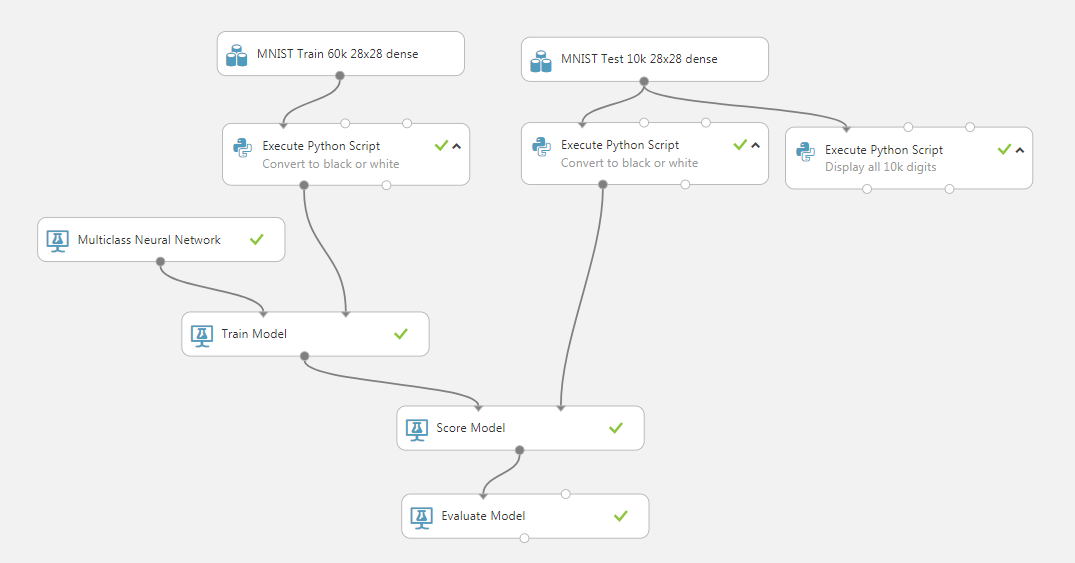
# Outline

Basic steps to build experiment:

* Create model
  1. Get data
  2. Prepare data (e.g. remove bad/missing data) – convert to black/white
  3. Select/ignore features – all features
* Train model
  1. Choose and apply ML algorithm against training data set
* Score the model
  1. Score against test data set
* Create a Web Service of score model
  1. Draw your own digits and see predicted result – new week



Reference

<https://github.com/Azure-Readiness/hol-azure-machine-learning/blob/master/010-lab-cs-ocr.md>

# Setup Experimental Workspace Get Data

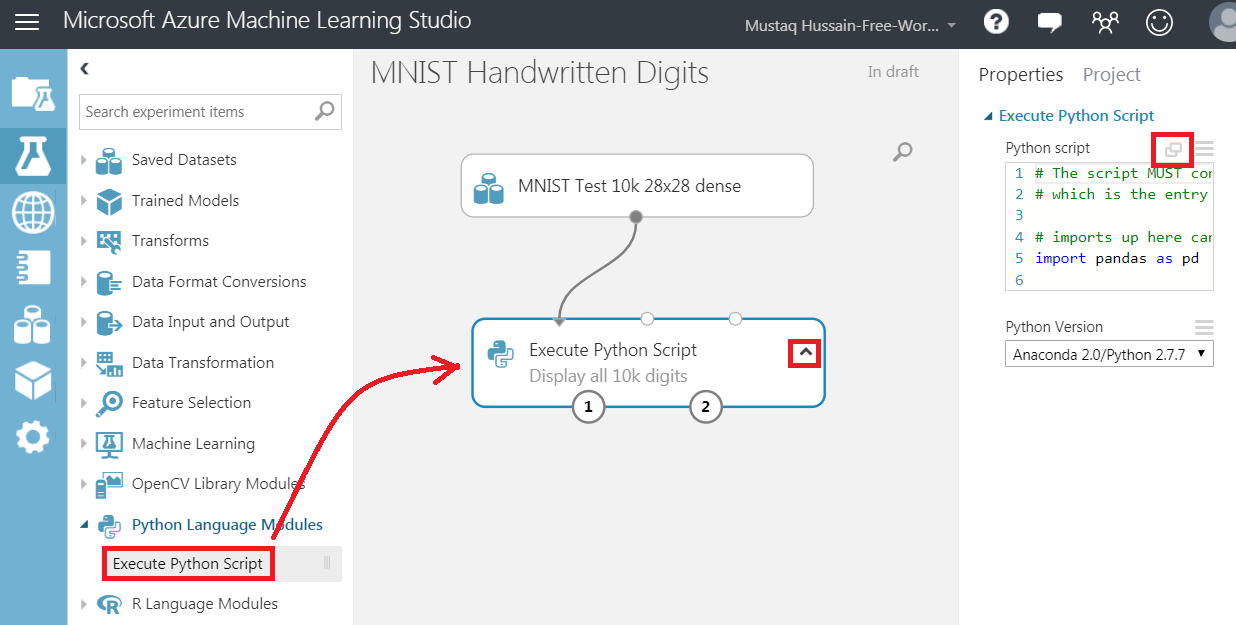
In this section after logging in you will:

* Create a new Azure ML experiment
* Get the MNIST Test dataset
* Create a Python script to display the digits it represents

1. Open browse at

<https://studio.azureml.net/>

1. Select ‘Sign In’ and sign in to Azure Machine Learning Studio with your Azure account
2. Create new experiment by clicking ‘+NEW’ at bottom of workspace
3. Then Select ‘NEW ‘EXPERIMENT’ then ‘Blank Experiment’
4. Change the experiment’s name from today’s date to ‘MNIST Handwritten Digits’
5. From the left hand pallet select ‘Saved Datasets’, ‘Samples’ then ‘MNIST Test 10k 28x28 dense’ and drop onto canvas.
6. From the left hand pallet select ‘Python Language Modules’ then ‘Execute Python Script’ and drop onto canvas.
7. Join output port of ‘MNIST Test 10k 28x28 dense’ to left hand input port of ‘Execute Python Script’
8. Double click ‘Execute Python Script’ and add comment ‘Display all 10k digits’ and click arrow to ensure comment is displayed
9. Select ‘Execute Python Script’ then in its Properties pane select Python script pop-out



1. Replace default code with following Python Script (which will display digits) and the press tick

import pandas as pd

import numpy as np

import math

from PIL import Image

# The entry point function contains to two input arguments:

# Param<dataframe1>: a pandas.DataFrame

# Param<dataframe2>: a pandas.DataFrame

def azureml\_main(df1 = None, df2 = None):

IMG\_W = 28 # Image width in pixels

IMG\_H = 28 # Image height in pixels

IMG\_C = 10000 # Image count, there are 10K images in the DB

# Tiling 100 \* 100

RES\_H = 100 # Horizontally how many images in the final resulting image

RES\_W = 100 # Vertically how many images in the final resulting image

# Dataframe to numpy array

npa = df1.as\_matrix()

# Create temp numpy array filled with 0s

res = np.zeros((RES\_H \* IMG\_H, RES\_W \* IMG\_W), dtype=np.uint8)

x = 0

y = 0

for didx in range(0, IMG\_C):

# Read raw digit data from column f0 to f800...

dgtpx = npa[didx, 1:]

# Convert img data to matrix form

dgtpx = np.reshape(dgtpx, (IMG\_H, IMG\_W))

cy = y \* IMG\_H

cx = x \* IMG\_W

res[cy:cy+IMG\_H, cx:cx+IMG\_H] = dgtpx

if y < RES\_H - 1:

y = y + 1

else:

y = 0

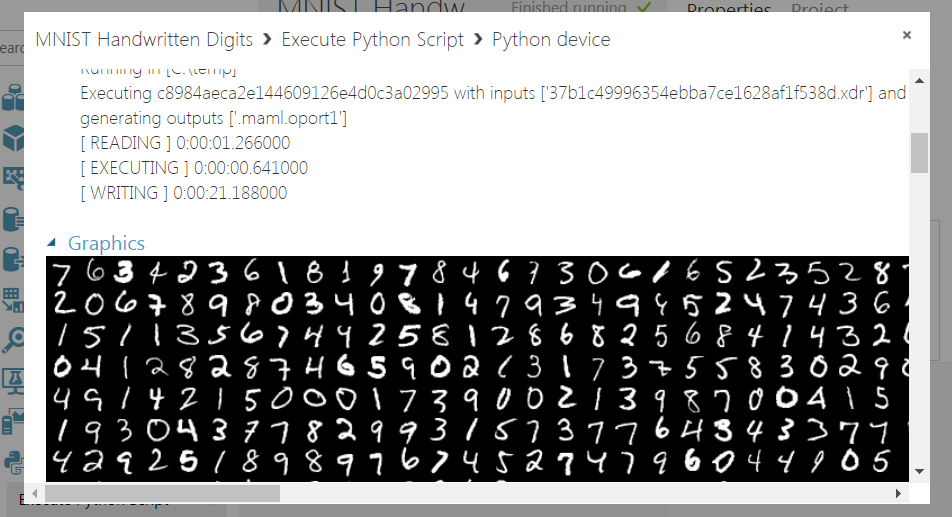
x = x + 1

img = Image.fromarray(res)

img.save('digit.png')

return df1,

1. Select ‘Run’ at bottom of canvas (~1min)
2. Select (right-hand mouse click) ‘Visualize‘ (~1min) from the right hand output port of ‘Execute Python Script’

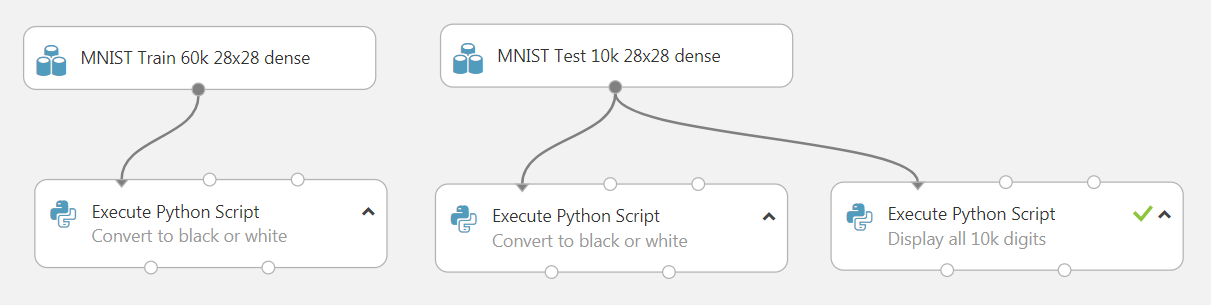


You should see grey scale graphical output of the digits in the test set – 1st one being a 7. Close dialogue box.

# Prepare Data by Converting Grey Scale to B/W

Convert MNIST grey scale features to black/white

1. From the ‘Saved Datasets’ drag and drop ‘MNIST Train 60k 28x28 dense’ onto canvas.
2. Drag another two ‘Execute Python Script’ modules onto canvas – adding comments ‘Convert to black or white’ to both.
3. Connect output from ‘MNIST Train 60k 28x28 dense’ to left hand port of one of the newly added ‘Execute Python Script’ module.
4. Connect output from ‘MNIST Train 10k 28x28 dense’ to left hand port of the other newly added ‘Execute Python Script’ module.



1. From the Properties pane of one of the ‘Convert to black or white’ module select pop-out script and replace with following code which will convert grey scale to black or white

import pandas as pd

import numpy as np

# The entry point function contains to two input arguments:

# Param<dataframe1>: a pandas.DataFrame

# Param<dataframe2>: a pandas.DataFrame

def azureml\_main(df1 = None, df2 = None):

# Dataframe to numpy array

npa = df1.as\_matrix()

# Read raw digit data: For each row :, starting from column 1: (pixel f0)

# (Note column 0 is the label that row represents and is being dropped here)

dgtpx = npa[:, 1:]

# If pixel!= 0 then set to 1

# (Convert all greyscale pixels to be either 0 or 1)

dgtpx[np.where(dgtpx != 0)] = 1

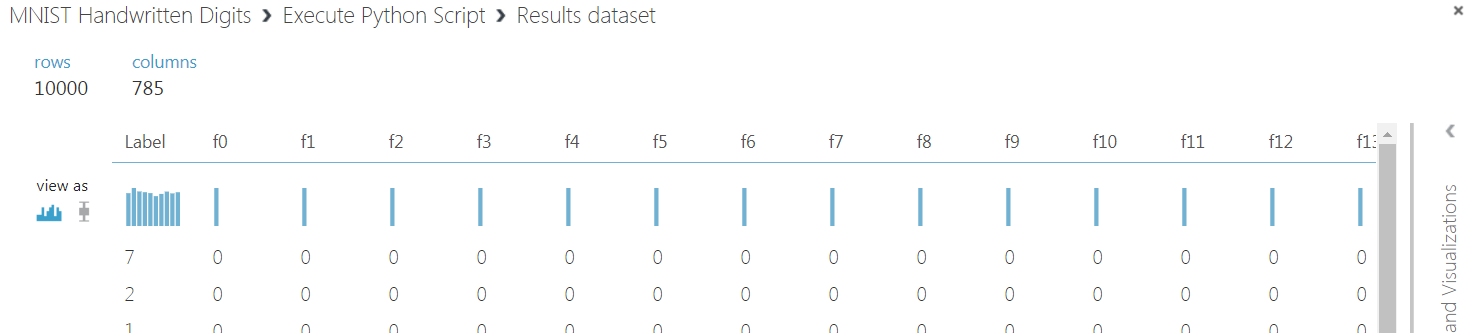
npa[:, 1:] = dgtpx

result = pd.DataFrame(npa)

result.columns = df1.columns.values

return result,

1. Do the same for the other ‘Convert to black or white’ module
2. Select the left-hand output port and ‘Visualize’ from 10k ‘Convert to black or white’ module

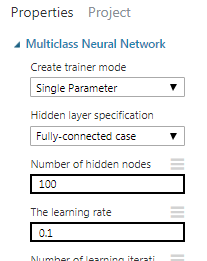


Note its ‘Label’ column (what digit it represents) and f0…f784 binary columns. Close dialog

# Choose ML algorithm

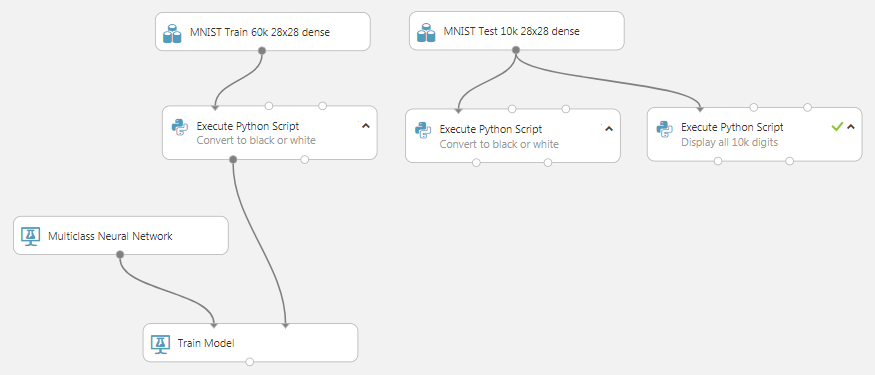
* The MNIST dataset (now black/white) needs to learn and classify the 10 types of digits (0-9) it represents, therefore a ‘Multiclass Neural Network’ would be a good choose.
* The model must be told that the ‘Label’ column in the dataset is what should be predicted.

1. From the pallet select ‘Machine Learning’, ‘Initialize Model’, ‘Classification’ and drag ‘Multiclass Neural Network’ module onto canvas
2. Select ‘Multiclass Neural Network’ and in the Properties pane you can see it defaults to 100 hidden nodes



1. From pallet select ‘Machine Learning’, ‘Train’ and drag ‘Train Model’ module onto canvas
2. Join ‘Multiclass Neural Network’ module to ‘Train Module’ left-hand port
3. Join MNIST 60k’s ‘Convert to black or white’ script module’s left-hand port to ‘Train Model’ right-hand port
4. Must tell model that ‘Label’ is what is going to be predicted, by selecting ‘Train Model’ Properties ‘Launch column selector’ and enter ‘Label’ – tick

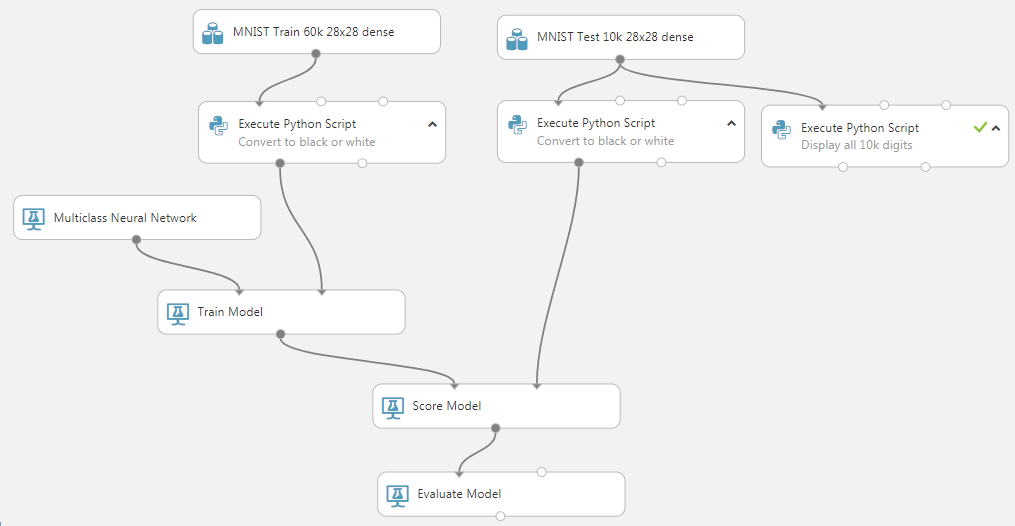




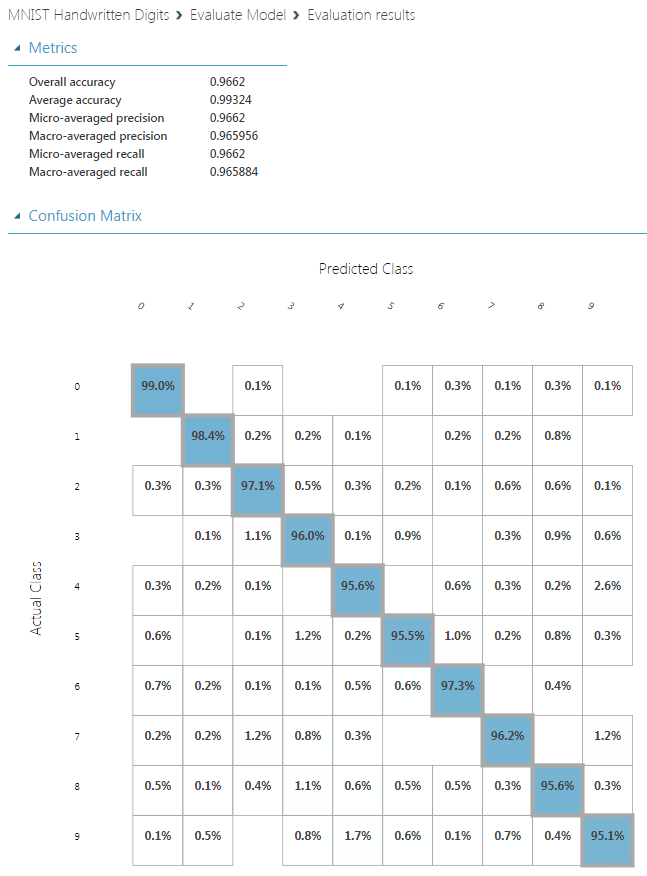
# Score Model & Evaluate Model

Finally run and train the NN and evaluate the results against the 10k dataset

1. Under ‘Machine Learning’, ‘Score’ drag and drop ‘Score Model’ module onto canvas and connect it to ‘Train Model’ and MNIST 10k’s ‘Convert to black or white’ script module
2. To evaluate experiment under ‘Machine Learning’, ‘Evaluate’ drag and drop ‘Evaluate Model’ module onto canvas and connect ‘Score Model’ to ’Evaluate Model’ left hand input port



1. Now re-run experiment ‘Run’ (~9min)
2. Select ‘Evaluate Model’ output port and ‘Visualize’



You can see the overall precision is ~96% in identifying the 10k test digits

Not bad, but not good enough for a realistic system – 1 in 20 letters being sent to wrong address. There are techniques to improve this such as convolution neural networks (99%).