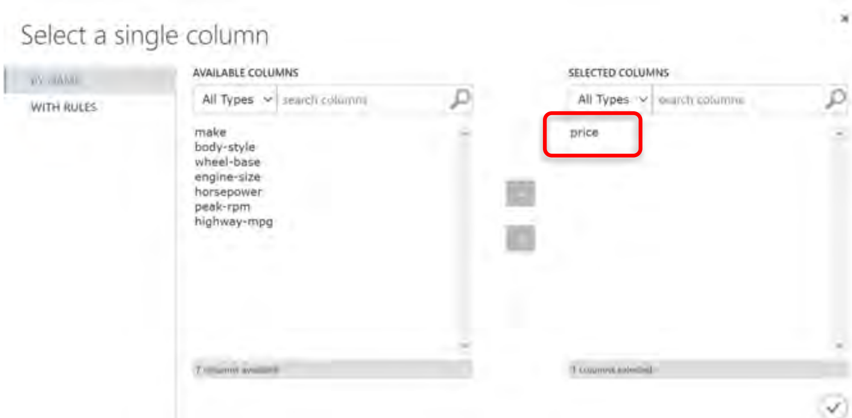
1) Add the **Linear Regression** and **Train Model** modules to the canvas.



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

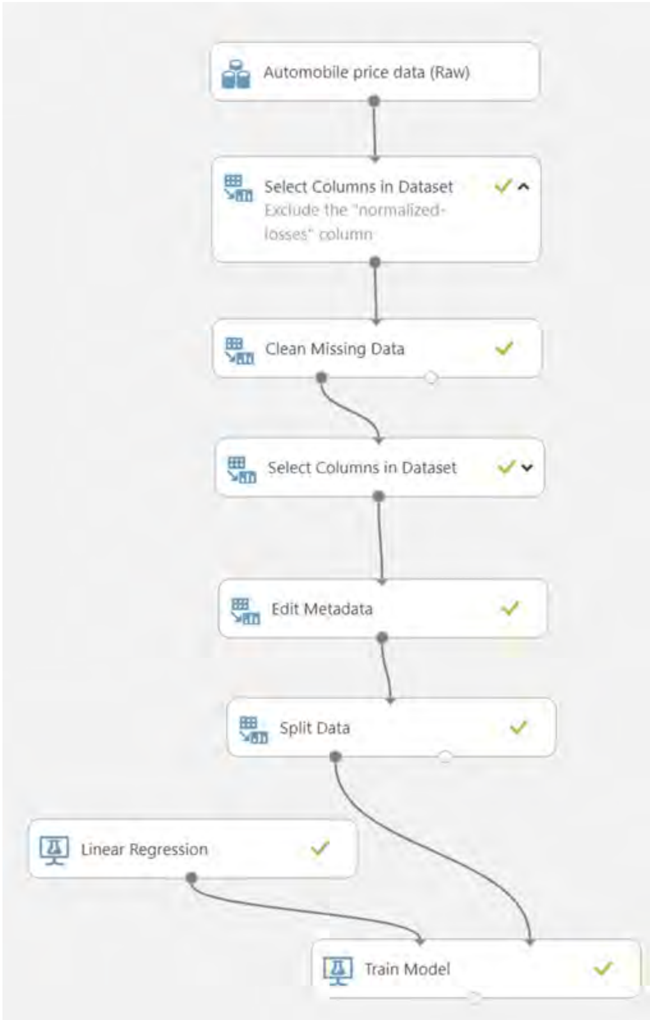


3) Select the **Train Model** module and in the **Properties** pane, click the **Launch column** selector button. Select **price** in the AVAILABLE COLUMNS pane and click on the > button to move it to the SELECTED COLUMNS pane. Click on the tick button to finish this step.

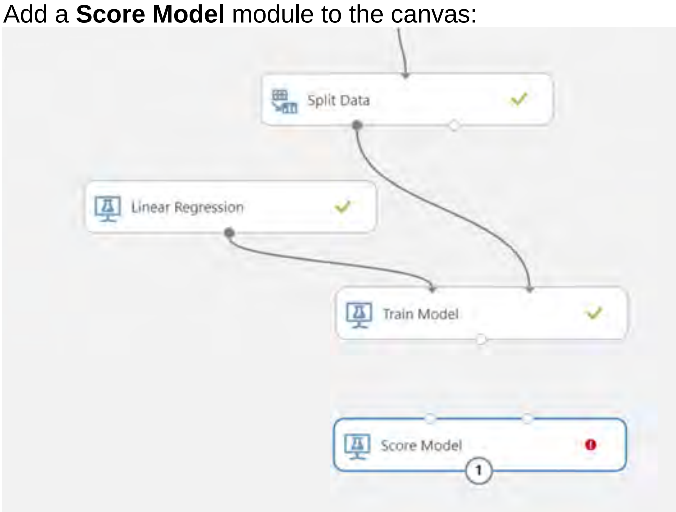


The above step indicates that you want the training model to predict the prices of vehicles.

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



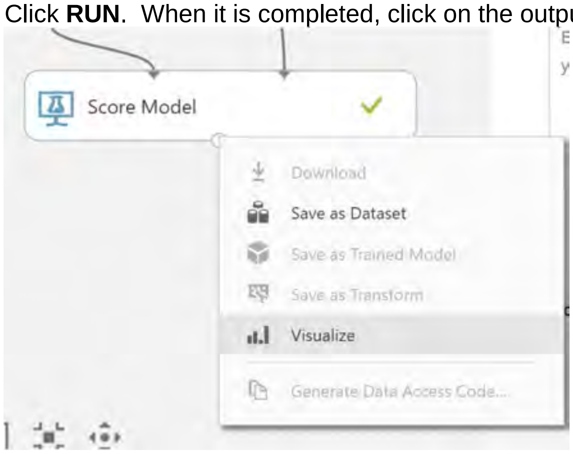
5) Add a **Score Model** module to the canvas:



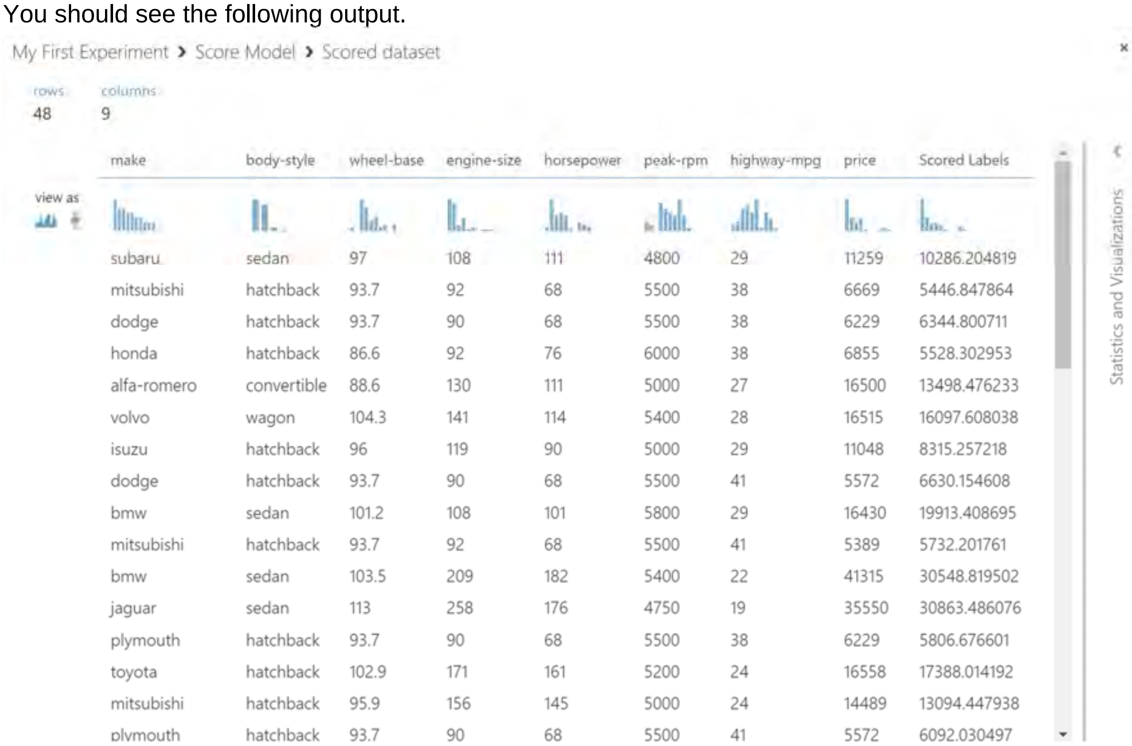
Ref: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/score-model

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

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



8) You should see the following output.

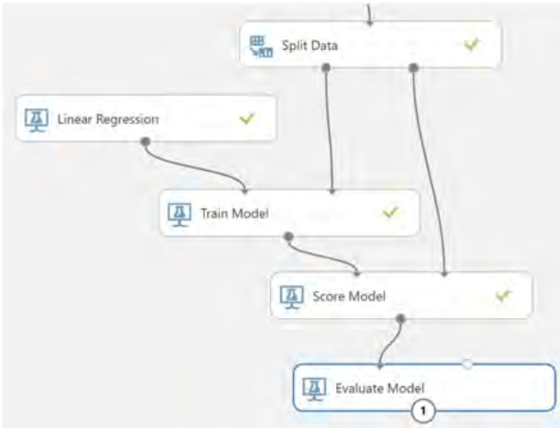


make	body-style	wheel-base	engine-size	horsepower	peak-rpm	highway-mpg	price	Scored Labels
subaru	sedan	97	108	111	4800	29	11259	10286.204819
mitsubishi	hatchback	93.7	92	68	5500	38	6669	5446.847864
dodge	hatchback	93.7	90	68	5500	38	6229	6344.800711
honda	hatchback	86.6	92	76	6000	38	6855	5528.302953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

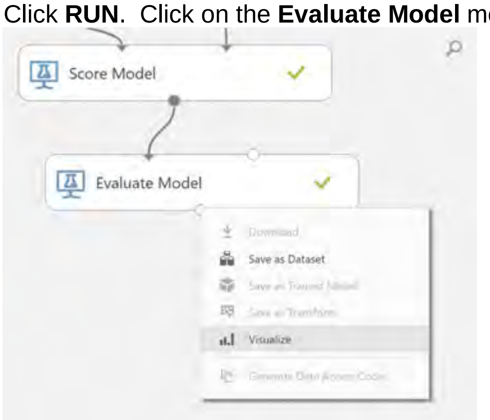
The price column shows the actual values and the Scored Labels columns shows the predicted values.

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:

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.

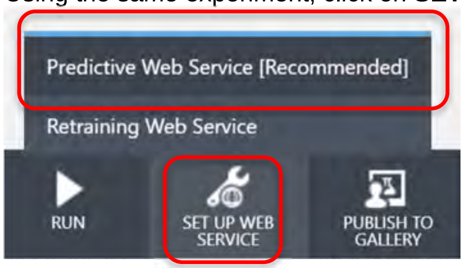
Coefficient of Determination: Also known as the R squared value, this is a statistical metric indicating how well a model fits the data.

Activity 2 – Deploying your experiment as a Web Service

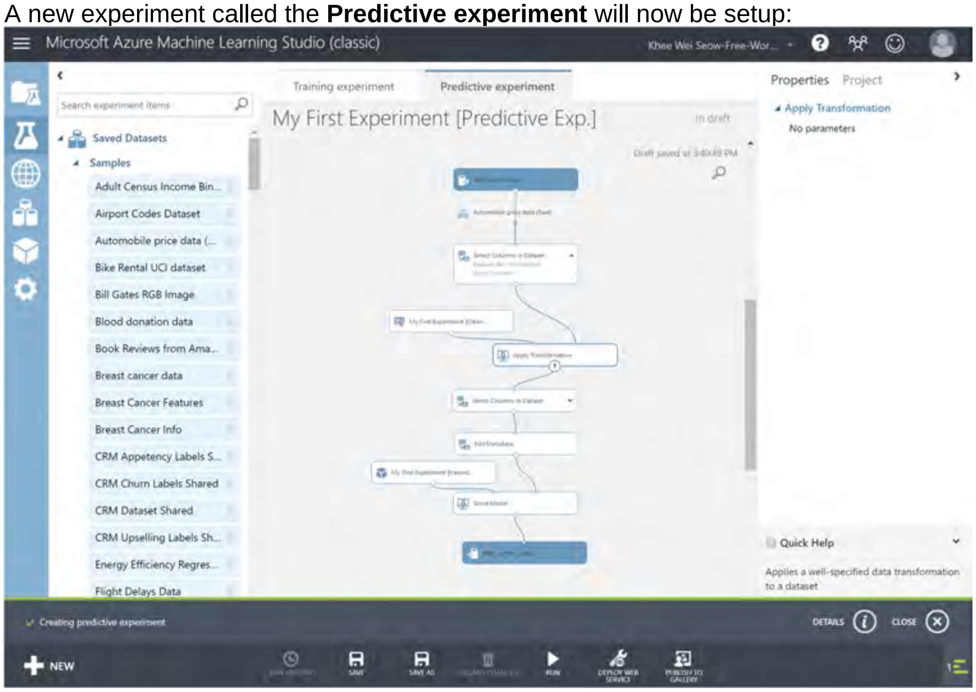
<u>In this activity, we will learn:</u>
- ❑ Deploy a training model as a Web Service
- ❑ Test the webservice
- ❑ [Optional] access the web service using Python 3
- ❑ Use the web services via Excel

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:

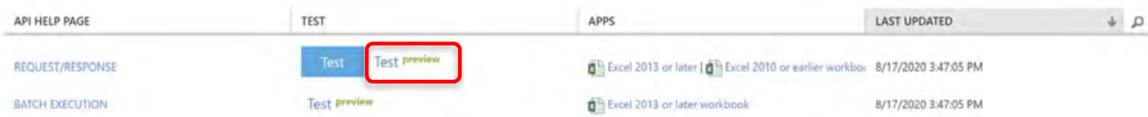


3) Click **RUN**. Then click **DEPLOY WEB SERVICE**.

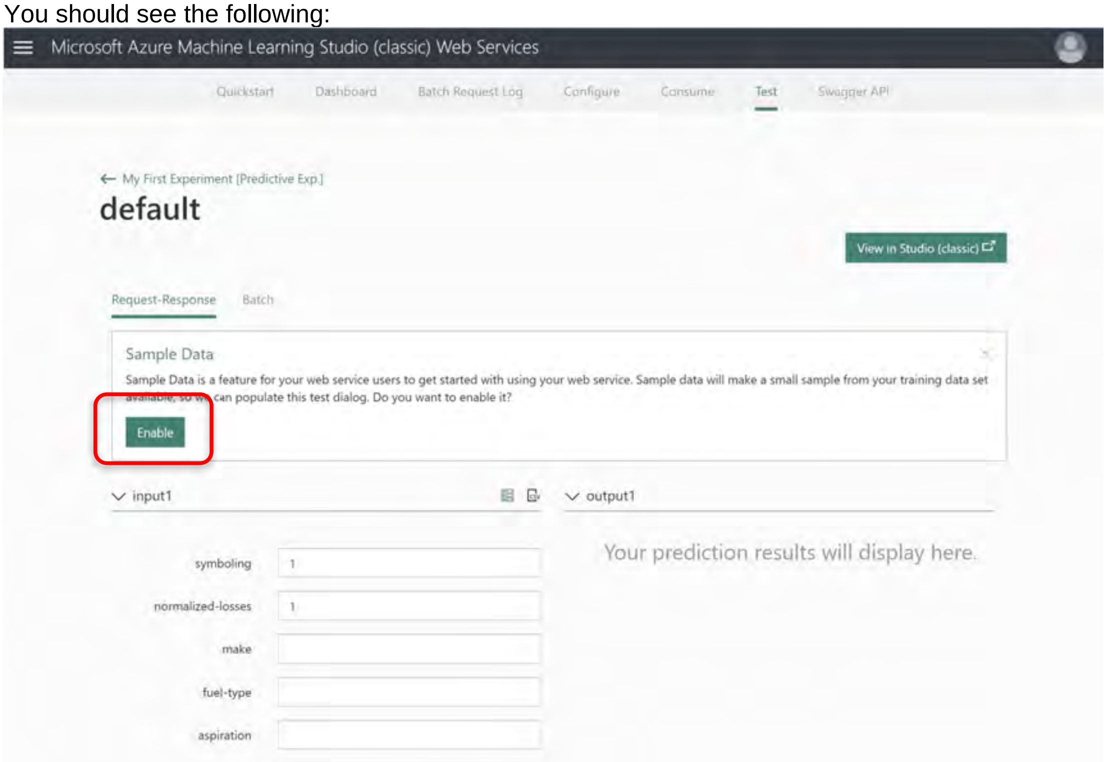
4) You should see the following after a few seconds.

2) Testing the Web Service

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

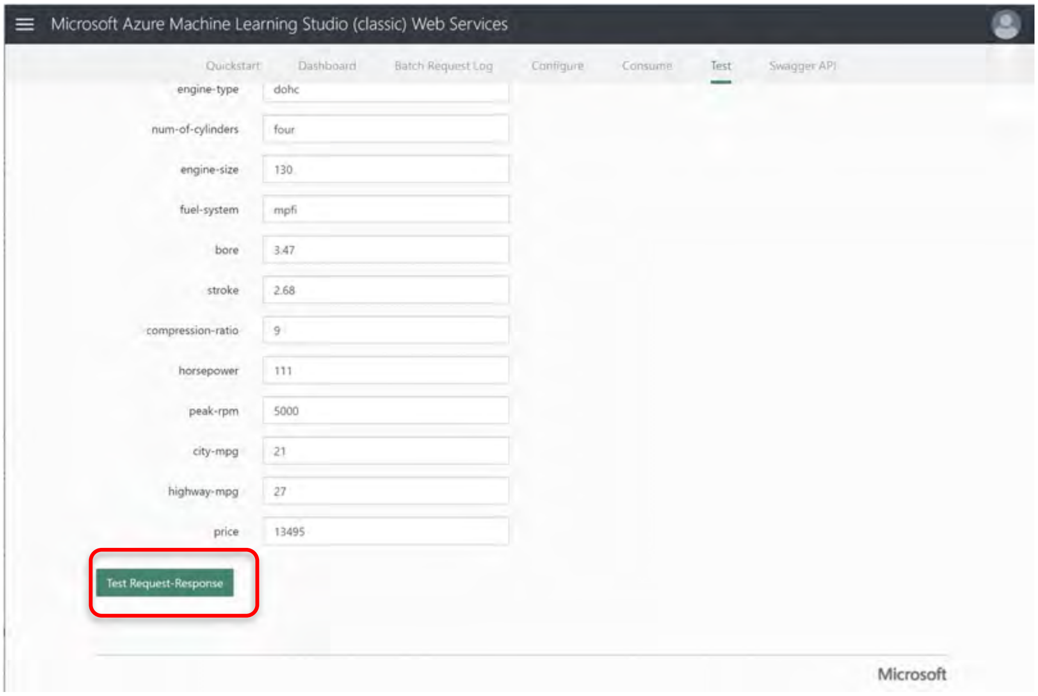


2) You should see the following:

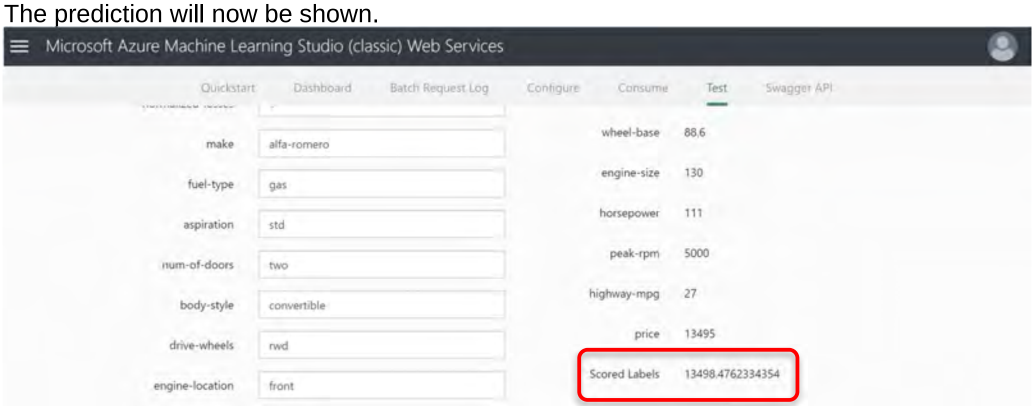


3) Click the **Enable** button to populate the various fields with sample data from your dataset.

4) At the bottom of the page, click the **Test Request-Response** button to test the web service.



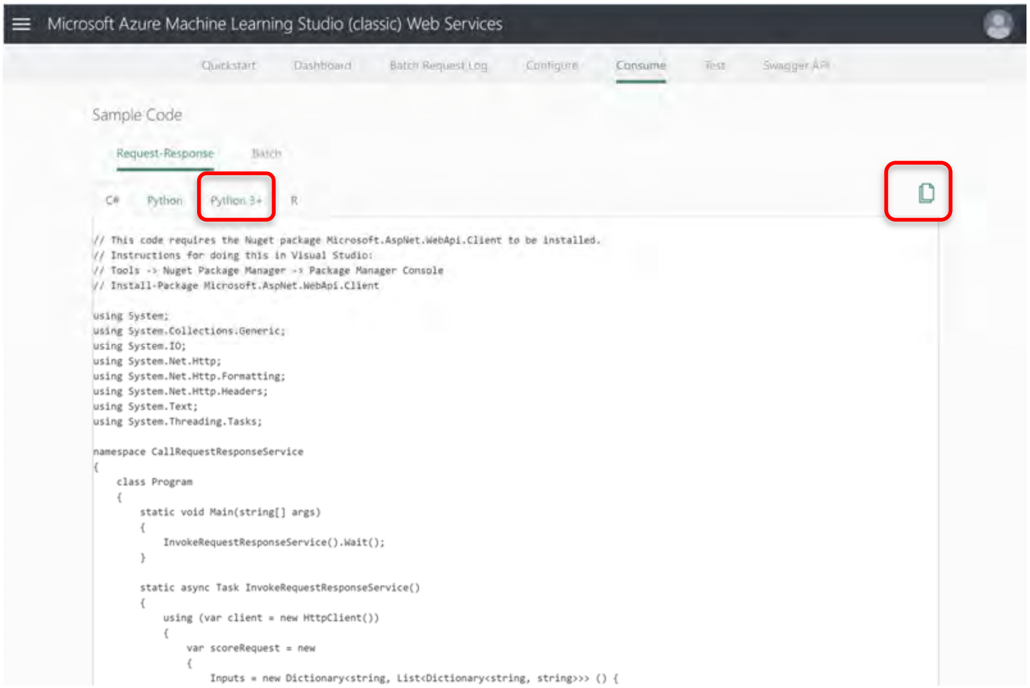
5) The prediction will now be shown.



3) [Optional] Consuming the web service programmatically

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

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

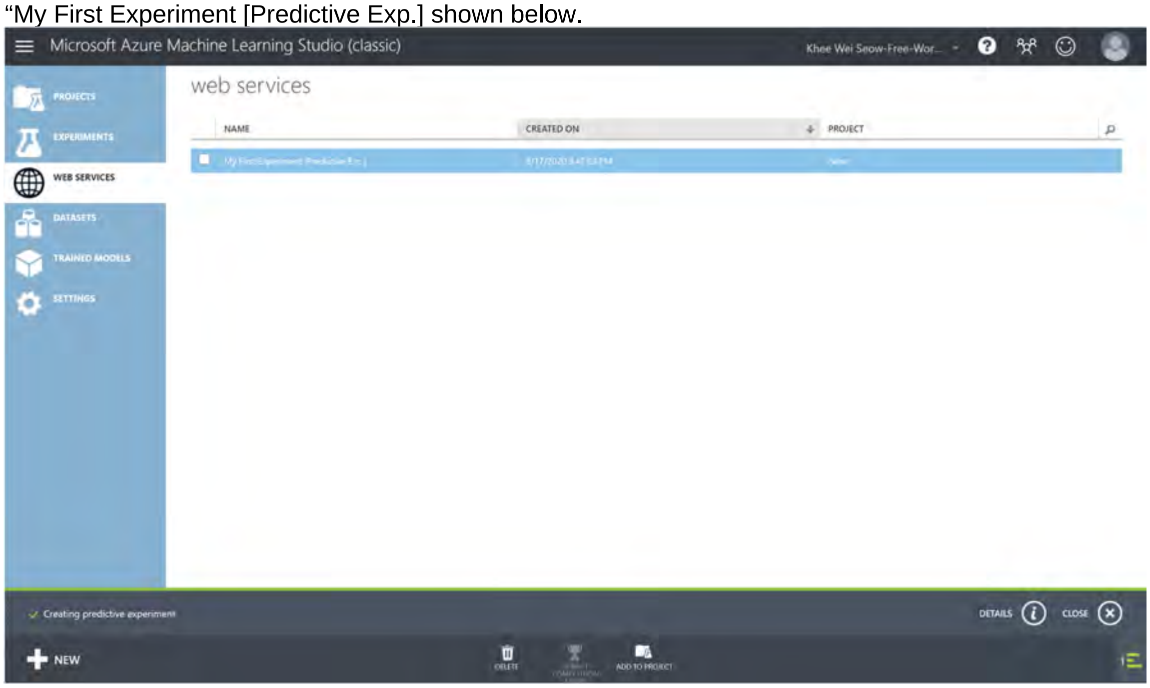


3) Click on **Python 3+** tab and copy the code.

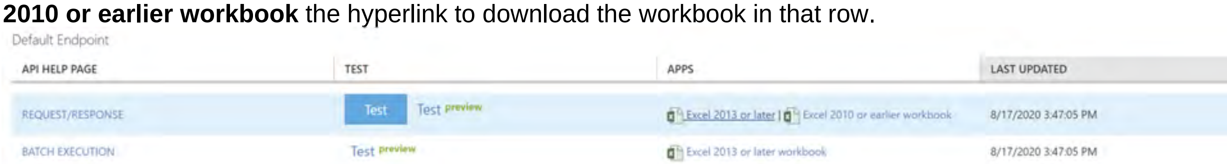
4) Make Prediction using Excel (2010 or earlier)

1) Azure Machine Learning Studio (classic) makes it easy to call web services directly from Excel without the need to write any code. If you are using Excel 2013 (or later) or Excel Online, then we recommend that you use the Excel Excel add-in (next section).

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

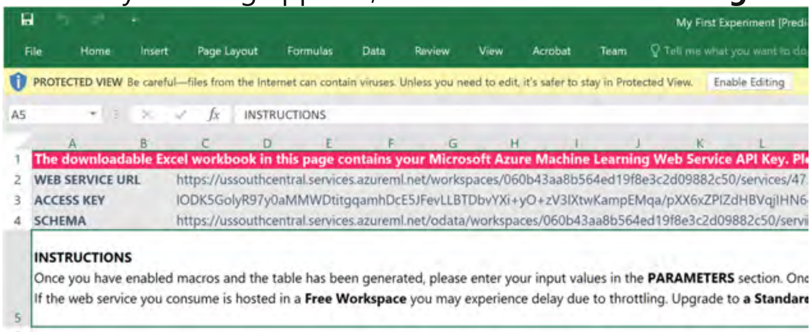


3) On the DASHBOARD tab for the web service is a row for the REQUEST/RESPONSE service. Click on **Excel 2010 or earlier workbook** the hyperlink to download the workbook in that row.

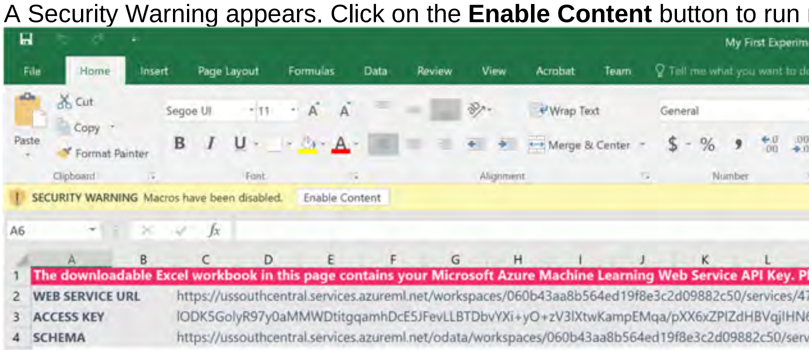


4) Open the workbook.

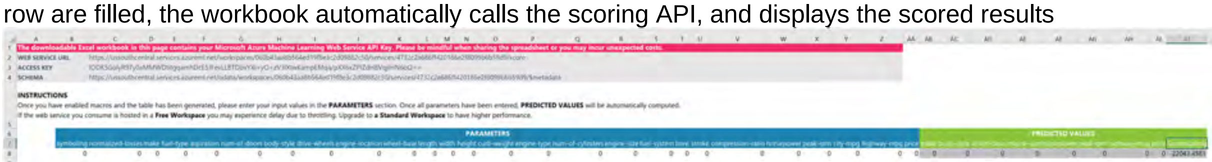
5) A Security Warning appears; click on the **Enable Editing** button.



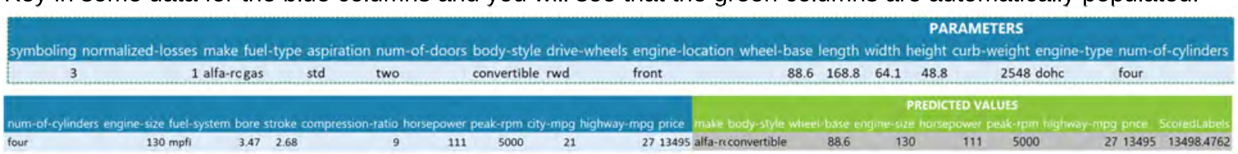
6) A Security Warning appears. Click on the **Enable Content** button to run macros on your spreadsheet.



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

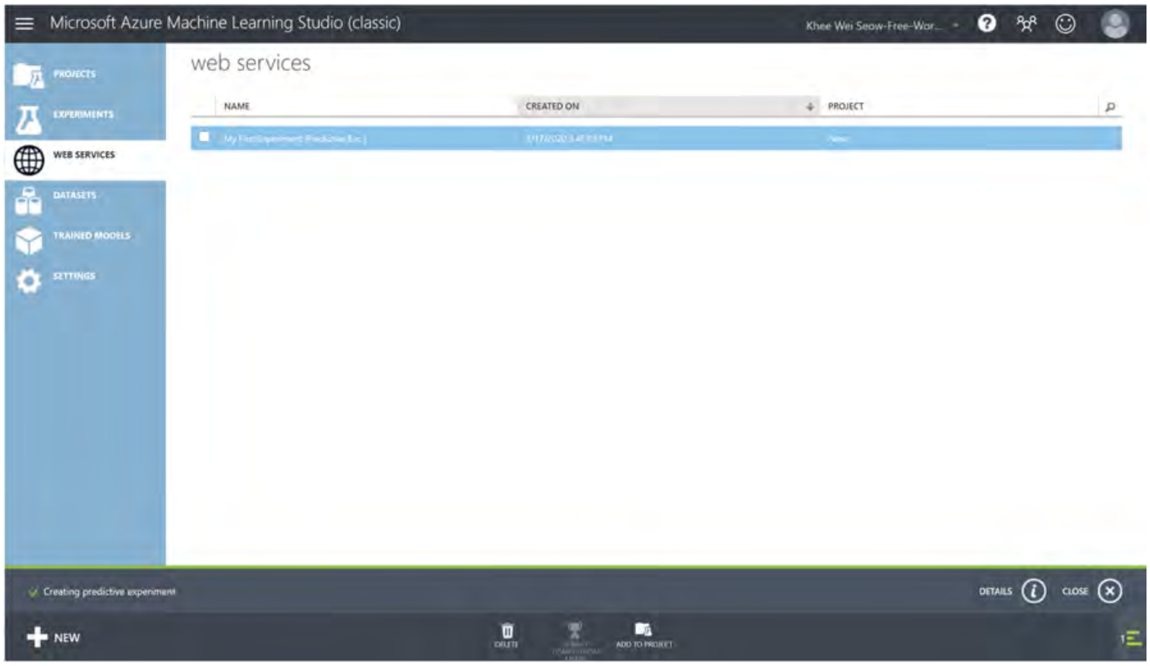


8) Key in some data for the blue columns and you will see that the green columns are automatically populated.

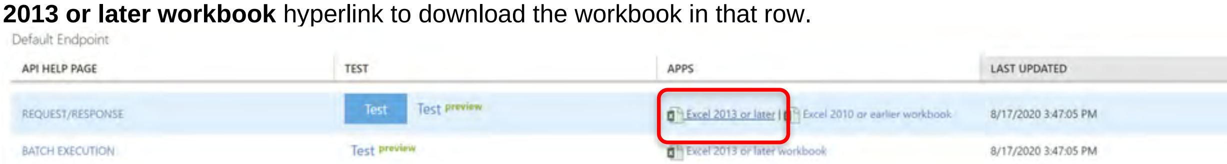


5) Make Prediction using Excel (after 2013)

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

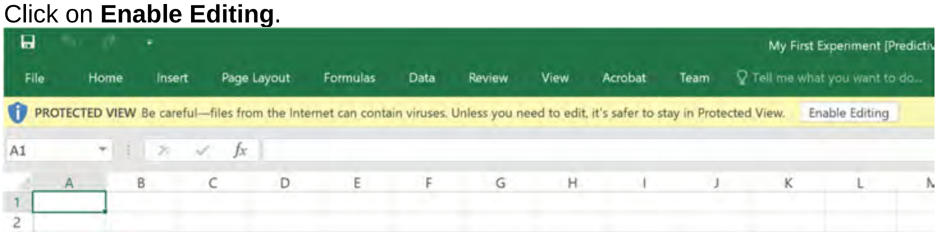


2) On the DASHBOARD tab for the web service is a row for the REQUEST/RESPONSE service. Click on **Excel 2013 or later workbook** hyperlink to download the workbook in that row.

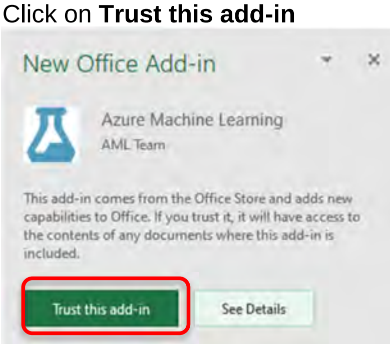


3) Open the sample Excel file, which contains the Excel add-in.

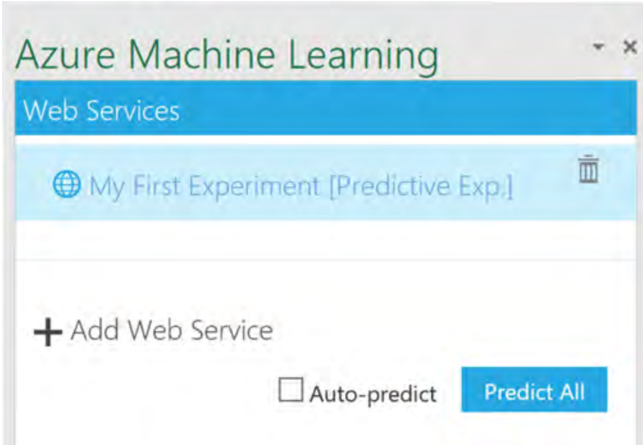
4) Click on **Enable Editing**.



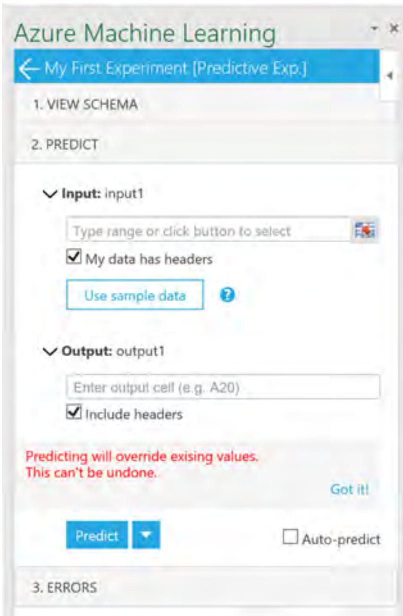
5) Click on **Trust this add-in**



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

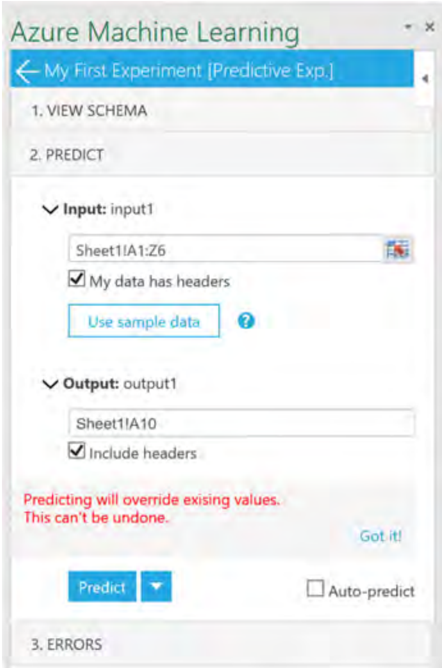


7) This takes you to the **Predict** section. For a blank workbook you can select a cell in Excel and click **Use sample** data.



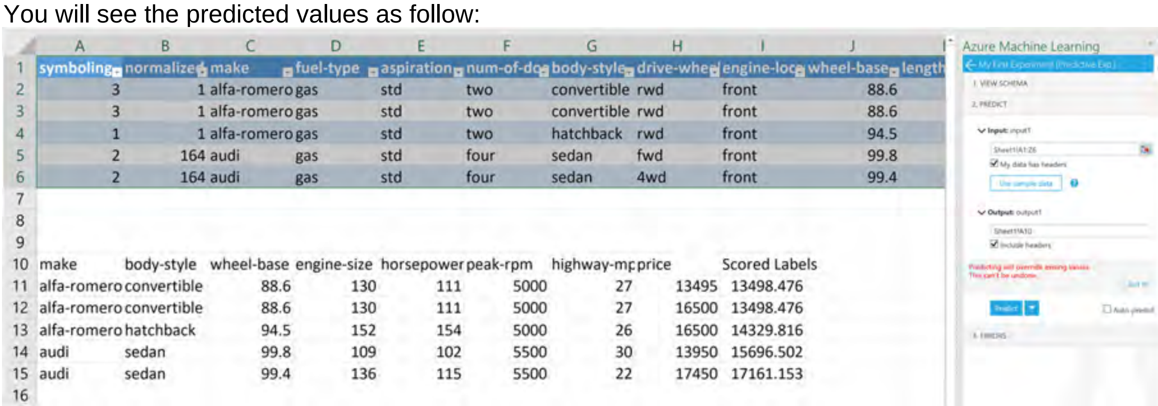
8) Select the data with headers and click the input data range icon. Make sure the "My data has headers" box is checked.

9) Under Output, enter the cell number where you want the output to be, for example "A10".



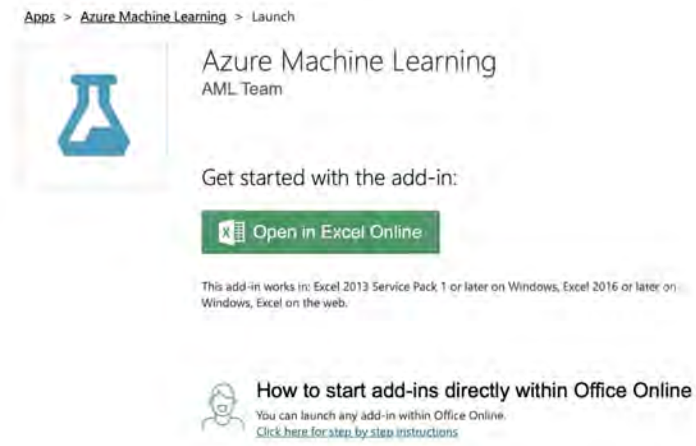
10) Click **Predict**. If you select the "auto-predict" checkbox any change on the selected areas (the ones specified as input) will trigger a request and an update of the output cells without the need for you to press the predict button.

11) You will see the predicted values as follow:

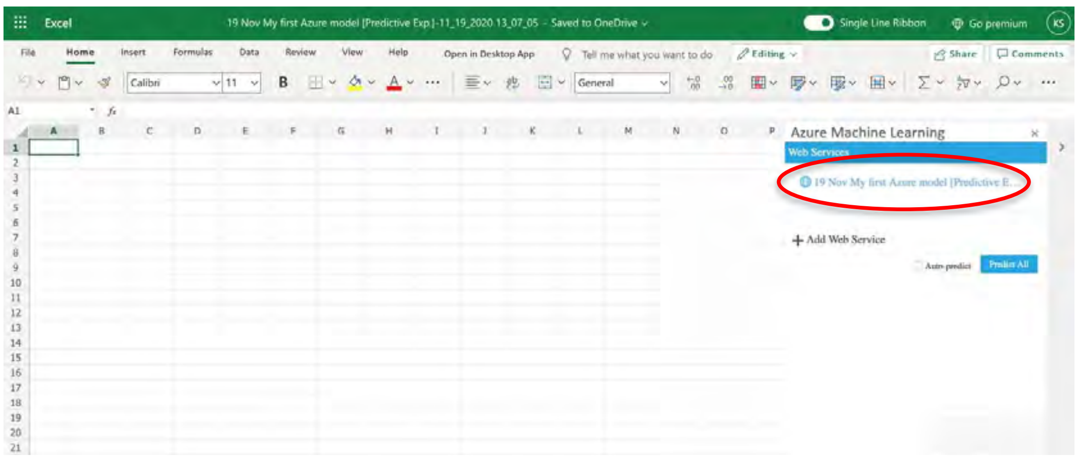


6) Make Prediction using Excel on the web (E.g. you are using MacOS)

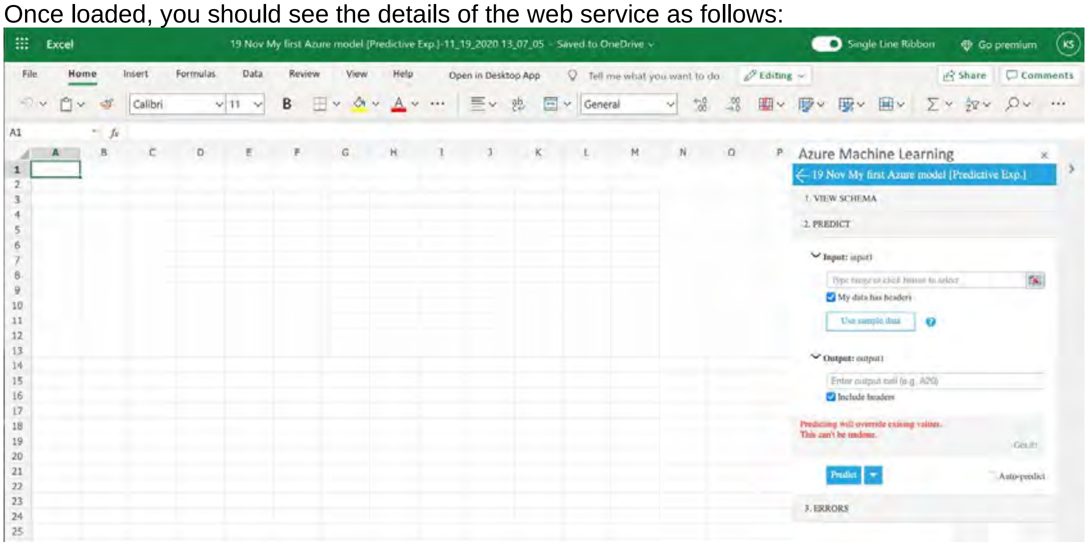
1) Since Microsoft Azure Machine Learning Excel add-in only works on windows and Excel on the web, if you are using MacOS, the workaround is to use Excel on the web instead. The easiest way is to copy the downloaded excel file in step (5) to your one drive (via OneDrive agent or Onedrive.com)
2) Open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



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



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



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

Activity 3 – Car Damage Assessment Classification

In this activity, we will learn:
- ❑ Save training time and create well-performing models with small datasets using transfer learning

Ref: A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning.
https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

1) The Problem

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

2) The Data

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

Unbalanced dataset

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

Classes

Each image belongs to one of the following classes:

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

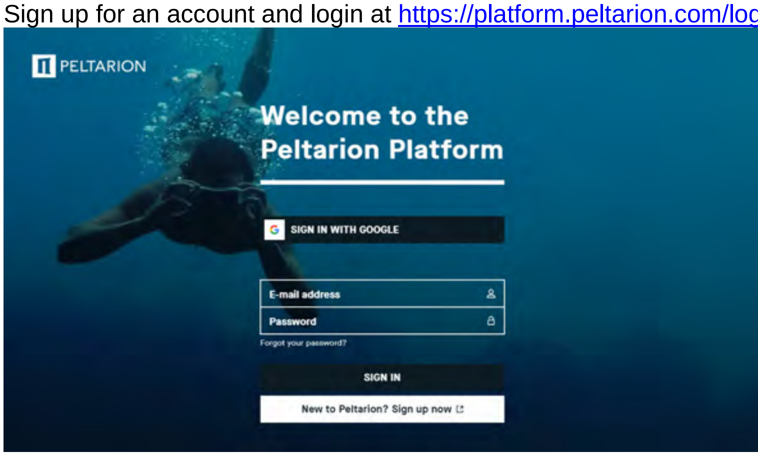
Below are sample images from the various classes in the dataset. Note that the unknown class contains images of cars that are in either a pristine or wrecked condition.

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

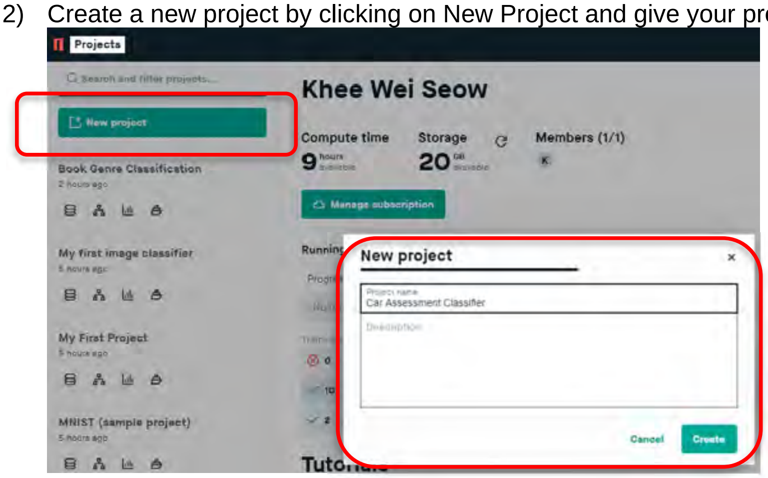
 Broken headlamp	 Broken tail lamp	 Glass shatter	 Door scratch
 Door dent	 Bumper dent	 Bumper scratch	 Unknown

3) Create new Project

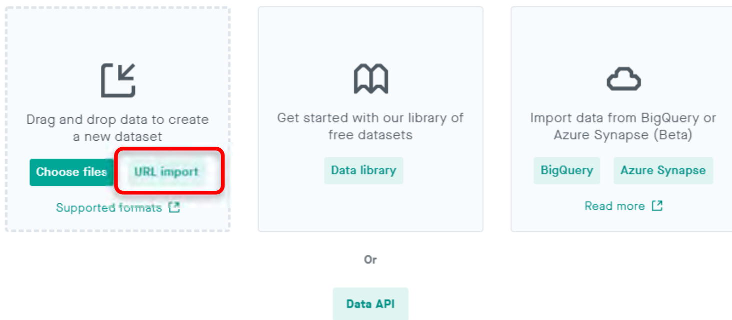
1) Sign up for an account and login at https://platform.peltarion.com/login



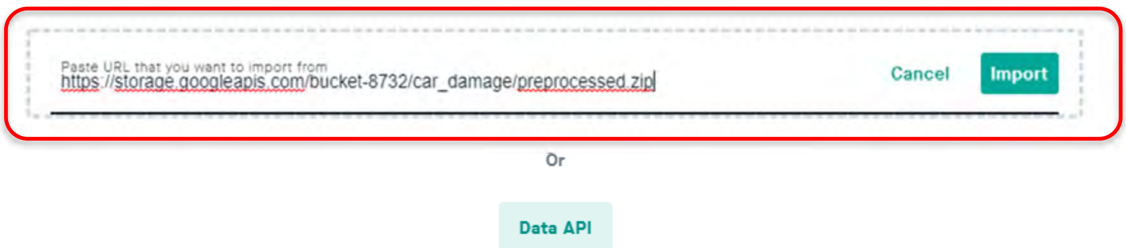
2) Create a new project by clicking on New Project and give your project a new. Click **Create** when done.



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

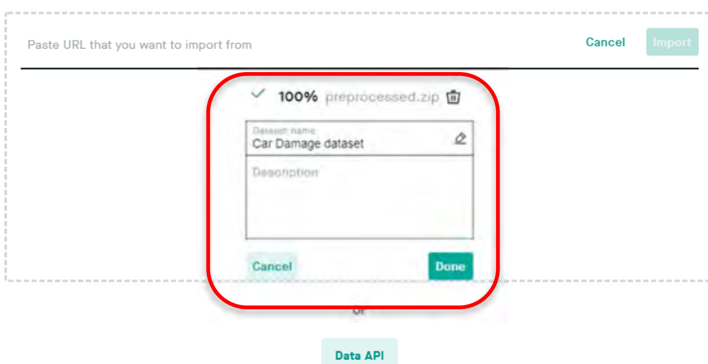


4) Copy the link and paste the link https://storage.googleapis.com/bucket-8732/car_damage/preprocessed.zip to the **Import data from URL** box. The zip includes the whole dataset.

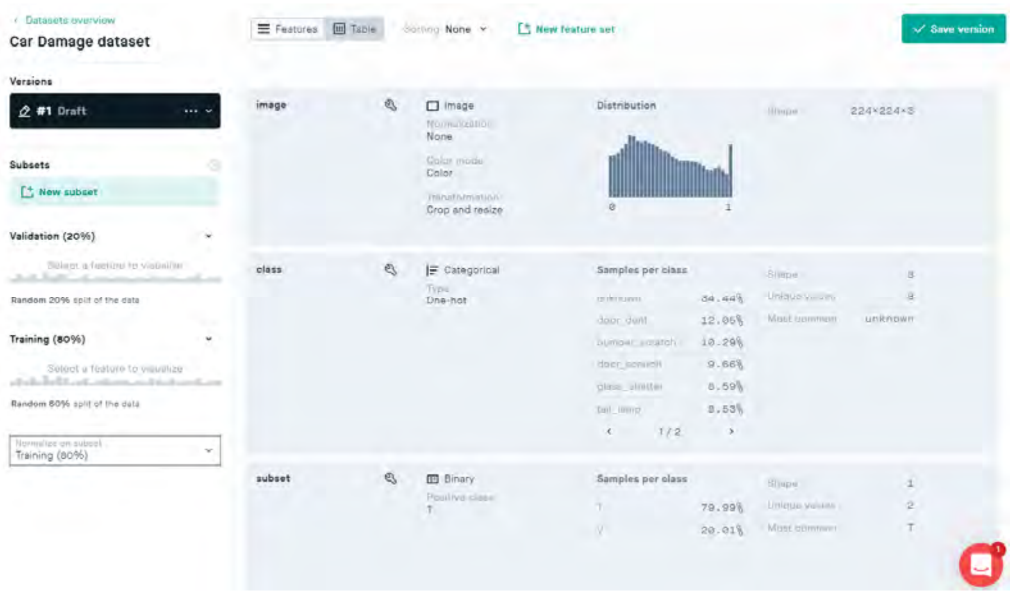


5) Click on **Import**.

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



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

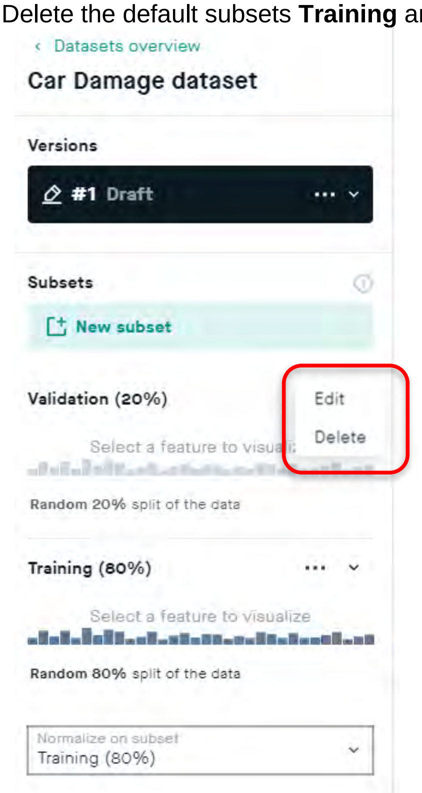


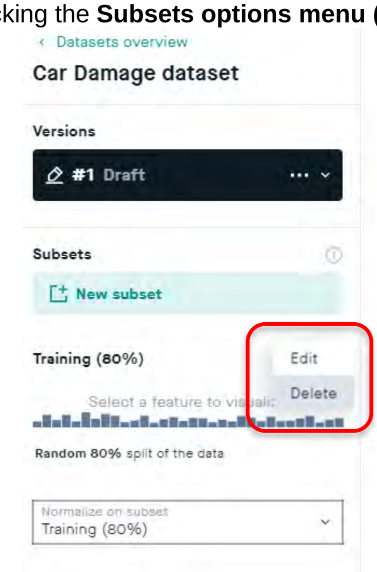
4) Create subsets of the car damage dataset

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

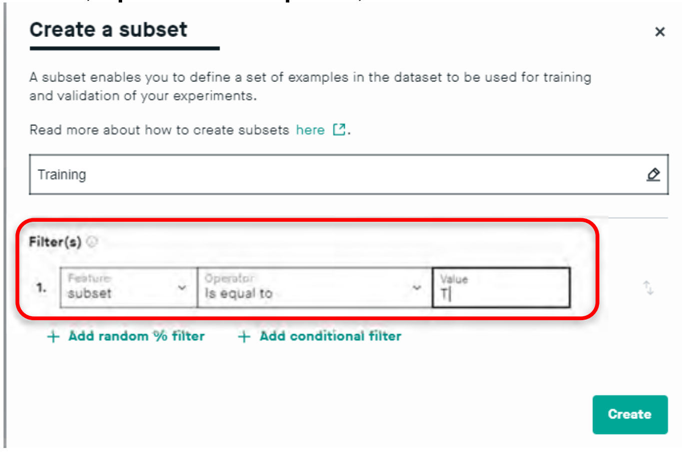
Even though it is possible to use the default subsets created by the platform when you upload the data, it is more advantageous to create a conditional split based on the subset column. For this dataset, there is no separate labelled test subset, in case you want to analyse the performance of the deployed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

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

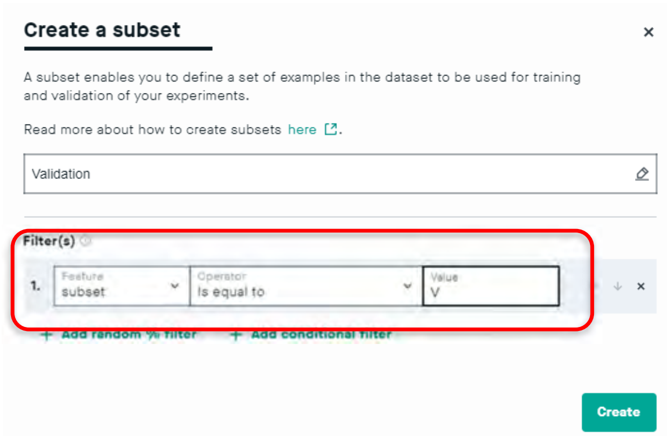




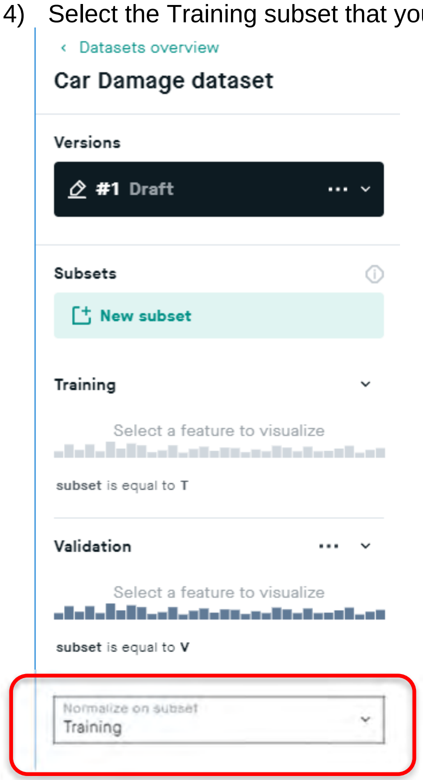
2) Click **New subset** and name the training subset **Training**. Then click **Add conditional filter** and set **Feature** to **subset**, **Operator** to **is equal to**, and Value to **T**. The details are shown below. Click **Create** to create this subset.



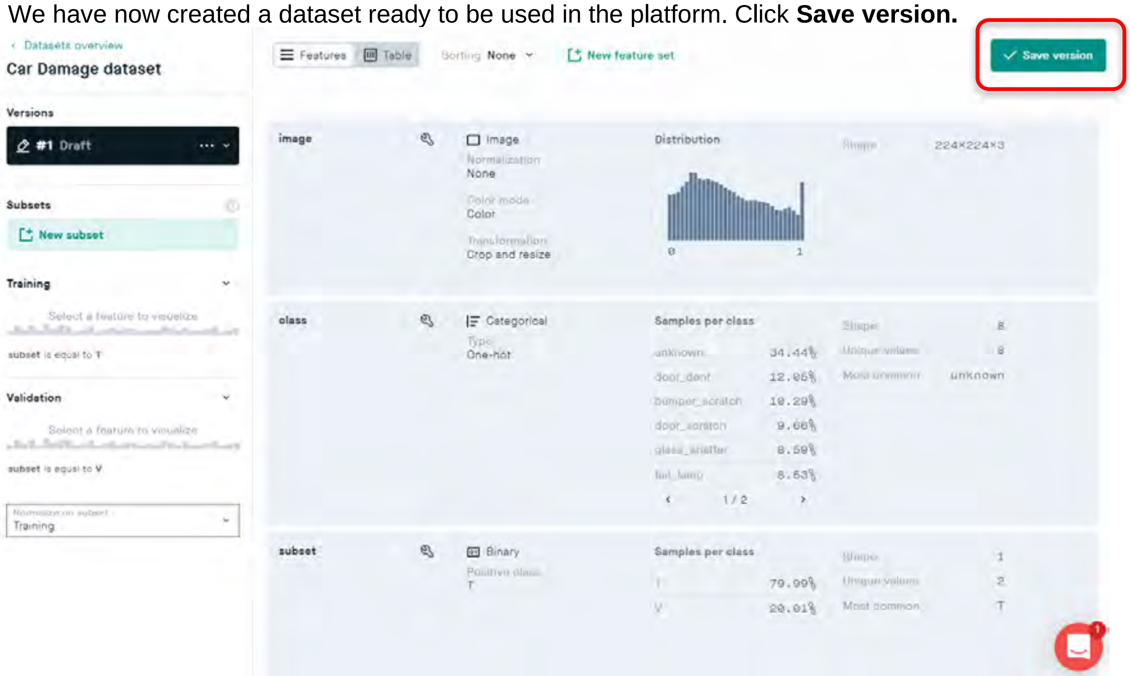
3) Repeat the procedure for a new **Validation** subset and set **Feature** to **subset**, **Operator** to **is equal to**, and Value to **V**. The details are shown below.



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



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

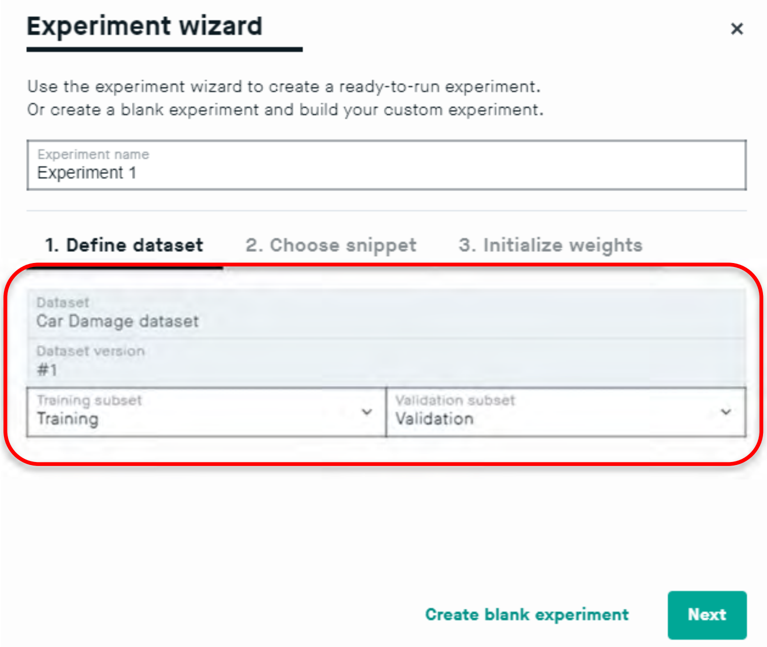


5) Create a new experiment

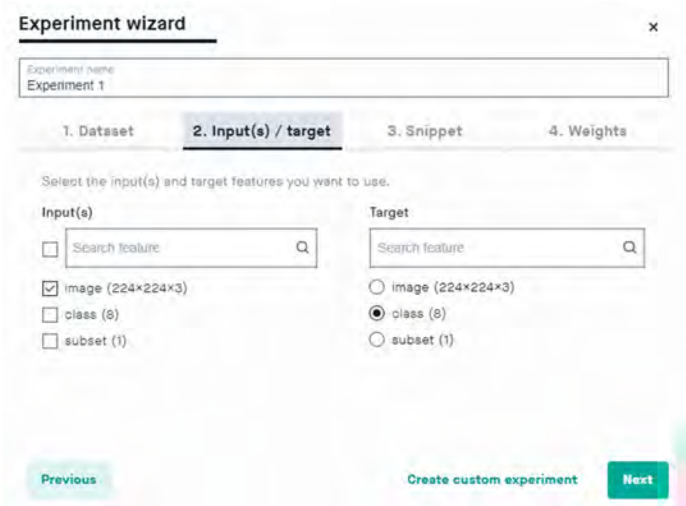
1) Click on **Use in new experiment** to create a new experiment using this dataset.

2) Name the experiment in the **Experiment wizard**.

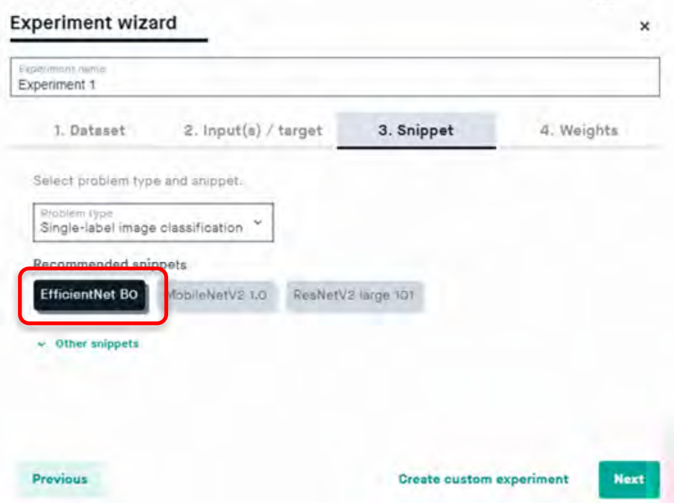
3) Make sure that the **Car damage dataset** is selected in the **Define dataset** tab. Click **Next** to continue.



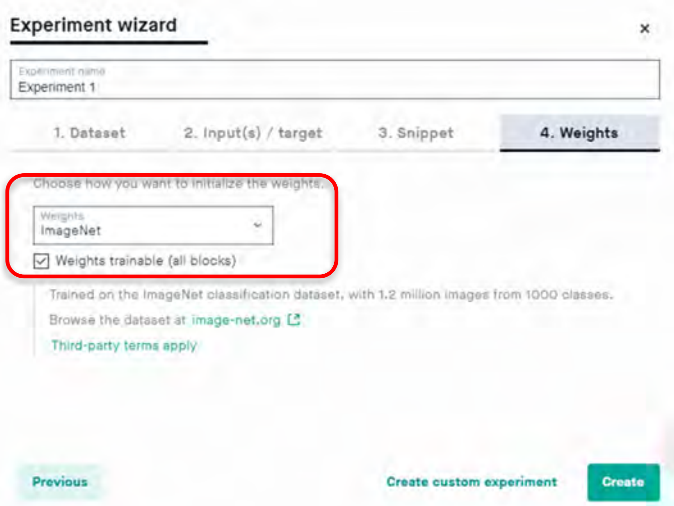
4) In the Input(s) / target tab, check that the following inputs are pre-populated. Click **Next** to move to the next step.



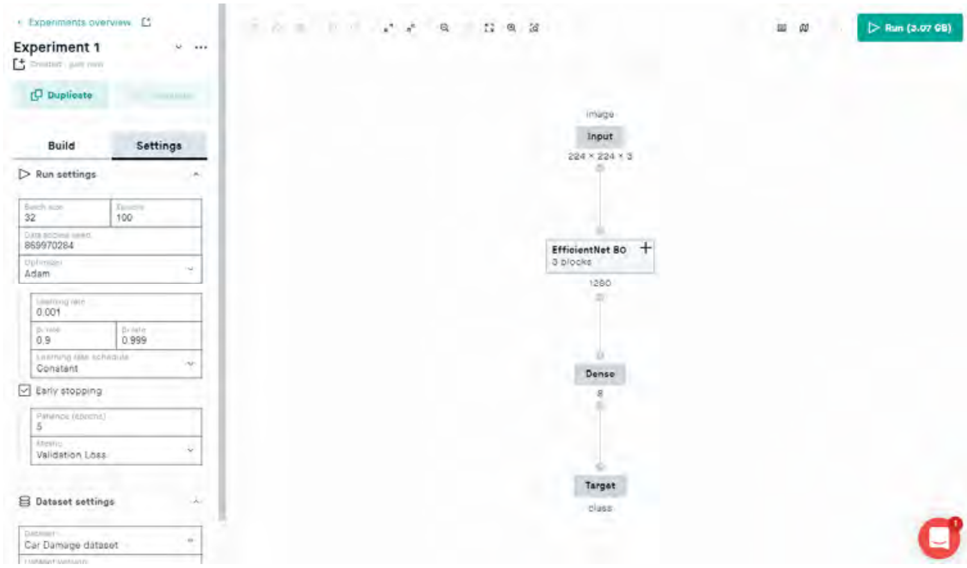
5) On the **Snippet** tab and select the **EfficientNet B0** snippet. Click **Next** to continue.



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



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

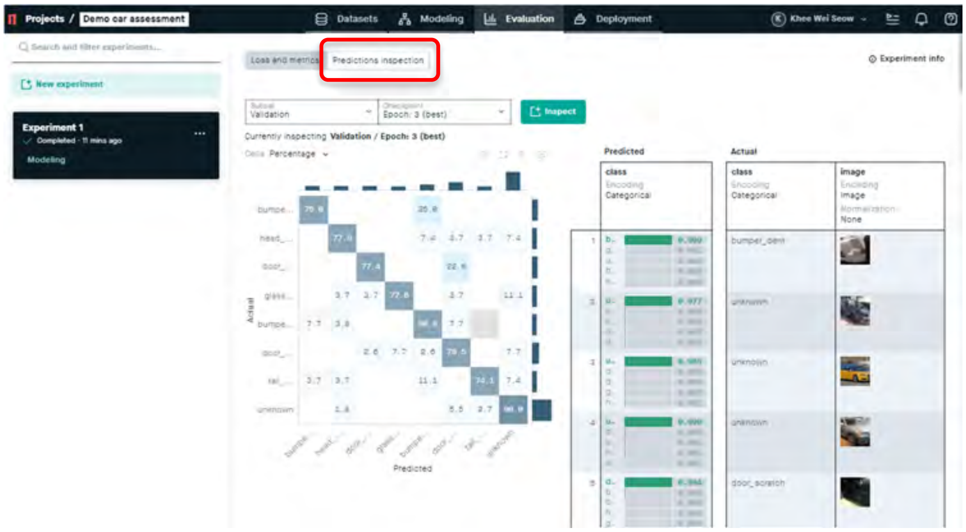


6) Run the experiment

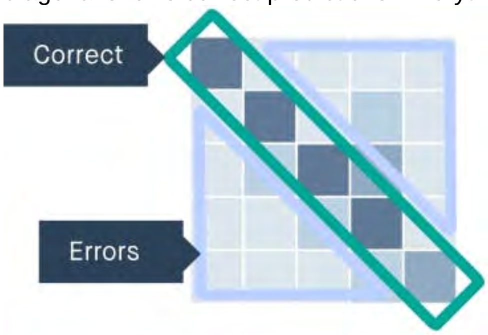
1) Click the **Settings** tab and change the **Learning rate** to 0.0005 and **Epoch** to 5. Click **Run** to start the training. The training may take a long time depending on complexity and the number of epochs. You can grab a coffee or tea at this point in time. For this case, we should take around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

7) Analyse the experiment

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

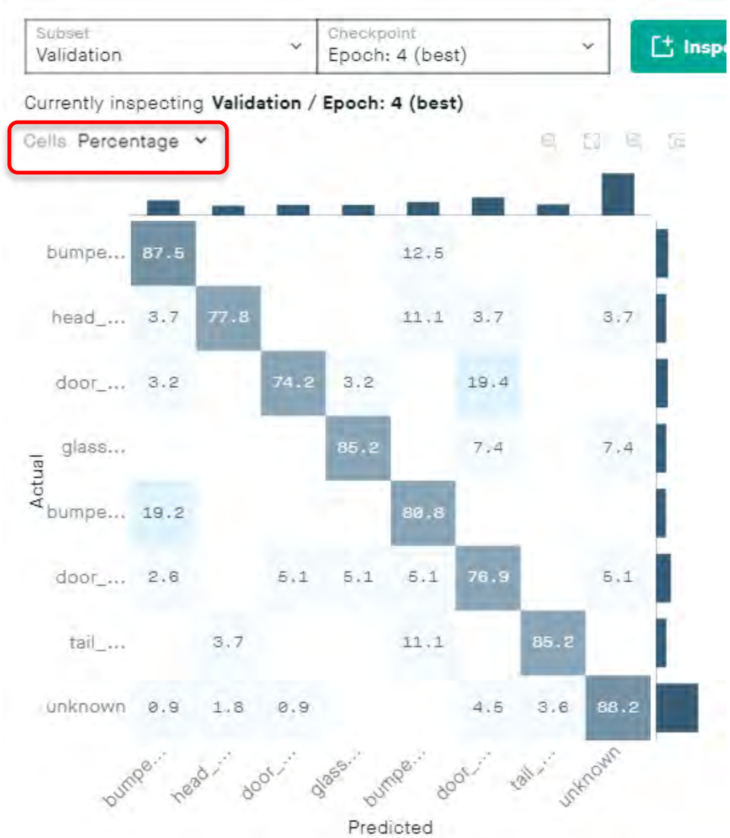


2) Since the model solves a classification problem, a confusion matrix is displayed. The top-left to bottom-right diagonal shows correct predictions. Everything outside this diagonal are errors.



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

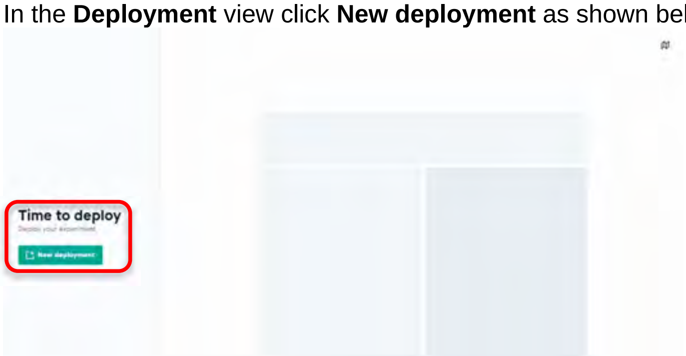
4) Click the dropdown next to **Cells** and select **Percentage**. The normalized values that are now displayed correspond to the recall for each class.



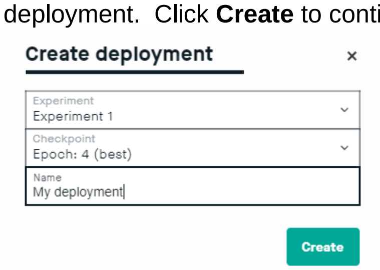
The recall values clearly indicate that the model has learned the features in the images.
Ref: https://peltarion.com/knowledge-center/documentation/glossary#R

8) Deploy the trained model

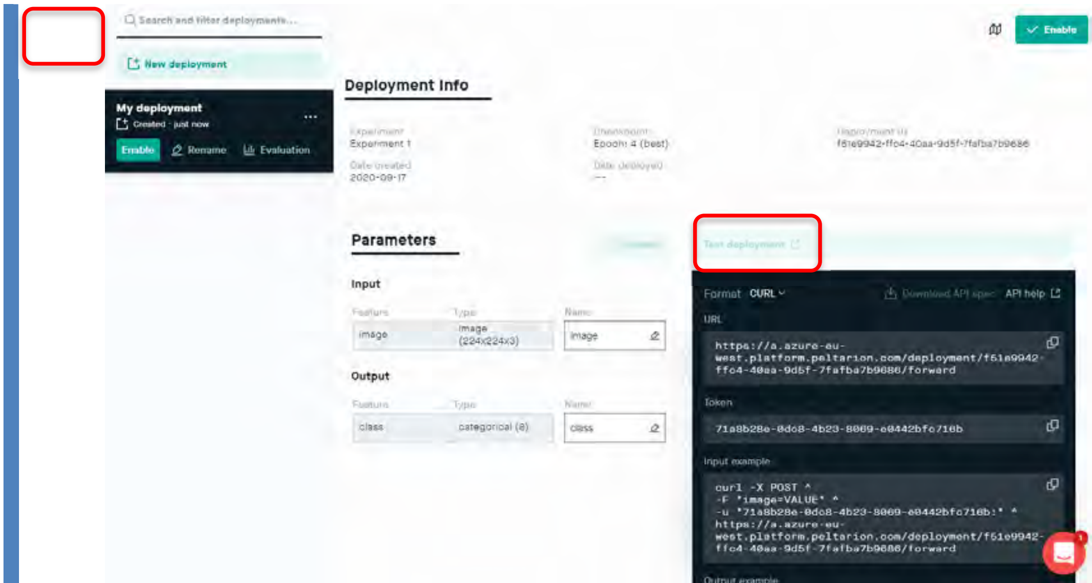
1) In the **Deployment** view click **New deployment** as shown below.



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



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



4) Click the **Enable** switch to deploy the experiment.

9) Test the classifier in a browser

1) 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.

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

10) Next Steps

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

Activity 4 – Creating a Sentiment Analyser

In this activity, we will learn:

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

Ref:

1) Deploy an operational AI model (https://peltarion.com/knowledge-center/documentation/tutorials/deploy-an-operational-ai-model)
2) Embeddings - If you want it could be a good idea to read about word embeddings, which is an important concept in NLP (Natural Language Processing). For an introduction and overview of different types of word embeddings, check out the links below:

3) Get Busy with Word Embeddings — An Introduction (https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction)
4) Introduction to Embedding in Natural Language Processing (https://www.datascience.com/blog/embedding-in-natural-language-processing)

Text embedding block (https://peltarion.com/knowledge

1) The Problem

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

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

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

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

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

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

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

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

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

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

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

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

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

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

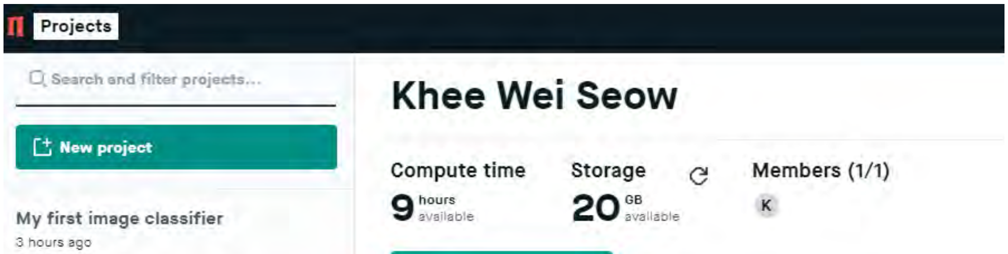
The review text may include commas, which will be interpreted as a field delimiter on the platform. To escape these commas, the text is surrounded by double-quotes.

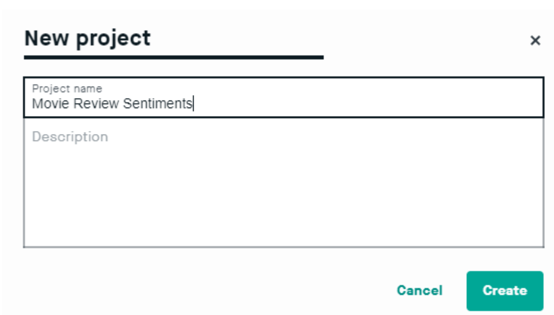
The processed dataset only includes the training data.

3) Create a new project

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

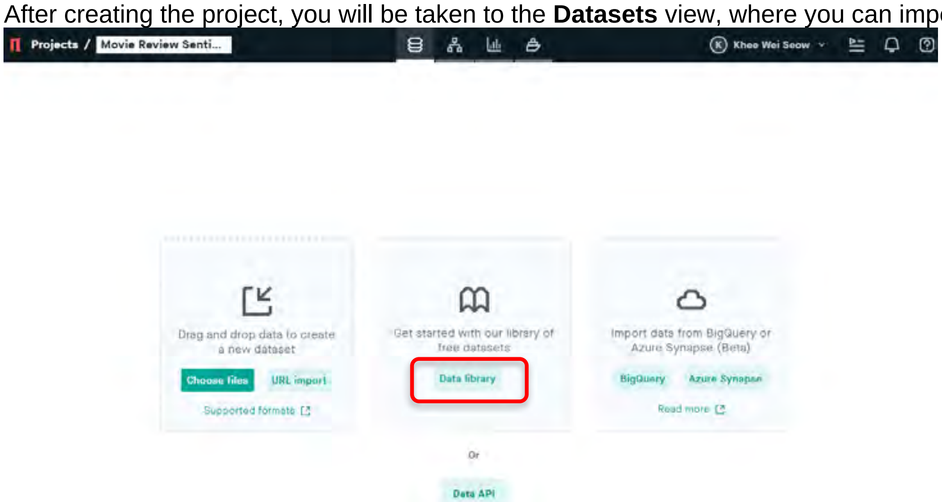
2) Create a new project by clicking on New Experiment and give your project a name. In this case, we can name it Movie Review Sentiments.



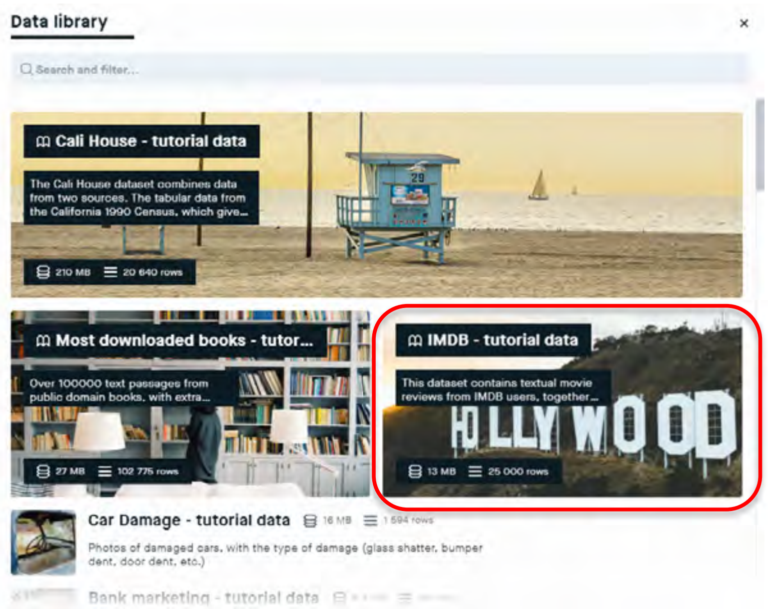


4) Add a new dataset

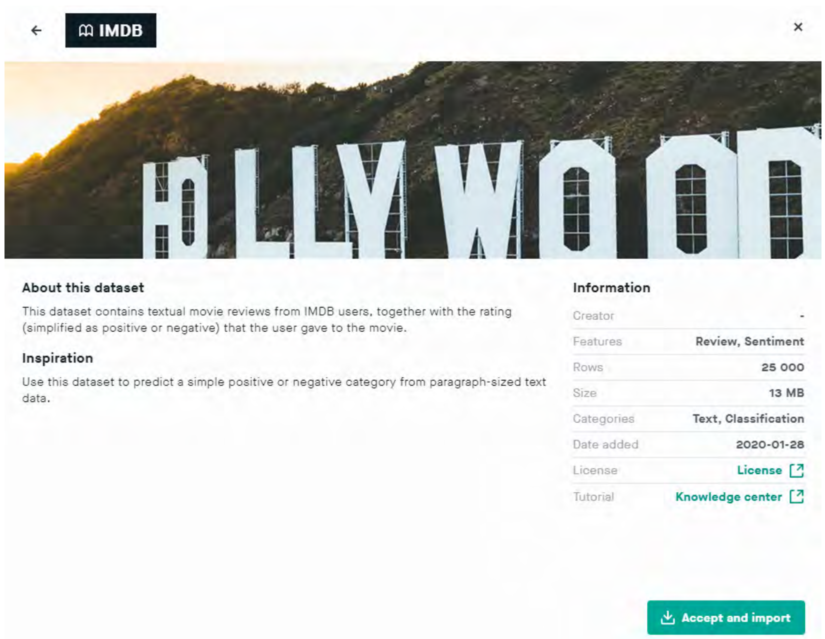
1) After creating the project, you will be taken to the **Datasets** view, where you can import data.



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

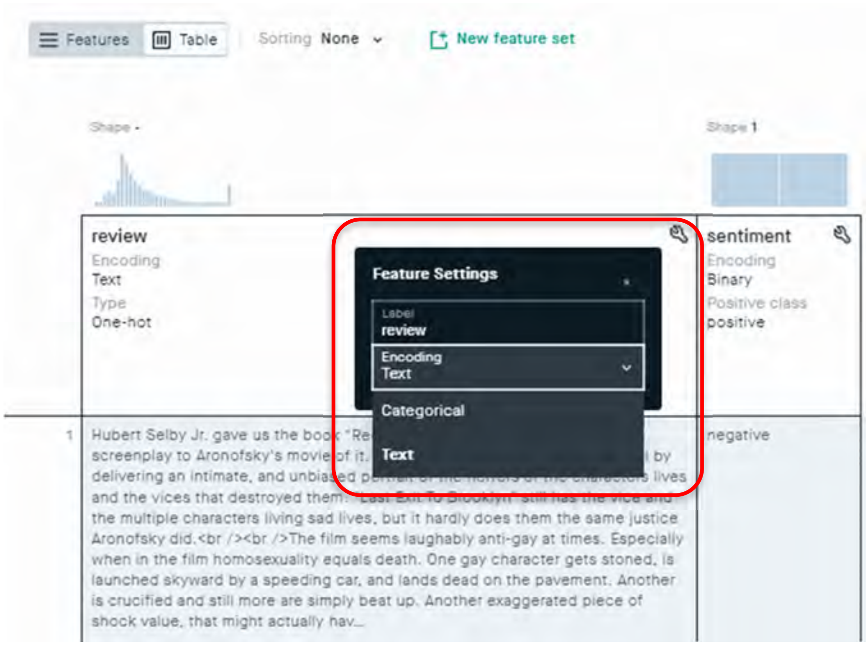


3) If you agree with the license, click **Accept and import**. This will import the dataset in your project, and you will be taken to the dataset's details where you can edit features and subsets.



5) Text Encoding

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

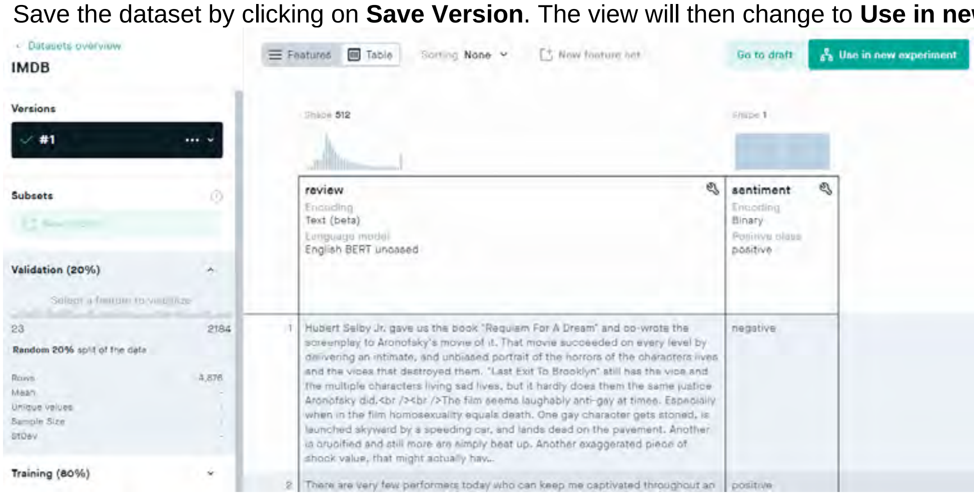


Note:

Subsets of the dataset

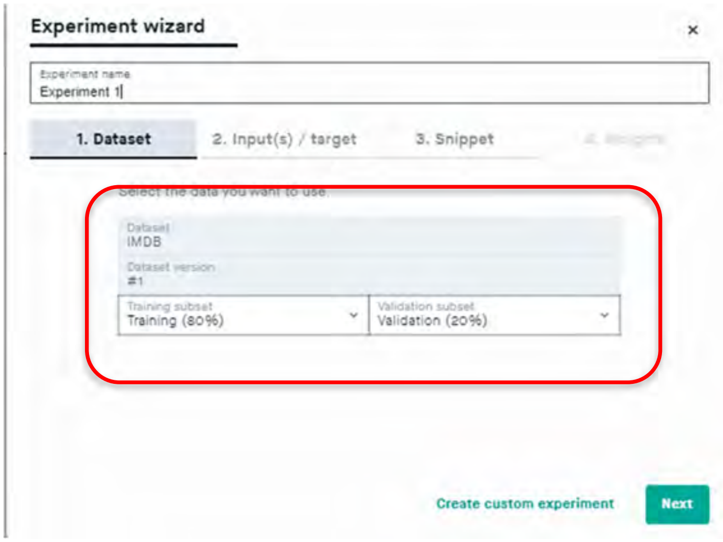
On the left, you will see the subsets. All samples in the dataset are by default split into 20% validation and 80% training subsets. Keep these default values in this project.

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

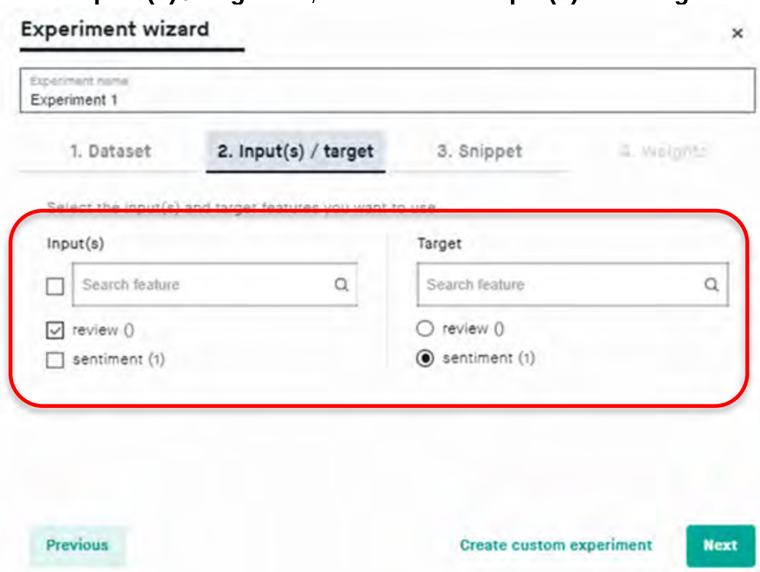


6) Design a text classification model with the BERT model

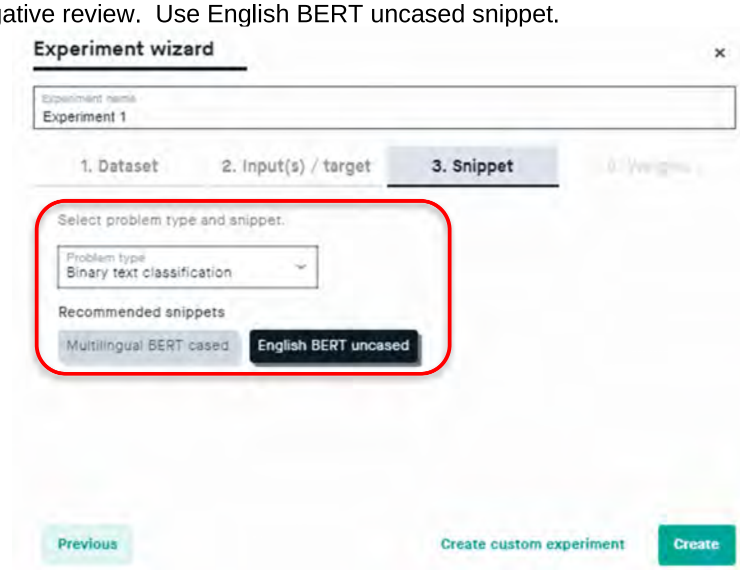
1) Make sure that the **IMDB** dataset is selected in the **Experiment wizard**. Click **Next**.



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



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

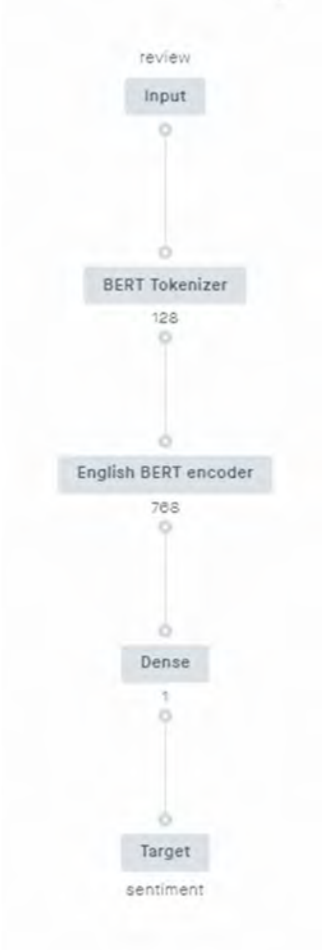


The BERT English uncased snippet includes the whole BERT network. The BERT block implements the base version of the BERT network. It is composed of 12 encoding layers from a Transformer network, each layer having 12 attention heads. The total number of parameters is 110 million. The snippet allows you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

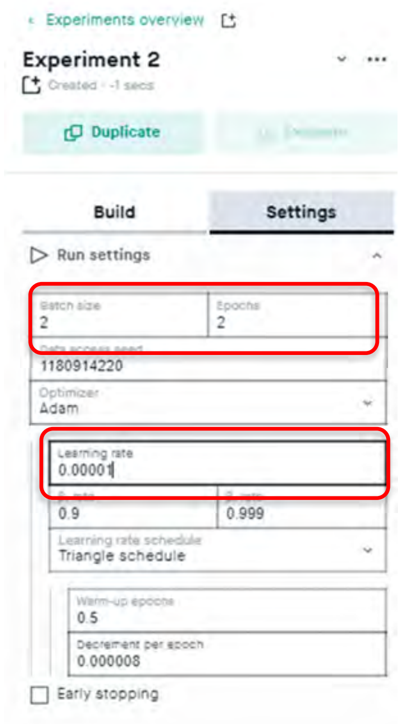
The BERT snippet includes:

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

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



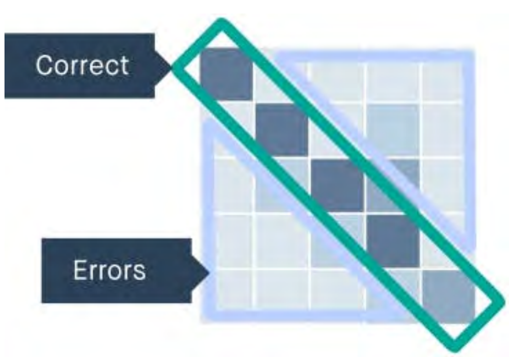
6) Click **Run** to start training the model. ***Note: This experiment will take 2 hours to complete***.

7) Evaluation

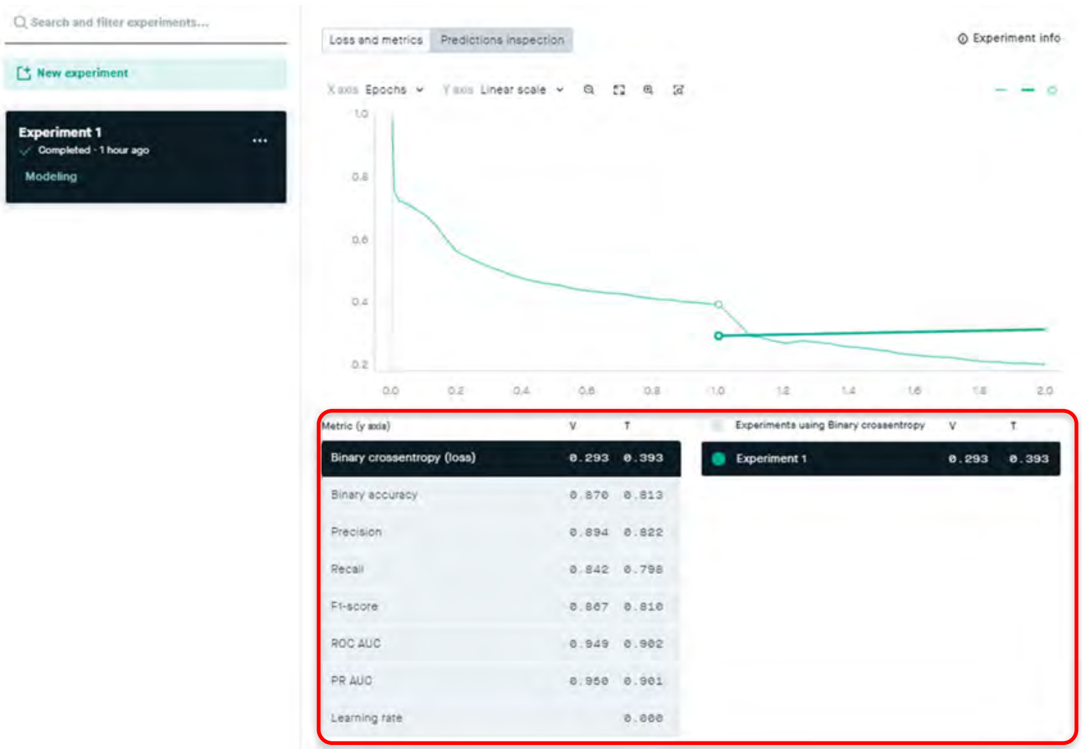
1) Navigate to the **Evaluation** view and watch the model train. The training will take quite a long time since BERT is a very large and complex model.

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

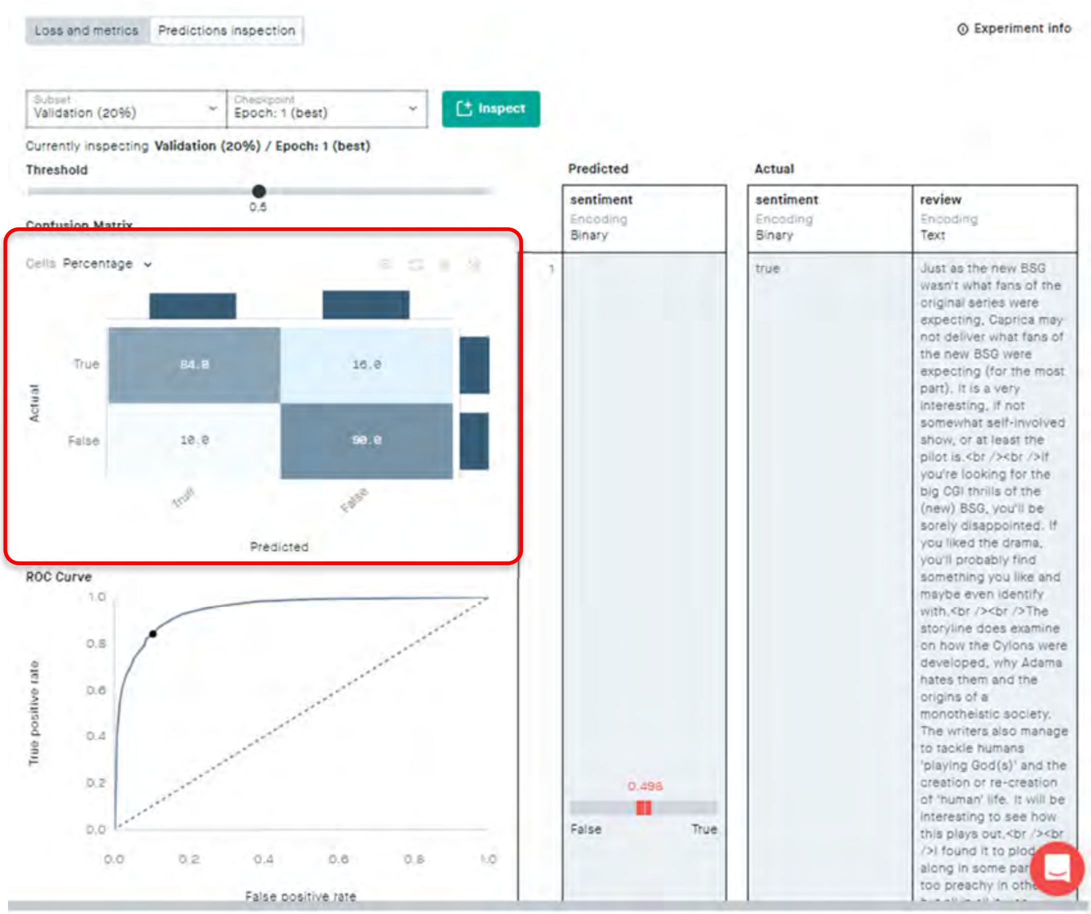
Confusion matrix
Since the model solves a classification problem, a confusion matrix is displayed. The top-left to bottom-right diagonal shows correct predictions. Everything outside this diagonal are errors.



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

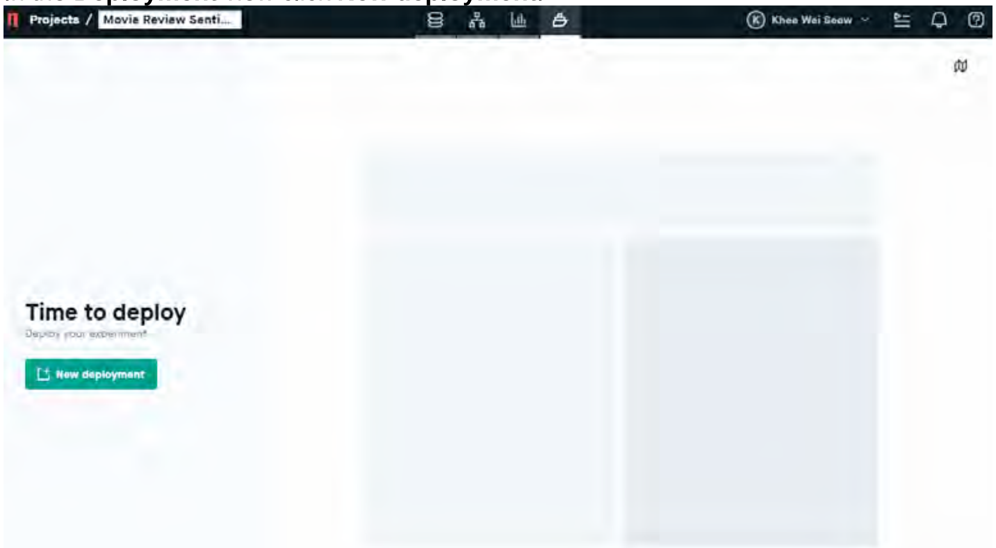


2) Navigate to the **Predictions Inspection** view. It will take awhile to create the confusion matrix. When all the examples where processed, you should see the following:

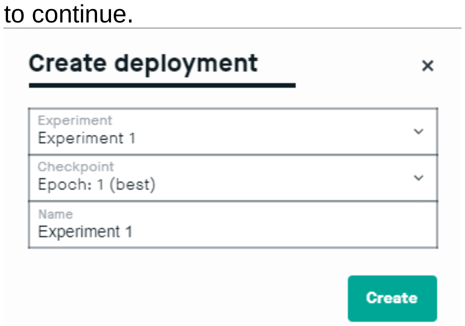


8) Create new deployment

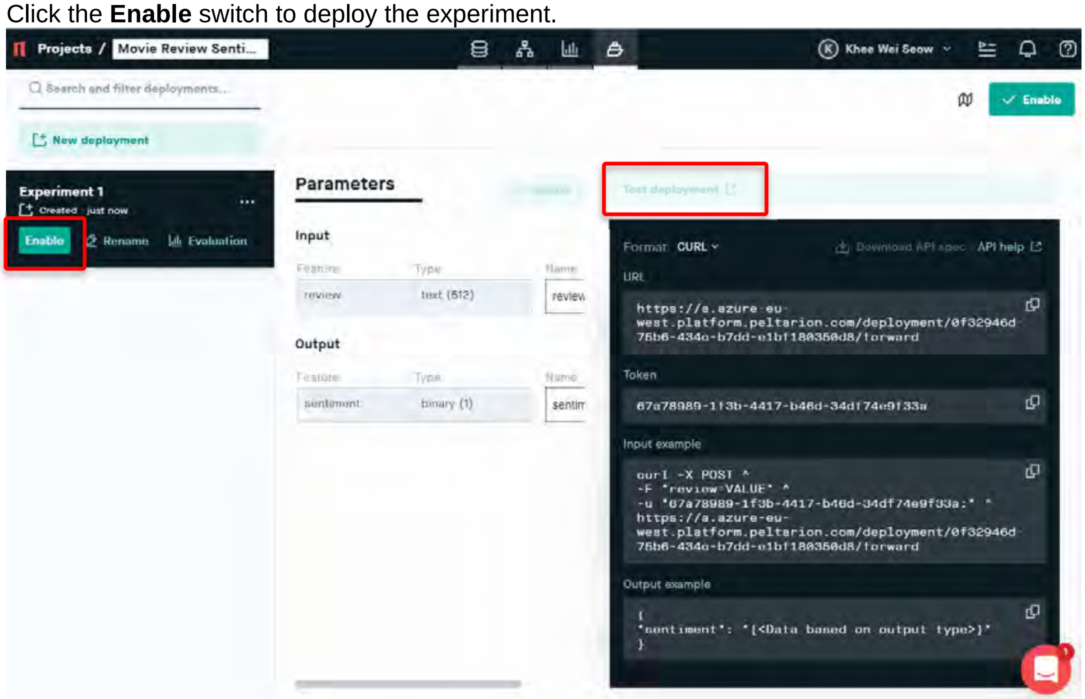
1) In the **Deployment** view click **New deployment**.



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

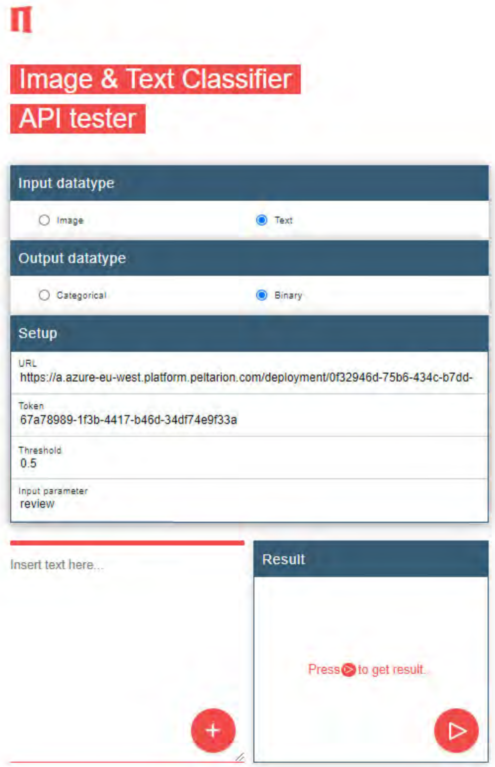


3) Click the **Enable** switch to deploy the experiment.



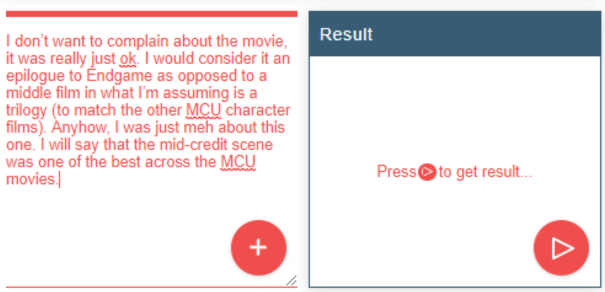
9) Test the text classifier in a browser

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



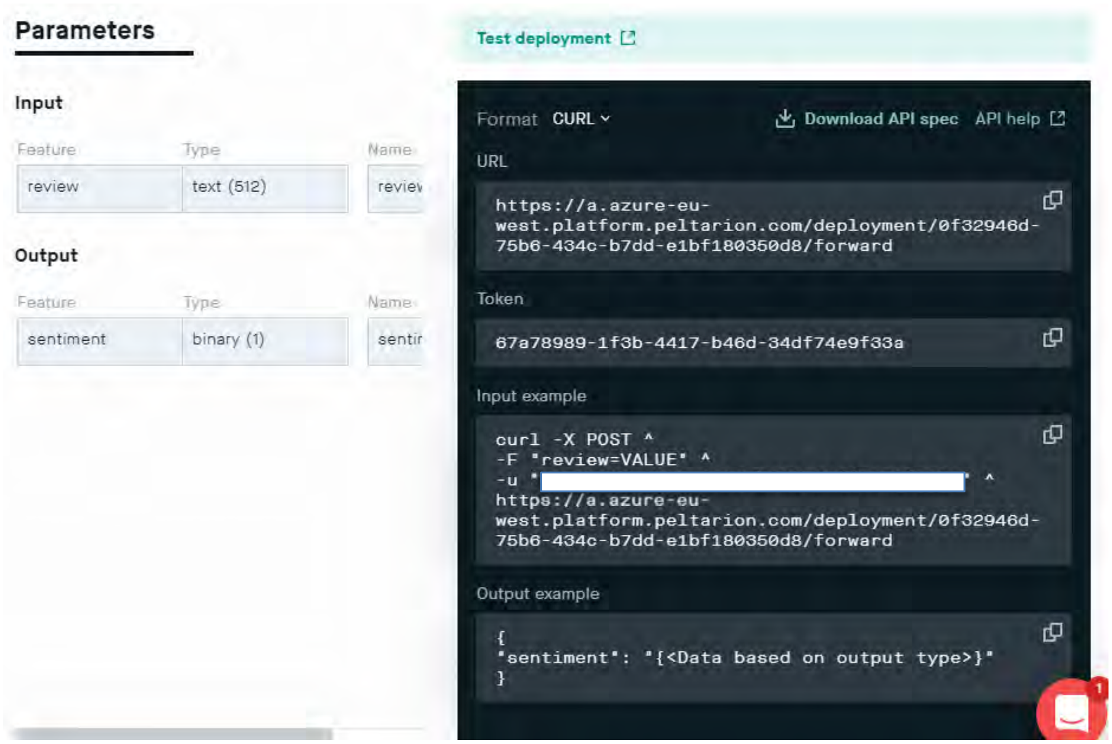
2) Now, write your own review, copy the example below or simply copy a recent review from, e.g., IMDB.

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.



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 "[redacted]:" ^
More? https://a.azure-eu-west.platform.peltarion.com/deployment/0f32946d-75b6-434c-b7dd-e1bf180350d8/forward
{"sentiment":0.95231044}
C:\Users\seow_khee_wei>
```

10) Next Step

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

Activity 5 – [Bonus] Book Genre Classifier

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

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

1) The Problem

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

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

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

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

2) Dataset - CMU book summary dataset

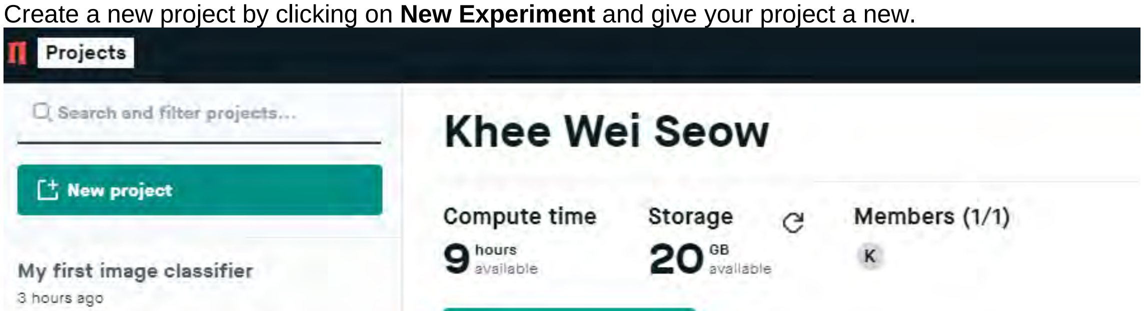
The data comes from the CMU Book Summary Dataset (http://www.cs.cmu.edu/~dbamman/booksummaries.html) , a dataset of over 16 000 book summaries. For this activity, we wanted a dataset with science fiction book summaries, so we chose to preprocess the data to our task, so it contains book summaries along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

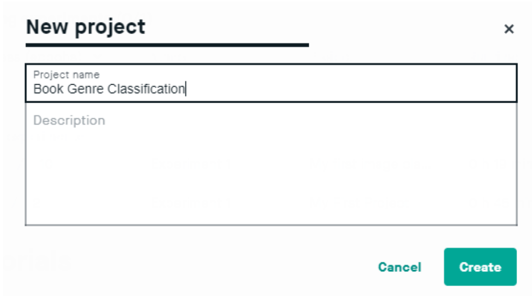
The dataset is intended to serve as a benchmark for sentiment classification. The overall distribution of labels is balanced, i.e., there are approximately 2 500 science fiction and 2 500 non-science fiction book summaries. Each summary is stored in a column, with a science fiction classification of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

3) Create a new Project

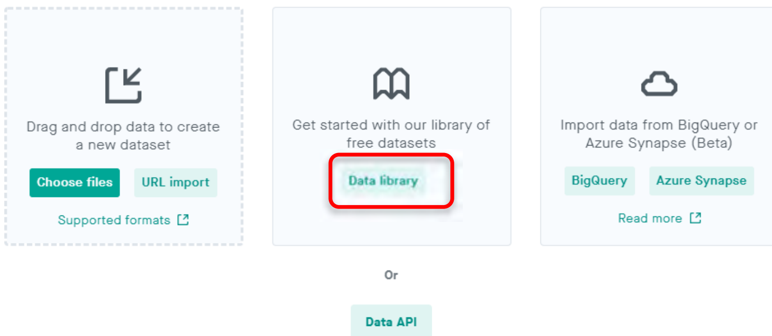
1) Sign up for an account and login at https://platform.peltarion.com/login

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

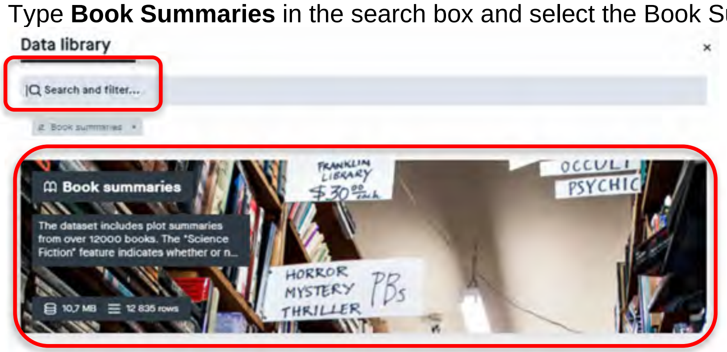




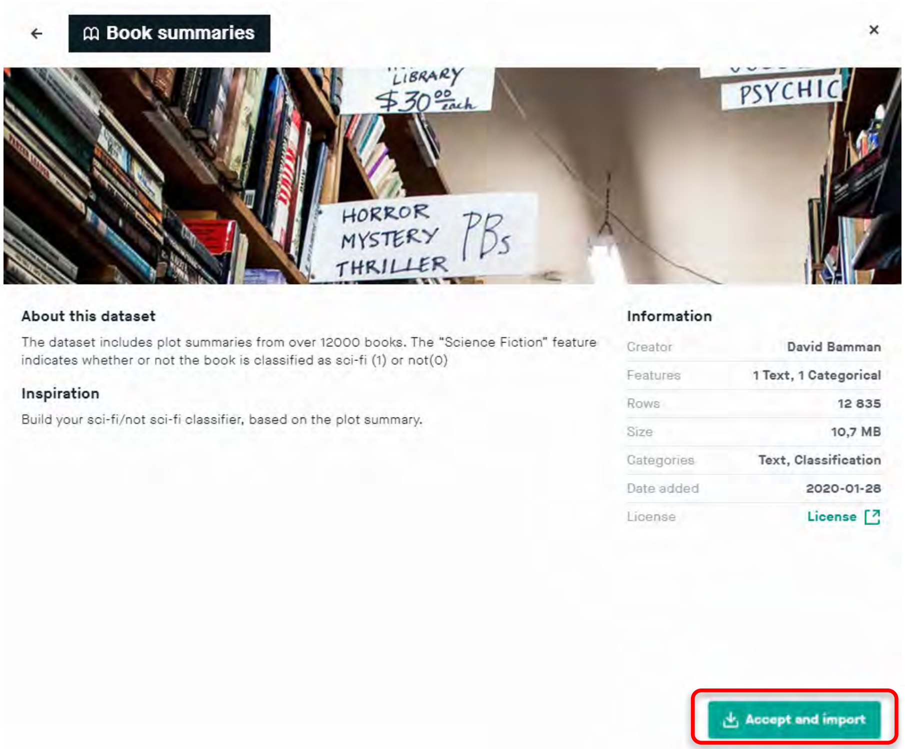
3) Navigate to the Datasets if you are not automatically brought there. Click on **Data Library**.



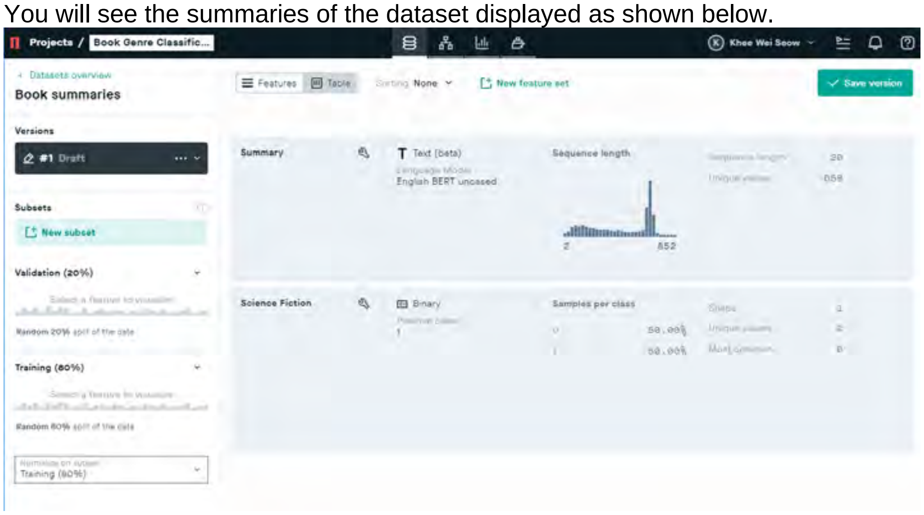
4) Type **Book Summaries** in the search box and select the Book Summaries dataset



5) Click on **Accept and import**



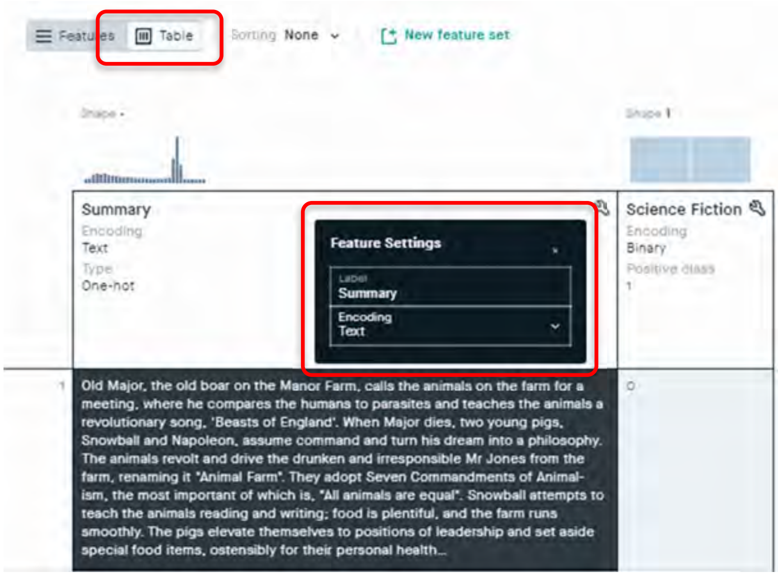
6) You will see the summaries of the dataset displayed as shown below.



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

4) Text encoding

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

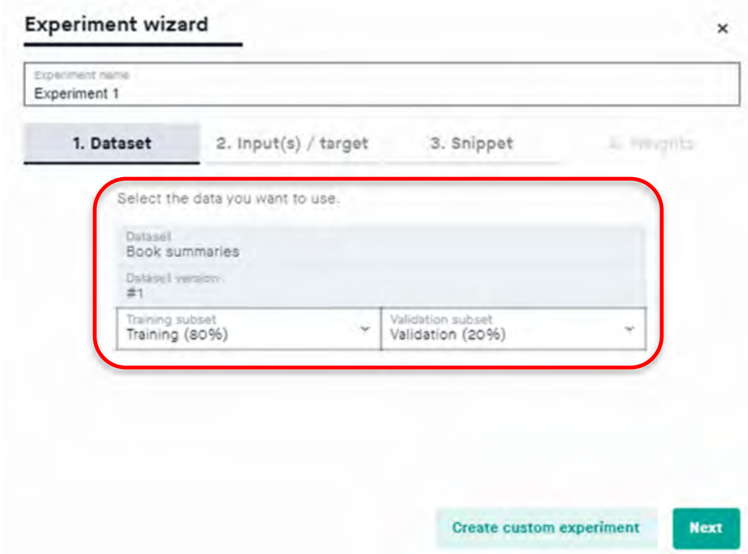


Note:
Subsets of the dataset
On the left, you will see the subsets. All samples in the dataset are by default split into 20% validation and 80% training subsets. Keep these default values in this project.

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

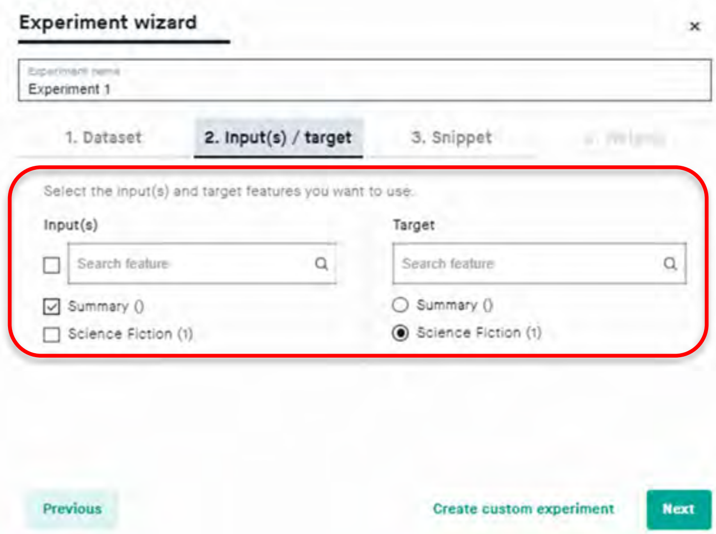
5) Design a text classification model

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



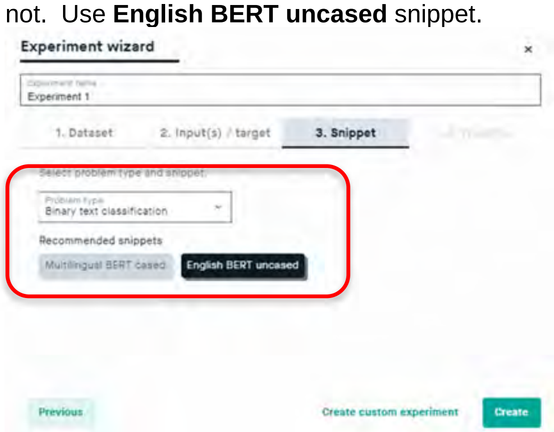
2) Click **Next** to continue.

3) Check that the Inputs and Target are set as follows:



Click **Next** to continue.

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



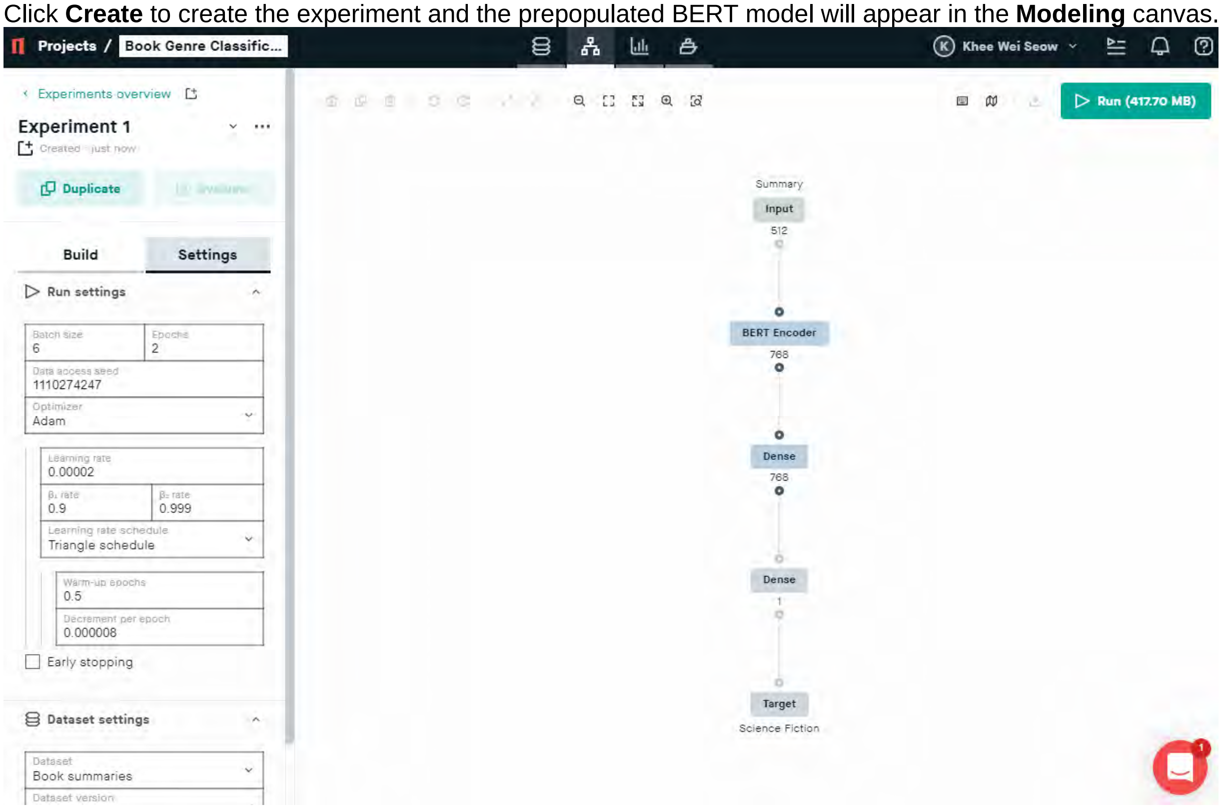
The BERT English uncased snippet includes the whole BERT network. The BERT block implements the base version of the BERT network. It is composed of 12 encoding layers from a Transformer network, each layer having

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:

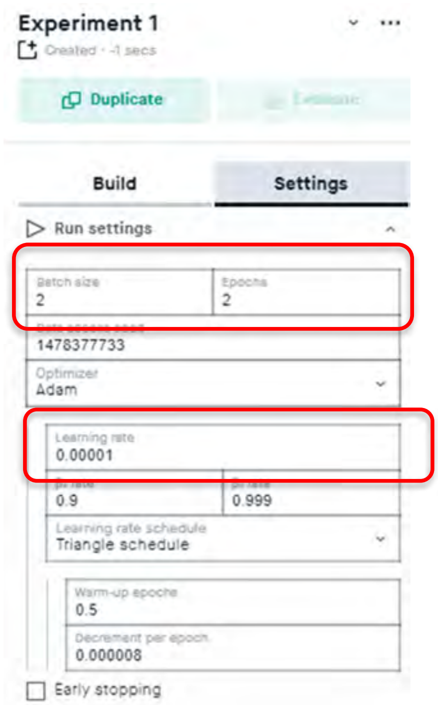
- An Input block.
- A BERT Encoder block with pre-trained weights which gives BERT a general understanding of English. The BERT encoder blocklooks at the input sequence as a whole, producing an output thatcontains an understanding of the sentence. The block outputs a singlevector of 768 size.
- A label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

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



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

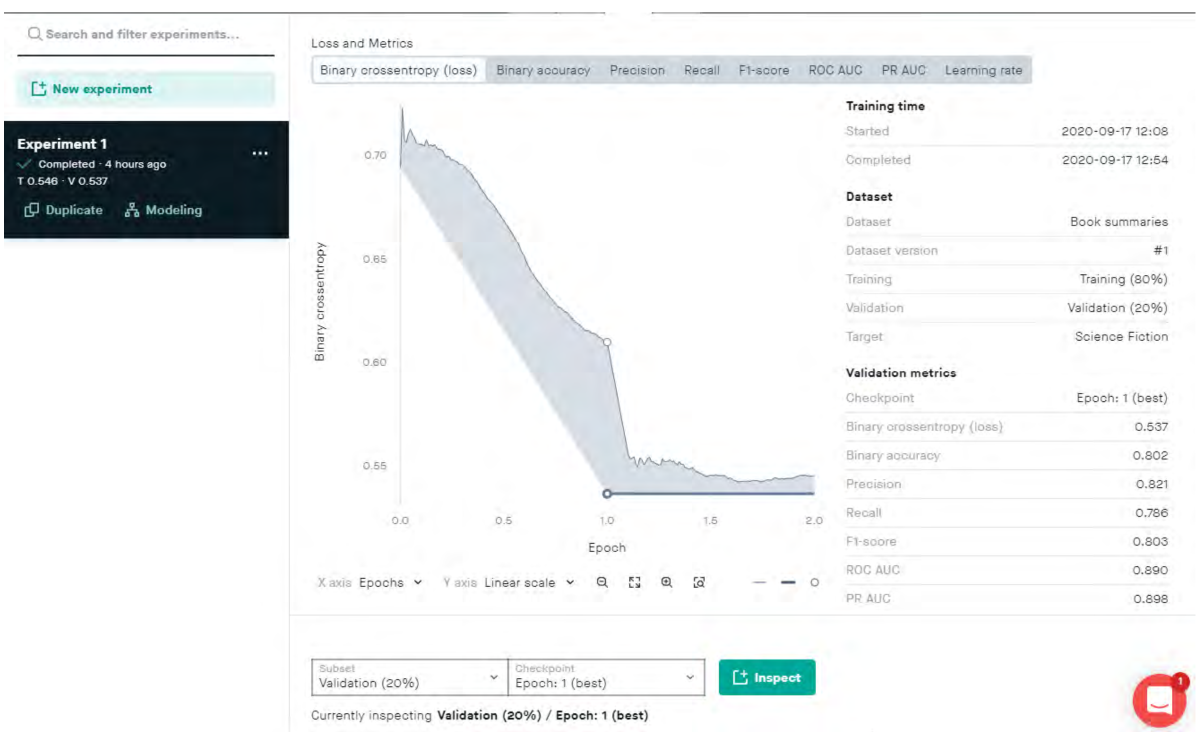
- **Batch Size** is 2. If you set a larger batch size you may run out of memory.
- **Epochs** is 2. Training takes a long time, so don't train for too long the first time when you check if your model is good.
- **Learning rate** is 0.00001 (4 zeros). To avoid catastrophic forgetting.



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

6) Evaluation

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

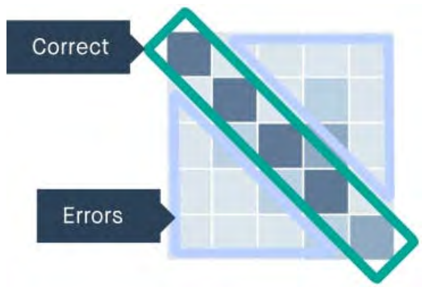


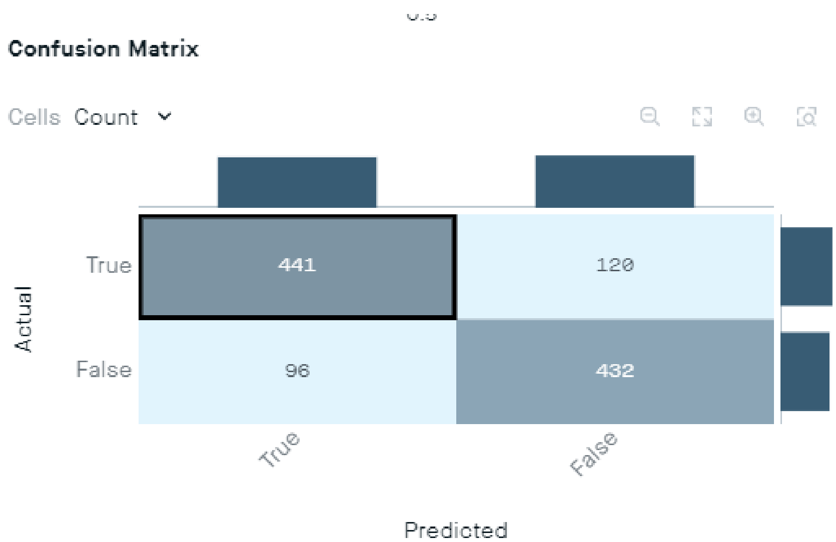
The training loss will decrease for each epoch, but the evaluation loss may start to increase. This means that the model is starting to overfit to the training data.

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

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

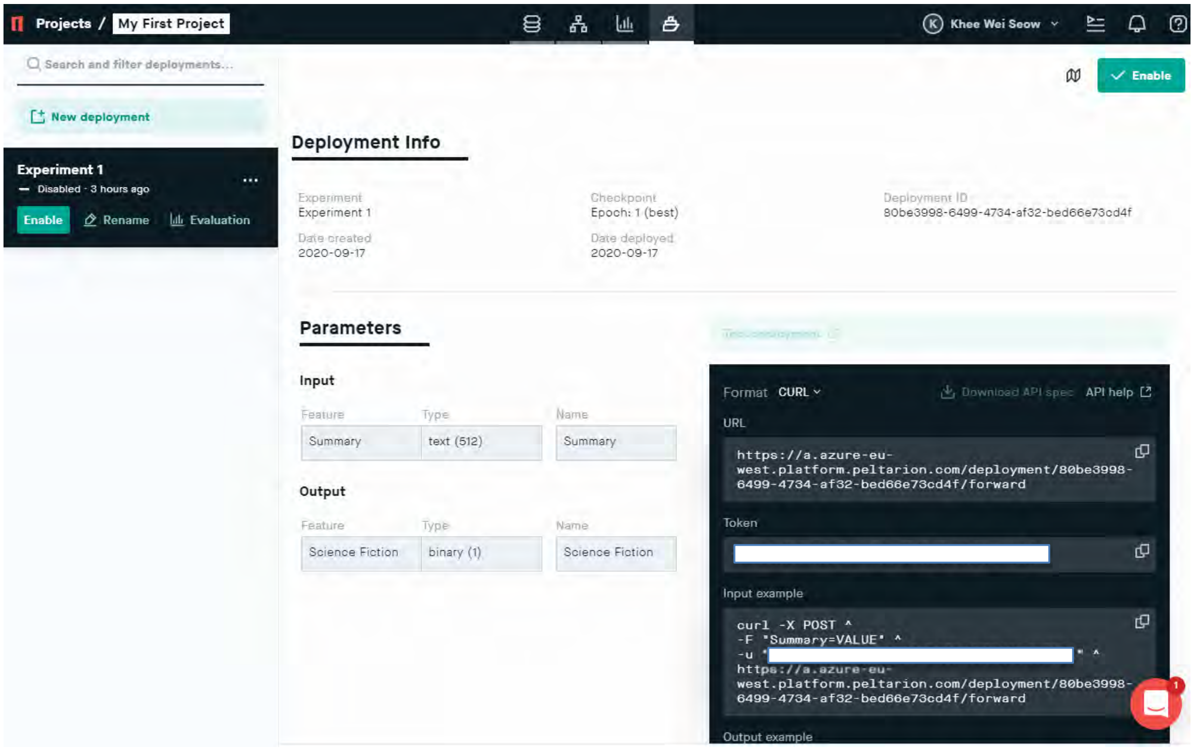
Confusion matrix
Since the model solves a classification problem, a confusion matrix is displayed. The top-left to bottom-right diagonal shows correct predictions. Everything outside this diagonal are errors.





7) Create new deployment

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

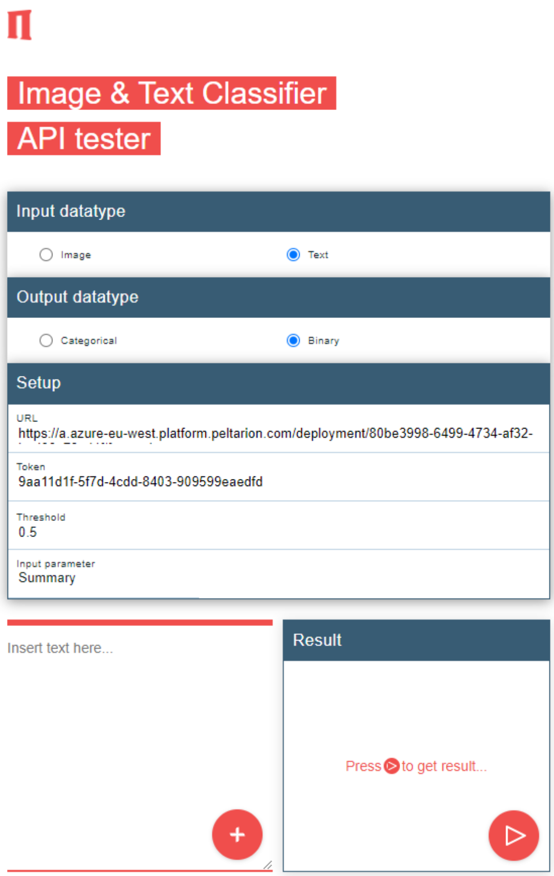


2) Select experiment and checkpoint of your trained model to test it for predictions, or enable for business product calls. Both best epoch and last epoch for each trained experiment are available for deployment.

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

8) Test the text classifier in a browser

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



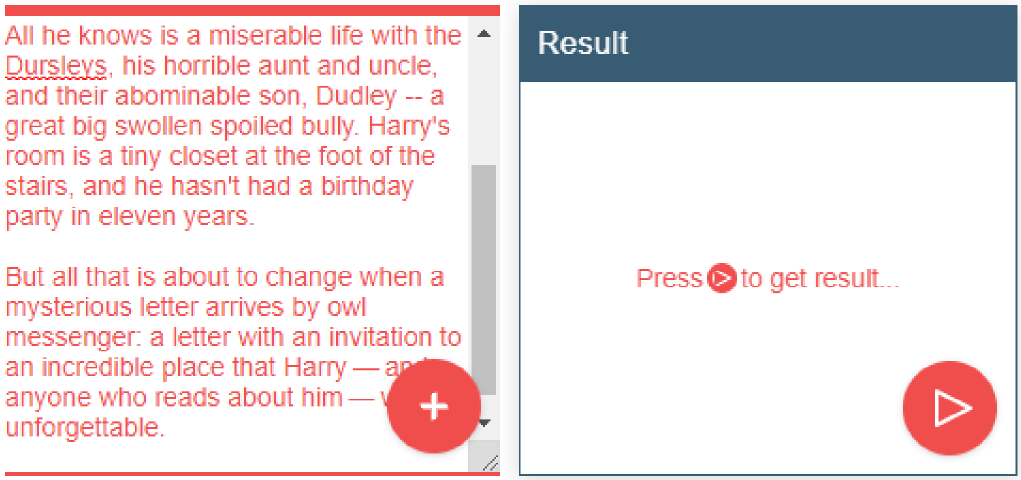
2) Now, write your own summary, copy the example below or simply copy a recent summary from:

Example:
Harry Potter has never been the star of a Quidditch team, scoringpoints 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.

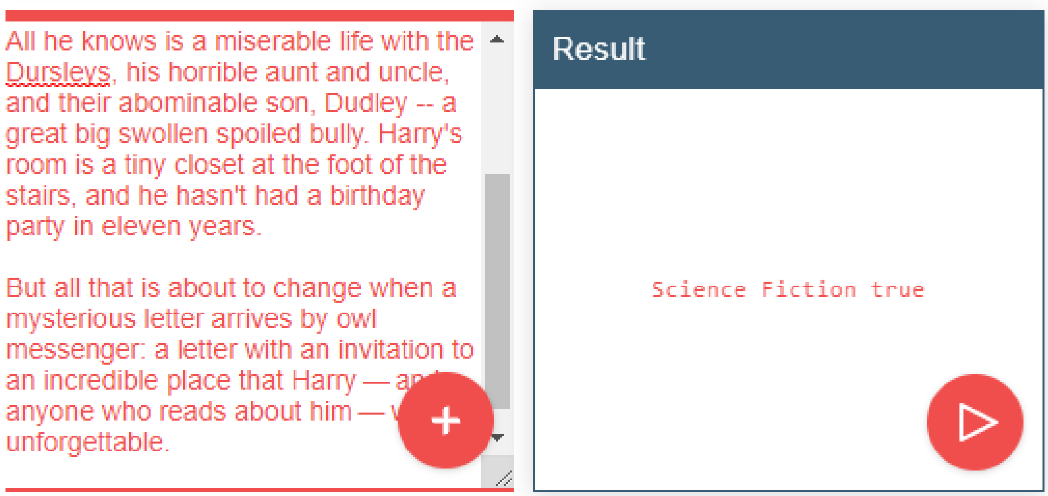
All he knows is a miserable life with the Dursleys, his horrible aunt and uncle, and their abominable son, Dudley -- a great big swollen spoiled bully. Harry's room is a tiny closet at the foot of the stairs, and he hasn't had a birthday party in eleven years.

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:



3) Click **Play** to get a result.



9) Next Steps

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

Activity 6 – [Bonus] Importing data

In this activity, we will learn:
- ❑ How to upload CSV file into Azure Machine Learning Studio (Classic)
- ❑ How to import a CSV file from the web

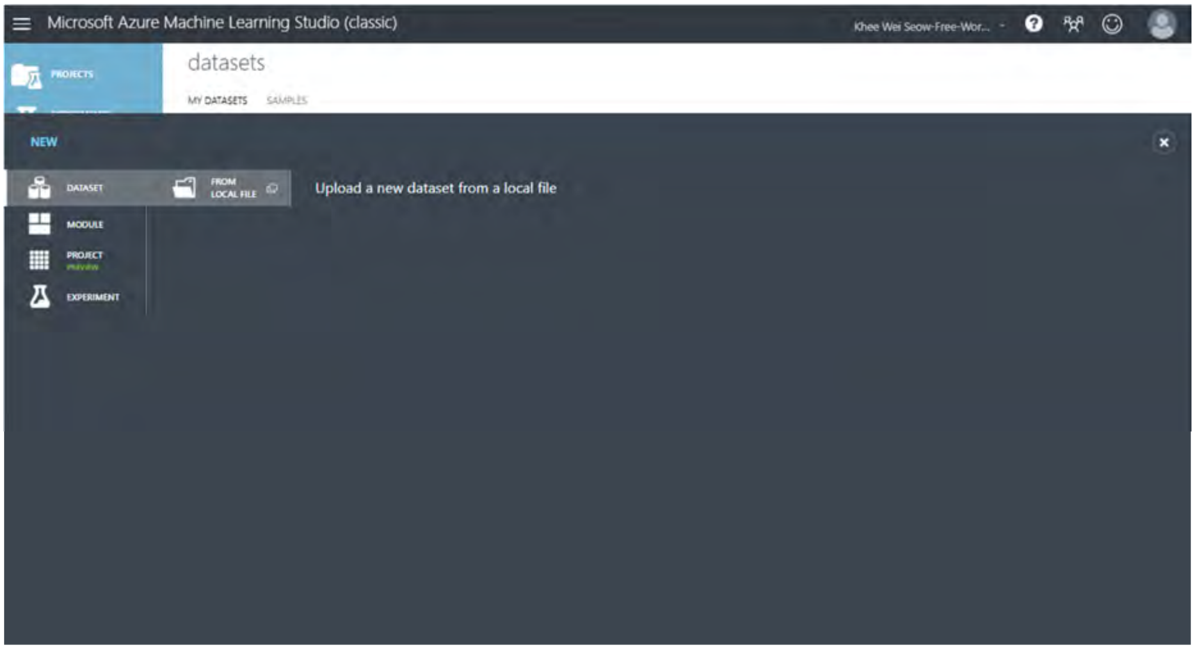
1) To use your own data in Machine Learning Studio (classic) to develop and train a predictive analytics solution, you can use data from:

Local file - Load local data ahead of time from your hard drive to create a dataset module in your experiment.
Online data sources - Use the Import Data module to access data from one of several online sources while your experiment is running
Machine Learning Studio (classic) experiment - Use data that was saved as a dataset in Machine Learning Studio (classic). For a list of datasets available in Studio (classic), you may want to refer to https://docs.microsoft.com/en-us/azure/machine-learning/studio/use-sample-datasets
SQL Server database - Use data from a SQL Server database without having to copy data manually

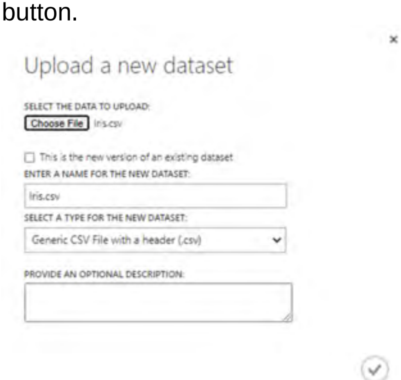
2) Import from a local file

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

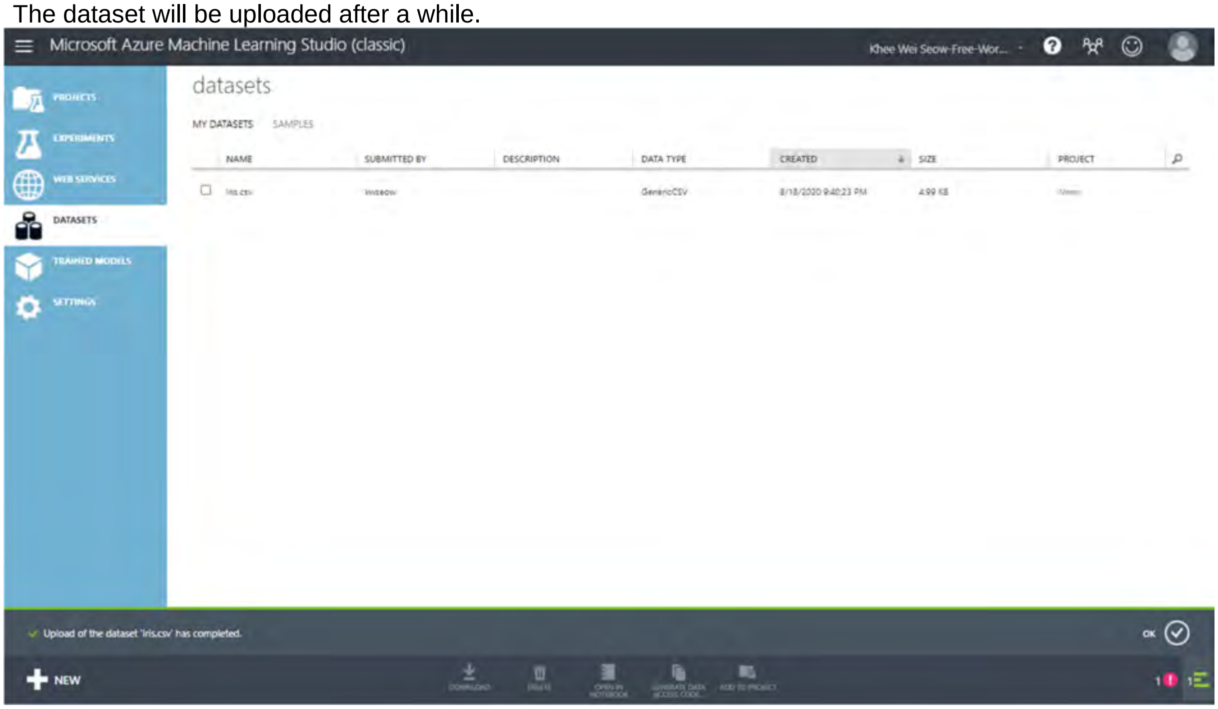
2) Click **+NEW** and then **FROM LOCAL FILE**.



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

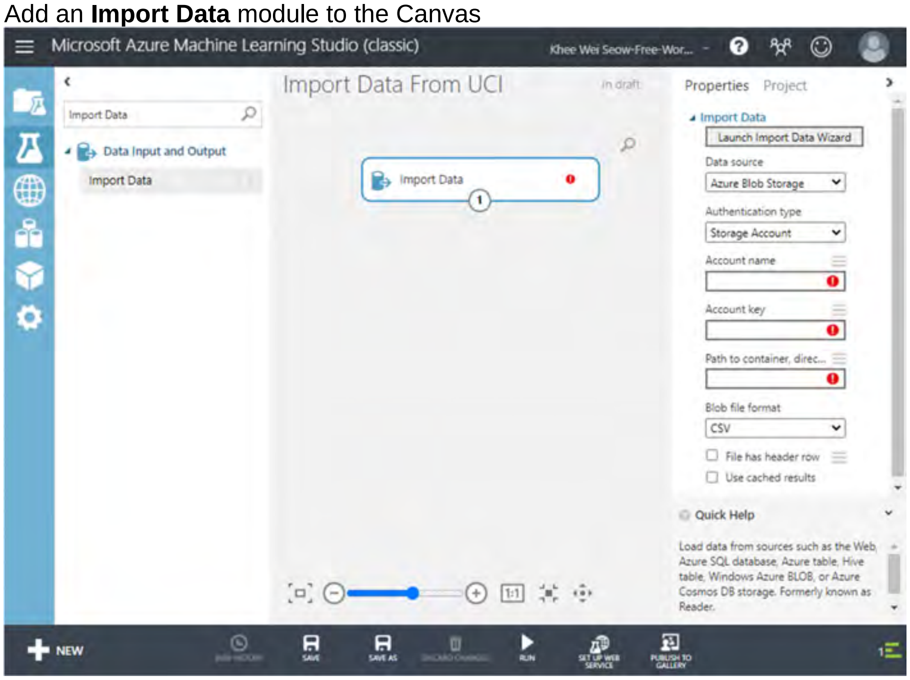


4) The dataset will be uploaded after a while.

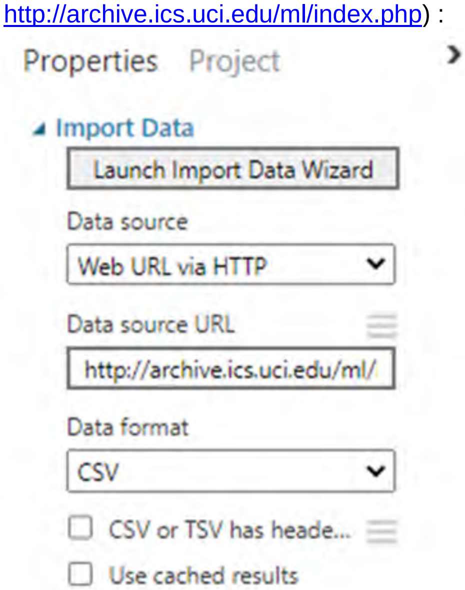


3) Import directly from the web

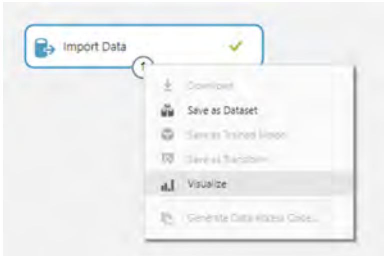
1) Create a new **Black Experiment** and name it Import dataset from UCI.
2) Add an **Import Data** module to the Canvas



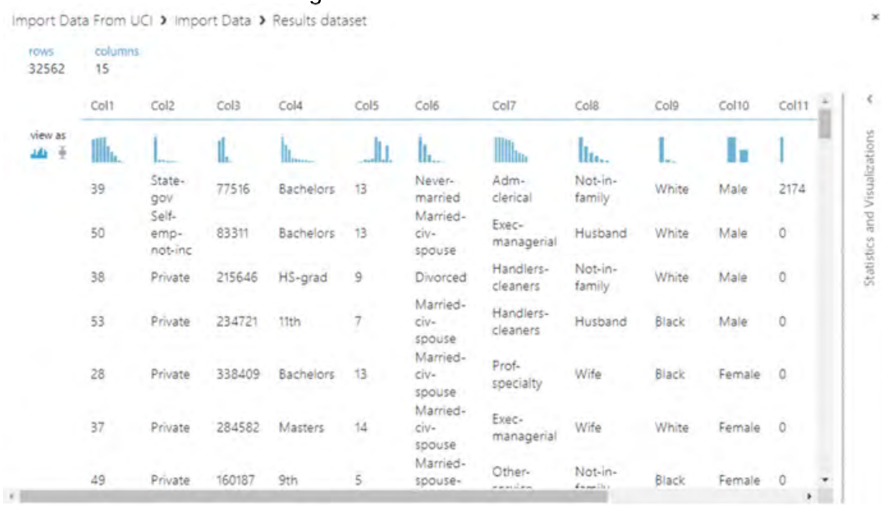
3) Set the properties of the Import Data module as follows (Data Source URL: http://archive.ics.uci.edu/ml/index.php) :



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



6) You should see the following:



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

Activity 7 – [Bonus] Cleaning and Structuring Data

In this activity, we will learn:
- ❑ How to summarise data
- ❑ How to remove missing values
- ❑ How to remove duplicate records
- ❑ How to remove outliers
- ❑ How to read a boxplot

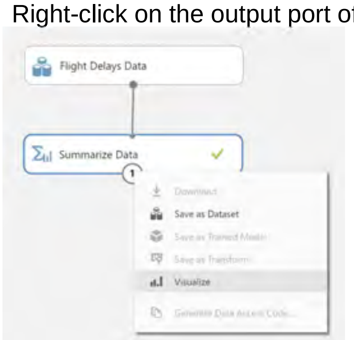
In Studio (classic), create a new Blank Experiment using **+ New** and call it **Data Cleaning**.

1) Summarizing Data

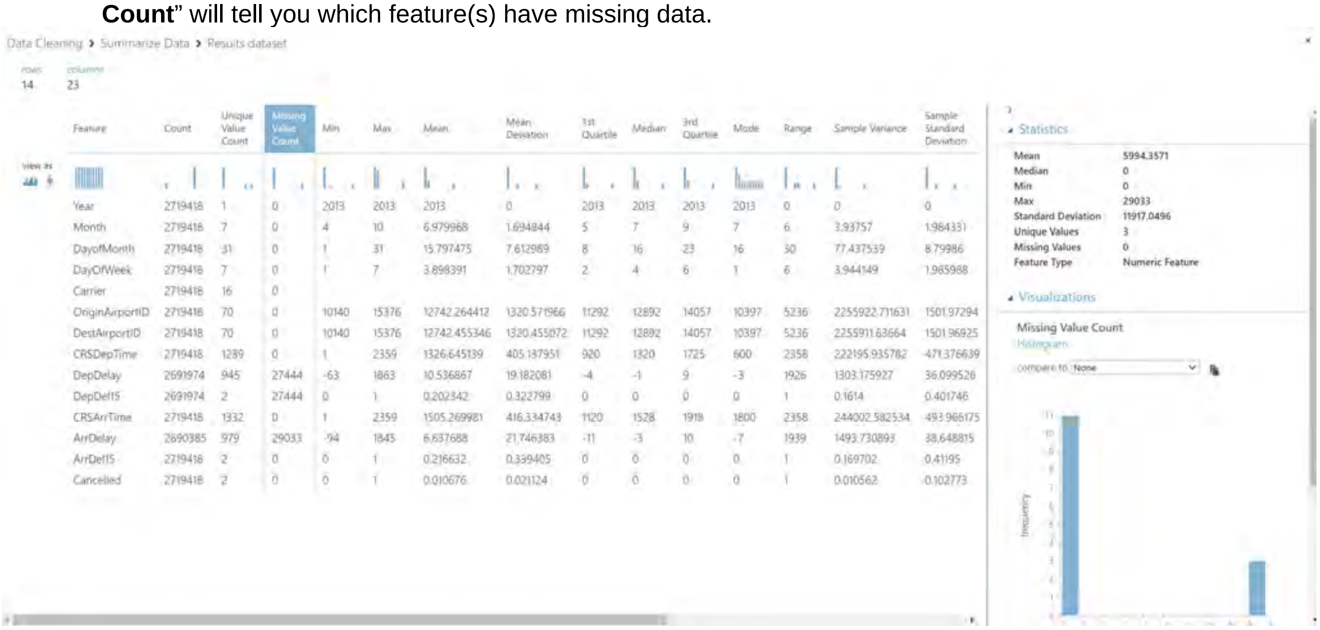
1) Add the following dataset and module onto the canvas.

2) Click **Run**,

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

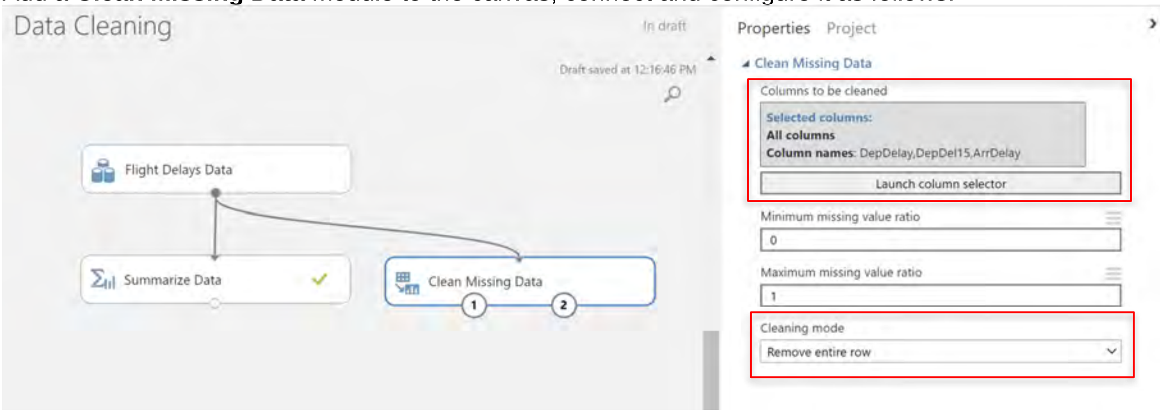


4) You should see a similar screen (below) that shows a summary of the dataset. The column "**Missing Value Count**" will tell you which feature(s) have missing data.

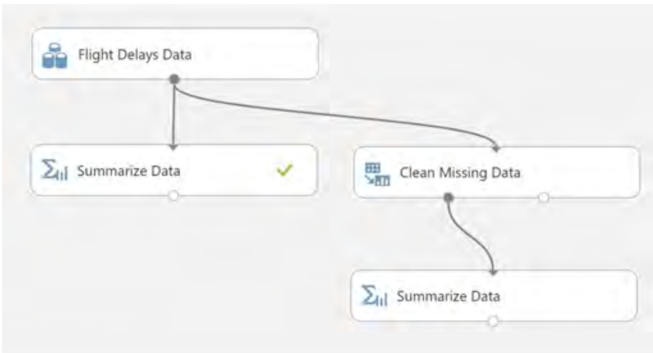


2) Missing Value

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

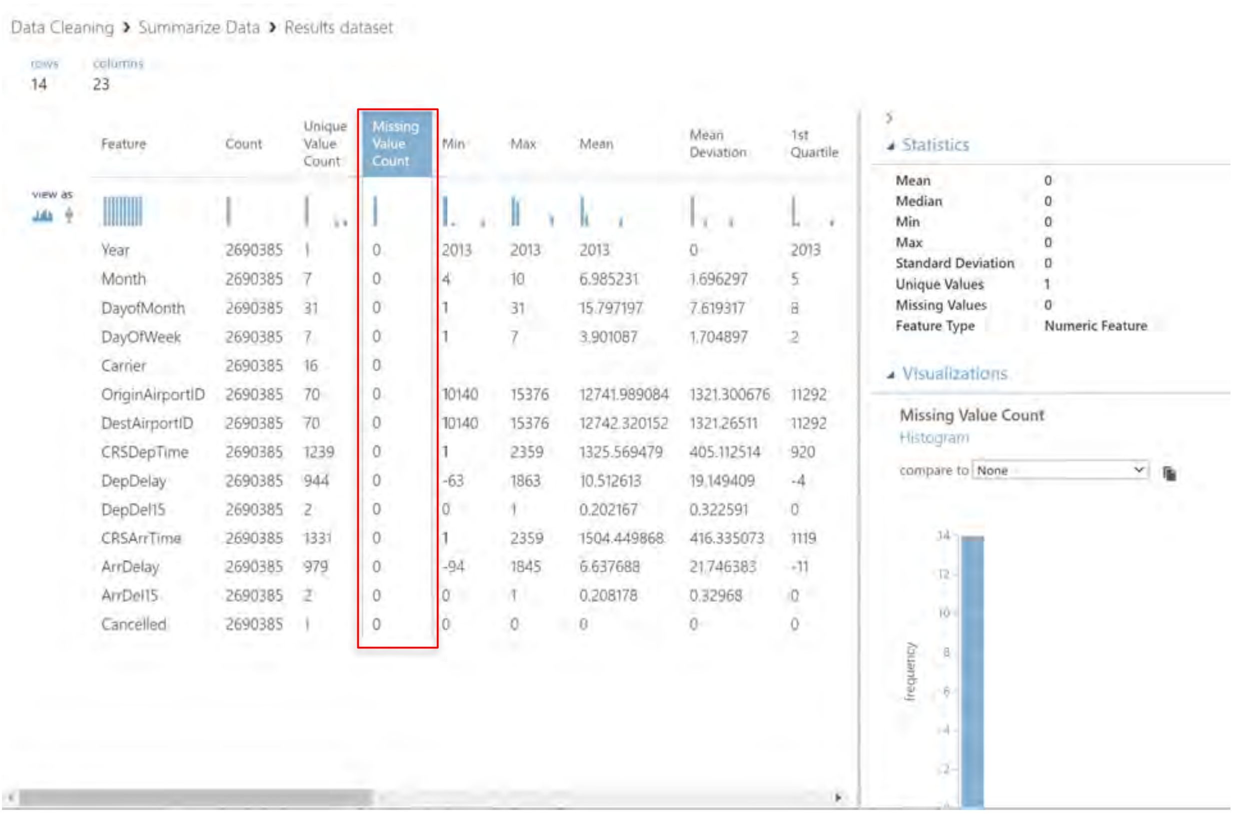


2) Add a **Summarize Data** module to the canvas as follows:



3) Click on **Run**

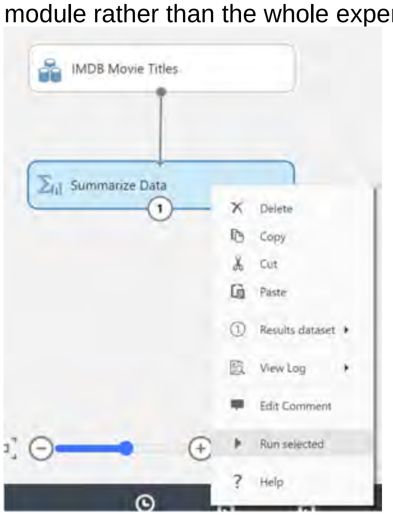
4) Visualise the output of the newly added **Summarize Data** module. There should be no missing values in the dataset now.



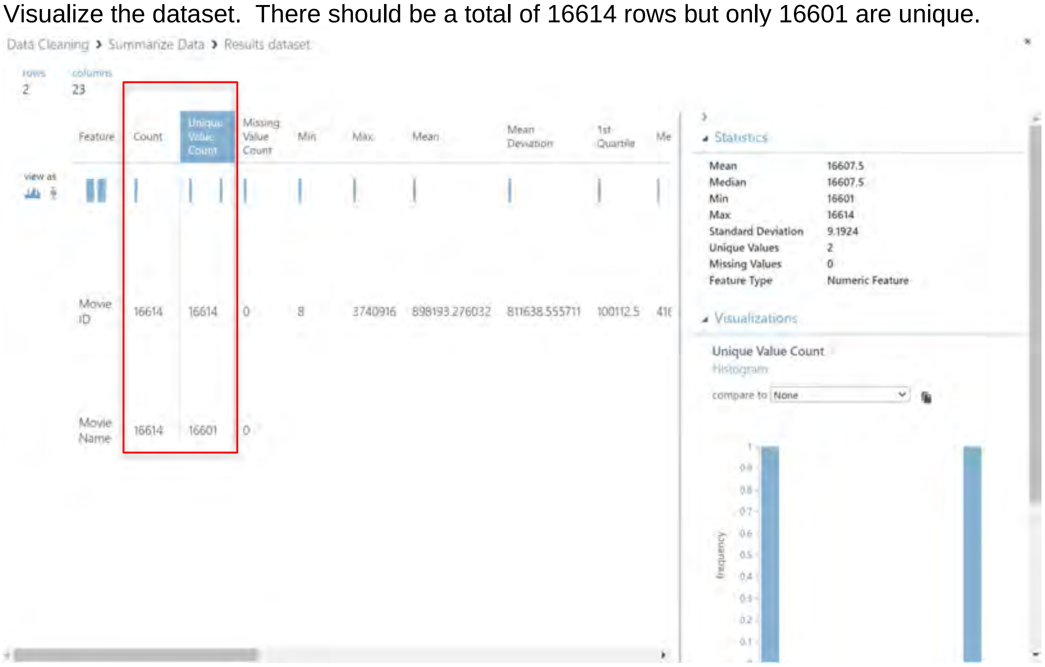
3) Duplicate Records

1) Add the **IMDB Movie Titles** and **Summarize Data** module to the canvas.

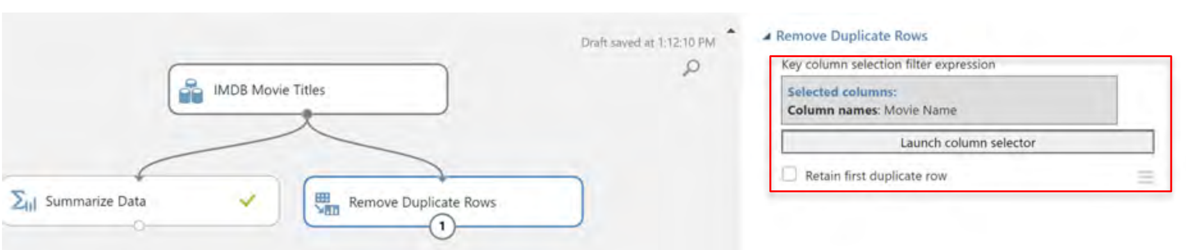
2) Right-click on the **Summarize Data** module and select **Run selected**. This action will only run the selected module rather than the whole experiment saving some time.



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

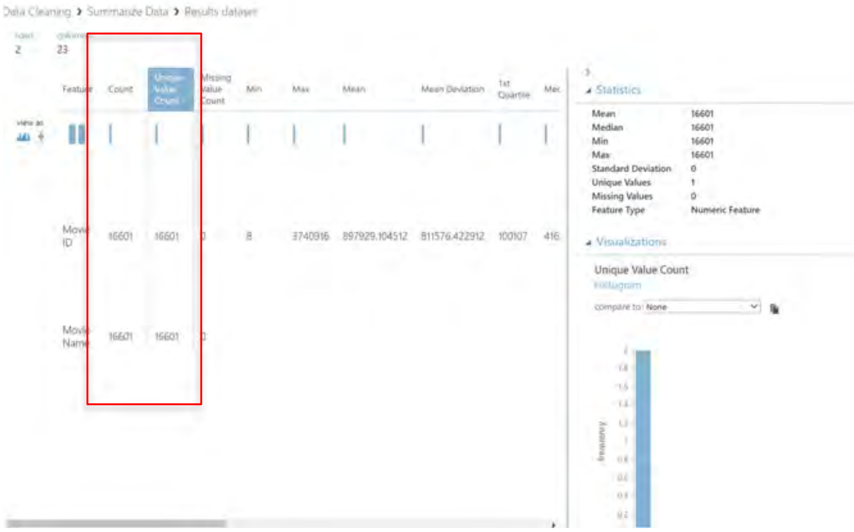


4) Add a **Remove Duplicate Rows** module to the canvas, connect and configure it as follows:



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

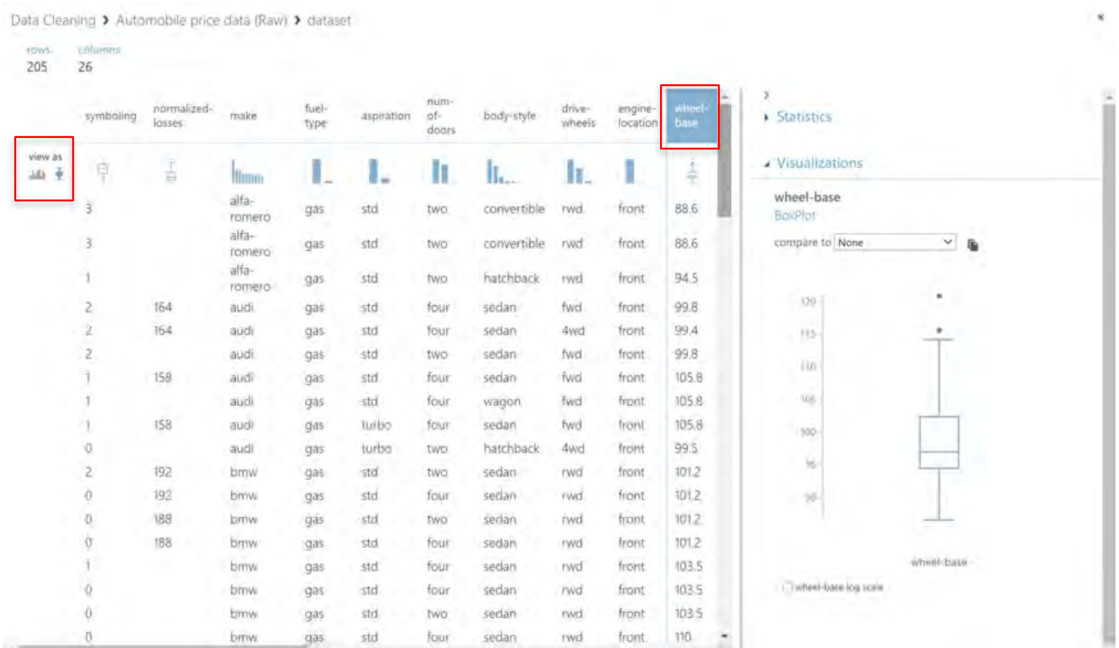
5) Add a **Summarize Data** module and **Visualize** the output. There should be a total of 16601 rows and the same number of unique rows.



4) Removing Outliers

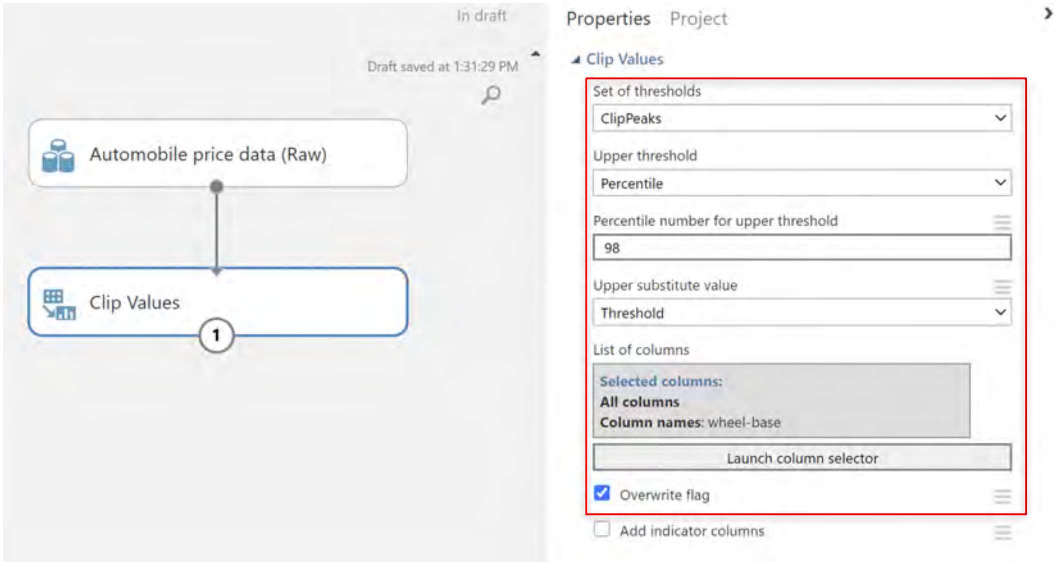
1) Add the **Automobile price data (Raw)** to the canvas.

2) **Visualize** and view as **Box-Plot** the wheel-base feature.

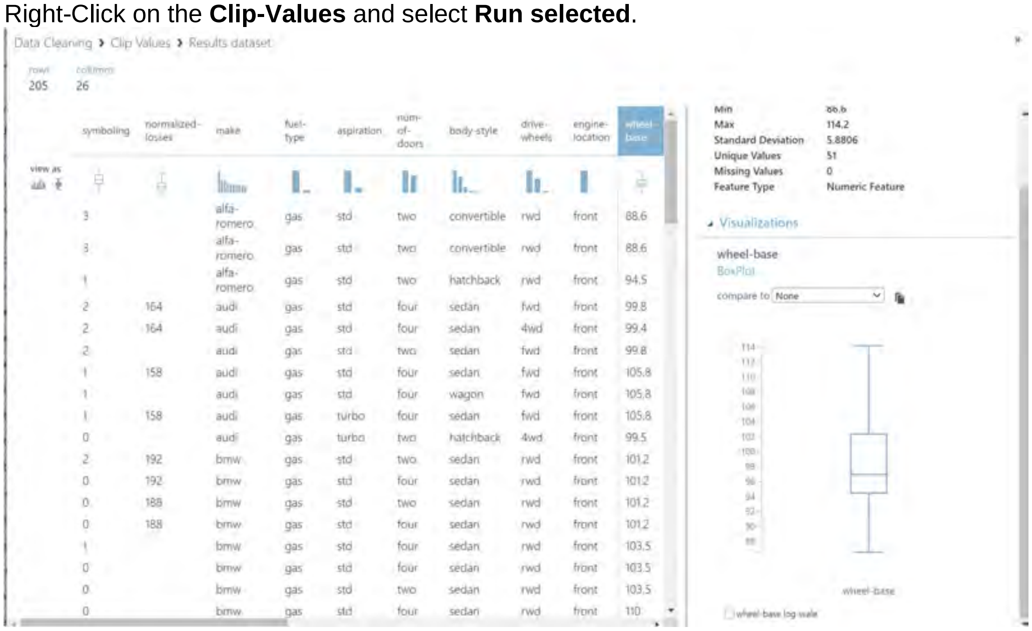


Note: The dots outside the boxplot are the outliers, and they should be removed from the dataset as they may affect the accuracy of the prediction.

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



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



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

Activity 8 – [Bonus] Using Binary Classification Algorithms

In this activity, we will learn:
- ❑ How to use the Two-Class Logistic Regression algorithm for training
- ❑ How to use the Two-Class Boosted Decision Tree algorithm for training
- ❑ How to evaluate two learning algorithms
- ❑ How to save a trained model

1) Create a new experiment and name it as **Binary Classification – Income**.

2) Add the **Adult Census Income Binary Classification** dataset to the canvas.

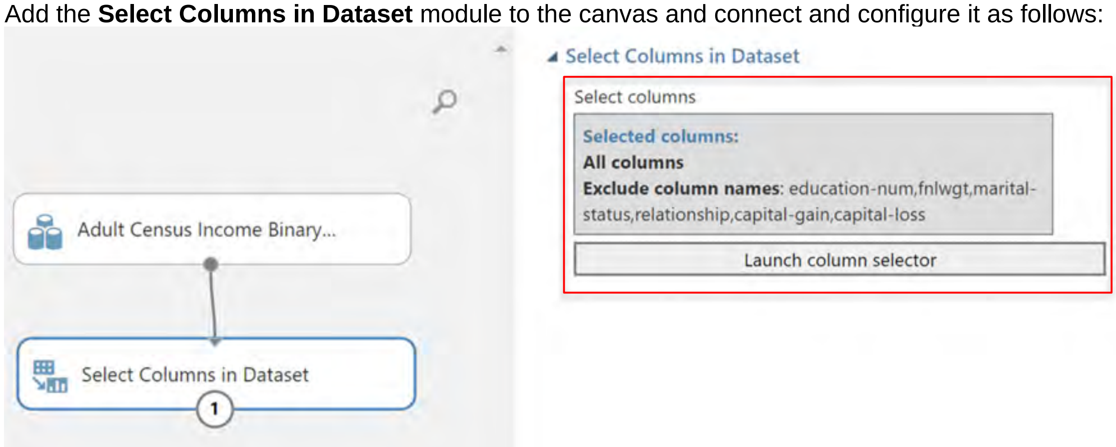
3) Right-click on the output port of the dataset and select Visualize. You should see the following:


Binary Classification - Income › Adult Census Income Binary Classification dataset › dataset

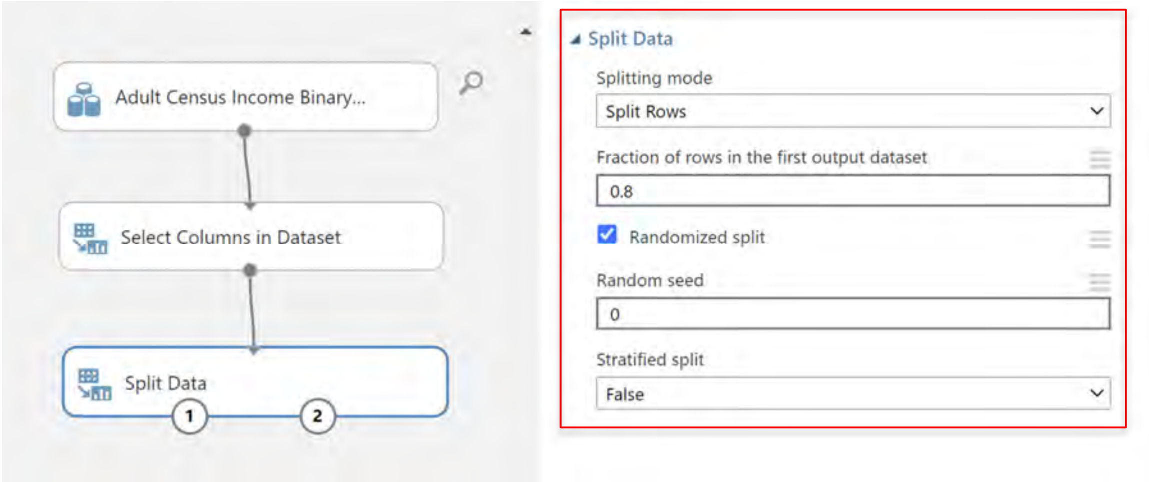
rows 32561 columns 15

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K

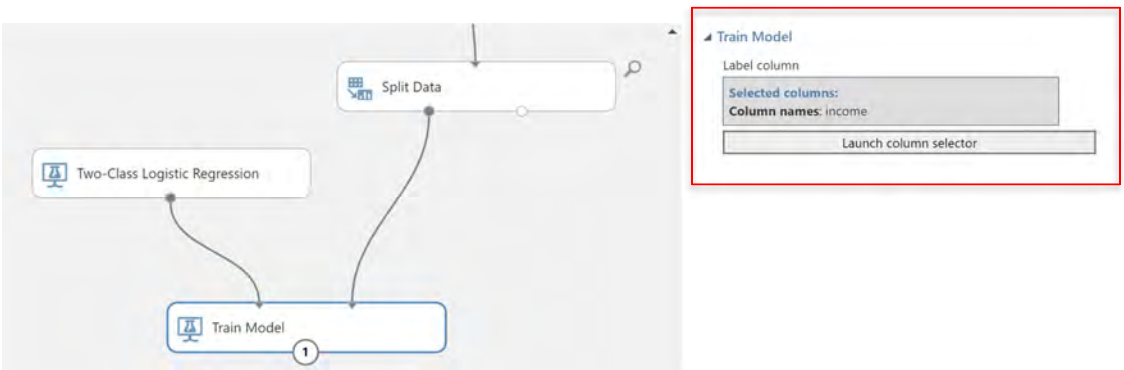
4) Add the **Select Columns in Dataset** module to the canvas and connect and configure it as follows:



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



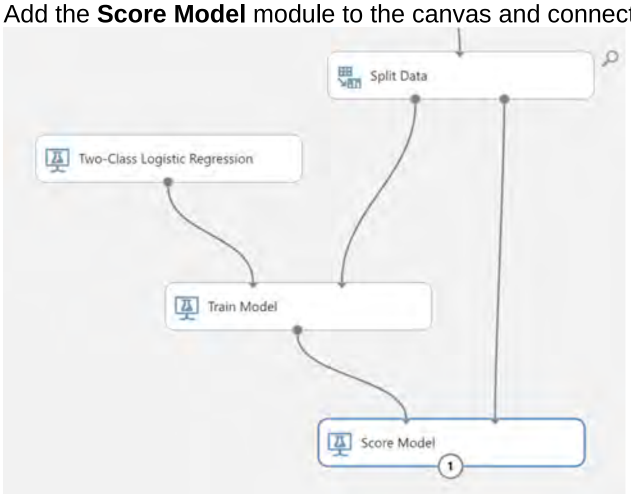
6) Add a **Two-Class Logistic Regression** and **Train Model** modules to the canvas, connect and configure them as follows:



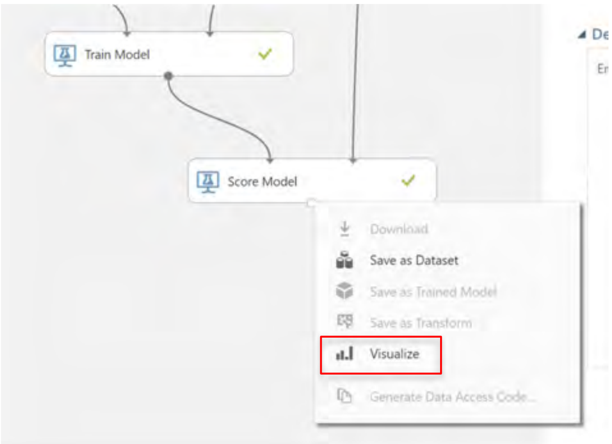
Ref:
https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-logistic-regression
https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/train-model

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

rows 6512 columns 11

age	workclass	education	occupation	race	sex	hours-per-week	native-country	income	Scored Labels	Scored Probabilities
34	Private	Some-college	Machine-op-inspct	White	Male	35	United-States	<=50K	<=50K	0.140938
17	Private	11th	Other-service	White	Male	15	United-States	<=50K	<=50K	0.006722
29	Private	Some-college	Prof-specialty	White	Male	15	United-States	<=50K	<=50K	0.130627
36	Private	Assoc-voc	Handlers-cleaners	White	Male	38	United-States	<=50K	<=50K	0.119357
21	Private	HS-grad	Farming-fishing	White	Male	53	United-States	<=50K	<=50K	0.054273
51	Self-emp-not-inc	HS-grad	Farming-fishing	White	Male	45	United-States	<=50K	<=50K	0.103781
37	Private	HS-grad	Other-service	Black	Female	40		<=50K		
53	Private	HS-grad	Craft-repair	White	Male	46	United-States	>50K	<=50K	0.370918
36	Private	Assoc-voc	Craft-repair	White	Male	40	United-States	<=50K	<=50K	0.282886
45	Private	Assoc-voc	Adm-clerical	White	Female	40	United-States	<=50K	<=50K	0.135465
36	Private	Assoc-voc	Craft-repair	White	Male	40	United-States	<=50K	<=50K	0.282886
24	Private	HS-grad	Sales	White	Female	30	United-States	<=50K	<=50K	0.034528
45	Local-gov	Masters	Prof-specialty	Black	Male	10	United-States	>50K	<=50K	0.325163
33	Private	Bachelors	Exec-managerial	White	Male	40	United-States	>50K	>50K	0.517066

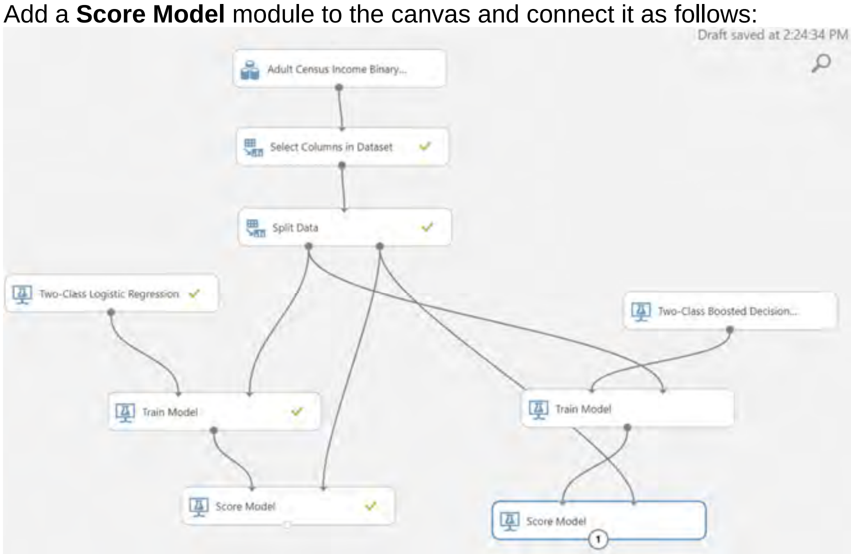
Note:
The **Scored Labels** column shows the predicted income group.
The **Scored Probabilities** column shows the confidence levels of the scored labels
If your test dataset is missing the dependent values, your **Scored Labels** may be empty.

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

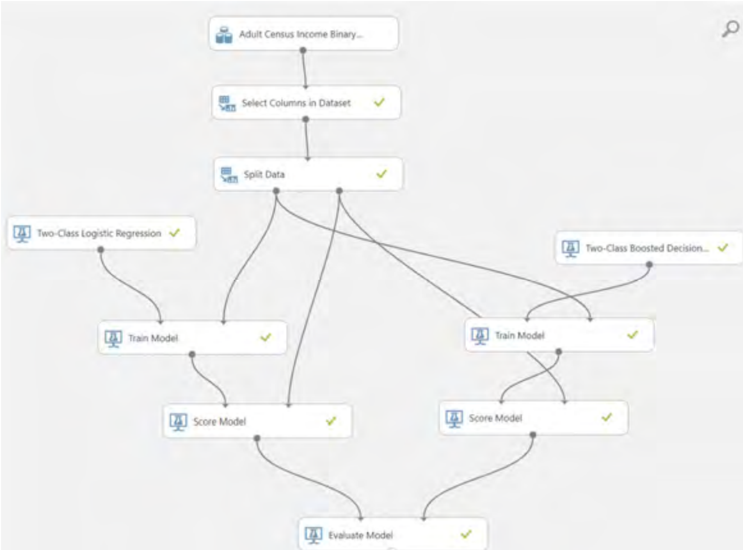
Ref:
https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-boosted-decision-tree

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:



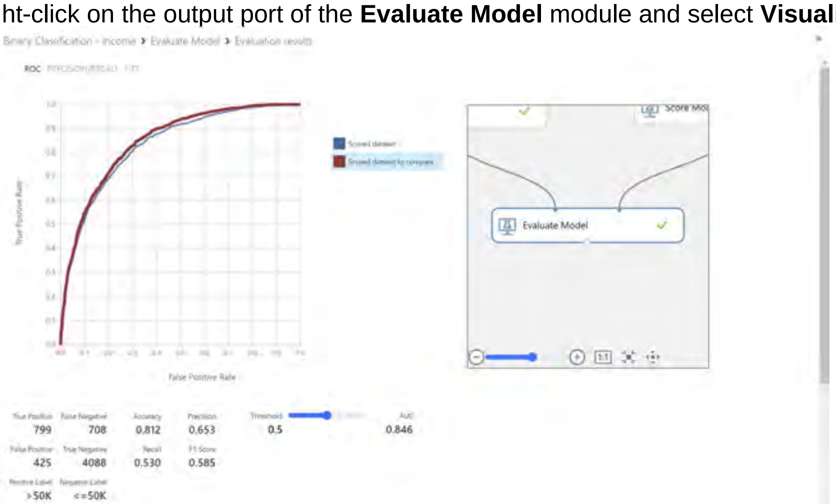
12) Add the **Evaluate Model** module to the canvas and connect it as follows:



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.

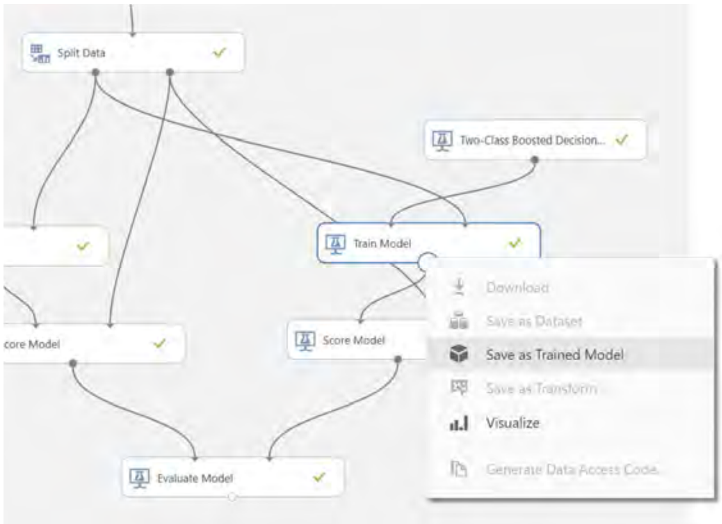
Evaluation metrics

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.

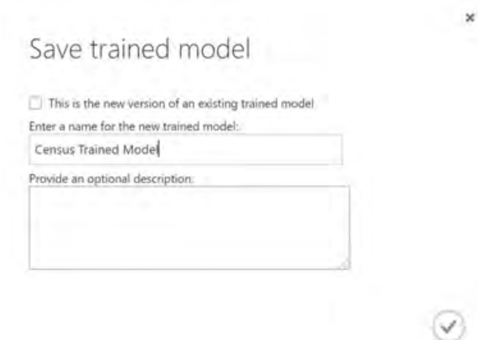
15) Saved a train model

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

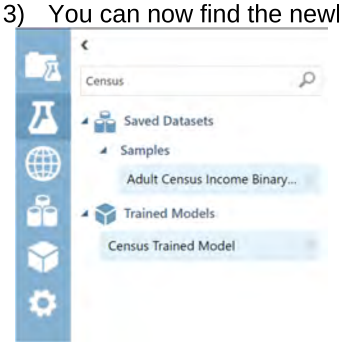


Note: Once a **Train Model** is trained, you can save it as a trained model so that you can later use it on the canvas without specify the algorithm.

2) Name the trained model as **Census Trained model**:



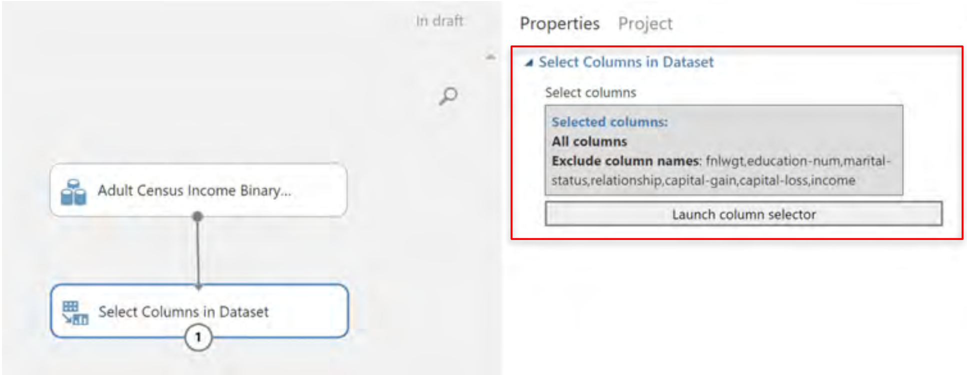
3) You can now find the newly saved trained model in the left panel.



4) Create a new experiment and name it as **Income Prediction**.

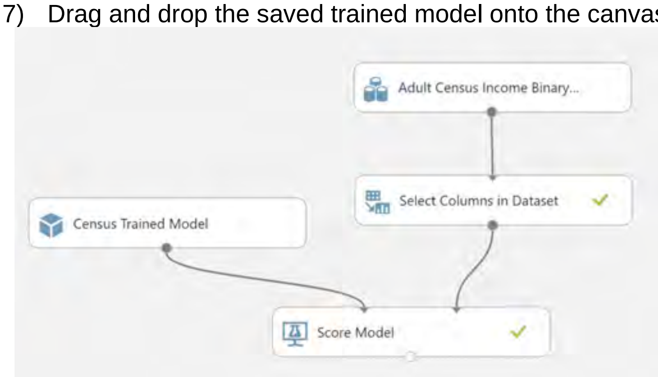
5) Add the **Adult Census Income Binary Classification** dataset to the canvas.

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



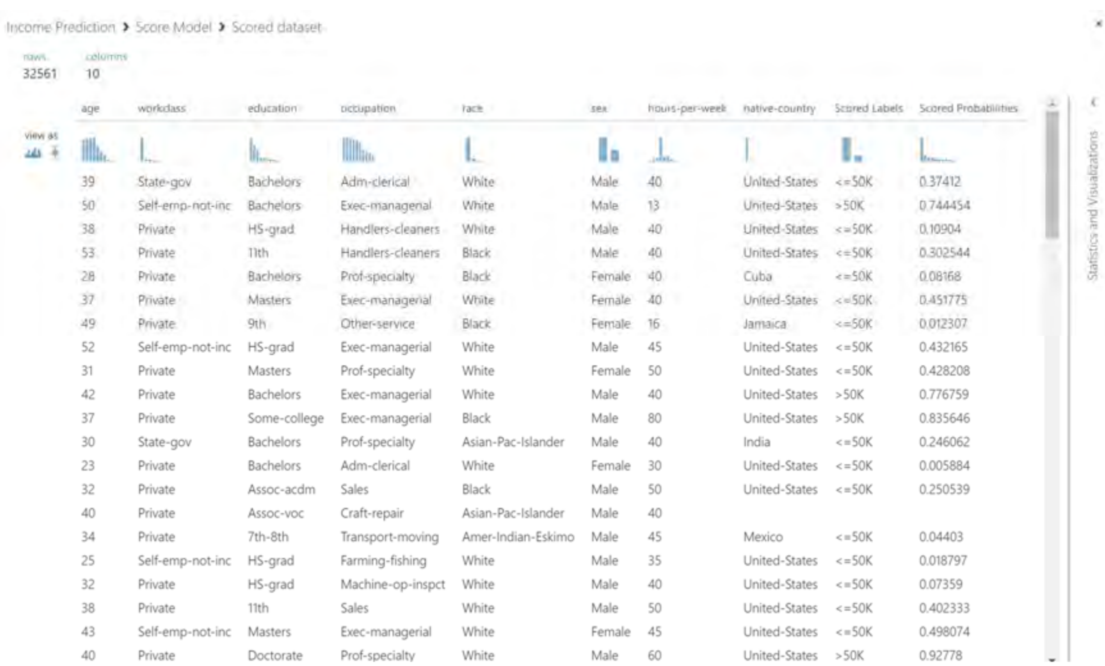
Note: We have excluded **income** column in this case. Our model is going to predict income, so there is no need to have the income as part of the inputs.

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**:



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

Note:
Observe that **Income** column is no longer visible.
Scored Labels and **Scored Probabilities** have been added. Scored Labels is the predicted income.

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