# Этапы решения соревнования

DMIA 2016,
Гущин Александр

# Этапы работы над соревнованием

1) Организаторы
   a) Постановка задачи
   b) Определение функционала качества (== минус функции потерь)
   c) Сбор данных
   d) Тестовое решение, создание базового решения
2) Участники (пайплайн):
   a) Подготовка данных (джойны табличек, базовые преобразования фич)
   b) Понимание функционала качества
   c) Кросс-валидация
   d) Сабмит, проверка соответствия LB и CV
3) Участники (улучшение решения):
   a) Генерация новых признаков
   b) Смешивание разных моделей
   c) Поиск ликов
   d) Использование визуализаций

# Понимание функционала качества

1. Выбор алгоритмов, оптимизирующих правильный функционал
2. Маштабирование признаков:
   a. Нужно для линейных моделей, KNN, NN, бустинга над линейными моделями
   b. Не нужно для деревьев, леса, бустинга над деревьями
3. Может ли понадобиться постобработка ответов
   a. Например, превращение отрицательных значений в ноль
   b. Или калибрация вероятностей
4. Выбор приемлимых способов смешивания моделей
   a. y = ypred1 ** 0.9 + ypred2 ** 1.4  может сработать для AUC, но не для Logloss

# Кросс-валидация (1)

1) Виды кросс валидации
    a) Holdout: n = 1
    b) K-fold: n = k
    c) Leave-one-out: n = len(train)
2) Stratified (стратификация)
    a) Классификация: сохраняется соотношение классов
    b) Регрессия: сохраняется распределение целевой переменной

# Кросс-валидация (2)

Разбиение:

a. Случайно, по строчкам
b. По времени
c. По географическим координатам (например, по городам)
d. По некоторой фиче (user_id, shop_id...)
e. Комбинировано, например, по географии и времени: со своего момента в каждом регионе

# Сабмит и проверка кросс-валидации (3)

Правильная кросс-валидация: изменение качества модели на валидации соответствует изменению качества модели на лидерборде.

Частые проблемы:

1. Мало данных в тесте - более маштабная CV.
   Обычно: KFold вместо Holdout, увеличить K
   a. Случайное разбиение: как обычно
   b. Разбиение по времени: (до 2015, 2016), (до 2014, 2015), ...
   c. Разбиение по другим признакам: KFold на уникальных значениях признака
2. ?

# Генерация новых признаков

1. Маштабирование числовых признаков:
   a. Нужно для линейных моделей, KNN, NN, бустинга над линейными моделями
   b. Не нужно для деревьев, леса, бустинга над деревьями
2. Подстройка категориальных признаков под алгоритм

   `pd.get_dummies`
3. Простые операции на парах признаков (умножить, сложить, ==)
4. Подсчёт признаков по сгруппированным данным

   `pd.groupby(['customer_id', 'shop_id']).visit.count()`
5. Сложные связи между признаками

   догадки по визуализациям + понимание данных

   `xnew = x1 + x2*24 + x3*24*60`
6. Категориальные признаки: кодирование средних

   не переобучитесь! (out-of-fold)

   `pd.groupby(['categorical_feature']).target.mean()`
7. ?

# Смешивание разных моделей

1. Средние: арифметическое, геометрическое, гармоническое...

   Взвешивайте модели
2. Сложные комбинации:
   a. y1 ** 2 + y2 / 14 (AUC)
   b. y1 * 1.1 - y2 * 0.1 (RMSE)
3. Блендинг

   Оставляем ещё один холдаут, чтобы генерировать новые признаки (предсказания) для него и учить метамодель на нём
4. Стекинг

   Генерируем признаки для всех данных (Out-of-fold):
   a. Разбиваем данных на K частей (==KFold)
   b. Для каждой часть
      i. учимся на оставшихся
      ii. Сохраняем предсказание как новый признак для выбранной части

# Поиск ликов

Потенциальные места

1. Сортировка данных
2. Разбиение на трейн и тест
3. Дублирующиеся строки
4. ?

# Использование визуализаций

1. Генерация новых признаков

   ```
   plt.scatter(x.feature1, x.feature2, color=target)
   ```

2. Смешивание моделей

   ```
   plt.scatter(ypred1, ypred2, color=target)   # color=target на валидации
   ```

3. Поиск ликов
4. ...