Variational Autoencoders & Missing Values Completion Algorithms GUI

by Christos Kormaris

A graphical user interface (GUI) has been implemented for the project of this thesis, using Python and the **Tkinter** library.

First, browse to the directory **"vaes_gui"** of the thesis project and install the required Python dependencies, by typing:

```
pip install -r depencencies.txt
```

Listing 1: command for installing Python dependencies

To run the GUI from the terminal, type:

```
python vaes_gui.py
```

Listing 2: command for running the GUI

To create an executable file for the GUI (".exe"), which you can run anytime from a Windows environment, type:

```
pip install pyinstaller
pyinstaller vaes_gui.spec
```

Listing 3: command for creating an executable file for the GUI

Then, download all the datasets from the URL in the file **"datasets_urls.txt"** and move them to the newly created **"dist"** folder. Inside, there should be a folder with the name **"vaes_gui"**, which contains the executable file **"vaes_gui.exe"**.

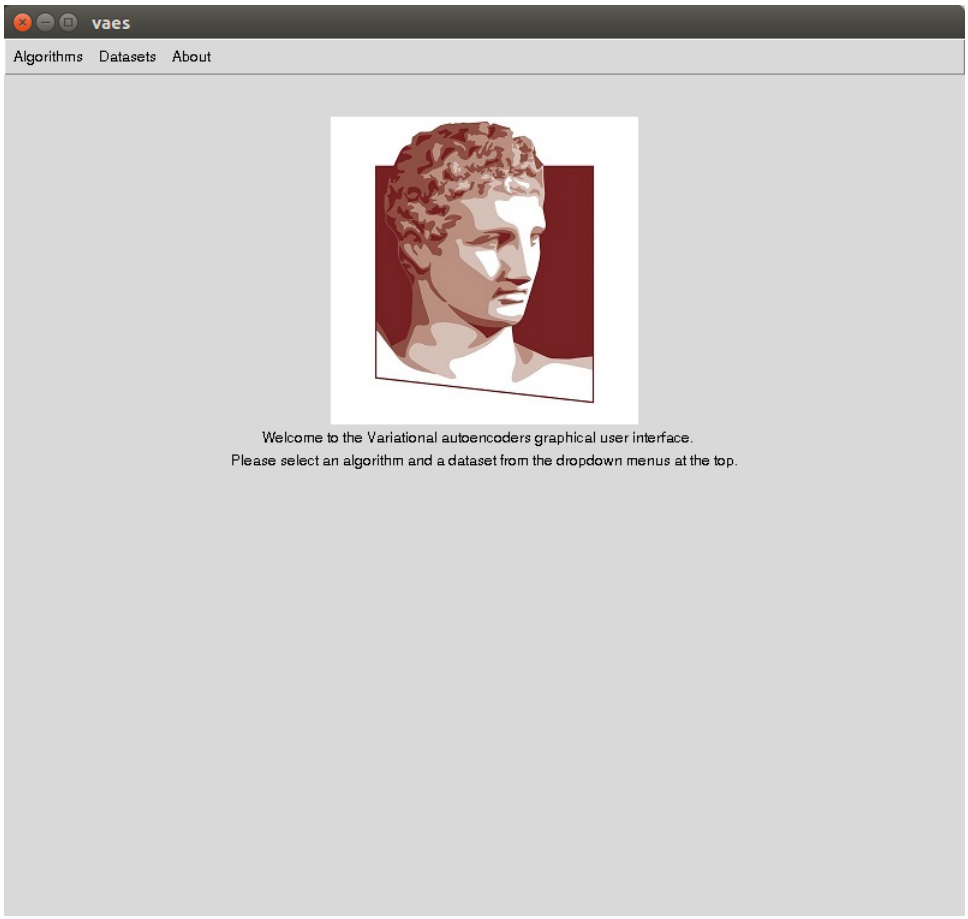
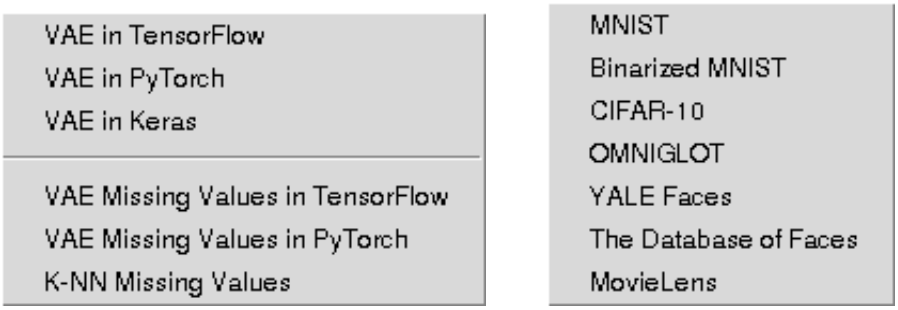


Figure 1: GUI Welcome page.



(a) GUI Algorithms dropdown menu.

(b) GUI Datasets dropdown menu.

Figure 2: Dropdown menus.

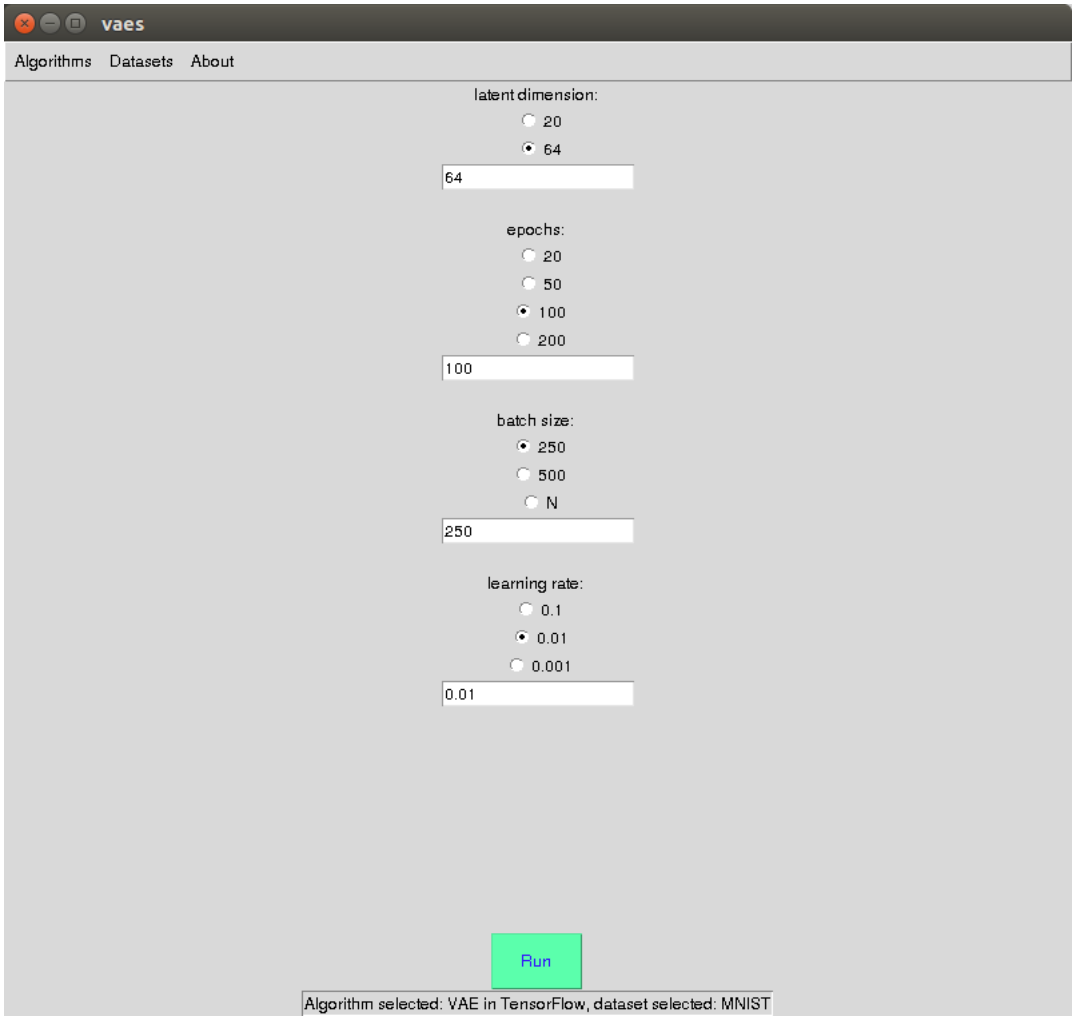


Figure 3: GUI VAE in TensorFlow, MNIST dataset.

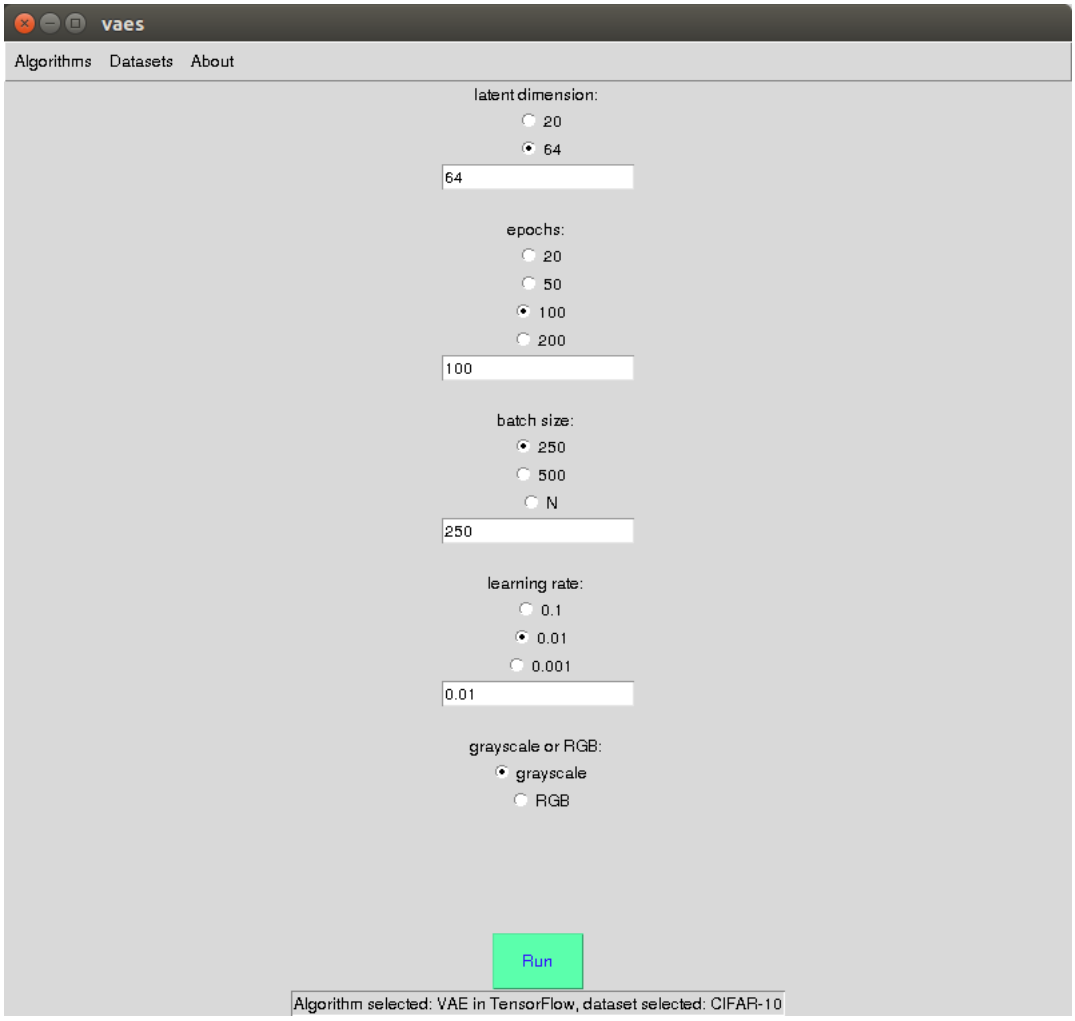


Figure 4: GUI VAE in TensorFlow, CIFAR-10 dataset.

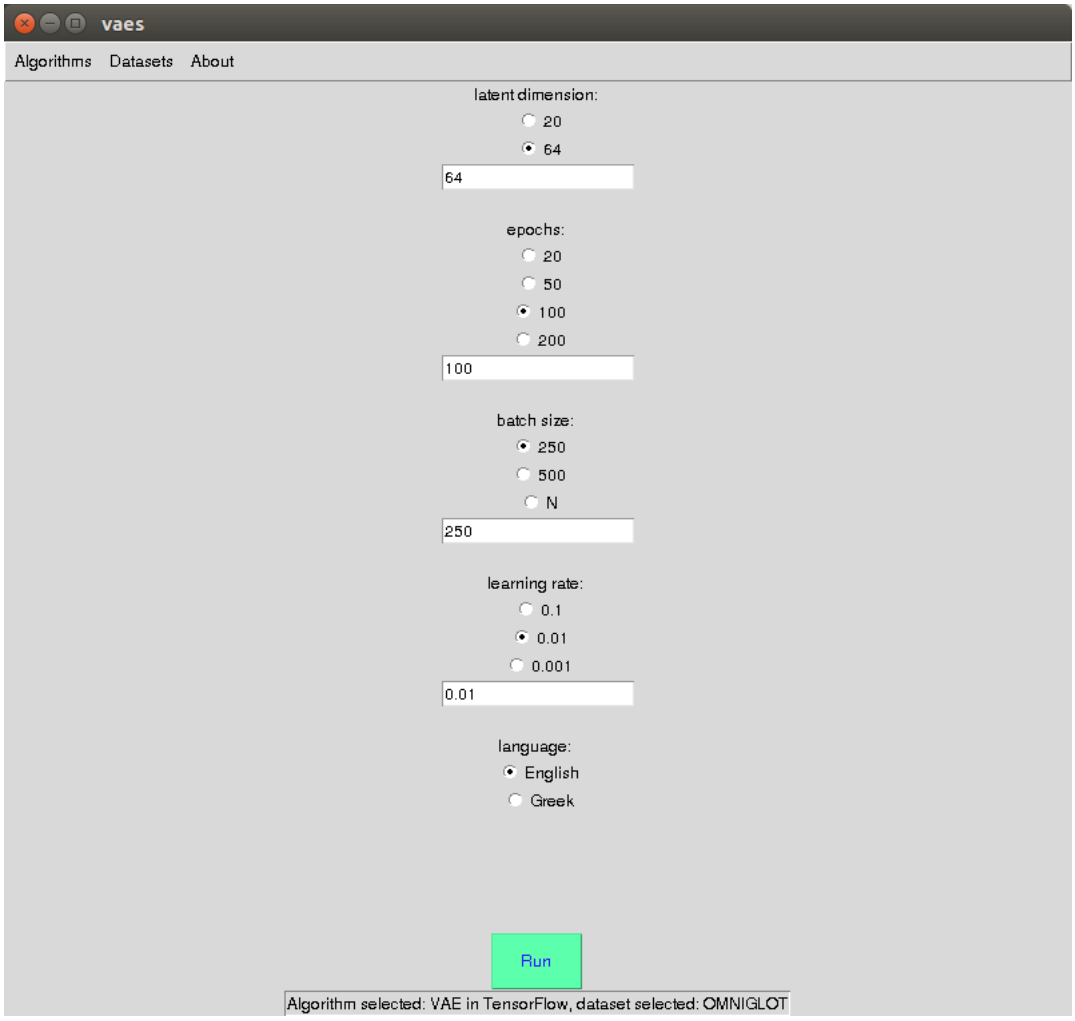


Figure 5: GUI VAE in TensorFlow, OMNIGLOT dataset.

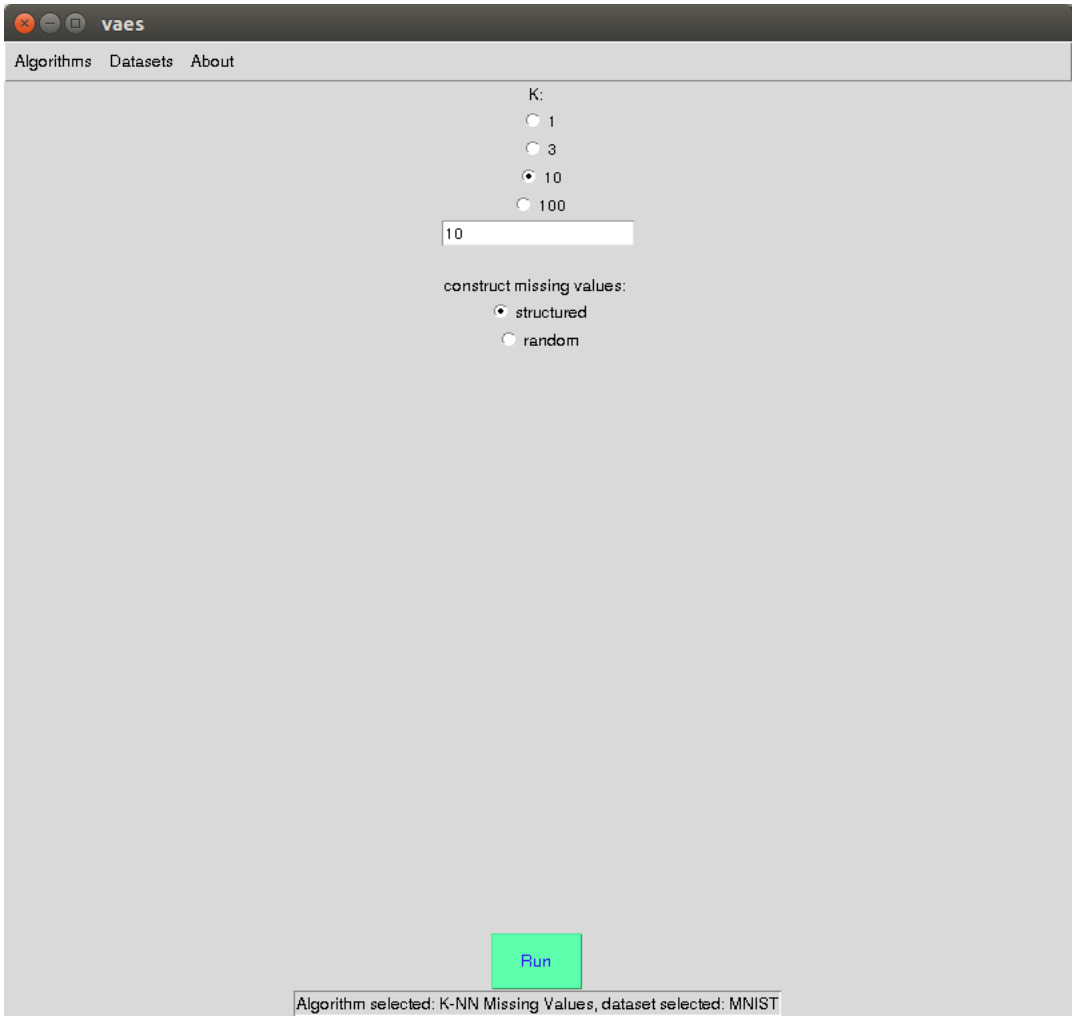


Figure 6: GUI K-NN Missing Values algorithm, MNIST dataset.

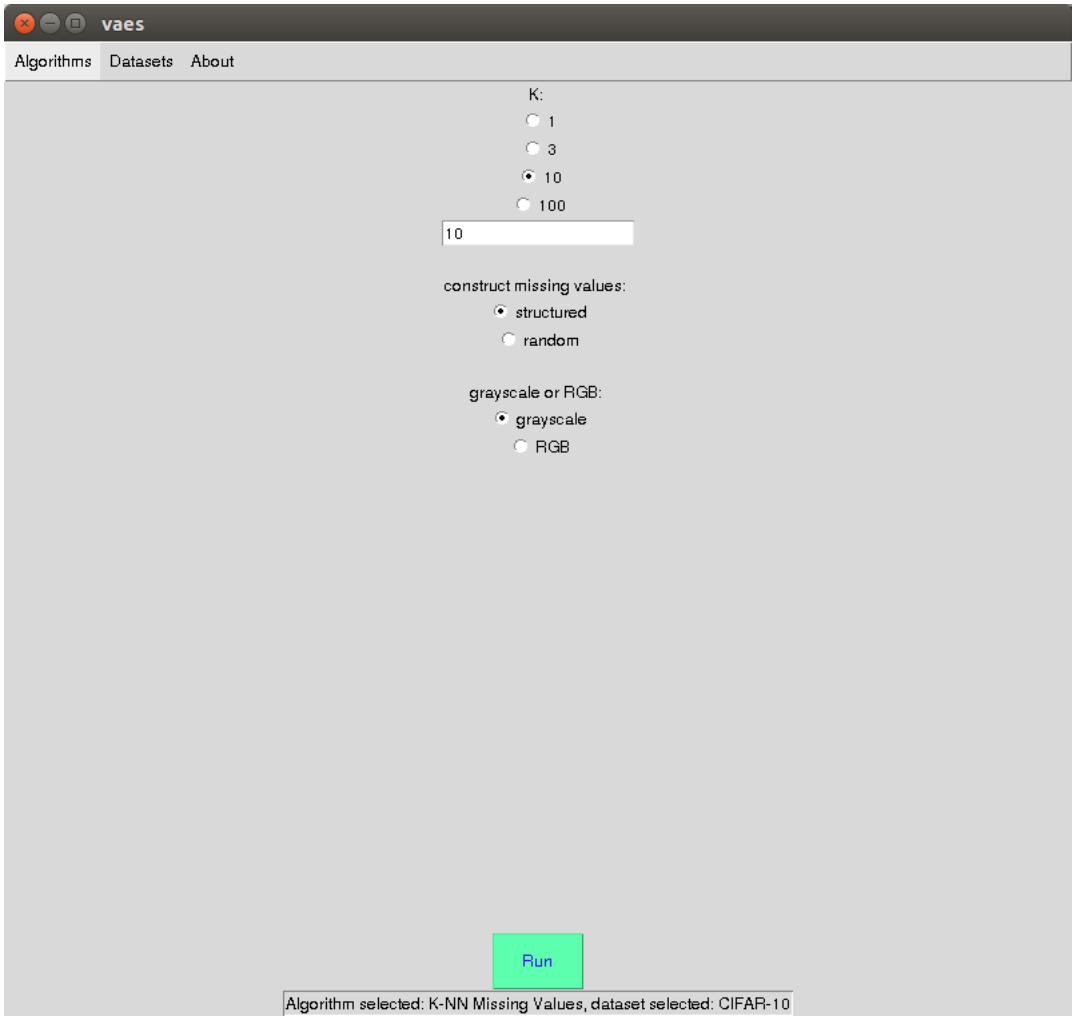


Figure 7: GUI K-NN Missing Values algorithm, CIFAR-10 dataset.

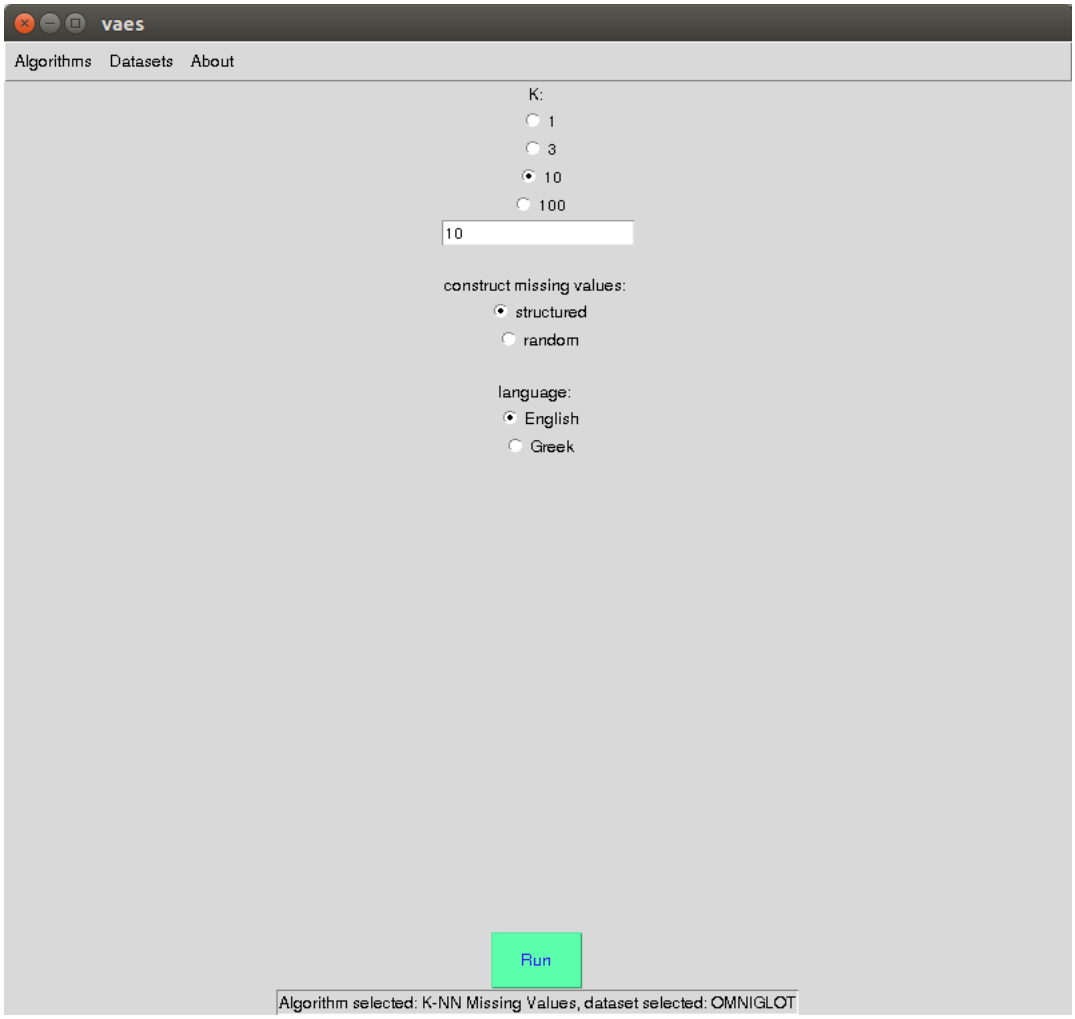


Figure 8: GUI K-NN Missing Values algorithm, OMNIGLOT dataset.

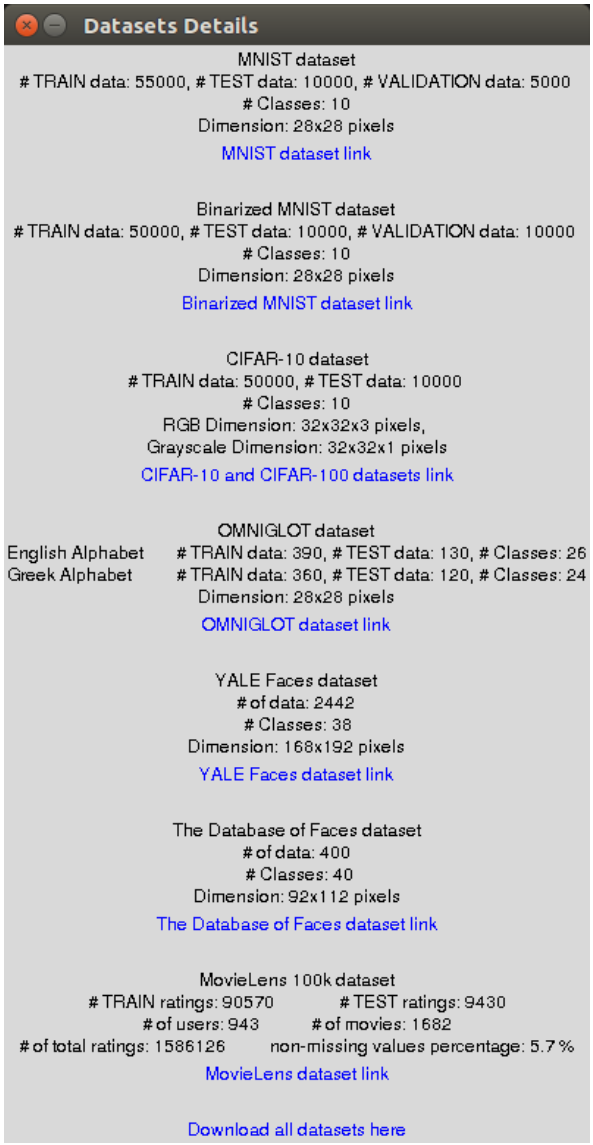


Figure 9:
GUI datasets details.

Figure 10: GUI About.

GUI Source Code in Python

```
import tkinter as tk
import webbrowser
import os


__author__ = 'c.kormaris'

# create window and set title
root = tk.Tk(className='vaes')

# change window size
root.geometry('800x700')
# change icon
icon = tk.PhotoImage(file='icons/vaes.png')
root.tk.call('wm', 'iconphoto', root._w, icon)

python_script_folder = tk.StringVar(root, '')
python_script_file = tk.StringVar(root, '')

algorithm_selected = ''
dataset_selected = ''

algorithm_names = ['VAE in TensorFlow', 'VAE in PyTorch', 'VAE in Keras',
                   'VAE Missing Values in TensorFlow', 'VAE Missing Values in PyTorch',
                   'K-NN Missing Values']
algorithms = ['VAEsInTensorFlow', 'VAEsInPyTorch', 'VAEsInKeras',
              'VAEsMissingValuesInTensorFlow', 'VAEsMissingValuesInPyTorch',
              'kNNMissingValues']

dataset_names = ['MNIST', 'Binarized MNIST', 'CIFAR-10', 'OMNIGLOT', 'YALE Faces', 'The Database of Faces', 'MovieLens']
datasets = ['mnist', 'binarized_mnist', 'cifar10', 'omniglot', 'yale_faces', 'orl_faces', 'movielens']


#####


def run(variables):
    arguments = []
    for variable in variables:
        arguments.append(variable.get())
    arguments = ' '.join(arguments)
    python_script = './' + python_script_folder.get() + '/' + python_script_file.get() + '.py'
    print('Running ' + python_script + '...')
    print('arguments: ' + arguments)
    # os.environ('PATH')
    os.system('python ' + python_script + ' ' + arguments)
    print('')


variables = []
```

```
51
52 algorithm_is_selected = False
53 dataset_is_selected = False
54
55
56 def hide_welcome_frame():
57     welcomeFrame.pack_forget()
58
59
60 def get_algorithm_name(algorithm):
61     for i, algorithm_name in enumerate(algorithm_names):
62         if algorithm == algorithms[i]:
63             return algorithm_names[i]
64
65
66 def get_dataset_name(dataset):
67     for i, dataset_name in enumerate(dataset_names):
68         if dataset == datasets[i]:
69             return dataset_names[i]
70
71
72 def check_algorithm_and_show_vae_frame():
73     hide_welcome_frame()
74     cifarDatasetFrame.pack_forget()
75     omniglotDatasetFrame.pack_forget()
76     missingValuesFrame.pack_forget()
77     kNNFrame.pack_forget()
78     runFrame.pack_forget()
79     vaeFrame.pack()
80     global algorithm_is_selected
81     algorithm_is_selected = True
82     global algorithm_selected
83     algorithm_selected = get_algorithm_name(python_script_folder.
       get())
84     global dataset_selected
85     dataset_selected = get_dataset_name(python_script_file.get())
86     if 'keras' in algorithm_selected.lower():
87         vae_empty_line_label.pack_forget()
88         learning_rate_label.pack_forget()
89         learning_rate_frame.pack_forget()
90         learning_rate_text.pack_forget()
91     else:
92         vae_empty_line_label.pack_forget()
93         learning_rate_label.pack()
94         learning_rate_frame.pack()
95         learning_rate_text.pack()
96         vae_empty_line_label.pack()
97     if 'missing' in algorithm_selected.lower():
98         datasetsMenu.entryconfig(6, state='normal')  # enable '
       MovieLens' dataset
99     else:
100        datasetsMenu.entryconfig(6, state='disabled')  # disable '
       MovieLens' dataset
101    if 'missing' in algorithm_selected.lower() and \
102            (not python_script_file.get() == 'movielens' and not
       python_script_file.get() == ''):
103        missingValuesFrame.pack()
```

```
104     if python_script_file.get() == 'cifar10':
105         cifarDatasetFrame.pack()
106     elif python_script_file.get() == 'omniglot':
107         omniglotDatasetFrame.pack()
108     if algorithm_is_selected and dataset_is_selected:
109         runFrame.pack(side='bottom')
110         status.config(text='Algorithm selected: ' +
    algorithm_selected + ', dataset selected: ' + dataset_selected)
111     global variables
112     variables = [latent_dim_var, epochs_var, batch_size_var]
113     if 'keras' not in algorithm_selected.lower():
114         variables.extend([learning_rate_var])
115     if 'missing' in algorithm_selected.lower() and dataset_selected
     is not None and 'movielens' not in dataset_selected.lower():
116         variables.extend([missing_values_var])
117     if python_script_file.get() == 'cifar10':
118         variables.extend([RGBOrGrayscale_var])
119     elif python_script_file.get() == 'omniglot':
120         variables.extend([language_var])
121
122
123 def check_algorithm_and_show_knn_frame():
124     hide_welcome_frame()
125     cifarDatasetFrame.pack_forget()
126     omniglotDatasetFrame.pack_forget()
127     missingValuesFrame.pack_forget()
128     vaeFrame.pack_forget()
129     runFrame.pack_forget()
130     kNNFrame.pack()
131     global algorithm_is_selected
132     algorithm_is_selected = True
133     global algorithm_selected
134     algorithm_selected = get_algorithm_name(python_script_folder.
    get())
135     global dataset_selected
136     dataset_selected = get_dataset_name(python_script_file.get())
137     if 'missing' in algorithm_selected.lower():
138         datasetsMenu.entryconfig(6, state='normal')  # enable '
    MovieLens' dataset
139     else:
140         datasetsMenu.entryconfig(6, state='disabled')  # disable '
    MovieLens' dataset
141     if 'missing' in algorithm_selected.lower() and \
142             (not python_script_file.get() == 'movielens' and not
    python_script_file.get() == ''):
143         missingValuesFrame.pack()
144     if python_script_file.get() == 'cifar10':
145         cifarDatasetFrame.pack()
146     elif python_script_file.get() == 'omniglot':
147         omniglotDatasetFrame.pack()
148     if algorithm_is_selected and dataset_is_selected:
149         runFrame.pack(side='bottom')
150         status.config(text='Algorithm selected: ' +
    algorithm_selected + ', dataset selected: ' + dataset_selected)
151     global variables
152     variables = [K_var]
153     if 'missing' in algorithm_selected.lower() and dataset_selected
```

```
     is not None and 'movielens' not in dataset_selected.lower():
154         variables.extend([missing_values_var])
155     if python_script_file.get() == 'cifar10':
156         variables.extend([RGBOrGrayscale_var])
157     elif python_script_file.get() == 'omniglot':
158         variables.extend([language_var])
159 
160 
161 def check_dataset():
162     cifarDatasetFrame.pack_forget()
163     omniglotDatasetFrame.pack_forget()
164     missingValuesFrame.pack_forget()
165     global dataset_is_selected
166     dataset_is_selected = True
167     global dataset_selected
168     if 'missing' in algorithm_selected.lower() and \
169             (not python_script_file.get() == 'movielens' and not
    python_script_file.get() == ''):
170         missingValuesFrame.pack()
171     dataset_selected = get_dataset_name(python_script_file.get())
172     if python_script_file.get() == 'cifar10' and not welcomeFrame.
    winfo_ismapped():
173         cifarDatasetFrame.pack()
174     elif python_script_file.get() == 'omniglot' and not
    welcomeFrame.winfo_ismapped():
175         omniglotDatasetFrame.pack()
176     if algorithm_is_selected and dataset_is_selected:
177         runFrame.pack(side='bottom')
178         status.config(text='Algorithm selected: ' +
    algorithm_selected + ', dataset selected: ' + dataset_selected)
179     global variables
180     if 'knn' not in python_script_folder.get().lower():
181         variables = [latent_dim_var, epochs_var, batch_size_var]
182         if 'keras' not in algorithm_selected.lower():
183             variables.extend([learning_rate_var])
184         if not algorithm_selected is None and 'missing' in
    algorithm_selected.lower() and 'movielens' not in
    dataset_selected.lower():
185             variables.extend([missing_values_var])
186         if python_script_file.get() == 'cifar10':
187             variables.extend([RGBOrGrayscale_var])
188         elif python_script_file.get() == 'omniglot':
189             variables.extend([language_var])
190     else:
191         variables = [K_var]
192         if not algorithm_selected is None and 'missing' in
    algorithm_selected.lower() and 'movielens' not in
    dataset_selected.lower():
193             variables.extend([missing_values_var])
194         if python_script_file.get() == 'cifar10':
195             variables.extend([RGBOrGrayscale_var])
196         elif python_script_file.get() == 'omniglot':
197             variables.extend([language_var])
198 
199 
200 def about_window():
201     window = tk.Toplevel(root)
```

```
202
203     # change title
204     window.wm_title('About')
205
206     window.resizable(False, False)
207
208     creator = tk.Label(window, text='Creator: Christos Kormaris')
209     creator.pack()
210     professor = tk.Label(window, text='Supervisor Professor: Dr.
        Michalis Titsias')
211     professor.pack()
212     thesis = tk.Label(window, text='Thesis on Variational
        Autoencoders & Missing Values Completion Algorithms')
213     thesis.pack()
214     university = tk.Label(window, text='Athens University of
        Economics & Business')
215     university.pack()
216     msc = tk.Label(window, text='MSc in Computer Science')
217     msc.pack()
218     date = tk.Label(window, text='Date: April 2018')
219     date.pack()
220
221     # change icon
222     icon = tk.PhotoImage(file='icons/info.png')
223     window.tk.call('wm', 'iconphoto', window._w, icon)
224
225     # make the child window transient to the root
226     window.transient(root)
227     window.grab_set()
228     root.wait_window(window)
229
230
231 def datasets_details_window():
232     window = tk.Toplevel(root)
233
234     # change title
235     window.wm_title('Datasets Details')
236
237     window.resizable(False, False)
238
239     mnist_ds_label = tk.Label(window, text='MNIST dataset\n# TRAIN
        data: 55000, # TEST data: 10000, # VALIDATION data: 5000 \n#
        Classes: 10\nDimension: 28x28 pixels')
240     mnist_ds_label.pack()
241     mnist_ds_link = tk.Label(window, text="MNIST dataset link", fg=
        "blue", cursor="hand2")
242     mnist_ds_link.pack()
243     mnist_ds_link.bind("<Button-1>", mnist_link_command)
244     empty_line_label = tk.Label(window, text='\r')
245     empty_line_label.pack()
246     binarized_mnist_ds_label = tk.Label(window, text='Binarized
        MNIST dataset\n# TRAIN data: 50000, # TEST data: 10000, #
        VALIDATION data: 10000 \n# Classes: 10\nDimension: 28x28 pixels
        ')
247     binarized_mnist_ds_label.pack()
248     binarized_mnist_ds_link = tk.Label(window, text="Binarized
        MNIST dataset link", fg="blue", cursor="hand2")
```

```
249     binarized_mnist_ds_link.pack()
250     binarized_mnist_ds_link.bind("<Button-1>",
        binarized_mnist_link_command)
251     empty_line_label = tk.Label(window, text='\r')
252     empty_line_label.pack()
253     cifar_10_ds_label = tk.Label(window, text='CIFAR-10 dataset\n#
        TRAIN data: 50000, # TEST data: 10000 \n# Classes: 10\nRGB
        Dimension: 32x32x3 pixels, \nGrayscale Dimension: 32x32x1
        pixels')
254     cifar_10_ds_label.pack()
255     cifar_10_link = tk.Label(window, text="CIFAR-10 and CIFAR-100
        datasets link", fg="blue", cursor="hand2")
256     cifar_10_link.pack()
257     cifar_10_link.bind("<Button-1>", cifar_link_command)
258     empty_line_label = tk.Label(window, text='\r')
259     empty_line_label.pack()
260     omniglot_ds_label = tk.Label(window, text='OMNIGLOT dataset\
        nEnglish Alphabet \t # TRAIN data: 390, # TEST data: 130, #
        Classes: 26\nGreek Alphabet \t # TRAIN data: 360, # TEST data:
        120, # Classes: 24\nDimension: 28x28 pixels')
261     omniglot_ds_label.pack()
262     omniglot_ds_link = tk.Label(window, text="OMNIGLOT dataset link
        ", fg="blue", cursor="hand2")
263     omniglot_ds_link.pack()
264     omniglot_ds_link.bind("<Button-1>", omniglot_link_command)
265     empty_line_label = tk.Label(window, text='\r')
266     empty_line_label.pack()
267     yale_ds_label = tk.Label(window, text='YALE Faces dataset\n# of
         data: 2442\n # Classes: 38\nDimension: 168x192 pixels')
268     yale_ds_label.pack()
269     yale_ds_link = tk.Label(window, text="YALE Faces dataset link",
         fg="blue", cursor="hand2")
270     yale_ds_link.pack()
271     yale_ds_link.bind("<Button-1>", yale_link_command)
272     empty_line_label = tk.Label(window, text='\r')
273     empty_line_label.pack()
274     the_db_of_faces_ds_label = tk.Label(window, text='The Database
        of Faces dataset\n# of data: 400\n # Classes: 40\nDimension: 92
        x112 pixels')
275     the_db_of_faces_ds_label.pack()
276     the_db_of_faces_ds_link = tk.Label(window, text="The Database
        of Faces dataset link", fg="blue", cursor="hand2")
277     the_db_of_faces_ds_link.pack()
278     the_db_of_faces_ds_link.bind("<Button-1>",
        the_db_of_faces_link_command)
279     empty_line_label = tk.Label(window, text='\r')
280     empty_line_label.pack()
281     movielens_ds_label = tk.Label(window, text='MovieLens 100k
        dataset\n# TRAIN ratings: 90570 \t # TEST ratings: 9430\n# of
        users: 943 \t # of movies: 1682\n# of total ratings: 1586126 \t
         non-missing percentage: 5.7 %')
282     movielens_ds_label.pack()
283     movielens_ds_link = tk.Label(window, text="MovieLens dataset
        link", fg="blue", cursor="hand2")
284     movielens_ds_link.pack()
285     movielens_ds_link.bind("<Button-1>", movielens_link_command)
286     empty_line_label = tk.Label(window, text='\r')
```

```
287     empty_line_label.pack()
288     download_all_datasets_link = tk.Label(window, text="Download
        all datasets here", fg="blue", cursor="hand2")
289     download_all_datasets_link.pack()
290     download_all_datasets_link.bind("<Button-1>",
        download_all_datasets_command)
291
292     # change icon
293     icon = tk.PhotoImage(file='icons/help.png')
294     window.tk.call('wm', 'iconphoto', window._w, icon)
295
296     # make the child window transient to the root
297     window.transient(root)
298     window.grab_set()
299     root.wait_window(window)
300
301
302 def mnist_link_command(event):
303     webbrowser.open_new(r"http://yann.lecun.com/exdb/mnist")
304
305
306 def binarized_mnist_link_command(event):
307     webbrowser.open_new(r"https://github.com/yburda/iwae/tree/
        master/datasets/BinaryMNIST")
308
309
310 def cifar_link_command(event):
311     webbrowser.open_new(r"https://www.cs.toronto.edu/~kriz/cifar.
        html")
312
313
314 def omniglot_link_command(event):
315     webbrowser.open_new(r"https://github.com/yburda/iwae/tree/
        master/datasets/OMNIGLOT")
316
317
318 def yale_link_command(event):
319     webbrowser.open_new(r"https://vision.ucsd.edu/content/extended-
        yale-face-database-b-b")
320
321
322 def the_db_of_faces_link_command(event):
323     webbrowser.open_new(r"http://www.cl.cam.ac.uk/research/dtg/
        attarchive/facedatabase.html")
324
325
326 def movielens_link_command(event):
327     webbrowser.open_new(r"https://grouplens.org/datasets/movielens"
        )
328
329
330 def download_all_datasets_command(event):
331     webbrowser.open_new(r"https://www.dropbox.com/sh/
        ucvad0dkcbxuyho/AAAjjrRPYiGLLPc_VKru4-Uva?dl=0")
332
333
334 #####
```

```
335
336
337 # Frames #
338 welcomeFrame = tk.Frame(root)
339 vaeFrame = tk.Frame(root)
340 kNNFrame = tk.Frame(root)
341 cifarDatasetFrame = tk.Frame(root)
342 omniglotDatasetFrame = tk.Frame(root)
343 runFrame = tk.Frame(root)
344 missingValuesFrame = tk.Frame(root)
345
346 # Widgets #
347
348 # 1. welcomeFrame Widgets #
349 empty_line_label = tk.Label(welcomeFrame, text='\n')
350 empty_line_label.pack()
351
352 aueb_logo = tk.PhotoImage(file='icons/aueb_logo_256.png')
353 image_label = tk.Label(welcomeFrame, image=aueb_logo, anchor=tk.
        CENTER)
354 image_label.pack()
355
356 welcome_label = tk.Label(welcomeFrame, text='Welcome to the
        Variational autoencoders graphical user interface.')
357 welcome_label.pack()
358 instructions_label = tk.Label(welcomeFrame, text='Please select an
        algorithm and a dataset from the dropdown menus at the top.')
359 instructions_label.pack()
360
361 # show welcomeFrame
362 welcomeFrame.pack()
363
364 # 2. vaeFrame Widgets #
365 # Tkinter variables
366 latent_dim_var = tk.StringVar(root, 64)
367 epochs_var = tk.StringVar(root, 100)
368 learning_rate_var = tk.StringVar(root, 0.01)
369 batch_size_var = tk.StringVar(root, 250)
370 K_var = tk.StringVar(root, 10)
371 RGBOrGrayscale_var = tk.StringVar(root, 'grayscale')
372 language_var = tk.StringVar(root, 'english')
373 missing_values_var = tk.StringVar(root, 'structured')
374
375 latent_dim_label = tk.Label(vaeFrame, text='latent dimension: ')
376 latent_dim_label.pack()
377 for i in [20, 64]:
378     tk.Radiobutton(vaeFrame,
379                    text=i,
380                    padx=2,
381                    variable=latent_dim_var,
382                    value=i).pack(anchor=tk.CENTER)
383 latent_dim_text = tk.Entry(vaeFrame, textvariable=latent_dim_var)
384 latent_dim_text.pack()
385
386 vae_empty_line_label = tk.Label(vaeFrame, text='\r')
387 vae_empty_line_label.pack()
388
```

```
389 epochs_label = tk.Label(vaeFrame, text='epochs: ')
390 epochs_label.pack()
391 for i in [20, 50, 100, 200]:
392     tk.Radiobutton(vaeFrame,
393                    text=i,
394                    padx=2,
395                    variable=epochs_var,
396                    value=i).pack(anchor=tk.CENTER)
397 epochs_text = tk.Entry(vaeFrame, textvariable=epochs_var)
398 epochs_text.pack()
399
400 vae_empty_line_label = tk.Label(vaeFrame, text='\r')
401 vae_empty_line_label.pack()
402
403 batch_size_label = tk.Label(vaeFrame, text='batch size: ')
404 batch_size_label.pack()
405 for value in [250, 500, 'N']:
406     tk.Radiobutton(vaeFrame,
407                    text=value,
408                    padx=2,
409                    variable=batch_size_var,
410                    value=value).pack(anchor=tk.CENTER)
411 batch_size_text = tk.Entry(vaeFrame, textvariable=batch_size_var)
412 batch_size_text.pack()
413
414 vae_empty_line_label = tk.Label(vaeFrame, text='\r')
415 vae_empty_line_label.pack()
416
417 learning_rate_label = tk.Label(vaeFrame, text='learning rate: ')
418 learning_rate_label.pack()
419 learning_rate_frame = tk.Frame(vaeFrame)
420 learning_rate_frame.pack()
421 for value in [0.1, 0.01, 0.001]:
422     tk.Radiobutton(learning_rate_frame,
423                    text=value,
424                    padx=2,
425                    variable=learning_rate_var,
426                    value=value).pack(anchor=tk.CENTER)
427 learning_rate_text = tk.Entry(vaeFrame, textvariable=
        learning_rate_var)
428 learning_rate_text.pack()
429
430 # 3. kNNFrame Widgets #
431 k_label = tk.Label(kNNFrame, text='K: ')
432 k_label.pack()
433 for value in [1, 3, 10, 100]:
434     tk.Radiobutton(kNNFrame,
435                    text=value,
436                    padx=2,
437                    variable=K_var,
438                    value=value).pack(anchor=tk.CENTER)
439 k_text = tk.Entry(kNNFrame, textvariable=K_var)
440 k_text.pack()
441
442 vae_empty_line_label = tk.Label(vaeFrame, text='\r')
443 vae_empty_line_label.pack()
444
```

```
445 knn_empty_line_label = tk.Label(kNNFrame, text='\r')
446 knn_empty_line_label.pack()
447
448 # 4. cifarDatasetFrame Widgets #
449 cifar_label = tk.Label(cifarDatasetFrame, text='grayscale or RGB: '
    )
450 cifar_label.pack()
451 for value in ['grayscale', 'RGB']:
452     tk.Radiobutton(cifarDatasetFrame,
453                    text=value,
454                    padx=2,
455                    variable=RGBOrGrayscale_var,
456                    value=value.lower()).pack(anchor=tk.CENTER)
457
458 # 5. omniglotDatasetFrame Widgets #
459 cifar_label = tk.Label(omniglotDatasetFrame, text='language: ')
460 cifar_label.pack()
461 for value in ['English', 'Greek']:
462     tk.Radiobutton(omniglotDatasetFrame,
463                    text=value,
464                    padx=2,
465                    variable=language_var,
466                    value=value.lower()).pack(anchor=tk.CENTER)
467
468 # 6. missing values Widgets #
469 missing_values_label = tk.Label(missingValuesFrame, text='missing
    values construction: ')
470 missing_values_label.pack()
471 for value in ['structured', 'random']:
472     tk.Radiobutton(missingValuesFrame,
473                    text=value,
474                    padx=2,
475                    variable=missing_values_var,
476                    value=value.lower()).pack(anchor=tk.CENTER)
477 empty_line_label = tk.Label(missingValuesFrame, text='\r')
478 empty_line_label.pack()
479
480 # Status Bar #
481
482 status = tk.Label(runFrame, bd=1, relief=tk.SUNKEN, anchor=tk.S)
483 status.pack(side=tk.BOTTOM, fill=tk.X)
484
485
486 # Menus #
487
488 menu = tk.Menu(root)
489 root.config(menu=menu)
490
491 algorithmsMenu = tk.Menu(menu, tearoff=False)
492 menu.add_cascade(label='Algorithms', menu=algorithmsMenu)  # adds
    drop-down menu
493 for i in range(len(algorithms)):
494     if algorithms[i] == 'VAEsMissingValuesInTensorFlow':
495         algorithmsMenu.add_separator()
496     if 'knn' in algorithms[i].lower():
497         algorithmsMenu.add_radiobutton(label=algorithm_names[i],
    variable=python_script_folder, value=algorithms[i],
```

```
498                                         command=
    check_algorithm_and_show_knn_frame)
499     else:
500         algorithmsMenu.add_radiobutton(label=algorithm_names[i],
    variable=python_script_folder, value=algorithms[i],
501                                         command=
    check_algorithm_and_show_vae_frame)
502
503 datasetsMenu = tk.Menu(menu, tearoff=False)
504 menu.add_cascade(label='Datasets', menu=datasetsMenu)  # adds drop-
    down menu
505 for i in range(len(datasets)):
506     if datasets[i] != 'movielens':
507         datasetsMenu.add_radiobutton(label=dataset_names[i],
    variable=python_script_file, value=datasets[i],
508                                      command=check_dataset)
509     else:
510         # Leave 'MovieLens' dataset disabled initially.
511         datasetsMenu.add_radiobutton(label=dataset_names[i],
    variable=python_script_file, value=datasets[i],
512                                      command=check_dataset, state='
    disabled')
513
514 aboutMenu = tk.Menu(menu, tearoff=False)
515 menu.add_cascade(label='About', menu=aboutMenu)  # adds drop-down
    menu
516 aboutMenu.add_command(label='About', command=about_window)
517 aboutMenu.add_command(label='Datasets Details', command=
    datasets_details_window)
518 aboutMenu.add_command(label="Exit", command=root.quit)
519
520 runButton = tk.Button(runFrame, text='Run', fg='#340DFD', bg='#5
    BFFAC',
521                       height=2, width=6, command=lambda: run(
    variables))
522 runButton.pack(side=tk.BOTTOM)
523
524
525 #####
526
527
528 # center the window on screen
529 def center(win):
530     win.update_idletasks()
531     width = win.winfo_width()
532     height = win.winfo_height()
533     x = (win.winfo_screenwidth() // 2) - (width // 2)
534     y = (win.winfo_screenheight() // 2) - (height // 2)
535     win.geometry('{}x{}+{}+{}'.format(width, height, x, y))
536
537
538 center(root)
539 root.mainloop()
```

Listing 4: GUI Source Code in Python