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

Лабораторная работа №1

по дисциплине «Машинное обучение»

Выплолнил студент
гр. 33534/5  Стойкоски Н.С.

Руководитель И.А. Селин

Санкт-Петербург
2019 г.

Оглавление

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.

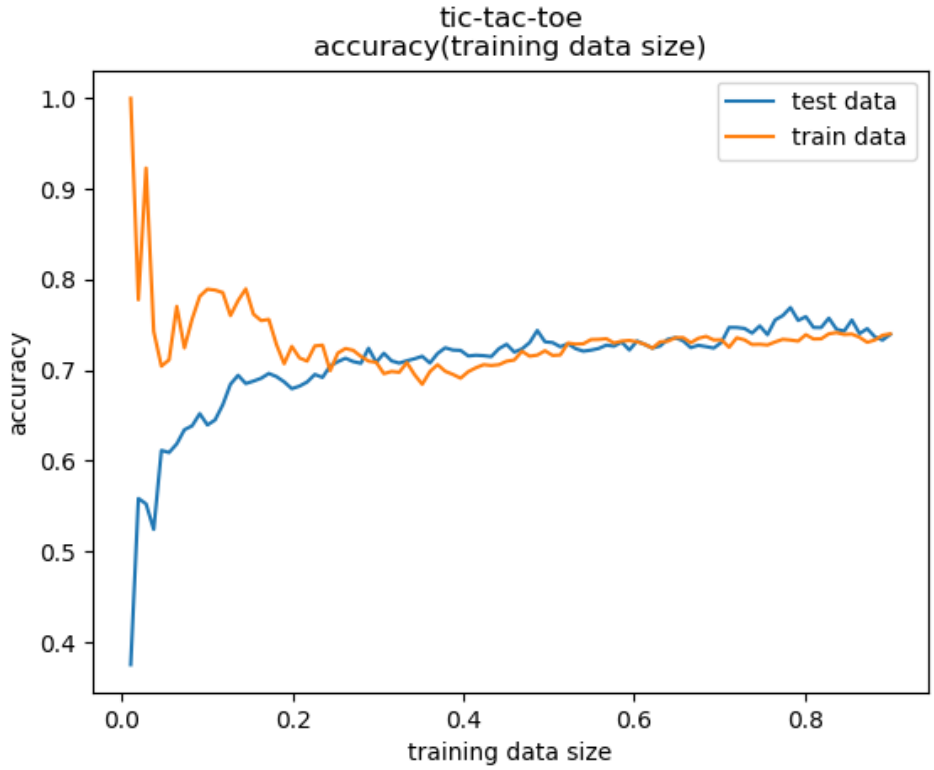
2. Сгенерируйте 100 точек с двумя признаками X1 и X2 в соответствии с нормальным распределением так, что первые 50 точек (class -1) имеют параметры: мат. ожидание X1 равно 10, мат. ожидание X2 равно 14, среднеквадратические отклонения для обеих переменных равны 4. Вторые 50 точек (class +1) имеют параметры: мат. ожидание X1 равно 20, мат. ожидание X2 равно 18, среднеквадратические отклонения для обеих переменных равны 3. Построить соответствующие диаграммы, иллюстрирующие данные. Построить байесовский классификатор и оценить качество классификации.

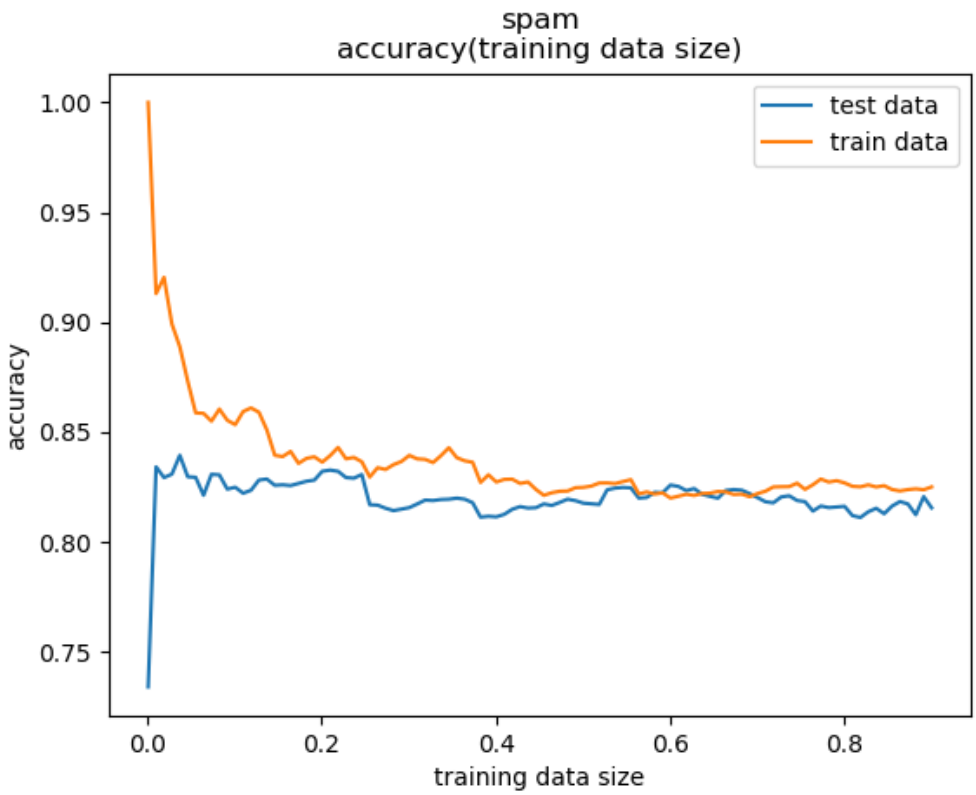
Ход работы

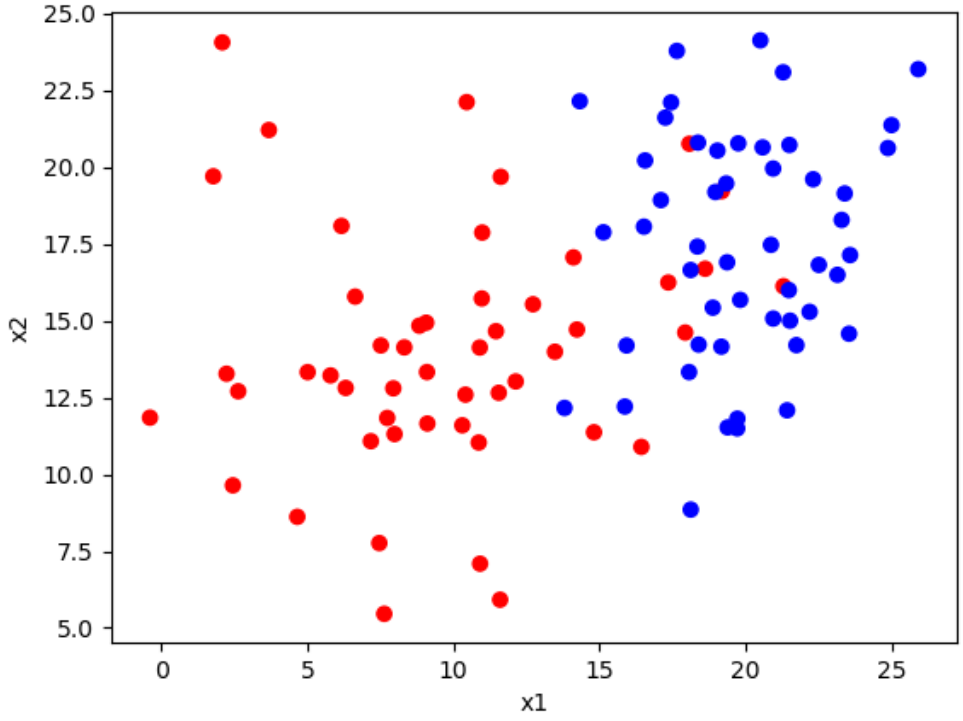
1. Была создана функция которая на входе принимает наборы признаков и классов, а так же коэффициент - отношение объёма обучающей выборки к общее число данных. С использованием функции sklearn.model_selection.train_test_split() входные массивы случайным образом перемешиваются, и разделяются на тестовой и обучающей выборки в соответствие с входного коэффициента. Строится байессовский классификатор (Sklearn.naive_bayes.GaussianNB), классифицируются выборки и на выходе подаются точность классификации на тестовой и на обучающей выборки соответственно. Эта функция далее используется для двух разных наборах данных с многократное варирование объёма обучающей выборки. Результаты выводятся в виде графиков, построени с использованием библиотеки matplotlib.

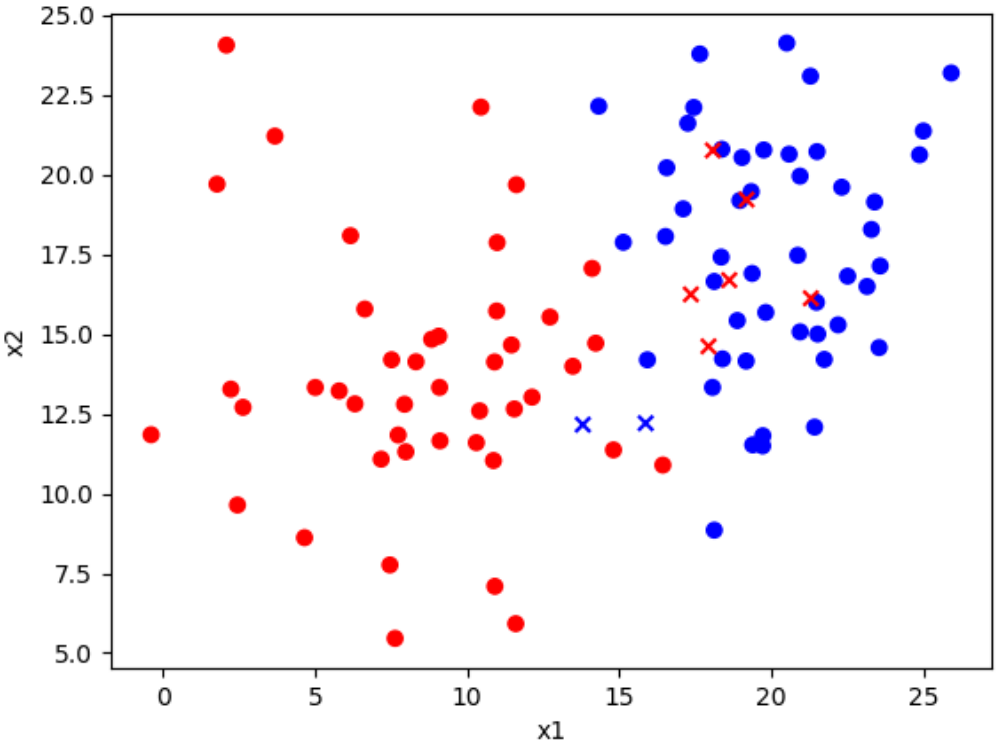
2. С использованием фунцкии np.random.normal библиотеки numpy был сгенерирован набор точек, с требуемые характеристики. Графическое представление этих точек было реализовано с использованием matplotlib.pyplot.scatter. Далее строится байесовский классификатор и оценивается точность классификации с использованием метода перекрестного контроля sklearn.model_selection.cross_val_score

Результаты


tic-tac-toe
accuracy(training data size)
test data
train data
accuracy
training data size
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8


spam
accuracy(training data size)
test data
train data
accuracy
training data size
1.00
0.95
0.90
0.85
0.80
0.75
0.0
0.2
0.4
0.6
0.8






```
Accuracy: 0.89 (+/- 0.04)
```

Вывод

В рамках данной лабораторной работы были получены навыки работы с наивным Байесовским классификатором на Python. Были построени зависимости точности классификации от размера обучающей выборки для два разные наборы данных. Так же был рассмотрен метод перекрестного контроля для оценки качество классификации.

Текст программы

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
from sklearn import metrics
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np

def calculate_accuracy(features, targets, train_size):
    test_size = 1 - train_size
    x_train, x_test, y_train, y_test = \
        train_test_split(features, targets, test_size=test_size, random_state=1)
    gnb = GaussianNB()
    gnb.fit(x_train, y_train)

    #all_x, all_y = [*x_train, *x_test], [*y_train, *y_test]

    return (metrics.accuracy_score(y_test, gnb.predict(x_test)),
            metrics.accuracy_score(y_train, gnb.predict(x_train)))
            #metrics.accuracy_score(all_y, gnb.predict(all_x)))

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.plot(ratios, [acc[2] for acc in accuracies], label='all 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 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 = [calculate_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 = [calculate_accuracy(features, targets_encoded, ratio) for ratio in ratios]
    make_plot(ratios, accuracies, 'spam')

def x1x2():
    np.random.seed(25)
    x1 = [*np.random.normal(10, 4, 50), *np.random.normal(20, 3, 50)]
    x2 = [*np.random.normal(14, 4, 50), *np.random.normal(18, 3, 50)]

    colors = ['red']*50 + ['blue']*50

    plt.figure()
    plt.scatter(x1, x2, color=['red']*50 + ['blue']*50)
    plt.xlabel('x1')
    plt.ylabel('x2')

    x = np.vstack((x1, x2)).T
    y = np.array([-1]*50 + [1]*50)
    indices = np.arange(x.shape[0])
    np.random.shuffle(indices)
    x, y = x[indices], y[indices]
    colors = np.array([colors[indices[i]] for i in range(x.shape[0])])
    scores = cross_val_score(GaussianNB(), x, y, cv = 5)
    print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

    clf = GaussianNB()
    clf.fit(x, y)
    x1, x2 = np.array([v[0] for v in x]), np.array([v[1] for v in x])
    pred_correct = np.isclose(y, clf.predict(x))

    plt.figure()
    plt.scatter(x1[pred_correct==True], x2[pred_correct==True],
                color=colors[pred_correct==True], marker='o')
    plt.scatter(x1[pred_correct==False], x2[pred_correct==False],
                color=colors[pred_correct==False], marker='x')
    plt.xlabel('x1')
    plt.ylabel('x2')

tic_tac_toe()
spam()
x1x2()
plt.show()
```