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

Лабораторная работа №4

по дисциплине «Машинное обучение»

Выполнил
студент гр. 3530904/00103 Плетнева А.Д.

Руководитель Селин И.А.

Санкт-Петербург
2023

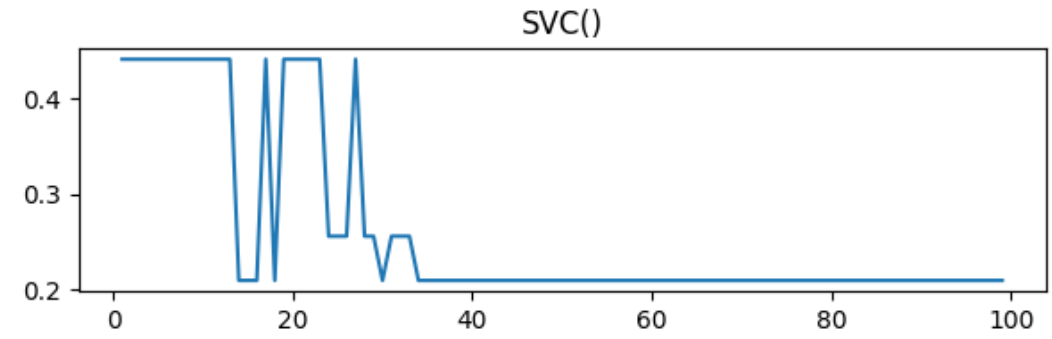
Оглавление

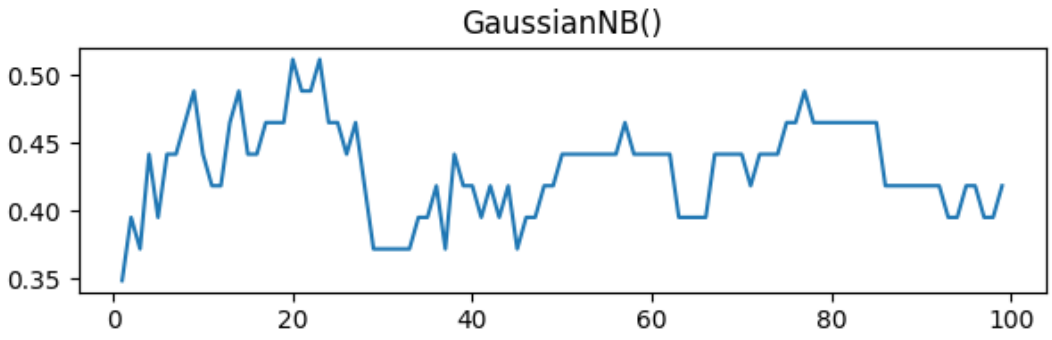
Задание

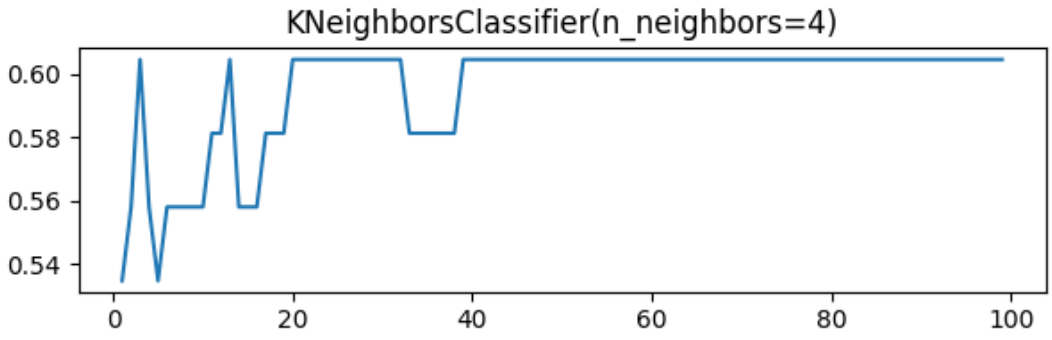
1. Исследуйте зависимость качества классификации от количества классификаторов в ансамбле для алгоритмов бэггинга на наборе данных glass.csv с различными базовыми классификаторами. Постройте графики зависимости качества классификации при различном числе классификаторов, объясните полученные результаты.
2. Исследуйте зависимость качества классификации от количества классификаторов в ансамбле для алгоритма бустинга (например, AdaBoost) на наборе данных vehicle.csv с различными базовыми классификаторами. Постройте графики зависимости качества классификации при различном числе классификаторов, объясните полученные результаты.
3. Постройте мета-классификатор для набора данных titanic_train.csv используя стекинг и оцените качество классификации на titanic_train.csv

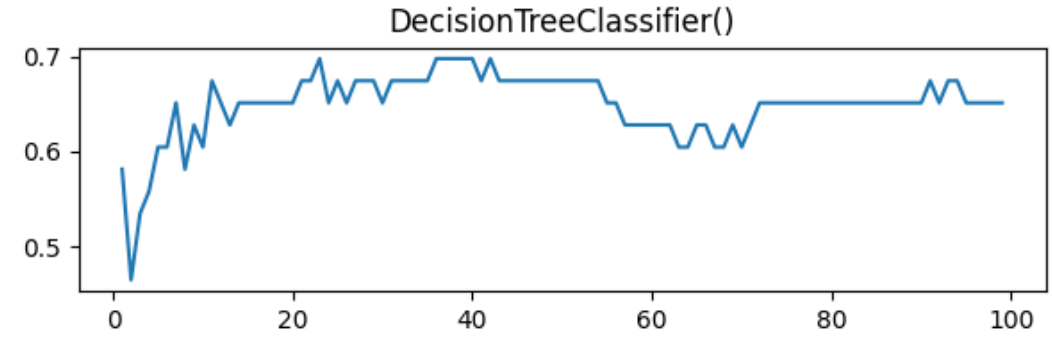
Задание 1

Исследовали зависимость качества классификации от количества классификаторов в ансамбле (от 1 до 100) для следующих классификаторов:
Метод опорных векторов: классифицирует объекты с точностью менее 50%, поэтому точность уменьшается по мере увеличения классификаторов в ансамбле
Наивный Байесовский метод: Тут тоже небольшая точность классификации одного классификатора менее 50%, при увеличении количества классификаторов точность остается низкой (до 50%)
Метод k ближайших соседей и решающее дерево: точность увеличивается при увеличении количества классификаторов

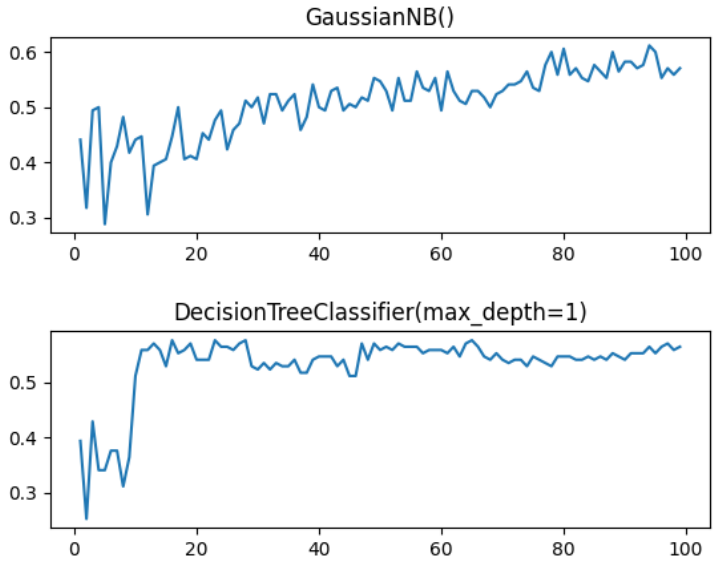








Задание 2



При увеличении числа классификаторов ошибка уменьшается, так как после композиции результатов классификаторов с учетом их весов, получается более точный результат.

Задание 3

В датасете убрали ненужные признаки такие, как 'PassengerId', 'Name', 'Ticket', 'Cabin', также у нас есть missing data в столбце 'Age', чтобы не терять данные, на место пропущенных данных вставим медианные значения возраста.
Используем такие классификаторы, как наивный Байесовский классификатор, SVM, метод к ближайших соседей и деревья решений, стачала по отдельности, а потом все вместе:

```
Для классификатора StackingClassifier(estimators=[('gnb', GaussianNB())]) точность: 0.8044692737430168
Для классификатора StackingClassifier(estimators=[('svc', SVC())]) точность: 0.7262569832402235
Для классификатора StackingClassifier(estimators=[('dt', DecisionTreeClassifier())]) точность: 0.7821229050279329
Для классификатора StackingClassifier(estimators=[('knn', KNeighborsClassifier())]) точность: 0.7430167597765364
Для классификатора StackingClassifier(estimators=[('gnb', GaussianNB()), ('svc', SVC()),
                              ('dt', DecisionTreeClassifier()),
                              ('knn', KNeighborsClassifier())]) точность: 0.8324022346368715
```

Видим, что, когда используем классификаторы по отдельности, точность получается меньше, чем при комбинации классификаторов.

Текст программы

Ex1

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import decomposition
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

df = pd.read_csv('glass.csv')
X = df.iloc[:, 1:-1]
y = df.iloc[:, -1]

X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

classifiers = [SVC(), GaussianNB(), KNeighborsClassifier(n_neighbors=4),
               DecisionTreeClassifier()]

accuracy = {}
for classifier in classifiers:
    accuracy_array = []
    for n_estimators in range(1, 100):
        clf = BaggingClassifier(estimator=classifier,
                                n_estimators=n_estimators,
random_state=0).fit(X_train, Y_train)
        accuracy_array.append(clf.score(X_test, Y_test))
    accuracy[classifier] = accuracy_array

fig, ax = plt.subplots(4, 1)
fig.set_figheight(10)
fig.set_figwidth(7)
plt.subplots_adjust(wspace=0.55, hspace=0.55)
for clf, array in accuracy.items():
    ax[list(accuracy).index(clf)].plot(range(1, 100), array)
    ax[list(accuracy).index(clf)].set_title(clf)

plt.show()
```

ex2

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

df = pd.read_csv('vehicle.csv')

X = df.iloc[:, :-1]
y = df.iloc[:, -1]

X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
classifiers = [GaussianNB(), SVC(), DecisionTreeClassifier(max_depth=1)]

accuracy_of_clf = {}
```

```
for classifier in classifiers:
    if hasattr(classifier, 'predict_proba'):
        accuracy = []
        for i in range(1, 100):
            clf = AdaBoostClassifier(estimator=classifier, n_estimators=i,
random_state=0)
            clf.fit(X_train, Y_train)
            accuracy.append(clf.score(X_test, Y_test))
        accuracy_of_clf[classifier] = accuracy

fig, ax = plt.subplots(len(accuracy_of_clf), 1)
plt.subplots_adjust(wspace=0.5, hspace=0.5)
for key, value in accuracy_of_clf.items():
    ax[list(accuracy_of_clf).index(key)].plot(range(1, 100), value)
    ax[list(accuracy_of_clf).index(key)].set_title(key)

plt.show()
```

ex3

```
import pandas as pd
from sklearn.ensemble import StackingClassifier
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

# загрузка данных
traindf = pd.read_csv('titanic_train.csv')

# удаление ненужных столбцов
traindf.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1,
inplace=True)
imputer = SimpleImputer(strategy='median')
traindf['Age'] = imputer.fit_transform(traindf[['Age']])
X = pd.get_dummies(traindf.drop('Survived', axis=1))
y = pd.factorize(traindf['Survived'])[0]

X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

# определение моделей для использования в стекинге
estimators = [('gnb', GaussianNB()), ('svc', SVC()), ('dt',
DecisionTreeClassifier()), ('knn', KNeighborsClassifier())]

# создание классификатора со стекингом моделей

clfs = [StackingClassifier(estimators=[('gnb', GaussianNB())]),
StackingClassifier(estimators=[('svc', SVC())]),
        StackingClassifier(estimators=[('dt', DecisionTreeClassifier())]),
        StackingClassifier(estimators=[('knn', KNeighborsClassifier())]),
        StackingClassifier(estimators=estimators)]

for clf in clfs:
    clf.fit(X_train, Y_train)

    print(f"Для классификатора {clf} точность: {clf.score(X_test, Y_test)}")
```