Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра информатики

ЛабоРАТОРНАЯ РАБОТА №10

«Градиентный бустинг»

Выполнил: Яловчук Валерий Валерьевич

магистрант кафедры информатики

группа №858641

Проверил: доцент, кандидат технических наук Стержанов Максим Валерьевич

Минск 2019

ХОД РАБОТЫ

**Данные.**

Для выполнения задания используйте набор данных boston из библиотеки sklearn: https://scikit-learn.org/stable/datasets/index.html#boston-dataset

**Выполнение:**

1. Загрузите данные с помощью библиотеки sklearn:

boston = datasets.load\_boston()

X, Y = boston.data, boston.target

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.25, random\_state=51)

3-6. Заведите массив для объектов DecisionTreeRegressor (они будут использоваться в качестве базовых алгоритмов) и для вещественных чисел (коэффициенты перед базовыми алгоритмами). 4. 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 соответствие с отклонением истинных значений от предсказанных. 5. 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. 6. 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:

mocked\_coefficient\_quality = check\_quality(X\_train, y\_train, X\_test, y\_test, mock\_coefficient)()

print(mocked\_coefficient\_quality)

Результат выполнения:

5.569226762770745

7. 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?

sequences\_coefficient\_quality = check\_quality(X\_train, y\_train, X\_test, y\_test, calculate\_coefficient)()

print(sequences\_coefficient\_quality)

Результат выполнения:

5.3607371501561625

Ошибка немного уменьшилась.

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

trees\_number = [50, 100, 150, 200, 250, 300, 400, 500]

params = []

for i in trees\_number:

params.append(DecisionTreeParams(X\_train, y\_train, X\_test, y\_test, calculate\_coefficient, i, 5))

pool = Pool(len(trees\_number))

errors = pool.map(job\_gb, params)

depth = [2, 4, 6, 8, 10, 13, 17, 20]

params = []

for i in depth:

params.append(DecisionTreeParams(X\_train, y\_train, X\_test, y\_test, calculate\_coefficient, 50, i))

errors = pool.map(job\_gb, params)

Результат выполнения:

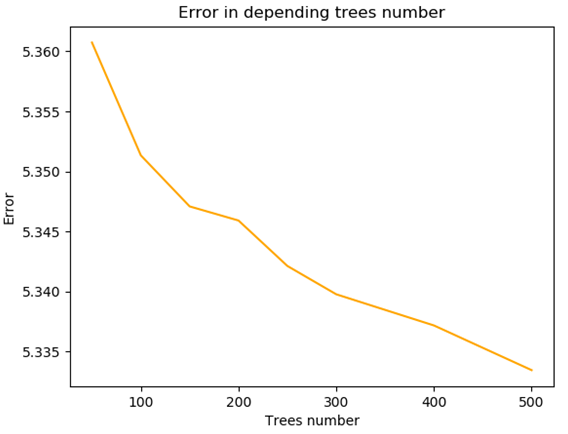


Рисунок 1 – график зависимости ошибки от количества деревьев (не библиотечная реализация)

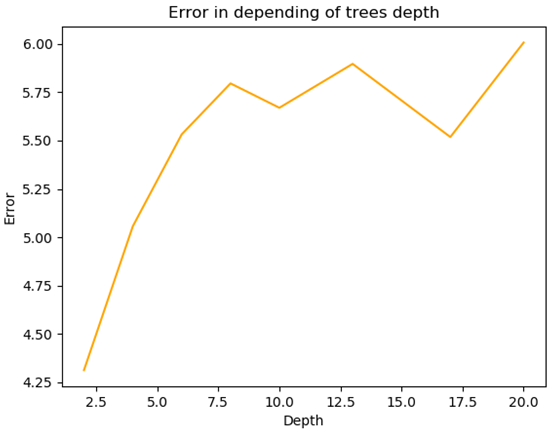


Рисунок 2 – Зависимость ошибки от глубины дерева (не библиотчечная реализация)

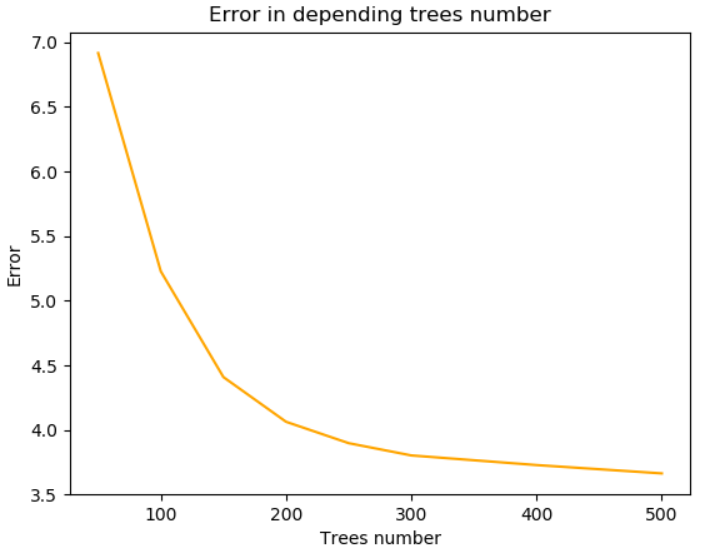


Рисунок 3 – зависимость ошибки от количества деревьев (библиотечная реализация)

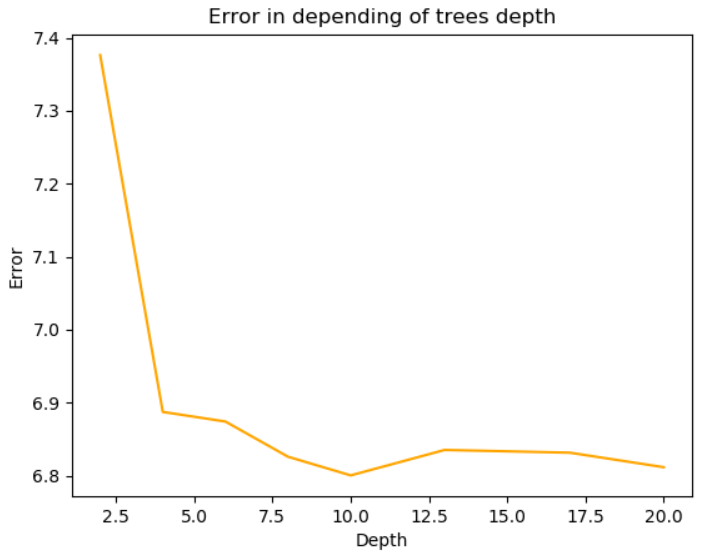


Рисунок 4 – зависимость ошибки од глубины дерева (библиотечная реализация)

С ростом кол-ва деревьев cost function снижается. Увеличивается сложность вычислений. С ростом depth деревьев решений cost function падает до определённого момента, а потом начинается overfitting.

11. 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:

reg = LinearRegression().fit(X\_train, y\_train)

pred = reg.predict(X\_test)

print('Linear regression MSE: ', np.sqrt(mean\_squared\_error(y\_test, pred)))

Результат выполнения:

('Linear regression MSE: ', 4.863502400380662)

Ошибка в linear regression меньше, чем в gradient boosting. Однако, из графиков выше видно, при увеличении моделей в gradient boosting и увеличении depth decision trees до определённого момента, можно получить лучшее значение error, чем в linear regression.