Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Отчет по лабораторной работе №5

«Ансамбли моделей машинного обучения. Часть 1»

по дисциплине «Технологии машинного обучения»

Выполнил:
студент группы ИУ5-64Б
Стукалов И.Д.
10.05.2024

Проверил:
Гапанюк Ю.Е.

2024 г.

Задание.

Выберите набор данных (датасет) для решения задачи классификации или регресии.

В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.

С использованием метода train_test_split разделите выборку на обучающую и тестовую.

Обучите следующие ансамблевые модели:

- две модели группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
- AdaBoost;
- градиентный бустинг.

Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Текст программы

Лабораторная работа 5. Ансамбли моделей машинного обучения. Часть 1.

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.ensemble import AdaBoostClassifier
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import BaggingClassifier
from IPython.display import Image
import pandas as pd
import graphviz
import pydotplus
from io import StringIO
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
```

```
[ ] from operator import itemgetter

    def draw_feature_importances(tree_model, X_dataset, figsize=(10,5)):
        """
        Вывод важности признаков в виде графика
        """
        # Сортировка значений важности признаков по убыванию
        list_to_sort = list(zip(X_dataset.columns.values, tree_model.feature_importances_))
        sorted_list = sorted(list_to_sort, key=itemgetter(1), reverse = True)
        # Названия признаков
        labels = [x for x,_ in sorted_list]
        # Важности признаков
        data = [x for _,x in sorted_list]
        # Вывод графика
        fig, ax = plt.subplots(figsize=figsize)
        ind = np.arange(len(labels))
        plt.bar(ind, data)
        plt.xticks(ind, labels, rotation='vertical')
        # Вывод значений
        for a,b in zip(ind, data):
            plt.text(a-0.05, b+0.01, str(round(b,3)))
        plt.show()
        return labels, data
```

Выберите набор данных (датасет) для решения задачи классификации или регресии.

```
[ ] # Используем датасет iris с двумя первыми признаками
    iris = load_iris()
    iris_X = iris.data[:, :2]
    iris_y = iris.target
```

С использованием метода train_test_split разделите выборку на обучающую и тестовую.

```
[ ] X_train, X_test, y_train, y_test = train_test_split(
        iris_X, iris_y, test_size=0.2, random_state=1)
```

Бэггинг

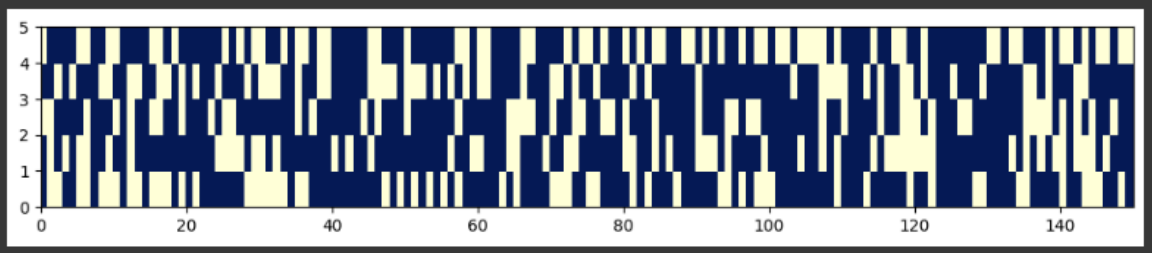
```
# Обучим классификатор на 5 деревьях
bg = BaggingClassifier(n_estimators=5, oob_score=True, random_state=10)
bg.fit(iris_X, iris_y)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_bagging.py:789: UserWarning: Some inputs do not have OOB scor
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_bagging.py:795: RuntimeWarning: invalid value encountered in
  oob_decision_function = predictions / predictions.sum(axis=1)[:, np.newaxis]
```

```
BaggingClassifier(n_estimators=5, oob_score=True, random_state=10)
```

```
[ ] # Сконвертируем эти данные в двоичную матрицу,
    # 1 соответствует элементам, попавшим в обучающую выборку
    bin_array = np.zeros((5, iris_X.shape[0]))
    for i in range(5):
        for j in bg.estimators_samples_[i]:
            bin_array[i][j] = 1

    fig, ax = plt.subplots(figsize=(12,2))
    ax.pcolor(bin_array, cmap='YlGnBu')
    plt.show()
```




```
[ ] # Out-of-bag error, возвращаемый классификатором
    # Для классификации используется метрика accuracy
    bg.oob_score_, 1-bg.oob_score_
```

```
(0.6933333333333334, 0.30666666666666664)
```

```
[ ] # точность предсказания
```

```
(0.6953333333333334, 0.5066666666666664)
```

```
[ ] # точность предсказания
    bagging_pred = bg.predict(X_test)
    accuracy_score(y_test,bagging_pred)
```

```
0.8666666666666667
```

Случайный лес

```
# Обучим классификатор на 5 деревьях
tree1 = RandomForestClassifier(n_estimators=5, oob_score=True, random_state=10)
tree1.fit(iris_X, iris_y)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:583: UserWarning: Some inputs do not have OOB score
  warn(
```

```
RandomForestClassifier(n_estimators=5, oob_score=True, random_state=10)
```

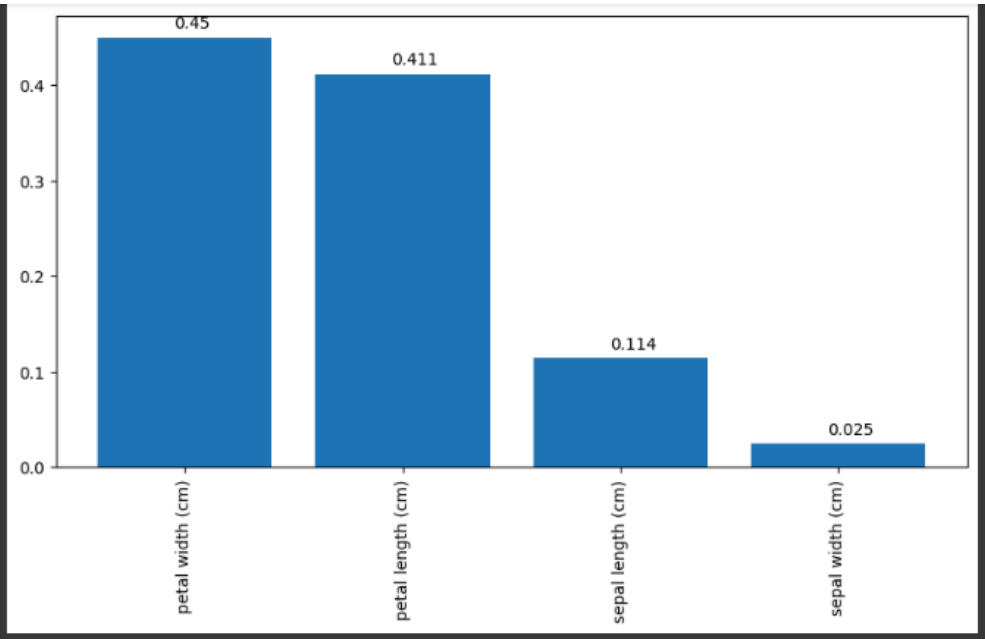
```
[ ] rf_pred = tree1.predict(X_test)
```

```
[ ] # Точность
    accuracy_score(y_test,rf_pred)
```

```
0.9
```

```
[ ] # Важность признаков
    iris_x_ds = pd.DataFrame(data=iris['data'], columns=iris['feature_names'])
    iris_rf_cl = RandomForestClassifier(random_state=1)
    iris_rf_cl.fit(iris_x_ds, iris.target)
    _,_ = draw_feature_importances(iris_rf_cl, iris_x_ds)
```

[ ]



AdaBoost

```
# Используем датасет iris с двумя первыми признаками
iris = load_iris()
iris_X = iris.data
iris_y = iris.target

X_train, X_test, y_train, y_test = train_test_split(
    iris_X, iris_y, test_size=0.2, random_state=1)

# Обучим классификатор на 5 деревьях
abl = AdaBoostClassifier(n_estimators=5, algorithm='SAMME', random_state=10)
abl.fit(iris_X, iris_y)
```

```
AdaBoostClassifier
AdaBoostClassifier(algorithm='SAMME', n_estimators=5, random_state=10)
```

[ ]
```
ada_pred = abl.predict(X_test)
```

[ ]
```
# Точность
accuracy_score(y_test,ada_pred)
```

```
0.9666666666666667
```

Градиентный бустинг

```
[ ] # Используем датасет iris с двумя первыми признаками
    iris = load_iris()
    iris_X = iris.data
    iris_y = iris.target

    X_train, X_test, y_train, y_test = train_test_split(
        iris_X, iris_y, test_size=0.2, random_state=1)

    # Обучим классификатор на 5 деревьях
    bcl = GradientBoostingClassifier(random_state=10)
    bcl.fit(X_train, y_train)
```

```
▾ GradientBoostingClassifier
GradientBoostingClassifier(random_state=10)
```

```
[ ] boosting_pred = bcl.predict(X_test)
```

```
[ ] accuracy_score(y_test,boosting_pred)
```

```
0.9666666666666667
```