|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт кибербезопасности и цифровых технологий |
| Кафедра КБ-2 «Прикладные информационные технологии» |

**ПРАКТИЧЕСКАЯ РАБОТА №1 по дисциплине**

**«Методы анализа данных»**

***(наименование дисциплины)***

**Тема практической работы   
«Предсказания победителя в онлайн-игре»**

Студент группы: БИСО-01-20 Митрофанов М.С.

*(шифр учебной группы) (Фамилия И.О)*

Преподаватель: Лесько С.А.

*(Фамилия И.О)*

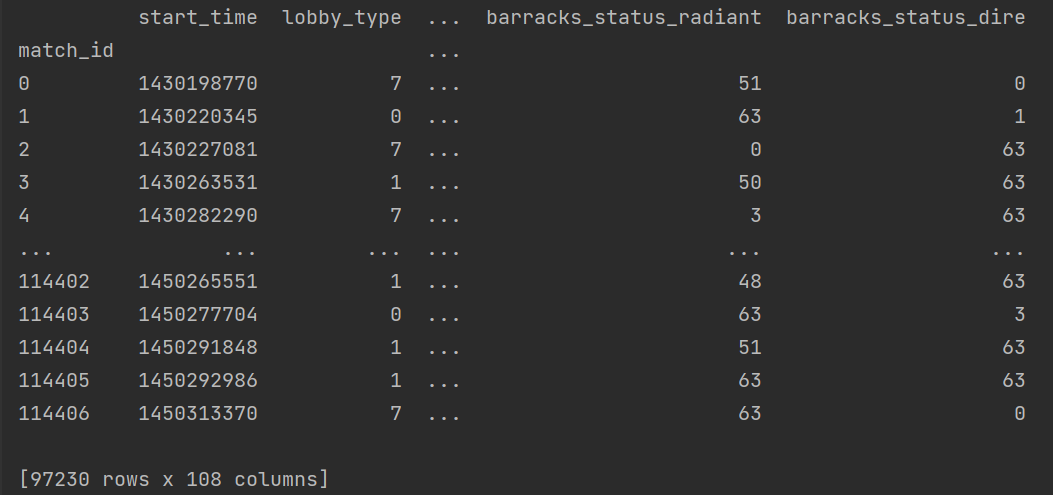
Москва, 2023

**Подход 1: градиентный бустинг "в лоб"**

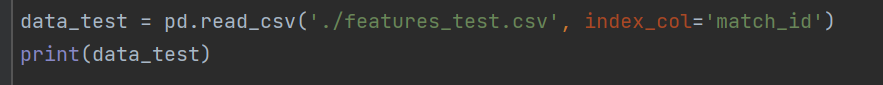
Один из самых универсальных алгоритмов, изученных в нашем курсе, является градиентный бустинг. Он не очень требователен к данным, восстанавливает нелинейные зависимости, и хорошо работает на многих наборах данных, что и обуславливает его популярность. Вполне разумной мыслью будет попробовать именно его в первую очередь.

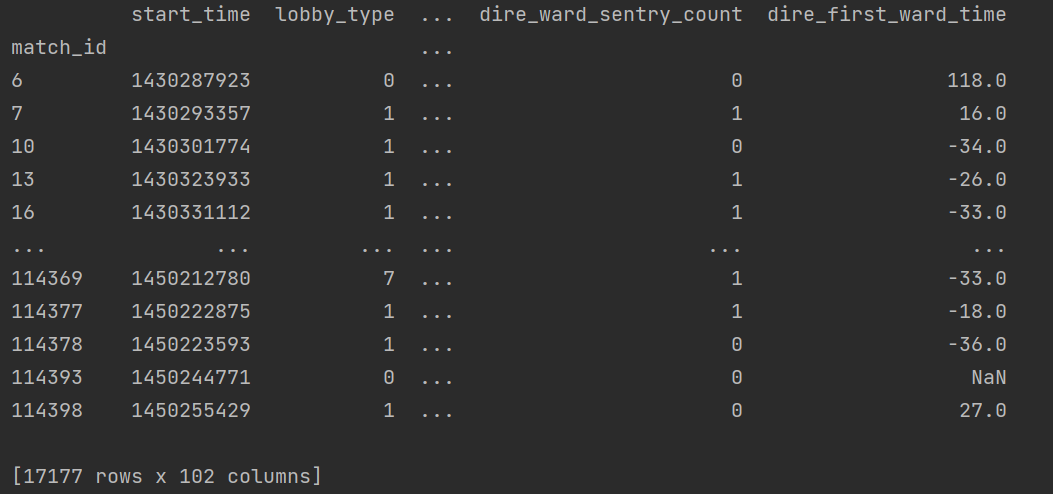
1. Считайте таблицу с признаками из файла features.csv



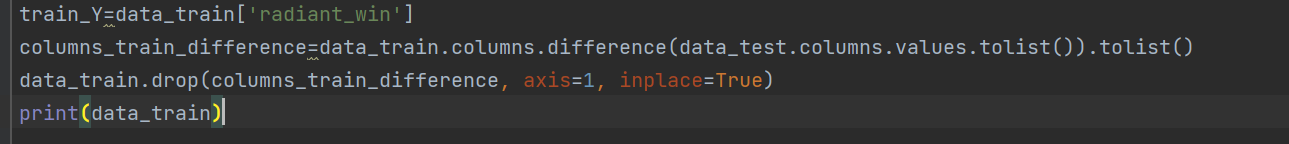


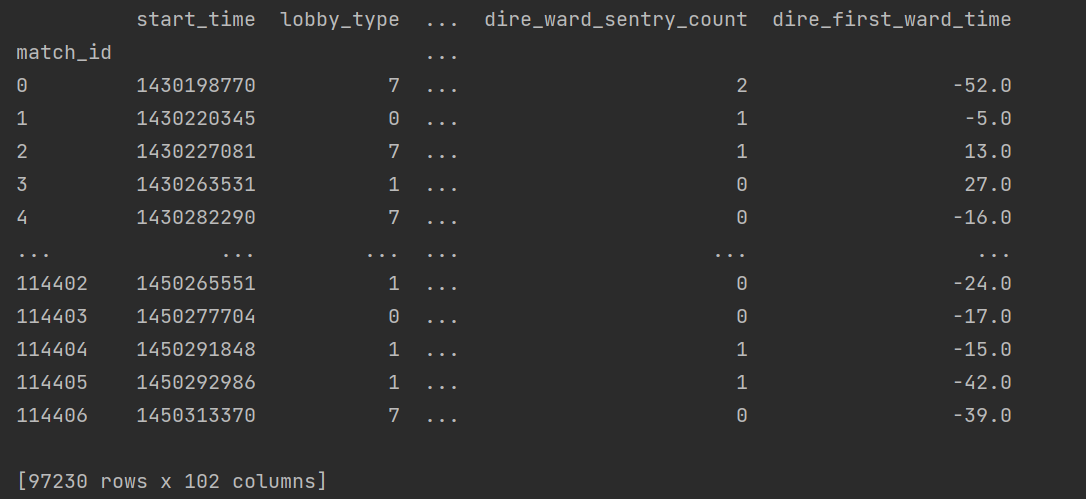
Считываем файл features\_test с тестовой выборкой



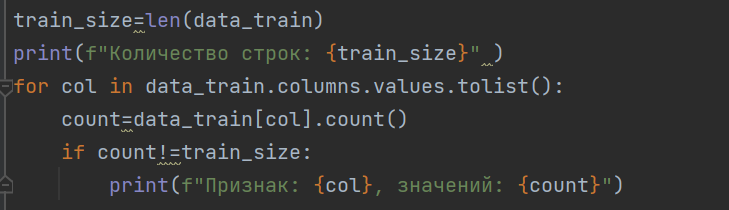


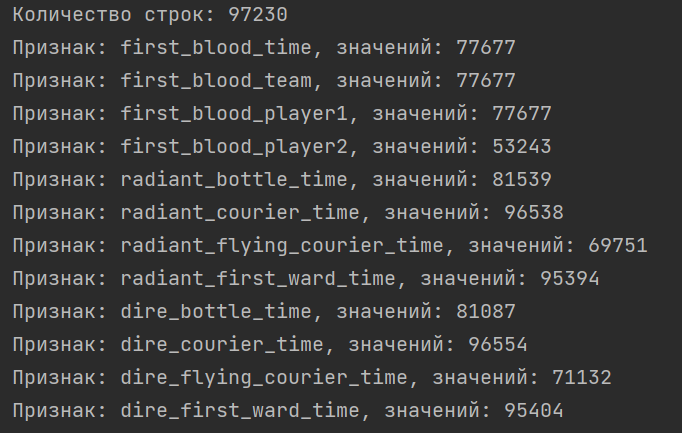
Удалите признаки, связанные с итогами матча (они помечены в описании данных как отсутствующие в тестовой выборке).



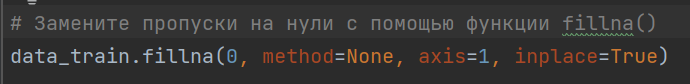


1. Проверьте выборку на наличие пропусков с помощью функции count(), которая для каждого столбца показывает число заполненных значений.





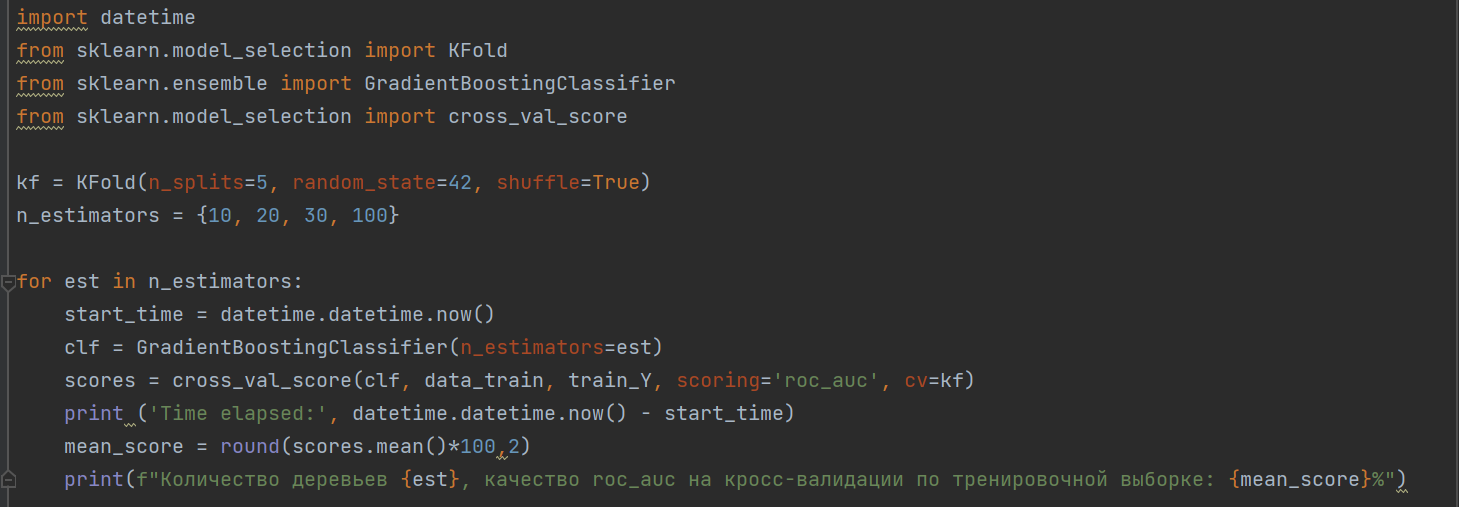
1. Замените пропуски на нули с помощью функции fillna(). На самом деле этот способ является предпочтительным для логистической регрессии, поскольку он позволит пропущенному значению не вносить никакого вклада в предсказание. Для деревьев часто лучшим вариантом оказывается замена пропуска на очень большое или очень маленькое значение — в этом случае при построении разбиения вершины можно будет отправить объекты с пропусками в отдельную ветвь дерева. Также есть и другие подходы — например, замена пропуска на среднее значение признака. Мы не требуем этого в задании, но при желании попробуйте разные подходы к обработке пропусков и сравните их между собой.

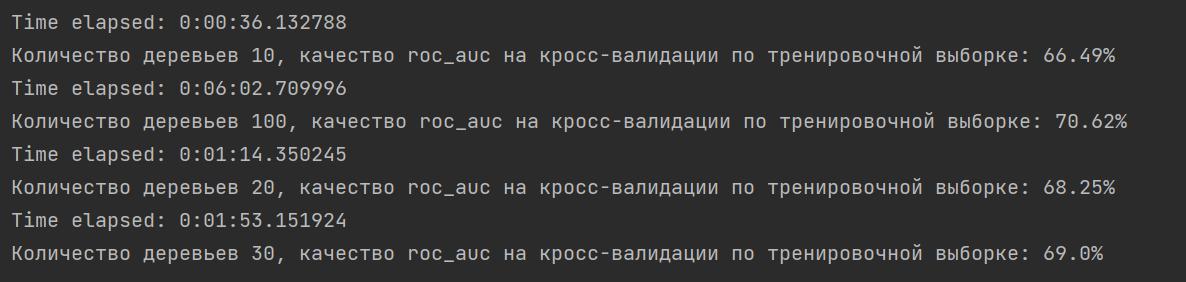


1. Какой столбец содержит целевую переменную? Запишите его название.

Целевая переменная 'radiant\_win' – 1, если победила команда Radiant, иначе – 0.

1. Забудем, что в выборке есть категориальные признаки, и попробуем обучить градиентный бустинг над деревьями на имеющейся матрице "объекты-признаки". Зафиксируйте генератор разбиений для кросс-валидации по 5 блокам (KFold), не забудьте перемешать при этом выборку (shuffle=True), поскольку данные в таблице отсортированы по времени, и без перемешивания можно столкнуться с нежелательными эффектами при оценивании качества. Оцените качество градиентного бустинга (GradientBoostingClassifier) с помощью данной кросс-валидации, попробуйте при этом разное количество деревьев (как минимум протестируйте следующие значения для количества деревьев: 10, 20, 30). Долго ли настраивались классификаторы? Достигнут ли оптимум на испытанных значениях параметра n\_estimators, или же качество, скорее всего, продолжит расти при дальнейшем его увеличении?





1) Какие признаки имеют пропуски среди своих значений? Что могут означать пропуски в этих признаках (ответьте на этот вопрос для двух любых признаков)?

Пропуски у признаков:

first\_blood\_time: игровое время первой крови

first\_blood\_team: команда, совершившая первую кровь (0 — Radiant, 1 — Dire)

first\_blood\_player1: игрок, причастный к событию

first\_blood\_player2: второй игрок, причастный к событию

radiant\_bottle\_time: время первого приобретения командой предмета "bottle"

radiant\_courier\_time: время приобретения предмета "courier"

radiant\_flying\_courier\_time: время приобретения предмета "flying\_courier

radiant\_first\_ward\_time: время установки командой первого "наблюдателя", т.е. предмета, который позволяет видеть часть игрового поля

dire\_bottle\_time: время первого приобретения командой предмета "bottle"

dire\_courier\_time: время приобретения предмета "courier"

dire\_flying\_courier\_time: время приобретения предмета "flying\_courier

dire\_first\_ward\_time: время установки командой первого "наблюдателя", т.е. предмета, который позволяет видеть часть игрового поля

Отсутствие остальных данных, скорее всего, говорит от том, что эти события не произошли в течение первых 5 минут игры.

2) Как называется столбец, содержащий целевую переменную?

Столбец, содержащий целевую переменную – 'radiant\_win'

3) Как долго проводилась кросс-валидация для градиентного бустинга с 30 деревьями? Инструкцию по измерению времени можно найти ниже по тексту. Какое качество при этом получилось? Напомним, что в данном задании мы используем метрику качества AUC-ROC.

Почти 2 минуты, качество – 69%

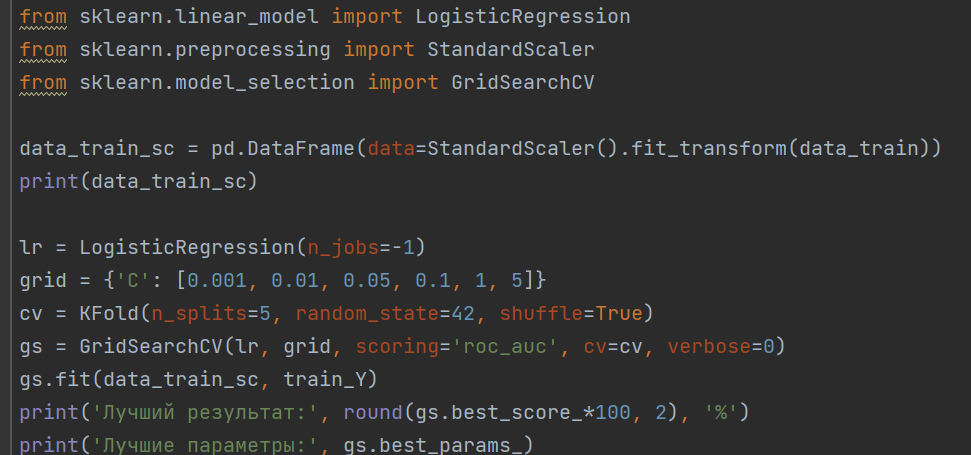
4) Имеет ли смысл использовать больше 30 деревьев в градиентном бустинге? Что бы вы предложили делать, чтобы ускорить его обучение при увеличении количества деревьев?

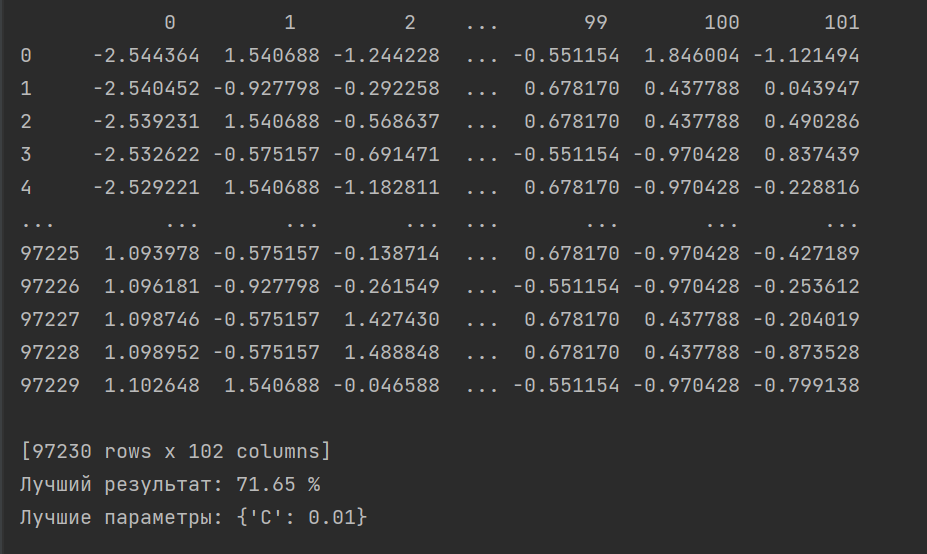
Увеличивать количество деревьев больше 30-ти стоит при наличии достаточного времени или вычислительных мощностей. В случае 100 деревьев качество увеличивается меньше, чем на 2%, а время вычислений становится больше примерно в 3 раза. Для ускорения процесса при увеличении числа деревьев следует уменьшать глубину деревьев и/или использовать только часть признаков.

**Подход 2: логистическая регрессия**

Линейные методы работают гораздо быстрее композиций деревьев, поэтому кажется разумным воспользоваться именно ими для ускорения анализа данных. Одним из наиболее распространенных методов для классификации является логистическая регрессия.

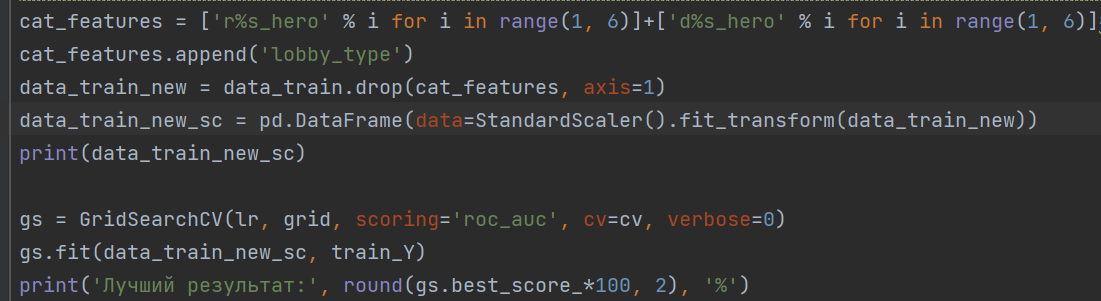
1. Оцените качество логистической регрессии (sklearn.linear\_model.LogisticRegression с L2-регуляризацией) с помощью кросс-валидации по той же схеме, которая использовалась для градиентного бустинга. Подберите при этом лучший параметр регуляризации (C). Какое наилучшее качество у вас получилось? Как оно соотносится с качеством градиентного бустинга? Чем вы можете объяснить эту разницу? Быстрее ли работает логистическая регрессия по сравнению с градиентным бустингом?

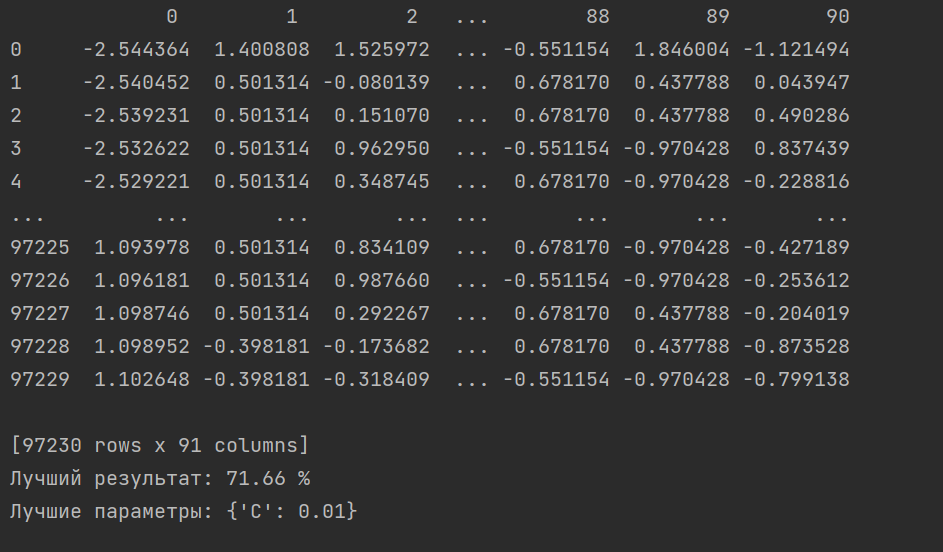




Для логистической регрессии был выбран более подходящий гиперпараметр, что позволило ей достичь лучшего качества на данном наборе данных. Логистическая регрессия работает быстрее, чем градиентный бустинг, потому что она является линейной моделью и не требует построения большого количества деревьев.

1. Среди признаков в выборке есть категориальные, которые мы использовали как числовые, что вряд ли является хорошей идеей. Категориальных признаков в этой задаче одиннадцать: lobby\_type и r1\_hero, r2\_hero, ..., r5\_hero, d1\_hero, d2\_hero, ..., d5\_hero. Уберите их из выборки, и проведите кросс-валидацию для логистической регрессии на новой выборке с подбором лучшего параметра регуляризации. Изменилось ли качество? Чем вы можете это объяснить?



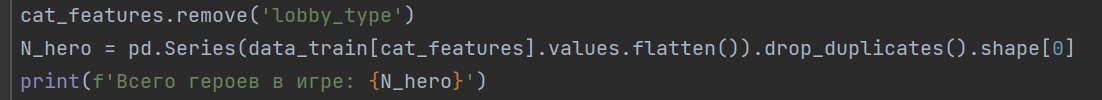


Убирая категориальные признаки, мы теряем информацию о конкретных героях, но при этом можем получить лучшие результаты, так как числовые представления этих категорий не содержат смысла. Номер героя не имеет никакого отношения к их характеристикам.

Это может улучшить работу модели, потому что логистическая регрессия будет использовать более информативные признаки для принятия решения.

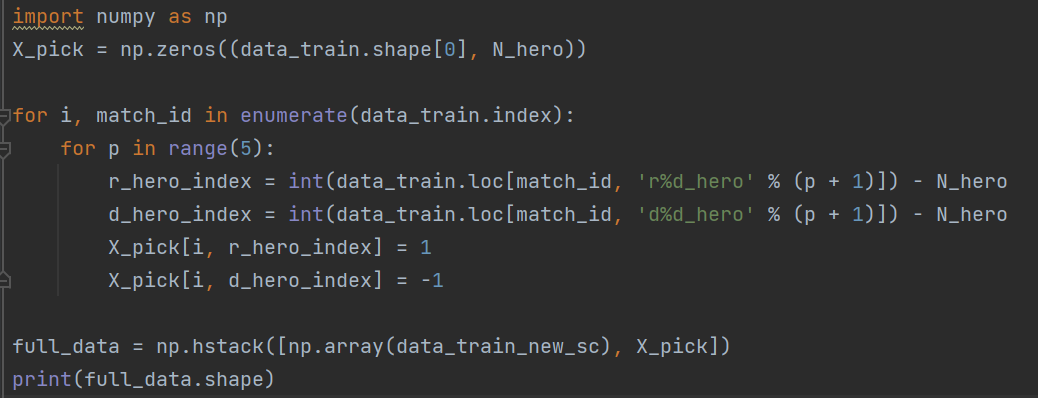
Мы получили результат чуть лучше, чем в предыдущем случае

1. На предыдущем шаге мы исключили из выборки признаки rM\_hero и dM\_hero, которые показывают, какие именно герои играли за каждую команду. Это важные признаки — герои имеют разные характеристики, и некоторые из них выигрывают чаще, чем другие. Выясните из данных, сколько различных идентификаторов героев существует в данной игре (вам может пригодиться фукнция unique или value\_counts).



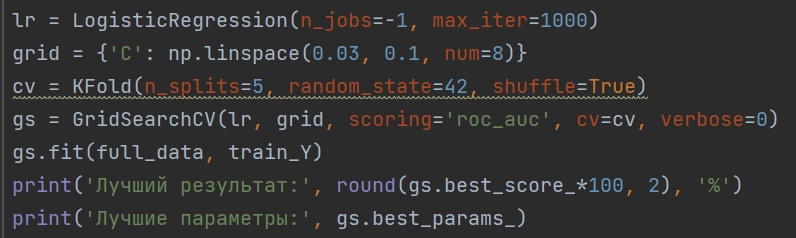


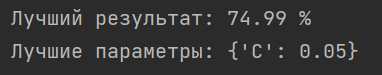
1. Воспользуемся подходом "мешок слов" для кодирования информации о героях. Пусть всего в игре имеет N различных героев. Сформируем N признаков, при этом i-й будет равен нулю, если i-й герой не участвовал в матче; единице, если i-й герой играл за команду Radiant; минус единице, если i-й герой играл за команду Dire. Добавьте полученные признаки к числовым, которые вы использовали во втором пункте данного этапа.





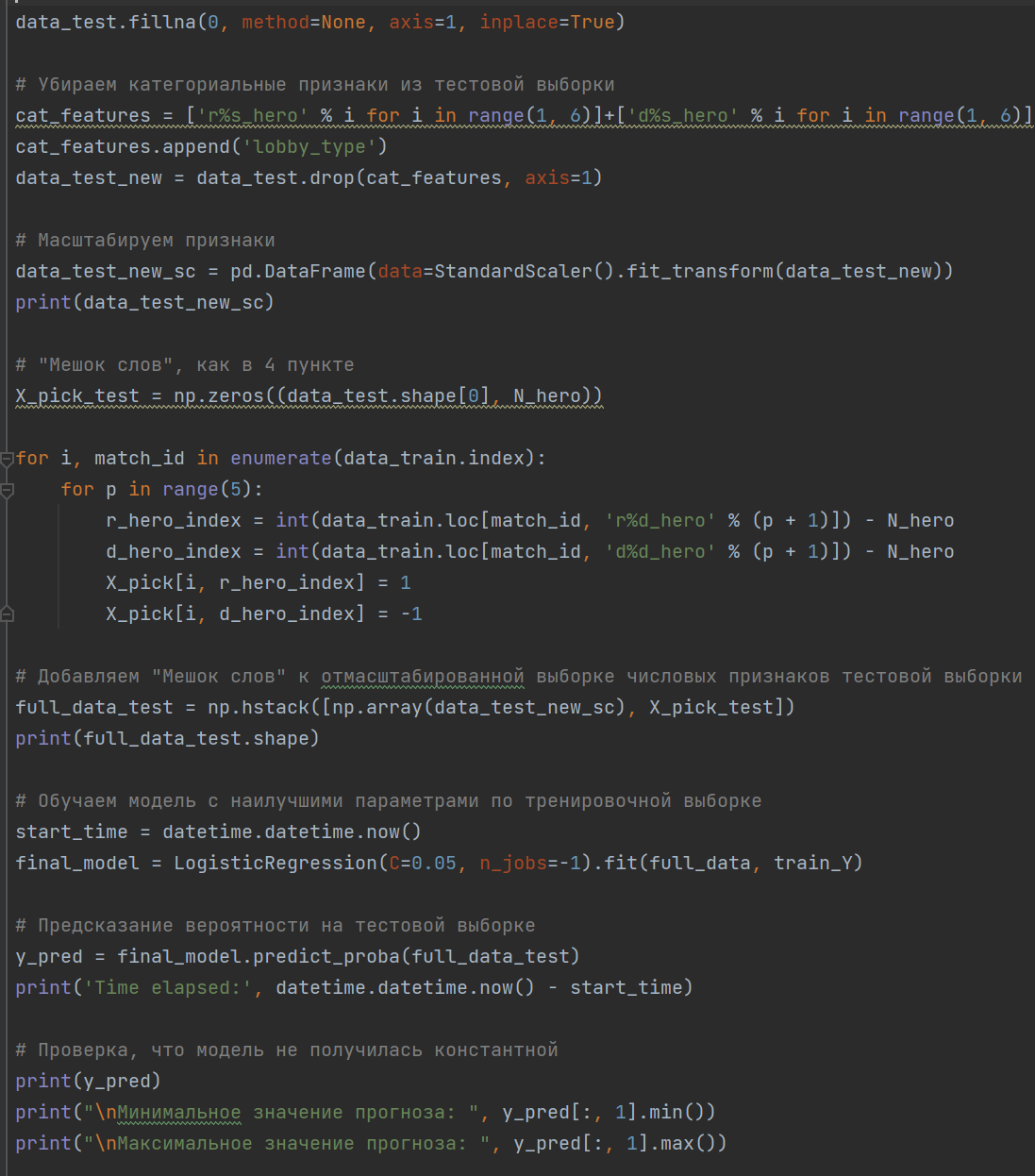
1. Проведите кросс-валидацию для логистической регрессии на новой выборке с подбором лучшего параметра регуляризации. Какое получилось качество? Улучшилось ли оно? Чем вы можете это объяснить?

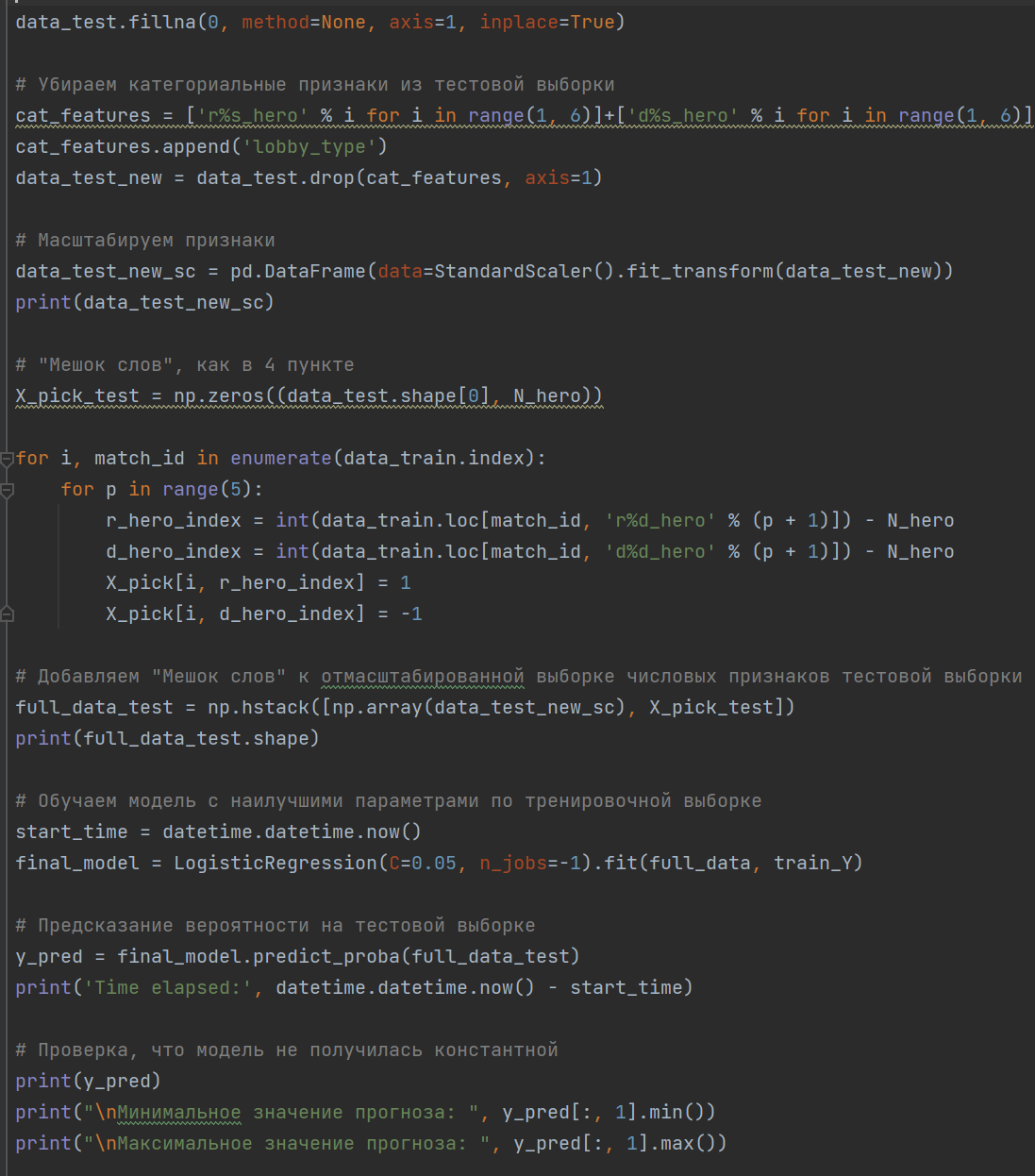


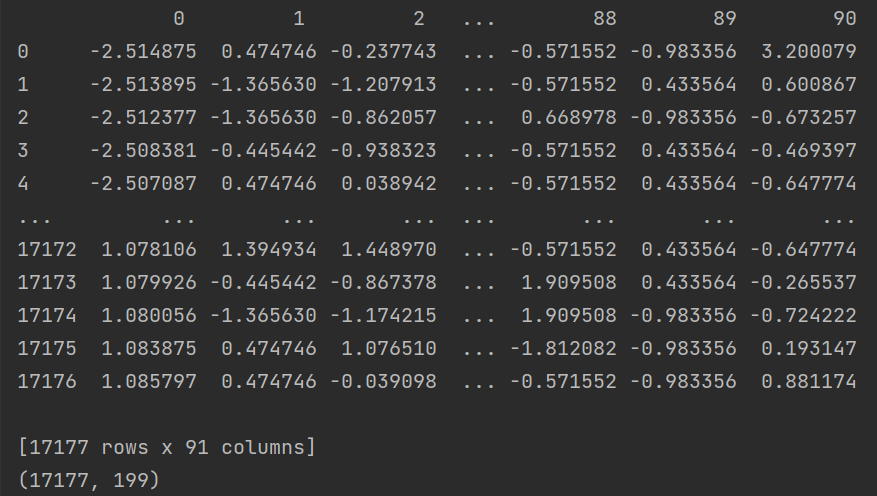


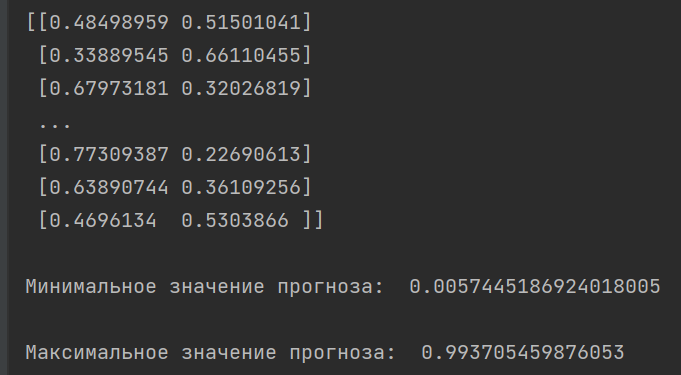
Качество улучшилось, это связано с тем, что удаление категориальных признаков и масштабирование данных помогли модели лучше обучиться.

1. Постройте предсказания вероятностей победы команды Radiant для тестовой выборки с помощью лучшей из изученных моделей (лучшей с точки зрения AUC-ROC на кросс-валидации). Убедитесь, что предсказанные вероятности адекватные — находятся на отрезке [0, 1], не совпадают между собой (т.е. что модель не получилась константной).









1) Какое качество получилось у логистической регрессии над всеми исходными признаками? Как оно соотносится с качеством градиентного бустинга? Чем вы можете объяснить эту разницу? Быстрее ли работает логистическая регрессия по сравнению с градиентным бустингом?

Логистическая регрессия, примененная к исходным признакам без дополнительной обработки, показала качество на кросс-валидации в 71.65%, что превосходит результат градиентного бустинга в 70.62%. Эту разницу можно объяснить особенностями исходных данных.

Логистическая регрессия работает быстрее, чем градиентный бустинг, потому что она является линейной моделью и не требует построения большого количества деревьев.

2) Как влияет на качество логистической регрессии удаление категориальных признаков (укажите новое значение метрики качества)? Чем вы можете объяснить это изменение?

Убирая категориальные признаки, мы теряем информацию о конкретных героях, но при этом можем получить лучшие результаты, так как числовые представления этих категорий не содержат смысла. Номер героя не имеет никакого отношения к их характеристикам.

Это может улучшить работу модели, потому что логистическая регрессия будет использовать более информативные признаки для принятия решения.

Мы получили результат чуть лучше, чем в предыдущем случае - на 0,01%

3) Сколько различных идентификаторов героев существует в данной игре?

108 различных героев.

4) Какое получилось качество при добавлении "мешка слов" по героям? Улучшилось ли оно по сравнению с предыдущим вариантом? Чем вы можете это объяснить?

При добавлении "мешка слов" по героям, качество улучшается и составляет 74.99%. Это связано с переводом категориальных признаков в числовые на работу с которыми рассчитана линейная регрессия.

5) Какое минимальное и максимальное значение прогноза на тестовой выборке получилось у лучшего из алгоритмов?

Минимальное значение прогноза - 0.005744518567402761

Максимальное значение прогноза - 0.9937054595749831