Міністерство освіти і науки України

НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського»

Фізико-технічний інститут

Лабораторна робота №3

З дисципліни

«Автоматизація обробки ІзОД»

Варіант 7

**Виконав:**

Студент 5 курсу ФТІ

групи ФЕ-91мп

Мазурок В. О.

**Перевірив:**

Прогонов Д. О.

Київ-2020

**І. Підготовка**

**Вхідні дані**

Тестовий пакет – MIRFlickr-20k (https://press.liacs.nl/mirflickr/#sec\_download)

Вибірка зображень – 250 зображень;

Формування вибірки зображень – псевдовипадкове, з використанням генератора Мерсена (стартове значення співпадає з номером студента в загальному списку групи) за модулем кількості зображень в тестовому пакеті.

**Завдання**

1. Сформувати тестову вибірку зображень з вихідного пакета;
2. Для кожного каналу кольору кожного зображення з тестового пакета обчислити наступні характеристики:
   1. Математичне сподівання і дисперсію;
   2. Коефіцієнти асиметрії та ексцесу (нормалізований);
3. Використовуючи моделі SPAM і CC-PEV (<http://dde.binghamton.edu/download/feature_extractors/>), розрахувати вектора характеристик кожного каналу кольору кожного зображення;
4. Отримані параметри зображень упакувати в матрицю ознак (кожен рядок відповідає результатам для окремого каналу кольору тестового зображення, кожен стовпець - параметру зображення). Матриці ознак побудувати окремо для статистичних характеристик зображень (1-4 центральні моменти), а також моделей SPAM и CC-PEV;
5. Побудувати вектор міток класів зображень. Назва цільового класу для кожного студента визначається згідно з позицією студента у списку групи (див. Вкладений файл - наприклад, для першого студента в списку групи цільової клас "explore", для другого студента - мітка "sky", для третього студента - мітка "nikon" і т.д.);
6. псевдовипадковий чином розділити вихідний пакет зображень на 2 рівні частини (тестова і контрольна підвибірки). З використанням тестової підвибірки провести настройку наступних класифікаторів:
   1. Лінійна регресія;
   2. Робастна регресія;
   3. Логістична регресія;
   4. Метод опорных векторів (SVM);
7. Використовуючи налаштовані класифікатори з п. 6 провести обробки зображень з контрольного підпакету. Оцінити ймовірності правильної класифікації (0 і 1 класи), а також ймовірності помилок першого (помилкове спрацьовування) і другого (пропуск цілі) роду;
8. Повторити пп. 6-10 разів для отримання усередненої точності класифікації

**ІI. Хід роботи**

Роботу виконуватимемо мовою Python за допомогою блокового інтерпритатора Jupyter. Також в роботі будуть використані такі бібліотеки як:

* Os
* Matplotlib
* Numpy
* Scipy
* Pandas
* Sklearn

1. **Формування тестової вибірки зображень з вихідного пакета**

Для цього скористаємося функцією numpy.random.choices() що обирає випадкові числа з переданого масива за допомогою генератора Мерсена. Також задамо початкове значення варіату за допомогою функції numpy.random.RandomState()

np.random.RandomState(5)

random\_indexes = np.random.choice(range(25000), 250)

loaded\_images = list()

for i in range(250):

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

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

loaded\_images.append(img\_data)

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

Після цього ориманий масив зображень буде знаходитись в loaded\_images в виді двомірного масиву з трьома значеннями яскравості в кожій комірці.

Тепер сформуємо матрицю для збору статистичних даних. Для цього створимо двомірний numpy масив на три рядки для кожного каналу кольору та на 256 стовпчиків, що відповідатиме кількості пікселів відповідної яскравості.

values = np.zeros((3, 256))

index = 0

for image in loaded\_images:

for i in range(image.shape[0]):

for j in range(image.shape[1]):

values[0][image[i][j][0]] += 1

values[1][image[i][j][1]] += 1

values[2][image[i][j][2]] += 1

index += 1

if (index % 10) == 0:

print('> processed %i images' % (index))

1. **Знаходження статистичних даних**
2. **Математичне сподівання і дисперсія**

Для знаходження скористаємось відповідними формулами:



- Математичне очікування

- Дисперсія

Де xi наше значення яскравості, а pi – ймовірність її появи. pi можна знайти як кількість пікселів даної яскравості поділену на всю кількуість пікселів

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:\tmat ochikuvannya - {0:.2f},\tdispersiya - {1:.2f}".format(M\_red, D\_red))

Отримуємо:

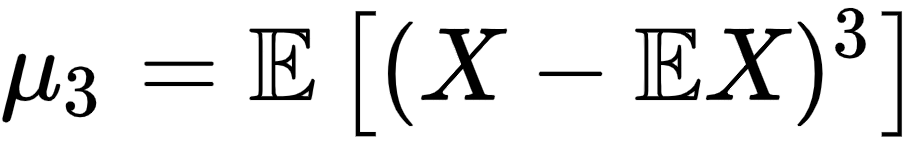
**Red: mat ochikuvannya - 108.70, dispersiya - 6163.60**

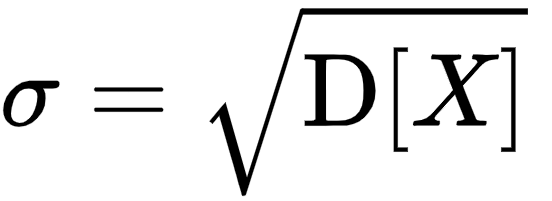
**Green: mat ochikuvannya - 99.80, dispersiya - 5495.78**

**Blue: mat ochikuvannya - 89.87, dispersiya - 5847.97**

**b. Коефіцієнти асиметрії та ексцесу**

Для цього також використаємо відповідні формули коефіцієнту асиметрії:





Для спрощення напишемо загальну формулу для оператора Е, що приймає степінь та мат очікування.

def E\_operator(arr\_values, M, power):

sum\_val = sum(arr\_values)

ans = 0

for index in range(len(arr\_values)):

p = (arr\_values[index] / sum\_val)

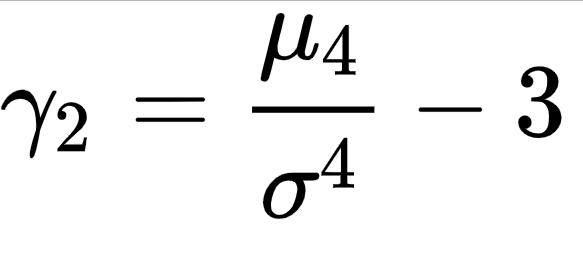
ans += p \* ((index - M) \*\* power)

return ans

Тепер формула отримає вигляд:

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

Та формули коефіцієнту ексцесу:



Ekscess\_red = E\_operator(values[RED], M\_red, 4) / (D\_red \*\* 2) - 3

Отримуємо:

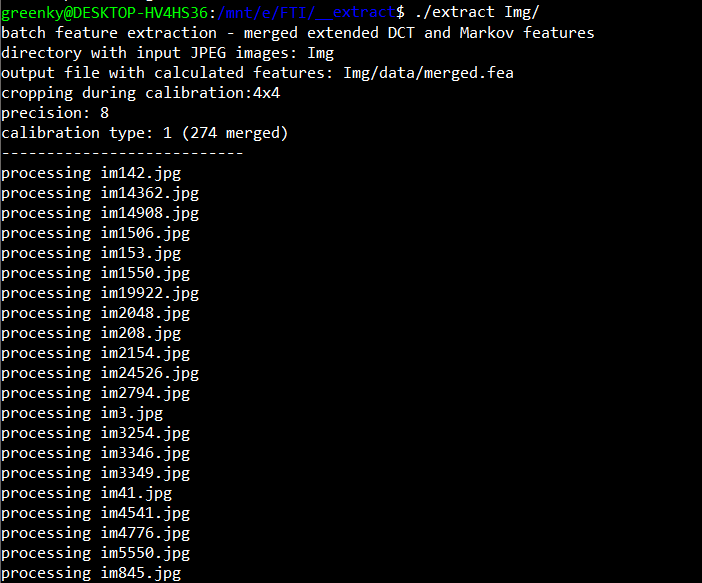
**Red: Asimmetriya - 0.252, Ekscess - -1.174**

**Green: Asimmetriya - 0.404, Ekscess - -0.958**

**Blue: Asimmetriya - 0.613, Ekscess - -0.826**

* 1. **Знаходження векторів ознак за допомогою функції CC-PEV**

Використаємо екстрактор CC-PEV для знаходження векторів ознак зображень:



Після виконання пограми буде сформовано вектор ознак для кожного фото та поміщено в файл merged.fea.

**5. Побудувати вектор міток класів зображень**

Тепер сформуємо вектори з мітками мого варіанту. В даному випадку це «portrait». Теги отримуємо з папки tags пакету.

Y = pd.Series()

for name in image\_names:\n

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())

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

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

Тепер сформуємо пакети даних та міток для навчання:

data\_static = DataFrame()

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. **Навчання моделі на першій вибірці**

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)

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)

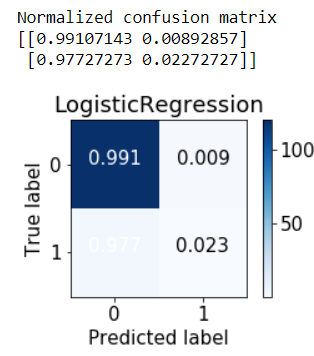
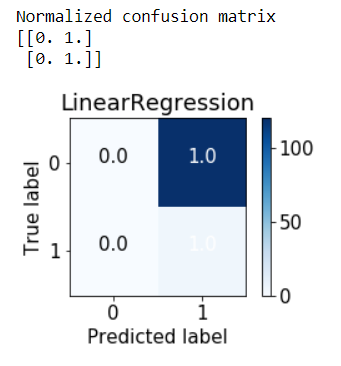
plt.figure(figsize=(4, 3))

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

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

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

plt.show()



**ІІІ. 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, 105 для зеленого і 96 для синьогоканалу відповідно і дисперсію 6024.15, 5724.37 та 5948.75.

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 «portrait» було сформовано дані для розрізняння що відповідають та не відповідають моделі.

Післф цього було натреновано моделі Лінійної та Логістичної Регресії для розрізняння. За даними таблиць результатів бачимо що він складав 0.991 та 1 відповідно для двох моделей.