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

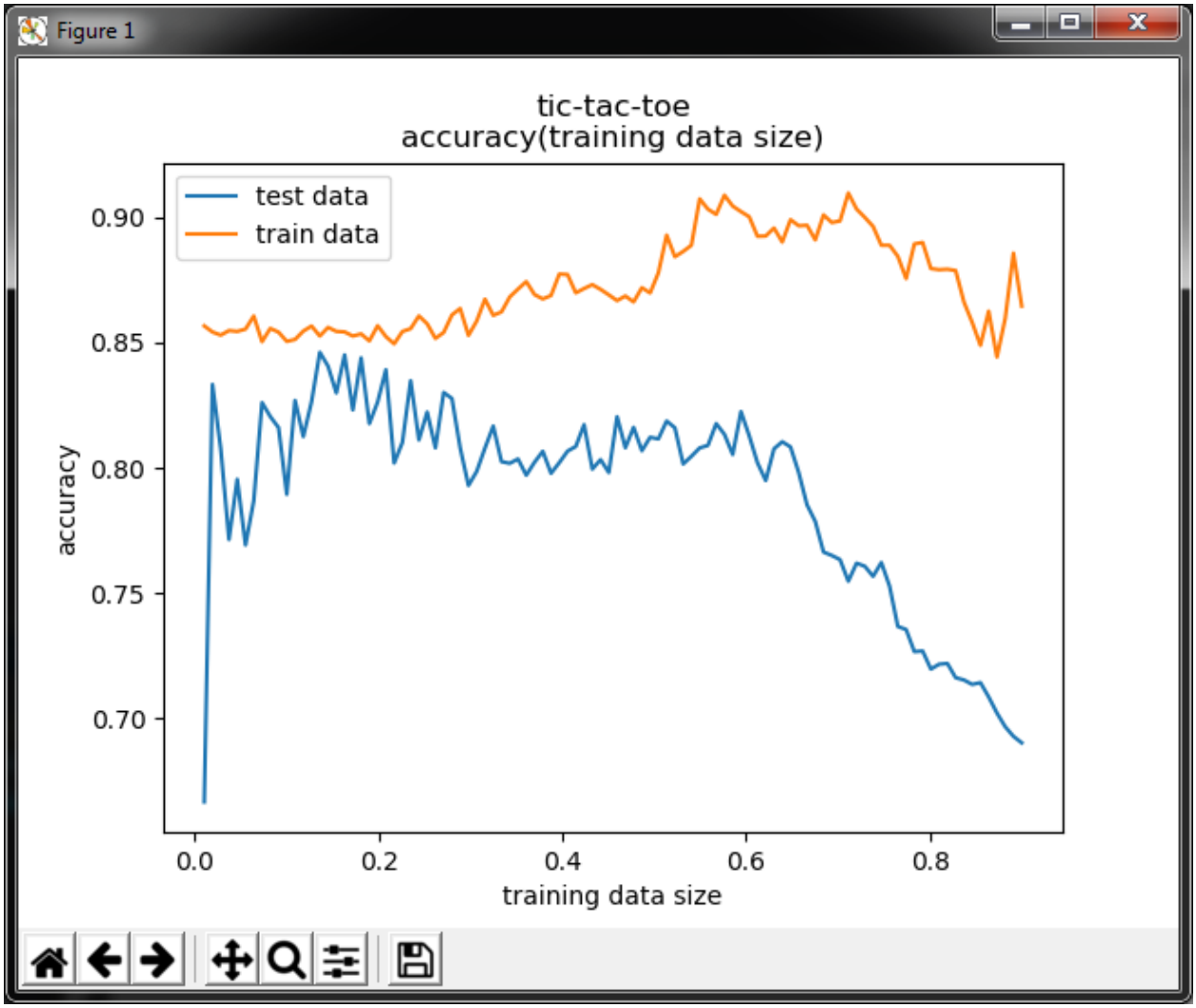
1. Для данных tic-tac-toe и spam были построены графики зависимости точности от размеров выборки, из которых видно, что при слишком большом или слишком малом размере обучающей выборки падает точность предсказания.
2. Для тестирования были взяты значения n_ neighbors в диапазоне от 2 до 29, были рассмотрены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

3. Были построены графики зависимости точности предсказания от значения n_ neighbors для данных svmdata4.txt, svmdata4test.txt.

Результаты работы



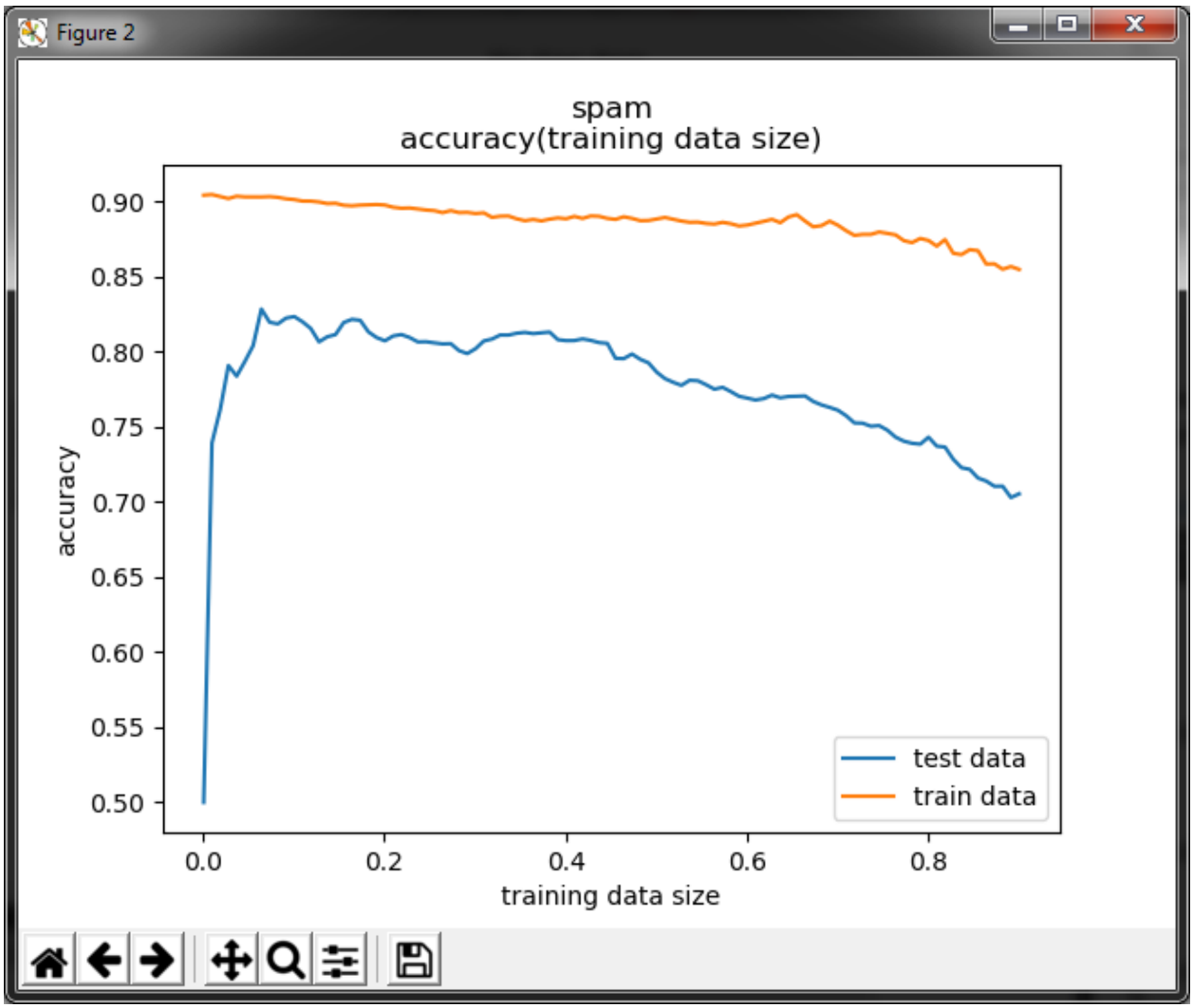


Figure 2
spam
accuracy(training data size)
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
accuracy
0.0
0.2
0.4
0.6
0.8
training data size
test data
train data

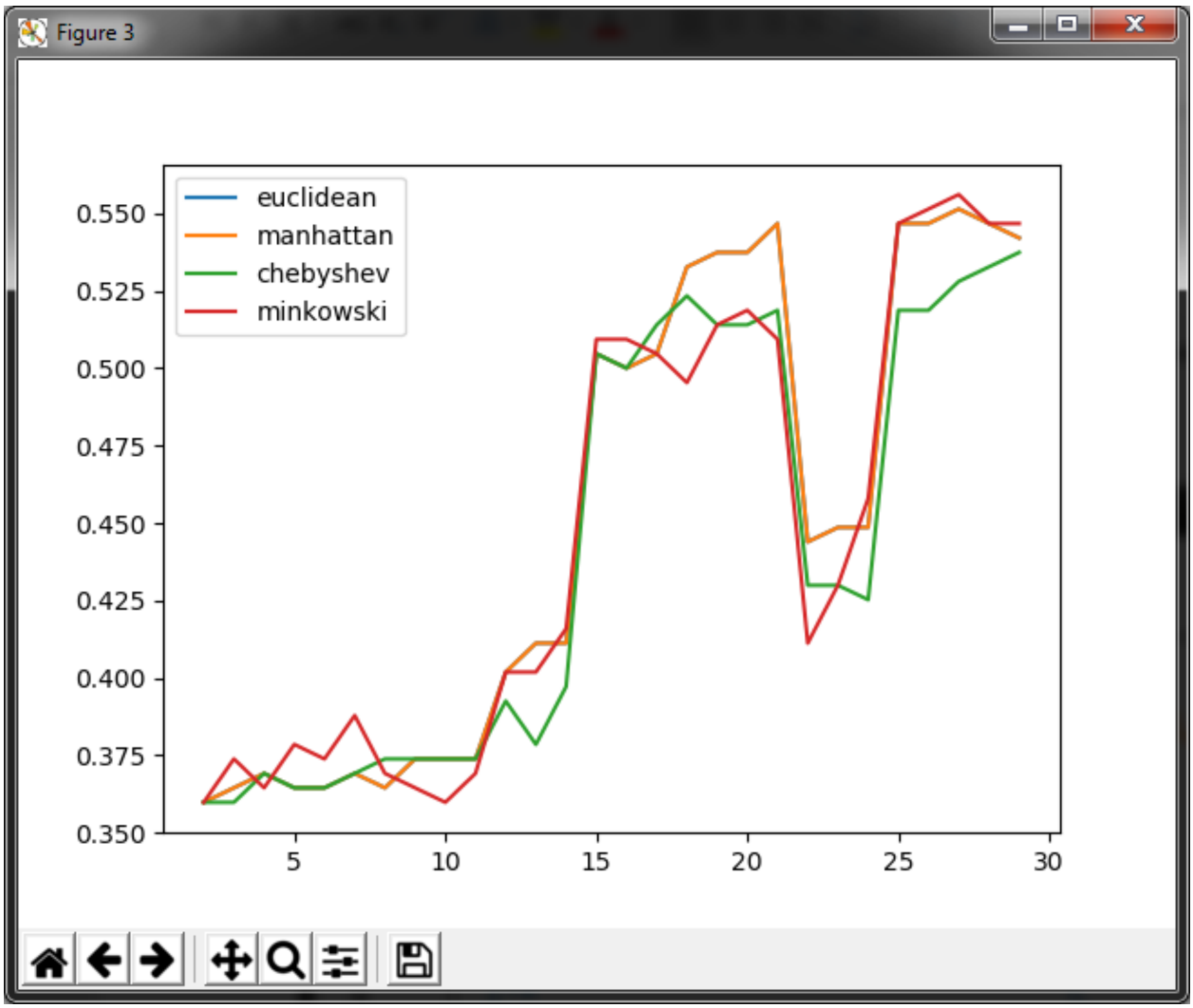
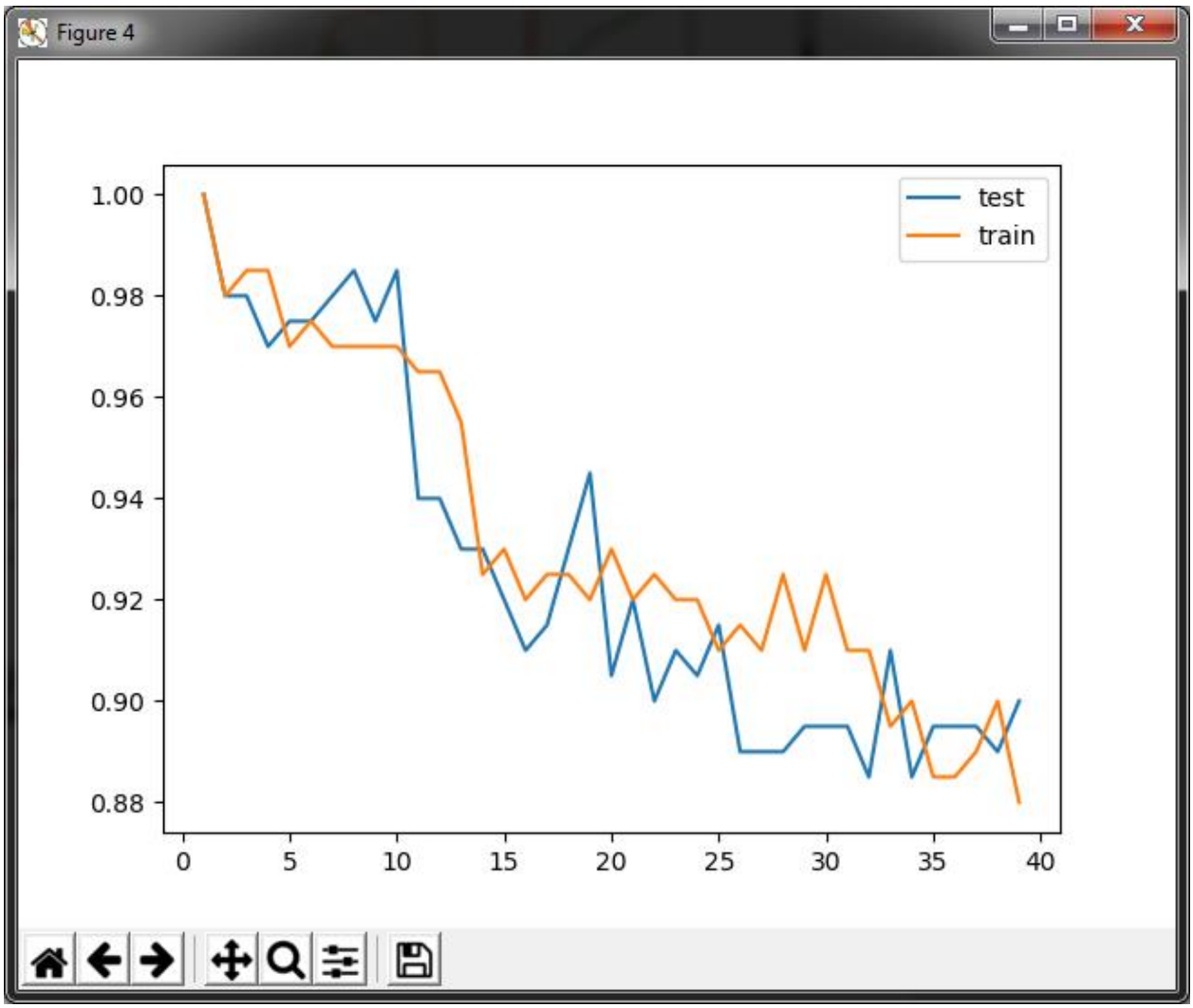


Figure 3
euclidean
manhattan
chebyshev
minkowski
0.550
0.525
0.500
0.475
0.450
0.425
0.400
0.375
0.350
5
10
15
20
25
30



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 получились следующие предсказания:

- Тип 2
- Тип 2
- Тип 2
- Тип 2
- Тип 6
- Тип 6
- Тип 6
- Тип 5
- Тип 5

Из чего можно сделать вывод, что параметры RI, Na, Mg, Al в меньшей степени влияют на результат предсказания.

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('********')
    print('Predicted type: {0}'.format(pred + 1))
    print('GLASSSSSS')
    for i in range(0, len(tst[0])):
        temp = tst[0]
        temp[i] = 0
        pred = neigh.predict([temp])
        print('********')
        print('Predicted type: {0}'.format(pred+1))


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()
```