Лабораторная работа № 6

Ансамбли моделей машинного обучения

Цель лабораторной работы: изучение ансамблей моделей машинного обучения.

Выполнил: Ханмурзин Тагир ИУ5-64

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 1) одну из линейных моделей, 2) SVM и 3) дерево решений. Оцените качество моделей с помощью трех подходящих для задачи метрик. Сравните качество полученных моделей.
5. Произведите для каждой модели подбор одного гиперпараметра с использованием GridSearchCV и кросс-валидации.
6. Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

In [3]:
```
import numpy as np
import pandas as pd
from scipy import stats
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import *
from sklearn.metrics import *
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import BaggingRegressor, RandomForestRegressor

%matplotlib inline

import warnings
```

In [4]:
```
warnings.filterwarnings('ignore') # Отключаем предупреждения
```

In [5]:
```
df = load_iris()
df = pd.DataFrame(data = np.c_[df['data'], df['target']], columns = df['feature_names'] + ['target'])
df.head()
```

Out[5]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

In [7]:
```
df.loc[:, df.columns!='target'] = df.loc[:, df.columns!='target'].apply(lambda x: x/x.max(), axis=0)
x_train, x_test, y_train, y_test = train_test_split(df.loc[:, df.columns!='target'],
                                                    df['target'],
                                                    test_size= 0.33)
```

In [9]:
```
def statistics(test_Y, target):
    print("Средняя абсолютная ошибка:", mean_absolute_error(test_Y, target))   # Средняя абсолютная ошибка
    print("Средняя квадратичная ошибка:", mean_squared_error(test_Y, target))    # Средняя квадратичная ошибка
    print("Медианная абсолютная ошибка:", median_absolute_error(test_Y, target)) # Медианная абсолютная ошибка
```

In [10]:
```
base = LinearRegression()
BR = BaggingRegressor(base_estimator = base)
BR.fit(x_train, y_train)

statistics(BR.predict(x_test), y_test)
```

```
Средняя абсолютная ошибка: 0.1880277790373699
Средняя квадратичная ошибка: 0.060225661096369947
Медианная абсолютная ошибка: 0.1662247532435397
```

In [11]:
```
RFR = RandomForestRegressor(max_depth=3, random_state=0,
                            n_estimators=100)
RFR.fit(x_train, y_train)

statistics(RFR.predict(x_test), y_test)
```

```
Средняя абсолютная ошибка: 0.05848531854497004
Средняя квадратичная ошибка: 0.032420871679692743
Медианная абсолютная ошибка: 0.0047994077994077955
```

In [12]:
```
base = LinearRegression()
BR = BaggingRegressor(base_estimator = base)
BR_GV = GridSearchCV(BR, {'n_jobs':range(1,10)}, cv=3).fit(x_train, y_train).best_estimator_

statistics(BR_GV.predict(x_test), y_test)
```

```
Средняя абсолютная ошибка: 0.19000054190673615
Средняя квадратичная ошибка: 0.061001174521747116
Медианная абсолютная ошибка: 0.16024763196024516
```

In [13]:
```
RFR = RandomForestRegressor(random_state=0,
                            n_estimators=100)
RFR_GV = GridSearchCV(RFR, {'max_depth':range(1,10)}, cv=3).fit(x_train, y_train).best_estimator_

statistics(RFR_GV.predict(x_test), y_test)
```

```
Средняя абсолютная ошибка: 0.05680000000000002
Средняя квадратичная ошибка: 0.032568
Медианная абсолютная ошибка: 0.0
```