МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Отчет для кафедры ММП об учатии в соревновании: *«Sustainable Industry: Rinse Over Run»* на платформе *DrivenData*

студента 204 учебной группы факультета ВМК МГУ
Шарыпова Р. Р.

Москва
2019

Содержание

Постановка задачи

В данном соревновании рассматривается работа CIP системы. Она применяется для очистки промышленного оборудования для производства продуктов питания и напитков, без разборки.

Каждый процесс имеет до пяти последовательных фаз:

1. Pre-rinse;
2. Caustic;
3. Intermediate rinse;
4. Acid;
5. Final rinse.

Целевым признаком является замутненность сточных вод возвращаемых во время последней фазы. Признак считается по формуле:

$$\sum(\max(0, return\_flow) * return\_turbidity),$$

где $return\_flow$ и $return\_turbidity$ соответствуют моментам для которых $target\_time\_period = True$.

В тестовой выборке у процессов убрана информация о *Final rince* фазе и о некоторых других:

- для 10% убраны все фазы начиная со 2-й;
- для 30% убраны все фазы начиная со 3-й;
- для 30% убраны две последнии фазы;
- для 30% убрана только последняя фаза.

Однако, так как по рецептуре у некоторых процессов отсутствуют определенные фазы, то:

- для 10% последняя известная фаза — *Pre-rinse*
- для 40% — *Caustic*
- для 23% — *Intermediate rinse*
- для 27% — *Acid*

Финальное решение

В финальном решении я сгенерировал для всех процессов метапризнаки, состоящие из индикатора того, проходил ли процесс в конкретном трубопроводе, рецепта и количества фаз по рецепту.

После я сгруппировал данные по фазе ('phase'). Далее, я рассмотрел столбцы:

- 'process_id',
- 'supply_flow',
- 'supply_pressure',
- 'return_temperature',
- 'return_conductivity',
- 'return_turbidity',
- 'return_flow',
- 'tank_level_pre_rinse',
- 'tank_level_caustic',
- 'tank_level_acid',
- 'tank_level_clean_water',
- 'tank_temperature_pre_rinse',
- 'tank_temperature_caustic',
- 'tank_temperature_acid',
- 'tank_concentration_caustic',
- 'tank_concentration_acid',

для каждой фазы, сгруппировал данные еще раз, но уже по процессам ('process_id') и применил к группам функции:

- 'min',

- 'max',
- 'mean',
- 'std' и
- среднее для 5 последних значений,

а так же вычислил размер каждой группы и использовал его как время протекания процесса, так как данные предоставляются через равные промежутки времени.

Таким образом я получил 4 новых датафрейма, каждый из которых содержал информацию о протекании конкретной фазы у всех процессов, у которых она была.

В следующем шаге я рассмотрел все возможные комбинации наличия/отсутствия фаз у процесса, кроме тривиальной.

№ комбинации	Фазы			
	'pre_rinse'	'caustic'	'intermediate_rinse'	'acid'
1	Есть	Нет	Нет	Нет
2	Нет	Есть	Нет	Нет
3	Есть	Есть	Нет	Нет
4	Нет	Нет	Есть	Нет
5	Есть	Нет	Есть	Нет
6	Нет	Есть	Есть	Нет
7	Есть	Есть	Есть	Нет
8	Нет	Нет	Нет	Есть
9	Есть	Нет	Нет	Есть
10	Нет	Есть	Нет	Есть
11	Есть	Есть	Нет	Есть
12	Нет	Нет	Есть	Есть
13	Есть	Нет	Есть	Есть
14	Нет	Есть	Есть	Есть
15	Есть	Есть	Есть	Есть

После, для тестового датафрейма для каждой комбинации я конкатенировал мета данные и датафреймы "присутствующих"фаз, выкинул все строки содержащие NaN, получив датафрейм *ts_features*. Я также вычислил 'process_id' процессов для которых есть информация об "отсутствующих"фазах и убрал из *ts_features* строки, 'process_id' которых входил в этот набор. Таким образом в *ts_features* содержалась информация о "присутствующих"фазах для процессов строго соответствующих комбинации, т.е. для них имелась информация о "присутствующих"фазах и не было информации об "отсутствующих".

В итоге я получил 15 различных датафреймов, 7 из которых были пустыми. Оставшиеся 8 соответствовали комбинациям:

№ комбинации	Фазы			
	'pre_rinse'	'caustic'	'intermediate_rinse'	'acid'
1	Есть	Нет	Нет	Нет
2	Нет	Есть	Нет	Нет
3	Есть	Есть	Нет	Нет
6	Нет	Есть	Есть	Нет
7	Есть	Есть	Есть	Нет
8	Нет	Нет	Нет	Есть
14	Нет	Есть	Есть	Есть
15	Есть	Есть	Есть	Есть

После я проделал то же самое для тренировочного датафрейма, за исключением отбрасывания процессов с "отсутствующими"фазами.

Таким образом я разбил задачу на 8 подзадач, для каждой из которых я обучал отдельную модель.

Так, за счет большого количества признаков и отсутствия необходимости заполнять пропуски, я смог добиться достаточно неплохой точности.

Для обучения я использовал *GradientBoostingRegressor* из библиотеки *scikit-learn*, с *loss='quantile'*. Оставшиеся параметры были подобраны с помощью *validation_curve*.

Тот факт, что модели никак не зависят друг от друга, позволил мне вести их настройку параллельно на 2 ноутбуках. Так же ради оптимизации процесс преобразования начальных данных в описанные ранее датафреймы, был выделен в отдельную программу, дабы при настройке не загружать каждый раз по 3Гб данных в ОЗУ.

Проделанные эксперименты, а так же эксперименты на которые не хватило времени

Изначально я разделил данные не по комбинациям наличия/отсутствия фаз, а по тому какой фазой ограничиваются данные. Т. е. я вычислил все те же 4 датафрейма для фаз и конкатенировал первые n из них, заполняя NaN нулями.

Так же до применения *GradientBoostingRegressor* я использовал *RandomForestRegressor*. Еще я эксперементировал с параметром *loss* у *GradientBoostingRegressor*, при *loss='lad'* алгоритм работл очень быстро и с приемлемой точностью — 0.3279, что на момент загрузки резульнатов соответствовало 32 месту.

Помимо этого без отправки я эусперементировал с *SVR*, *KNeighborsRegressor* и *MLPRegressor*, однако их результаты были не достаточно хороши. Так же я пробовал использовать стекинг с *SVR* в качестве базового алгоритма, но опять же результаты были не очень.

Будь у меня больше времени я бы попробовал использовать квантильную регрессию в *LightGBM*, поскольку данная модель более гибкая в плане настройки.

Итог

Итог посылок:

Место в момент посылки	Score	Private Score	Timestamp	Важные изменения
—	2.0161	2.2958	2019-02-16 15:37:20 UTC	—
—	1.7990	2.0598	2019-02-18 20:30:31 UTC	—
—	1.1374	1.1233	2019-02-23 02:37:57 UTC	Разбиение на 4 типа
135	1.0736	1.1216	2019-02-24 23:12:09 UTC	—
49	0.4079	0.4236	2019-02-25 15:56:43 UTC	Разбиение на 8 типов
32	0.3279	0.3185	2019-02-26 20:08:44 UTC	GBR, loss='lad'
27	0.3137	0.3086	2019-02-27 22:14:22 UTC	GBR, loss='quantile'
27	0.3141	0.3097	2019-03-01 23:42:41 UTC	—

Моя лучшая посылка имеет точность — 0.3086. По итогу соревнования я занял 22 место. Ник на сайте: zelgades