

# Этапы решения соревнования

ML SCHOOL 2018

# Этапы работы над соревнованием

## 1) Организаторы

- Постановка задачи
- Определение функционала качества
- Сбор данных
- Тестовое решение, создание базового решения

## 2) Пайплайн:

- Понимание функционала качества
- Кросс-валидация
- Подготовка данных (джойны таблицек, базовые преобразования фич)
- Сабмит, проверка соответствия LB и CV

## 3) Улучшение решения:

- Генерация новых признаков
- Смешивание разных моделей
- Поиск ликов
- Использование визуализаций

# Понимание функционала качества

- 1) MAE
- 2) Logistic Loss
- 3) AUC
- 4) RMSE
- 5) Accuracy
- 6) RMSLE ???












```
import math

#A function to calculate Root Mean Squared Logarithmic Error (RMSLE)
def rmsle(y, y_pred):
    assert len(y) == len(y_pred)
    terms_to_sum = [(math.log(y_pred[i] + 1) - math.log(y[i] + 1)) ** 2.0 for i, pred in
enumerate(y_pred)]
    return (sum(terms_to_sum) * (1.0/len(y))) ** 0.5
```












# Понимание функционала качества

- 1) Выбор алгоритмов, оптимизирующих правильный функционал
- 2) Масштабирование признаков:
  - Нужно для линейных моделей, kNN, NN, бустинга над линейными моделями
  - Не нужно для деревьев, леса, бустинга над деревьями
- 3) Может ли понадобиться постобработка ответов
  - Например, превращение отрицательных значений в ноль
  - Или калибровка вероятностей
- 4) Выбор приемлемых способов смешивания моделей
  - $y = \text{ypred1} ** 0.9 + \text{ypred2} ** 1.4$  — может сработать для AUC, но не для Logloss

# Кросс-валидация

1031	▲ 833	MAIZA		0.55037	27	3mo
1032	▲ 837	SVJ24		0.55037	16	3mo
1033	▲ 886	Jonathon		0.55035	15	4mo
1034	▲ 1799	weijunchen		0.55035	1	5mo
1035	▼ 670	Yasin		0.55034	40	3mo
1036	▼ 738	Leonard Zhao		0.55034	29	3mo
1037	▼ 789		  	0.55034	149	3mo
1038	▼ 306	Avishek		0.55033	32	3mo

# Yet another example

10	▲ 16	TheFirstBrazilianSniper - Fed...		0.67372	74	5mo
11	▼ 2	Peace Data - Skoltech - Russia		0.67361	98	5mo
12	▼ 4	UCUpnic - UCU - Ukraine		0.67354	63	5mo
13	▲ 6	Make Latin America Great Ag...		0.67341	81	5mo
14	▼ 9	E3 Analytics-UNI-Peru		0.67310	121	5mo
15	—	Imagouille - ENSIMAG - France		0.67296	132	5mo
16	▼ 9	Outliers - SPbSU - Russia		0.67236	47	5mo
17	▲ 1	SID - UPS - France		0.67207	100	5mo
18	▲ 6	Lab Rats - HSE NN - Russia		0.67193	79	5mo
19	▼ 3	Confounders		0.67080	112	5mo
20	▼ 8	Rebyatishki - MIPT - Russia		0.67046	72	5mo

# Кросс-валидация

## 1) Виды кросс валидации

- Holdout:  $n = 1$
- K-fold:  $n = k$
- Leave-one-out:  $n = \text{len}(\text{train})$
- Sliding window

## 2) Стратификация

- Классификация: сохраняется соотношение классов

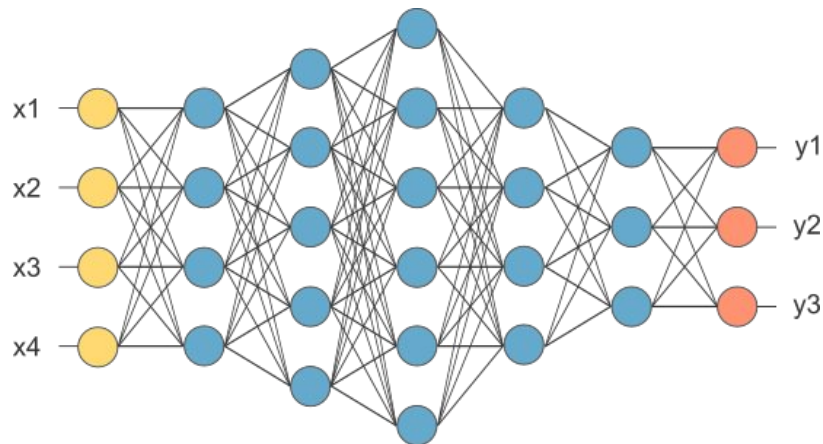
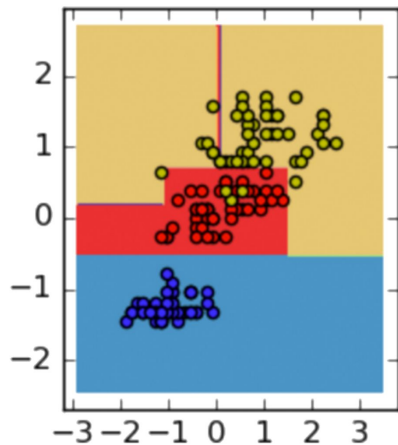
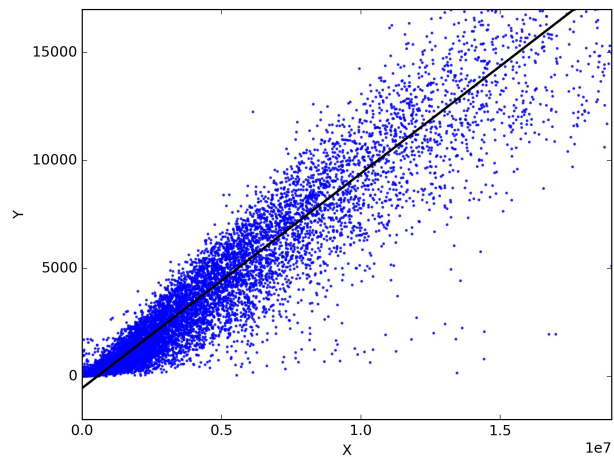
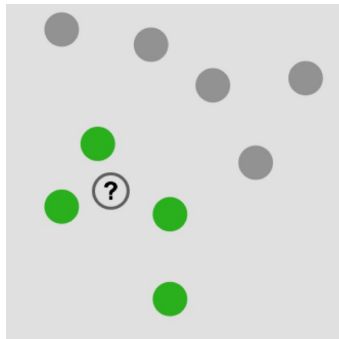
# Кросс-валидация

Разбиение:

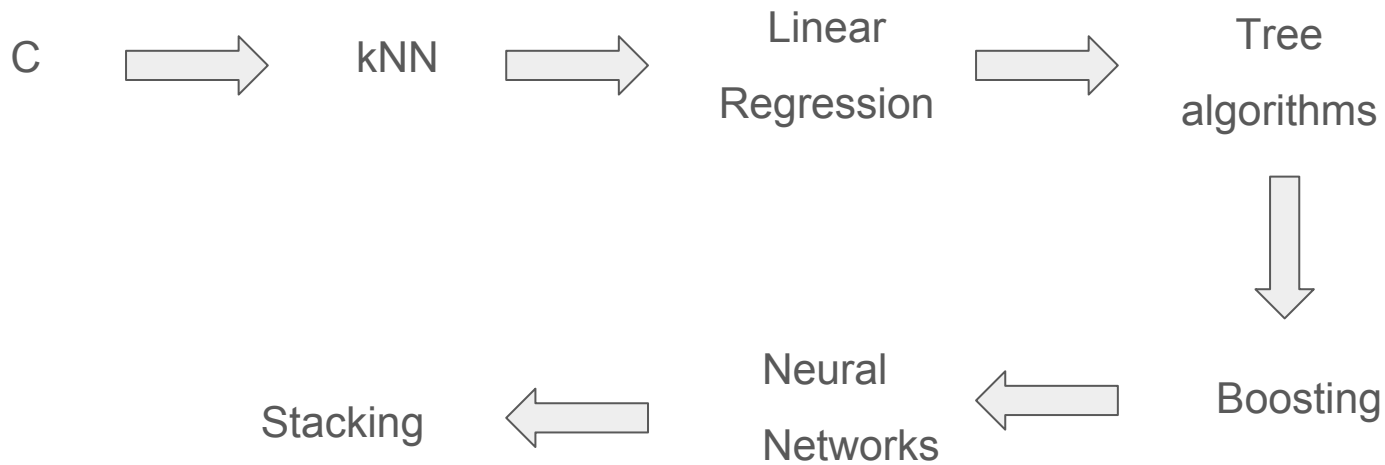
- Случайно, по строчкам
- По времени
- По географическим координатам
- По некоторой фиче (user\_id, shop\_id ...)
- Комбинировано, например, по географии и времени: со своего момента в каждом регионе



# Бейзлайн



# Бейзлайн



# Сабмит и проверка кросс-валидации

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

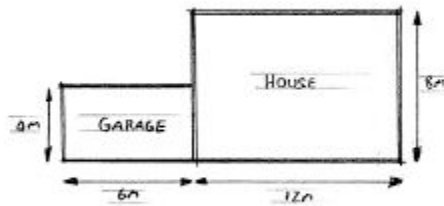
Частые проблемы:

Мало данных в тесте - более масштабная CV.

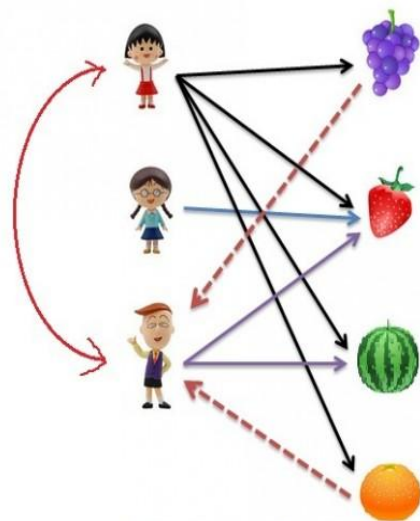
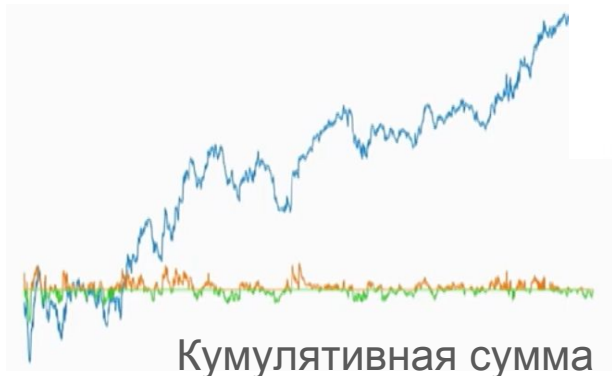
Обычно: KFold вместо Holdout, увеличить K

- Случайное разбиение
- Разбиение по времени
- Разбиение по другим признакам: KFold на уникальных значениях признака

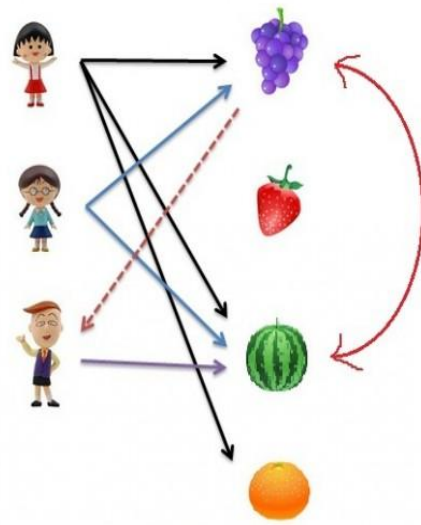
# Генерация новых признаков



$$S = H * W$$



User-based filtering



Item-based filtering

# Генерация новых признаков

## 1) Масштабирование числовых признаков:

- Нужно для линейных моделей, KNN, NN, бустинга над линейными моделями
- 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

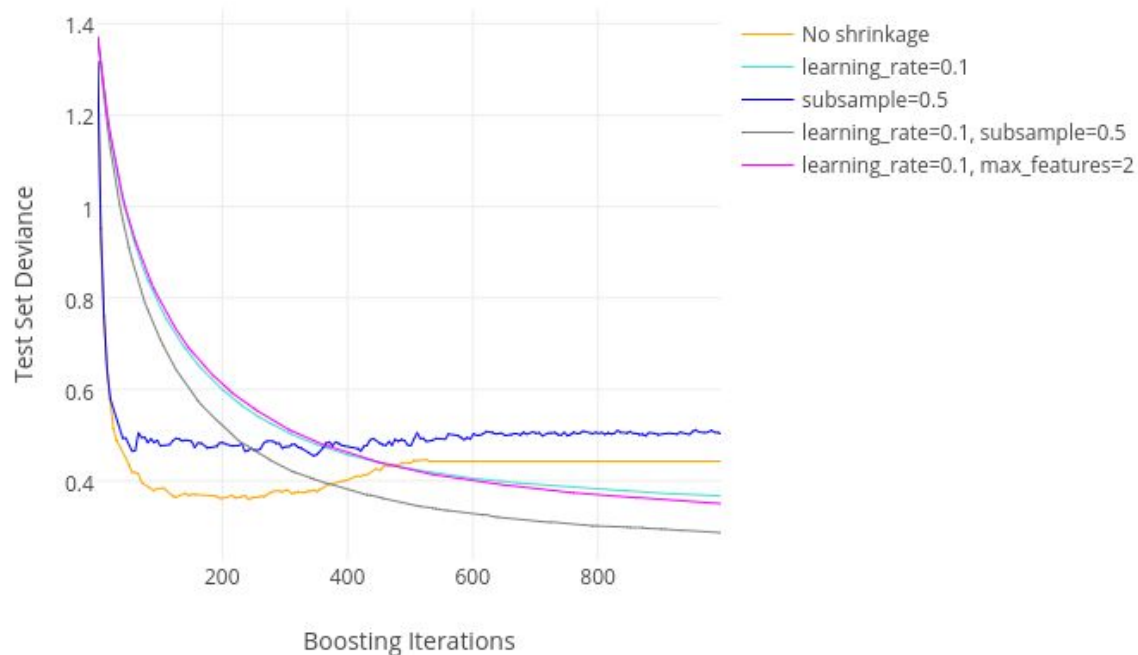
$x_{new} = x_1 + x_2 * 24 + x_3 * 24 * 60$

## 6) Категориальные признаки: кодирование средних

не переобучитесь! (out-of-fold)

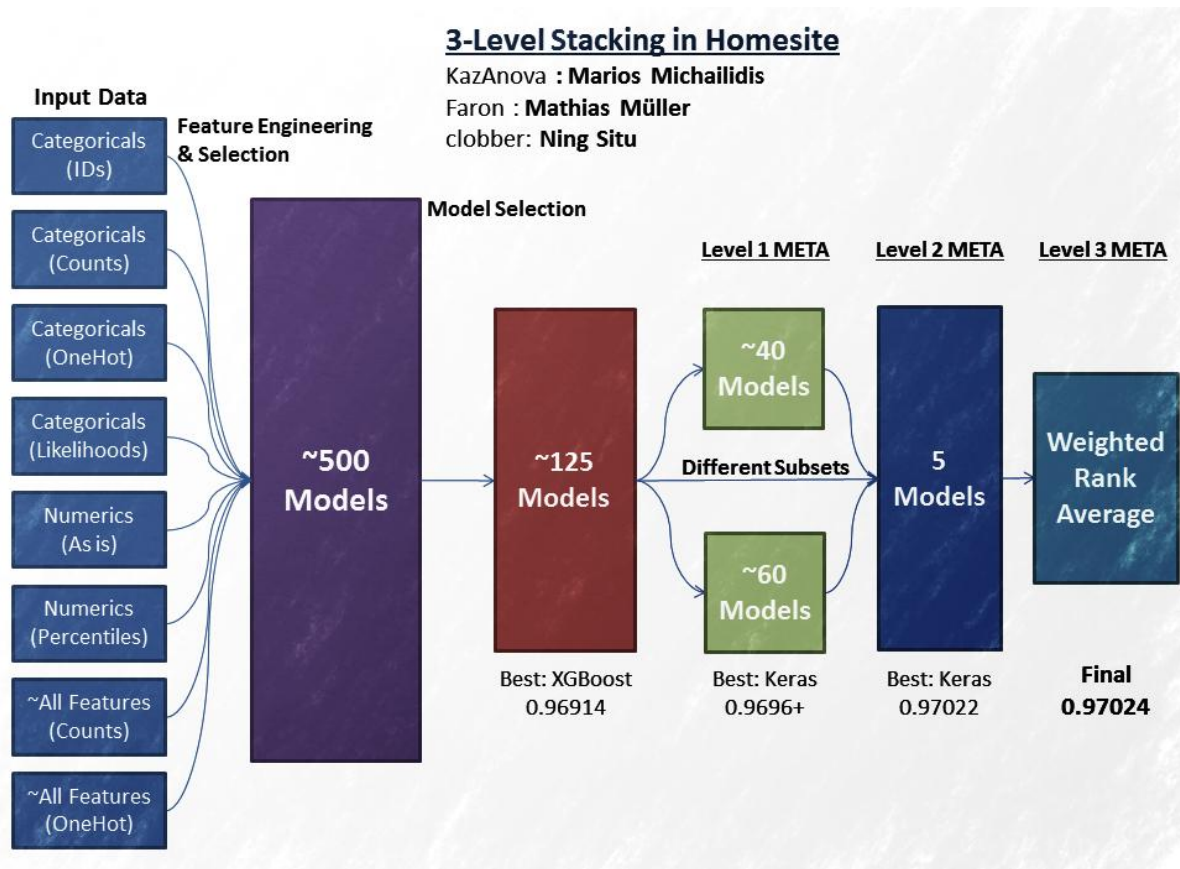
`pd.groupby(['categorical_feature']).target.mean()`

# Тюнинг моделей



- Objective
- Learning rate
- Max depth
- Subsample
- Colsample by tree
- Colsample by level

# Смешивание разных моделей



# Смешивание разных моделей

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

Взвешивайте модели

## 2. Сложные комбинации:

- $y_1 ** 2 + y_2 / 14$  (AUC)
- $y_1 * 1.1 - y_2 * 0.1$  (RMSE)

## 3. Блендинг

Оставляем ещё один холдаут, чтобы генерировать новые признаки (предсказания) для него и учить метамоделю на нём

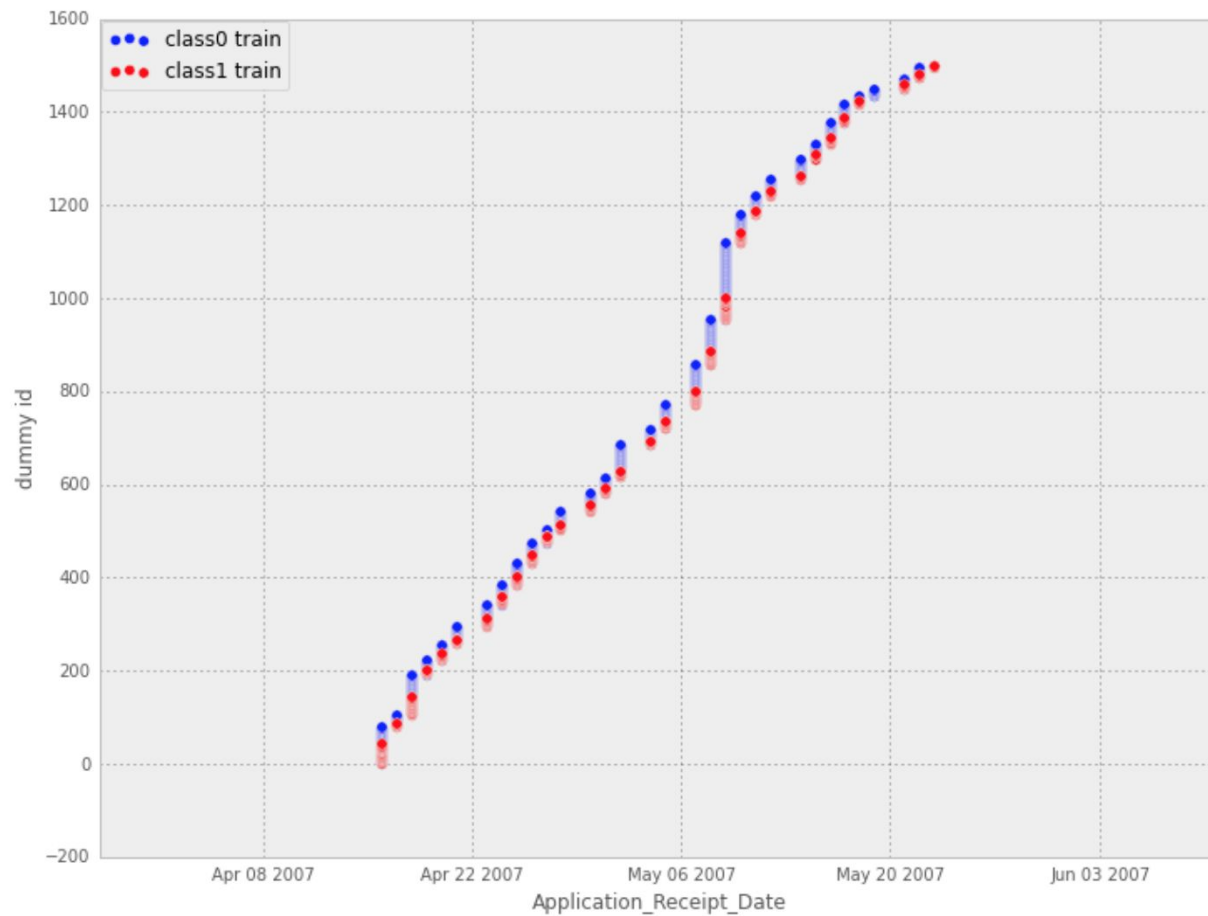
## 4. Стекинг

Генерируем признаки для всех данных (Out-of-fold):

- Разбиваем данных на K частей (==KFold)
- Для каждой части
  - учимся на оставшихся
  - Сохраняем предсказание как новый признак для выбранной части



# Поиск лиц



# Поиск ликов

Потенциальные места

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

# Использование визуализаций

## 1. Генерация новых признаков

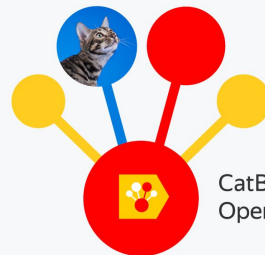
```
plt.scatter(x.feature1, x.feature2, color=target)
```

## 2. Смешивание моделей

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

## 3. Поиск ликов

# Инструменты



CatBoost  
Open-source ML library

*dmlc*  
**XGBoost**



theano



+

