

Структура презентации

1. Соревнование по машинному обучению по оценке недвижимости
2. Наш опыт создания сервиса для конечного пользователя на основе такой модели

Решение задачи оценки
стоимость коммерческой
залоговой недвижимости
команды
skyNet

Наша команда



Автомонов
Андрей



Волков
Дмитрий



Горкунов
Михаил



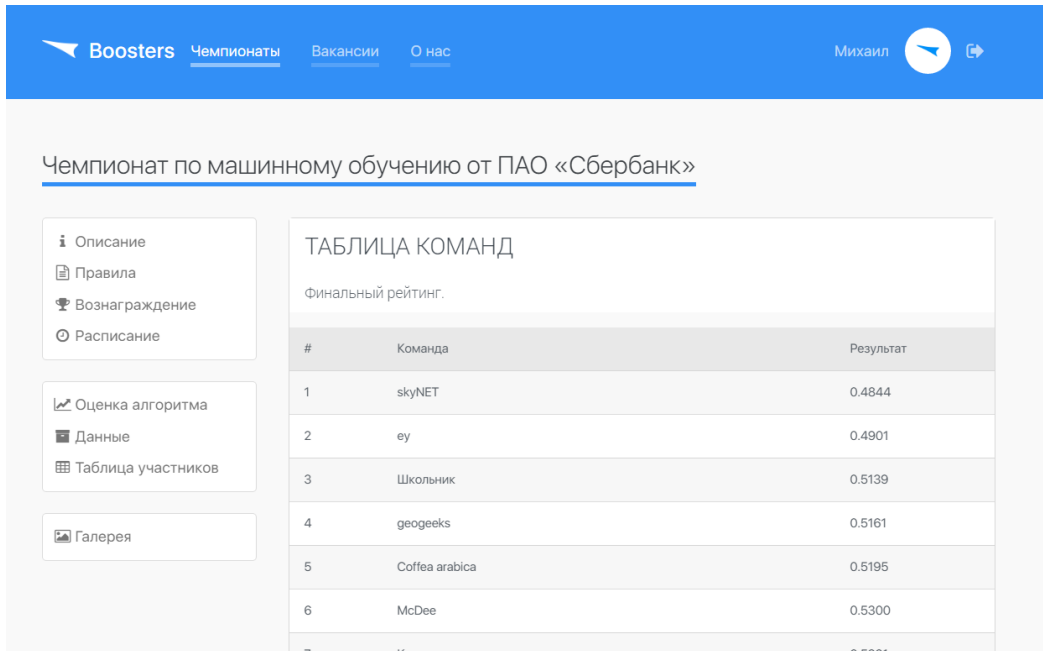
Загорулькин
Дмитрий



Швец Павел

Результаты

- Первое место по Leaderboard
- Второе место после презентации итогов соревнования



The screenshot shows the 'Boosters' website with a blue header. The navigation bar includes 'Boosters', 'Чемпионаты' (highlighted), 'Вакансии', and 'О нас'. On the right, there is a user profile 'Михаил' and a search icon. The main content area is titled 'Чемпионат по машинному обучению от ПАО «Сбербанк»'. On the left, there is a sidebar with links: 'Описание', 'Правила', 'Вознаграждение', 'Расписание', 'Оценка алгоритма', 'Данные', 'Таблица участников', and 'Галерея'. The main part of the page displays the 'ТАБЛИЦА КОМАНД' (Team Table) with the subtitle 'Финальный рейтинг.' (Final ranking.). The table lists 7 teams and their scores.

| # | Команда | Результат |
|---|----------------|-----------|
| 1 | skyNET | 0.4844 |
| 2 | eu | 0.4901 |
| 3 | Школьник | 0.5139 |
| 4 | geogeeks | 0.5161 |
| 5 | Coffea arabica | 0.5195 |
| 6 | McDee | 0.5300 |
| 7 | Kanna | 0.5231 |

Предоставленные данные

Train:

- ID записи
- Тип объекта (офис, склад...)
- Дата
- Адрес в текстовой форме
- Цена
- Площадь
- Метро - Список (если имеются) ближайших станций метро с расстоянием до них
- Широта
- Долгота
- Текстовое описание недвижимости

Test:

- ID записи
- Город
- Адрес (улица, дом)
- Тип помещения
- Общая площадь
- Площадь 1-го этажа (кв. м.)
- Назначение площадей 1-го этажа
- Площадь цоколя (кв. м.)
- Назначение площадей цоколя
- Площадь подвала (кв. м.)
- Назначение площадей подвала
- Площадь антресоли (кв. м.)

И так далее...

Постановка задачи

Описание:

Необходимо предсказать стоимость объектов недвижимости в тысячах рублей.

Метрика:

RMSLE:
$$\sqrt{\frac{1}{n} \sum (\log(p_i + 1) - \log(a_i + 1))^2}$$

Первичный анализ данных

1. Сразу обнаружился Dataleak по ID
2. Очень мало одинаковых по смыслу фичей в трейне и тесте
3. Даже те фичи, которые по смыслу совпадают необходимо преобразовать
4. Единственный способ успешно участвовать в соревновании – собрать дополнительные данные

Объемы собранных данных

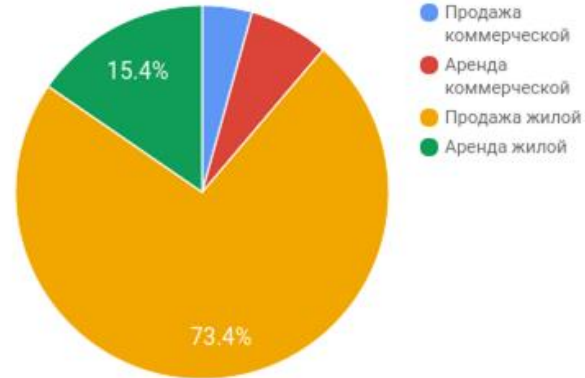
ЦИАН ~ 750 000 объектов

Домофонд ~ 1 258 701 объектов

Циан



Домофонд



Сбор данных

Парсинг данных из ЦИАНА и Домофонда

- Продажа жилой недвижимости
- Аренда жилой недвижимости
- Продажа коммерческой недвижимости
- Аренда коммерческой недвижимости

Сбор данных в виде картинок из Яндекс карт, Google карт и Викимании

Развертывание поисковой геолокационной системы на базе OpenStreetMaps и Postgis

Извлечение текстовых признаков из описаний

Использование данных ЦИАНА и Домофонда

Генерация признаков на основе ближайших соседей в квадрате 4 км:

- Средняя и медианная цена аренды и продажи
- Взвешенная по расстоянию цена аренды и продажи
- Количество объектов на продажу и аренду
- Среднее и медианное количество этажей в соседних домах
- Наличие рядом с объектом премиальной недвижимости

Геокодирование

Задача : Даны адреса, нужны координаты.

Проблема : Адреса часто грязные

Решение : Многоступенчатый геокодинг

1. Пробуем с Google API, если не находит, то
2. Пробуем с Yandex API, если не находит, то
3. Отсекаем последнюю часть адреса до разделителя (часто из за неё и были проблемы, например “офис 25”) , повторяем п.1

Вывод : подход позволил присвоить координаты всем адресам. Google в среднем справляется лучше, чем Yandex.

Использование данных OpenStreetMaps



Загружаем osm.pbf файл всей России:

<http://download.geofabrik.de/europe.html>

Загрузим данную информацию в postgres и настроим расширение

<http://www.postgis.net/>



Основные сущности

Nodes – определяет точки в пространстве

Ways – определяет границы площадей и их характеристики

Relations – определяет логические и географические отношения между элементами

Tag – состоит из двух элементов (K:V) – описывает свойства nodes, ways or relations

Пример: [Name : {tag1, tag2}] : Kari | {shop,shoes,name,Kari}

Поиск пяти ближайших объектов

```
select planet_osm_point.osm_id, planet_osm_nodes.id, tags
from planet_osm_point
join planet_osm_nodes
on planet_osm_point.osm_id = planet_osm_nodes.id
order by way <=> ST_Transform('SRID=4326;POINT(48.330585499999998 41.988205000000001)')::geometry, 3857)
limit 5;
```

```
osm_id | id | tags
-----+---+-----
3427420470 | 3427420470 | {railway, halt, name, "2423 км"}
1862202044 | 1862202044 | {place, village, name, Вавилово}
3924186313 | 3924186313 | {religion, muslim, amenity, place_of_worship}
3924186314 | 3924186314 | {religion, muslim, amenity, place_of_worship}
903947892 | 903947892 | {population, 4809, place, village, name, Хазар}
(5 rows)
```

Time: 0.859 ms

Поиск минимального расстояния в метрах на сфере

```
SELECT points.name, points.dist, nodes.tags FROM
|(SELECT name, osm_id, dist FROM
(SELECT name, osm_id, ST_Distance_Sphere(ST_Transform(way::geometry, 4326),
ST_SetSRID(ST_MakePoint(47.24982, 56.131380000000007), 4326)) AS dist
FROM planet_osm_point
WHERE ST_Within(ST_Transform(way::geometry, 4326),
ST_MakeEnvelope(47.217506827000101, 56.113372177580452, 47.282133172999899, 56.149387822419577, 4326)) = true
) AS points_dist
WHERE dist < 1000
) AS points
LEFT JOIN
(SELECT id, tags FROM planet_osm_nodes) AS nodes
ON points.osm_id = nodes.id;
```

Результат выполнения запроса

| name | dist | tags |
|--------------|---------------|---------------------------------------|
| Нижний вход | 999.074806705 | {highway,crossing} |
| | 939.913026595 | {barrier,gate} |
| | 939.237991209 | {foot,yes,bicycle,yes,barrier,gate} |
| | 979.02067602 | {highway,crossing} |
| | 975.57752925 | {highway,crossing} |
| | 819.636723304 | {barrier,lift_gate} |
| | 954.017963838 | {highway,crossing} |
| | 953.356623145 | {highway,crossing} |
| | 764.54831291 | {name,"Нижний вход",access,customers} |
| Kari | 685.670886 | {highway,crossing} |
| | 714.064856131 | {shop,shoes,name,Kari} |
| Warning! | 751.798918333 | {name,Warning!} |
| Кинокаскад | 736.246855347 | {amenity,cinema,name,Кинокаскад} |
| | 747.209155859 | {shop,jewelry} |
| Gloria Jeans | 694.643372154 | {shop,clothes,name,"Gloria Jeans"} |
| Fix Price | 717.212176978 | {shop,variety_store,name,"Fix Price"} |
| O'stin | 689.330430878 | {shop,clothes,name,O'stin} |

Time: 2863.077 ms

Использование геолокационной системы на базе Postgis

Генерация признаков на основе координат объекта

- Поиск объектов инфраструктуры находящихся в радиусе r
- Нахождение свойств этих объектов (школы, магазины, магистрали...)
- Поиск численности населения населенного пункта
- Поиск типа населенного пункта

Использование карт местности в виде картинок

Пусть есть 2 здания

1.Находится посреди леса

2.Находится на берегу реки в большом городе

Понятно, что цены таких объектов должны различаться. Теоретически, информация об окружении объектов должна быть отображена на картинке из Яндекс карт или его аналогов.

Для решения этой задачи мы обучили нейронную сеть на изображениях. Сеть пыталась научиться прогнозировать цену квадратного метра недвижимости смотря только на картинку с картой.

Алгоритмы распределенные по городам

| | Москва | Тула |
|---|----------|---------|
| Средняя стоимость 1 кв. метра магазина в рублях | 402 тыс. | 65 тыс. |
| Медиана стоимости 1 кв. метра магазина в рублях | 201 тыс. | 55 тыс. |

Вывод: рынки недвижимости городов сильно различны, следовательно, обучение регрессионной модели на единой выборке по всей стране сразу сильно увеличивает дисперсию.

Решение: для каждого города строить свою регрессионную модель.

Используемые алгоритмы

- Xgboost на данных сбербанка
- Xgboost на данных сбербанка + внешних данных
- Нейронная сеть на данных сбербанка
- Xgboost на каждом отдельном городе, если он достаточно большой на данных сбербанка
- Xgboost на каждом отдельном городе, если он достаточно большой на данных сбербанка + внешних данных

Настройка алгоритмов

- Hyperopt для бустинга
- `scipy.optimize.differential_evolution` для нейронной сети

Нереализованное 1. На разных рынках по разному торгуются.

Рыночная стоимость объекта оценки может резко отличаться от заявленной, однако выдача кредита (в идеале) производится по цене близкой к рыночной.

Поскольку в качестве обучения была предоставлена информация с сайтов объявлений, то, соответственно, модели обучаются предсказывать стоимость объявлений, а не рыночную стоимость.

Решение: необходимо вводить эмпирическую корректировку на торг для каждого города, чтобы сместить распределения цен в сторону рыночных.

Наш опыт создания сервиса для
конечного пользователя на основе
такой модели

Что нужно для создания такого решения

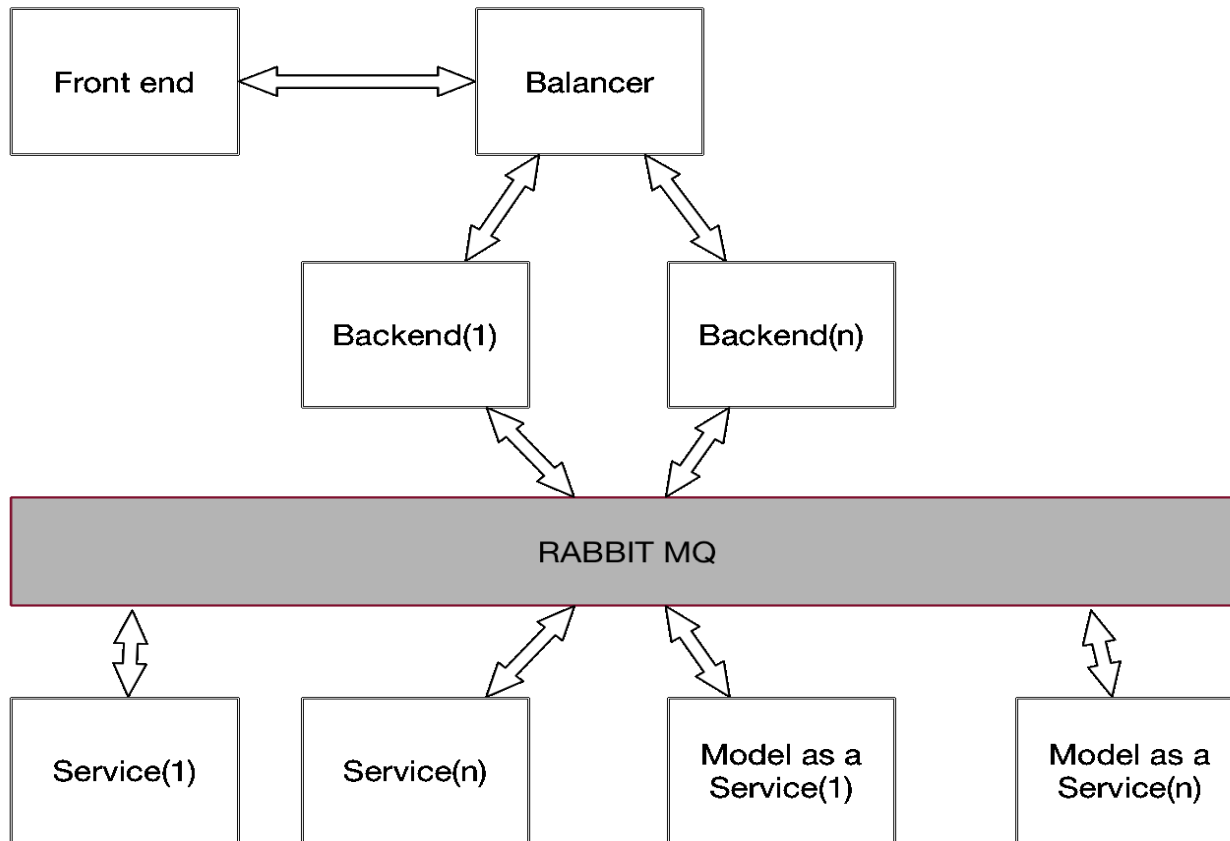
- Команда разработчиков (Frontend, Backend, Model)
- Возможность получить данные для обучения
- Сервер
- Желание и время

Более подробно

Требования к решению:

1. Горизонтальное масштабирование и возможность наращивания необходимые компонент;
2. Technology agnostic – возможность создание сервисов на любых технологиях;
3. Возможной резервирования различных компонент;
4. Отказоустойчивость;
5. Мониторинг всей системы и отдельных компонент.
6. Быстрый отклик (не более 500мс)

Структурная схема



Какую модель использовать?

Стоит использовать простую и интерпретируемую модель. Хорошим примером является линейная регрессия. Такая модель будет работать очень быстро, занимать мало памяти. Кроме этого, всегда можно будет посмотреть на веса модели и интерпретировать каждую фичу.

У НАС ТУТ НЕ КЕГЛ!



Мы решили заострить внимание только на качестве предсказаний алгоритма, а не на его интерпретируемости и скорости работы. Естественно, выбор очевиден. Кроме этого, мы не сэкономили ресурсы при генерации признаков. Весь упор только на качество алгоритма.

Итоговое количество фичей ~ 140000

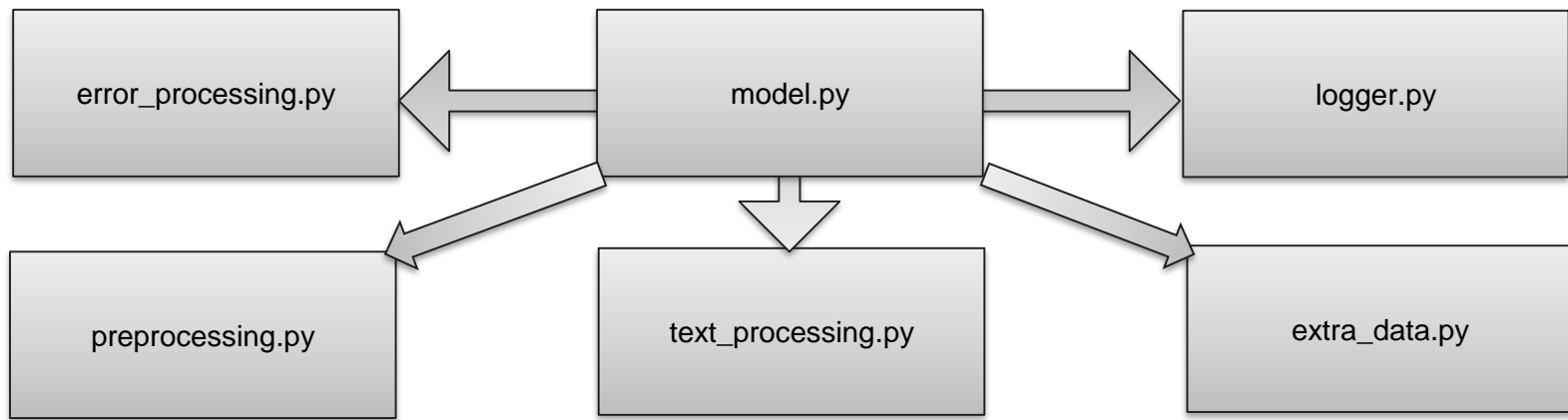
Итоговое количество деревьев ~ 3000 - 35000

Итоговое время отклика ~ 400 мс



Первая версия модели

Сначала были попытки внедрить модель без сложного процессинга данных. Такую модель можно внедрять без использования принципов ООП.



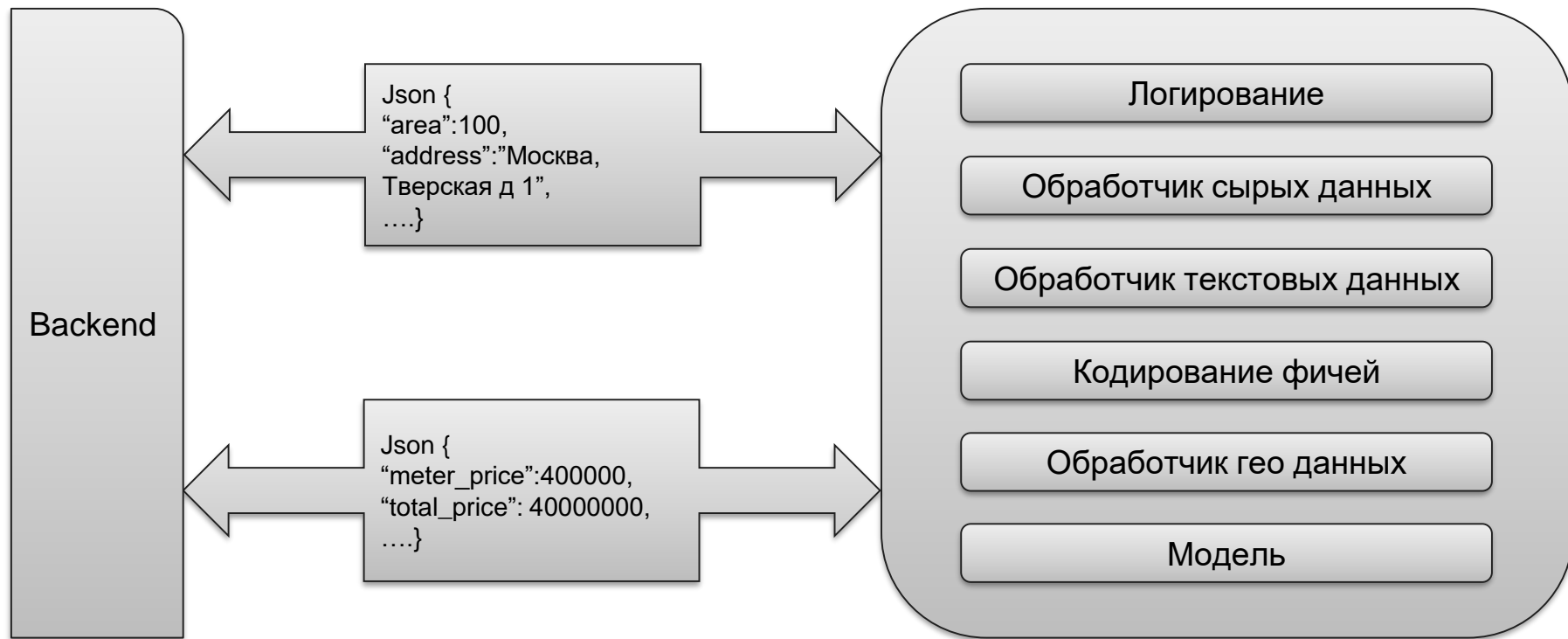
Проблема не ООП кода

Со временем количество кода растёт. Количество параметров скрипта тоже растёт. Логика кода тоже значительно усложняется.

Вот такие дикие вызовы функций могут получиться:

```
run_preprocessing(path_to_cian_raw_file, path_to_domofond_file,  
                  columns_to_encode, filter_values,  
                  to_pop_from_stop_words, to_append_to_stop_words,  
                  to_replace_patterns, min_freq, domofond_useless_columns,  
                  cian_useless_columns, path_to_preprocessed_file,  
                  categorical_cols, vectorizer_params_list,  
                  columns_with_text_preproc, not_for_ml_cols,  
                  geo_obj_folder, to_rename_columns_in_places,  
                  columns_to_get_from_places,  
                  preprocessed_files_folder, research_files_folder,  
                  columns_to_get_dummy_from_address)
```

Итоговая версия модели



Какие библиотеки использовать?

Numpy

Плюсы:

- Достаточно быстро
- Не так требователен к памяти, как Pandas

Минусы:

- Нечитаемый код
- Код разработки модели не используется. Для продакшена приходится писать новый

Pandas

Плюсы:

- Код разработки переиспользуется в продакшене
- Через неделю все-еще понятно, что вы имели в виду

Минусы:

- Медленнее Numpy
- Требователен к памяти

Про логирование

Почему нужно логировать каждый логический блок обработки данных в вашей модели?

1. Это очень сильно поможет на этапе интеграции с backend.
2. Это поможет выявить баги, о наличии которых вы даже не подозревали
3. Python иногда работает вопреки здравому смыслу.

Такой код на python 2 выполнится:

```
>>> True, False = False, True
>>> True
False
```

Спасибо за внимание!