ТЕХНОСФЕРА

Занятие № 5

Ансамбли моделей

Евгений Некрасов

План занятия

1. Оценка качества моделей
2. Ансамбли моделей
3. Стэкинг
4. Бэггинг
5. Random Forest
6. Gradient Boosting

Как строить модель

Вы получили размеченный набор данных

Задача:

1. Построить хорошую модель
2. Оценить качество модели

Проблема переобучения

Способы обучиться на наборе данных:

1. Запомнить правильные ответы
2. Найти общие закономерности в предоставленном наборе данных

Для оценки качества модели нельзя использовать те же данные, что и для построения модели.

Shuffle & Split

Перемешиваем самплы, делим датасет на две части (Train и Test) в некоторой пропорции.
На Train обучаем модель, на Test оцениваем качество.

Особенности:

- Простая реализация
- Разумно использовать когда данных "много"

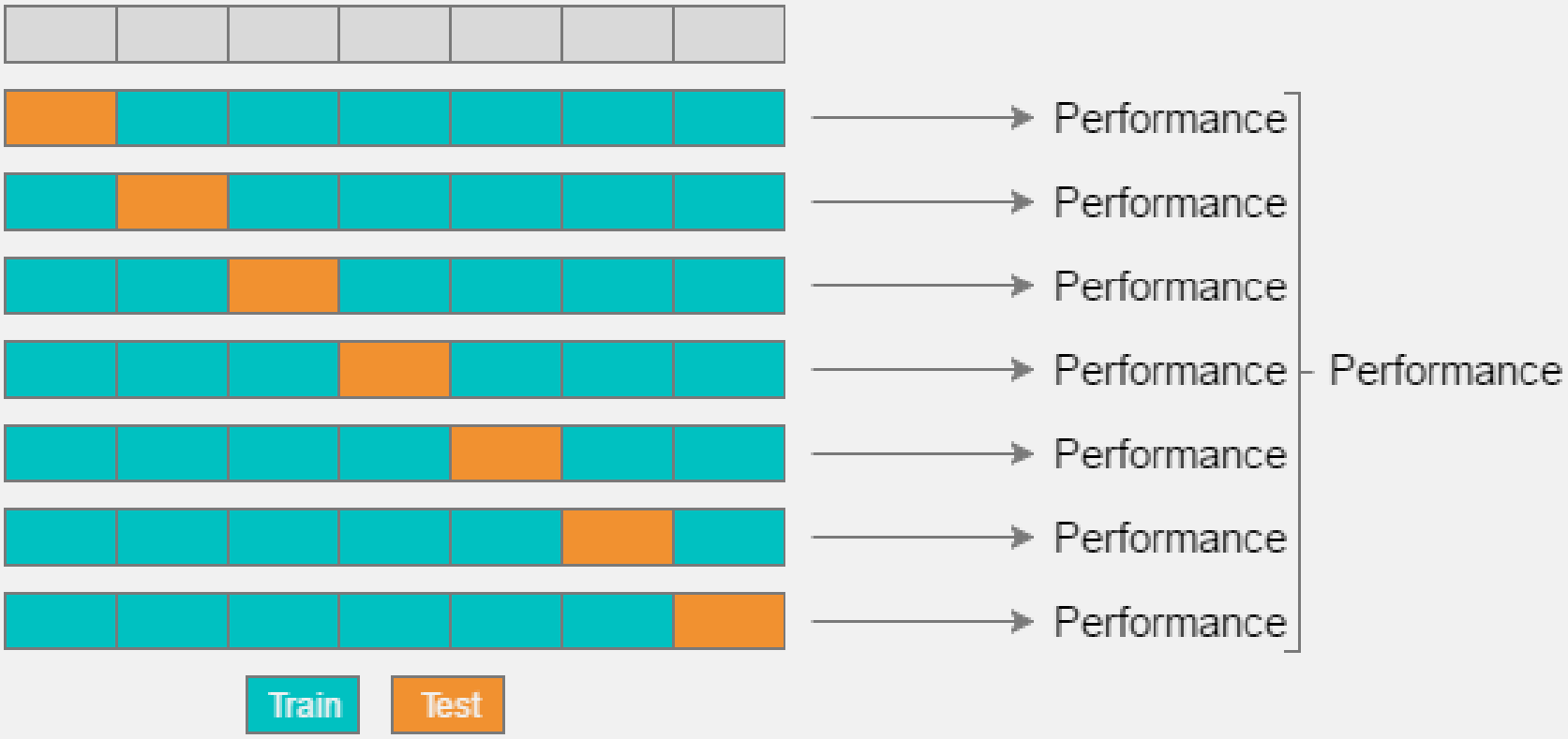
K-fold

Разбиваем датасет на K равных частей, затем строим K моделей где в качестве Test берем одну из частей, а все остальные используем как Train. На Train обучаем модель, на Test оцениваем качество.

Особенности:

- Используем все данные как для построения моделей, так и для оценки качества
- Один из наиболее популярных методов оценки качества моделей

K-fold


Performance
Performance
Performance
Performance
Performance
Performance
Performance
Performance
Train
Test

Leave One Out

Экстремальный случай **K-fold**, когда K равно числу самплов в наборе данных.

Особенности:

- Модель на датасете без одного самплa практически идентична модели на полном датасете
- Может быть эффективно посчитан для некоторых видов моделей

Повторные разбиения

K-fold или **Shuffle & Split**, повторенный N раз с различными разбиениями.

Особенности:

- В N раз выше вычислительная сложность
- Эффективны когда данных "мало" или данные "шумные"

Стратифицированные разбиения

Стратифицированные разбиения - это такие разбиения, которые сохраняют определенные свойства исходной выборки.

Свойствами могут быть:

- Распределение целевой переменной
- Распределения некоторых признаков

Стратифицированные разбиения

Плюсы:

- Могут обеспечить большую точность, чем простой случайный выбор
- Позволяют избежать "непредставительной" выборки (например отсутствие какого-либо класса объектов в разбиении)

Минусы:

- Сложнее в реализации

Групповые разбиения

Иногда в наборе данных самплы разбиты на **группы**.

Например:

- Фотографии разных опухолей у одного и того же экспериментального животного составляют **группу**
- Оценки одного и того же преподавателя на экзамене составляют **группу**

Игнорирование **групп** приводит к некорректной оценке качества модели!

Групповые разбиения

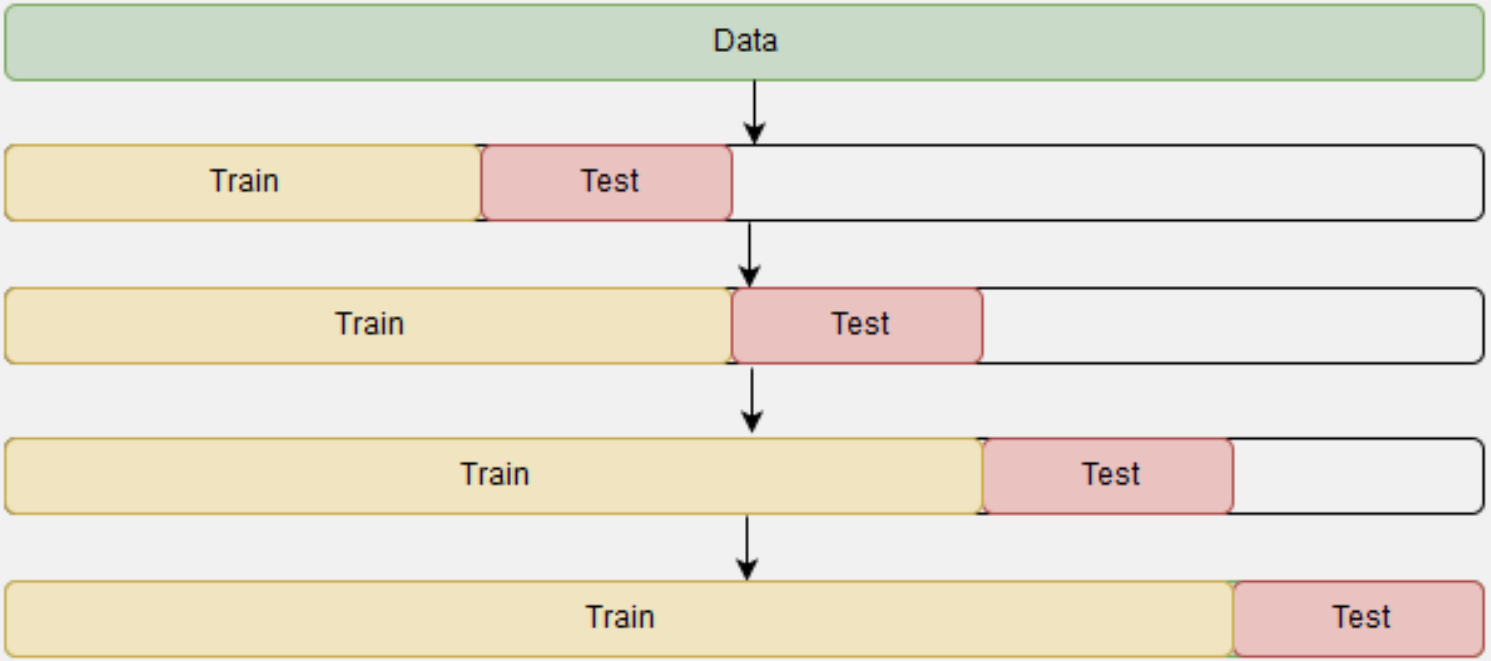
Групповые K-fold или **Shuffle & Split** - это такие разбиения при которых все самплы одной группы попадают в один и то же фолд или сплит.

Особенности:

- Позволяют корректно работать с данными в которых есть группы

Разбиение временных рядов

Разбиение временных рядов следует проводить по времени события.



Train, Test, Validation

На **Train** обучаем модели-кандидаты

На **Test** оцениваем модели-кандидаты и выбираем лучшую

На **Validation** проверяем, что все работает как ожидалось

- **Validation** никак не используем при построении модели!
- На гиперпараметрах модели тоже можно переобучиться!

Общие рекомендации

Выбор схемы валидации – это баланс вычислительной сложности и точности оценки

Схемы оценки моделей:

1. Shuffle & Split
2. K-fold K=2-4
3. K-fold K=5-10
4. Repeated K-fold K=5-10

Как правило 3x5-fold работает лучше чем 1x15-fold

Ансамбли моделей

Коллективное принятие решений как правило превосходит по качеству индивидуальное принятие решений

О значимости ансамблей

Лучшие алгоритмы машинного обучения по точности:

- Градиентный бустинг для классических задач
- Искусственные нейронные сети для изображений, видео, звука

В соревнованиях kaggle всегда* побеждают ансамбли

Простое голосование

Классификация: класс определяется большинством голосов

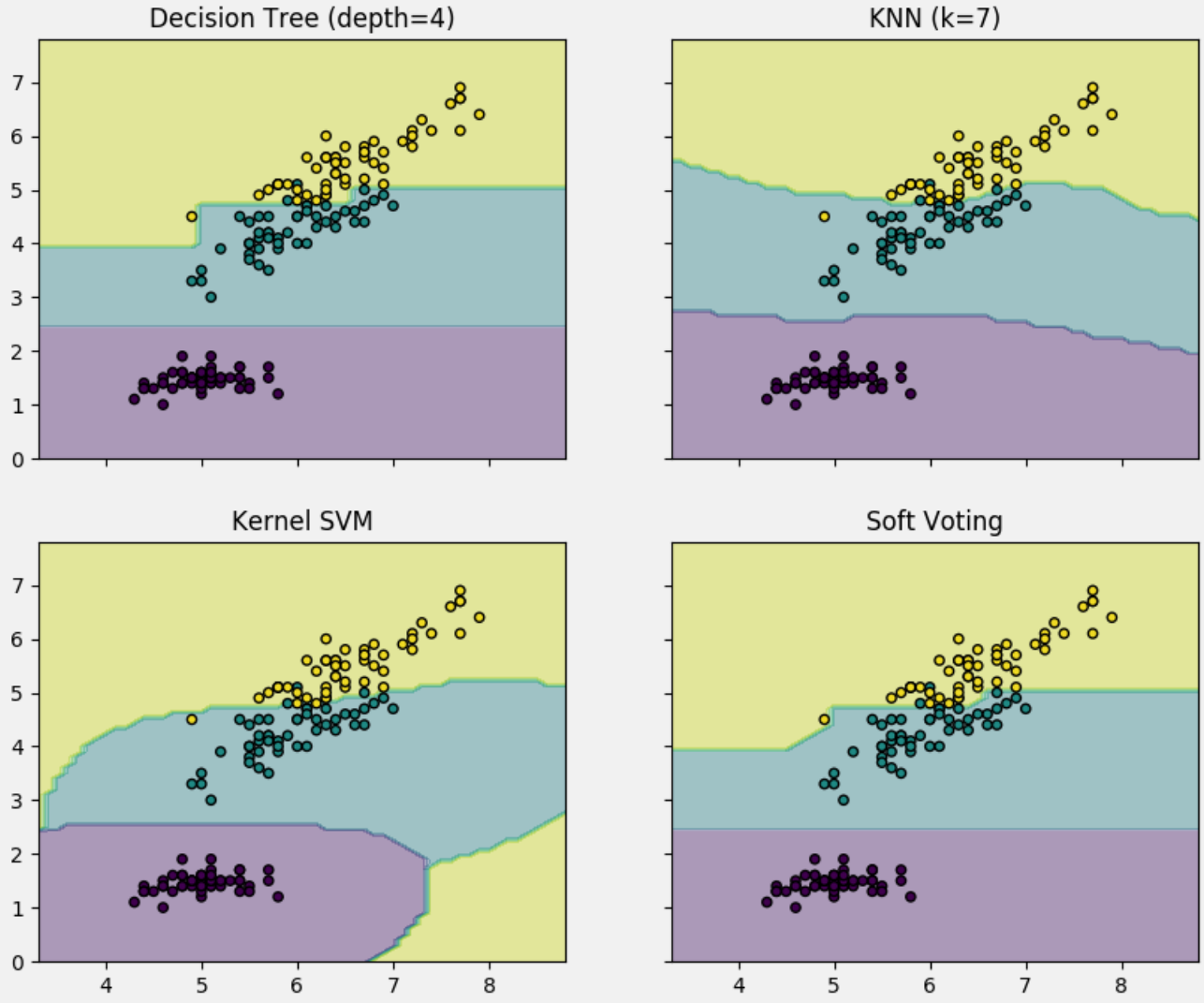
Регрессия: среднее значение

Взвешенное голосование

Классификация: класс определяется большинством голосов с учетом веса

Регрессия: среднее взвешенное значение

Пример голосования



Недостатки голосования

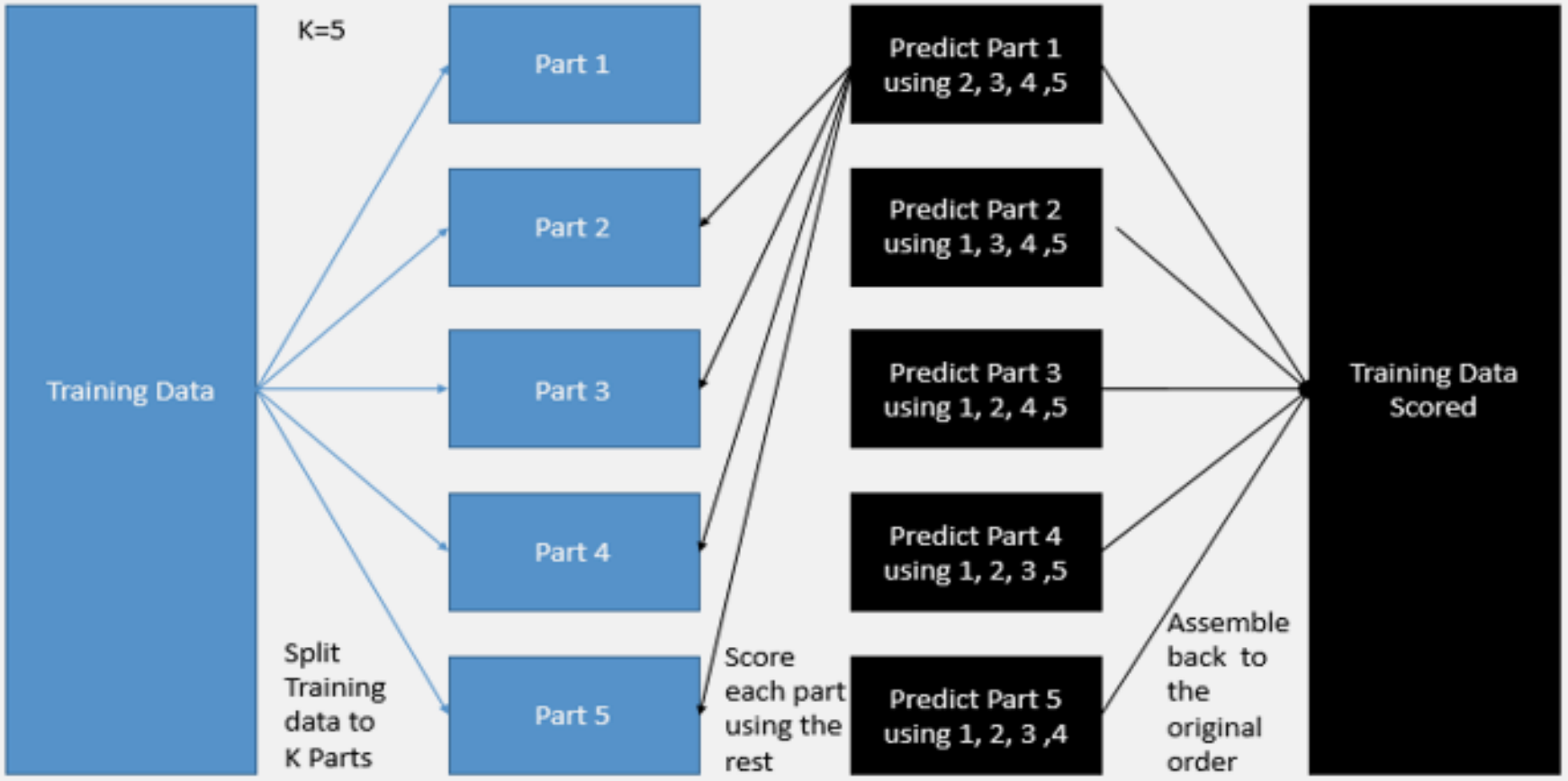
1. Голосование не учитывает особенностей сампла
2. Голосование не учитывает особенностей поведения отдельных моделей
3. Голосование по сути является простой моделью

Стэкинг

Идея:

Построим модель, которая будет предсказывать правильный ответ на основе предсказаний других моделей

Стэкинг - преобразование Train



Стэкинг - преобразование Test

Вариант 1:

1. Обучаем модель на полном датасете
2. Выполняем предсказание на Test

Вариант 2:

1. Выполняем предсказание на Test каждой из K моделей кросс-валидационного предсказания
2. Усредняем K предсказаний

Стэкинг

Кросс-валидационное предсказание будем называть **метапризнаком**.

Стэкинг можно делать как на наборе метапризнаков, так и на наборе метапризнаков + набор исходных признаков.

Стэкинг может быть многоуровневым.

Стэкинг реальный пример

Соревнование:

Dato Truly Native?

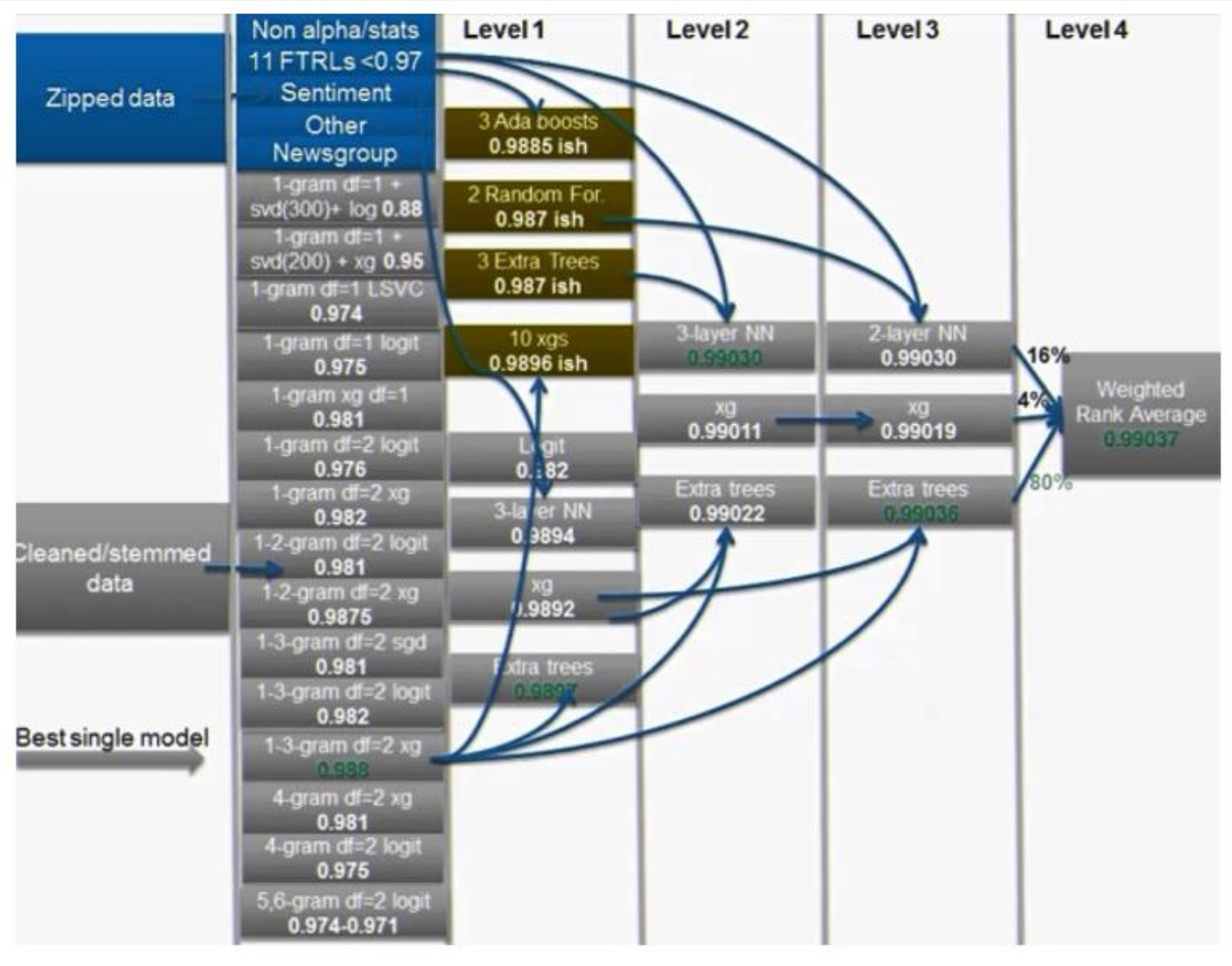
Задача:

На основе данных html страниц определять скрытую рекламу

Пример:

Решение команды, занявшей 1 место

Стэкинг реальный пример



Бэггинг и Бустинг

Идея бэггинга:

1. Построим много слегка различающихся моделей
2. Методом усреднения выберем итоговый ответ

Идея бустинга:

Каждая следующая модель в ансамбле пытается предсказать ошибку всех предыдущих моделей ансамбля

Бутстрэп

Бутстрэп (bootstrap) – метод исследования распределения статистик вероятностных распределений, основанный на многократной генерации псевдовыборок на базе имеющейся выборки.

1. Из исходной выборки генерим псевдовыборки методом случайного выбора с возвращением.
2. На псевдовыборках считаем целевую статистику.
3. Анализируем распределение целевой статистики на псевдовыборках.

Бэггинг

Bagging – Bootstrap aggregating (Leo Breiman, 1994)

- Из Train генерим методом случайного выбора самплов с возвращением $Train'_1$ … $Train'_N$
- На каждом Train' строим модель
- Итоговое предсказание получаем усреднением предсказаний всех моделей или простым голосованием

RSM

RSM – Random Subspace Method или feature bagging

- Из Train генерим методом случайного выбора признаков без возвращения $Train'_1$ … $Train'_N$
- На каждом Train' строим модель
- Итоговое предсказание получаем усреднением предсказаний всех моделей

Random Forest

Алгоритм:

1. Выполняем N раз:

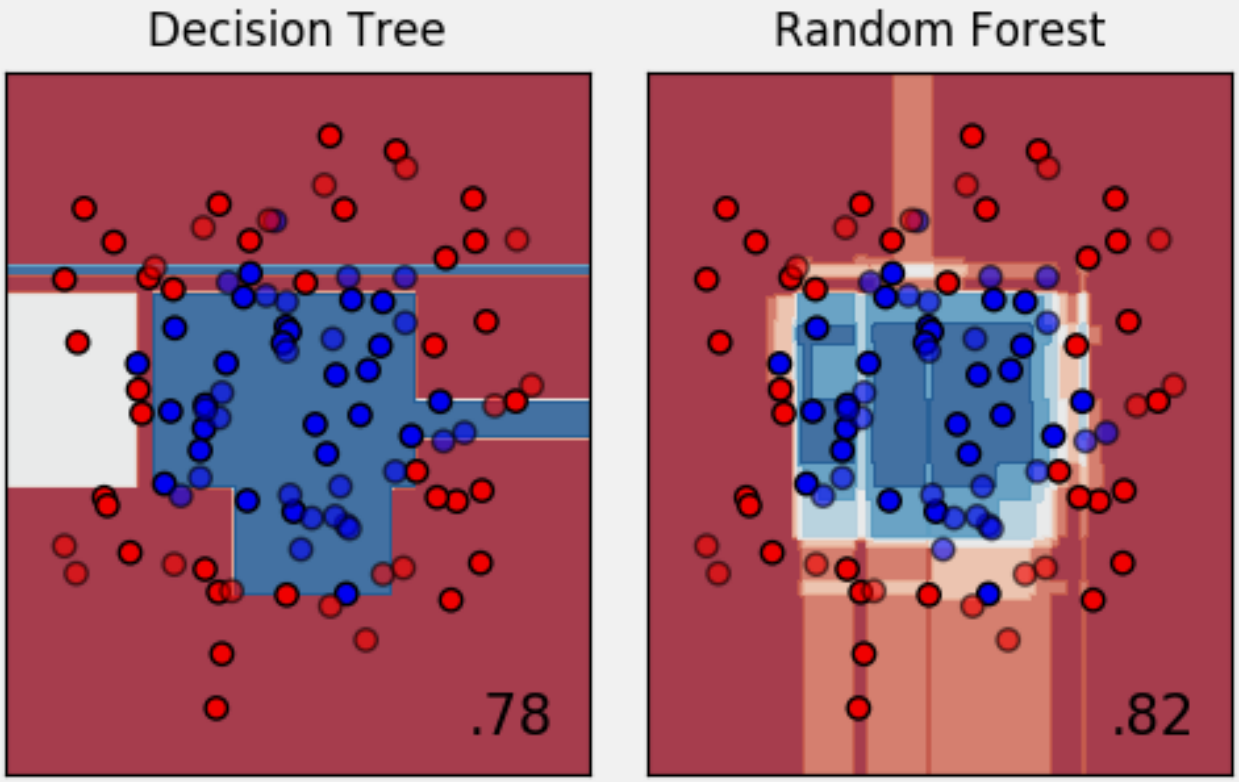
1.1. Бутстрэп самплов

1.2. Случайное подпространство признаков

1.3. Построение дерева решений

2. Выбираем ответ модели методом усреднения предсказаний или простого голосования

Random Forest



Random Forest не переобучается с ростом числа деревьев

Random Forest

Плюсы:

1. Алгоритм прост
2. Не переобучается
3. Хорошо параллелится
4. Не требует сложной настройки параметров
5. Не требует нормализации данных

Минусы:

1. Модели не интерпретируемые
2. Плохо работает с полиномиальными зависимостями

Градиентный бустинг

Идея:

1. Представляем итоговую модель f(x) как сумму слабых моделей h(x).
2. Пусть задана дифференцируемая функция потерь L(y, f(x))
3. На каждом шаге мы ищем модель h(x), которая бы апроксимировала вектор антиградиента L

Градиентный бустинг

1. Инициализировать GBM константным значением $\hat{f}(x) = \hat{f}_0, \hat{f}_0 = \gamma, \gamma \in \mathbb{R}$

$$\hat{f}_0 = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

2. Для каждой итерации $t = 1, \ldots, M$ повторять:

   1. Посчитать псевдо-остатки r_t

$$r_{it} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}(x)}, \quad \text{for } i = 1, \ldots, n$$

   2. Построить новый базовый алгоритм $h_t(x)$ как регрессию на псевдо-остатках $\{(x_i, r_{it})\}_{i=1,\ldots,n}$

   3. Найти оптимальный коэффициент ρ_t при $h_t(x)$ относительно исходной функции потерь

$$\rho_t = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i) + \rho \cdot h(x_i, \theta))$$

   4. Сохранить $\hat{f}_t(x) = \rho_t \cdot h_t(x)$

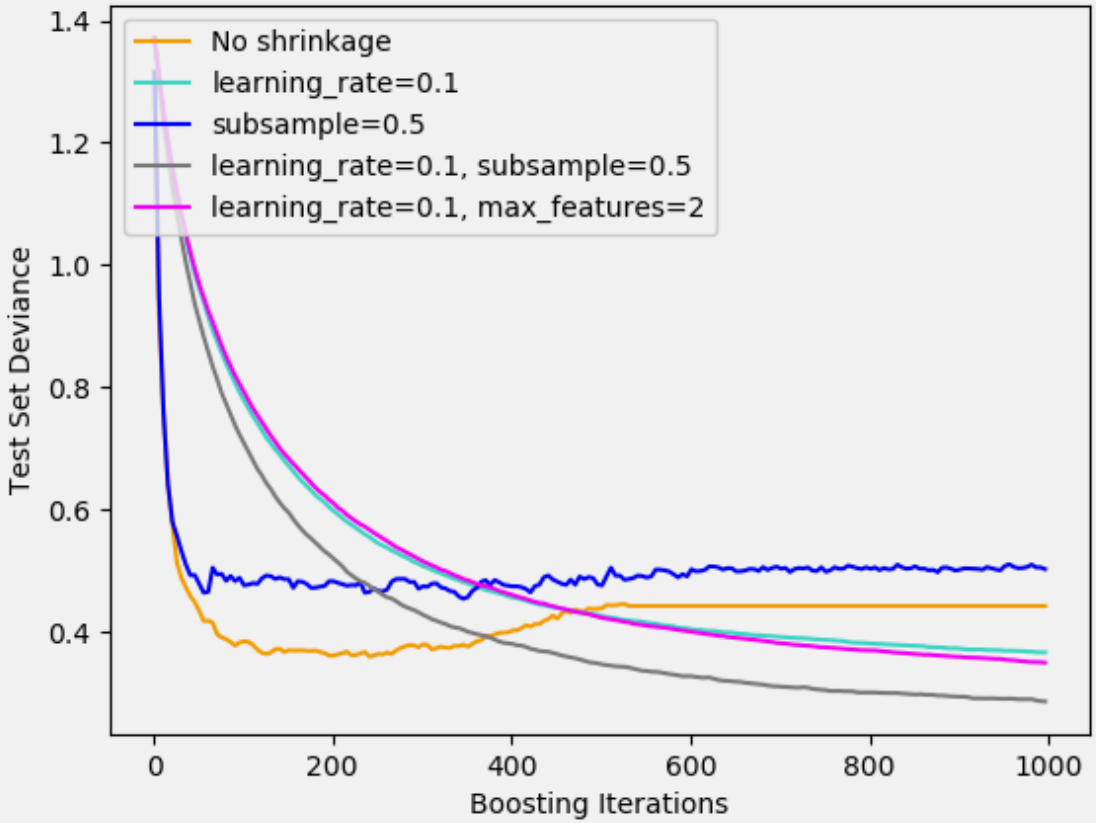
   5. Обновить текущее приближение $\hat{f}(x)$

$$\hat{f}(x) \leftarrow \hat{f}(x) + \hat{f}_t(x) = \sum_{i=0}^{t} \hat{f}_i(x)$$

3. Скомпоновать итоговую GBM модель $\hat{f}(x)$

$$\hat{f}(x) = \sum_{i=0}^{M} \hat{f}_i(x)$$

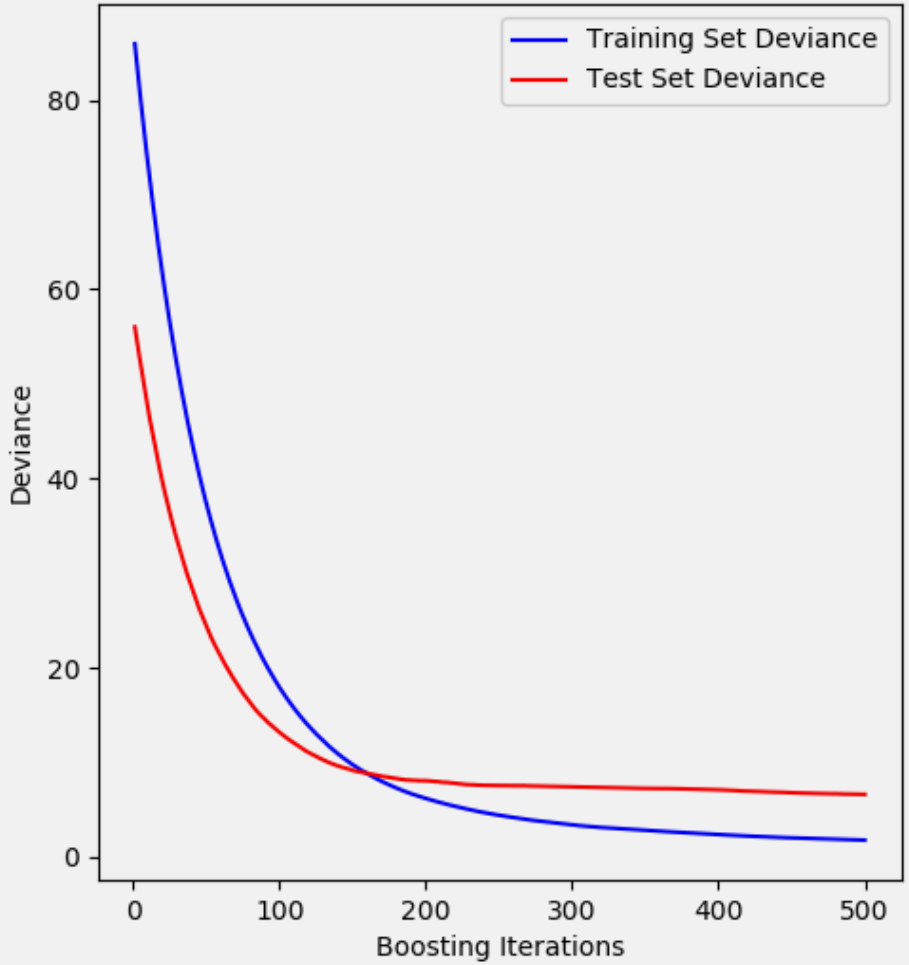
Регуляризация градиентного бустига



В алгоритме градиентного бустинга также применимы bagging и RSM

Переобучение градиентного бустига

Градиентный бустинг почти не переобучается



Интерактивные примеры

http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html

http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html

Домашнее задание

Реализовать функцию:

stack_pred(estimator, X, y, Xt, k=3, method='predict')

return sX, sXt

Для разбиения на фолды использовать:

sklearn.model_selection.KFold

Реализовать стэкинг вариант 2:

предсказания на Xt - усреднение предсказаний k моделей

Домашнее задание №#5

- Реализовать алгоритм стэкинга
- Выложить решение на github.com
- Прислать ссылку на код решения

Срок сдачи

23 октября 2017

ТЕХНОСФЕРА

Спасибо за внимание!

Евгений Некрасов

e.nekrasov@corp.mail.ru