Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Высшая школа программной инженерии

Лабораторная работа №2

По дисциплине «Машинное обучение»

Выполнил студент гр 33534/5  Донцов А. Д.

Руководитель И. А. Селин

Санкт-Петербург  
2019 г.

# Постановка задачи

1. Исследуйте, как объем обучающей выборки и количество тестовых данных, влияет на

точность классификации или на вероятность ошибочной классификации в датасетах про

крестики-нолики и о спаме e-mail сообщений.

2. Постройте классификатор для обучающего множества Glass (glass.csv), данные которого

характеризуются 10-ю признаками:

1. Id number: 1 to 214; 2. RI: показатель преломления; 3. Na: сода (процент содержания в

соответствующем оксиде); 4. Mg; 5. Al; 6. Si; 7. K; 8. Ca; 9. Ba; 10. Fe.

Классы характеризуют тип стекла:

(1) окна зданий, плавильная обработка

(2) окна зданий, не плавильная обработка

(3) автомобильные окна, плавильная обработка

(4) автомобильные окна, не плавильная обработка (нет в базе)

(5) контейнеры

(6) посуда

(7) фары

Посмотрите заголовки признаков и классов. Перед построением классификатора необходимо

также удалить первый признак Id number, который не несет никакой информационной нагрузки.

Постройте графики зависимости ошибки классификации от значения n\_neighbors.

Определите подходящие метрики из класса DistanceMetric и исследуйте, как тип метрики

расстояния влияет на точность классификации.

Определите, к какому типу стекла относится экземпляр с характеристиками

RI =1.516 Na =11.7 Mg =1.01 Al =1.19 Si =72.59 K=0.43 Ca =11.44 Ba =0.02 Fe =0.1

Определите, какой из признаков оказывает наименьшее влияние на определение класса путем

последовательного исключения каждого признака.

3. Для построения классификатора используйте заранее сгенерированные обучающие и тестовые

выборки, хранящиеся в файлах svmdata4.txt, svmdata4test.txt. Найдите оптимальное значение

n\_neighbors, обеспечивающее наименьшую ошибку классификации. Посмотрите, как выглядят

данные на графике.

# Ход работы

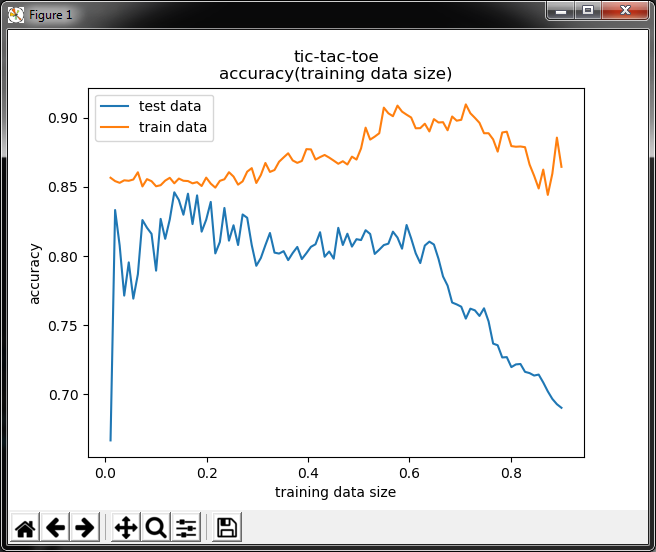
1. Для данных tic-tac-toe и spam были построены графики зависимости точности от размеров выборки, из которых видно, что при слишком большом или слишком малом размере обучающей выборки падает точность предсказания.
2. Для тестирования были взяты значения n\_ neighbors в диапазоне от 2 до 29, были рассмотрены метрики Euclidean, Manhattan, Chebyshev, Minkowski. Были построены графики зависимости точности от значения n\_neighbors  
   Наивысшая точность:
   1. euclidean, n\_neighbors: 2 - Score: 0.6401869158878505
   2. manhattan, n\_neighbors: 2 - Score: 0.6401869158878505
   3. chebyshev, n\_neighbors: 2 - Score: 0.6401869158878505
   4. minkowski, n\_neighbors: 2 - Score: 0.6401869158878505

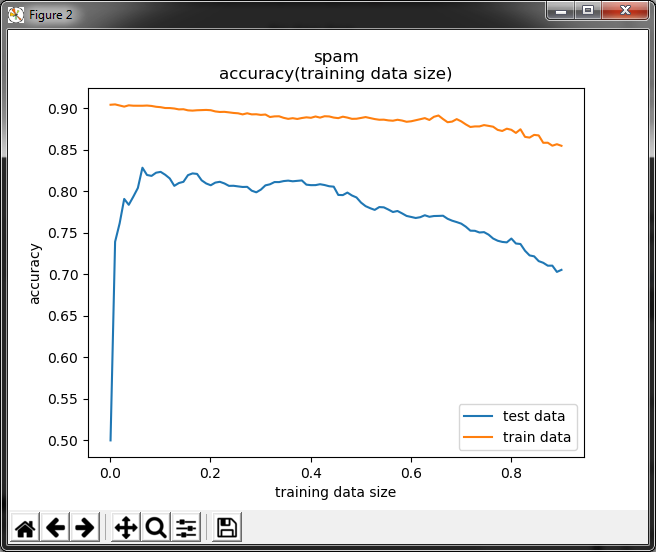
Для заданного значения RI =1.516 Na =11.7 Mg =1.01 Al =1.19 Si =72.59 K=0.43 Ca =11.44 Ba =0.02 Fe =0.1

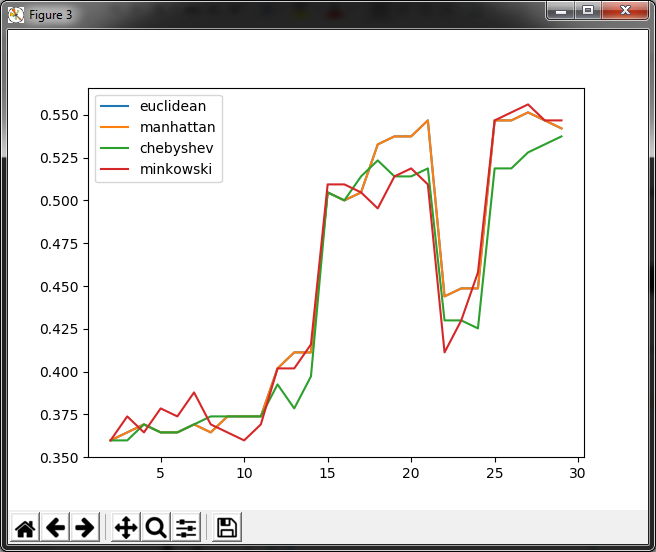
Был определен тип: 2 – окна зданий, не плавильная обработка.

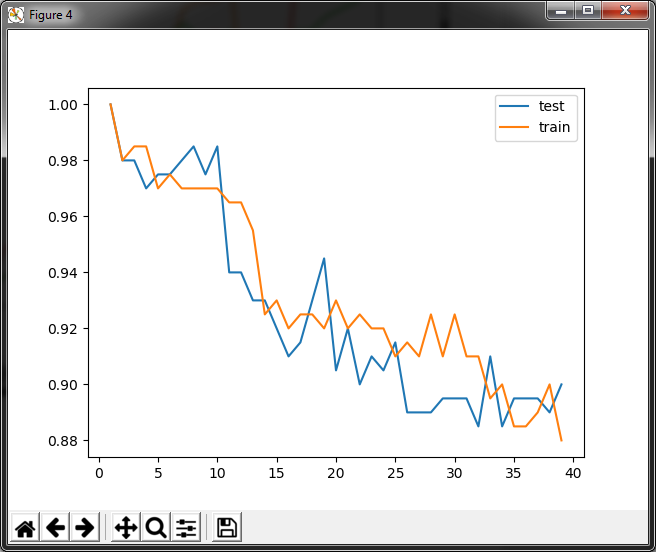
1. Были построены графики зависимости точности предсказания от значения n\_ neighbors для данных svmdata4.txt, svmdata4test.txt.

# Результаты работы









# Вывод

Было установлено, что для данных tic-tac-toe и spam точность предсказания падает с увеличением величины обучающей выборки.

Для набора данных glass было установлено, что наивысшая точность предсказания достигается при значении n\_neighbors = 2, изменение точности предсказания в зависимости от метрики и изменения параметра показана на графике. Также было установлено, что на данном наборе данных метрики Euclidean и manhattan имеют одинаковую точность.

При последовательном удалении параметров (на 1 элементе) получились следующие предсказания:

* Тип 2
* Тип 2
* Тип 6
* Тип 2
* Тип 1
* Тип 2
* Тип 1
* Тип 2
* Тип 2

С учетом полученного на тестовом варианте результата можно сделать вывод, что наиболее ощутимо влияют на результат параметры Mg Si Ca

# Текст программы

from matplotlib import pyplot as plt  
from sklearn.neighbors import KNeighborsClassifier as nbh  
from sklearn.neighbors import DistanceMetric  
from sklearn.model\_selection import train\_test\_split  
from sklearn import metrics  
import pandas as pd  
from sklearn import preprocessing  
import numpy as np  
  
  
def make\_plot(ratios, accuracies, title):  
 plt.figure()  
 plt.plot(ratios, [acc[0] for acc in accuracies], label='test data')  
 plt.plot(ratios, [acc[1] for acc in accuracies], label='train data')  
 plt.xlabel('training data size')  
 plt.ylabel('accuracy')  
 plt.title(f'{title}\naccuracy(training data size)')  
 plt.legend()  
 plt.savefig(f'{title}.png')  
  
  
def accuracy(feat, train, tr\_size):  
 tr\_size = 1-tr\_size  
 x\_test, x\_targ, y\_test, y\_targ = \  
 train\_test\_split(feat,train, test\_size=tr\_size, random\_state=1)  
 neigh = nbh(n\_neighbors=3, n\_jobs=-1)  
 neigh.fit(x\_targ, y\_targ)  
 neigh.predict\_proba(x\_targ)  
 return (metrics.accuracy\_score(y\_test, neigh.predict(x\_test)),  
 metrics.accuracy\_score(y\_targ, neigh.predict(x\_targ)))  
  
  
def tic\_tac\_toe():  
 features, targets = [], []  
 with open("Tic\_tac\_toe.txt") as inp:  
 for line in inp:  
 features.append(line.split(',')[0:9])  
 targets.append(line.split(',')[9].strip())  
 le = preprocessing.LabelEncoder()  
 features\_encoded = [le.fit\_transform(sample) for sample in features]  
 targets\_encoded = le.fit\_transform(targets)  
 ratios = np.linspace(0.01, 0.9, 100)  
 accuracies = [accuracy(features\_encoded, targets\_encoded, ratio) for ratio in ratios]  
 make\_plot(ratios, accuracies, 'tic-tac-toe')  
  
  
def spam():  
 df = pd.read\_csv('spam.csv', sep=',')  
 features = df.iloc[:, 1:58].values  
 targets = df['type'].values  
 targets\_encoded = preprocessing.LabelEncoder().fit\_transform(targets)  
 ratios = np.linspace(0.001, 0.9, 100)  
 accuracies = [accuracy(features, targets\_encoded, ratio) for ratio in ratios]  
 make\_plot(ratios, accuracies, 'spam')  
  
  
def glass():  
 df = pd.read\_csv('glass.csv', sep=',')  
 df = df.drop('Id', 1)  
 features = df.drop('Type', 1).values  
 targets = df['Type'].values  
 le = preprocessing.LabelEncoder()  
 features\_encoded = [le.fit\_transform(sample) for sample in features]  
 targets\_encoded = le.fit\_transform(targets)  
 nbhrs = []  
 scores = []  
 total\_nbh = []  
 total\_scores = []  
 neigh = nbh()  
 for i in ('euclidean', 'manhattan', 'chebyshev', 'minkowski'):  
 for j in range(2, 30):  
 neigh = nbh(n\_neighbors=j, metric=i, n\_jobs=-1)  
 classifier = neigh.fit(features\_encoded, targets\_encoded)  
 print('Metric: {0}, n\_neighbors: {1} - Score: {2}'.format(i, j, classifier.score(features\_encoded, targets\_encoded)))  
 nbhrs.append(j)  
 scores.append(1-classifier.score(features\_encoded, targets\_encoded))  
 total\_nbh.append(nbhrs)  
 total\_scores.append(scores)  
 nbhrs = []  
 scores = []  
 plt.figure()  
 lbl = ['euclidean', 'manhattan', 'chebyshev', 'minkowski']  
 for i, l in enumerate(lbl):  
 plt.plot(total\_nbh[i-1], total\_scores[i-1], label=l)  
 plt.legend()  
 tst = [[1.516, 11.7, 1.01, 1.19, 72.59, 0.43, 11.44, 0.02, 0.1]]  
 pred = neigh.predict(tst)  
 print('\*\*\*\*\*\*\*\*')  
 type(features\_encoded)  
 print('Predicted type: {0}'.format(pred + 1))  
 print('TEST WITH DELETED ELEMENTS')  
  
 for i in features\_encoded:  
 for j in range(len(i)):  
 print('0 is element{0}'.format(j))  
 temp = i.copy()  
 temp[j] = 0  
 pred = neigh.predict([temp])  
 print('Predicted type: {0}'.format(pred+1))  
 print(temp)  
 print('next element')  
  
  
def svm():  
 df = pd.read\_csv('svmdata4.txt', sep='\t')  
 train\_data = df.drop('Colors', 1).values  
 train\_res = df['Colors'].values  
 df = pd.read\_csv('svmdata4test.txt', sep='\t')  
 user\_data = df.drop('Colors', 1).values  
 user\_res = df['Colors'].values  
 le = preprocessing.LabelEncoder()  
 train\_res\_encoded = le.fit\_transform(train\_res)  
 user\_res\_encoded = le.fit\_transform(user\_res)  
 total\_scores\_train = []  
 total\_scores\_test = []  
 total\_nbh = []  
 for i in range(1, 40):  
 nbhr = nbh(n\_neighbors=i, metric='euclidean', n\_jobs=1)  
 classifier = nbhr.fit(train\_data, train\_res\_encoded)  
 total\_scores\_train.append(classifier.score(train\_data, train\_res\_encoded))  
 classifier = nbhr.fit(user\_data, user\_res\_encoded)  
 total\_scores\_test.append(classifier.score(user\_data, user\_res\_encoded))  
 total\_nbh.append(i)  
 plt.figure()  
 plt\_lst = [total\_scores\_test, total\_scores\_train]  
 for i, lbl in enumerate(['test', 'train']):  
 plt.plot(total\_nbh, plt\_lst[i], label=lbl)  
 plt.legend()  
  
  
#tic\_tac\_toe()  
print('tic\_tac\_toe')  
#spam()  
print('spam')  
glass()  
print('glass')  
#svm()  
print('svm')  
plt.show()