Assignment One

CS 499

Richard McCormick (RLM443)

Python Program:

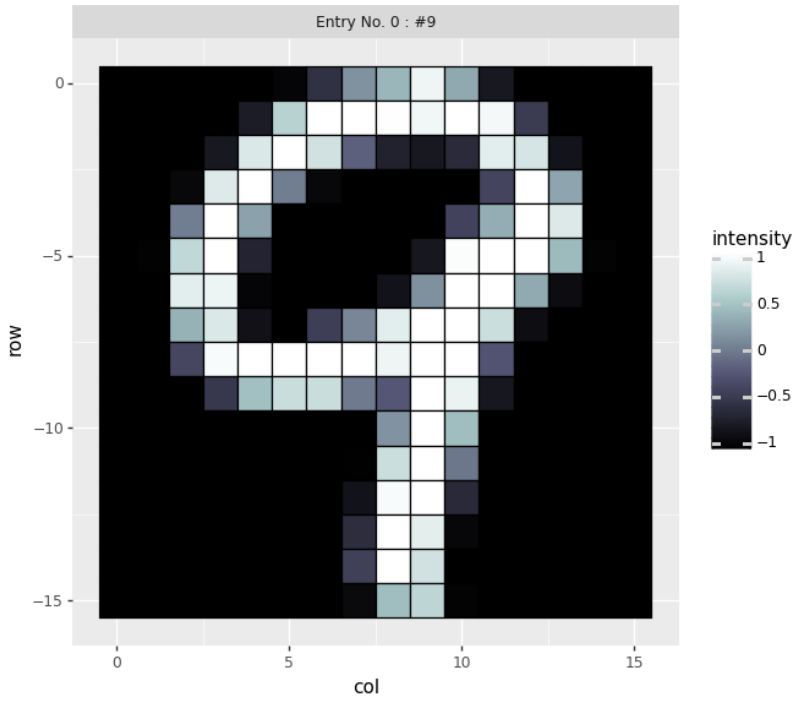
```
# <-- BEGIN IMPORTS / HEADERS -->
import os
import urllib
import urllib.request
import pandas as pd
import numpy as np
import plotnine as p9
from plotnine import ggplot, geom_tile
import warnings
# <-- END IMPORTS / HEADERS -->

# <-- BEGIN INITIALIZATION -->
# FILE VARIABLES
download_directory = "."
file = "zip.test.gz"
url = "https://github.com/tdhock/cs570-spring-2022/raw/master/data/zip.test.gz"
file_path = os.path.join(download_directory, file)
# CONSTANT VARIABLES
NUM_PIXELS = 16  # 16 pixels per side of image
IMAGE_SIZE = 256 # 256 pixels total
NUM_IMAGES = 9   # 9 images in a grid to be produced as final product
# <-- END INITIALIZATION -->

# <-- BEGIN FUNCTIONS -->
# FUNCTION: MAIN
#   Description  : Main driver for Assignment One
#   Inputs       : None
#   Outputs      : PlotNine graphs saved to program directory
#   Dependencies : build_image_df_from_dataframe
def main():
    # Display the title
    print("\nCS 499: Homework 1 Program Start")
    print("================================\n")

    # Suppress annoying plotnine warnings
    warnings.filterwarnings('ignore')
```

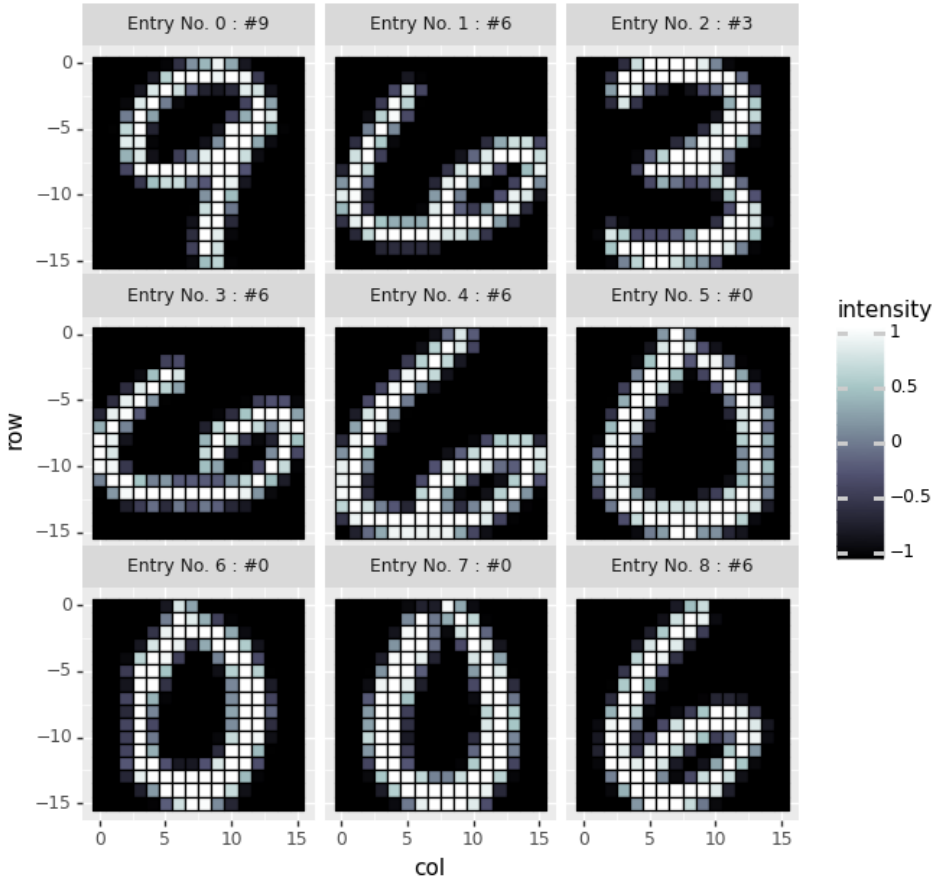
```
    # Check for data file. If not found, download
    if not os.path.isfile(file_path):
        try:
            print("Getting file: " + str(file) + "...\n")
            urllib.request.urlretrieve(url, file_path)
            print("File downloaded.\n")
        except(error):
            print(error)
    else:
        print("File: " + str(file) + " is already downloaded.\n")

    # Open the data file and assign to dataframe
    input_dataframe = pd.read_csv(file, header=None, sep=" ")

    # Print dataframe shape
    print("Data shape: " + str(input_dataframe.shape) + " [Unmodified]\n")

    # Drop index and print new size
    input_dataframe_no_index = input_dataframe.iloc[: , 1:]
    print("Data shape: " + str(input_dataframe_no_index.shape) + " [No Index]\n")

    # Convert dataframe to np array, print size
    input_data_array = input_dataframe.to_numpy()
    input_data_array_no_index = input_dataframe_no_index.to_numpy()
    print("Data shape: " + str(input_data_array_no_index.shape)
            + " [np Array, No Index]\n")

    # Print first row of data_array
    print("First row of Data: " + str(input_data_array[0]) + "\n")

    # Create a single image dataframe
    buffer_frame = build_image_df_from_dataframe(0, input_dataframe)

    # Turn single image dataframe into visual chart
    single_plot = (p9.ggplot(buffer_frame,
                             p9.aes(x='col', y='row', fill='intensity'))
                   + p9.geom_tile(color='black', size=0.5)
                   + p9.scale_fill_cmap('bone')
                   + p9.facet_wrap('uid')
                   + p9.theme(aspect_ratio=1))

    # Save single digit plot
    single_plot.save(filename = 'CS499_A1_SingleDigit.png')

    # Print confirmation of file save
```

```
    print("\nSaved plot: CS499_A1_SingleDigit.png")

    # Initialize the dataframe we will use to hold our multiple image frames
    multi_frames = pd.DataFrame()

    # Loop across a given range, building a dataframe for each image, and
    # concatenating it to a master image
    for index in range(NUM_IMAGES):
        buffer_frame = build_image_df_from_dataframe(index, input_dataframe)
        multi_frames = pd.concat([multi_frames, buffer_frame])

    # Generate the plot of all the images
    multi_plot = (p9.ggplot(multi_frames,
                            p9.aes(x='col', y='row', fill='intensity'))
            + p9.geom_tile(color='black', size=0.5)
            + p9.scale_fill_cmap('bone')
            + p9.facet_wrap('uid', labeller='both')
            + p9.theme(aspect_ratio=1))

    # Save multiple digits plot
    multi_plot.save(filename = 'CS499_A1_MultipleDigits.png')

    # Print confirmation of file save
    print("\nSaved plot: CS499_A1_MultipleDigits.png")

    # Display end of program
    print("\nCS 499: Homework 1 Program End")
    print("==============================")

# FUNCTION : BUILD_IMAGE_DF_FROM_DATAFRAME
#   Description: Builds data from a single data entry into a dataframe
#   Inputs:
#       - index     : index of input dataframe entry to convert
#       - dataframe : master dataframe of image data
#   Outputs:
#       - final_dataframe : completed dataframe for one image
def build_image_df_from_dataframe(index, input_dataframe):
    # Get a data entry at the given index of a given dataframe,
    # form it into a np array
    input_data_array = np.resize(((input_dataframe.iloc[index]).to_numpy()),
                                 (IMAGE_SIZE+1))

    # Get the numeric label of the data entry
    label_num = int(input_data_array[0])
    # Remove the label so we only have data
```

```
    input_data_array = input_data_array[1:]

    # Build dictionary which will become our dataframe
    data_dictionary = {
        # Columns go 1, 2, 3, ..., 1, 2, 3, ...
        'col' : np.resize(np.arange(NUM_PIXELS), IMAGE_SIZE),
        # Rows go 1, 1, 1, ..., 2, 2, 2, ...
        'row' : -np.repeat(np.arange(NUM_PIXELS), NUM_PIXELS),
        # Intensity data is unaltered
        'intensity' : input_data_array,
        # Unique ID (UID) is the entry number and label
        'uid' : np.repeat("Entry No. " + str(index) + " : #" + str(label_num),
                          IMAGE_SIZE)
    }

    # Build final dataframe
    final_dataframe = pd.DataFrame.from_dict(data_dictionary)

    # Show output to confirm dataframe was created, confirm size
    print("\nCompleted Dataframe: No. " + str(index) + " | #" + str(label_num))
    print(final_dataframe.shape)

    return(final_dataframe)
# <-- END FUNCTIONS -->

# Launch main
if __name__ == "__main__":
    main()
```

Program Output:





```
CS 499: Homework 1 Program Start
================================

File: zip.test.gz is already downloaded.

Data shape: (2007, 257) [Unmodified]

Data shape: (2007, 256) [No Index]

Data shape: (2007, 256) [np Array, No Index]

First row of Data: [ 9.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -9.48e-01

 -5.61e-01  1.48e-01  3.84e-01  9.04e-01  2.90e-01 -7.82e-01 -1.00e+00
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00
 -7.48e-01  5.88e-01  1.00e+00  1.00e+00  9.91e-01  9.15e-01  1.00e+00
  9.31e-01 -4.76e-01 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00 -7.87e-01  7.94e-01  1.00e+00  7.27e-01 -1.78e-01 -6.93e-01
 -7.86e-01 -6.24e-01  8.34e-01  7.56e-01 -8.22e-01 -1.00e+00 -1.00e+00
 -1.00e+00 -1.00e+00 -9.22e-01  8.10e-01  1.00e+00  1.00e-02 -9.28e-01
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -3.90e-01  1.00e+00  2.71e-01
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00  1.20e-02  1.00e+00  2.48e-01
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -4.02e-01  3.26e-01
  1.00e+00  8.01e-01 -9.98e-01 -1.00e+00 -1.00e+00 -9.81e-01  6.45e-01
  1.00e+00 -6.87e-01 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -7.92e-01
  9.76e-01  1.00e+00  1.00e+00  4.13e-01 -9.76e-01 -1.00e+00 -1.00e+00
 -9.93e-01  8.34e-01  8.97e-01 -9.51e-01 -1.00e+00 -1.00e+00 -1.00e+00
 -8.31e-01  1.40e-01  1.00e+00  1.00e+00  3.02e-01 -8.89e-01 -1.00e+00
 -1.00e+00 -1.00e+00 -1.00e+00  3.56e-01  7.94e-01 -8.36e-01 -1.00e+00
 -4.45e-01  7.40e-02  8.33e-01  1.00e+00  1.00e+00  6.96e-01 -8.81e-01
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -3.68e-01  9.55e-01
  1.00e+00  1.00e+00  1.00e+00  1.00e+00  9.05e-01  1.00e+00  1.00e+00
 -2.62e-01 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00 -5.07e-01  4.51e-01  6.92e-01  6.92e-01 -7.00e-03 -2.37e-01
  1.00e+00  8.82e-01 -7.95e-01 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00  1.55e-01  1.00e+00  4.36e-01 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00 -1.00e+00 -9.91e-01  7.03e-01  1.00e+00 -2.50e-02 -1.00e+00
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -8.33e-01  9.59e-01  1.00e+00
 -6.29e-01 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -6.00e-01
  9.98e-01  8.41e-01 -9.32e-01 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00 -4.24e-01  1.00e+00  7.32e-01 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00
 -1.00e+00 -1.00e+00 -1.00e+00 -9.08e-01  4.30e-01  6.22e-01 -9.73e-01
 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00 -1.00e+00]


Completed Dataframe: No. 0 | #9
(256, 4)
```

```
Completed Dataframe: No. 0 | #9
(256, 4)

Saved plot: CS499_A1_SingleDigit.png

Completed Dataframe: No. 0 | #9
(256, 4)

Completed Dataframe: No. 1 | #6
(256, 4)

Completed Dataframe: No. 2 | #3
(256, 4)

Completed Dataframe: No. 3 | #6
(256, 4)

Completed Dataframe: No. 4 | #6
(256, 4)

Completed Dataframe: No. 5 | #0
(256, 4)

Completed Dataframe: No. 6 | #0
(256, 4)

Completed Dataframe: No. 7 | #0
(256, 4)

Completed Dataframe: No. 8 | #6
(256, 4)

Saved plot: CS499_A1_MultipleDigits.png

CS 499: Homework 1 Program End
==============================

C:\Users\richard\Documents\School\CS 499\Assignment 1>
```

Question Answers:

1. Opening the raw data from the CSV file shows that the data is formatted as a two-by-two matrix. Each row represents a single image, and each column in that row represents one pixel of the image.

2. After dropping the index column, we have data in the size (2007x264). The size of the data represents 2007 images. Each image has one row with 264 columns, with each column representing a pixel. The singular value for each column represents the color intensity of that pixel. Each image has 264 columns, so we can say also that each image has 264 pixels.

3. I made use of a support function to create and format the data frame for each of my images. The function takes in both the input file as a data frame and an index and creates a data frame of the image at that index. It also makes use of the extra column in the original data to set the label of the image.

4. My code successfully creates a black-and-white 16px by 16px image of each entry in the data set. Additionally, it has the capability to create a data frame and image for any given index of the input data, as well as the ability to make a multi-faceted panel of any given number of data entries.

   The only thing that may need to be changed is the way I uniquely identified each image. Having a long string of text may not be useful in future applications.