Федеральное государственное автономное  
образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет  
**информационных технологий, механики и оптики**

Факультет программной инженерии и компьютерной техники

Кафедра информатики и прикладной математики

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

по дисциплине «Машинное обучение»

**Выполнил:**

студент группы P4117

Данилов М.В.

**Проверил:**

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

Санкт-Петербург

2017 г.

#### **1. Постановка задачи**

Получить практические навыки работы с методом дискриминантного анализа и визуализацией данных на практических примерах с использованием языка программирования python.

#### **2. Исходные данные**

Датасет: <http://archive.ics.uci.edu/ml/datasets/banknote+authentication>

Предметная область: Компьютер

Задача: Определить является ли банкнота поддельной

Количество записей: 1372

Количество атрибутов: 5

Данные были получены из изображений, поддельный и подлинных купюр. Для оцифровки использовалась промышленная камера, предназначенная для проверки печати.

Атрибуты:

1. Дисперсия изображения после Вейвлет-преобразования (число с плавающей запятой)
2. Асимметрия изображения после Вейвлет-преобразования (число с плавающей запятой)
3. Эксцесс изображения после Вейвлет-преобразования (число с плавающей запятой)
4. Энтропия изображения (число с плавающей запятой)
5. Подлинность банкноты (0 или 1)

#### **3. Ход работы**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from matplotlib import pyplot as plt

from scipy.stats import pearsonr

from sklearn import preprocessing

from mpl\_toolkits.mplot3d import Axes3D

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA

from sklearn.discriminant\_analysis import QuadraticDiscriminantAnalysis as QDA

from sklearn import metrics

def load\_data(filename):

return pd.read\_csv(filename, header=None).values

def split\_data():

dataset = load\_data('data\_banknote\_authentication.txt')

money\_attr = dataset[:,-1]

money\_attr = money\_attr.astype(np.float64)

money\_class = dataset[:,:-1]

money\_class = money\_class.astype(np.float64, copy=False)

return money\_class, money\_attr

def data\_description(money\_attr, money\_class):

columns = ['VarianceOWTI', 'SkewnessOWTI', 'CurtosisOWTI', 'EntropyOWTI', 'Class']

data = pd.DataFrame(load\_data('data\_banknote\_authentication.txt'), columns=columns)

print(data.head())

print('Number of records:', money\_class.shape[0])

print('Number of signs:', money\_attr.shape[1])

print('\nThe shares of each of the classes')

print('Class 0 (Fake): {:.2%}'.format(list(money\_class).count(0) / money\_class.shape[0]))

print('Class 1 (Real): {:.2%}'.format(list(money\_class).count(1) / money\_class.shape[0]))

def data\_2D\_visualization(money\_attr, money\_class):

plt.figure(figsize=(6, 5))

for label, marker, color in zip(

range(0, 2), ('x', 'o'), ('black', 'blue')):

# Вычисление коэффициента корреляции Пирсона

R = pearsonr(money\_attr[:, 0][money\_class == label], money\_attr[:, 1][money\_class == label])

plt.scatter(x=money\_attr[:, 0][money\_class == label],

y=money\_attr[:, 1][money\_class == label],

marker=marker,

color=color,

alpha=0.7,

label='class {:}, R={:.2f}'.format(label, R[0])

)

plt.title('Banknote authentication Data Set')

plt.xlabel('VarianceOWTI')

plt.ylabel('SkewnessOWTI')

plt.legend(loc='upper right')

plt.show()

def train\_test\_visualization(data\_train, data\_test, class\_train, class\_test):

std\_scale = preprocessing.StandardScaler().fit(data\_train)

data\_train = std\_scale.transform(data\_train)

data\_test = std\_scale.transform(data\_test)

f, ax = plt.subplots(1, 2, sharex=True, sharey=True, figsize=(10, 5))

for a, x\_dat, y\_lab in zip(ax, (data\_train, data\_test), (class\_train, class\_test)):

for label, marker, color in zip(

range(0, 2), ('x', 'o'), ('black', 'blue')):

a.scatter(x=x\_dat[:, 0][y\_lab == label],

y=x\_dat[:, 1][y\_lab == label],

marker=marker,

color=color,

alpha=0.7,

label='class {}'.format(label)

)

a.legend(loc='upper right')

ax[0].set\_title('VarianceOWTI')

ax[1].set\_title('SkewnessOWTI')

f.text(0.5, 0.04, 'VarianceOWTI (standardized)', ha='center', va='center')

f.text(0.08, 0.5, 'SkewnessOWTI (standardized)', ha='center', va='center', rotation='vertical')

# plt.show()

def train\_data(money\_attr, money\_class):

data\_train, data\_test, class\_train, class\_test = train\_test\_split(money\_attr, money\_class, test\_size=0.30,

random\_state=123)

print('\nThe shares of each of the classes')

print('\nTraining Dataset:')

print('Class 0 (Fake): {:.2%}'.format(list(class\_train).count(0) / class\_train.shape[0]))

print('Class 1 (Real): {:.2%}'.format(list(class\_train).count(1) / class\_train.shape[0]))

print('\nTest Dataset:')

print('Class 0 (Fake): {:.2%}'.format(list(class\_test).count(0) / class\_test.shape[0]))

print('Class 1 (Real): {:.2%}'.format(list(class\_test).count(1) / class\_test.shape[0]))

train\_test\_visualization(data\_train, data\_test, class\_train, class\_test)

return data\_train, data\_test, class\_train, class\_test

def linear\_discriminant\_analysis(data\_train, class\_train):

sklearn\_lda = LDA()

sklearn\_transf = sklearn\_lda.fit(data\_train, class\_train).transform(data\_train)

plt.figure(figsize=(8, 8))

for label, marker, color in zip(

range(0, 2), ('x', 'o'), ('red', 'green')):

plt.scatter(x=sklearn\_transf[class\_train == label],

y=sklearn\_transf[class\_train == label],

marker=marker,

color=color,

alpha=0.7,

label='class {}'.format(label))

plt.xlabel('vector 1')

plt.ylabel('vector 2')

plt.legend()

# Визуализация разбиения классов после линейного преобразования LDA

plt.title('Most significant singular vectors after linear transformation via LDA')

plt.show()

def train\_linear\_discriminant\_analysis(data\_train, data\_test, class\_train, class\_test):

lda\_clf = LDA()

lda\_clf.fit(data\_train, class\_train)

# LDA(n\_components=None, priors=None, shrinkage=None, solver='svd', store\_covariance=False, tol=0.0001)

pred\_train = lda\_clf.predict(data\_train)

print('LDA')

print('Точность классификации на тестовом наборе данных')

print('{:.2%}'.format(metrics.accuracy\_score(class\_train, pred\_train)))

pred\_test = lda\_clf.predict(data\_test)

print('Точность классификации на обучающем наборе данных')

print('{:.2%}'.format(metrics.accuracy\_score(class\_test, pred\_test)))

def train\_quadratic\_discriminant\_analysis(data\_train, data\_test, class\_train, class\_test):

qda\_clf = QDA()

qda\_clf.fit(data\_train, class\_train)

pred\_train = qda\_clf.predict(data\_train)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

print('QDA')

print('Точность классификации на тестовом наборе данных')

print('{:.2%}'.format(metrics.accuracy\_score(class\_train, pred\_train)))

pred\_test = qda\_clf.predict(data\_test)

print('Точность классификации на обучающем наборе данных')

print('{:.2%}'.format(metrics.accuracy\_score(class\_test, pred\_test)))

def main():

money\_attr, money\_class = split\_data()

data\_description(money\_attr, money\_class)

data\_2D\_visualization(money\_attr, money\_class)

data\_train, data\_test, class\_train, class\_test = train\_data(money\_attr, money\_class)

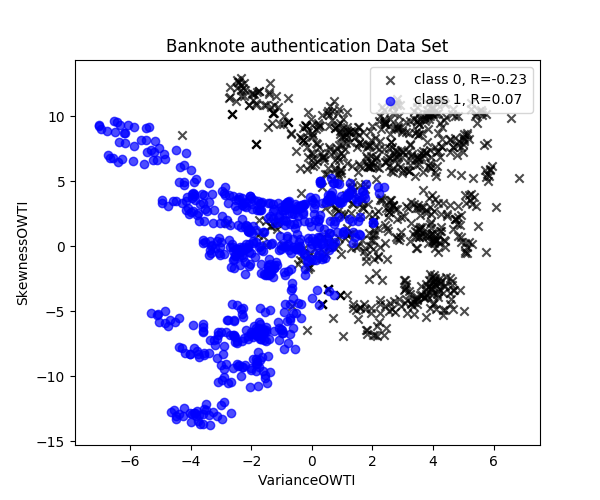
linear\_discriminant\_analysis(data\_train, class\_train)

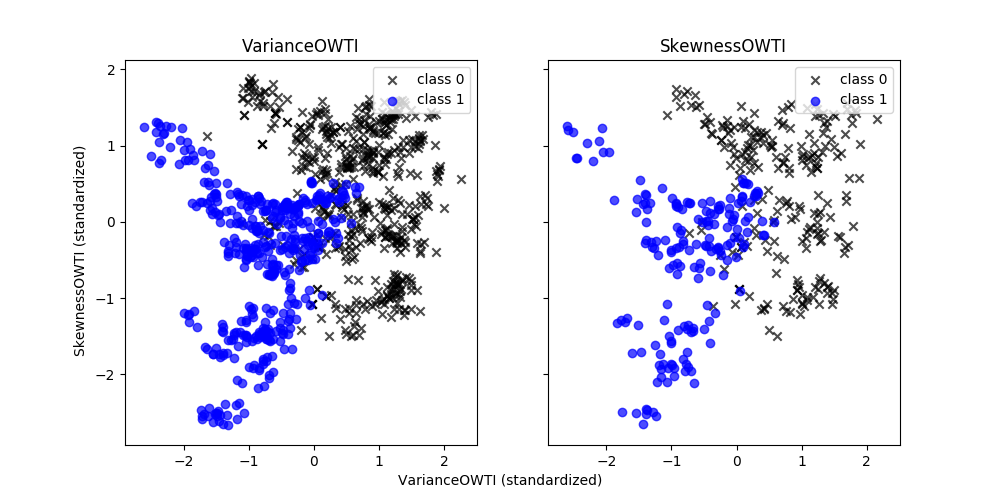
train\_linear\_discriminant\_analysis(data\_train, data\_test, class\_train, class\_test)

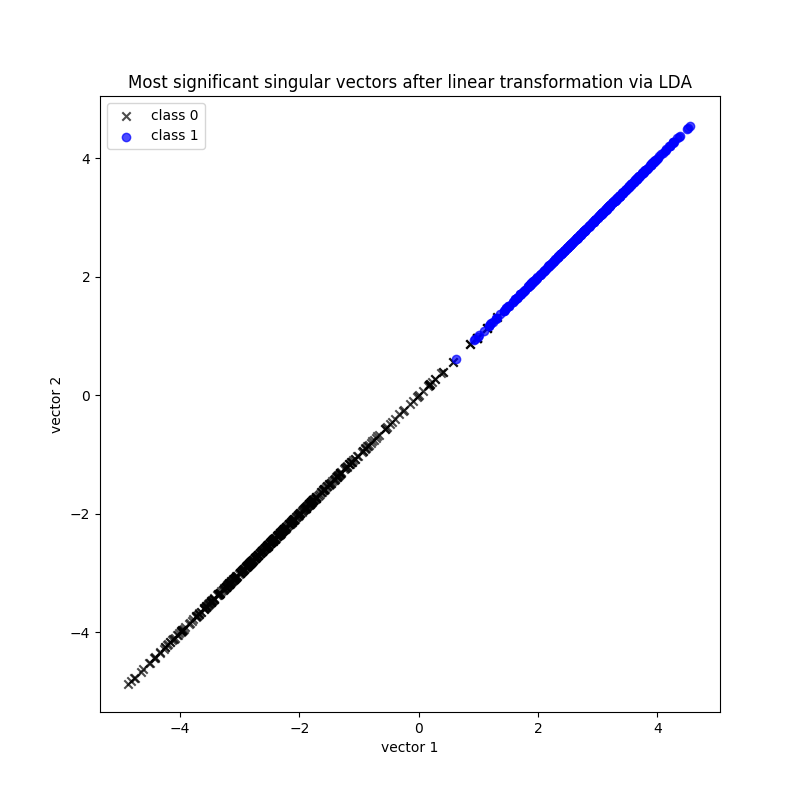
train\_quadratic\_discriminant\_analysis(data\_train, data\_test, class\_train, class\_test)

main()

Результаты:







The shares of each of the classes

Class 0 (Fake): 55.54%

Class 1 (Real): 44.46%

LDA

Точность классификации на тестовом наборе данных

97.92%

Точность классификации на обучающем наборе данных

97.09%

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

QDA

Точность классификации на тестовом наборе данных

98.44%

Точность классификации на обучающем наборе данных

98.06%

**Вывод:**

В ходе лабораторной работы были получены навыки работы методом дискриминантного анализа и визуализацией данных на практических примерах с использованием языка программирования python.

Была выполнена визуализация двух признаков (Дисперсия изображения после Вейвлет-преобразования и асимметрия изображения после Вейвлет-преобразования). Выполнено разбиение класса набора данных с помощью LDA.

В результате классификации методы LDA и QDA показали схожие результаты с разницей ̴ 1%