- Уделить больше внимания второй части, описать почему вы выбрали ROC AUC (и как интерпретировать результирующую оценку модели)?

Выбор Roc AUC:
В первую очередь – решалась задача бинарной классификации – сузился выбор метрик качества.
Была проведена проверка на соотношение классов целевой переменной – по итогу обнаружен дисбаланс классов, то есть один класс сильно превосходит другой.

	prediction
prediction	
0	46638
1	6072

В связи с этим, доля правильных ответов не подходит, так как даже примитивный бесполезный константный алгоритм покажет высокую точность, если будет равен наибольшему классу.
ROC AUC представляет из себя кривую площадь под кривой ошибок и она не зависит от баланса классов, которая не зависит от баланса классов за счет.
ROC AUC измеряет долю верных срабатываний и долю ложных срабатываний. Для идеального алгоритма результат будет 1, для худшего около ½.
ROC AUC позволяет оценить модель в целом, не привязываясь к конкретному порогу в случае, когда алгоритм выдает вероятность принадлежность к каждому классу. Для меня это важно в этой задаче, так как я точно не знаю, какая Вам (заказчику) позволительна точность для преобразования класса 0 в класс 1.
ROC AUC может быть интерпретирован как вероятность того, что случайно выбранный объект класса 1 будет иметь более высокую вероятность принадлежать к классу 1, нежели случайно выбранный объект класса 0.

Точность 0.734 считается достаточно неплохой, основываясь на статьях и личном опыте.

Почему выбрали именно LGBM?

Нейронные сети я сразу отбросила, так как не вижу смысла пихать их везде и всюду, лишь бы было. Они хорошо работают только на больших данных, но и не всегда лучше обычных «классических» моделей, которые настраиваются/обучаются быстрее и проще.

После нахождения коэффициентов корреляции с целевой переменной – из моего рассмотрения выбыли все Линейные Классификаторы, так как ни один из признаков не связан с целевым сильной линейной зависимостью.

```
datetime       -0.249299
type           -0.070599
winner         -0.004740
round          -0.002500
length          0.042690
id_y            0.046720
player_cards    0.047452
user_id         0.049940
magic_used      0.125693
prediction      1.000000
Name: prediction, dtype: float64
```

Далее я основывалась только исходя из своего опыта – что обычно Бустинги работают лучше всего.

Я попробовала несколько моделей: LGBM, GradientBoostingClassifier, XGBClassifier – применила к ним поиск по сетке для нахождения лучших гиперпараметров и потом сравнивала по точности исходя из кроссвалидации.

Отдельно алгоритмы с деревьями решений я не использовала, но использовала, им обычно (исходя из моего опыта) требуется много признаков – от 15.

Я решила в готовом варианте ТЗ оставить только лучшую модель, чтобы это было похоже на реальный рабочий проект, ведь заказчику зачастую все равно как мы там что сделали – главное результат и его интерпретация.