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 №1**

по дисциплине «Машинное обучение»

**Выполнил:**

студент группы P4117

Данилов М.В.

**Проверил:**

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

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

Санкт-Петербург

2017 г.

#### **1. Постановка задачи**

1. На языке Python программно реализовать два метрических алгоритма классификации: Naive Bayes и K Nearest Neighbours
2. Сравнить работу реализованных алгоритмов с библиотечными из scikit-learn
3. Для тренировки, теста и валидации использовать один из предложенных датасетов (либо найти самостоятельно и внести в таблицу)
4. Сформировать краткий отчет (постановка задачи, реализация, эксперимент с данными, полученные характеристики, вывод

#### **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**

1. 1. Реализация алгоритма Naive Bayes.

import math

import numpy as np

import pandas as pd

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

from sklearn.naive\_bayes import GaussianNB

def load\_data(filename):

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

def split\_dataset(test\_size):

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

money\_attr = dataset[:, 0:-1]

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

money\_class = dataset[:, -1]

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

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

random\_state=55)

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

def separate\_by\_class(data\_train, class\_train):

classes\_dict = {}

for i in range(len(data\_train)):

classes\_dict.setdefault(class\_train[i], []).append(data\_train[i])

return classes\_dict

def mean(numbers):

return sum(numbers) / float(len(numbers))

def stand\_dev(numbers):

var = sum([pow(x - mean(numbers), 2) for x in numbers]) / float(len(numbers) - 1)

return math.sqrt(var)

def summarize(data\_train):

summaries = [(mean(att\_numbers), stand\_dev(att\_numbers)) for att\_numbers in zip(\*data\_train)]

return summaries

def summarize\_by\_class(data\_train, class\_train):

classes\_dict = separate\_by\_class(data\_train, class\_train)

summaries = {}

for class\_name, instances in classes\_dict.items():

summaries[class\_name] = summarize(instances)

return summaries

def calc\_probability(x, mean, stdev):

if stdev == 0:

stdev += 0.000001

exponent = math.exp(-(math.pow(x - mean, 2) / (2 \* math.pow(stdev, 2))))

return (1 / (math.sqrt(2 \* math.pi) \* stdev)) \* exponent

def calc\_class\_probabilities(summaries, instance\_attr):

probabilities = {}

for class\_name, class\_summaries in summaries.items():

probabilities[class\_name] = 1.0

for i in range(len(class\_summaries)):

mean, stdev = class\_summaries[i]

x = float(instance\_attr[i])

probabilities[class\_name] \*= calc\_probability(x, mean, stdev)

return probabilities

def predict\_one(summaries, instance\_attr):

probabilities = calc\_class\_probabilities(summaries, instance\_attr)

best\_class, max\_prob = None, -1

for class\_name, probability in probabilities.items():

if best\_class is None or probability > max\_prob:

max\_prob = probability

best\_class = class\_name

return best\_class

def predict(summaries, data\_test):

predictions = []

for i in range(len(data\_test)):

result = predict\_one(summaries, data\_test[i])

predictions.append(result)

return predictions

def calc\_accuracy(summaries, data\_test, class\_test):

correct\_answ = 0

predictions = predict(summaries, data\_test)

for i in range(len(data\_test)):

if class\_test[i] == predictions[i]:

correct\_answ += 1

return correct\_answ / float(len(data\_test))

def main():

data\_train, class\_train, data\_test, class\_test = split\_dataset(0.3)

summaries = summarize\_by\_class(data\_train, class\_train)

accuracy = calc\_accuracy(summaries, data\_test, class\_test)

print('myNBClass ', 'Accuracy: ', accuracy)

clf = GaussianNB()

clf.fit(data\_train, class\_train)

print('sklNBClass ', 'Accuracy: ', clf.score(data\_test, class\_test))

with open('result.NaiveBayes.txt', 'w') as ouf:

ouf.writelines('myNBClass ' + 'Accuracy: ' + str(calc\_accuracy(summaries, data\_test, class\_test)) + '\n')

ouf.writelines('sklNBClass ' + 'Accuracy: '+ str(clf.score(data\_test, class\_test)))

main()

Результаты:

myNBClass Accuracy: 0.8616504854368932

sklNBClass Accuracy: 0.849514563107

1. Реализация алгоритма K Nearest Neighbours

from \_\_future\_\_ import division

import pandas as pd

import numpy as np

import operator

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

from math import sqrt

from collections import Counter

from sklearn.neighbors import KNeighborsClassifier

def load\_data():

dataset = pd.read\_csv('data\_banknote\_authentication.txt', header=None).values

money\_attr = dataset[:, 0:-1]

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

money\_class = dataset[:, -1]

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

return train\_test\_split(money\_attr, money\_class, test\_size=0.35)

def euclidean\_distance(instance1, instance2):

squares = [(i - j) \*\* 2 for i, j in zip(instance1, instance2)]

return sqrt(sum(squares))

def get\_neighbours(instance, data\_train, class\_train, k):

distances = []

for i in data\_train:

distances.append(euclidean\_distance(instance, i))

distances = tuple(zip(distances, class\_train))

return sorted(distances, key=operator.itemgetter(0))[:k]

def get\_response(neigbours):

return Counter(neigbours).most\_common()[0][0][1]

def get\_predictions(data\_train, class\_train, data\_test, k):

predictions = []

for i in data\_test:

neigbours = get\_neighbours(i, data\_train, class\_train, k)

response = get\_response(neigbours)

predictions.append(response)

return predictions

def get\_accuracy(data\_train, class\_train, data\_test, class\_test, k):

predictions = get\_predictions(data\_train, class\_train, data\_test, k)

mean = [i == j for i, j in zip(class\_test, predictions)]

return sum(mean) / len(mean)

def main():

data\_train, data\_test, class\_train, class\_test = load\_data()

print('myKNClass', 'Accuracy: ', get\_accuracy(data\_train, class\_train, data\_test, class\_test, 15))

clf = KNeighborsClassifier(n\_neighbors=15)

clf.fit(data\_train, class\_train)

print('sklKNClass', 'Accuracy: ', clf.score(data\_test, class\_test))

with open('result.NearestNeighbours.txt', 'w') as ouf:

ouf.writelines('myKNClass ' + 'Accuracy: ' + str(get\_accuracy(data\_train, class\_train, data\_test, class\_test, 15)) + '\n')

ouf.writelines('sklKNClass ' + 'Accuracy: ' + str(clf.score(data\_test, class\_test)))

main()

Результаты:

myKNClass Accuracy: 0.991683991684

sklKNClass Accuracy: 0.997920997921

**Вывод:**

В ходе лабораторной работы были получены навыки работы с метрическими методами машинного обучения на практических примерах с использованием языка программирования python и библиотеки sklearn.

Были имплементированы собственные реализации алгоритмов. При сравнении реализованных алгоритмов с библиотечными выяснилось, что на одной и той же выборке они показывают схожие результаты.