

Лабораторная работа №10 “Градиентный бустинг”

Долматович Алина, 858641

```
In [12]: 1 from sklearn.datasets import load_boston
2 import pandas as pd
3 import sklearn
4 import sklearn.cross_validation
5 from sklearn.tree import DecisionTreeRegressor
6 from sklearn.metrics import mean_squared_error
7 import matplotlib.pyplot as pyplot
8 import numpy as np
9 from sklearn.linear_model import LinearRegression
```

Загрузите данные с помощью библиотеки sklearn.

```
In [2]: 1 boston = load_boston()
2 bostonData = pd.DataFrame(boston.data)
3 bostonData.columns = boston.feature_names
4 bostonData['PRICE'] = boston.target
5 print(bostonData.head())
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TA
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.

	PTRATIO	B	LSTAT	PRICE
0	15.3	396.90	4.98	24.0
1	17.8	396.90	9.14	21.6
2	17.8	392.83	4.03	34.7
3	18.7	394.63	2.94	33.4
4	18.7	396.90	5.33	36.2

Разделите выборку на обучающую (75%) и контрольную (25%).

```
In [3]: 1 x = bostonData.drop('PRICE', axis = 1).values
        2 y = bostonData['PRICE'].values
        3 trainX, testX, trainY, testY = sklearn.cross_validation.train_test_
        4 print(trainX.shape, trainY.shape)
        5 print(testX.shape, testY.shape)

((379, 13), (379,))
((127, 13), (127,))
```

В процессе реализации обучения вам потребуется функция, которая будет вычислять прогноз построенной на данный момент композиции деревьев на выборке X. Реализуйте ее. Эта же функция поможет вам получить прогноз на контрольной выборке и оценить качество работы вашего алгоритма с помощью `mean_squared_error` в `sklearn.metrics`.

```
In [4]: 1 def boostingPredict(X, regressors, alphas=[1 for i in range(50)]):
        2     result = [sum([alpha * regressor.predict([x])[0] for regressor
        3     return result
```

Заведите массив для объектов `DecisionTreeRegressor` (они будут использоваться в качестве базовых алгоритмов) и для вещественных чисел (коэффициенты перед базовыми алгоритмами).

В цикле обучите последовательно 50 решающих деревьев с параметрами `max_depth=5` и `random_state=42` (остальные параметры - по умолчанию). Каждое дерево должно обучаться на одном и том же множестве объектов, но ответы, которые учится прогнозировать дерево, будут меняться в соответствии с отклонением истинных значений от предсказанных.

```
In [5]: 1 def boosting(x, y, countOfTree, depth, alphas=[1 for i in range(50)
        2     regressors = []
        3     shift = y.copy()
        4     for i in range(countOfTree):
        5         regressor = DecisionTreeRegressor(max_depth=depth, random_
        6         regressor.fit(x, shift)
        7         regressors.append(regressor)
        8         predict = boostingPredict(x, regressors, alphas)
        9         shift = -(predict-y)
        10    return regressors
        11
        12 def mse(alpha=1, countOfTree=50, depth=5):
        13     alphas=[alpha(i) if callable(alpha) else alpha for i in range(
        14     regressors = boosting(trainX, trainY, countOfTree, depth, alph
        15     return mean_squared_error(y_true=np.array(testY), y_pred=boost
        16
        17 mse()
```

```
Out[5]: 6.1027230908259078
```

Попробуйте всегда брать коэффициент равным 0.9. Обычно оправдано выбирать коэффициент значительно меньшим - порядка 0.05 или 0.1, но на стандартном наборе данных будет всего 50 деревьев, возьмите для начала шаг побольше.

```
In [6]: 1 mse(0.9)
```

```
Out[6]: 5.470486096542726
```

Попробуйте уменьшать вес перед каждым алгоритмом с каждой следующей итерацией по формуле $0.9 / (1.0 + i)$, где i - номер итерации (от 0 до 49). Какое получилось качество на контрольной выборке?

```
In [7]: 1 def mutableAlpha(i):  
2         return 0.9 / (1. + i)  
3  
4 mse(mutableAlpha)
```

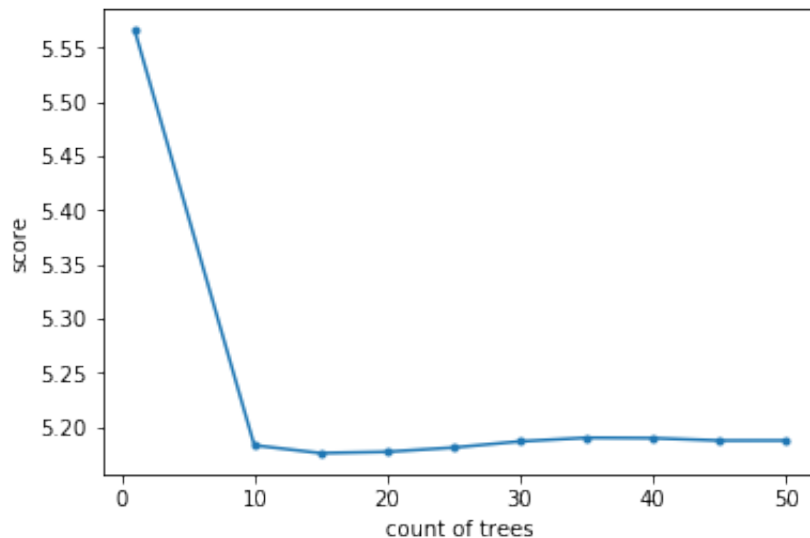
```
Out[7]: 5.187553592288821
```

Исследуйте, переобучается ли градиентный бустинг с ростом числа итераций, а также с ростом глубины деревьев. Постройте графики. Какие выводы можно сделать?

```
In [8]: 1 def plot(parameters, scores, xlabel):  
2         pyplot.plot(parameters, scores, marker='.')  
3         pyplot.xlabel(xlabel)  
4         pyplot.ylabel('score')  
5         pyplot.show()
```

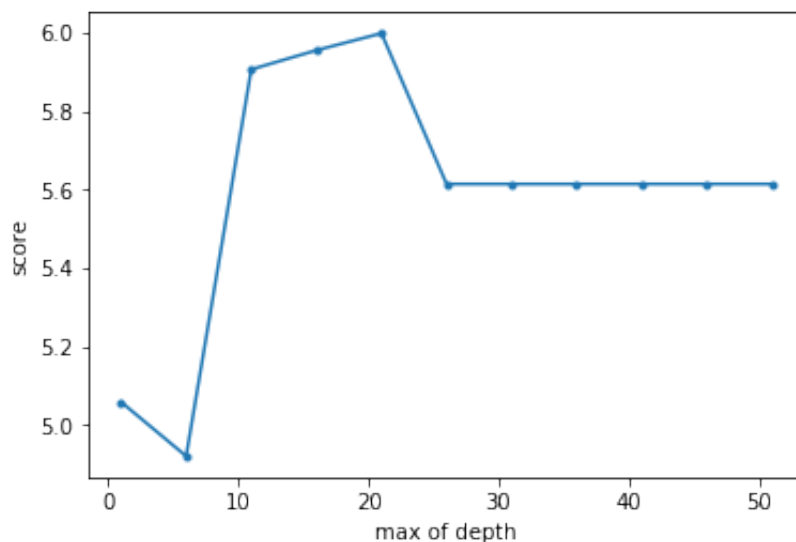
In [9]:

```
1 countOfTrees = [1] + range(10, 55, 5)
2 scores = []
3 for count in countOfTrees:
4     error = mse(mutableAlpha, countOfTree=count, depth=5)
5     scores.append(error)
6 plot(countOfTrees, scores, "count of trees")
```



In [10]:

```
1 depths = range(1, 55, 5)
2 scores = []
3 for depth in depths:
4     error = mse(mutableAlpha, countOfTree=50, depth=depth)
5     scores.append(error)
6 plot(depths, scores, "max of depth")
```



Сравните качество, получаемое с помощью градиентного бустинга с качеством работы линейной регрессии. Для этого обучите `LinearRegression` из `sklearn.linear_model` (с параметрами по умолчанию) на обучающей выборке и оцените для прогнозов полученного алгоритма на тестовой выборке RMSE.

```
In [11]: 1 linearRegression = LinearRegression()  
         2 linearRegression.fit(trainX, trainY)  
         3 mean_squared_error(y_true=testY, y_pred=linearRegression.predict(testX))
```

```
Out[11]: 4.9293108816992275
```