

Итоговая работа

Тема: Использование машинного обучения для прогнозирования отправления железнодорожных вагонов в ремонт.

Введение:

Отправка вагона в плановый ремонт может происходить по разным причинам, как по регламенту(срок/пробег), так и из-за накопления мелких дефектов:

- было много текущих ремонтов;
- не было вариантов на погрузку и т.д.

Этих причин много, и все они влияют на возможность осуществления ремонта.

Вагон отправляют в ремонт после получения уведомления об их неисправности. К сожалению, текущий процесс не позволяет распределять нагрузку на ремонт депо, управлять последней заявкой на погрузку пред ремонтом и многое другое.

Цель: снизить нагрузку на железнодорожную систему путем своевременного обслуживания вагонов.

Задача: создать модель прогнозирования даты отправления вагона в плановый ремонт.

Постановка задачи:

1. найти закономерности и оценить значимые признаки;
2. спрогнозировать, что вагон отправится в ПР в течение месяца;
3. спрогнозировать, что вагон отправится в ПР в течение 10 дней.

Обзор моделей:

Наивная модель построена на правилах с использованием минимального набора данных, без применения машинного обучения.

Реальный процесс выглядит следующим образом - в начале месяца берется срез по парку по всем вагонам, за ремонт которых несёт ответственность ПГК. Для выбранных вагонов требуется установить, какие из них будут отремонтированы в текущем месяце. Данная информация помогает планировать нагрузку на вагоноремонтное предприятие (ВРП). Вторая модель определяет критичные вагоны, которые будут отправлены в ремонт

в первую очередь (в ближайшие 10 дней). Это помогает фокусировать внимание диспетчеров.

Основными критериями, по которым вагон отправляется в плановый ремонт - является его остаточный пробег и срок до планового ремонта.

В регламентах РЖД используется следующее правило - если ресурс по пробегу не превышает 500 км, срок службы не превышает 500 дней, число текущих ремонтов больше 5 и/или плановый ремонт должен наступить через 15 дней (или меньше), то вагон может ехать только на ВРП.

Из этого регламента вытекают две особенности:

1. Диспетчер старается отправить вагон раньше положенных значений. Это позволяет выбрать предприятия, на которых ремонтироваться дешевле, а не ближайшее.

2. Компания-оператор может выбирать какому из нормативов нужно следовать - ремонтировать вагон по сроку, или по пробегу, или по обоим критериям сразу. Поэтому встречаются вагоны, у которых пробег может не отслеживаться.

Вагон может быть отправлен в плановый ремонт и раньше положенного. На это может влиять, например, история грузовых операций и количество текущих(мелких) ремонтов.

Описание процесса решения (модель, тест, тренировка, результат):

Изначально производится импорт необходимых библиотек (pandas, numpy, os) и загрузка данных.

▼ Импорт библиотек

```
✓ 19 сек. [1] from google.colab import drive
      drive.mount('/content/drive')
```

Mounted at /content/drive

```
✓ 2 сек. ▶ import os # библиотеки работы с папками и системой

# и другие уже известные нам библиотеки
import pandas as pd
import numpy as np

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
from sklearn.preprocessing import MinMaxScaler
```

▼ Загрузка данных

```
54 ✓ !unrar x /content/drive/MyDrive/DataWagon/train_2.rar /content/drive/MyDrive/DataWagon/train_2/
CSK.

UNRAR 6.11 beta 1 freeware      Copyright (c) 1993-2022 Alexander Roshal

Extracting from /content/drive/MyDrive/DataWagon/train_2.rar

Would you like to replace the existing file /content/drive/MyDrive/DataWagon/train_2/test/dislok_wagons.parquet
11392957 bytes, modified on 2023-11-08 12:28
with a new one
11392957 bytes, modified on 2023-11-08 12:28

[Y]es, [N]o, [A]ll, n[E]ver, [R]ename, [Q]uit q

Program aborted

[ ] !cp -r /content/train/content/имя_папки /content/drive/MyDrive/DataWagon

cp: cannot stat '/content/train/content/имя_папки': No such file or directory

[ ] !ls /content/drive/MyDrive/DataWagon/train

dislok_wagons.parquet  metrics_f1.py      stations.parquet      tr_rems.parquet
freight_info.parquet  prediction          target                wagons_probeg_ownership.parquet
kti_izm.parquet       pr_rems.parquet    task2_base_model.ipynb wag_params.parquet

[5] path = '/content/drive/MyDrive/DataWagon/train'

23 ✓ # список вагонов с остаточным пробегом на момент прогноза
CSK. wag_prob = pd.read_parquet(path + '/wagons_probeg_ownership.parquet').convert_dtypes()
# данные по дислокации
dislok = pd.read_parquet(path + '/dislok_wagons.parquet').convert_dtypes()
# параметры вагона
wag_param = pd.read_parquet(path + '/wag_params.parquet').convert_dtypes()
# данные по текущим ремонтам
pr_rem = pd.read_parquet(path + '/pr_rems.parquet').convert_dtypes()
# текущие ремонты вагонов
tr_rem = pd.read_parquet(path + '/tr_rems.parquet').convert_dtypes()
# данные по КТИ
kti_izm = pd.read_parquet(path + '/kti_izm.parquet').convert_dtypes()
# справочник грузов
freight_info = pd.read_parquet(path + '/freight_info.parquet').convert_dtypes()
# справочник станций
stations = pd.read_parquet(path + '/stations.parquet').convert_dtypes()
# таргет по прогнозу выезда вагонов в ПР на месяц и на 10 дней
target = pd.read_csv('/content/drive/MyDrive/DataWagon/train_2/test/target/y_test.csv').convert_dtypes()
```

Исходные данные содержат 9 таблиц wag_prob, dislok, wag_param, pr_rem, tr_rem, kti_izm, freight_info, stations, target. Таблицы включают в себя следующее:

- Таблица wag_prob. Список вагонов, по которым известен пробег и тип владения на дату среза: Дата среза, Номер вагона, Остаточный пробег, Вид управления по договору (1-собственные, 2-лизинговые, 11-сервисные), Тип РПС, Состояние в реестре (1-слежение, 3-включение, 4-готовится к искл.), Тип собственности (1-собственные, 2-принятые в аренду, 4-привлеченный парк, 6-сервис), Месяц.

✓ [7] wag_prob

	repdate	wagnum	ost_prob	manage_type	rod_id	reestr_state	ownership_type	month
0	2022-08-01	33361	7541	0	1	1	0	8
1	2022-08-02	33361	7243	0	1	1	0	8
2	2022-08-03	33361	6990	0	1	1	0	8
3	2022-08-04	33361	6347	0	1	1	0	8
4	2022-08-05	33361	6027	0	1	1	0	8
...
9249584	2022-12-20	33350	35062	0	1	0	0	12
9249585	2022-12-21	33350	35062	0	1	0	0	12
9249586	2022-12-22	33350	35062	0	1	0	0	12
9249587	2022-12-23	33350	35062	0	1	0	0	12
9249588	2022-12-24	33350	35062	0	1	0	0	12

6249857 rows × 8 columns

- Таблица dislok. Информация по дислокации: Дата отчета, Номер вагона, Дата последнего капитального ремонта, Дата последнего деповского ремонта, Код предстоящего планового ремонта, Дата предстоящего планового ремонта, Код дороги дислокации, Ид. станции назначения, Код дороги назначения, Индекс станции отправления, Код дороги отправления, Остаточный пробег, Грузеный/порожний, Код груза, Ид. последнего груза, Расстояние между станцией отправления и прибытия.

dislok

	plan_date	wagnum	date_kap	date_dep	kod_vrab	date_pl_rem	id_road_disl	st_id_dest	id_road_dest	st_id_send	id_road_send	ost_prob	isload
0	2022-08-01	1071	NaT	2020-02-16	0	2023-02-16	42	22	42	11470	42	36899	0
1	2022-08-02	1071	NaT	2020-02-16	0	2023-02-16	42	11663	0	11470	42	36899	1
2	2022-08-03	1071	NaT	2020-02-16	0	2023-02-16	42	11663	0	11470	42	36711	1
3	2022-08-04	1071	NaT	2020-02-16	0	2023-02-16	42	11663	0	11470	42	36711	1
4	2022-08-05	1071	NaT	2020-02-16	0	2023-02-16	42	11663	0	11470	42	36711	1
...
1052920	2023-01-27	9554	2009-10-13	2021-08-23	1	2024-08-23	37	22	37	12647	37	87693	0
1052921	2023-01-28	9554	2009-10-13	2021-08-23	1	2024-08-23	37	22	37	12647	37	87693	0
1052922	2023-01-29	9554	2009-10-13	2021-08-23	1	2024-08-23	37	22	37	12647	37	87693	0
1052923	2023-01-30	9554	2009-10-13	2021-08-23	1	2024-08-23	37	22	37	12647	37	87693	0
1052924	2023-01-31	9554	2009-10-13	2021-08-23	1	2024-08-23	37	22	37	12647	37	87693	0

6250933 rows × 16 columns

- Таблица wag_param. Данные по характеристикам вагона: Номер вагона, Модель вагона, Тип РПС, Грузоподъемность (в центнерах), Предельная грузоподъемность, Объем кузова, Масса тары в центнерах, Дата постройки, Дата окончания срока службы, Завод

постройки, Норма пробега после ДР в тыс. км, Норма пробега после КР в тыс. км, Кузов, Код модели тележки, Тип тормозов, Тип воздухораспределителя, Межремонтный норматив пробега ($\neq 0$, если вагон на пробеге), Признак передачи вагона в аренду (1-собст, 2-арендованный, 3-инвентарный).

wagonnum	model	rod_id	груз	cnsi_gruz_capacity	cnsi_volumek	tara	date_build	srok_sl	zavod_build	date_iskl	cnsi_probeg_dr	cnsi_probeg_kr	k
3218	26318	12-600-04	1	682	682	85.0	240	1992-12-25	2022-04-27	5	2023-02-16	160	160
19128	28344	12-132	1	700	700	88.0	240	2003-08-12	2024-12-24	0	2022-12-14	110	160
21526	8099	11-286	0	670	670	138.0	270	1995-08-31	2027-10-01	1	NaT	110	160
32353	33350	12-9850-02	1	750	750	90.0	248	2014-10-27	2047-02-05	19	NaT	250	500
81	5308	11-276	0	680	680	122.0	260	1995-09-17	2027-09-28	1	NaT	110	160
...
33703	18766	11-280	0	680	680	138.0	259	2013-01-07	2046-03-20	1	NaT	110	160
33704	18769	11-280	0	680	680	138.0	259	2013-01-18	2046-02-14	1	NaT	110	160
33705	18899	11-280	0	680	680	138.0	260	2013-07-24	2044-03-03	1	NaT	110	160
33706	18912	11-280	0	680	680	138.0	259	2015-02-28	2044-08-04	1	NaT	110	160
33707	18914	11-280	0	680	680	138.0	259	2013-06-16	2046-07-02	1	NaT	110	160

- Таблица pr_rem. Данные по плановым ремонтам: Номер вагона, Месяц ремонта, Тип РПС, модель, Дорога выбытия в ПР, Дорога прибытия на ВРП, Код ремонта, Станция выгрузки, код станции ВРП, Расстояние со станции до ВРП, Месяц.

pr_rem

	wagnum	rem_month	rod_id	model	road_id_send	road_id_rem	kod_vrab	st_id_send	st_id_rem	distance	month
0	15000	2022-08-05	1	12-132	3	3	0	493	493	37	8
2	25485	2022-08-21	1	12-132-03	13	13	0	6152	4793	360	8
3	25944	2022-08-09	1	12-132-03	13	13	1	6152	4793	322	8
4	24080	2022-08-13	1	12-296-01	13	13	0	6152	4793	308	8
5	28612	2022-08-09	1	12-132	13	13	0	6152	4793	373	8
...
1578	11193	2023-01-30	1	12-132	12	12	0	12607	12607	885	1
509	24703	2023-01-24	1	12-9780	6	6	1	4025	4025	890	1
511	26437	2023-01-29	1	12-9780	6	6	1	4025	4025	892	1
711	29345	2023-01-15	1	12-132	13	13	1	6152	5960	397	1
1515	15740	2023-01-07	1	12-132	23	12	0	12610	11915	4101	1

10441 rows × 11 columns

- Таблица tr_rem. Данные по текущим ремонтам вагона: Номер вагона, Дата ремонта, Код работ, Код неисправности 1 (Заполняется по с.5353), Код неисправности 2 (Заполняется по с.5353), Код неисправности 3 (Заполняется по с.5353), Код модернизации 1 (Заполняется по с.5354), Код модернизации 2 (Заполняется по с.5354), Код модернизации 3 (Заполняется по с.5354).

по с.5354), Код модернизации 4 (Заполняется по с.5354), Код модернизации 5 (Заполняется по с.5354), Код модернизации 6 (Заполняется по с.5354), Код модернизации 7 (Заполняется по с.5354), ID дороги перевода вагона в неисправные(nsi_db.railway.RW_ID), Пробег вагона в груженом состоянии, Код станции перевода вагона в неисправные (nsi_db.station.ST_CODE).

[11] tr_rem

	wagnum	rem_month	kod_vrab	neis1_kod	neis2_kod	neis3_kod	mod1_kod	mod2_kod	mod3_kod	mod4_kod	mod5_kod	mod6_kod	mod7_kod	road_id_send	gr_
0	29938	2022-08-01	3	0	98	54	7	4	2	0	0	0	0	38	
1	29938	2022-08-01	3	14	98	54	7	4	2	0	0	0	0	38	
2	29852	2022-08-01	2	0	98	54	7	4	2	0	0	0	0	28	
3	29852	2022-08-01	2	36	98	54	7	4	2	0	0	0	0	28	
4	13674	2022-08-01	2	95	98	54	7	4	2	0	0	0	0	28	
...
7695	31868	2023-01-01	3	14	98	54	7	4	2	0	0	0	0	42	
7696	22921	2023-01-01	5	0	98	54	7	4	2	0	0	0	0	28	
7697	22921	2023-01-01	3	34	95	54	7	4	2	0	0	0	0	28	
7698	4978	2023-01-01	3	131	68	54	7	4	2	0	0	0	0	3	
7699	16226	2023-01-01	3	6	98	54	7	4	2	0	0	0	0	43	

48652 rows x 17 columns

- Таблица kti_izm. Данные по КТИ: Номер вагона, Дата и время измерения КТИ, Пробег общий, Толщина гребня 1 ось слева, Толщина гребня 1 ось справа, Толщина гребня 2 ось слева, Толщина гребня 2 ось справа, Толщина гребня 3 ось слева, Толщина гребня 3 ось справа, Толщина гребня 4 ось слева, Толщина гребня 4 ось справа, Толщина обода 1 ось слева, Толщина обода 1 ось справа, Толщина обода 2 ось слева, Толщина обода 2 ось справа, Толщина обода 3 ось слева, Толщина обода 3 ось справа, Толщина обода 4 ось слева, Толщина обода 4 ось справа.

[12] kti_izm

	wagnum	operation_date_dttm	mileage_all	axl1_l_w_flange	axl1_r_w_flange	axl2_l_w_flange	axl2_r_w_flange	axl3_l_w_flange	axl3_r_w_flange	axl4_l_w_flange	axl4_r_w_flange
0	325	2022-08-01	112091	30.5	31.0	26.7	27.4	28.8	28.6		
1	325	2022-08-03	112471	30.4	31.4	27.5	27.6	28.9	29.0		
2	325	2022-08-05	113938	30.0	31.1	27.1	27.9	28.5	28.9		
3	325	2022-09-15	121071	30.5	31.4	26.4	27.3	28.9	28.4		
4	325	2022-09-09	117341	30.7	31.5	26.9	27.1	28.9	28.5		
...		
194723	28221	2023-01-31	27116	28.4	31.4	29.0	29.2	27.0	28.3		
194752	29023	2023-01-31	1911	29.7	28.6	29.6	30.2	27.9	30.3		
194780	33124	2023-01-31	118504	28.7	29.3	32.8	28.2	29.7	29.4		
194825	28089	2023-01-31	22183	28.4	28.3	29.3	28.6	28.8	28.0		
194859	28090	2023-01-31	158013	28.7	26.6	28.0	28.7	29.8	30.7		

67967 rows x 19 columns

- Таблица freight_info. Справочник грузов: Ид груза, Класс груза, Признак скоропортящегося груза, Признак погрузки навалом, Признак погрузки насыпи, Признак погрузки наливом, Перевозится в открытом подвижном составе, Груз требует сопровождения, Признак смерзающего груза.

freight_info

	fr_id	fr_class	skoroport	naval	nasip	naliv	openvavons	soprovod	smerz
0	4989	2	0	0	0	0	0	0	0
1	4990	2	0	0	0	0	0	0	0
2	4991	0	0	0	0	0	0	0	0
3	4992	2	0	0	0	0	0	0	0
4	4993	2	0	0	0	0	0	0	0
...
5074	989	0	0	1	1	0	0	0	0
5075	990	0	0	1	1	0	0	0	0
5076	991	0	0	1	1	0	0	0	0
5077	992	0	0	0	0	0	0	0	0
5078	993	0	0	1	1	0	0	0	0

5079 rows x 9 columns

- Таблица stations. Справочник станции: Ид станции, Ид дороги, Признак граничной станции, Признак перевалки на море, Признак перевалки на реку, Признак перевалки на автотранспорт, Признак перевалки на паромную переправу, Признак станции, открытой для грузовой работы, Признак опорной станции.

stations

	st_id	road_id	st_border_sign	st_sea_sign	st_river_sign	st_car_sign	st_ferry_sign	st_freigh_sign	opor_station_sign
0	13069	47	0	0	0	0	0	0	0
1	9353	18	0	0	0	0	0	0	0
2	9973	19	0	0	0	0	0	1	0
3	13745	6	0	0	0	0	0	0	0
4	14554	15	0	0	0	0	0	0	0
...
16443	5235	12	0	0	0	0	0	0	0
16444	14918	18	0	0	0	0	0	0	0
16445	15937	21	0	0	0	0	0	0	0
16446	12866	24	0	0	0	0	0	1	0
16447	4146	7	0	0	0	0	0	0	0

148032 rows x 9 columns

- Таблица target: Номер вагона, Месяц ремонта, Ремонт в течение месяца (1-был ремонт, 0-не было ремонта), Ремонт в первые 10 дней (1-был ремонт, 0-не было ремонта). Данная таблица содержит две целевые переменные, так как стоит задача независимо друг от друга определить выбытие вагона в ремонт в первые 10 дней и в течении месяца.

target

	wagnum	month	target_month	target_day
0	33361	2023-02-01	0	0
1	33364	2023-02-01	0	0
2	33366	2023-02-01	0	0
3	33358	2023-02-01	0	0
4	33349	2023-02-01	0	0
...
33703	17621	2023-02-01	0	0
33704	25045	2023-02-01	0	0
33705	27156	2023-02-01	0	0
33706	21361	2023-02-01	0	0
33707	8061	2023-02-01	0	0

33708 rows x 4 columns

Подготовка данных:

Перед обучением модели необходимо подготовить данные. Для этого:

- Произвелась оценка среднесуточного пробега по пробегу вагона, на тот случай если данных по нормативу нет.

Подготовка данных

```
[16] target_data= pd.to_datetime('2023-01-01')
pred_data = pd.to_datetime('2023-02-01')

[17] wag_prob = wag_prob[(wag_prob.repdata == pd.to_datetime('2023-01-31')) | (wag_prob.repdata == wag_prob.repdata.min())]

[18] # оценим среднесуточный пробег из данных по пробегу вагона, на тот случай, если данных по нормативу нет
wag_prob = wag_prob.groupby('wagnum', as_index = False).agg(['repdate': ['max', 'min'], 'ost_prob': ['max', 'min']],)#.droplevel(1)
wag_prob.columns = [head+'_' + name
                    if head!='wagnum'
                    else head
                    for head, name in wag_prob.columns ]

wag_prob['diff_days'] = wag_prob.repdata_max - wag_prob.repdata_min
wag_prob['mean_run'] = (wag_prob.ost_prob_max - wag_prob.ost_prob_min )/ wag_prob.diff_days.dt.days
wag_prob = wag_prob[wag_prob.repdata == wag_prob.repdata.max()][['repdate', 'wagnum', 'ost_prob', 'manage_type', 'rod_id', 'reestr_state', 'month']]
wag_prob = wag_prob.merge(wag_prob[['wagnum', 'mean_run']])
```

- Удалены дубликаты номеров вагонов из всех таблиц.

```
[19] #удаляем дубли номеров вагонов из всех таблиц
wag_prob = wag_prob.drop_duplicates(subset='wagnum', keep='last')
dislok = dislok.drop_duplicates(subset='wagnum', keep='last')
wag_param = wag_param.drop_duplicates(subset='wagnum', keep='last')
pr_rem = pr_rem.drop_duplicates(subset='wagnum', keep='last')
kti_izm = kti_izm.drop_duplicates(subset='wagnum', keep='last')
```

- Из таблиц по фильтру отобраны все значения на дату обучения.

```
wag_prob['month'] = wag_prob['repdate'].apply(lambda x: str(pd.to_datetime(x, format='%Y-%m-%d').year) + '-' + str(pd.to_datetime(x, format='%Y-%m-%d').month))
dislok['month'] = dislok['plan_date'].apply(lambda x: str(pd.to_datetime(x, format='%Y-%m-%d').year) + '-' + str(pd.to_datetime(x, format='%Y-%m-%d').month))
```


- Посчитано сколько текущих ремонтов было за прошедший период.

```
✓ [23] # посчитаем сколько текущих ремонтов было за прошедший период
0 OK tr_rem = tr_rem.groupby('wagonum', as_index=False).kod_vrab.count()
```

- Выбраны необходимые колонки из таблиц wag_prod_train, dislok_train, wag_param_pred, pr_rem_train, kti_izm_train, tr_term_train и объединены в один общий датафрейм для обучения модели.

```
✓ [24] #выбор необходимых колонок из таблиц
0 OK wag_prod_train = wag_prob[wag_prob.month==(str(pd.to_datetime(target_data, format='%Y-%m-%d').year)+'-'+str(pd.to_datetime(target_data, format='%Y-%m-%d').month)+'-'+str(pd.to_datetime(target_data, format='%Y-%m-%d').day))
dislok_train = dislok[dislok.month==(str(pd.to_datetime(target_data, format='%Y-%m-%d').year)+'-'+str(pd.to_datetime(target_data, format='%Y-%m-%d').month)+'-'+str(pd.to_datetime(target_data, format='%Y-%m-%d').day))
wag_param_pred = wag_param[['wagonum', 'gruz', 'cnsi_gruz_capacity', 'cnsi_volumek', 'tara', 'date_build', 'srok_sl', 'zavod_build', 'date_iskl', 'cnsi_p
pr_rem_train = pr_rem[['wagonum', 'road_id_send', 'road_id_rem', 'st_id_send', 'st_id_rem', 'distance']]
kti_izm_train = kti_izm[['wagonum', 'mileage_all', 'axl1_l_w_flange', 'axl1_r_w_flange', 'axl2_l_w_flange', 'axl2_r_w_flange', 'axl3_l_w_flange', 'axl3_r_w
axl3_l_w_rim', 'axl3_r_w_rim', 'axl4_l_w_rim', 'axl4_r_w_rim']]
tr_term_train = tr_rem

✓ [25] #объединение всех таблиц в один датафрейм для обучения
0 OK df_train = train[['wagonum', 'target_month', 'target_day']].merge(wag_prod_train, on='wagonum', how='left')\
.merge(dislok_train, how='left')\
.merge(wag_param_pred, how='left')\
.merge(pr_rem_train, how='left')\
.merge(tr_term_train, how='left')\
.merge(kti_izm_train, how='left')
```

- Отсеяны вагоны, которые проехали 0 км в день, т.к. по ним невозможно обучить модель.

```
✓ [26] df_train[['cnsi_probeg_dr', 'cnsi_probeg_kr', 'mean_run']] = df_train[['cnsi_probeg_dr', 'cnsi_probeg_kr', 'mean_run']].fillna(0)#заполняем нулями
0 OK df_train['day_run'] = df_train.apply(lambda x : [ val for val in [x.cnsi_probeg_kr, x.cnsi_probeg_dr, x.mean_run] if val != 0], axis = 1 )#заменяем нул
df_train['day_run'] = df_train.apply(lambda x : np.mean(x.day_run) if len(x.day_run)> 0 else 0, axis = 1 )#отсеиваем вагоны, которые проехали меньше 0 км
```

- Также рассмотрено количество дней, которое осталось до истечения срока службы вагона, дней до ближайшего ПР и какой остаточный ресурс будет на момент окончания месяца.

```
✓ [29] # определим, сколько дней осталось до истечения срока службы
0 OK df_train['date_diff_srk_sl'] = df_train['srok_sl'] - df_train['current_date']
```

```
✓ [30] # определим, сколько дней осталось до ближайшего ПР
0 OK df_train['date_diff_pl_rem'] = df_train['date_pl_rem'] - df_train['current_date']
```

```
✓ [31] # определим, какой остаточный ресурс будет на момент окончания месяца
0 OK df_train['prob_end_month'] = df_train['ost_prob'] - df_train['day_run']* 30
```

- Найдены параметры, которые больше всего коррелируют с target_month (будет ли в этом месяце ремонт).

```
#СМОТРИМ корреляции
max_corr = df_train.corr()['target_month'].apply(lambda x:abs(x)).sort_values(ascending=False,axis=0)
max_corr.iloc[:50]
```

```
<ipython-input-32-08fa8b8a8f51>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default
max_corr = df_train.corr()['target_month'].apply(lambda x:abs(x)).sort_values(ascending=False,axis=0)
target_month      1.000000
target_day        0.582239
mileage_all       0.312712
prob_end_month    0.274162
ost_prob          0.271934
day_run           0.127527
mean_run         0.124532
axl1_r_w_flange   0.099007
axl2_l_w_flange   0.089090
axl4_l_w_flange   0.085205
axl3_r_w_flange   0.082267
axl2_r_w_flange   0.080600
axl4_r_w_flange   0.080570
axl3_l_w_flange   0.070061
axl1_l_w_flange   0.069558
wagnum            0.050047
cnsi_probeg_dr    0.050458
telega           0.047280
cnsi_probeg_kr    0.045641
kuzov            0.036369
tara             0.035516
rod_id           0.034066
kod_vrab         0.030483
cnsi_volumek     0.029897
norma_km         0.027939
zavod_build      0.018505
tippogl          0.013471
isload           0.009228
axl4_l_w_rim     0.008525
axl4_r_w_rim     0.008262
axl2_l_w_rim     0.007614
cnsi_gruz_capacity 0.006952
gruz             0.006952
axl2_r_w_rim     0.006822
axl3_r_w_rim     0.004735
axl1_l_w_rim     0.002883
tormoz           0.002289
distance         0.001744
axl3_l_w_rim     0.001706
axl1_r_w_rim     0.000353
road_id_send     NaN
road_id_rem      NaN
st_id_send       NaN
st_id_rem        NaN
Name: target_month, dtype: float64
```

- Удалены нулевые значения по столбцам. Взяты 24 лучших параметров, которые коррелируют с target_month.

```
✓ 0 [36] #заменяем нан на 0 во всей таблице  
OCC df_train_ = df_train_.fillna(0)
```

```
✓ 0 [37] #по столбцу убираем нулевые значения  
OCC mileage_mean = df_train[df_train['mileage_all'] != 0]['mileage_all'].mean()  
df_train['mileage_all'] = df_train['mileage_all'].apply(lambda x: mileage_mean if x == 0 else x)
```

```
✓ 0 1 #по столбцу убираем нулевые значения  
OCC axl1_l_w_flange_mean = df_train[df_train['axl1_l_w_flange'] != 0]['axl1_l_w_flange'].mean()  
df_train['axl1_l_w_flange'] = df_train['axl1_l_w_flange'].apply(lambda x: axl1_l_w_flange_mean if x == 0 else x)  
  
axl1_r_w_flange_mean = df_train[df_train['axl1_r_w_flange'] != 0]['axl1_r_w_flange'].mean()  
df_train['axl1_r_w_flange'] = df_train['axl1_r_w_flange'].apply(lambda x: axl1_r_w_flange_mean if x == 0 else x)  
  
axl2_l_w_flange_mean = df_train[df_train['axl2_l_w_flange'] != 0]['axl2_l_w_flange'].mean()  
df_train['axl2_l_w_flange'] = df_train['axl2_l_w_flange'].apply(lambda x: axl2_l_w_flange_mean if x == 0 else x)  
  
axl2_r_w_flange_mean = df_train[df_train['axl2_r_w_flange'] != 0]['axl2_r_w_flange'].mean()  
df_train['axl2_r_w_flange'] = df_train['axl2_r_w_flange'].apply(lambda x: axl2_r_w_flange_mean if x == 0 else x)  
  
axl3_l_w_flange_mean = df_train[df_train['axl3_l_w_flange'] != 0]['axl3_l_w_flange'].mean()  
df_train['axl3_l_w_flange'] = df_train['axl3_l_w_flange'].apply(lambda x: axl3_l_w_flange_mean if x == 0 else x)  
  
axl3_r_w_flange_mean = df_train[df_train['axl3_r_w_flange'] != 0]['axl3_r_w_flange'].mean()  
df_train['axl3_r_w_flange'] = df_train['axl3_r_w_flange'].apply(lambda x: axl3_r_w_flange_mean if x == 0 else x)  
  
axl4_l_w_flange_mean = df_train[df_train['axl4_l_w_flange'] != 0]['axl4_l_w_flange'].mean()  
df_train['axl4_l_w_flange'] = df_train['axl4_l_w_flange'].apply(lambda x: axl4_l_w_flange_mean if x == 0 else x)  
  
axl4_r_w_flange_mean = df_train[df_train['axl4_r_w_flange'] != 0]['axl4_r_w_flange'].mean()  
df_train['axl4_r_w_flange'] = df_train['axl4_r_w_flange'].apply(lambda x: axl4_r_w_flange_mean if x == 0 else x)  
  
axl1_l_w_rim_mean = df_train[df_train['axl1_l_w_rim'] != 0]['axl1_l_w_rim'].mean()  
df_train['axl1_l_w_rim'] = df_train['axl1_l_w_rim'].apply(lambda x: axl1_l_w_rim_mean if x == 0 else x)  
  
axl1_r_w_rim_mean = df_train[df_train['axl1_r_w_rim'] != 0]['axl1_r_w_rim'].mean()  
df_train['axl1_r_w_rim'] = df_train['axl1_r_w_rim'].apply(lambda x: axl1_r_w_rim_mean if x == 0 else x)  
  
axl2_l_w_rim_mean = df_train[df_train['axl2_l_w_rim'] != 0]['axl2_l_w_rim'].mean()  
df_train['axl2_l_w_rim'] = df_train['axl2_l_w_rim'].apply(lambda x: axl2_l_w_rim_mean if x == 0 else x)  
  
axl2_r_w_rim_mean = df_train[df_train['axl2_r_w_rim'] != 0]['axl2_r_w_rim'].mean()  
df_train['axl2_r_w_rim'] = df_train['axl2_r_w_rim'].apply(lambda x: axl2_r_w_rim_mean if x == 0 else x)  
  
axl3_l_w_rim_mean = df_train[df_train['axl3_l_w_rim'] != 0]['axl3_l_w_rim'].mean()  
df_train['axl3_l_w_rim'] = df_train['axl3_l_w_rim'].apply(lambda x: axl3_l_w_rim_mean if x == 0 else x)  
  
axl3_r_w_rim_mean = df_train[df_train['axl3_r_w_rim'] != 0]['axl3_r_w_rim'].mean()  
df_train['axl3_r_w_rim'] = df_train['axl3_r_w_rim'].apply(lambda x: axl3_r_w_rim_mean if x == 0 else x)  
  
axl4_l_w_rim_mean = df_train[df_train['axl4_l_w_rim'] != 0]['axl4_l_w_rim'].mean()  
df_train['axl4_l_w_rim'] = df_train['axl4_l_w_rim'].apply(lambda x: axl4_l_w_rim_mean if x == 0 else x)  
  
axl4_r_w_rim_mean = df_train[df_train['axl4_r_w_rim'] != 0]['axl4_r_w_rim'].mean()  
df_train['axl4_r_w_rim'] = df_train['axl4_r_w_rim'].apply(lambda x: axl4_r_w_rim_mean if x == 0 else x)
```

```
✓ 0 1 max_corr = df_train_.corr()['target_day'].apply(lambda x:abs(x)).sort_values(ascending=False,axis=0)  
OCC max_corr.iloc[:26]
```

```
target_day      1.000000  
target_month    0.582258  
prob_end_month  0.158686  
ost_prob        0.157307  
day_run         0.076598  
mean_run        0.069795  
tipvozdz_3      0.044693  
axl1_r_w_flange 0.036771  
axl4_r_w_flange 0.034141  
axl2_l_w_flange 0.033077  
isload          0.032756  
cnsi_probeg_dr  0.032540  
cnsi_probeg_kr  0.030562  
axl3_r_w_flange 0.029426  
axl2_r_w_flange 0.028705  
telega         0.025706  
axl3_l_w_flange 0.025053  
cnsi_volumek    0.024210  
kod_vrab        0.022651  
axl1_l_w_flange 0.022611  
kuzov           0.021447  
rod_id          0.021301  
tormoz          0.019480  
tipvozdz_6      0.017800  
norma_km        0.016718  
axl3_l_w_rim    0.016295  
Name: target_day, dtype: float64
```

Обучение модели:

Для прогнозирования выбытия вагона в плановый ремонт применяется модель К ближайших соседей (KNN). Количество соседей вышло 3 в связи с лучшей сходимостью.

```
Обучение модели

[52] my_random_state = 1234

[53] y_day= df_train['target_day']
     y_month = df_train['target_month']
     x = df_train.drop(['target_month', 'target_day'], axis=1)

[54] x_train, x_test, y_train, y_test = train_test_split(x, y_day, test_size=0.2, random_state = my_random_state)

[55] KNN_day = KNeighborsClassifier(n_neighbors=3)
     KNN_day.fit(x, y_day)

KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)

[56] y_pred = KNN_day.predict(x_test)
     print(accuracy_score(y_test, y_pred))

0.9891752577319588

[57] x_train, x_test, y_train, y_test = train_test_split(x, y_month, test_size=0.2, random_state = my_random_state)

[58] KNN_month = KNeighborsClassifier(n_neighbors=3)
     KNN_month.fit(x, y_month)

KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)

y_pred = KNN_month.predict(x_test)
print(accuracy_score(y_test, y_pred))

0.9690721649484536
```

Обработка данных для предсказания производилась так же, как и для обучения модели. Перед прогнозированием проверялось условие на наличие ремонта вагона в прошлом месяце. Если в прошлом месяце у вагона производился плановый ремонт, то в текущем месяце его не будет.

▼ Проведение предсказания модели

```
[ ] for i in pred['wagnum'].values:
    #если в прошлом месяце был ремонт, то в текущем его не будет
    if train[train['wagnum']== i]['target_month'].values[0] == 1:
        pred.loc[pred['wagnum']==i, 'target_month'] = 0
        pred.loc[pred['wagnum']==i, 'target_day'] = 0
    else:
        x = df_pred[df_pred['wagnum'] == i].drop('wagnum', axis=1)
        pred_day = KNN_day.predict(x)
        if pred_day == 1:
            pred.loc[pred['wagnum']==i, 'target_month'] = int(pred_day[0])
            pred.loc[pred['wagnum']==i, 'target_day'] = int(pred_day[0])
        else:
            pred_month = KNN_month.predict(x)
            pred.loc[pred['wagnum']==i, 'target_month'] = int(pred_month[0])
            pred.loc[pred['wagnum']==i, 'target_day'] = int(pred_day[0])
pred
```

	wagnum	month	target_month	target_day
0	33361	2023-03-01	0	0
1	33364	2023-03-01	0	0
2	33366	2023-03-01	0	0
3	33358	2023-03-01	0	0
4	33349	2023-03-01	0	0
...
33702	17621	2023-03-01	0	0
33703	25045	2023-03-01	0	0
33704	27156	2023-03-01	0	0
33705	21361	2023-03-01	0	0
33706	8061	2023-03-01	1	1

33707 rows x 4 columns

```
[ ] pred.to_csv('/content/y_pred_submit.csv')
```

```
[ ] print(calc_f1_score('/content/y_pred_submit.csv', '/content/drive/MyDrive/DataWagon/y_predict.csv'))
```

0.4004992141102256

В результате предсказания общий счет модели равен 40%. Счет модели складывался из половины точности предсказания на месяц и половины точности предсказания на день.

▼ Функция для оценки на приватном лидерборде (все данные)

```
✓ 0 сек. ▶ # Функция для оценки на приватном лидерборде (все данные)
def calc_f1_score(test_url: str, prediction_url: str) -> float:

    true_labels = pd.read_csv(test_url)
    pred_labels = pd.read_csv(prediction_url)
    # Таргет для месячного прогноза
    true_labels_month = true_labels['target_month'].values
    pred_labels_month = pred_labels['target_month'].values

    # Таргет для 10 дневного прогноза
    true_labels_day = true_labels['target_day'].values
    pred_labels_day = pred_labels['target_day'].values

    # Посчитаем метрику для месяца и 10 дней
    score_month = f1_score(true_labels_month, pred_labels_month)
    score_day = f1_score(true_labels_day, pred_labels_day)
    # Посчитаем метрику с весом для двух таргетов
    score = 0.5 * score_month + 0.5 * score_day
    return score
```

Заключение:

Исходя из работы можно сделать вывод, что модель имеет удовлетворительную прогностическую способность и для дальнейшего ее использования в производстве необходима доработка, например, обучение на большем количестве данных, так как в представленном наборе в большом количестве строк отсутствовали значения.