**Міністерство освіти і науки України**

**Національний університет «Запорізька політехніка»**

Кафедра програмних засобів

**ЗВІТ**

Дисципліна «Розробка прикладних програм»

Робота №5

Тема «Машинне навчання»

**Виконав варіант 19**

Студент КНТ-122 Онищенко О. А.

**Прийняли**

Викладач Дейнега Л. Ю.

2024

Мета роботи

Ознайомитись з основними пакетами, які використовуються для машинного навчання в програмах, написаних мовою Python.

Навчитися розробляти сучасні інтелектуальні системи з використанням методів машинного навчання.

Індивідуальне завдання

Розробити програмне забезпечення, яке дозволяє виділити в даному зображені набір основних кольорів. Кількість кольорів може бути вибрана попередньо з заданого діапазону. Алгоритм кластеризації обрати самостійно. Для обраного зображення визначити також набір зображень з подібними кольорами. Банк зображень повинен формуватися автоматизовано.

Текст файлу

from sklearn.cluster import KMeans

import matplotlib.image as img

from nicegui import ui, app

import numpy as np

import os

class Color:

def \_\_init\_\_(

self,

r: int,

g: int,

b: int,

):

self.red=int(r)

self.green=int(g)

self.blue=int(b)

def \_\_str\_\_(self):

return f'{self.red} {self.green} {self.blue}'

def \_\_repr\_\_(self):

return f'{self.red} {self.green} {self.blue}'

def get\_html\_rgb\_value(self):

return f'rgb({self.red},{self.green},{self.blue})'

def get\_average(self):

return (self.red+self.green+self.blue)//3

def get\_n\_most\_common\_colors(

image\_path:str,

colors\_count:int,

):

'''

< image\_path: str

full image path as string

< colors\_count: int

number of most common colors to return for a given image

> palette: list[list[int]]

has length of colors\_count

each element is a color in RGB format:

red\_value, green\_value, blue\_value

'''

image=img.imread(image\_path)

w,h,d=tuple(image.shape)

pixel=np.reshape(image,(w\*h,d))

model=KMeans(n\_clusters=colors\_count,random\_state=40).fit(pixel)

palette=np.uint8(model.cluster\_centers\_)

res=[]

for c in palette:

r,g,b=c

res.append(Color(r,g,b))

return res

def get\_colors\_dataset(

images\_folder:str,

image\_files:list[str],

):

colors\_dataset=dict()

for image\_file\_name in image\_files:

image\_path=os.path.join(images\_folder,image\_file\_name)

this\_image\_colors=get\_n\_most\_common\_colors(

image\_path=image\_path,

colors\_count=colors\_count

)

colors\_dataset[image\_file\_name]=this\_image\_colors

return colors\_dataset

def get\_average\_colors(

colors\_dataset:dict,

):

def get\_average\_color(

colors:list[Color]

):

return round(sum([c.get\_average() for c in colors])/len(colors))

return {name:get\_average\_color(colors) for name,colors in colors\_dataset.items()}

ROOT\_FOLDER=os.path.dirname(os.path.abspath(\_\_file\_\_))

IMAGE\_FOLDER\_PATH=os.path.join(ROOT\_FOLDER,'images')

IMAGE\_FILE\_NAMES=os.listdir(IMAGE\_FOLDER\_PATH)

LABEL\_CLASSES='font-medium text-lg'

COLOR\_CLASSES='flex-1 py-7 rounded-md'

app.add\_media\_files('/images',IMAGE\_FOLDER\_PATH)

image\_file=IMAGE\_FILE\_NAMES[0]

colors\_count=3

image\_colors=get\_colors\_dataset(IMAGE\_FOLDER\_PATH,IMAGE\_FILE\_NAMES)

average\_image\_colors=get\_average\_colors(image\_colors)

def update\_image\_output():

results\_image.source=os.path.join(IMAGE\_FOLDER\_PATH,image\_file)

def update\_colors():

colors\_count\_label.text=f'The image has {colors\_count} most common color{"s" if colors\_count>1 else ""}'

image\_average\_color=average\_image\_colors[image\_file]

colors\_list=list(set((image\_name,color\_value) for image\_name,color\_value in average\_image\_colors.items() if color\_value<=image\_average\_color and image\_name!=image\_file))

closest\_colors=sorted(

colors\_list,

key=lambda color: color[1],

reverse=True

)

this\_image\_colors:list[Color]=image\_colors[image\_file]

closest\_color\_one.style(f'background: {this\_image\_colors[0].get\_html\_rgb\_value()};')

closest\_color\_two.style(f'background: {this\_image\_colors[1].get\_html\_rgb\_value()};')

closest\_color\_two.visible=colors\_count>=2

closest\_color\_three.style(f'background: {this\_image\_colors[2].get\_html\_rgb\_value()};')

closest\_color\_three.visible=colors\_count>=3

closest\_color\_image\_paths=[

os.path.join(IMAGE\_FOLDER\_PATH,this\_file\_name) for this\_file\_name,this\_color\_value in closest\_colors

]

similar\_image\_one.source=closest\_color\_image\_paths[0]

similar\_image\_two.source=closest\_color\_image\_paths[1]

similar\_image\_three.source=closest\_color\_image\_paths[2]

def update\_ui():

update\_image\_output()

update\_colors()

ui.label('Image to process').classes(LABEL\_CLASSES)

ui.select(

options=IMAGE\_FILE\_NAMES,

with\_input=True,

value=image\_file,

on\_change=update\_ui,

).classes('w-full').bind\_value(globals(),'image\_file')

ui.label('Number of colors').classes('mt-7 '+LABEL\_CLASSES)

ui.slider(

min=1,

max=3,

value=colors\_count,

on\_change=update\_ui,

).bind\_value(globals(),'colors\_count')

with ui.row().classes('flex justify-between w-full'):

ui.label(1)

ui.label(2)

ui.label(3)

ui.label(f'Results for {image\_file}').classes('mt-12 '+LABEL\_CLASSES)

results\_image=ui.image(os.path.join(IMAGE\_FOLDER\_PATH,image\_file)).classes('rounded-md object-center').style('max-height: 37vh;')

colors\_count\_label=ui.label(f'Most common colors ({colors\_count})').classes('mt-7 '+LABEL\_CLASSES)

with ui.row().classes('w-full flex gap-3'):

closest\_color\_one=ui.element('span').classes(COLOR\_CLASSES)

closest\_color\_two=ui.element('span').classes(COLOR\_CLASSES)

closest\_color\_three=ui.element('span').classes(COLOR\_CLASSES)

ui.label(f'Images with similar colors').classes('mt-7 '+LABEL\_CLASSES)

with ui.row().classes('w-full flex gap-3'):

similar\_image\_one=ui.image().classes('flex-1 rounded-md max-h-60 display-cover')

similar\_image\_two=ui.image().classes('flex-1 rounded-md max-h-60 display-cover')

similar\_image\_three=ui.image().classes('flex-1 rounded-md max-h-60 display-cover')

update\_ui()

ui.run(favicon='🖼️',title='Image Colors')

Результати виконання

Контрольні питання

Для чого призначена бібліотека pandas?

Які структури даних та для чого використовуються в пакеті pandas?

Які функції бібліотеки pandas можуть бути використані для введення/виведення даних?

Яким чином можна модифікувати дані, що зберігаються в структурах даних пакету pandas?

Яким чином виконується індексація даних у пакеті pandas?

Які фунції для роботи з декількома структурами даних підтримуються бібліотекою pandas?

Яким чином виконується оброблення часових рядів у бібліотеці pandas?

Яку галузь знань охоплює машинне навчання та які завдання воно дозволяє розв’язати?

Для виконання яких завдань призначення бібліотека scikit-learn?

Які засоби вибору властивостей надає бібліотека scikit-learn?

Які засоби попереднього оброблення даних надає бібліотека scikit-learn?

Які засоби кластеризації надає бібліотека scikit-learn?

Які засоби класифікації надає бібліотека scikit-learn?

Які методи навчання без учителя представлені в бібліотеці scikit-learn?

Яким чином інтегруються засоби бібліотек pandas та scikit-learn?

Які засоби перетворення даних використовуються в бібліотеці scikit-learn?

Які засоби пониження розмірності представлені в бібліотеці scikit-learn?

Які регресійні моделі можуть бути застосовані за допомогою засобів scikit-learn?

Які метрики надаються бібліотекою scikit-learn?

За допомогою яких засобів бібліотеки scikit-learn можна оцінити коваріацію?

Яким чином можна візуалізувати результати машинного навчання?

Для чого необхідні методи крос-валідації та які інструменти для цього використовуютьяс в бібліотеці scikit-learn?

Яким чином можна згенерувати дані для машинного навчання?

Дані яких типів та з яких джерел можна обробляти за допомогою засобів бібліотеки scikit-learn?