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

по дисциплине «Технология машинного обучения»

Выполнил:
студент группы ИУ5-63Б Кузнецов В.А.
подпись, дата

Проверил:
Гапанюк Ю.Е.
подпись, дата

2024 г.

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите следующие ансамблевые модели:
5. две модели группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
6. AdaBoost;
7. градиентный бустинг.
8. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Текст программы:

Основные характеристики датасета

MedInc - медианный доход в районе

HouseAge - средний возраст домов в районе

AveRooms - среднее количество комнат на дом

AveBedrms - среднее количество спален на дом

Population - население района

AveOccup - среднее количество жителей на дом

Latitude - географическая широта района

Longitude - географическая долгота района

MedHouseVal - медианная стоимость домов в районе (целевая переменная)

Подготовка

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.ensemble import BaggingRegressor, RandomForestRegressor, AdaBoostRegressor, GradientBoostingRegressor
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor

RANDOM_STATE=123


# Загрузка данных
california = fetch_california_housing()
data = pd.DataFrame(data= np.c_[california['data'], california['target']],
                    columns= california['feature_names'] + ['target'])


data.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

```
data.isnull().sum()
```

```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
target        0
dtype: int64
```

```
data.describe()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.631861	-119.569704	2.068558
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952	2.003532	1.153956
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.540000	-124.350000	0.149990
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.930000	-121.800000	1.196000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.260000	-118.490000	1.797000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.710000	-118.010000	2.647250
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41.950000	-114.310000	5.000010

Разделение на выборки

```
X = data.iloc[:, :-1]
y = data.target


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=RANDOM_STATE)


# Масштабирование
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Обучение моделей

```
# Бэггинг
bagging_params = {
    'n_estimators':[5, 10, 50, 100]
}

grid_search = GridSearchCV(estimator=BaggingRegressor(random_state=RANDOM_STATE), param_grid=bagging_params, cv=3)
grid_search.fit(X_train_scaled, y_train)

print(f"Лучшие параметры: {grid_search.best_params_}")
bagging = grid_search.best_estimator_
```

```
Лучшие параметры: {'n_estimators': 100}
```

```
# Случайный лес
tree_params = {
    'n_estimators':[50, 100, 150, 200]
}

grid_search = GridSearchCV(estimator=RandomForestRegressor(random_state=RANDOM_STATE), param_grid=tree_params, cv=3)
grid_search.fit(X_train_scaled, y_train)

print(f"Лучшие параметры: {grid_search.best_params_}")
random_forest = grid_search.best_estimator_
```

```
Лучшие параметры: {'n_estimators': 200}
```

```
# AdaBoost
adaboost_params = {
    'n_estimators':[30, 50, 100, 150]
}

grid_search = GridSearchCV(estimator=AdaBoostRegressor(estimator=DecisionTreeRegressor(), random_state=RANDOM_STATE), param_grid=adaboos
grid_search.fit(X_train_scaled, y_train)

print(f"Лучшие параметры: {grid_search.best_params_}")
adaboost = grid_search.best_estimator_
```

```
Лучшие параметры: {'n_estimators': 150}
```

```
grid_search = GridSearchCV(estimator=AdaBoostRegressor(random_state=RANDOM_STATE), param_grid=adaboost_params, cv=3)
grid_search.fit(X_train_scaled, y_train)


# Градиентный бустинг
gradient_params = {
    'n_estimators':[50, 100, 150, 200]
}

grid_search = GridSearchCV(estimator=GradientBoostingRegressor(random_state=RANDOM_STATE), param_grid=gradient_params, cv=3)
grid_search.fit(X_train_scaled, y_train)

print(f"Лучшие параметры: {grid_search.best_params_}")
gradient_boosting = grid_search.best_estimator_
```

```
Лучшие параметры: {'n_estimators': 200}
```

Оценка моделей

```
y_pred_bagging = bagging.predict(X_test_scaled)
y_pred_rf = random_forest.predict(X_test_scaled)
y_pred_adaboost = adaboost.predict(X_test_scaled)
y_pred_adaboost_limited = adaboost_limited_tree_depth.predict(X_test_scaled)
y_pred_gb = gradient_boosting.predict(X_test_scaled)


# MAE
print(f"Bagging: {mean_absolute_error(y_test, y_pred_bagging):.4f}")
print(f"Random Forest: {mean_absolute_error(y_test, y_pred_rf):.4f}")
print(f"AdaBoost: {mean_absolute_error(y_test, y_pred_adaboost):.4f}")
print(f"AdaBoost (tree depth = 3): {mean_absolute_error(y_test, y_pred_adaboost_limited):.4f}")
print(f"Gradient Boosting: {mean_absolute_error(y_test, y_pred_gb):.4f}")
```

```
Bagging: 0.3243
Random Forest: 0.3223
AdaBoost: 0.2902
AdaBoost (tree depth = 3): 0.7656
Gradient Boosting: 0.3376
```

```
# R^2
print(f"Bagging: {r2_score(y_test, y_pred_bagging):.4f}")
print(f"Random Forest: {r2_score(y_test, y_pred_rf):.4f}")
print(f"AdaBoost: {r2_score(y_test, y_pred_adaboost):.4f}")
print(f"AdaBoost (tree depth = 3): {r2_score(y_test, y_pred_adaboost_limited):.4f}")
print(f"Gradient Boosting: {r2_score(y_test, y_pred_gb):.4f}")
```

```
Bagging: 0.8127
Random Forest: 0.8142
AdaBoost: 0.8320
AdaBoost (tree depth = 3): 0.4136
Gradient Boosting: 0.8178
```

Вывод

На основе анализа метрик MAE и R^2, AdaBoost является лучшей моделью для данной задачи, так как она показала наименьшую среднюю абсолютную ошибку и наибольшее значение коэффициента детерминации.

Однако по умолчанию в AdaBoost используется DecisionTreeRegressor с ограничением глубины дерева, что сильно ухудшает результаты.

Остальные модели показали примерно одинаковый результат.