jupyter lesson_6 Last Checkpoint: 6 минут назад (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

Задание 1 Импортируйте библиотеки pandas и numpy.

In [4]:
```
import pandas as pd
import numpy as np
```

Загрузите "Boston House Prices dataset" из встроенных наборов данных библиотеки sklearn.

In [5]:
```
from sklearn.datasets import load_boston
```

In [9]:
```
boston=load_boston()
```

In [10]:
```
boston.keys()
```

Out[10]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

In [13]:
```
data=boston.data
```

In [17]:
```
target=boston.target
```

In [18]:
```
feature_names=boston.feature_names
```

Создайте датафреймы X и Y из этих данных.

In [19]:
```
x=pd.DataFrame(data, columns=feature_names)
```

In [20]:
```
y=pd.DataFrame(target, columns=['price'])
```

Разбейте эти датафреймы на тренировочные (X_train, y_train) и тестовые (X_test, y_test) с помощью функции train_test_split так, чтобы размер тестовой выборки составлял 30% от всех данных, при этом аргумент random_state должен быть равен 42.

In [21]:
```
from sklearn.model_selection import train_test_split
```

In [33]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

Создайте модель линейной регрессии под названием lr с помощью класса LinearRegression из модуля sklearn.linear_model. Обучите модель на тренировочных данных (используйте все признаки) и сделайте предсказание на тестовых. Вычислите R2 полученных предсказаний с помощью r2_score из модуля sklearn.metrics.

In [34]:
```
from sklearn.linear_model import LinearRegression
```

In [35]:
```
lr=LinearRegression()
```

In [36]:
```
lr.fit(X_train, y_train)
```

Out[36]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [39]:
```
y_pred=lr.predict(X_test)
```

In [41]:
```
check_test=pd.DataFrame({'y_test': y_test['price'],
                         'y_pred': y_pred.flatten()},
                        columns=['y_test', 'y_pred'])
```

In [43]:
```
check_test.head(10)
```

jupyter lesson_6 Last Checkpoint: 6 минут назад (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [43]:
```
check_test.head(10)
```

Out[43]:

	y_test	y_pred
173	23.6	28.648960
274	32.4	36.495014
491	13.6	15.411193
72	22.8	25.403213
452	16.1	18.855280
76	20.0	23.146689
316	17.8	17.392124
140	14.0	14.078599
471	19.6	23.038927
500	16.8	20.599433

In [44]:
```
from sklearn.metrics import r2_score
```

In [45]:
```
r2_score(y_test, y_pred)
```

Out[45]:
```
0.7112260057484925
```

Задание 2 Создайте модель под названием model с помощью RandomForestRegressor из модуля sklearn.ensemble.

Сделайте агрумент n_estimators равным 1000, max_depth должен быть равен 12 и random_state сделайте равным 42.

Обучите модель на тренировочных данных аналогично тому, как вы обучали модель LinearRegression, но при этом в метод fit вместо датафрейма y_train поставьте y_train.values[:, 0],

чтобы получить из датафрейма одномерный массив Numpy,

так как для класса RandomForestRegressor в данном методе для аргумента y предпочтительно применение массивов вместо датафрейма.

Сделайте предсказание на тестовых данных и посчитайте R2. Сравните с результатом из предыдущего задания.

Напишите в комментариях к коду, какая модель в данном случае работает лучше.

In [46]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [51]:
```
model = RandomForestRegressor(n_estimators = 1000, max_depth = 12, random_state = 42)
```

In [52]:
```
model.fit(X_train, y_train.values[:, 0])
```

Out[52]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=12,
                      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=1000,
                      n_jobs=None, oob_score=False, random_state=42, verbose=0,
                      warm_start=False)
```

In [54]:
```
y_pred2 = model.predict(X_test)
r2_score(y_test, y_pred2)
```

Out[54]:
```
0.8749965273218174
```

Модель линейной регрессии работает хуже, чем использование ансамблей деревьев решений. RandomForestRegressor в данном случае лучше, чем LinearRegression. (0,87 и 0,71 соответственно)

In [54]:
```
y_pred2 = model.predict(X_test)
r2_score(y_test, y_pred2)
```

Out[54]:
```
0.8749965273218174
```

Модель линейной регрессии работает хуже, чем использование ансамблей деревьев решений. RandomForestRegressor в данном случае лучше, чем LinearRegression. (0,87 и 0,71 соответственно)

- Задание 3 Вызовите документацию для класса RandomForestRegressor, найдите информацию об атрибуте feature_importances_. С помощью этого атрибута найдите сумму всех показателей важности, установите, какие два признака показывают наибольшую важность.

In [57]:
```
?RandomForestRegressor
feature_importances = model.feature_importances_
feature_importances
```

Out[57]:
```
array([0.03211748, 0.00154999, 0.0070941 , 0.0011488 , 0.01436832,
       0.40270459, 0.01424477, 0.06403265, 0.00496762, 0.01169177,
       0.01808961, 0.0123114 , 0.41567892])
```

Сумма всех показателей важности:

In [58]:
```
feature_importances.sum()
```

Out[58]:
```
0.9999999999999999
```

Индексы двух лучших показателей:

In [59]:
```
bst_ind = feature_importances.argsort()[-2:]
bst_ind
```

Out[59]:
```
array([ 5, 12], dtype=int64)
```

Два самых важных признака (с их показателями важности):

In [60]:
```
two_best_features = dict()
for i in bst_ind:
    two_best_features[feature_names[i]] = feature_importances[i]
two_best_features
```

Out[60]:
```
{'RM': 0.402704591696731, 'LSTAT': 0.4156789214509943}
```

Сумма показателей важности двух самых выжных признаков:

In [61]:
```
sum(two_best_features.values())
```

Out[61]:
```
0.8183835131477253
```

Важность признаков RM и LSTAT составляет 82% от важности всех признаков.

In [ ]: