Лабораторная работа №6

Водка Игорь, ИУ5-61

Ансамбли моделей машинного обучения.

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

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

С использованием метода train_test_split разделите выборку на обучающую и тестовую.

Ладно, давайте вернёмся к нашему любимому датасету. ../lab3/winemag-data-130k-v2.csv

In [82]:
```
# read the data
import pandas as pd
reviews = pd.read_csv("../lab3/winemag-data-130k-v2.csv", index_col=0)
pd.set_option('max_rows', 5)

for column in ["country", "region_1", "region_2"]:
    reviews[column] = reviews[column].fillna("Unknown")

reviews['price'] = reviews.groupby('country').transform(lambda x: x.fillna(x.mean()))
```

In [83]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

# country ok, winery ok
for feature in ['country', 'province', 'region_1', 'region_2', 'variety', 'winery']:
    le = LabelEncoder()
    reviews[feature] = reviews[feature].dropna()
    processed = pd.DataFrame({'result': reviews[feature]})
    reviews[feature] = le.fit_transform(processed['result'].astype(str))
```

In [84]:
```
reviews
```

Out[84]:

	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle
0	22	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	87.0	331	424	15	Kerin O'Keefe	@kerinokeefe
1	31	This is ripe and fruity, a wine that is smooth...	Avidagos	87	87.0	108	1094	15	Roger Voss	@vossroger
...	...	...	...	...	...	...	...	...	...	...
129969	15	A dry style of Pinot Gris, this is crisp with ...	NaN	90	90.0	11	21	15	Roger Voss	@vossroger
129970	15	Big, rich and off-dry, this is powered by inte...	Lieu-dit Harth Cuvée Caroline	90	90.0	11	21	15	Roger Voss	@vossroger

129971 rows × 13 columns

In [85]:
```
from sklearn.model_selection import train_test_split

X = reviews[['country', 'price', 'province', 'region_1', 'variety', 'winery']]
y = reviews['points']
train_data, test_data, train_target, test_target = train_test_split(X, y, test_size=0.1,
 random_state=42)
```

Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

In [86]:
```
from sklearn.ensemble import BaggingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_er
ror, median_absolute_error, r2_score
```

In [87]:
```
base = LinearRegression()
reg = BaggingRegressor(base_estimator = base)
reg.fit(train_data, train_target)
```

Out[87]:
```
BaggingRegressor(base_estimator=LinearRegression(copy_X=True, fit_intercept=True, n_jobs=
None,
         normalize=False),
         bootstrap=True, bootstrap_features=False, max_features=1.0,
         max_samples=1.0, n_estimators=10, n_jobs=None, oob_score=False,
         random_state=None, verbose=0, warm_start=False)
```

Возьмём полюбившуюся функцию из прошлой лабораторной работы:

In [88]:
```
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))
```

In [89]:
```
metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 2.276448165395061e-13
Mean squared error: 7.267538121100085e-26
Median absolute error: 2.1316282072803006e-13
```

In [90]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [91]:
```
reg = RandomForestRegressor(max_depth=2, random_state=0, n_estimators=100)
reg.fit(train_data, train_target)
```

Out[91]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=2,
           max_features='auto', 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, n_estimators=100, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [92]:
```
metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 0.7349767393856221
Mean squared error: 0.9189615236579426
Median absolute error: 0.9655417152699357
```

Произведите для каждой модели подбор значений одного гиперпараметра. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.

In [93]:
```
from sklearn.model_selection import GridSearchCV
```

Довольно долго:

In [94]:
```
reg = RandomForestRegressor(random_state=0, n_estimators=30)
param = {'max_depth':range(1,10)}
GV = GridSearchCV(reg, param, cv=3)
GV.fit(train_data, train_target)
```

Out[94]:
```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', 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, n_estimators=30, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False),
       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 [95]:
```
GV.best_estimator_
```

Out[95]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=8,
           max_features='auto', 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, n_estimators=30, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [96]:
```
reg = RandomForestRegressor(max_depth=3, random_state=0, n_estimators=30)
reg.fit(train_data, train_target)

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

```
Mean absolute error: 0.40311575864292687
Mean squared error: 0.2554229628905424
Median absolute error: 0.42376576783307485
```

Качество моделей подросло! Оно стало очень высоким.

In [97]:
```
print("Prediction =", len(prediction))
print("Test target =", len(test_target))
```

```
Prediction = 12998
Test target = 12998
```

In [98]:
```
comparison = pd.DataFrame(
    {
        "predictions": prediction,
        "real target": test_target
    }
)

comparison['diff'] = comparison.apply(lambda row: abs(row['predictions'] - row['real target']), axis=1)
comparison = comparison.sort_values('diff')
```

In [99]:
```
comparison.head()
```

Out[99]:

	predictions	real target	diff
27523	88.0	88	0.0
4253	88.0	88	0.0
17168	89.0	89	0.0
42835	89.0	89	0.0
41600	88.0	88	0.0

In [100]:
```
comparison.tail()
```

Out[100]:

	predictions	real target	diff
116141	94.620707	99	4.379293
114973	94.620707	99	4.379293
60880	94.620707	99	4.379293
7335	94.620707	100	5.379293
123545	94.620707	100	5.379293