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

Лабораторная работа №6

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 зависимость тестовой ошибки от количества классификаторов в ансамбле для алгоритмов **sklearn.ensemble.AdaBoostClassifier** и **sklearn.ensemble.GradientBoostingClassifier** на наборе данных vehicle.csv с различными базовыми классификаторами. Постройте график зависимости тестовой ошибки при различном числе классификаторов, объясните полученные результаты.

2) Исследуйте зависимость тестовой ошибки от количества классификаторов в ансамбле для алгоритмов **sklearn.ensemble.BaggingClassifier** и **sklearn.ensemble.RandomForestClassifier** на наборе данных glass.csv с различными базовыми классификаторами для BaggingClassifier. Постройте график зависимости тестовой ошибки при различном числе классификаторов, объясните полученные результаты.

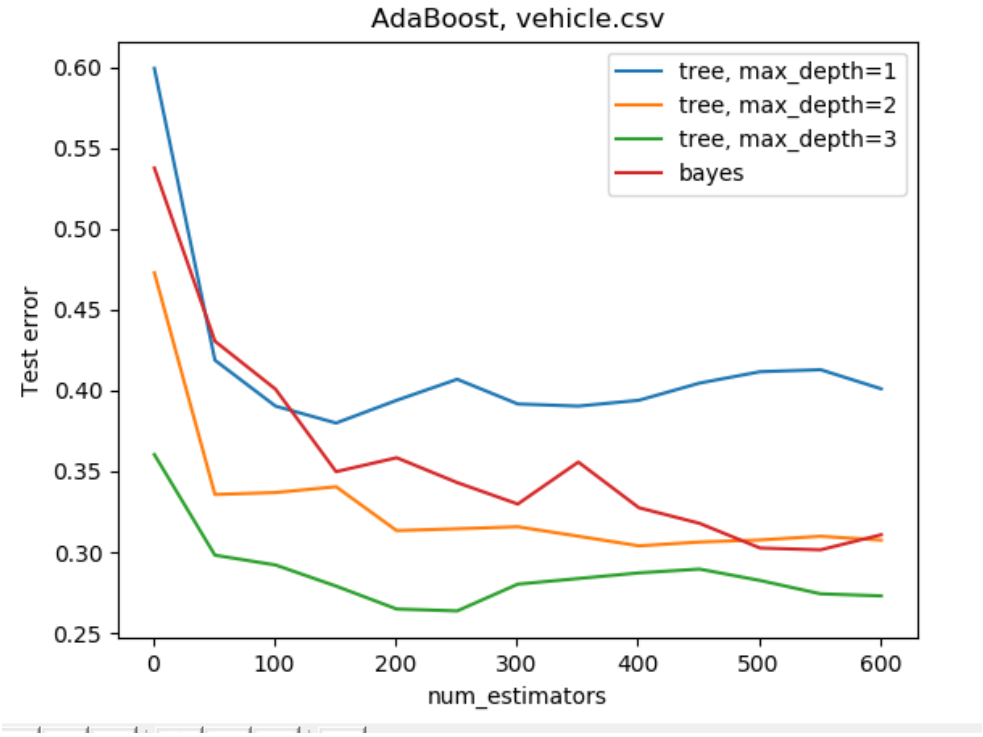
Ход работы

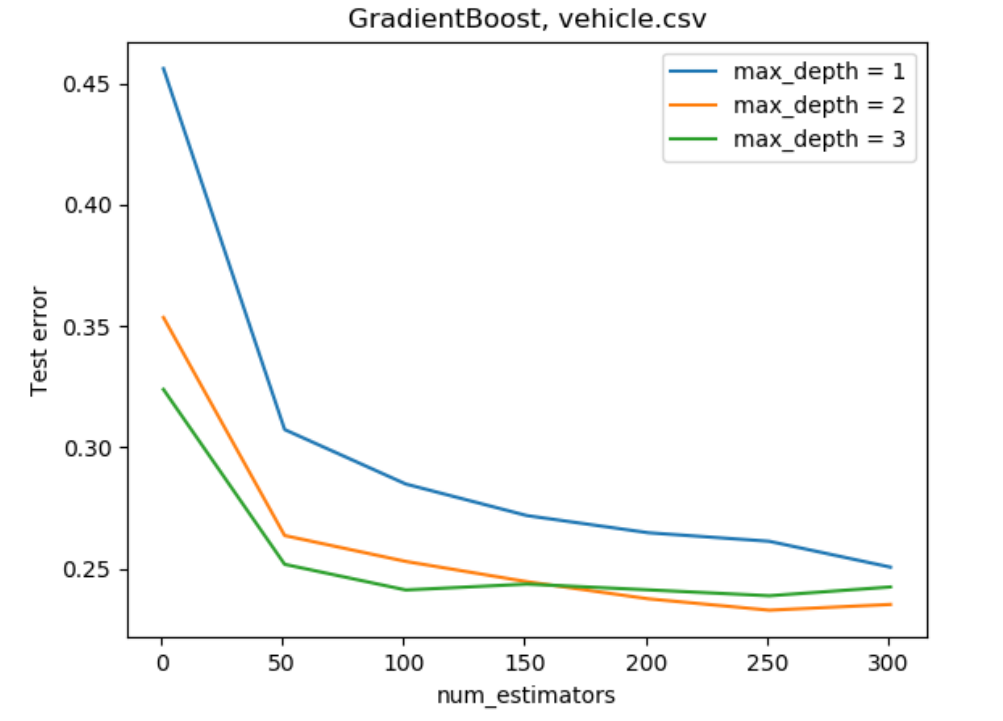
Была создана функция которая для данный классификатор, набор признаков и классов оценивает погрешность на тестирующую выборку с использованием крос-валидации (sklearn.model_selection.cross_validate)

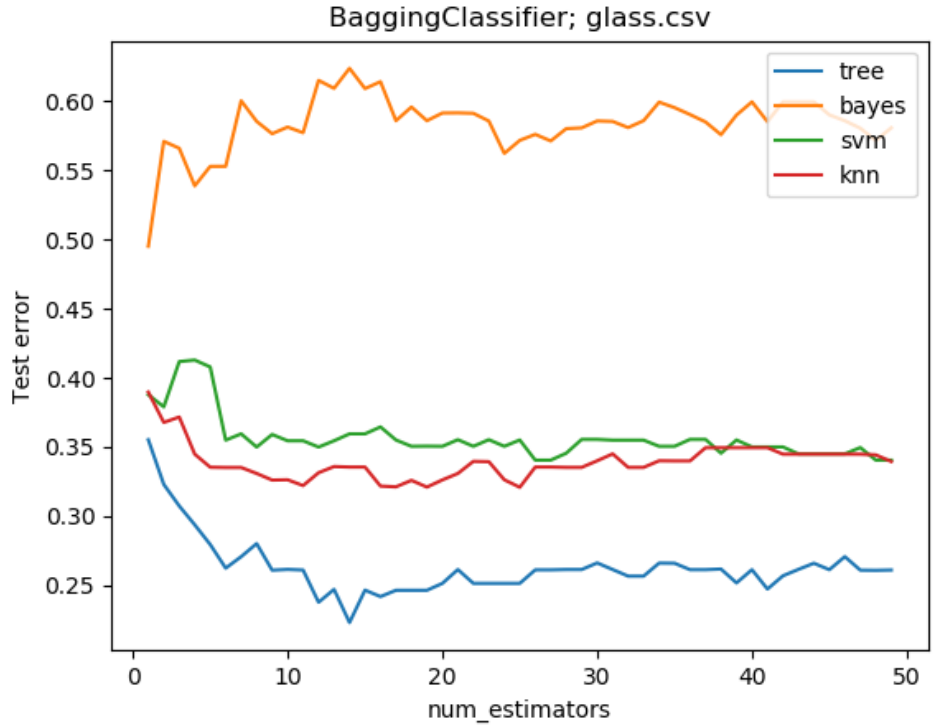
1. Была исследована зависимость тестовой ошибки в зависимости от количества классификаторов в ансамбле на наборе glass. Для алгоритма AdaBoost был использован байесовский классификатор, и деревя решении с различнымы значениямы параметра максимальной глубины (max_depth = 1, 2, 3). По полученными зависимости строются графики с использованием библиотеки matplotplib. По графикам видно что при увеличении количества классификаторов в ансамбле, уменьшается ошибка классификации. При использование деревя решений с максимальную глубину равную 3, получаются найлучшие результаты, так как в данном наборе есть 9 признаков и увеличение глубины позволяет закодировать больше информации, хотя в AdaBoost это быстро приводит к переобучением. На данном наборе GradientBoost работает лучше чем AdaBoost.

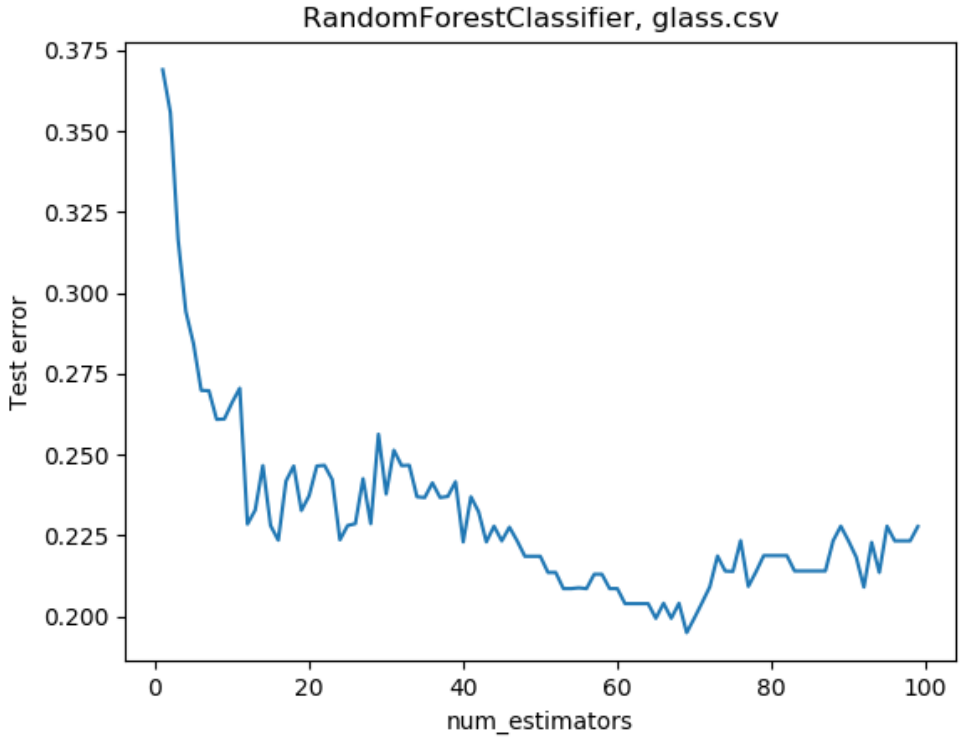
2. Была исследована зависимость тестовой ошибки в зависимости от количества классификаторов в ансамбле на наборе vehicle. Для алгоритма BaggingClassifier были использованы классификаторы NaiveBayes, KNearestNeighbors, SVM, и DecisionTree. Аналогично как и в (1) при многократное изменение количества классификаторов в ансамбле и вычисления погрешности при использованием ранее созданной функции были построены требуемые зависимости. Реализация с байесовским классификатором не эффективна, потому что на данном наборе наивний байесовский классификатор не обеспечивает точность больше чем 0.5. RandomForestClassifier дает лучшие результаты на данном наборе. По графиком так же видно, что использования больше чем 70 классификаторов в ансамбле приводит к переобучением.

Результаты


AdaBoost, vehicle.csv
tree, max_depth=1
tree, max_depth=2
tree, max_depth=3
bayes
Test error
num_estimators
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0
100
200
300
400
500
600


GradientBoost, vehicle.csv
max_depth = 1
max_depth = 2
max_depth = 3
Test error
num_estimators
0.45
0.40
0.35
0.30
0.25
0
50
100
150
200
250
300


BaggingClassifier; glass.csv
tree
bayes
svm
knn
Test error
num_estimators
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0
10
20
30
40
50


RandomForestClassifier, glass.csv
Test error
num_estimators
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.200
0
20
40
60
80
100

Вывод

В ходе данной лабораторной работы был получен опыт работы с алгоритмов классификации AdaBoost, GradientBoost, Bagging и RandomForest. Была исследована зависимость тестовой ошибки от количества классификаторов в ансамбле с использованием разных базовых классификаторов.

Текст программы

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_validate
from sklearn.preprocessing import LabelEncoder

def plotGraphs(xrange, errors, labels, title):

    plt.figure()
    for i in range(len(errors)):
        if labels != None:
            plt.plot(xrange, errors[i], label=labels[i])
        else:
            plt.plot(xrange, errors[i])

    if labels != None:
        plt.legend(loc='upper right')

    plt.xlabel('num_estimators')
    plt.ylabel('Test error')
    plt.title(title)

def calculateTestError(clf, x, y):
    kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
    scores = cross_validate(clf, x, y, scoring='accuracy', return_train_score=False, cv=kfold)
    return 1 - scores['test_score'].mean()

def task1():

    df = pd.read_csv('vehicle.csv')
    x = df.iloc[:, 0:-1].values
    y = LabelEncoder().fit_transform(df.iloc[:, -1].values)

    base_estimators = {'tree, max_depth=1': DecisionTreeClassifier(max_depth=1),
                       'tree, max_depth=2': DecisionTreeClassifier(max_depth=2),
```

```
                      'tree, max_depth=3': DecisionTreeClassifier(max_depth=3),
                      'bayes': GaussianNB()}

    errors = [[] for _ in range(len(base_estimators))]
    for i, (name, base_estimator) in enumerate(base_estimators.items()):
        for n_estimators in range(1, 650, 50):
            clf = AdaBoostClassifier(base_estimator=base_estimator, n_estimators=n_estimators,
random_state=1)
            errors[i].append(calculateTestError(clf, x, y))
    plotGraphs(range(1, 650, 50), errors, list(base_estimators.keys()), 'AdaBoost,
vehicle.csv')

    errors = [[] for _ in range(3)]
    for max_depth in range(1, 4):
        print('.')
        for n_estimators in range(1, 302, 50):
            print(n_estimators, end=' ')
            clf = GradientBoostingClassifier(max_depth=max_depth, n_estimators=n_estimators,
random_state=1)
            errors[max_depth-4].append(calculateTestError(clf, x, y))
    plotGraphs(range(1, 302, 50), errors, [f'max_depth = {i}' for i in range(1, 4)],
'GradientBoost, vehicle.csv')

def task2():
    df = pd.read_csv('glass.csv')
    x = df.iloc[:, 1:-1].values
    y = df.iloc[:, -1].values

    base_estimators = {'tree': DecisionTreeClassifier(),
                       'bayes': GaussianNB(),
                       'svm': SVC(gamma='auto'),
                       'knn': KNeighborsClassifier()}

    errors = [[] for _ in range(len(base_estimators))]
    for i, (name, base_estimator) in enumerate(base_estimators.items()):
        for n_estimators in range(1, 50):
            clf = BaggingClassifier(base_estimator=base_estimator, n_estimators=n_estimators,
random_state=1)
            errors[i].append(calculateTestError(clf, x, y))
    plotGraphs(range(1, 50), errors, list(base_estimators.keys()), 'BaggingClassifier;
glass.csv')

    errors = []
    for n_estimators in range(1, 100):
        clf = RandomForestClassifier(n_estimators=n_estimators, random_state=1)
        errors.append(calculateTestError(clf, x, y))
    plotGraphs(range(1, 100), [errors], None, 'RandomForestClassifier, glass.csv')

task1()
task2()

plt.show()
```