Московский государственный технический

университет им. Н. Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №5
«Ансамбли моделей машинного обучения»

Выполнил:
студент группы РТ5-61Б

Алиев Тимур
Подпись и дата:

Проверил:
доцент каф. ИУ5

Гапанюк Ю.Е.
Подпись и дата:

Москва, 2022 г.

Описание задания

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:
   - одну из моделей группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
   - одну из моделей группы бустинга;
   - одну из моделей группы стекинга.

Jupyter notebook

Лабораторная работа №5. Алиев Тимур РТ5-61Б.

Подготовка данных

Ввод [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import PolynomialFeatures, MinMaxScaler, StandardScaler
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.tree import DecisionTreeRegressor, export_graphviz, export_text
from sklearn.svm import SVR
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split, GridSearchCV
from IPython.display import Image
from IPython.core.display import HTML
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from heamy.dataset import Dataset
from heamy.estimator import Regressor
from heamy.pipeline import ModelsPipeline
from sklearn.neural_network import MLPRegressor
from gmdhpy import gmdh
from warnings import simplefilter

simplefilter('ignore')
```

Ввод [2]:
```
data = pd.read_csv('WineQT.csv')
data.head()
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	2
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	3
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	4

Ввод [3]:
```
# Корреляция с алкоголем по модулю - top
best_params = data.corr()['alcohol'].map(abs).sort_values(ascending=False)[1:]
best_params = best_params[best_params.values > 0.35]
best_params
```

Out[3]:
```
density    0.494727
quality    0.484866
Name: alcohol, dtype: float64
```

Разделение выборки на обучающую и тестовую

Ввод [4]:
```
x_train, x_test, y_train, y_test = train_test_split(data[best_params.index], data['alcohol'], test_size=0.3, random_state=3)
```

Масштабирование данных

Ввод [5]:
```
scaler = StandardScaler().fit(x_train)
x_train_scaled = pd.DataFrame(scaler.transform(x_train), columns=x_train.columns)
x_test_scaled = pd.DataFrame(scaler.transform(x_test), columns=x_train.columns)
```

Ввод [6]:
```
# функция для вывода метрики
def print_metrics(y_test, y_pred):
    print(f"R^2: {r2_score(y_test, y_pred)}")
    print(f"MSE: {mean_squared_error(y_test, y_pred)}")
    print(f"MAE: {mean_absolute_error(y_test, y_pred)}")
```

1) Случайный лес

Ввод [7]:
```
print_metrics(y_test, RandomForestRegressor(random_state=17).fit(x_train, y_train).predict(x_test))
```

```
R^2: 0.4363568915200067
MSE: 0.6606961915609506
MAE: 0.6317937744229445
```

Ввод [8]:
```
# Подбор гиперпараметров
rf = RandomForestRegressor(random_state=17)
params = {'n_estimators': [100, 1000], 'criterion': ['squared_error', 'absolute_error', 'poisson'],
          'max_features': ['auto', 'sqrt'], 'min_samples_leaf': [1, 3, 5]}
grid_cv = GridSearchCV(estimator=rf, cv=5, param_grid=params, n_jobs=-1, scoring='neg_mean_absolute_error')
grid_cv.fit(x_train, y_train)
print(grid_cv.best_params_)
```

```
{'criterion': 'absolute_error', 'max_features': 'sqrt', 'min_samples_leaf': 5, 'n_estimators': 1000}
```

Ввод [9]:
```
best_rf = grid_cv.best_estimator_
best_rf.fit(x_train, y_train)
y_pred_rf = best_rf.predict(x_test)
print_metrics(y_test, y_pred_rf)
```

```
R^2: 0.4900907703255626
MSE: 0.5977099356296611
MAE: 0.601562973760933
```

2) Градиентный бустинг

Ввод [10]:
```
print_metrics(y_test, GradientBoostingRegressor(random_state=17).fit(x_train, y_train).predict(x_test))
```

```
R^2: 0.5187256098228614
MSE: 0.564144494808706
MAE: 0.5925955880465614
```

Ввод [11]:
```
# Подбор гиперпараметров
gb = GradientBoostingRegressor(random_state=17)
params = {'loss': ['squared_error', 'absolute_error', 'huber'], 'n_estimators': [10, 50, 100, 200],
          'criterion': ['friedman_mse', 'squared_error', 'mse', 'mae'], 'min_samples_leaf': [1, 3, 5]}
grid_cv = GridSearchCV(estimator=gb, cv=5, param_grid=params, n_jobs=-1, scoring='r2')
grid_cv.fit(x_train, y_train)
print(grid_cv.best_params_)
```

```
{'criterion': 'friedman_mse', 'loss': 'squared_error', 'min_samples_leaf': 1, 'n_estimators': 50}
```

Ввод [12]:
```
best_gb = grid_cv.best_estimator_
best_gb.fit(x_train, y_train)
y_pred_gb = best_gb.predict(x_test)
print_metrics(y_test, y_pred_gb)
```

```
R^2: 0.5265950960016097
MSE: 0.5549199705137814
MAE: 0.589858793540792
```

3) Стекинг

Ввод [13]:
```
dataset = Dataset(x_train, y_train, x_test)
```

Ввод [14]:
```
model_lr = Regressor(dataset=dataset, estimator=LinearRegression, name='lr')
model_rf = Regressor(dataset=dataset, estimator=RandomForestRegressor,
                     parameters={'criterion': 'absolute_error', 'n_estimators': 1000, 'random_state': 17}, name='rf')
model_gb = Regressor(dataset=dataset, estimator=GradientBoostingRegressor,
                     parameters={'loss': 'huber', 'random_state': 17}, name='rf')
```

Ввод [15]:
```
pipeline = ModelsPipeline(model_lr, model_rf)
stack_ds = pipeline.stack(k=10, seed=1)
stacker = Regressor(dataset=stack_ds, estimator=GradientBoostingRegressor)
results = stacker.validate(k=10, scorer=mean_absolute_error)
```

```
Metric: mean_absolute_error
Folds accuracy: [0.6194110924984411, 0.661961907062875, 0.6253281354764146, 0.647977261569514, 0.46334402329637425, 0.542899450
9989188, 0.5361135714228367, 0.47879256948668053, 0.4813988120354393, 0.634546756726363]
Mean accuracy: 0.5691773580573857
Standard Deviation: 0.07323033099064555
Variance: 0.005362681376999502
```

Ввод [16]:
```
y_pred_stack = stacker.predict()
print_metrics(y_test, y_pred_stack)
```

```
R^2: 0.4654915465854482
MSE: 0.6265448724823064
MAE: 0.6213646378838801
```

4) Многослойный персептрон ¶

Ввод [17]:
```
print_metrics(y_test, MLPRegressor(random_state=17).fit(x_train, y_train).predict(x_test))
```

```
R^2: 0.30581687622485887
MSE: 0.8137137476247516
MAE: 0.7275229510423318
```

Ввод [18]:
```
# Подбор гиперпараметров
mlp = MLPRegressor(random_state=17)
params = {'solver': ['lbfgs', 'sgd', 'adam'], 'hidden_layer_sizes': [(100,), (50, 30,), (100, 40,)],
          'alpha': [1e-4, 3e-4, 5e-4], 'max_iter': [500, 1000]}
grid_cv = GridSearchCV(estimator=mlp, cv=5, param_grid=params, n_jobs=-1, scoring='r2')
grid_cv.fit(x_train, y_train)
print(grid_cv.best_params_)
```

```
{'alpha': 0.0005, 'hidden_layer_sizes': (100,), 'max_iter': 500, 'solver': 'lbfgs'}
```

Ввод [19]:
```
best_mlp = grid_cv.best_estimator_
best_mlp.fit(x_train, y_train)
y_pred_mlp = best_mlp.predict(x_test)
print_metrics(y_test, y_pred_mlp)
```

```
R^2: 0.4791641439073404
MSE: 0.6105180057586421
MAE: 0.6211604340563389
```

5) Метод группового учёта аргументов

Ввод [20]:
```
gm = gmdh.Regressor(n_jobs=-1)
gm.fit(np.array(x_train_scaled), np.array(y_train))
y_pred_gm = gm.predict(np.array(x_test_scaled))
print()
print_metrics(y_test, y_pred_gm)
```

```
train layer0 in 0.00 sec
train layer1 in 0.00 sec
train layer2 in 0.00 sec
train layer3 in 0.00 sec
train layer4 in 0.00 sec
train layer5 in 0.00 sec
train layer6 in 0.00 sec

R^2: 0.4630314373942027
MSE: 0.6294285851526578
MAE: 0.6345983648345276
```

Сравнение моделей

Ввод [21]:
```
print("Случайный лес")
print_metrics(y_test, y_pred_rf)

print("\nГрадиентный бустинг")
print_metrics(y_test, y_pred_gb)

print("\nСтекинг")
print_metrics(y_test, y_pred_stack)

print("\nМногослойный персептрон")
print_metrics(y_test, y_pred_mlp)

print("\nМетод группового учёта аргументов")
print_metrics(y_test, y_pred_gm)
```

```
Случайный лес
R^2: 0.4900907703255626
MSE: 0.5977099356296611
MAE: 0.601562973760933

Градиентный бустинг
R^2: 0.5265950960016097
MSE: 0.5549199705137814
MAE: 0.589858793540792

Стекинг
R^2: 0.4654915465854482
MSE: 0.6265448724823064
MAE: 0.6213646378838801

Многослойный персептрон
R^2: 0.4791641439073404
MSE: 0.6105180057586421
MAE: 0.6211604340563389

Метод группового учёта аргументов
R^2: 0.4630314373942027
MSE: 0.6294285851526578
MAE: 0.6345983648345276
```