Lab Exercise #2 - Logistic Regression

Instruction:
Read the instructions carefully and follow each step.

Provide your best answers for all questions.

The name, email, and photo associated with your Google account will be recorded when you upload files and submit this form

* Indicates required question

Email *

Record my email address with my response

Name *

Example: Dela Cruz, Juan C.

Your answer

Student Number *

Example: 24B1234

Your answer

Section *

- ○ 4A
- ○ 4B
- ○ 4C
- ○ 4D
- ○ 4E
- ○ 4F

Building a Logistic Regression Model

For this lab exercise, you will build a **machine learning model using Logistic Regression to classify photos of handwritten numbers from 0 to 9.**

You will also deploy this model in an interactive web app, where it will classify photos uploaded by users.

Note: For this exercise, assume that the uploaded photo will contain only one handwritten number. You may explore additional scenarios, such as recognizing multiple numbers in a single photo, beyond the scope of this lab.

Follow these steps to build the model and answer the questions along the way.

Step 1: Create a New Orange Workflow

Open Orange Data Mining. *

Go to File > New to create a new workflow.

- ○ Done

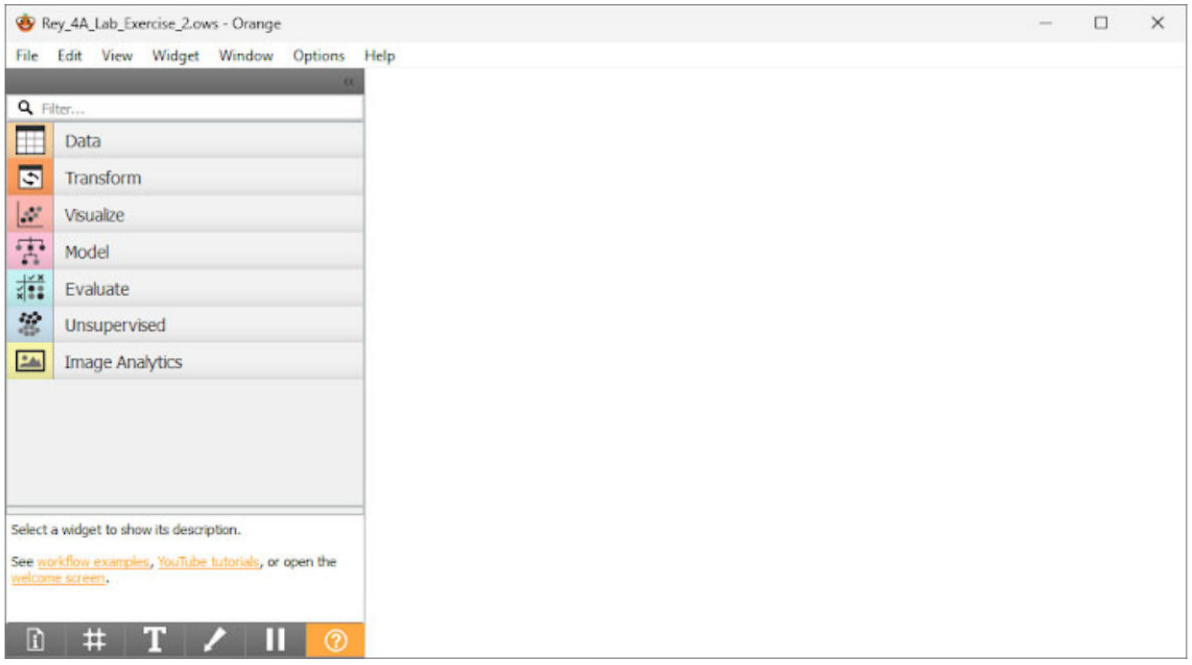
To save your workflow, go to File > Save or press Ctrl + S. *

Choose an accessible directory and name the file using this format:
[Last_Name]_[Section]_Lab_Exercise_2 (e.g., Rey_4A_Lab_Exercise_2).

◯ Done

Take a screenshot of the window showing your file name and upload it here. *

Example:



Add file

Step 2: Additional Setup in Orange

To load and view images in Orange, the Image Analytics add-on must be installed.

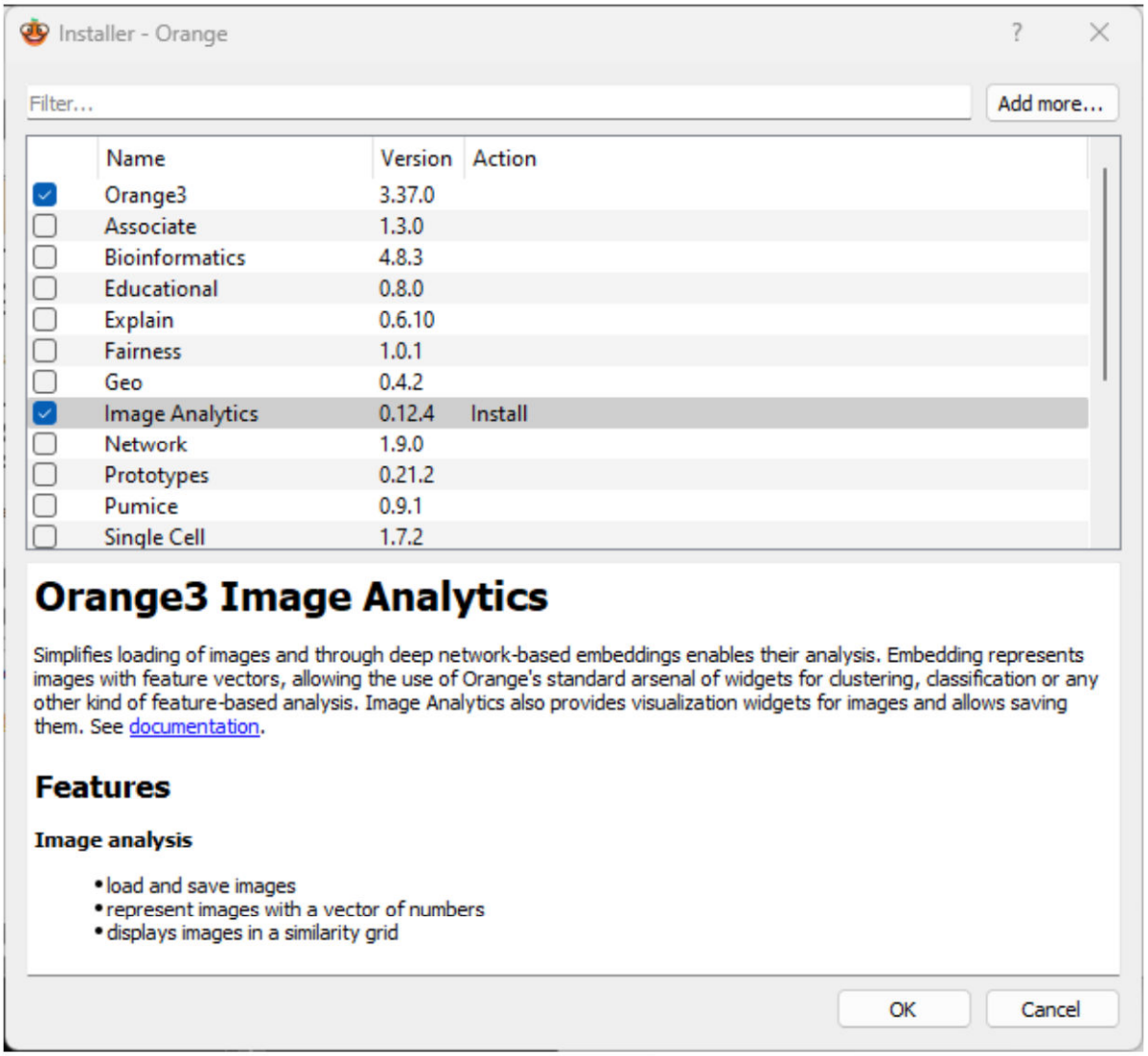
An internet connection is required.

Go to Options > Add-ons to open the Installer window. *

◯ Done

Check the box for the Image Analytics add-on. *

You should end up with this:



◯ Done

Click OK to install the add-on. *

This will take a few minutes.

- [ ] Done

Orange needs to restart for the changes to take effect. *

Click OK to restart Orange and save the workflow if prompted.

- [ ] Done

To reopen your workflow, go to File > Open Recent and find your file. *

- [ ] Done

Step 3: Collect the Data

If you build a Logistic Regression model to classify handwritten digits from 0 to 9, you will need images of handwritten numbers for training.

Download the MNIST handwritten digit dataset in PNG format from Kaggle here. *

An account is required.

- [ ] Done

This dataset includes 28x28 pixel images of digits, sorted into subfolders from 0 to 9. *

It contains 60,000 training images and 10,000 test images.

The original MNIST dataset is in a binary format, which often requires additional preprocessing to read and use, whereas the PNG version on Kaggle is more straightforward to handle, especially for beginners.

- ◯ I have read and understood.

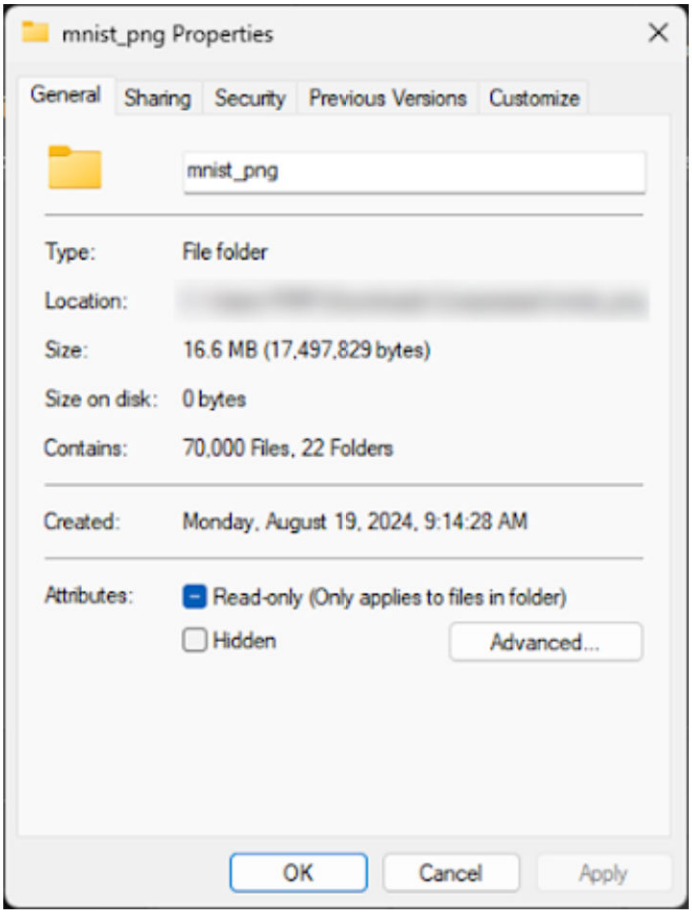
After downloading, unzip the file to an accessible directory. *

You will get a folder named "mnist_png".

- ◯ Done

Right-click the folder and select Properties. *

For Windows users, a window will show that it contains 70,000 files and 22 subfolders, like this:



- [ ] Done

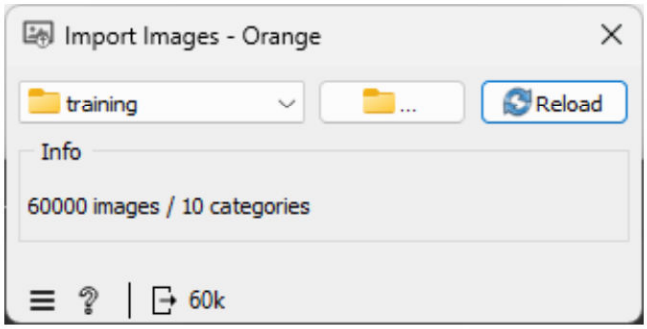
In your workflow, drag and drop an "Import Images" widget. *

- [ ] Done

Open the widget. *

Click the Folder icon and locate the "training" folder within "mnist_png", and click Reload.

You should end up with this:



- [ ] Done

To view images, drag and drop an "Image Viewer" widget. *

Connect it to the "Import Images" widget.

- [ ] Done

Open the widget to review the dataset images. *

Use the "Image Size" slider to zoom in and out.

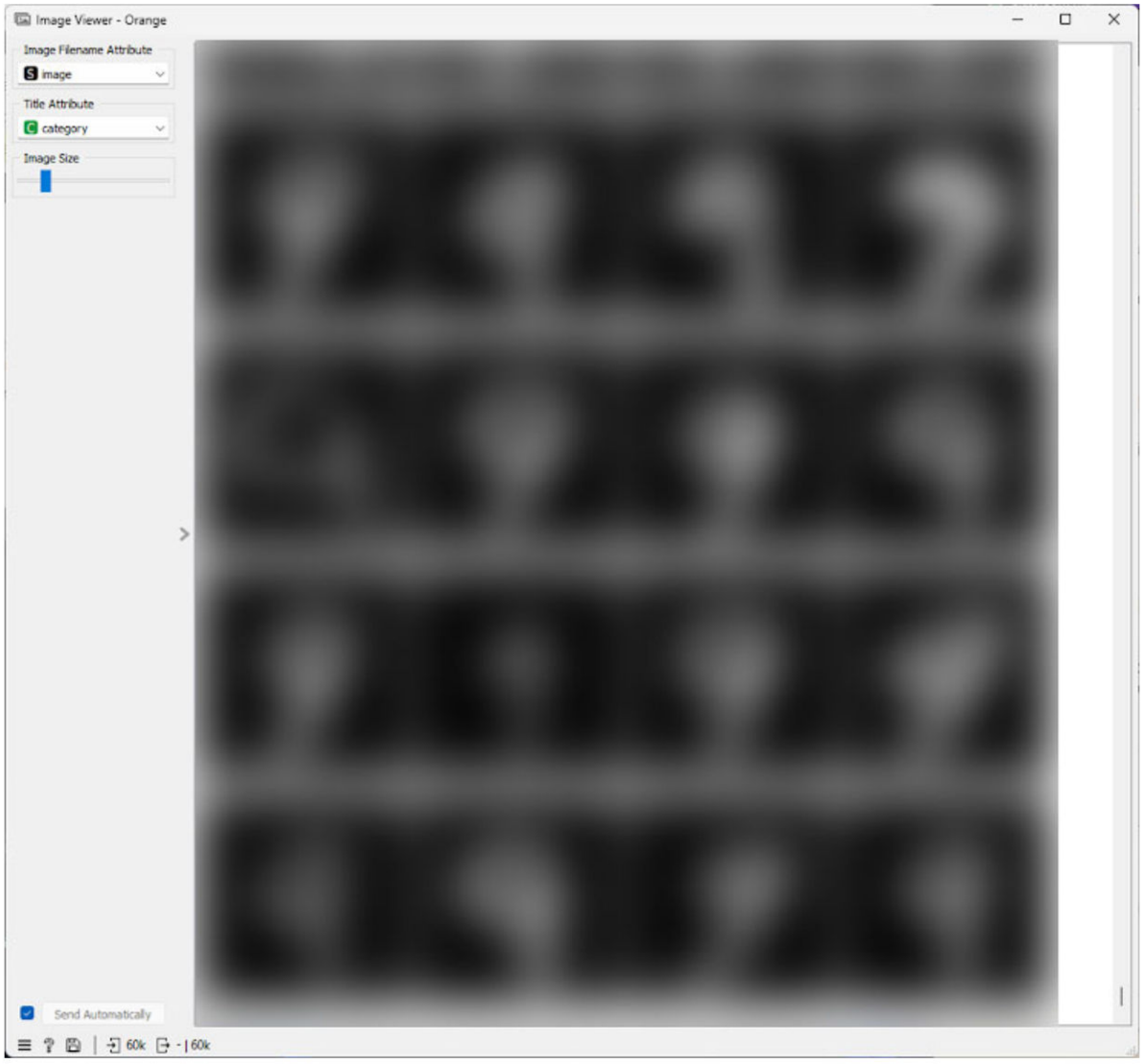
- [ ] Done

Take a screenshot of the last 12 images in the "Image Viewer" window, showing labels of 9. *

Scroll to the bottom and use the zoom slider if needed.

Upload it here.

Example:



Add file

Step 3: Preprocess the Images

To enable Logistic Regression to process the images, they must be converted to a numerical format.

You can use the "Image Embedding" widget to convert images into numerical formats using deep neural networks, but in this exercise, **we will work directly with pixel values.** *

A pixel is the smallest unit of an image, similar to pieces of a puzzle.

In an 8-bit grayscale image, each pixel value ranges from 0 to 255, where:
• **0** represents **black**,
• **255** represents **white**, and
• **Values in between** represent varying shades of **gray**.

For your lab exercise, you will convert images into numerical values.

Since each image is 28x28 pixels, this results in 784 pixel values per image, each ranging from 0 to 255.

These pixel values will be used as input features for Logistic Regression.

◯ I have read and understood.

Due to Orange's limitations, we will do some coding. *

Drag and drop a "Python Script" widget and connect it to the "Import Images" widget.

◯ Done

Open the widget. *

In the "Library" section, click the "+" icon to add a new script.

This will create a script called "New script".

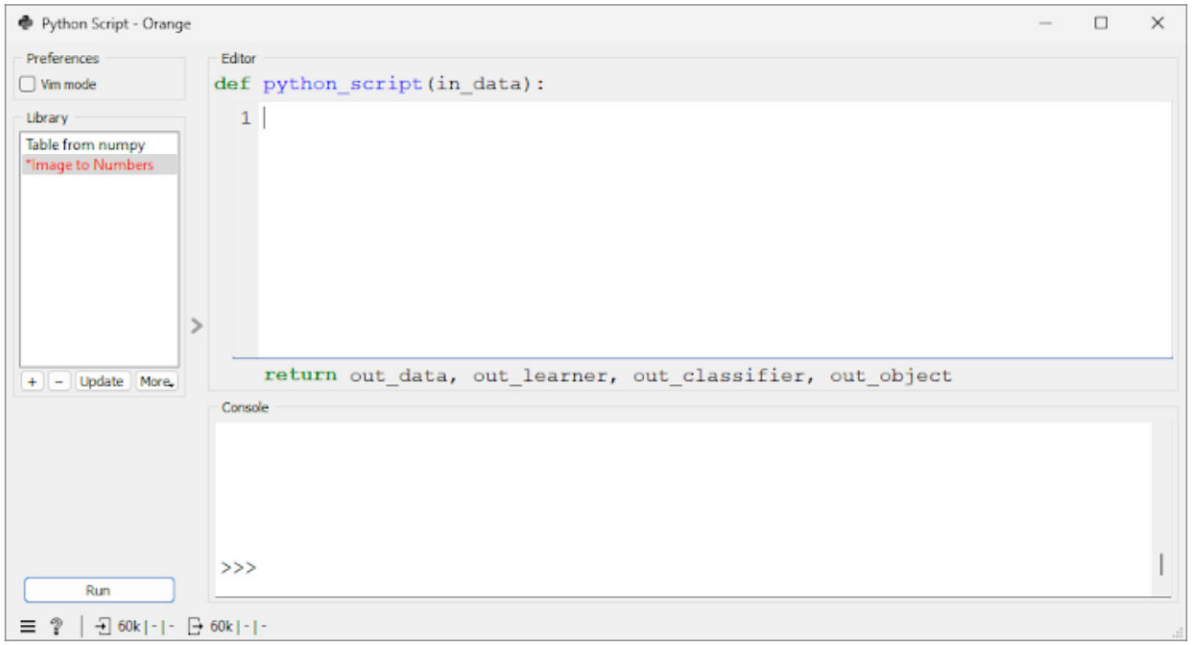
Double-click on "New script" and rename it to "Image to Numbers".

◯ Done

Select "Image to Numbers". *

In the "Editor" section, remove any existing code.

You should end up with this:



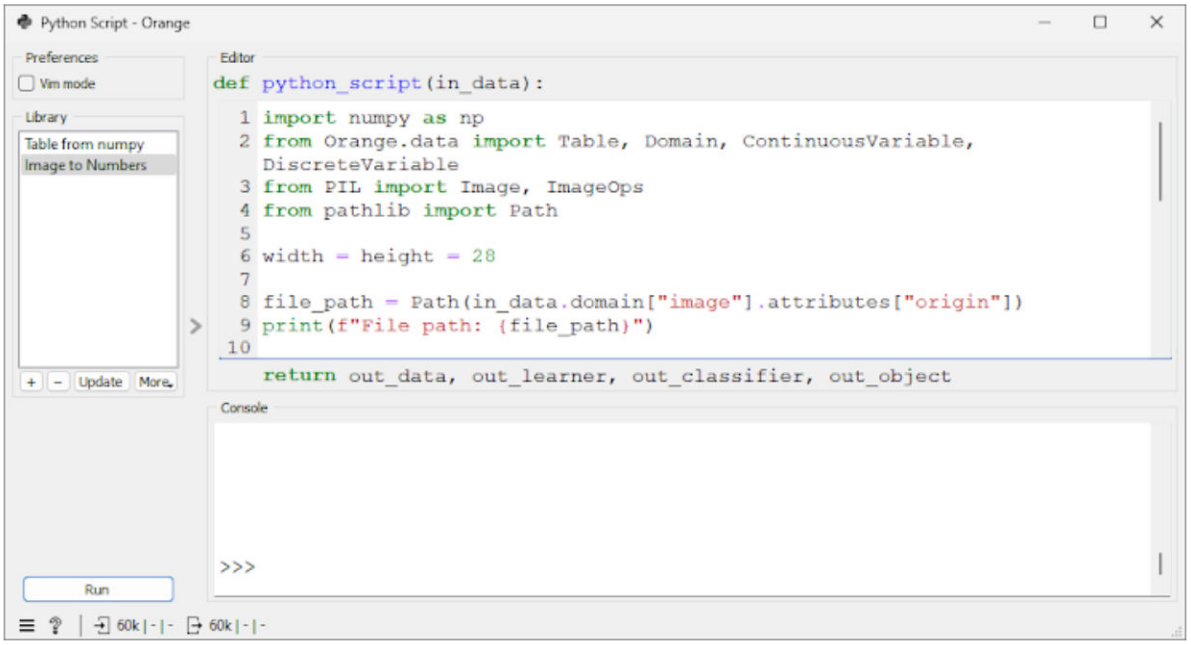
◯ Done

To convert images to pixel values, use the Python script from this Google Drive link. *

Copy the script and paste it into "Image to Numbers".

Save it by pressing Ctrl + S.

You should end up with this:



◯ Done

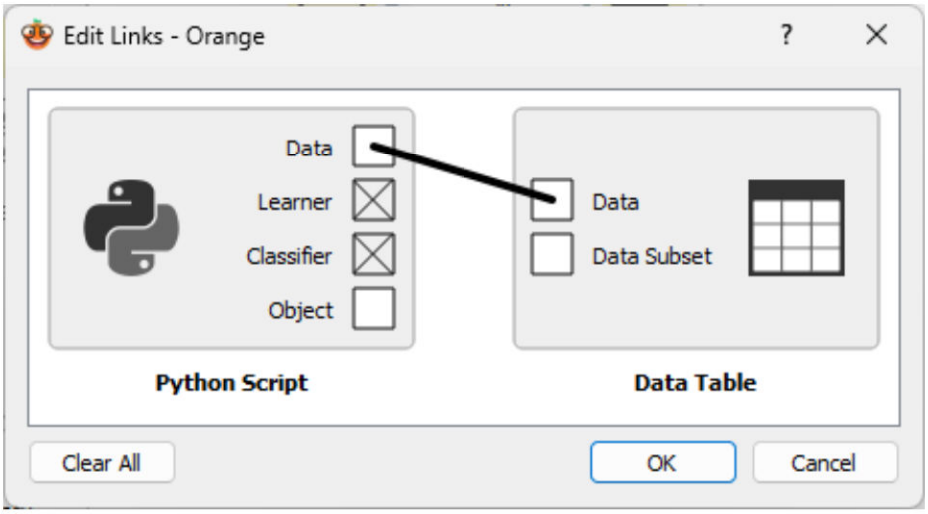
Click the "Run" button to execute the script. *

It may take some time to complete.

◯ Done

To view the numerical features of each image, drag and drop a "Data Table" widget. *

Edit the link between the "Python Script" widget and the "Data Table" widget as follows:



- [ ] Done

Rename the widget to "Image Data Table". *

- [ ] Done

Open the widget. *

You should see 784 numeric features with column names in the format x_{pixel_number}.

- [ ] Done

In the Info section, verify the following: *

- 60000 instances (no missing data)
- 784 features
- Target with 10 values
- 5 meta attributes

- ◯ Done

In the data table, what does "60000 instances" represent? *

- ◯ The number of pixels in each image
- ◯ The number of distinct labels
- ◯ The number of imported images

What does "784 features" represent? *

- ◯ The number of distinct labels
- ◯ The number of pixels in each image
- ◯ The number of imported images

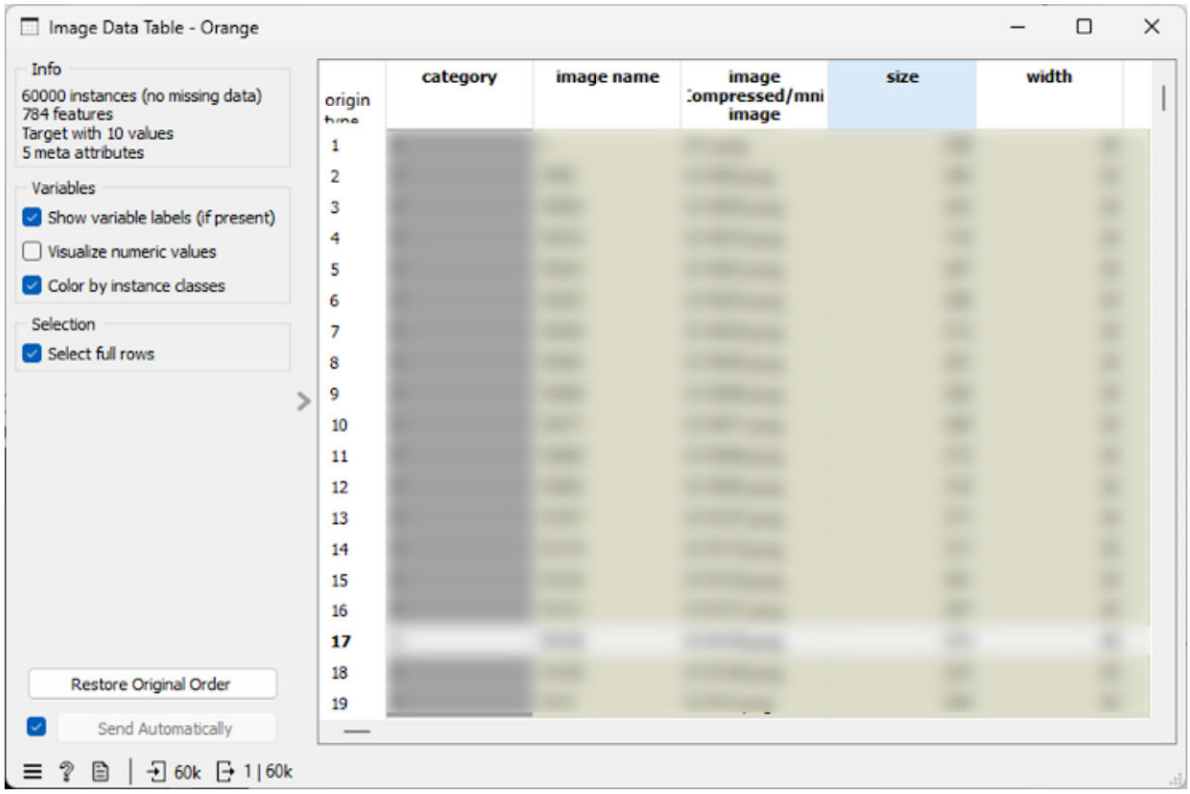
What does "Target with 10 values" represent? *

- ◯ The number of distinct labels
- ◯ The number of imported images
- ◯ The number of pixels in each image

Click "Restore Original Order". *

Take a screenshot of the "Image Data Table" window, showing at least the first 10 rows.

Upload it here.



Add file

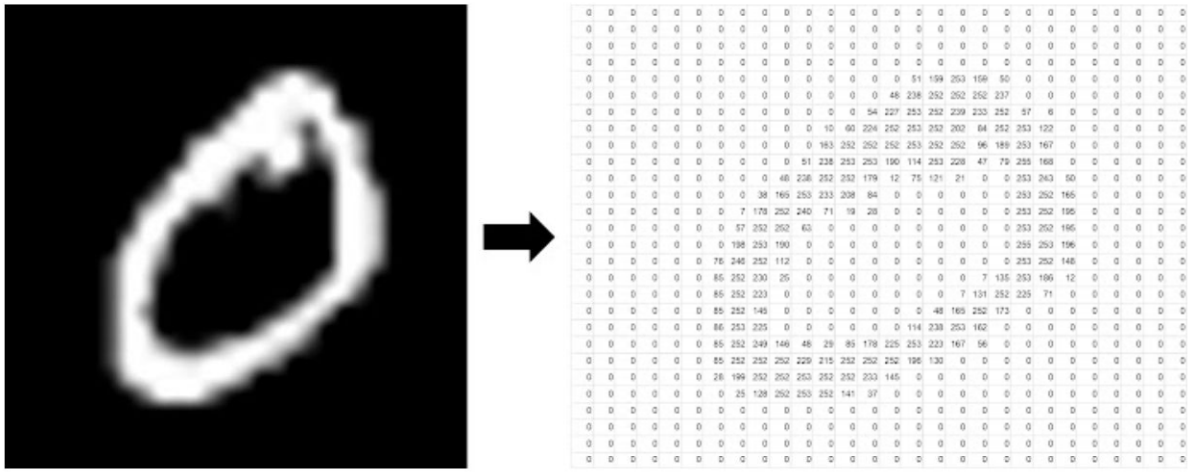
What does the script do?

The script converts a 28x28 image into a 2D array with 28 rows and 28 columns, where each value ranges from 0 to 255. *

The image below illustrates this conversion.

In the array, most values are 0 due to the black background, resembling the number zero, similar to the image.

This 2D array is then flattened into a 1D array for use as numerical features.



- ◯ I have read and understood.

Step 4: Split the Data

To reduce computation, we will use only 1,000 of the 60,000 images to train the Logistic Regression model.

Drag and drop a "Data Sampler" widget to split the data. *

Connect it to "Image Data Table".

- ◯ Done

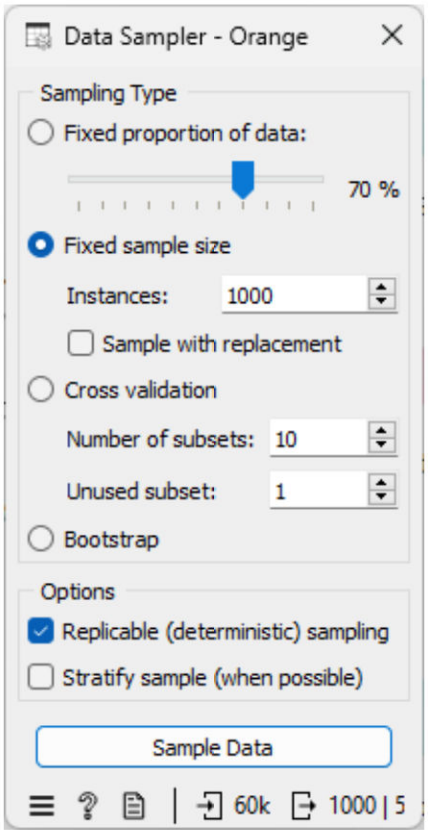
Open the widget. *

Set "Sampling Type" to "Fixed sample size", and enter 1000 for "Instances" (number of images for training).

Leave "Sample with replacement" unchecked.

In "Options", check "Replicable (deterministic) sampling".

You should end up with this:



- [ ] Done

Click the "Sample Data" afterwards. *

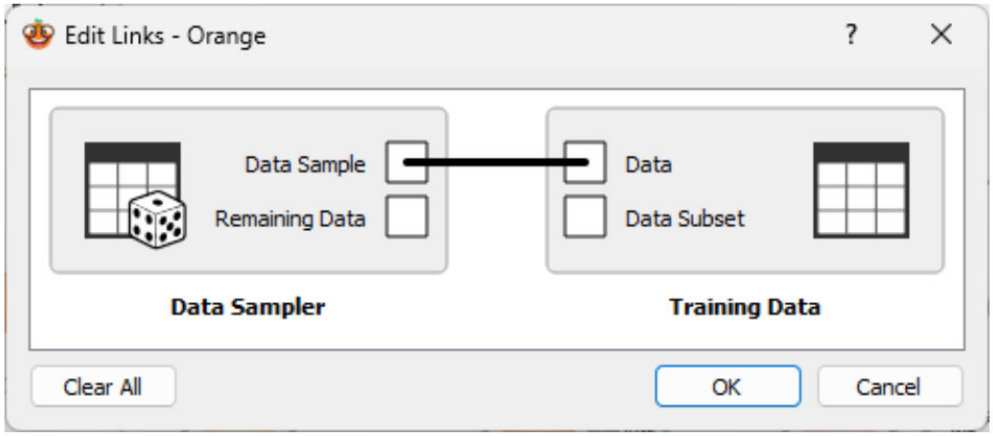
- [ ] Done

To view the training data, drag and drop a "Data Table" widget. *

Connect it to the "Data Sampler" widget and rename it as "Training Data".

- ◯ Done

Edit the link between the "Data Sampler" widget and the "Training Data" widget as follows: *



- ◯ Done

Open the "Training Data" widget. *

Click "Restore Original Order".

Take a screenshot of the "Training Data" window, showing at least the first 10 rows.

Upload it here.

Add file

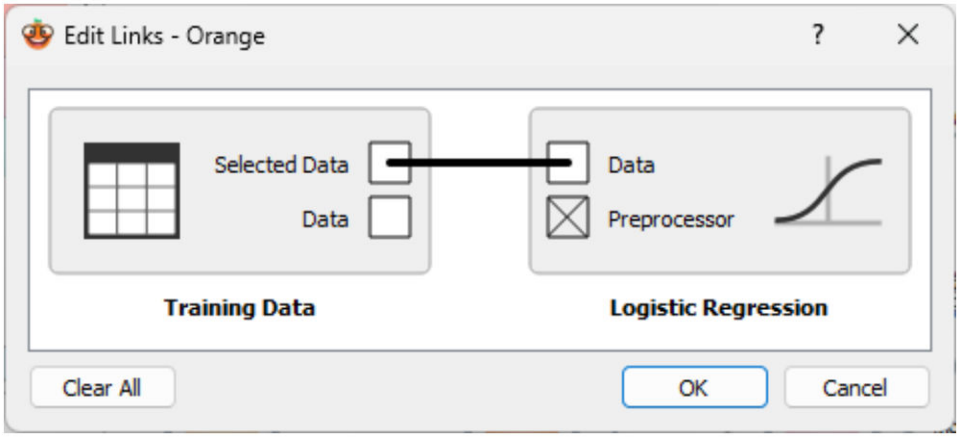
Step 5: Train the Model

Drag and drop a "Logistic Regression" widget. *

Connect it to "Training Data".

- ◯ Done

Edit the link between "Training Data" and the "Logistic Regression" widget as follows: *



- ◯ Done

Cross-validation is a technique used in machine learning to estimate a model's performance by evaluating it on different subsets of training data. [1] *

A widely used method of cross-validation is K-fold. You can learn more about it here.

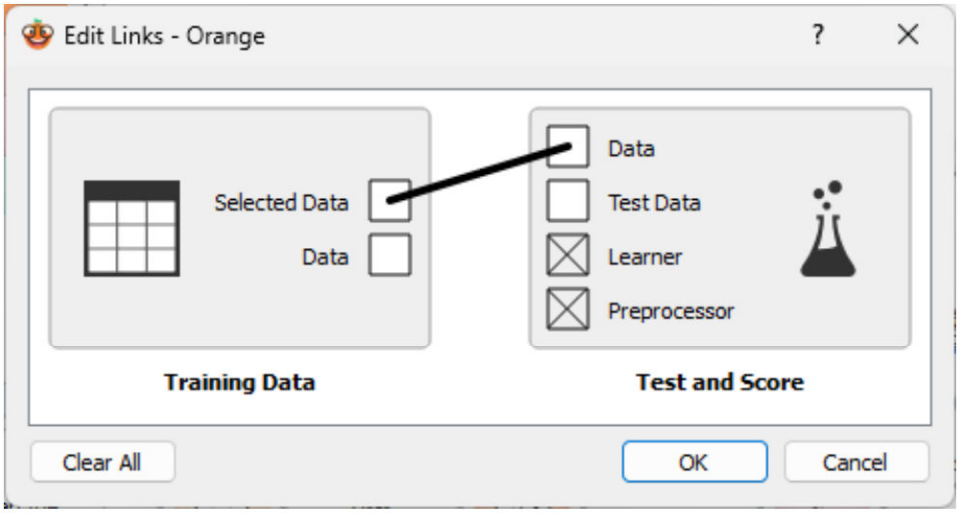
- ◯ I have read and understood.

To evaluate the performance of your Logistic Regression model during cross-validation, drag and drop a "Test and Score" widget. *

Connect it to "Training Data" and "Logistic Regression".

- ◯ Done

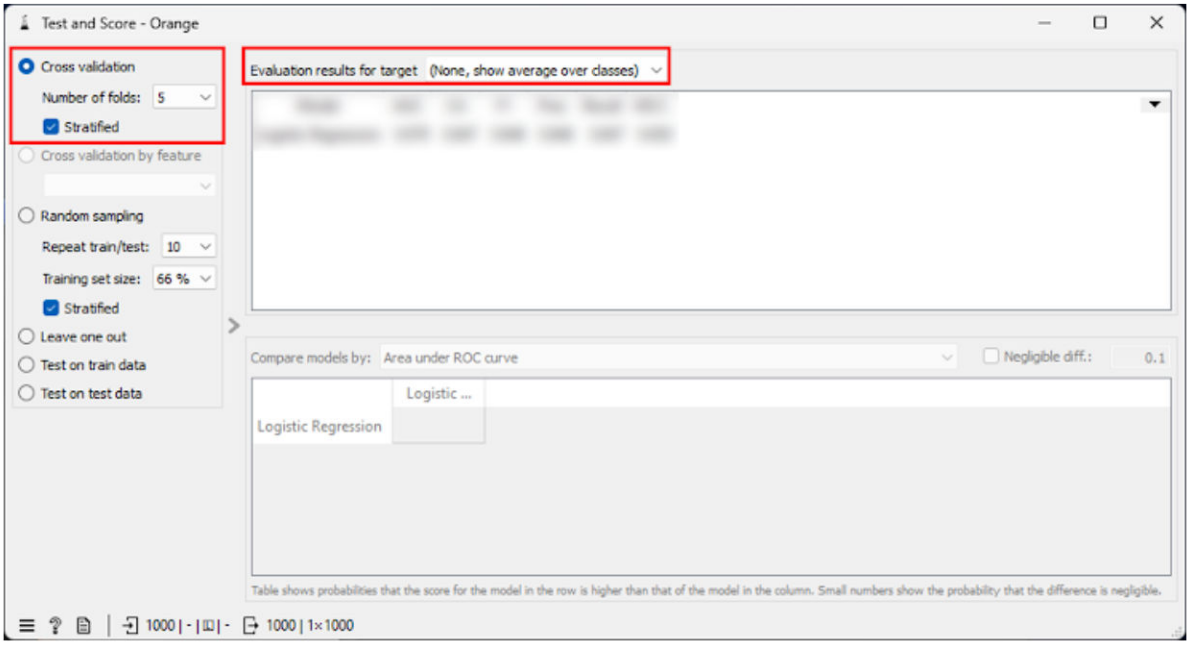
Edit the link between "Training Data" and the "Test and Score" widget as follows: *



◯ Done

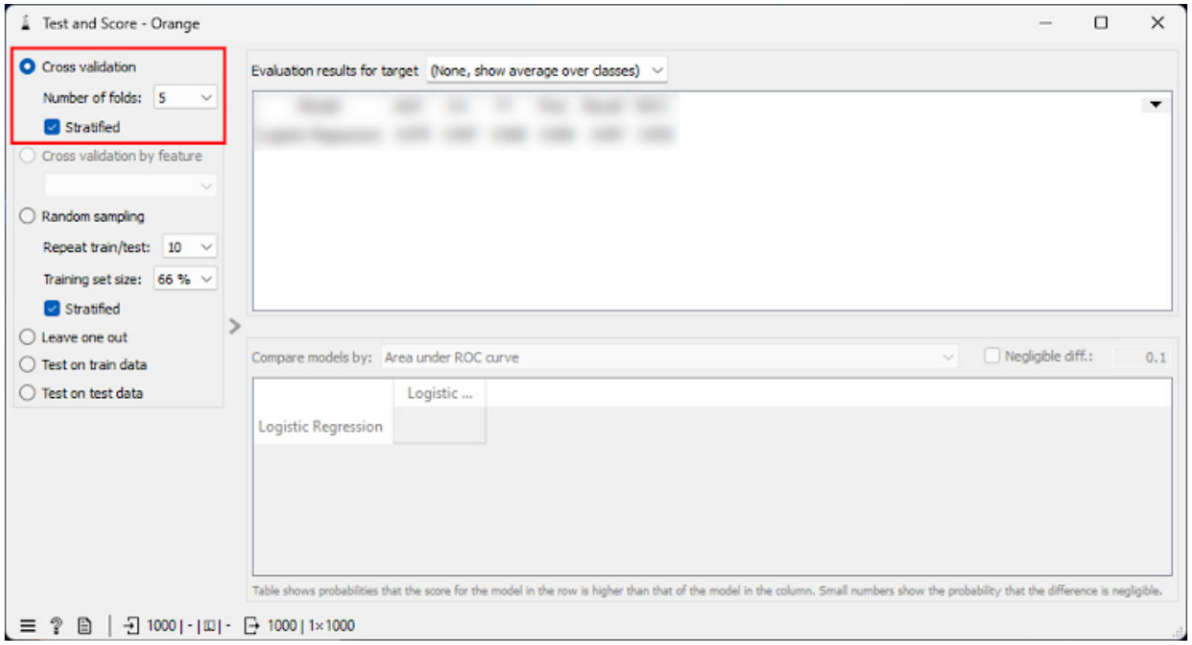
Open the "Test and Score" widget. *

Keep the default settings as follows:



◯ Done

What does this setting in the red box do? *



- It randomly splits the data into 66% training and 34% testing, repeating the process 10 times.
- It randomly splits the data into 5 groups, using one for evaluation and the others for training. This is repeated five times so each group is evaluated once.

Based on the model's performance in cross-validation, **what is its accuracy (CA)?** *

If you do not have these values, review your process.

- 0.830
- 0.846
- 0.847
- 0.979

Based on the model's performance in cross-validation, **what is its precision?** *

- ◯ 0.830
- ◯ 0.846
- ◯ 0.847
- ◯ 0.979

Based on the model's performance in cross-validation, **what is its recall?** *

- ◯ 0.830
- ◯ 0.846
- ◯ 0.847
- ◯ 0.979

Based on the model's performance in cross-validation, **what is its F1 score?** *

- ◯ 0.830
- ◯ 0.846
- ◯ 0.847
- ◯ 0.979

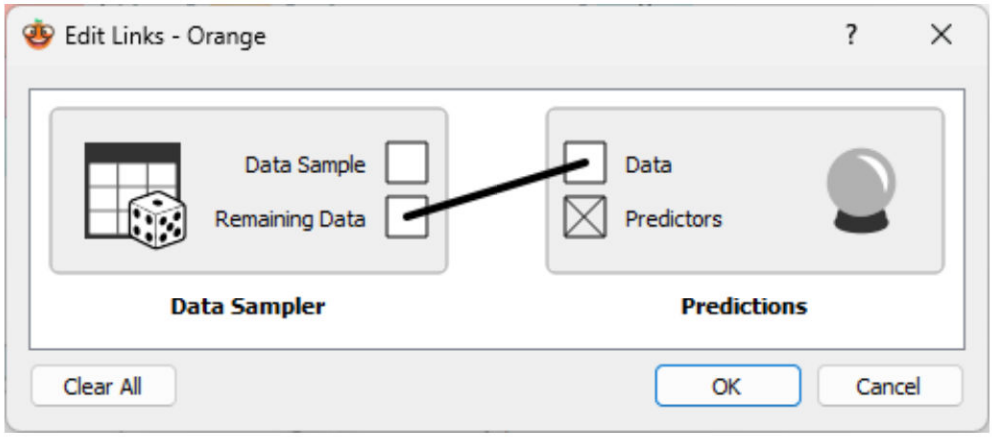
Step 6: Test the Model

To make predictions from your testing data, connect a "Predictions" widget. *

Connect it to the "Data Sampler" widget and "Logistic Regression".

- ◯ Done

Edit the link between the "Data Sampler" widget and the "Predictions" widget as follows: *



- ◯ Done

Open the "Predictions" widget. *

- ◯ Done

What is the **model's accuracy (CA)** on the test data? *

- ◯ -2.645
- ◯ 0.834
- ◯ 0.850
- ◯ 0.851

What is the **model's precision** on the test data? *

- ◯ -2.645
- ◯ 0.834
- ◯ 0.850
- ◯ 0.851

What is the **model's recall** on the test data? *

- -2.645
- 0.834
- 0.850
- 0.851

What is the **model's F1 score** on the test data? *

- -2.645
- 0.834
- 0.850
- 0.851

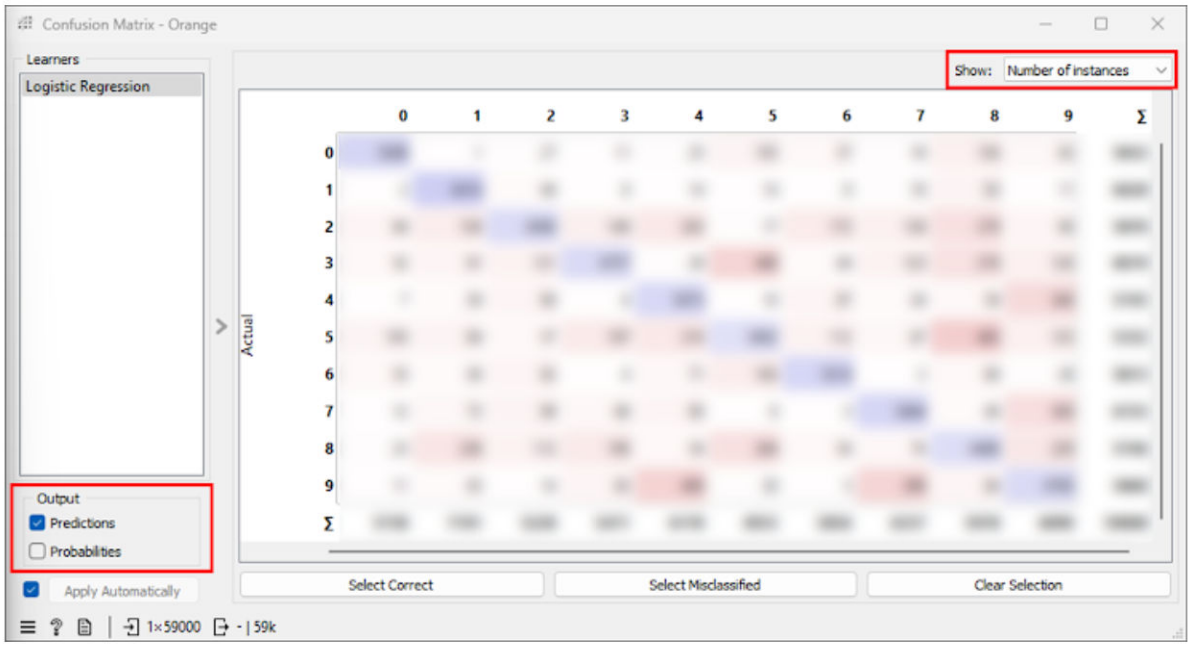
To display the model's predictions for the test images in a table, drag and drop a "Confusion Matrix" widget. *

Connect it to the "Predictions" widget.

- Done

Open the widget. *

Keep the default settings as follows:



◯ Done

Check the diagonal of the confusion matrix to see the number of instances the model correctly classified. *

What is the total number of images that were correctly identified?

Hint: Dividing this by the total test images should match the accuracy in the "Predictions" widget.

Your answer

Which label (number) is most often misidentified? *

Hint: Look for the reddest cell in the confusion matrix and check the "Actual" label for that row.

Choose

Take a screenshot of the "Confusion Matrix" window and upload it here. *

Example:

Add file

To see which images were classified correctly or incorrectly, drag and drop an "Image Viewer" widget. *

Connect it to the "Confusion Matrix" widget.

Done

Open both the "Confusion Matrix" and "Image Viewer" widgets. *

Examine each cell in the confusion matrix to determine which images were classified correctly or incorrectly.

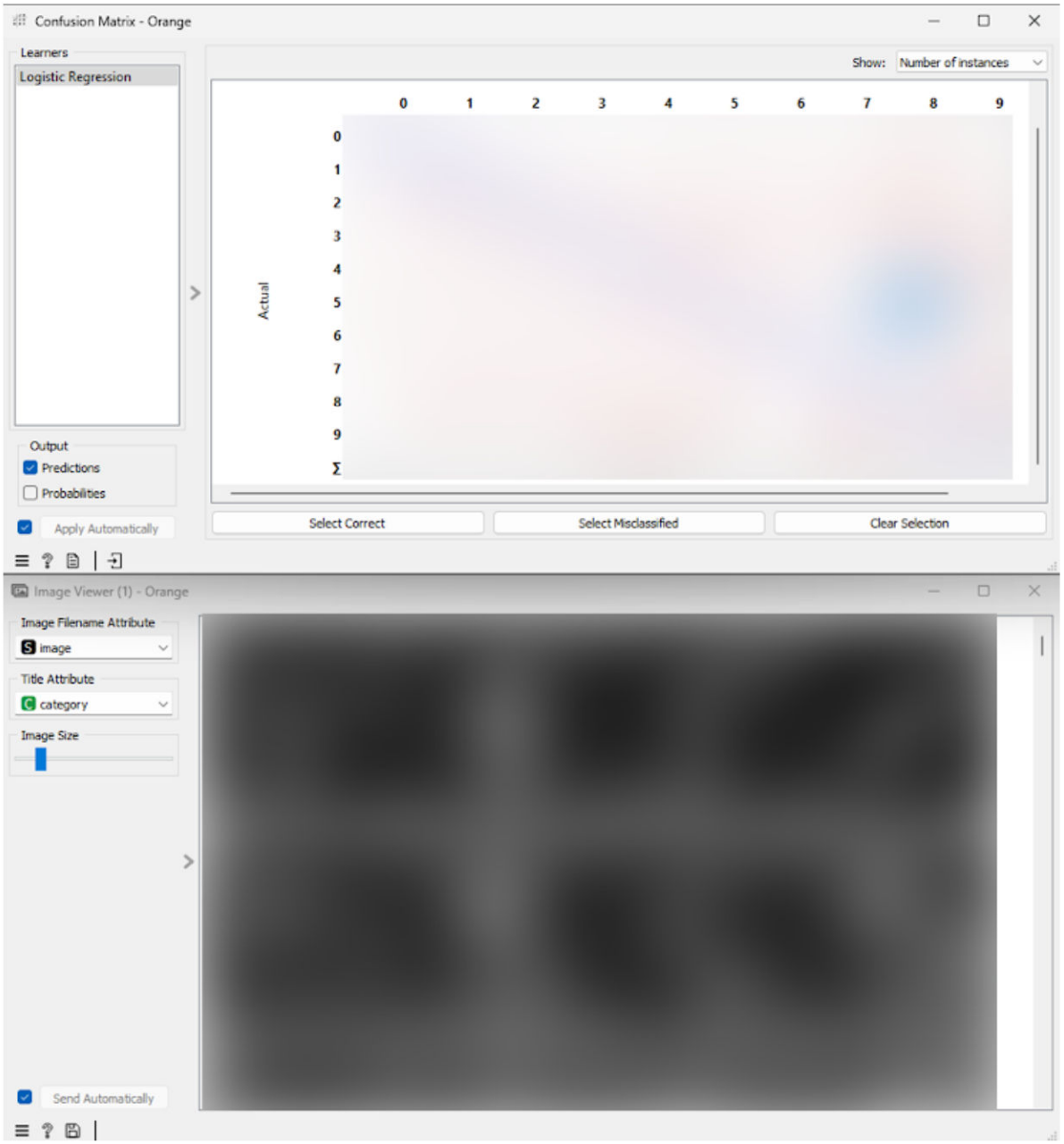
- ◯ Done

Take a screenshot of the "Confusion Matrix" and "Image Viewer" windows, selecting the cell with the most misclassifications (the reddest one) in the confusion matrix. *

Adjust the Image Size slider so that the first 8 photos are visible.

Upload it here.

Example:



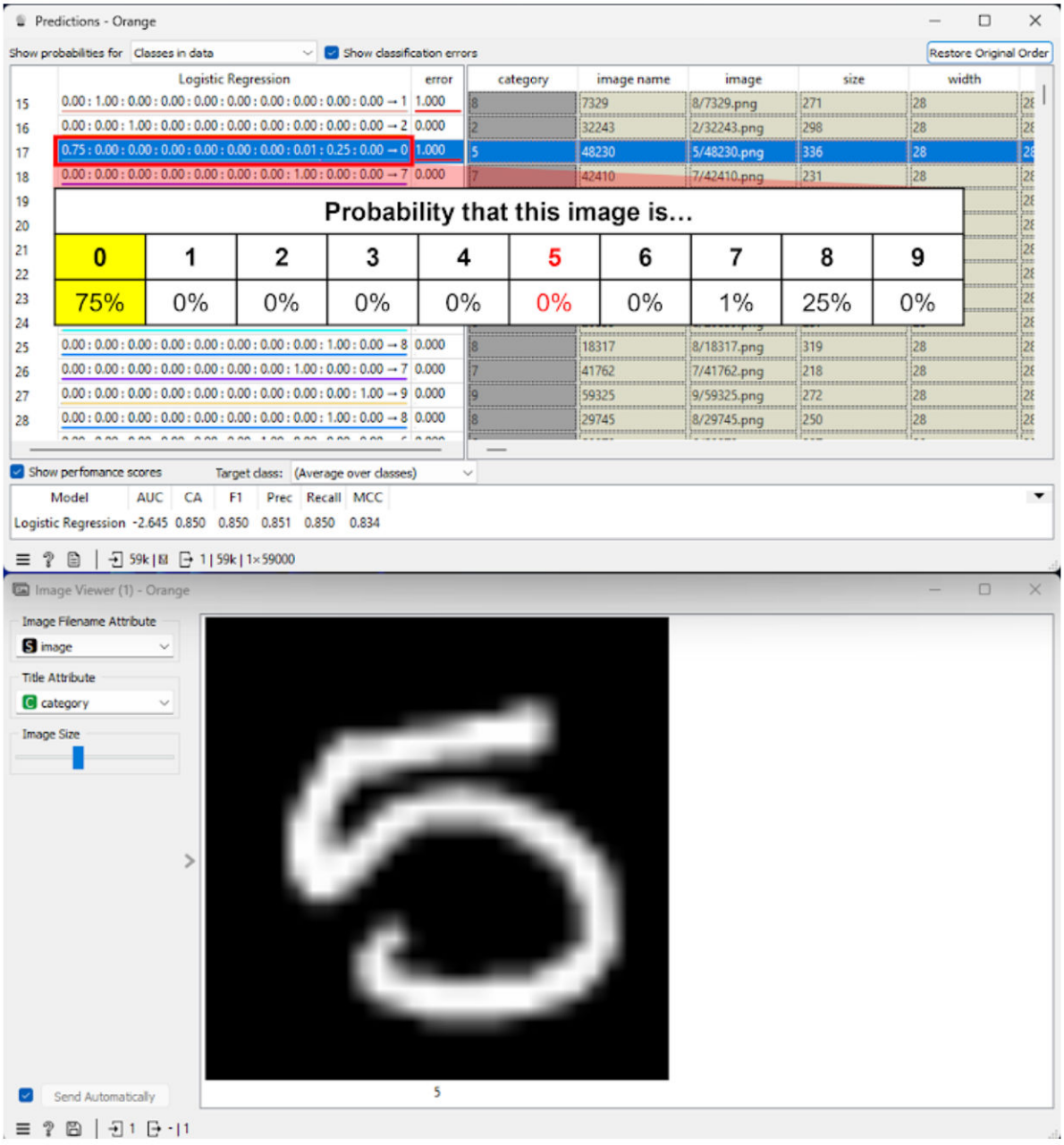
Add file

What do the numbers in the "Predictions" widget represent?

The numbers in the Logistic Regression column show the probabilities for each digit (0-9). *

The digit with the highest probability is chosen for classification.

For example, the model predicts that the image in row 17 most likely contains a zero, with some probability for 7 and 8, but none for 5, which is the correct label.

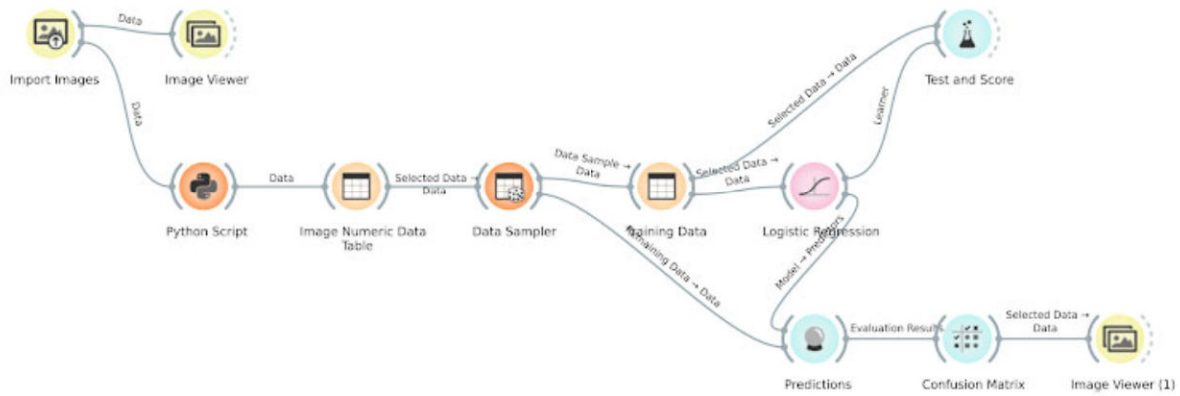


I have read and understood.

Step 7: Review your Workflow

Verify that your workflow includes the widgets and connections shown in the photo below. *

If it does not, review your process.



◯ Done

If your result matches the one above, save your workflow. *

Compress it into a ZIP file using the same name as your original workflow file (e.g., Rey_4A_Lab_Exercise_2).

Upload it here.

Add file

Step 8: Deploy the Model

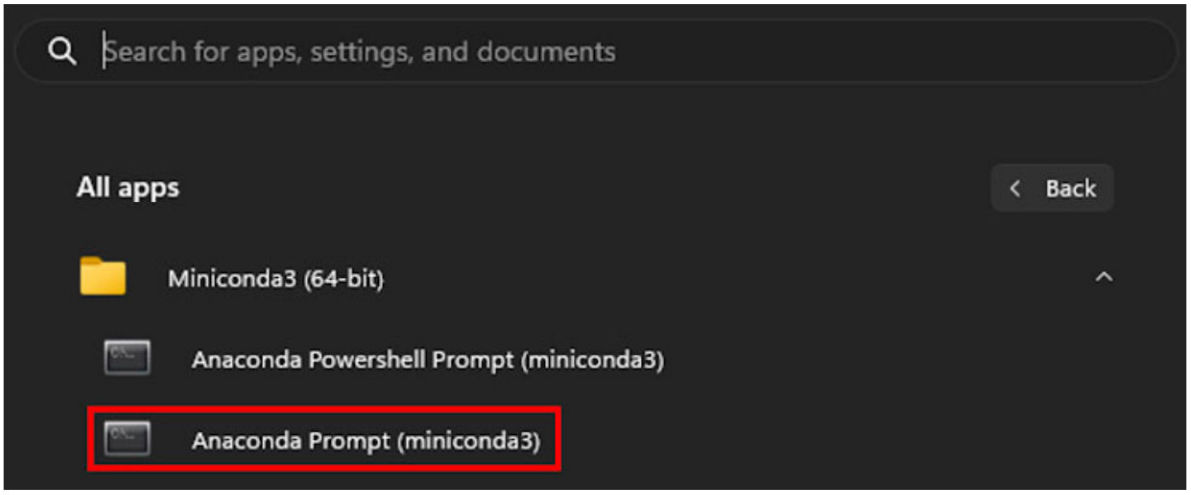
Let's deploy the model using Streamlit, an open-source Python framework for creating dynamic data apps with minimal code.

For more details, refer to the Streamlit documentation.

An internet connection is required.

Since Orange requires Miniconda, open "Anaconda Prompt (miniconda3)" from your Start menu. *

Example:



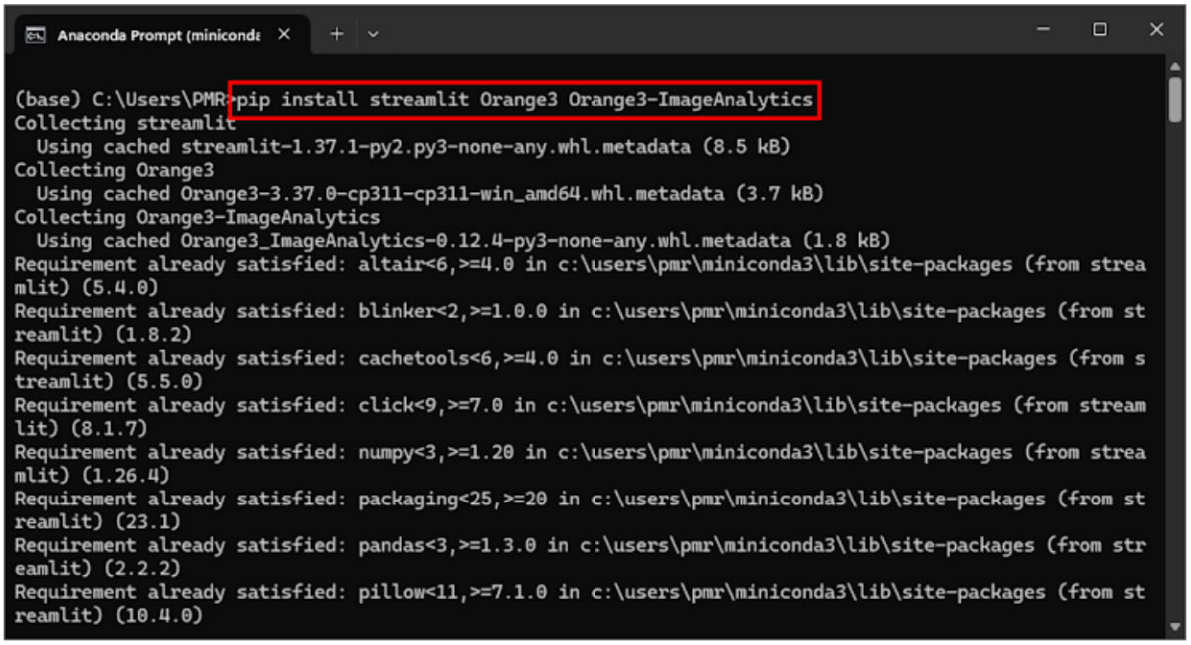
- [ ] Done

Once open, type: *

pip install streamlit Orange3 Orange3-ImageAnalytics

Press Enter to download Streamlit, Orange3, and Orange3 Image Analytics and their dependencies.

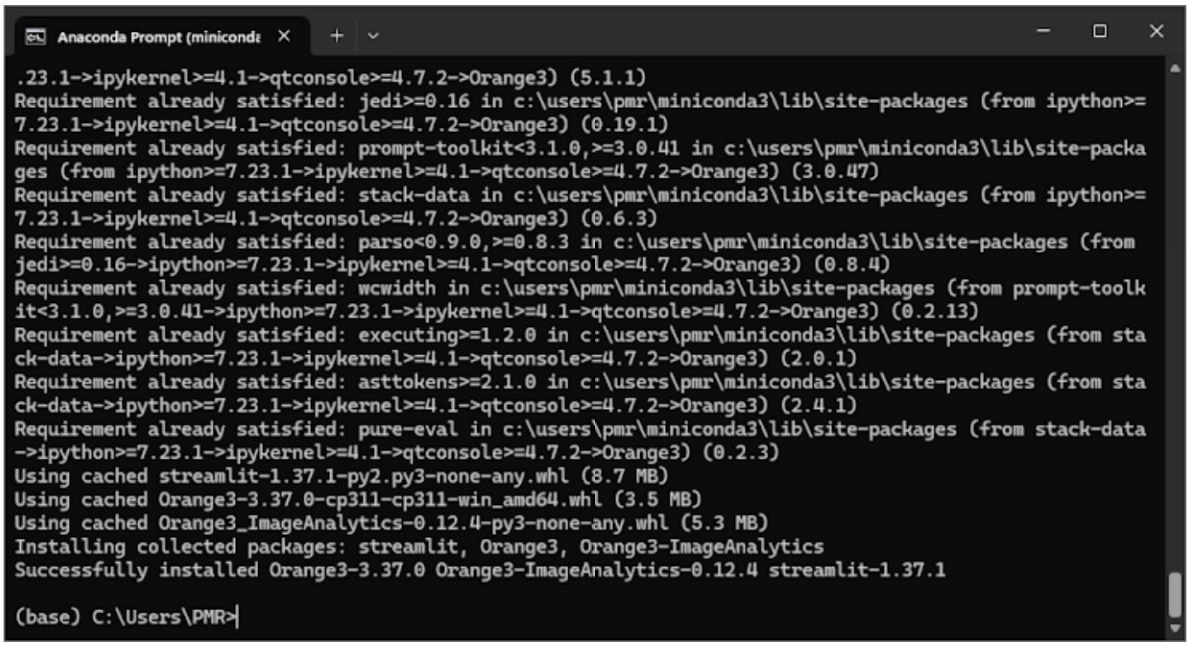
Example:



◯ Done

It may take some time to complete. *

You should end up with this:



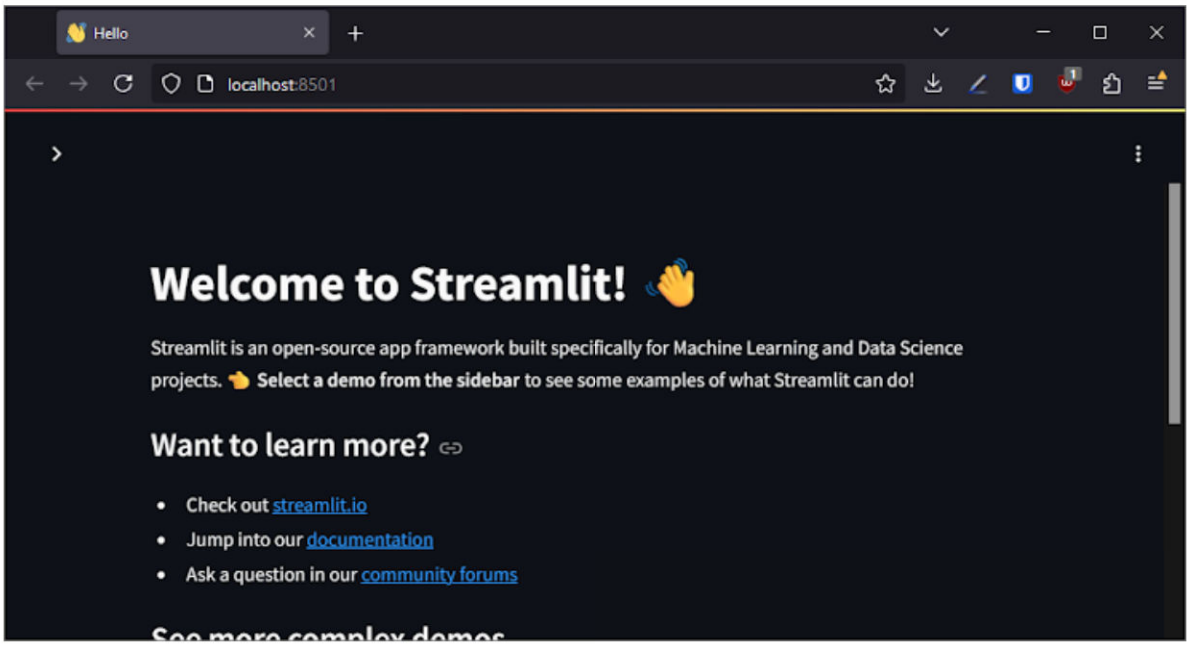
◯ Done

To check if it's working, type: *

streamlit hello

This will launch a demo app in your browser.

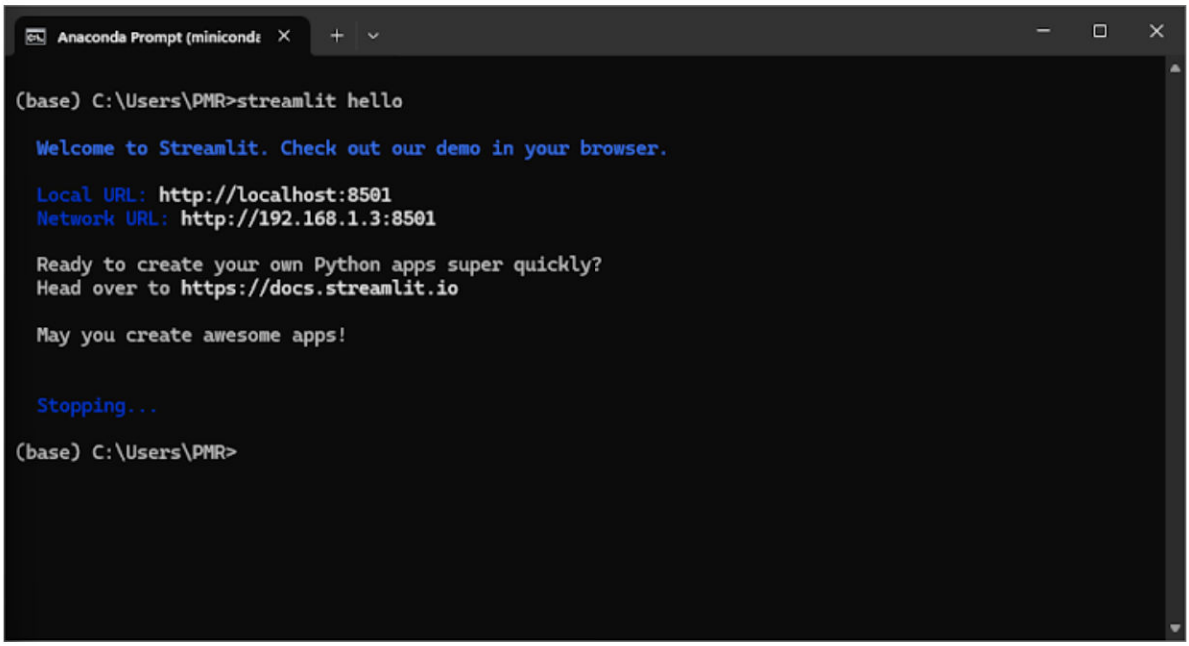
You should end up with this:



◯ Done

To stop the app, press Ctrl + C in the terminal. *

You should end up with this:



- [ ] Done

Return to your workflow. *

To save your model, drag and drop a "Save Model" widget.

Connect it to "Logistic Regression".

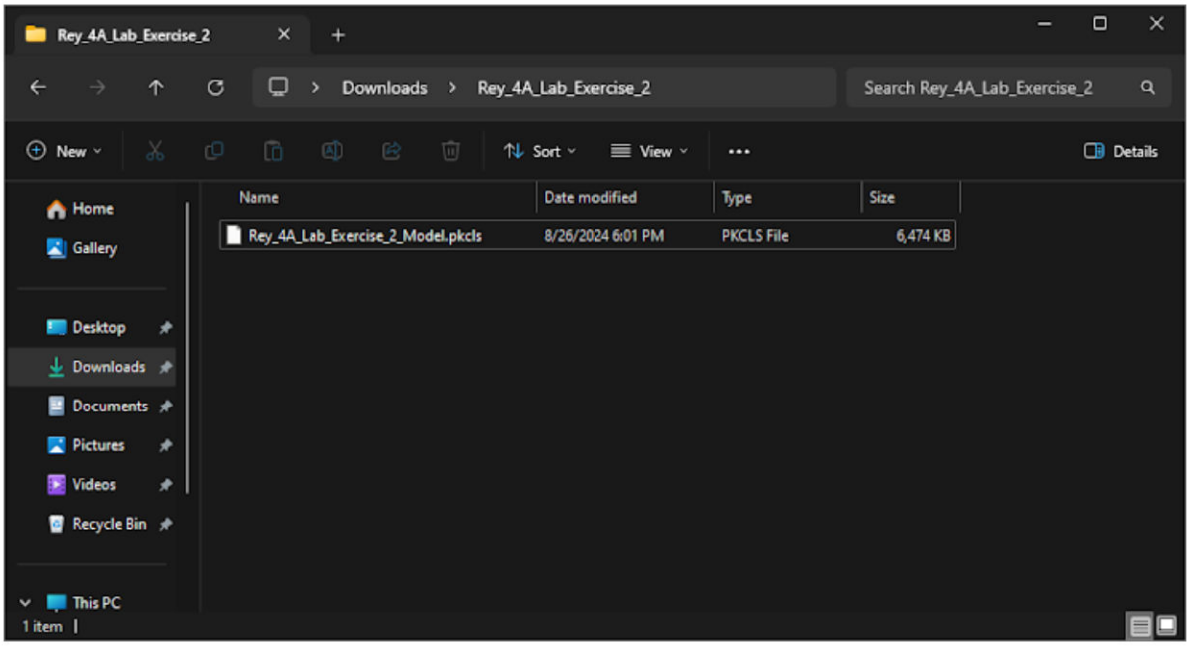
- [ ] Done

Open the widget. *

Click 'Save' and save the model **in a new folder with the same name as your workflow file.**

Name the model file using this format:
[Last_Name]_[Section]_Lab_Exercise_2_Model (e.g., Rey_4A_Lab_Exercise_2_Model.pkcls).

You should end up with this:



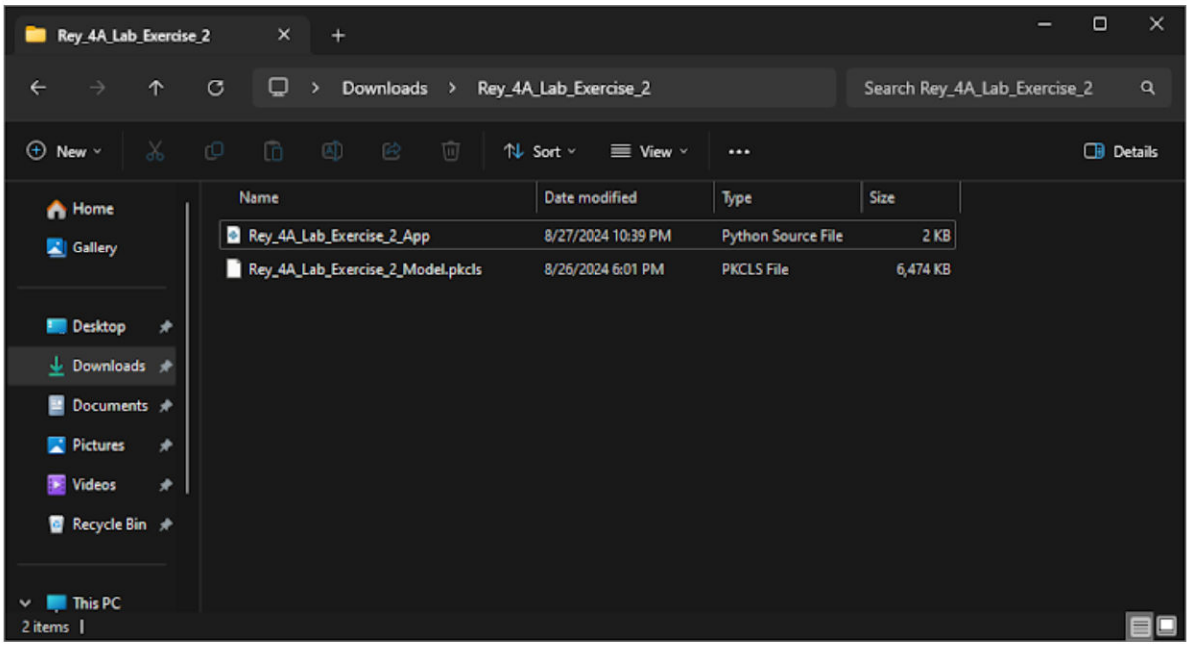
◯ Done

Download the script here and name the file using this format: *
[Last_Name]_[Section]_Lab_Exercise_2_App (e.g., Rey_4A_Lab_Exercise_2_App.py).

◯ Done

Save it in the same folder as your model. *

You should end up with this:



- ◯ Done

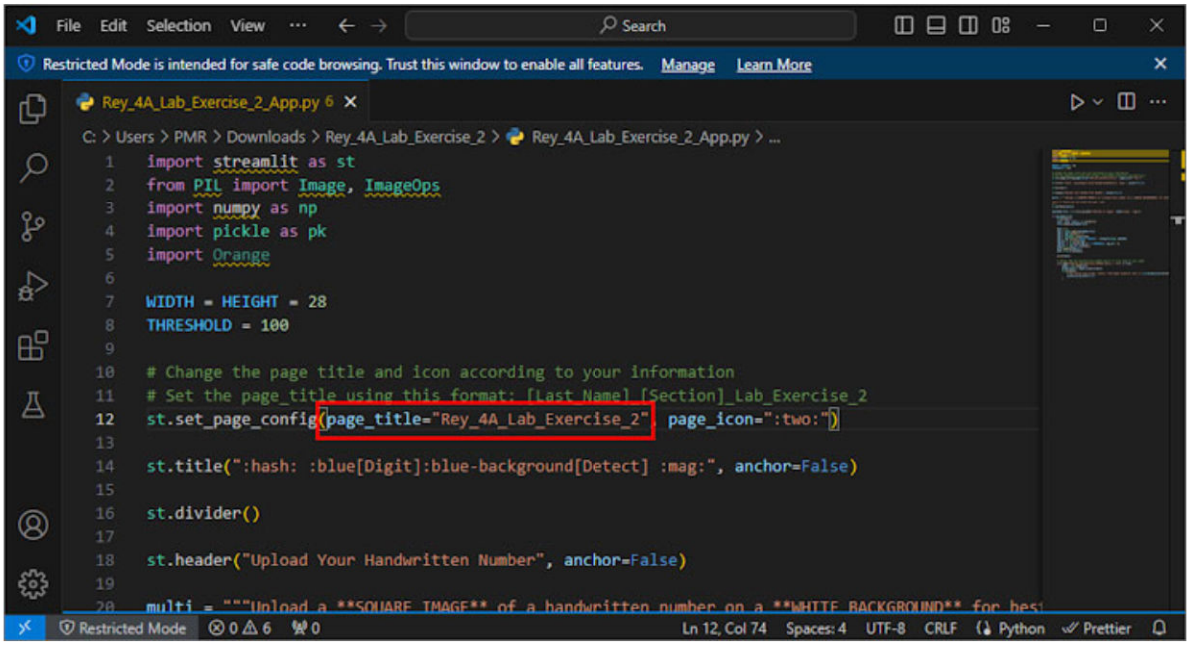
Open the Python file with your favorite text editor. *

- ◯ Done

In line 12 of the code, set the page_title in the following format: *

[Last_Name]_[Section]_Lab_Exercise_2

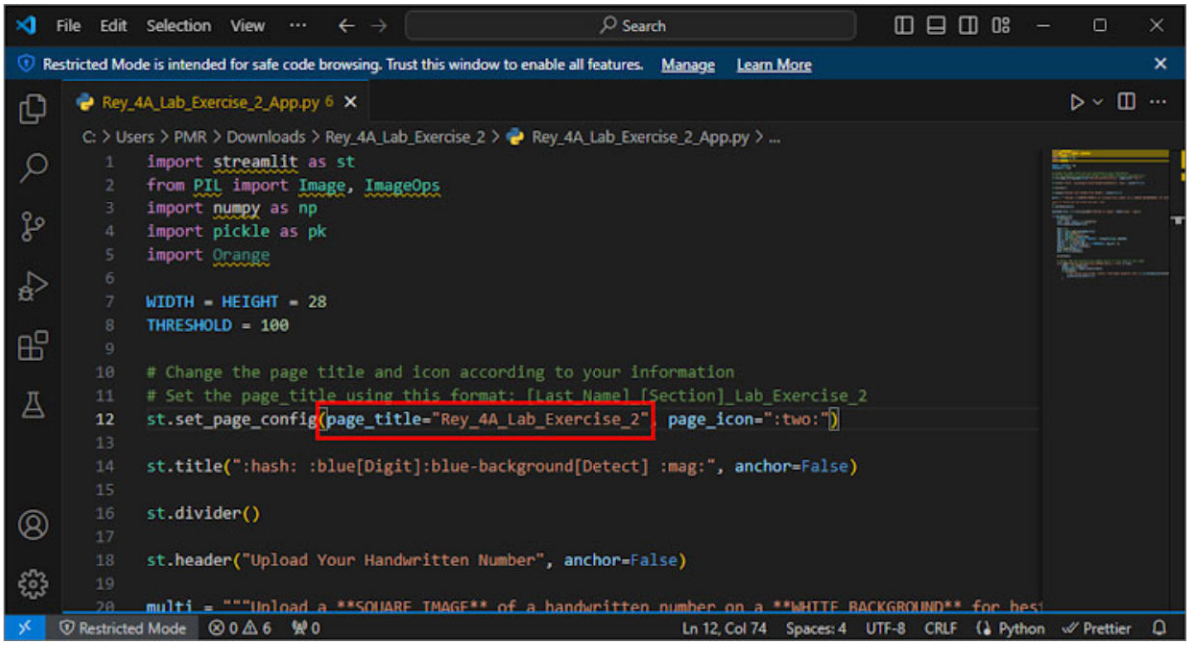
Example:



◯ Done

In line 46 of the code, replace "Rey_4A_Lab_Exercise_2_Model.pkcls" with the name of your model file. *

Example:



Done

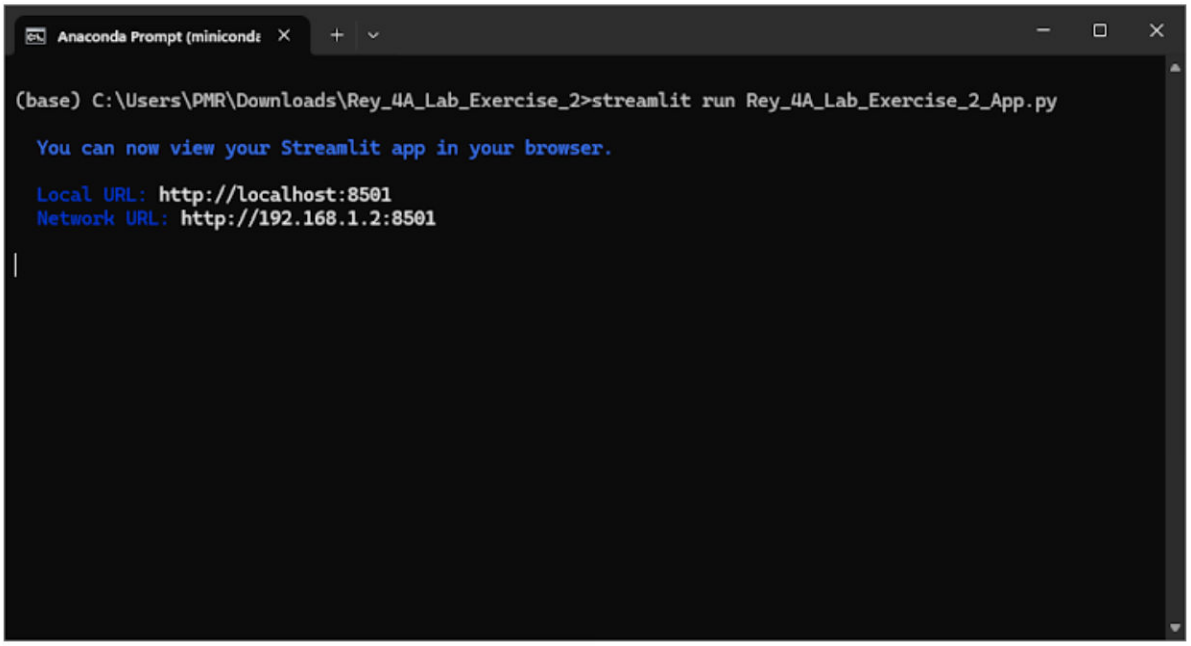
Save the code. *

Then, return to the "Anaconda Prompt (miniconda3)" and type:

streamlit run [Name of your Python file].py

This will open your app in the browser.

Example:



◯ Done

Follow the instructions in the app. *

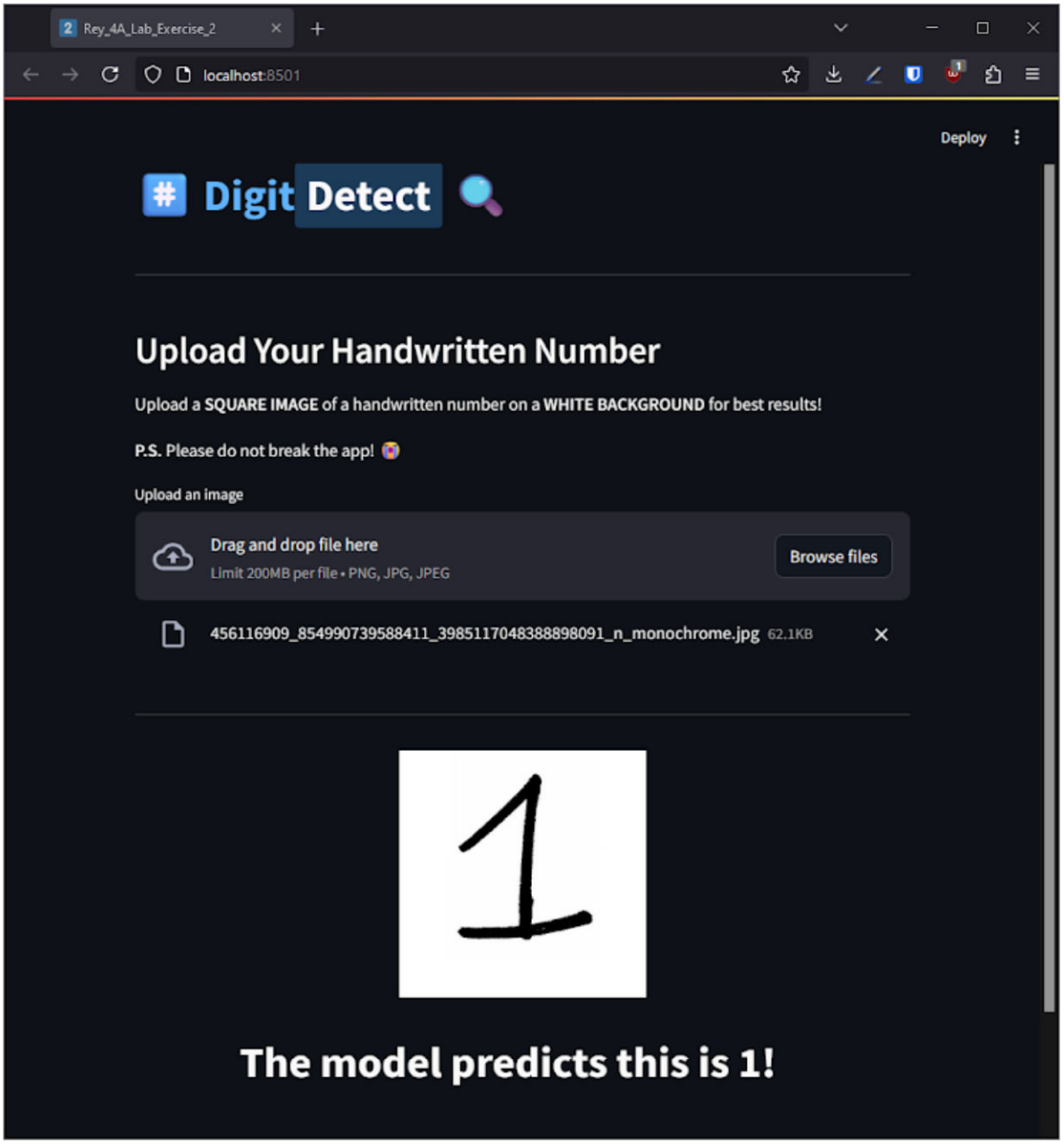
Use the **last digit of your student number** as the target for the model to predict.

For example, if your student number ends in 1, upload an image of the digit 1 and see if the model can recognize it. Keep trying until it gets it right.

◯ Done

Take a screenshot of your browser showing the model correctly predicting your image, then upload it here. *

Example:



Add file

Upload your model here (e.g., Rey_4A_Lab_Exercise_2_Model.pckls). *

This will be checked for uniqueness.

Add file

Get link

Never submit passwords through Google Forms.

This form was created inside of Marinduque State College.
Does this form look suspicious? Report

Google Forms