Лабораторная работа №5. Линейные модели, SVM и деревья решений.

Водка Игорь, ИУ5-61

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

In [101]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [102]:

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split

data = pd.DataFrame(load_diabetes().data)
target = pd.DataFrame(load_diabetes().target)
data.columns = load_diabetes().feature_names
data.head()
```

Out[102]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068330	-0.092204
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002864	-0.025930
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022692	-0.009362
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031991	-0.046641

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

In [103]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [104]:

```
sc1 = MinMaxScaler()
for i in data.columns:
    data[i] = sc1_data = sc1.fit_transform(pd.DataFrame(data[i]))
data.head()
```

Out[104]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.666667	1.0	0.582645	0.549296	0.294118	0.256972	0.207792	0.282087	0.562217	0.439394
1	0.483333	0.0	0.148760	0.352113	0.421569	0.306773	0.623377	0.141044	0.222443	0.166667
2	0.883333	1.0	0.516529	0.436620	0.289216	0.258964	0.246753	0.282087	0.496584	0.409091
3	0.083333	0.0	0.301653	0.309859	0.495098	0.447211	0.233766	0.423131	0.572936	0.469697
4	0.516667	0.0	0.206612	0.549296	0.465686	0.417331	0.389610	0.282087	0.362369	0.333333

С использованием метода train_test_split разделите выборку на обучающую и тестовую.

In [105]:
```
train_data, test_data, train_target, test_target = train_test_split(data, target, random
_state=42)
train_data.head()
```

Out[105]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
16	0.466667	0.0	0.508264	0.661972	0.539216	0.291833	0.623377	0.141044	0.686869	0.606061
408	0.783333	0.0	0.152893	0.901408	0.563725	0.429283	0.298701	0.382228	0.709045	0.651515
432	0.533333	0.0	0.557851	0.436620	0.656863	0.509960	0.350649	0.380818	0.699975	0.893939
316	0.566667	1.0	0.400826	0.464789	0.455882	0.299801	0.246753	0.423131	0.774234	0.651515
3	0.083333	0.0	0.301653	0.309859	0.495098	0.447211	0.233766	0.423131	0.572936	0.469697

In [106]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(train_data, train_target)
```

In [107]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_er
ror, median_absolute_error, r2_score
```

In [108]:
```
def metrics(data, target):
    print("Mean absolute error:", mean_absolute_error(data, target))
    print("Mean squared error:", mean_squared_error(data, target))
    print("Median absolute error:", median_absolute_error(data, target))

metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 41.548363283252066
Mean squared error: 2848.295307932943
Median absolute error: 35.207936652961706
```

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.

In [109]:
```
from sklearn.svm import SVR
reg = SVR(gamma='auto').fit(train_data, train_target)
```

```
/home/igor-vodka/.local/lib/python3.6/site-packages/sklearn/utils/validation.py:761: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [110]:
```
metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 62.19052723285024
Mean squared error: 5247.927904086744
Median absolute error: 57.78011108207377
```

In [111]:
```
from sklearn.tree import DecisionTreeRegressor
reg = DecisionTreeRegressor(max_depth=2)
reg.fit(train_data, train_target)
```

Out[111]:
```
DecisionTreeRegressor(criterion='mse', max_depth=2, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [112]:
```
metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 47.902763313772496
Mean squared error: 3649.4090253107397
Median absolute error: 39.9816513761468
```

Произведите для каждой модели подбор одного гиперпараметра с использованием GridSearchCV и кросс-валидации.

In [113]:

```
from sklearn.model_selection import GridSearchCV
reg = LinearRegression()
param = {'n_jobs':range(10)}
GV = GridSearchCV(reg, param, cv=3)
GV.fit(train_data, train_target)
GV.best_estimator_
```

Out[113]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=0, normalize=False)
```

In [114]:

```
metrics(GV.predict(test_data), test_target)
```

```
Mean absolute error: 41.548363283252066
Mean squared error: 2848.295307932943
Median absolute error: 35.207936652961706
```

In [115]:

```
reg = SVR(gamma='auto')
param = {'degree':range(1,10)}
GV = GridSearchCV(reg, param, cv=3)
GV.fit(train_data, train_target[0])
```

Out[115]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='aut
o',
  kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'degree': range(1, 10)}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score='warn', scoring=None, verbose=0)
```

In [116]:

```
GV.best_estimator_
```

Out[116]:

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=1, epsilon=0.1, gamma='auto',
  kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [117]:

```
metrics(GV.predict(test_data), test_target)
```

```
Mean absolute error: 62.19052723285024
Mean squared error: 5247.927904086744
Median absolute error: 57.78011108207377
```

In [118]:

```
reg = DecisionTreeRegressor( )
param = {'max_depth':range(1,10)}
GV = GridSearchCV(reg, param, cv=3)
GV.fit(train_data, train_target)
```

```
/home/igor-vodka/.local/lib/python3.6/site-packages/sklearn/model_selection/_search.py:84
1: DeprecationWarning: The default of the `iid` parameter will change from True to False
in version 0.22 and will be removed in 0.24. This will change numeric results when test-s
et sizes are unequal.
  DeprecationWarning)
```

Out[118]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=Non
e,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best'),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'max_depth': range(1, 10)}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score='warn', scoring=None, verbose=0)
```

In [119]:

```
GV.best_estimator_
```

Out[119]:

```
DecisionTreeRegressor(criterion='mse', max_depth=2, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [120]:

```
metrics(GV.predict(test_data), test_target)
```

```
Mean absolute error: 47.902763313772496
Mean squared error: 3649.4090253107397
Median absolute error: 39.9816513761468
```

```
In [121]: reg = LinearRegression().fit(train_data, train_target)
          reg.fit(train_data, train_target)
          metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 41.548363283252066
Mean squared error: 2848.295307932943
Median absolute error: 35.207936652961706
```

```
In [122]: reg = SVR(degree=1, gamma='auto').fit(train_data, train_target)
          reg.fit(train_data, train_target)
          metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 62.19052723285024
Mean squared error: 5247.927904086744
Median absolute error: 57.78011108207377

/home/igor-vodka/.local/lib/python3.6/site-packages/sklearn/utils/validation.py:761: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/home/igor-vodka/.local/lib/python3.6/site-packages/sklearn/utils/validation.py:761: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
In [123]: reg = DecisionTreeRegressor(max_depth=3)
          reg.fit(train_data, train_target)
          mean_absolute_error(reg.predict(test_data), test_target)
```

```
Out[123]: 47.34495703928109
```

Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

Ну, стало неплохо!

```
In [ ]:
```