

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.

*

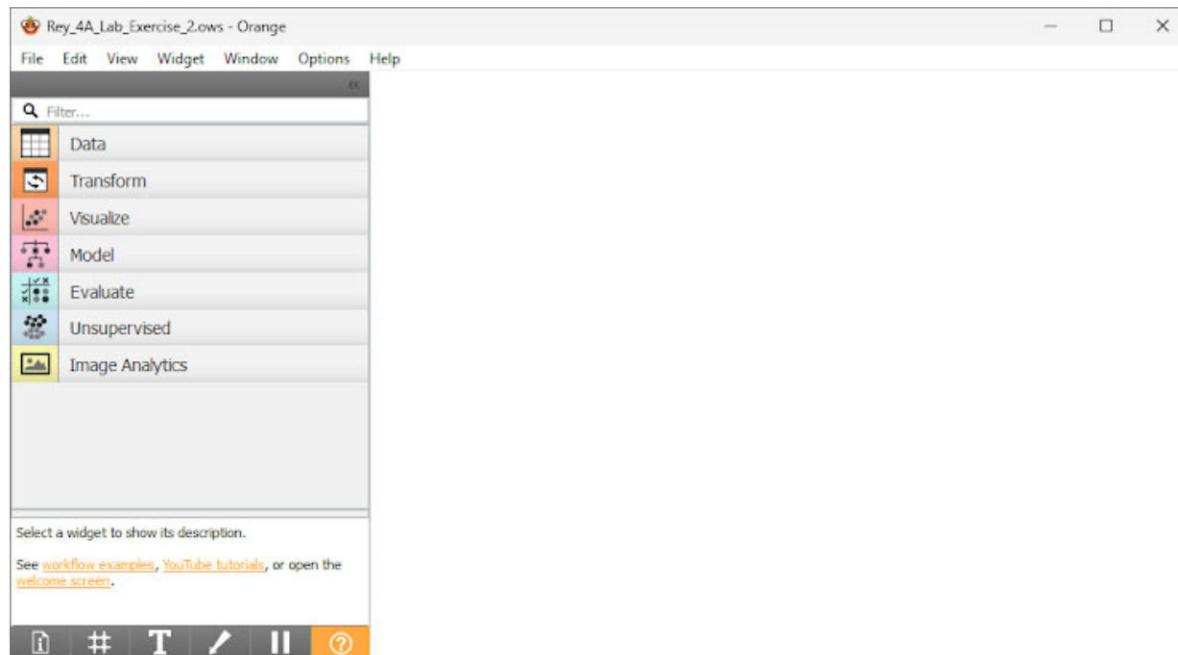
Choose an accessible directory and name the file using this format:

[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.

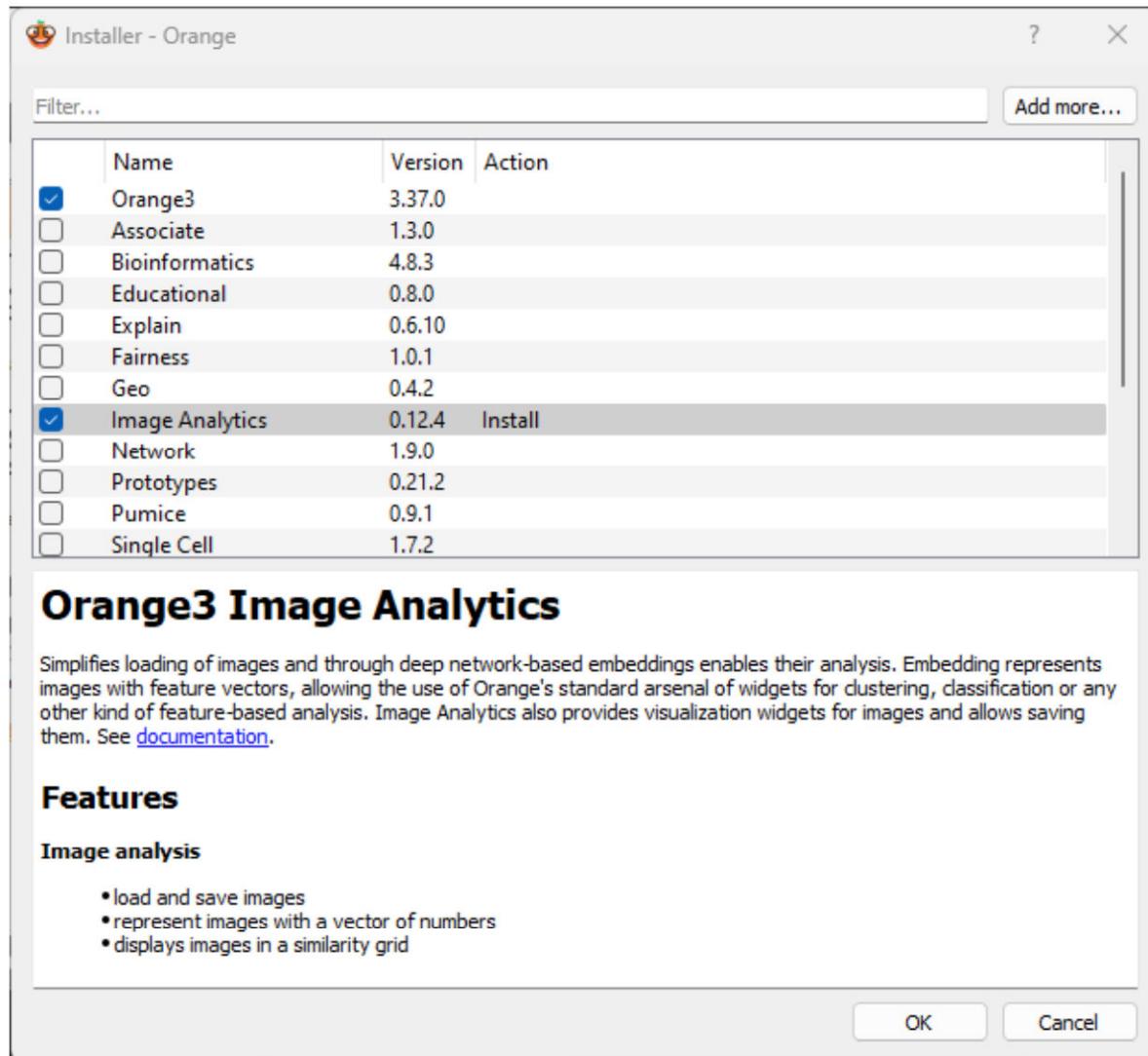
An internet connection is required.

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:



The screenshot shows the 'Installer - Orange' window. At the top, there is a toolbar with a magnifying glass icon, a question mark icon, and a close button. Below the toolbar is a search bar labeled 'Filter...' and a 'Add more...' button. The main area is a table with three columns: 'Name', 'Version', and 'Action'. The 'Action' column contains status indicators like 'Install' and progress bars. The 'Image Analytics' row has a checked checkbox in the 'Name' column, indicating it is selected for installation. The table data is as follows:

Name	Version	Action
Orange3	3.37.0	
Associate	1.3.0	
Bioinformatics	4.8.3	
Educational	0.8.0	
Explain	0.6.10	
Fairness	1.0.1	
Geo	0.4.2	
Image Analytics	0.12.4	Install
Network	1.9.0	
Prototypes	0.21.2	
Pumice	0.9.1	
Single Cell	1.7.2	

Orange3 Image Analytics

Simplifies loading of images and through deep network-based embeddings enables their analysis. Embedding represents images with feature vectors, allowing the use of Orange's standard arsenal of widgets for clustering, classification or any other kind of feature-based analysis. Image Analytics also provides visualization widgets for images and allows saving them. See [documentation](#).

Features

Image analysis

- load and save images
- represent images with a vector of numbers
- displays images in a similarity grid

OK Cancel

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. *

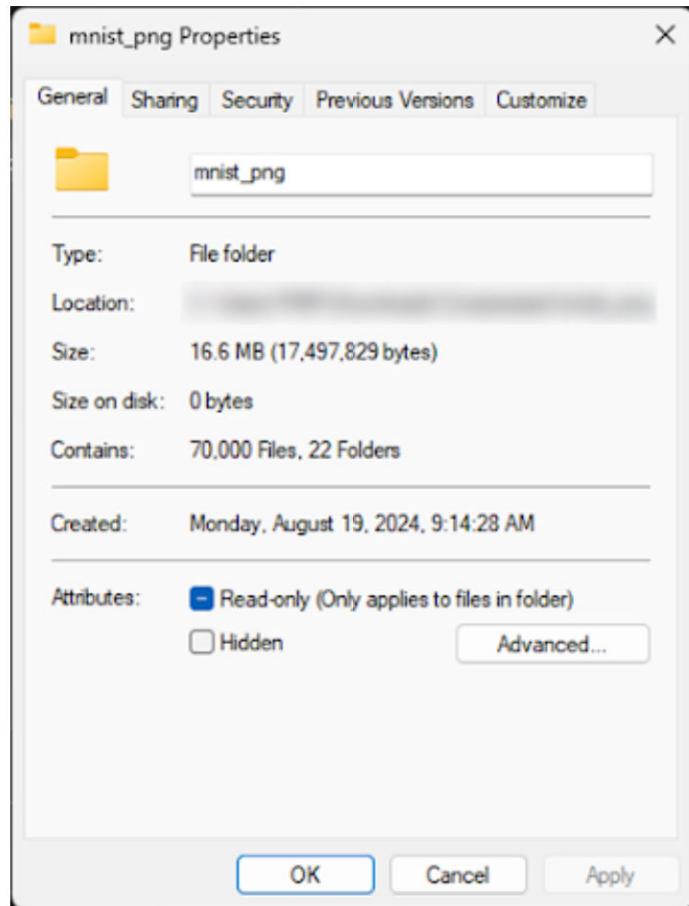
You will get a folder named "mnist_png".

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. *

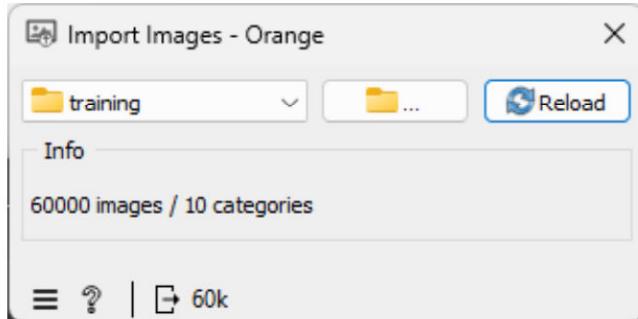
Done

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.

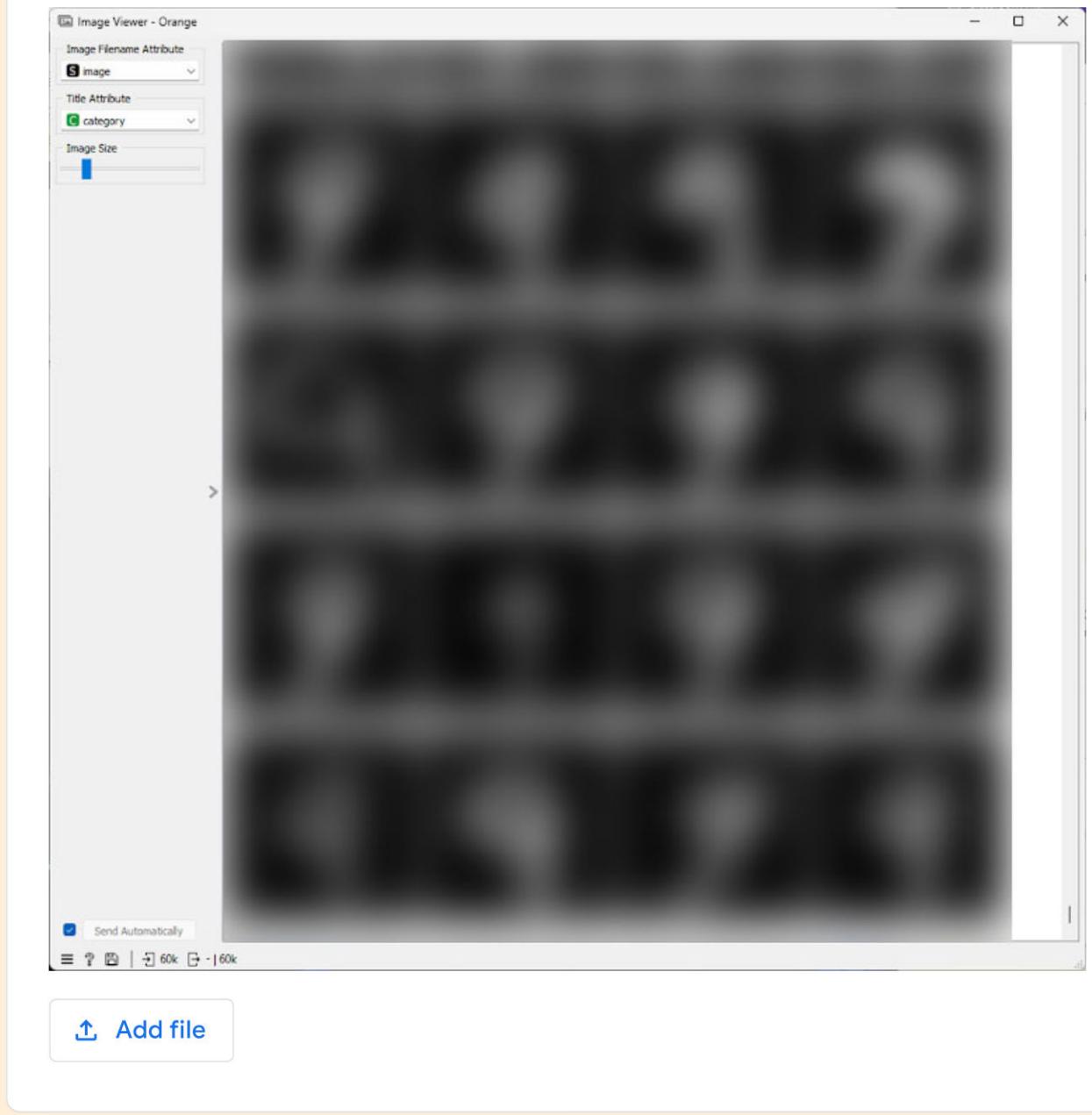
Done

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:



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".

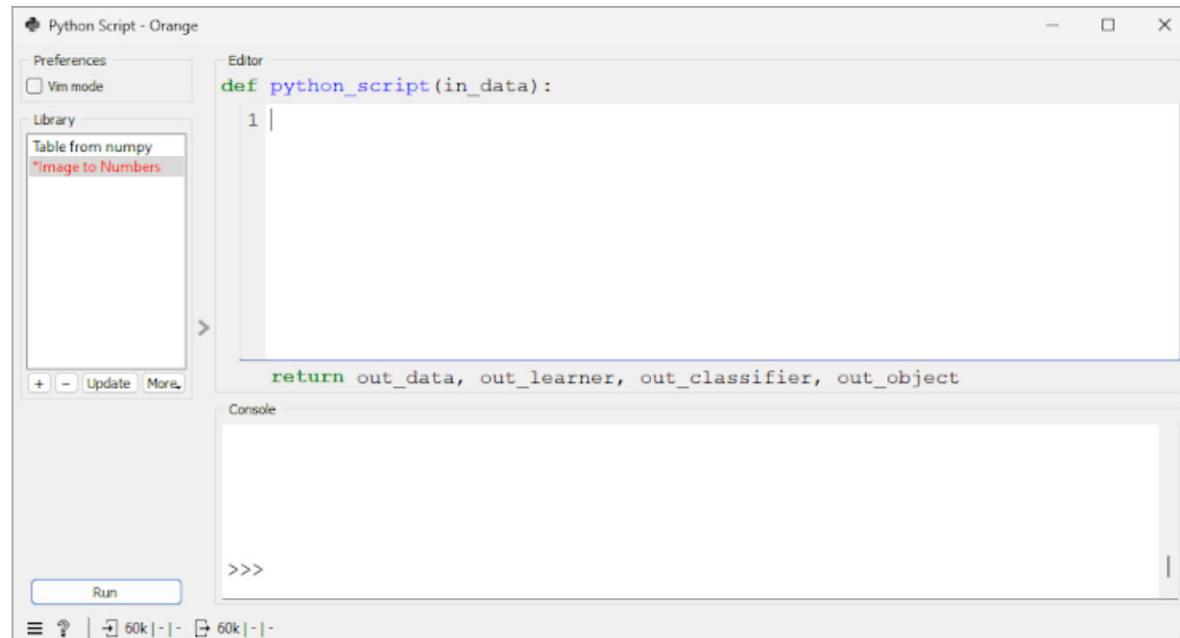
Done

Select "Image to Numbers".

*

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

You should end up with this:



The screenshot shows the Orange Python Script interface. The title bar says "Python Script - Orange". The left sidebar has "Preferences" and "Vim mode" checkboxes. The "Library" section contains "Table from numpy" and "Image to Numbers", with "Image to Numbers" highlighted in red. Below the library are buttons for "+", "-", "Update", and "More...". The main area is the "Editor" with the following code:

```
def python_script(in_data):
    1 |
    |
    return out_data, out_learner, out_classifier, out_object
```

Below the editor is the "Console" window, which is currently empty. At the bottom are standard window controls and status icons.

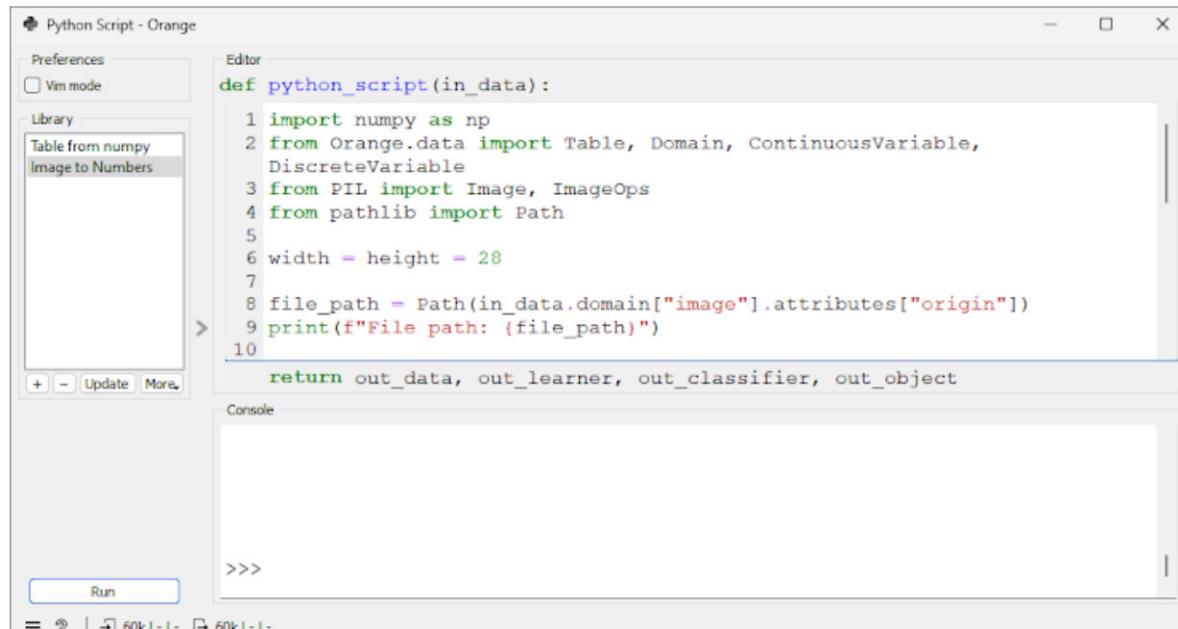
Done

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:



The screenshot shows the Orange Python Script interface. The 'Editor' tab contains the following Python code:

```
def python_script(in_data):
    1 import numpy as np
    2 from Orange.data import Table, Domain, ContinuousVariable,
        DiscreteVariable
    3 from PIL import Image, ImageOps
    4 from pathlib import Path
    5
    6 width = height = 28
    7
    8 file_path = Path(in_data.domain["image"].attributes["origin"])
    9 print(f"File path: {file_path}")
    10
    return out_data, out_learner, out_classifier, out_object
```

The 'Library' panel on the left lists 'Table from numpy' and 'Image to Numbers'. The 'Console' tab at the bottom shows an empty prompt: '>>>'. A 'Run' button is visible at the bottom left of the editor area.

Done

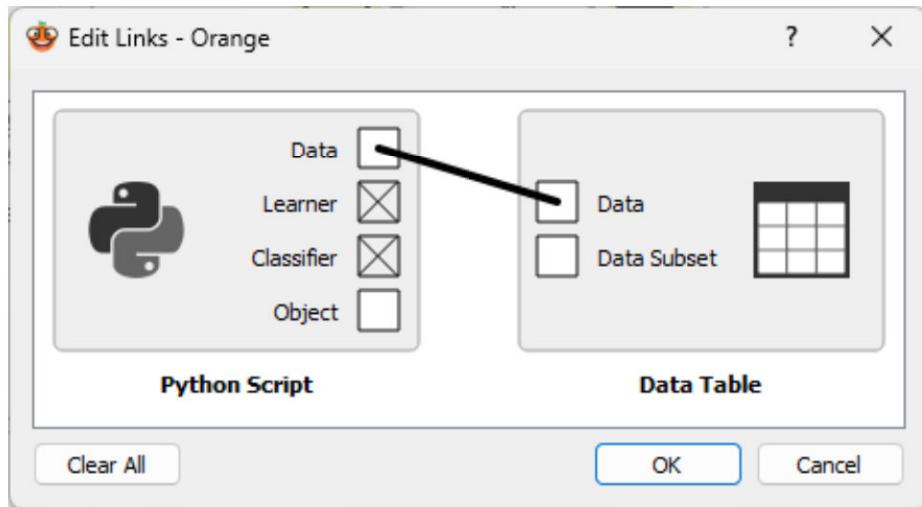
Click the "Run" button to execute the script. *

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.

The screenshot shows the 'Image Data Table - Orange' window. On the left, there's a sidebar with 'Info' (60000 instances, 784 features, Target with 10 values, 5 meta attributes), 'Variables' (checkboxes for Show variable labels, Visualize numeric values, Color by instance classes), and 'Selection' (checkbox for Select full rows). At the bottom, there are buttons for 'Restore Original Order' (which is highlighted with a red box) and 'Send Automatically'. The main area is a table with columns: 'origin time', 'category', 'image name', 'image Compressed/mni image' (which is highlighted with a blue box), 'size', and 'width'. The table contains 19 rows, numbered 1 to 19. Row 17 is currently selected, indicated by a dark gray background. The 'image Compressed/mni image' column shows blurred versions of the images corresponding to each row.

origin time	category	image name	image Compressed/mni image	size	width
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					

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

*

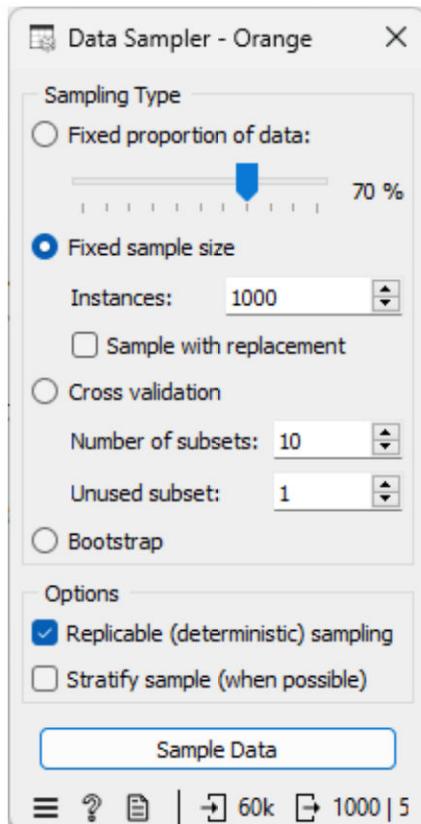
Open the widget.

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. *

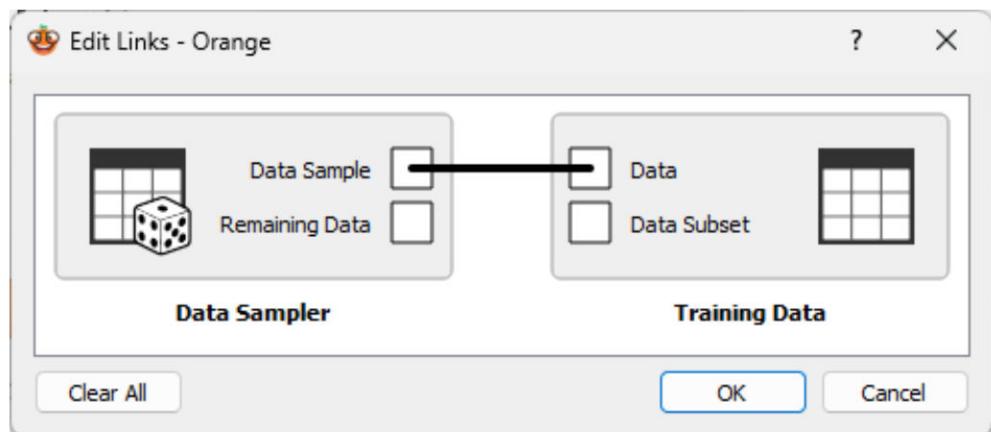
Done

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.

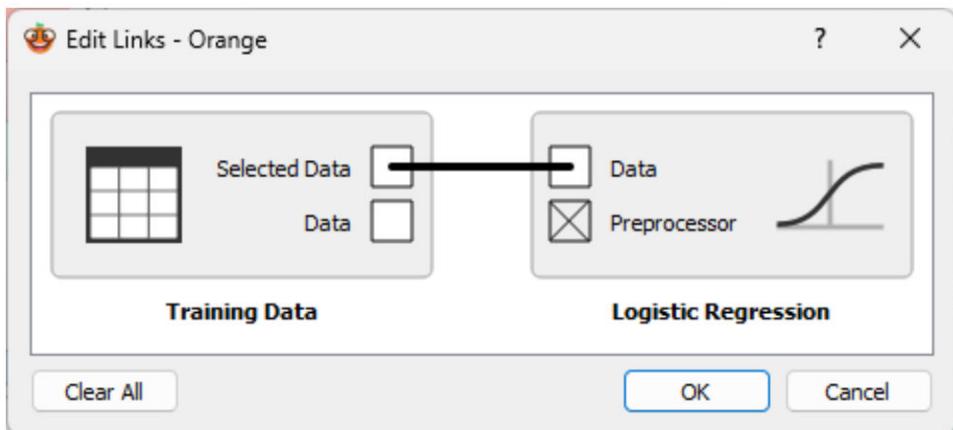
Step 5: Train the Model

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](#).

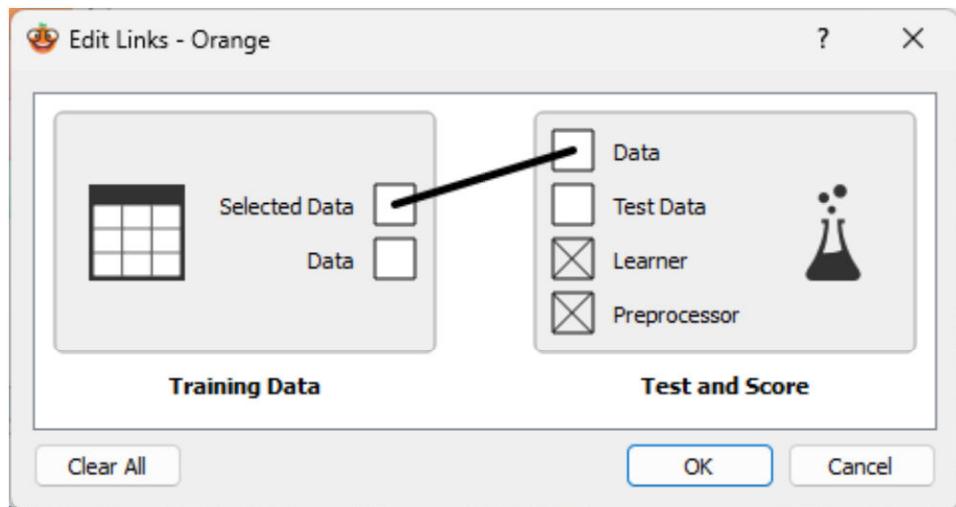
I have read and understood.

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

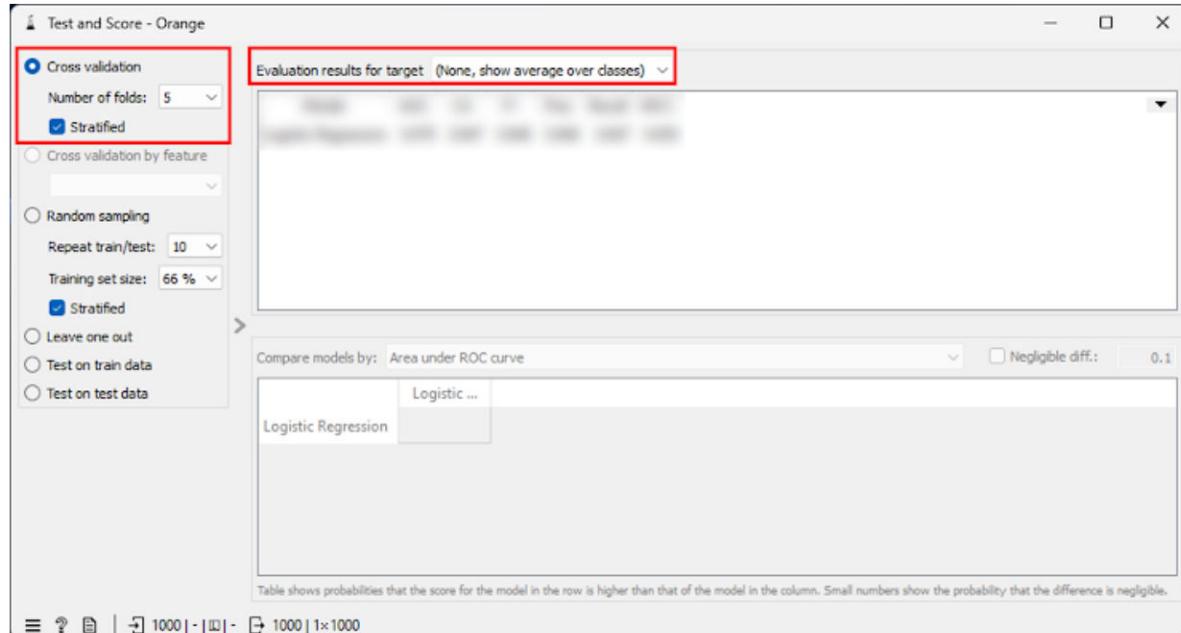
Edit the link between "Training Data" and the "Test and Score" widget as follows: *



Done

Open the "Test and Score" widget. *

Keep the default settings as follows:



Done

What does this setting in the red box do? *

The screenshot shows the 'Test and Score' widget in the Orange data mining software. The 'Cross validation' tab is active, with 'Number of folds' set to 5 and 'Stratified' checked. The 'Random sampling' tab shows 'Repeat train/test' set to 10 and 'Training set size' at 66%, also with 'Stratified' checked. The 'Evaluation results' section displays a table comparing models based on the Area under ROC curve, with 'Logistic ...' and 'Logistic Regression' listed. The 'Compare models by' dropdown is set to 'Area under ROC curve'. A note at the bottom states: 'Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.'

- 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

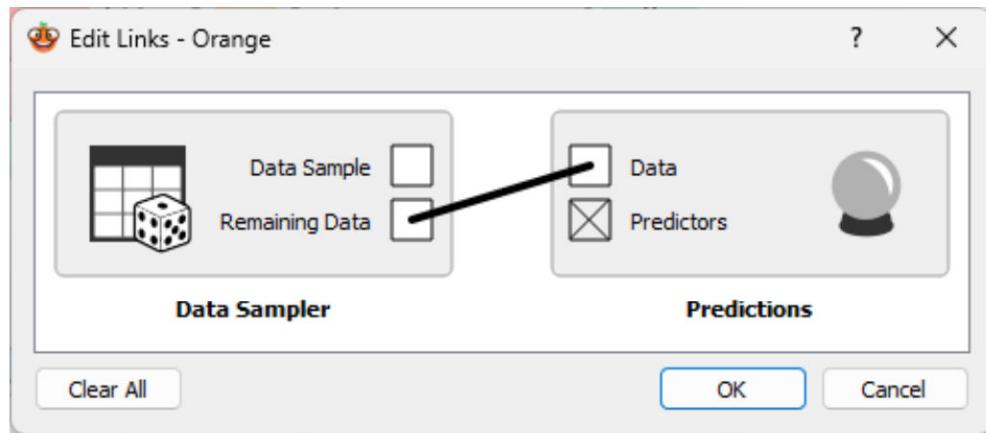
Step 6: Test the Model

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.

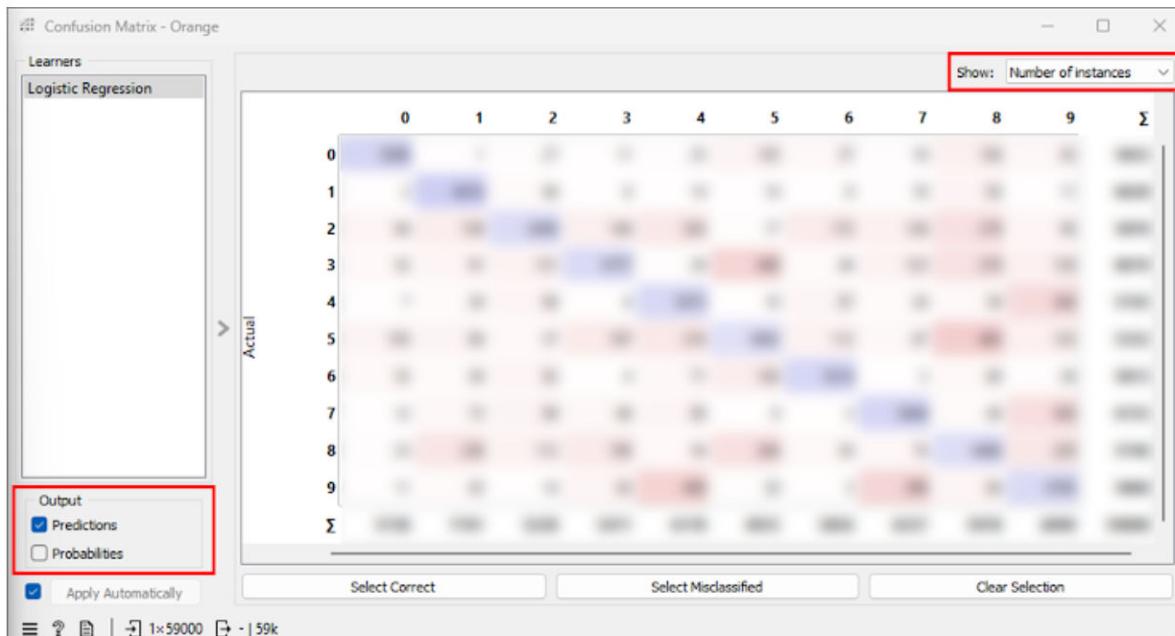
Connect it to the "Predictions" widget.

- 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? *

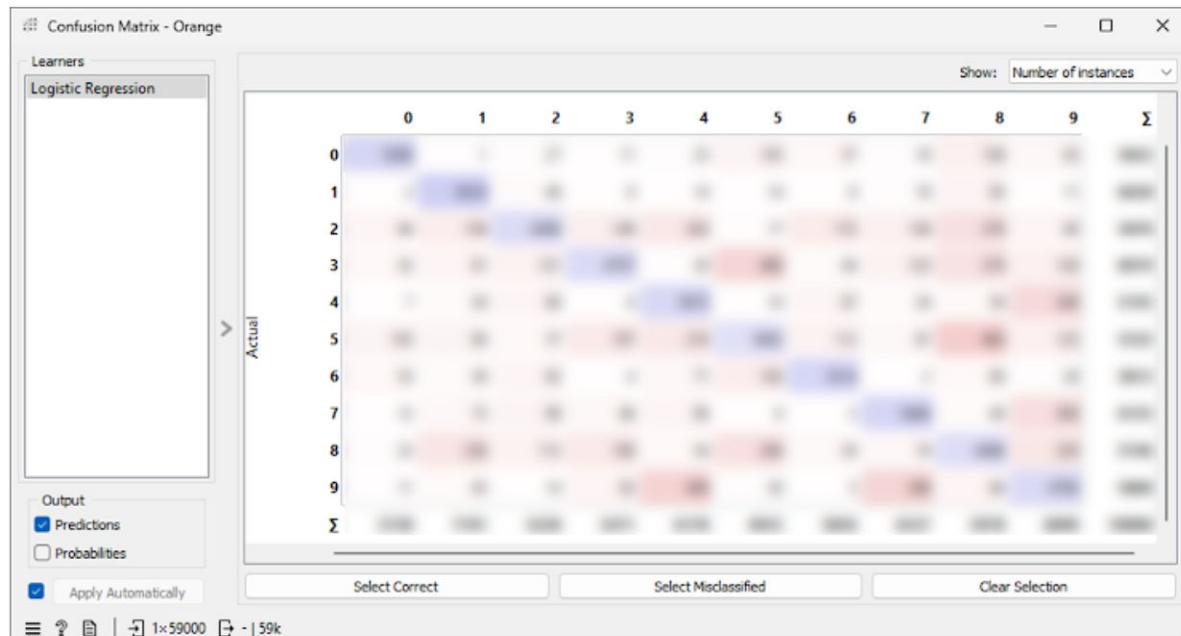
Hint: Look for the reddest cell in the confusion matrix and check the "Actual" label for that row.

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.

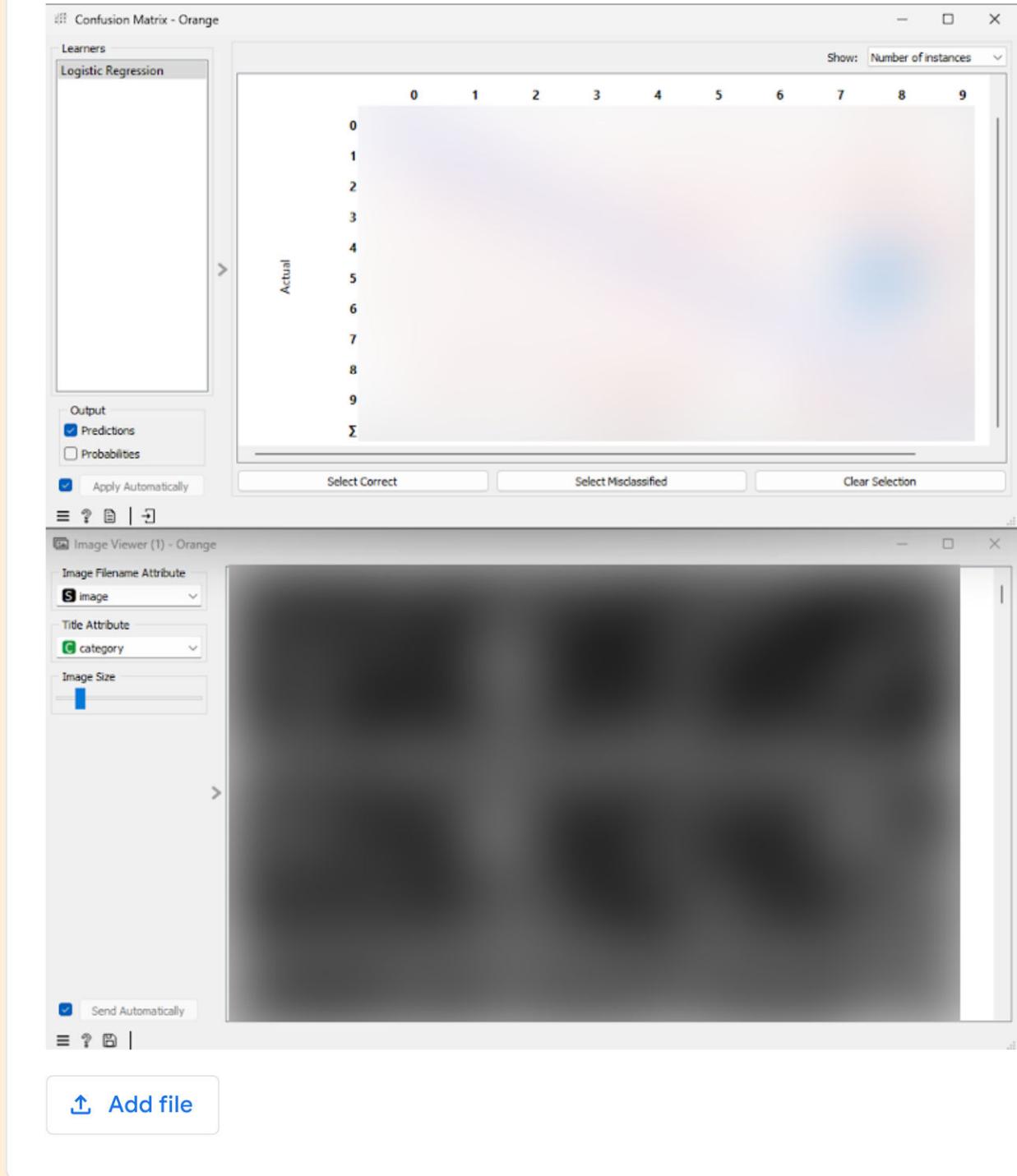
Done

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:



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.

Screenshot of the Orange data mining software interface showing the 'Predictions' and 'Image Viewer' windows.

Predictions - Orange window:

	Logistic Regression	error	category	image name	image	size	width
15	0.00 : 1.00 : 0.00 : 0.00 : 0.00 : 0.00 : 0.00 : 0.00 → 1	1.000	8	7329	8/7329.png	271	28
16	0.00 : 0.00 : 1.00 : 0.00 : 0.00 : 0.00 : 0.00 : 0.00 → 2	0.000	2	32243	2/32243.png	298	28
17	0.75 : 0.00 : 0.00 : 0.00 : 0.00 : 0.00 : 0.01 : 0.25 : 0.00 → 0	1.000	5	48230	5/48230.png	336	28
18	0.00 : 0.00 : 0.00 : 0.00 : 0.00 : 0.00 : 1.00 : 0.00 : 0.00 → 7	0.000	7	42410	7/42410.png	231	28

Probability that this image is...

0	1	2	3	4	5	6	7	8	9
75%	0%	0%	0%	0%	0%	0%	1%	25%	0%

Show performance scores Target class: (Average over classes)

Model	AUC	CA	F1	Prec	Recall	MCC
Logistic Regression	-2.645	0.850	0.850	0.851	0.850	0.834

Image Viewer (1) - Orange window:

Image Filename Attribute: image
Title Attribute: category
Image Size: 28x28



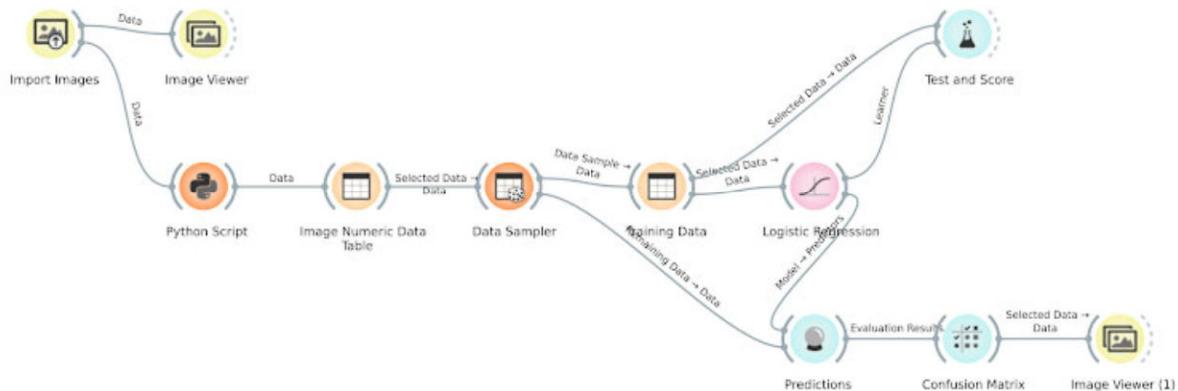
Send Automatically

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.

Step 8: Deploy the Model

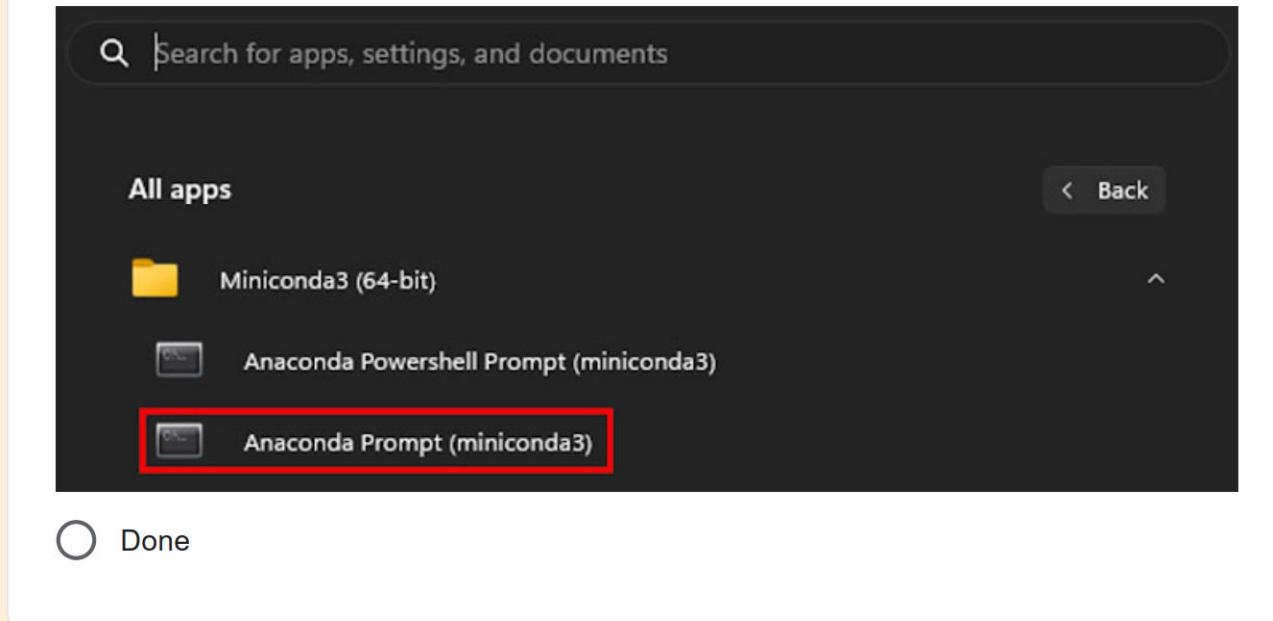
Let's deploy the model using Streamlit, an open-source Python framework for creating dynamic data apps with minimal code.

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:



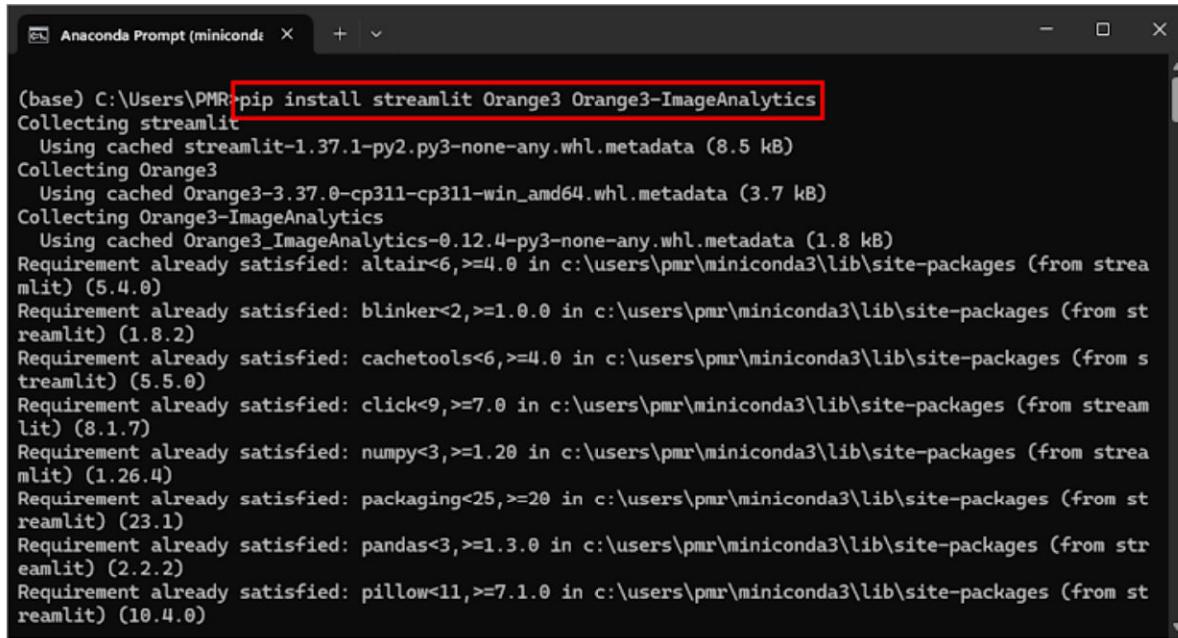
Once open, type:

*

pip install streamlit Orange3 Orange3-ImageAnalytics

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

Example:

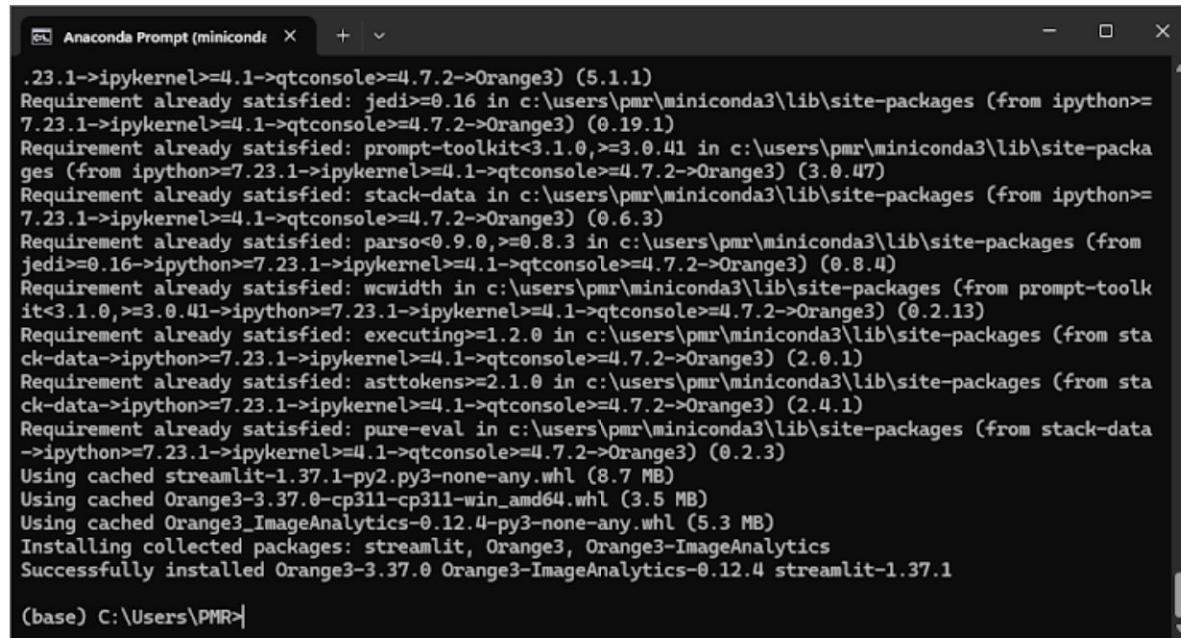


```
(base) C:\Users\PMR>pip install streamlit Orange3 Orange3-ImageAnalytics
Collecting streamlit
  Using cached streamlit-1.37.1-py2.py3-none-any.whl.metadata (8.5 kB)
Collecting Orange3
  Using cached Orange3-3.37.0-cp311-cp311-win_amd64.whl.metadata (3.7 kB)
Collecting Orange3-ImageAnalytics
  Using cached Orange3_ImageAnalytics-0.12.4-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: altair<6,>=4.0 in c:\users\pmr\miniconda3\lib\site-packages (from streamlit) (5.4.0)
Requirement already satisfied: blinker<2,>=1.0.0 in c:\users\pmr\miniconda3\lib\site-packages (from streamlit) (1.8.2)
Requirement already satisfied: cachetools<6,>=4.0 in c:\users\pmr\miniconda3\lib\site-packages (from streamlit) (5.5.0)
Requirement already satisfied: click<9,>=7.0 in c:\users\pmr\miniconda3\lib\site-packages (from streamlit) (8.1.7)
Requirement already satisfied: numpy<3,>=1.20 in c:\users\pmr\miniconda3\lib\site-packages (from streamlit) (1.26.4)
Requirement already satisfied: packaging<25,>=20 in c:\users\pmr\miniconda3\lib\site-packages (from streamlit) (23.1)
Requirement already satisfied: pandas<3,>=1.3.0 in c:\users\pmr\miniconda3\lib\site-packages (from streamlit) (2.2.2)
Requirement already satisfied: pillow<11,>=7.1.0 in c:\users\pmr\miniconda3\lib\site-packages (from streamlit) (10.4.0)
```

Done

It may take some time to complete. *

You should end up with this:



```
.23.1->ipykernel>=4.1->qtconsole>=4.7.2->Orange3) (5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\pmr\miniconda3\lib\site-packages (from ipython>=7.23.1->ipykernel>=4.1->qtconsole>=4.7.2->Orange3) (0.19.1)
Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.41 in c:\users\pmr\miniconda3\lib\site-packages (from ipython>=7.23.1->ipykernel>=4.1->qtconsole>=4.7.2->Orange3) (3.0.47)
Requirement already satisfied: stack-data in c:\users\pmr\miniconda3\lib\site-packages (from ipython>=7.23.1->ipykernel>=4.1->qtconsole>=4.7.2->Orange3) (0.6.3)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in c:\users\pmr\miniconda3\lib\site-packages (from jedi>=0.16->ipython>=7.23.1->ipykernel>=4.1->qtconsole>=4.7.2->Orange3) (0.8.4)
Requirement already satisfied: wctwidth in c:\users\pmr\miniconda3\lib\site-packages (from prompt-toolkit<3.1.0,>=3.0.41->ipython>=7.23.1->ipykernel>=4.1->qtconsole>=4.7.2->Orange3) (0.2.13)
Requirement already satisfied: executing>=1.2.0 in c:\users\pmr\miniconda3\lib\site-packages (from stack-data->ipython>=7.23.1->ipykernel>=4.1->qtconsole>=4.7.2->Orange3) (2.0.1)
Requirement already satisfied: asttokens>=2.1.0 in c:\users\pmr\miniconda3\lib\site-packages (from stack-data->ipython>=7.23.1->ipykernel>=4.1->qtconsole>=4.7.2->Orange3) (2.4.1)
Requirement already satisfied: pure-eval in c:\users\pmr\miniconda3\lib\site-packages (from stack-data->ipython>=7.23.1->ipykernel>=4.1->qtconsole>=4.7.2->Orange3) (0.2.3)
Using cached streamlit-1.37.1-py2.py3-none-any.whl (8.7 MB)
Using cached Orange3-3.37.0-cp311-cp311-win_amd64.whl (3.5 MB)
Using cached Orange3_ImageAnalytics-0.12.4-py3-none-any.whl (5.3 MB)
Installing collected packages: streamlit, Orange3, Orange3-ImageAnalytics
Successfully installed Orange3-3.37.0 Orange3-ImageAnalytics-0.12.4 streamlit-1.37.1

(base) C:\Users\PMR>
```

Done

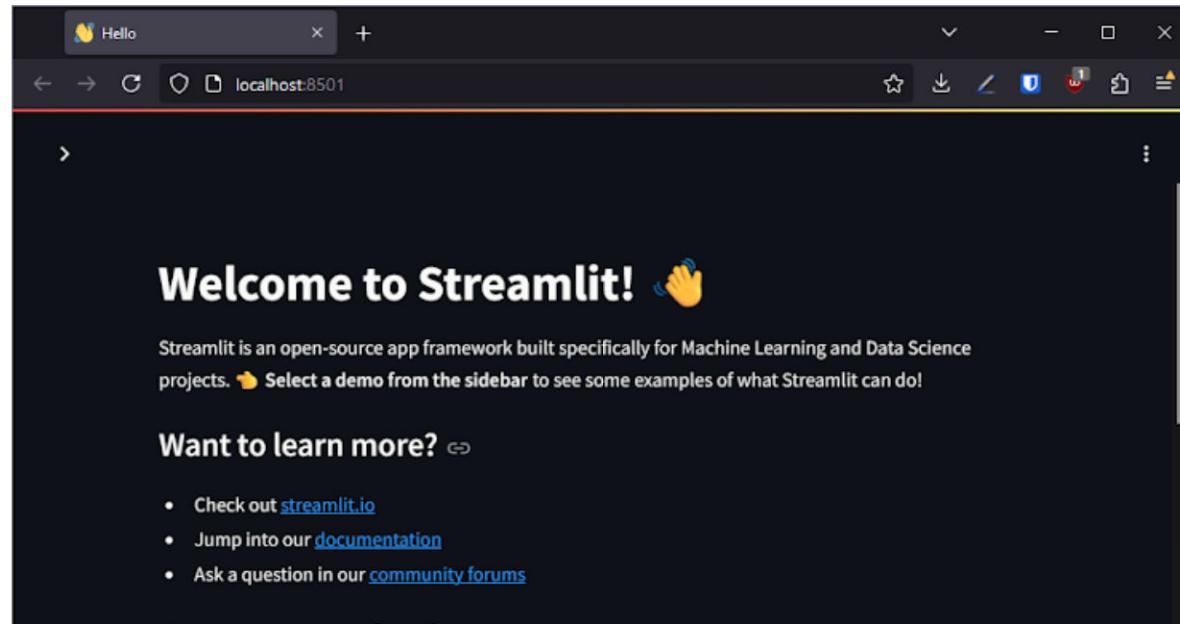
To check if it's working, type:

*

streamlit hello

This will launch a demo app in your browser.

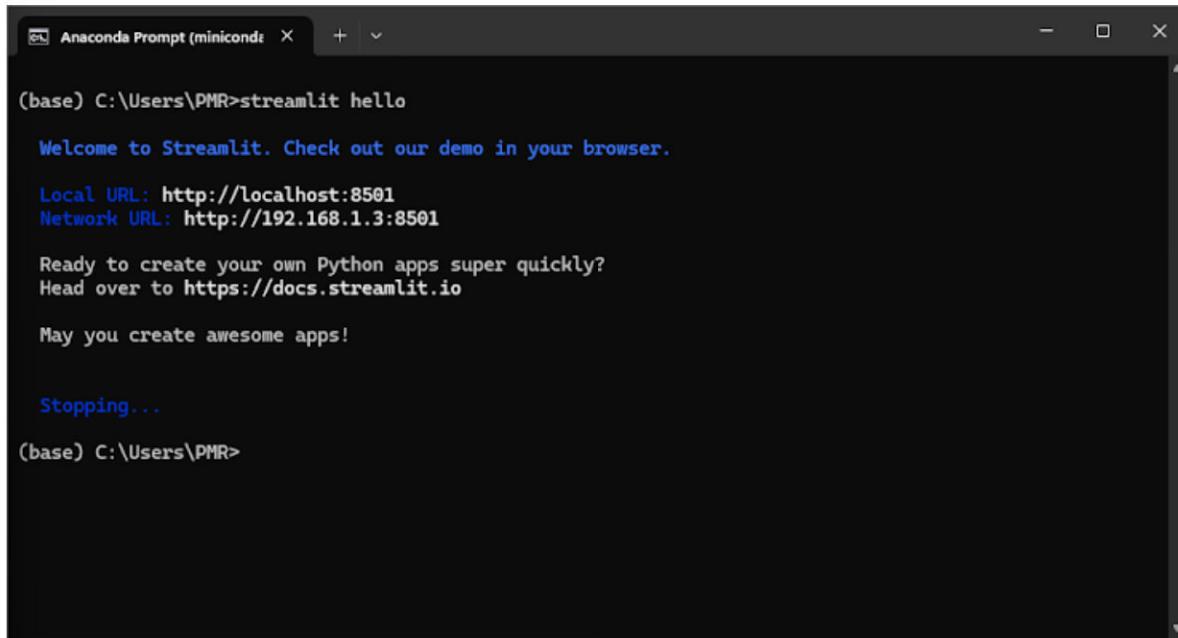
You should end up with this:



Done

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

You should end up with this:



A screenshot of an Anaconda Prompt window titled "Anaconda Prompt (miniconda)". The command entered is "streamlit hello". The output shows Streamlit's welcome message, local and network URLs, and a prompt to create awesome apps. It ends with "Stopping..." and a final prompt at "(base) C:\Users\PMR>".

```
(base) C:\Users\PMR>streamlit hello
Welcome to Streamlit. Check out our demo in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.1.3:8501

Ready to create your own Python apps super quickly?
Head over to https://docs.streamlit.io

May you create awesome apps!

Stopping...
(base) C:\Users\PMR>
```

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.

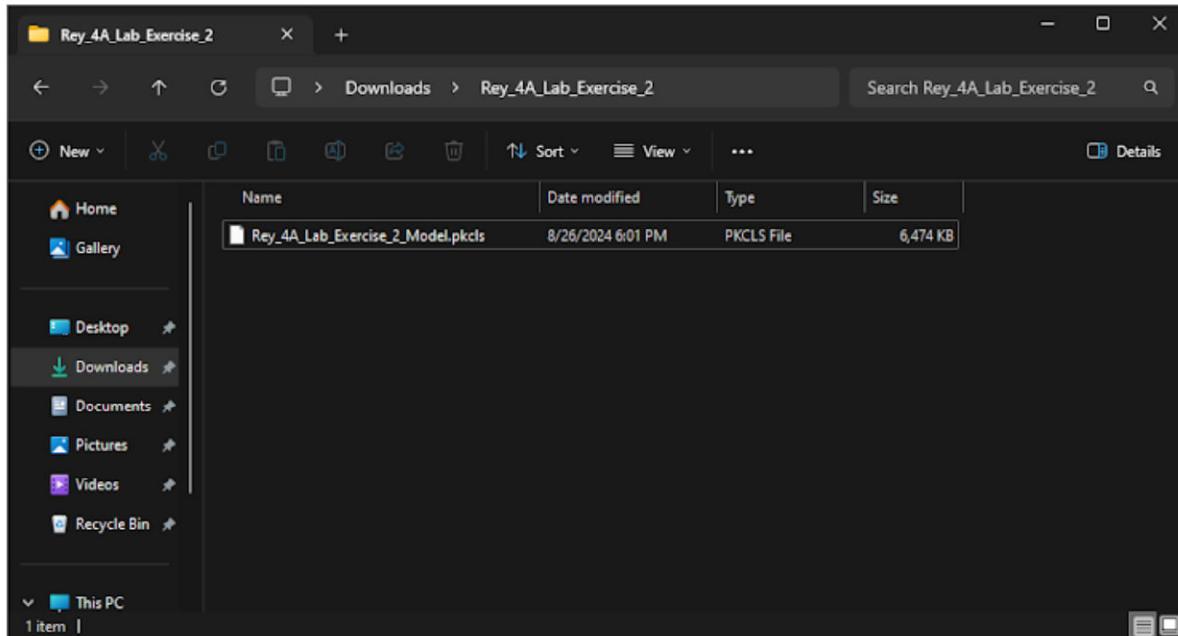
*

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:

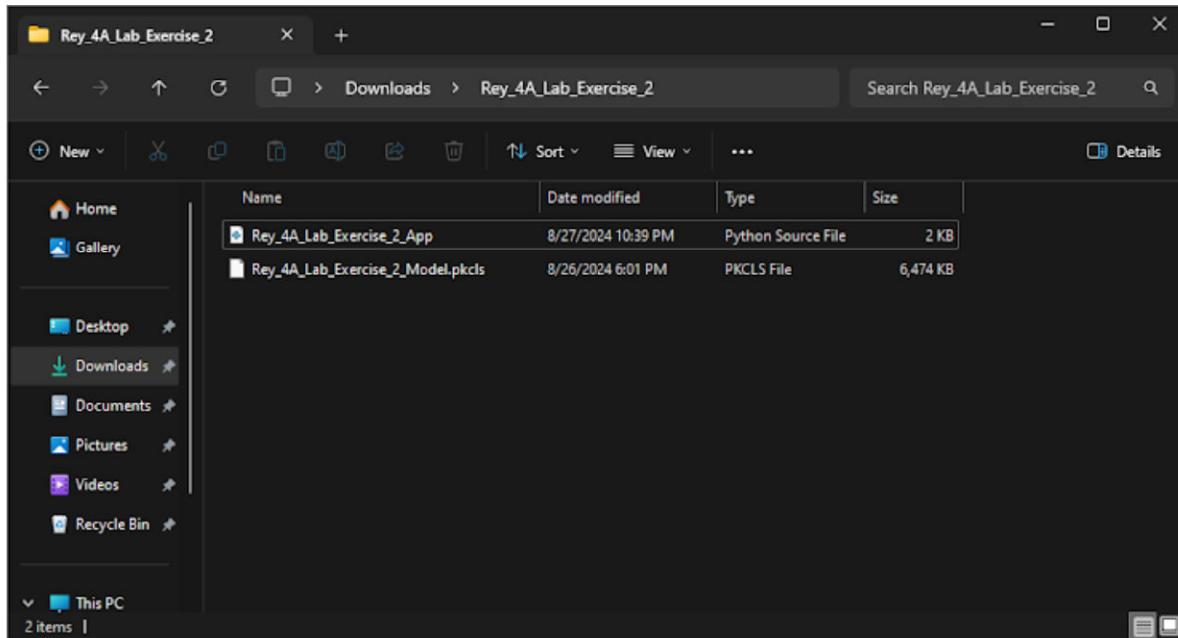
*

[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

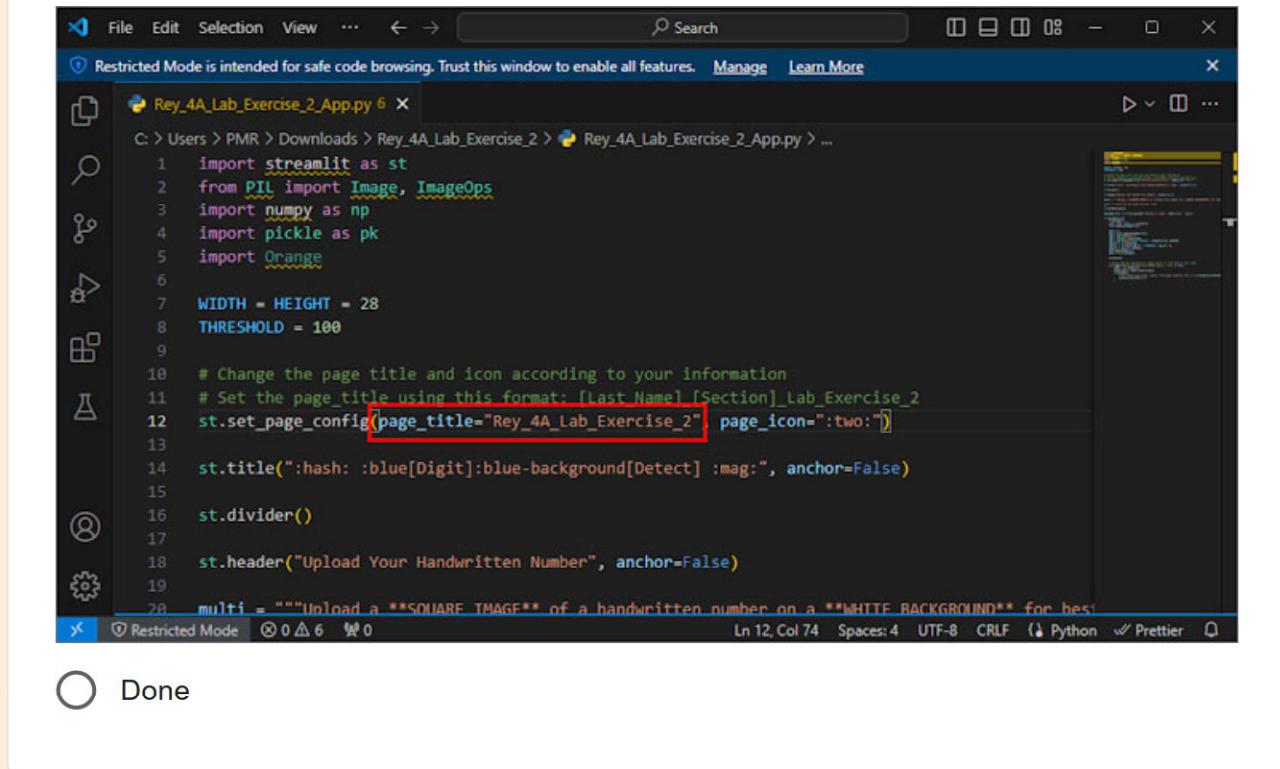
Open the Python file with your favorite text editor. *

Done

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

[Last_Name]_[Section]_Lab_Exercise_2

Example:

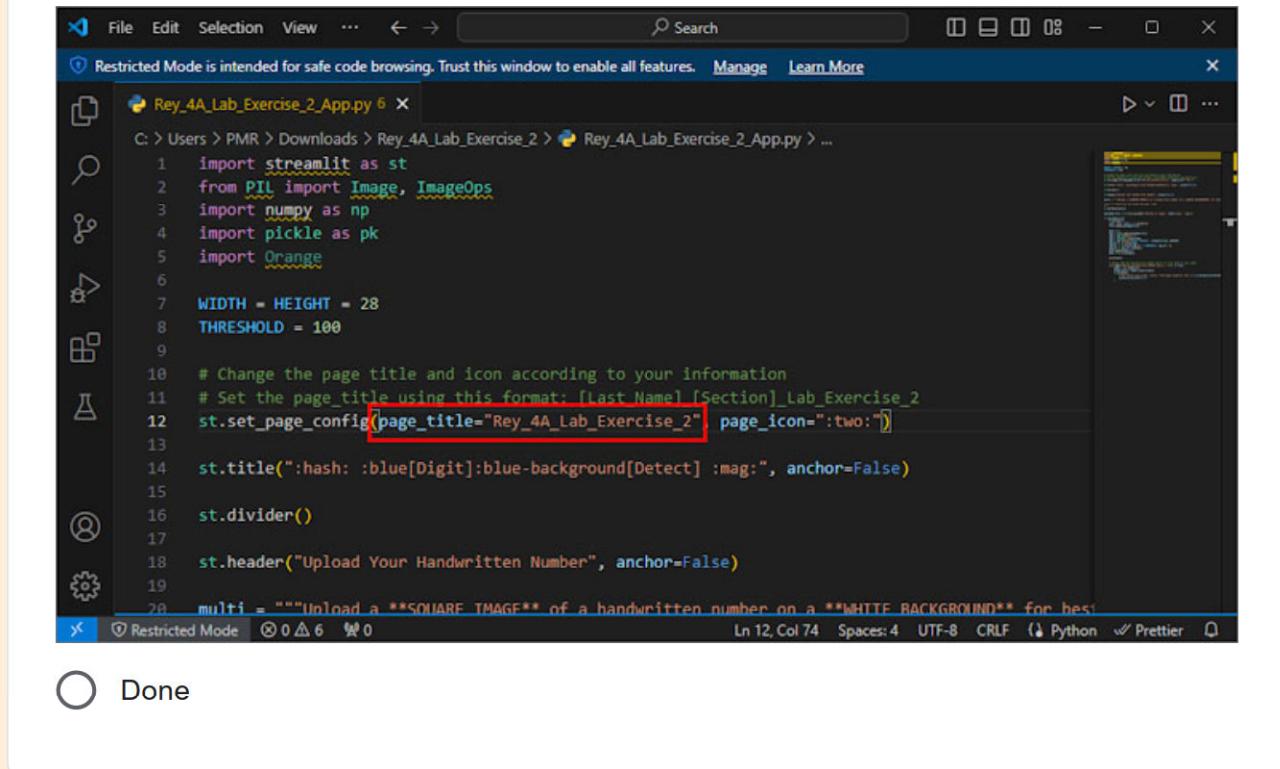


```
C:\> Users > PMR > Downloads > Rey_4A_Lab_Exercise_2 > Rey_4A_Lab_Exercise_2_App.py > ...
1 import streamlit as st
2 from PIL import Image, ImageOps
3 import numpy as np
4 import pickle as pk
5 import Orange
6
7 WIDTH = HEIGHT = 28
8 THRESHOLD = 100
9
10 # Change the page title and icon according to your information
11 # Set the page_title using this format: [Last Name]_[Section]_Lab_Exercise_2
12 st.set_page_config(page_title="Rey_4A_Lab_Exercise_2", page_icon=":two:")
13
14 st.title(":hash: :blue[Digit]:blue-background[Detect] :mag:", anchor=False)
15
16 st.divider()
17
18 st.header("Upload Your Handwritten Number", anchor=False)
19
20 multi = """Upload a **SQUARE IMAGE** of a handwritten number on a **WHITE BACKGROUND** for best
Ln 12, Col 74  Spaces: 4  UTF-8  CRLF  (Python)  ✓ Prettier  □
```

Done

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

Example:



```
File Edit Selection View ... ← → Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Rey_4A_Lab_Exercise_2_App.py 6 ×
C: > Users > PMR > Downloads > Rey_4A_Lab_Exercise_2 > Rey_4A_Lab_Exercise_2_App.py > ...
1 import streamlit as st
2 from PIL import Image, ImageOps
3 import numpy as np
4 import pickle as pk
5 import Orange
6
7 WIDTH = HEIGHT = 28
8 THRESHOLD = 100
9
10 # Change the page title and icon according to your information
11 # Set the page_title using this format: [Last Name] [Section]_Lab_Exercise_2
12 st.set_page_config(page_title="Rey_4A_Lab_Exercise_2", page_icon=":two:")
13
14 st.title(":hash: :blue[Digit]:blue-background[Detect] :mag:", anchor=False)
15
16 st.divider()
17
18 st.header("Upload Your Handwritten Number", anchor=False)
19
20 multi = """Upload a **SQUARE IMAGE** of a handwritten number on a **WHITE BACKGROUND** for best results.
Ln 12, Col 74 Spaces: 4 UTF-8 CRLF ⓘ Python ⓘ Prettier ⓘ
```

Done

Save the code.

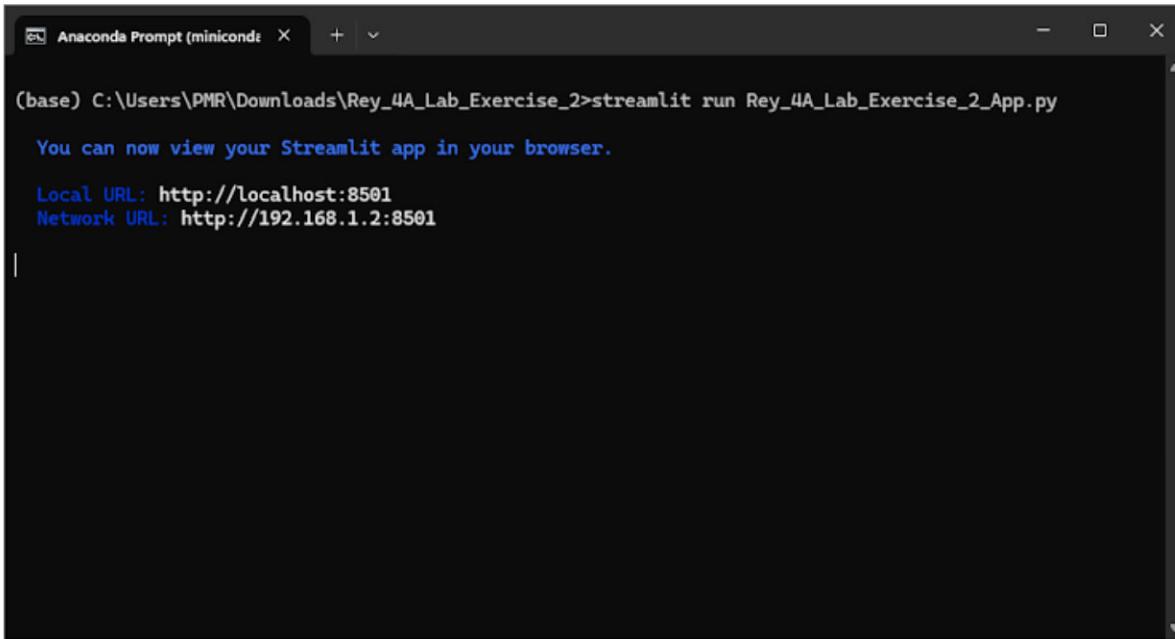
*

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:

A screenshot of a Windows terminal window titled "Anaconda Prompt (miniconda3)". The command "streamlit run Rey_4A_Lab_Exercise_2_App.py" has been run, and the output shows: "You can now view your Streamlit app in your browser.", "Local URL: http://localhost:8501", and "Network URL: http://192.168.1.2:8501".

```
(base) C:\Users\PMR\Downloads\Rey_4A_Lab_Exercise_2>streamlit run Rey_4A_Lab_Exercise_2_App.py
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://192.168.1.2:8501
```

Done

Follow the instructions in the app.

*

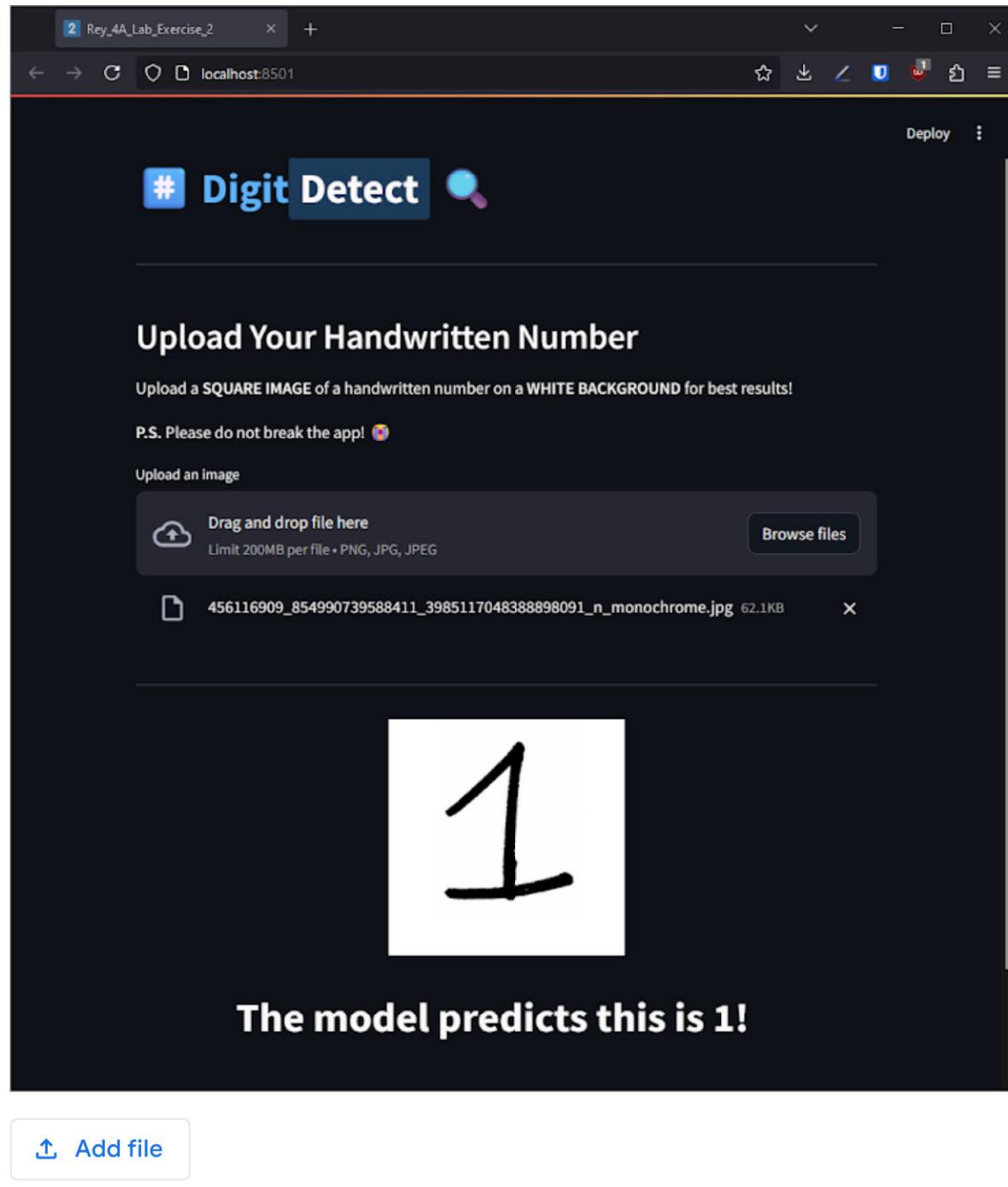
Use the **last digit of your student number** as the target for the model to predict.

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:



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

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

Content Curated by Pollux M. Rey