Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №4

«Ансамбли моделей машинного обучения»

Выполнил:
студент группы ИУ5-62Б
Перова Анна

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

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.

5. **(+1 балл на экзамене)** Дополнительно к указанным моделям обучите еще две модели:

- Модель многослойного персептрона. По желанию, вместо библиотеки scikit-learn возможно использование библиотек TensorFlow, PyTorch или других аналогичных библиотек.

- Модель МГУА с использованием библиотеки
- https://github.com/kvoyager/GmdhPy (или аналогичных библиотек). Найдите такие параметры запуска модели, при которых она будет по крайней мере не хуже, чем одна из предыдущих ансамблевых моделей.

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

Описание датасета

Датасет `SHOP-SALES.xlsx` содержит информацию о продажах магазина.

Параметры покупки:

- *Date* - дата покупки,
- *Hour* - час совершения покупки,
- *Product* - тип купленного товара,
- *Gender* - пол,
- *Color* - цвет товара,
- *Size* - размер товара,
- *Price* - цена товара без учета НДС,
- *Vat* - НДС,
- *Total* - Цена товара с НДС.

Подключение библиотек для анализа данных

```
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
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from heamy.estimator import Regressor
from heamy.pipeline import ModelsPipeline
from heamy.dataset import Dataset
from sklearn.neural_network import MLPRegressor
from gmdhpy import gmdh
from warnings import simplefilter
from importlib import reload

simplefilter('ignore')
```

Загрузка датасета из файла `shop-sales.csv`

```
data = pd.read_excel('SHOP-SALES.xlsx')
```

Кодирование категорий товаров и времени суток

```
le = LabelEncoder()
data['TIME OF DAY'] = le.fit_transform(data['TIME OF DAY'])
```

```
data['GENDER'] = le.fit_transform(data[ 'GENDER'])
```

```
data['PRODUCT'] = le.fit_transform(data['PRODUCT'])
```

```
data['SALE CONSULTANT'] = le.fit_transform(data['SALE CONSULTANT'])
```

```
data['DAY_OF_WEEK'] = le.fit_transform(data['DAY_OF_WEEK'])
```

```
data['CATEGORY'] = le.fit_transform(data['CATEGORY'])
```

```
data['TIME OF DAY'].unique()
```

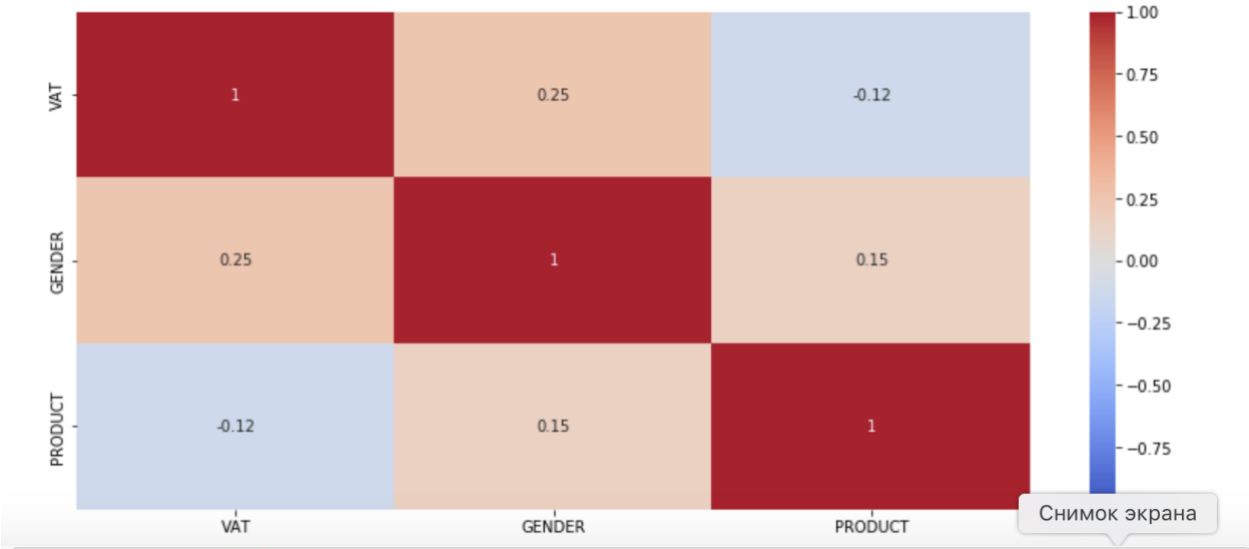
```
array([2, 1, 0, 3])
```

```
print('Признаки, имеющие максимальную по модулю корреляцию с ценой товара')
best_params = data.corr()['PRICE'].map(abs).sort_values(ascending=False)[2:]
best_params = best_params[best_params.values >=0.15]
best_params
```

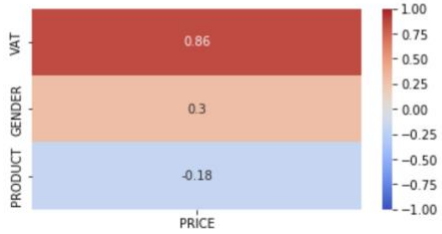
```
Признаки, имеющие максимальную по модулю корреляцию с ценой товара

VAT        0.856014
GENDER     0.297276
PRODUCT    0.180018
Name: PRICE, dtype: float64
```

```
plt.figure(figsize=(14, 6))
sns.heatmap(data[best_params.index].corr(), vmin=-1, vmax=1, cmap='coolwarm', annot=True)
plt.show()
```




```
plt.figure(figsize=(6, 3))
sns.heatmap(pd.DataFrame(data[np.append(best_params.index.values, 'PRICE')].corr()['PRICE'].sort_values(ascending=False
plt.show()
```



Разделим выборку на обучающую и тестовую

```
y = data['PRICE']
X = data[best_params.index]
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
```

Масшабирование данных

```
scaler = MinMaxScaler().fit(x_train)
x_train = pd.DataFrame(scaler.transform(x_train), columns=x_train.columns)
x_test = pd.DataFrame(scaler.transform(x_test), columns=x_train.columns)
x_train.describe()
```

	VAT	GENDER	PRODUCT
count	6987.000000	6987.000000	6987.000000
mean	0.196925	0.620390	0.575511
std	0.099331	0.323928	0.254893
min	0.000000	0.000000	0.000000
25%	0.143203	0.666667	0.354839
50%	0.179003	0.666667	0.612903
75%	0.250605	0.666667	0.806452
max	1.000000	1.000000	1.000000

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)}")
```

Модель №1: Случайный лес

```
print_metrics(y_test, RandomForestRegressor(random_state=17).fit(x_train, y_train).predict(x_test))
```

```
R^2: 0.9594747980845331
MSE: 22.05461159488202
MAE: 0.7534983109341062
```

Модель №2: Градиентный бустинг

```
print_metrics(y_test, GradientBoostingRegressor(random_state=17).fit(x_train, y_train).predict(x_test))
```

```
R^2: 0.959934648904529
MSE: 21.804351738120452
MAE: 1.0483068603354821
```

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': 'squared_error', 'loss': 'squared_error', 'min_samples_leaf': 1, 'n_estimators': 200}
```

```
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: 0.9593660392953846

MSE: 22.11380026609956

MAE: 1.0042704624683885

Модель №3: Стекинг

```
dataset = Dataset(x_train, y_train, x_test)
```

```
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')
```

```
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: [1.0070100419260326, 0.763195756894537, 0.9223319545914171, 1.346861667275437, 1.1195676259693035, 1.
0003180235131586, 0.960406475666008, 1.0528317574502681, 0.9713982464630863, 1.1320870497325028]
Mean accuracy: 1.0276008599481752
Standard Deviation: 0.14552424751230889
Variance: 0.02117730661402374
```

```
y_pred_stack = stacker.predict()
print_metrics(y_test, y_pred_stack)
```

```
R^2: 0.9595464482164046
MSE: 22.015618184498948
MAE: 0.90738655634562
```

Модель №4: Многослойный персептрон

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

```
R^2: 0.7568696875900107
MSE: 132.31629587755637
MAE: 6.456742762877424
```

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': 0.0001, 'hidden_layer_sizes': (50, 30), 'max_iter': 1000, 'solver': 'adam'}
```

```
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: 0.9484277791823769
MSE: 28.066616462287424
MAE: 1.5096108267655122
```

Модель №5: Метод группового учёта аргументов

```
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 0.01 sec
train layer1 in 0.05 sec
train layer2 in 0.04 sec
train layer3 in 0.05 sec
train layer4 in 0.04 sec
train layer5 in 0.05 sec
train layer6 in 0.04 sec
train layer7 in 0.04 sec
train layer8 in 0.03 sec

R^2: 0.9494277791823769
MSE: 28.076616462287424
MAE: 1.5196108267655122
```

```
Случайный лес
R^2: 0.9594747980845331
MSE: 22.05461159488202
MAE: 0.7534983109341062

Градиентный бустинг
R^2: 0.9593660392953846
MSE: 22.11380026609956
MAE: 1.0042704624683885

Стекинг
R^2: 0.9595464482164046
MSE: 22.015618184498948
MAE: 0.90738655634562

Многослойный персептрон
R^2: 0.7568696875900107
MSE: 132.31629587755637
MAE: 6.456742762877424

Метод группового учёта аргументов
R^2: 0.9494277791823769
MSE: 28.076616462287424
MAE: 1.5196108267655122
```