Отчет по финальному проекту

**Градиентный бустинг в лоб**

1)Следующие признаки имеют пропуски среди своих значений:

*['first\_blood\_time', 'first\_blood\_team', 'first\_blood\_player1', 'first\_blood\_player2', 'radiant\_bottle\_time', 'radiant\_courier\_time', 'radiant\_flying\_courier\_time', 'radiant\_first\_ward\_time', 'dire\_bottle\_time', 'dire\_courier\_time', 'dire\_flying\_courier\_time', 'dire\_first\_ward\_time']*

Например, если взять ‘first\_blood\_time’, то там, где значение было NaN, «первая кровь» еще не случилась в первые пять минут. Аналогично, для ‘dire\_courier\_time’, если значение стоит NaN, то покупка предмета 'courier' еще не случилась за прошедшие с начала игры пять минут.

2)Целевой переменной является столбец ‘radiant\_win’. Единица в столбце означает победу команды radiant, а ноль означает поражение.

3) На 30 деревьях качество получилось следующее:

[0.68349320109209599, 0.69180598030193641, 0.66516221933438335, 0.68555120197581476, 0.69283583438804286]

Среднее качество для 10 деревьев: 0.676535942071

Среднее качество для 20 деревьев: 0.677013406329

Среднее качество для 30 деревьев: 0.683769687418

Время затраченное на выполнение кода:

Time elapsed: 0:01:48.816888(1 минута и почти 49 секунд)

На 100 деревьях среднее качество получилось следующее:

0.706232388471

Таким образом, да, при увеличении параметра n\_estimators качество растет, но с этим сильно возрастают и затраты ресурсов и времени, а разница в качестве не такая уж и колоссальная между, например, 30-ю деревьями и 100.

4)Судя по данной выборке, нет особого смысла увеличивать количество деревьев, потому как, если качество и будет расти, но не настолько значительно, чтобы оправдать затрату ресурсов. Можно «поиграть» с другими параметрами, например, learning\_rate и посмотреть какая комбинация параметров лучше решает задачу.

**Логистическая регрессия**

1)С параметром C=1 по умолчанию у меня получилось среднее качество 0.7166.

С параметром C=10(меньше регуляризации) у меня получилось такое же качество, как и при C = 1.

С параметром C=0.1(больше регуляризации, то есть модель попроще) получились такие же результаты, как и в предыдущих случаях.

По сравнению с градиентным бустингом код для логистической регрессии работает гораздо быстрее, при этом еще и дает небольшой, но выигрыш в оценке качества(0.7166 против 0.706).

2)После того, как были удалены признаки lobby\_type и r1\_hero, r2\_hero, ..., r5\_hero, d1\_hero, d2\_hero, ..., d5\_hero качество стало равным единицы для моделей с разными значениями параметра регуляризации. Такое качество «кричит» о том, что что-то сделано не так. Вероятно, удаление вышеуказанных признаков сильно влияет на реальную предсказательную силу модели, таким образом, получается, что не стоит убирать эти признаки, так как даже интуитивно, а после рассмотрения частоты выбора героев в этом остается мало сомнений, выбор героев в игре не случаен и влияет на результат игры.

3)Различных идентификаторов героев у меня получилось 108

4)При добавлении мешка слов у меня получилось следующее качество: 0.751883193642

Да, качество улучшилось по сравнению с предыдущими вариантами модели.

С разными значениями параметра C качество выходит одинаковое.

Если предположить, что выбор героев имеет влияние на конечный результат, то подход «мешка слов» лучше охватывает эту самую значимость выбора, учитывая больше сотни разных героев в игре.

5)У лучшего алгоритма на тестовой выборке(логистическая регрессия с мешком слов) получились следующие значения прогноза на тестовой выборке:

Минимальное значение оценки принадлежности: 0.0079782105601599283

Максимальное значение оценки принадлежности: 0.99647120915460308