МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. І.СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра фізико-технічних засобів захисту інформації

Лабораторна робота № 3

з дисципліни: «Автоматизація обробки ІзОД»

Варіант №1

Керівник: Виконав:

Прогонов Дмитро Олександрович студент 5 курсу

групи ФЕ-91мп

Захищено з оцінкою Баліцький Олег Анатолійович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, підпис

Київ – 2020 р.

**Хід роботи**

1. Сформувати тестову вибірку зображень з вихідного пакета;

Блок коду програми мовою високого рівня:

from os import listdir

from matplotlib import image

import random

import numpy as np

from dataclasses import dataclass

# load images in a directory

random.seed(1)

random\_indexes = random.sample(range(25000), 250)

loaded\_images = list()

for i in range(250):

# load image

filename = 'im' + str(random\_indexes[i]) + '.jpg'

img\_data = image.imread('Q:/mirflickr25k/mirflickr/' + filename)

# store loaded image

loaded\_images.append(img\_data)

print('> loaded %s %s' % (filename, img\_data.shape))

2. Для кожного каналу кольору кожного зображення з тестового пакета обчислити наступні характеристики:

a. Математичне сподівання і дисперсію;

# Expected value & Variance

#RED

sum\_val = sum(values[RED])

M\_red = 0

for index in range(len(values[RED])):

p = (values[RED][index] / sum\_val)

M\_red += p \* index

D\_red = 0

for index in range(len(values[RED])):

p = (values[RED][index] / sum\_val)

D\_red += p \* ((index - M\_red) \*\* 2)

print("Red: Expected value - {0:.2f}, Variance - {1:.2f}"\

.format(M\_red, D\_red))

#GREEN

sum\_val = sum(values[GREEN])

M\_green = 0

for index in range(len(values[GREEN])):

p = (values[GREEN][index] / sum\_val)

M\_green += p \* index

D\_green = 0

for index in range(len(values[GREEN])):

p = (values[GREEN][index] / sum\_val)

D\_green += p \* ((index - M\_green) \*\* 2)

print("Green: Expected value - {0:.2f}, Variance - {1:.2f}"\

.format(M\_green, D\_green))

#BLUE

sum\_val = sum(values[BLUE])

M\_blue = 0

for index in range(len(values[BLUE])):

p = (values[BLUE][index] / sum\_val)

M\_blue += p \* index

D\_blue = 0

for index in range(len(values[BLUE])):

p = (values[BLUE][index] / sum\_val)

D\_blue += p \* ((index - M\_blue) \*\* 2)

print("Blue: Expected value - {0:.2f}, Variance - {1:.2f}"\

.format(M\_blue, D\_blue))

Вихідні дані:

*Red: Expected value - 113.95, Variance - 5852.65*

*Green: Expected value - 107.31, Variance - 5392.78*

*Blue: Expected value - 99.49, Variance - 5774.33*

b. Коефіцієнти асиметрії та ексцесу (нормалізований);

#Asymmetry and excess

Asym\_red = E\_operator(values[RED], M\_red, 3) / (D\_red \*\* (3 / 2))

Asym\_green = E\_operator(values[GREEN], M\_green, 3) / (D\_green \*\* (3 / 2))

Asym\_blue = E\_operator(values[BLUE], M\_blue, 3) / (D\_blue \*\* (3 / 2))

Excess\_red = E\_operator(values[RED], M\_red, 4) / (D\_red \*\* 2)

Excess\_green = E\_operator(values[GREEN], M\_green, 4) / (D\_green \*\* 2)

Excess\_blue = E\_operator(values[BLUE], M\_blue, 4) / (D\_blue \*\* 2)

print('Red: Asymmetry - {0:.3f}, Excess - {1:.3f}'\

.format(Asym\_red, Excess\_red))

print('Green: Asymmetry - {0:.3f}, Excess - {1:.3f}'\

.format(Asym\_green, Excess\_green))

print('Blue: Asymmetry - {0:.3f}, Excess - {1:.3f}'\

.format(Asym\_blue, Excess\_blue))

Вихідні дані:

*Red: Asymmetry - 0.174, Excess - 1.853*

*Green: Asymmetry - 0.275, Excess - 1.994*

*Blue: Asymmetry - 0.425, Excess - 2.024*

* 1. Використовуючи моделі SPAM і CC-PEV, розрахувати вектори характеристик кожного каналу кольору кожного зображення;

**SPAM:**

for c in (LogisticRegression, LinearRegression, SVC):

cls = LinearRegression(n\_jobs=-1)

cnf\_matrix = []

for i in range(N):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5)

cls.fit(X\_train, y\_train)

y\_predict = cls.predict(X\_test).astype(int)

#print(y\_predict)

cnf\_matrix.append(confusion\_matrix(y\_test, y\_predict.astype(bool).astype(int))) # .astype(int)

#print(confusion\_matrix(y\_test, y\_predict.astype(int)))

cnf\_matrix = np.mean(np.array(cnf\_matrix), axis=0)

plt.figure(figsize=(4, 3))

plot\_confusion\_matrix(cnf\_matrix, classes=['0', '1'],

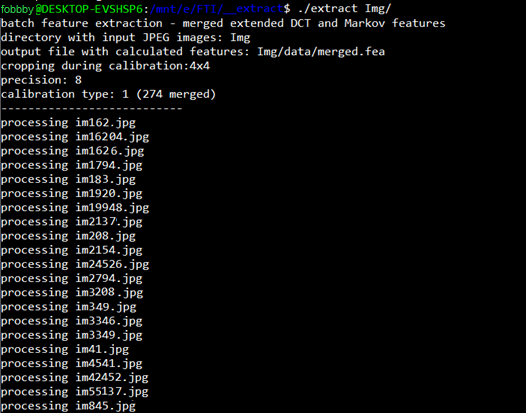
title=c.\_\_name\_\_)

plt.savefig("spam" + c.\_\_name\_\_ + ".png")

plt.show()

print(cnf\_matrix)

Використавши екстрактор CC-PEV для знаходження векторів ознак зображень - отримаємо:



Вектор ознак для кожного зображення сформовано та поміщено в файл merged.fea.

5. Побудувати вектор міток класів зображень. Назва цільового класу для кожного студента визначається згідно з позицією студента у списку групи. Для варіанту №1 цей тег – «explore»

Y = pd.Series()

for name in image\_names:

with codecs.open(os.path.join(DIRNAME\_TAGS, 'tags%d.txt' % (name,)), "r", "utf\_8\_sig") as file:

set\_tags = set(tag.strip() for tag in file.readlines())

#print('tags%d.txt' % (name,), ' ',set\_tags)

Y.at[name] = int(MAIN\_TAG in set\_tags)

Y = pd.DataFrame(Y, columns=['target'])

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Формування пакетів даних:

num = 0

d = []

for image\_name in image\_names:

image = np.array(Image.open(os.path.join(DIRNAME\_IMAGES, 'im%d.jpg' % (image\_name))))

data = []

for name, num in COLOR.items():

a = image[:, :, num].ravel()

data.extend([np.mean(a), np.var(a), sp.stats.skew(a), sp.stats.kurtosis(a)])

d.append([image\_name, \*data])

data\_static = pd.DataFrame.from\_records(d)

data\_static = data\_static.rename(columns={0:'name'}).set\_index('name')

1. Псевдовипадковий чином розділити вихідний пакет зображень на 2 рівні частини (тестова і контрольна підвибірки). З використанням тестової підвибірки провести настройку наступних класифікаторів:

def plot\_confusion\_matrix(cm, classes,

normalize=True,

title='Confusion matrix',

cmap=plt.cm.Blues):

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes)

plt.yticks(tick\_marks, classes)

if normalize:

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

print("Normalized confusion matrix")

else:

print('Confusion matrix, without normalization')

print(cm)

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, round(cm[i, j], 3),

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

font = {'size' : 15}

plt.rc('font', \*\*font)

а-с. Лінійна та Логістична регресія;

for c in (LogisticRegression, LinearRegression, SVC):

cls = c()

cnf\_matrix = []

for i in range(N):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5)

cls.fit(X\_train, y\_train)

y\_predict = cls.predict(X\_test)

cnf\_matrix.append(confusion\_matrix(y\_test, y\_predict.astype(bool).astype(int)))

cnf\_matrix = np.mean(np.array(cnf\_matrix), axis=0)

plt.figure(figsize=(4, 3))

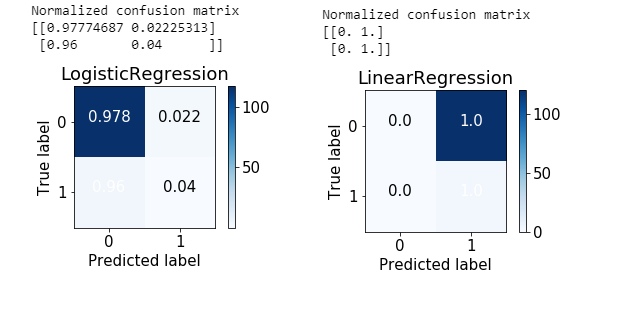
plot\_confusion\_matrix(cnf\_matrix, classes=['0', '1'],

title=c.\_\_name\_\_)

plt.savefig("static\_" + c.\_\_name\_\_ + ".png")

plt.show()

Вихідні дані:



**Висновки**

В даній лабораторній роботі було проаналізовано вибірку з 250 зображень датасету MIRFlickr-20k. Було знайдено що всі канали охоплюють увесь спектр значень. Було знайдено мат. очікування – 114 для чевоного каналу, 107 для зеленого і 99 для синьогоканалу відповідно і дисперсію 5852.65, 5392.78 та 5774.33.

Було проведено формування ознак на основі CC-PEV екстрактора та сформовано вектори данних для подальшого аналізу. Також всі дані було упаковано в матрицю ознак для тренування моделей.

Використовуючи мітку цільового класу «explore» було сформовано дані для розрізнення що відповідають та не відповідають моделі.

Після цього було натреновано моделі Лінійної та Логістичної Регресії для розрізняння. За даними таблиць результатів бачимо що він складав 0.978 та 1.0 відповідно для двох моделей.