МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра фізико-технічних засобів захисту інформації

Лабораторна робота № 3

з дисципліни: «Автоматизація обробки ІзОД»

Керівник: Виконав:

Прогонов Дмитро Олександрович студент 5 курсу групи ФЕ-91мп

Захищено з оцінкою Павлусь Олександр Сергійович

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

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

дата, підпис

Київ – 2020 р.

Завдання роботи

1. Сформировать тестовую выборку изображений из исходного пакета;
2. Для каждого канала цвета каждого изображения из тестового пакета вычислить следующие характеристики:
   1. Математическое ожидание и дисперсию;
   2. Коэффициенты асимметрии и эксцесса (нормализованный);
3. Используя модели SPAM и CC-PEV (<http://dde.binghamton.edu/download/feature_extractors/>), рассчитать вектора характеристик каждого канала цвета каждого изображения;
4. Полученные параметры изображений упаковать в матрицу признаков (каждая строка соответствует результатам для отдельного канала цвета тестового изображения, каждый столбец - параметру изображения). Матрицы признаков построить отдельно для статистических характеристик изображений (1-4 центральные моменты), а также моделей SPAM и CC-PEV;
5. Построить вектор меток классов изображений. Название целевого класса для каждого студента определяется согласно позиции студента в списке группы (см. вложенный файл - например, для первого студента в списке группы целевой класс "explore", для второго студента - метка "sky", для третьего студента - метка "nikon" и т.д.);
6. Псевдослучайным образом разделить исходный пакет изображений на 2 равные части (тестовая и контрольная подвыборки). С использованием тестовой подвыборки провести настройку следующих классификаторов:
   1. Линейная регрессия;
   2. Робастная регрессия;
   3. Логистическая регрессия;
   4. Метод опорных векторов (SVM);
7. Используя настроенные классификаторы из п. 6 провести обработки изображений из контрольного подпакета. Оценить вероятности правильной классификации (0 и 1 классы), а также вероятности ошибок первого (ложное срабатывание) и второго (пропуск цели) рода;
8. Повторить пп. 6-7 10 раз для получения усредненной точности классификации изображений.

1) Використовуючи моделі SPAM і CC-PEV розрахувати вектора характеристик для кожного зображення, побудувати вектор міток класів зображень та упакувати дані в матрицю ознак окремо для SPAM та СС-PEV.

За допомогою моделі SPAM та СС-PEV було отримано набір ознак для перших 1000 зображень із тестової вибірки. Дані були занесені у таблицю. Де кожному рядку відповідає зображення, для якого розраховувався набір ознак, а у кожному стовпчику значення ознак для одного зображення.

В останньому стовпчику були записані мітки класів відповідно до заданого за умовою завдання тегу.

Для 8 варіанту заданий тег - “water”

В результаті було сформовано csv-таблицю із набором ознак, на якій виконали налаштування обраних класифікаторів.

2) З використанням тестової та контрольної вибірки налаштувати наступні класифікатори:

* Логістична регресія;
* Лінійна регресія;
* Метод опорних векторів (SVM);

Для налаштованих класифікаторів на моделі SPAM отримуємо такі результати:

# 1) Логістична регресія:

Процент верных предсказаний логистической регрессии: 95.0%

Ошибки 1 рода - 0.0%

Ошибки 2 рода - 5.0%

# 2) Лінійна регресія:

Процент верных предсказаний для LDA: 90.5%

Ошибки 1 рода - 4.5%

Ошибки 2 рода — 5.0%

# 3) Метод Опорних Векторів

Процент верных предсказаний для SVM: 95.0%

Ошибки 1 рода - 0.0%

Ошибки 2 рода — 5.0%

**ВИСНОВКИ**

Отже, у роботі були проаналізовані загальні методи класифікації об'єктів та систем. З використанням моделі SPAM було отримано матрицю ознак, за допомогою якої було налаштовано наступні класифікатори:

* Логістична регресія;
* Лінійна регресія;
* Метод опорних векторів (SVM).

В результаті отриманих даних було встановлено, що для заданої задачі краще за все підходять такі класифікатори:

* Логістична регресія — 95% вірних передбачень;
* Лінійна регресія — 90.5% вірних передбачень;
* Метод опорних векторів (SVM) — 95% вірних передбачень;

Лістинг програми на мові програмування Python

from pandas import \*

import numpy as np

from sklearn.linear\_model import LinearRegression, LogisticRegression

from sklearn.svm import SVR

from sklearn.discriminant\_analysis import QuadraticDiscriminantAnalysis as QDA

count\_logistic1 = 0

count\_logistic2 = 0

count\_QDA1 = 0

count\_QDA2 = 0

count\_svm1 = 0

count\_svm2 = 0

set\_option('display.max\_columns', 50)

set\_option('display.width', 500)

data\_tune = read\_csv('Spam\_tune.csv') #Набор тестовых данных

data\_test = read\_csv('Spam\_test.csv') # Набор контрольных данных

x = data\_tune.iloc[:,:-1] # Набор признаков

y = data\_tune.iloc[:,-1] # Метки классов

x\_train = x

x\_test = data\_test.iloc[:,:-1]

y\_train = y

y\_test = data\_test.iloc[:,-1]

d = DataFrame(index=y\_test.index) # Набор данных для проверки модели

d["Actual"] = y\_test # Реальные значения тегов

model1 = LogisticRegression()

model1.fit(x\_train, y\_train)

d['Predict\_LR'] = model1.predict(x\_test) # Тест

d["Correct\_LR"] = (d['Predict\_LR']\*d["Actual"]) # Считаем процент совпадений

hit\_rate1 = np.mean(d["Correct\_LR"])

print("Процент верных предсказаний логистической регрессии: %.1f%%" % (hit\_rate1\*100))

d["Error\_LR"] = (d["Actual"]-d['Predict\_LR'])

for i in d["Error\_LR"]:

if i == -1:

count\_logistic2 += 1

elif i == 1:

count\_logistic1 += 1

print("Ошибки 1 рода - %.1f%%" % (count\_logistic1/len(d["Error\_LR"])\*100))

print("Ошибки 2 рода - %.1f%%" % (count\_logistic2/len(d["Error\_LR"])\*100))

print()

# ----------------------- LinearRegression---------------------------

model2 = LinearRegression()

model2.fit(x\_train, y\_train)

d['Predict\_ LinearRegression'] = model2.predict(x\_test) # Тест

d["Correct\_ LinearRegression"] = (d['Predict\_LDA']\*d["Actual"]) # Считаем процент совпадений

hit\_rate2 = np.mean(d["Correct\_ LinearRegression"])

print("Процент верных предсказаний для LinearRegression: %.1f%%" % (hit\_rate2\*100))

d["Error\_ LinearRegression"] = ((d["Actual"])-d['Predict\_ LinearRegression'])

for i in d["Error\_ LinearRegression"]:

if i == -1:

count\_svm2 += 1

elif i == 1:

count\_svm1 += 1

print("Ошибки 1 рода - %.1f%%" % (count\_svm1/len(d["Error\_ LinearRegression"])\*100))

print("Ошибки 2 рода - %.1f%%" % (count\_svm2/len(d["Error\_ LinearRegression"])\*100))

print(d)

model3 = SVM()

model3.fit(x\_train, y\_train)

count\_svm1 = 0

count\_svm2 = 0

d['Predict\_SVM'] = model3.predict(x\_test) # Тест

d["Correct\_SVM"] = (d['Predict\_SVM']\*d["Actual"]) # Считаем процент совпадений

hit\_rate3 = np.mean(d["Correct\_SVM"])

print("Процент верных предсказаний для SVM: %.1f%%" % (hit\_rate3\*100))

d["Error\_SVM"] = (d["Actual"]-d['Predict\_SVM'])

for i in d["Error\_SVM"]:

if i == -1:

count\_svm2 += 1

elif i == 1:

count\_svm1 += 1

print("Ошибки 1 рода - %.1f%%" % (count\_svm1/len(d["Error\_SVM"])\*100))

print("Ошибки 2 рода - %.1f%%" % (count\_svm2/len(d["Error\_SVM"])\*100))