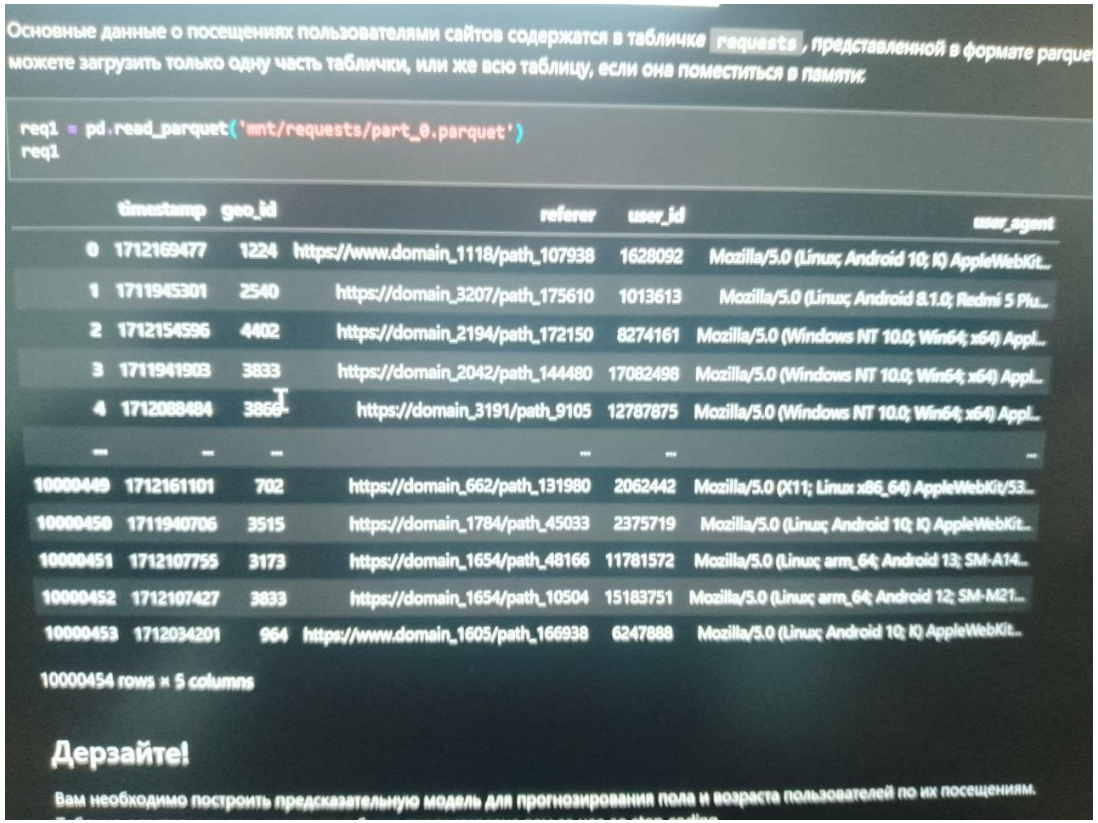
ВСЕМ ПРИВЕТ

От команды 1.5 программиста

Проданные:

Очевидно, что данные данные нужно модифицировать



Основные данные о посещениях пользователями сайтов содержатся в табличке `requests`, представленной в формате parquet
можете загрузить только одну часть таблички, или же всю таблицу, если она поместиться в памяти.

```
req1 = pd.read_parquet('mnt/requests/part_0.parquet')
req1
```

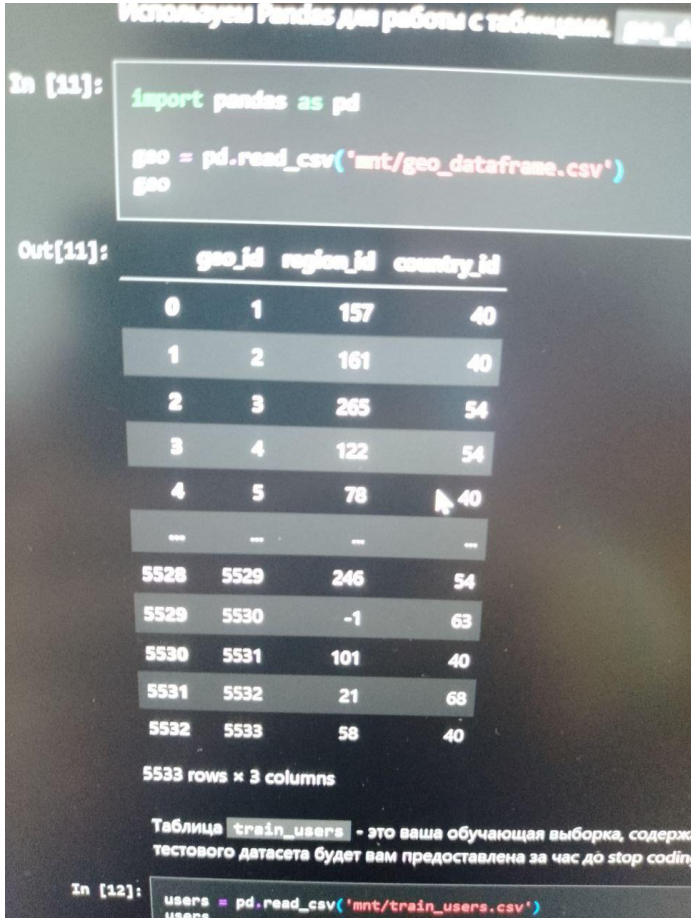
	timestamp	geo_id	referer	user_id	user_agent
0	1712169477	1224	https://www.domain_1118/path_107938	1628092	Mozilla/5.0 (Linux; Android 10; K) AppleWebKit...
1	1711945301	2540	https://domain_3207/path_175610	1013613	Mozilla/5.0 (Linux; Android 8.1.0; Redmi 5 Plu...
2	1712154596	4402	https://domain_2194/path_172150	8274161	Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl...
3	1711941903	3833	https://domain_2042/path_144480	17082498	Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl...
4	1712088484	3866	https://domain_3191/path_9105	12787875	Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl...
...	...	...	...	...	...
10000449	1712161101	702	https://domain_662/path_131980	2062442	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/53...
10000450	1711940706	3515	https://domain_1784/path_45033	2375719	Mozilla/5.0 (Linux; Android 10; K) AppleWebKit...
10000451	1712107755	3173	https://domain_1654/path_48166	11781572	Mozilla/5.0 (Linux; arm_64; Android 13; SM-A14...
10000452	1712107427	3833	https://domain_1654/path_10504	15183751	Mozilla/5.0 (Linux; arm_64; Android 12; SM-M21...
10000453	1712034201	964	https://www.domain_1605/path_166938	6247888	Mozilla/5.0 (Linux; Android 10; K) AppleWebKit...

10000454 rows × 5 columns

Дерзайте!

Вам необходимо построить предсказательную модель для прогнозирования пола и возраста пользователей по их посещениям.

Проданные:



In [11]:
```
import pandas as pd

geo = pd.read_csv('mnt/geo_dataframe.csv')
geo
```

Out[11]:

	geo_id	region_id	country_id
0	1	157	40
1	2	161	40
2	3	265	54
3	4	122	54
4	5	78	40
...	...	...	...
5528	5529	246	54
5529	5530	-1	63
5530	5531	101	40
5531	5532	21	68
5532	5533	58	40

5533 rows × 3 columns

Таблица train_users - это ваша обучающая выборка, содерж
тестового датасета будет вам предоставлена за час до stop codin

In [12]:
```
users = pd.read_csv('mnt/train_users.csv')
```

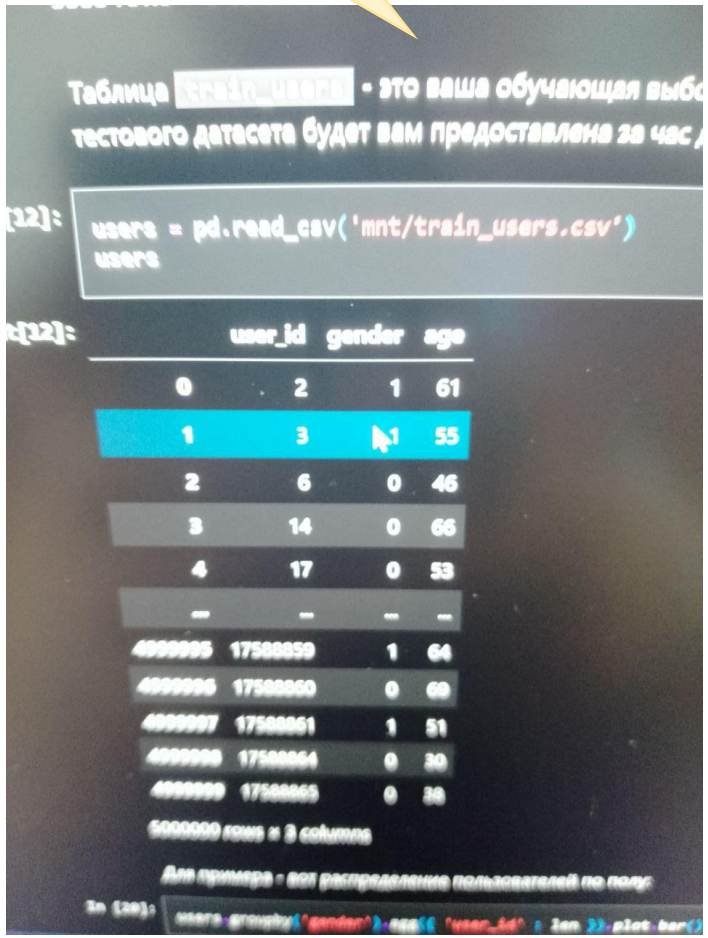


Таблица [redacted] - это ваша обучающая выбо
тестового датасета будет вам предоставлена за час

[12]:
```
users = pd.read_csv('mnt/train_users.csv')
users
```

[12]:

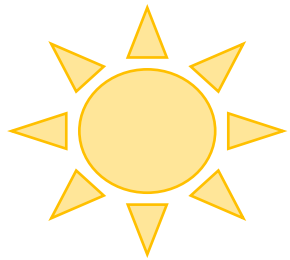
	user_id	gender	age
0	2	1	61
1	3	1	55
2	6	0	46
3	14	0	66
4	17	0	53
...	...	...	...
4999995	17588859	1	64
4999996	17588860	0	60
4999997	17588861	1	51
4999998	17588864	0	30
4999999	17588865	0	38

5000000 rows × 3 columns

Для примера - вот распределение пользователей по полу:

In [20]:
```
users.groupby('gender').agg({'user_id' : len}).plot.bar()
```

Наши данные



	user_id	gender	age	domain_top1	domain_top2	domain_top3	device_type_top1	device_type_top2	brand_top1	hour_top1	hour_top2	hour_top3	weekday_top1	weekday_top2
0	2	1	61	domain_1654	NaN	NaN	mobile	NaN	Huawei	0	NaN	NaN	1	NaN
1	3	1	55	domain_2867	www.domain_78	NaN	mobile	NaN	Generic_Android	15	18.0	NaN	2	1.0
2	6	0	46	domain_3194	domain_1834	www.domain_1123	mobile	NaN	Generic_Android	23	8.0	19.0	0	1.0
3	14	0	66	domain_2238	NaN	NaN	PC	NaN	NaN	17	NaN	NaN	1	NaN
4	17	0	53	domain_2285	www.domain_2582	www.domain_824	mobile	NaN	Generic_Android	14	17.0	15.0	2	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4999994	17588855	1	50	domain_145		domain_3019	mobile	NaN	Generic_Android	0	18.0	13.0	0	2.0
4999996	17588860	0	69	domain_2238	domain_2042	domain_609	PC	NaN	NaN	1	16.0	12.0	1	0.0
4999997	17588861	1	51	domain_21	NaN	NaN	mobile	NaN	Generic_Android	5	NaN	NaN	2	NaN
4999998	17588864	0	30	domain_2194	NaN	NaN	PC	NaN	NaN	2	4.0	6.0	0	1.0
4999999	17588865	0	38	domain_2998	NaN	NaN	PC	NaN	NaN	19	NaN	NaN	2	NaN

Решение

Что мы пробовали:

- Кластеризация (Nearest Neighbours)
- Статистический подход
- Catboost
- LightGBM

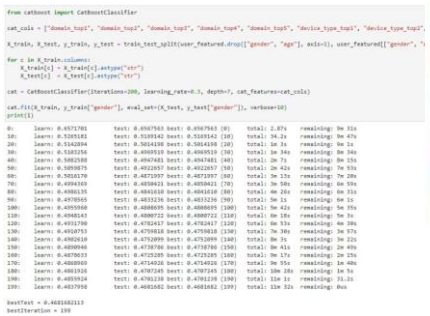
Проблемы

KERNEL ERROR (80% of time)

Проблемы

Много много много много данных

ПОЛОВЫЕ РЕЗУЛЬТАТЫ



```
from catboost import CatBoostClassifier

cat_cols = ["domain_top1", "domain_top2", "domain_top3", "domain_top4", "domain_top5", "device_type_top1", "device_type_top2",

X_train, X_test, y_train, y_test = train_test_split(user_featured.drop(["gender", "age"], axis=1), user_featured[["gender", "

for c in X_train.columns:
    X_train[c] = X_train[c].astype("str")
    X_test[c]  = X_test[c].astype("str")

cat = CatBoostClassifier(iterations=200, learning_rate=0.3, depth=7, cat_features=cat_cols)

cat.fit(X_train, y_train["gender"], eval_set=(X_test, y_test["gender"]), verbose=10)
print()
```

```
0:      learn: 0.6571701        test: 0.6567563 best: 0.6567563 (0)     total: 2.07s    remaining: 9m 31s
10:     learn: 0.5265181        test: 0.5169142 best: 0.5169142 (10)    total: 34.2s    remaining: 9m 47s
20:     learn: 0.5142894        test: 0.5014198 best: 0.5014198 (20)    total: 1m 3s    remaining: 9m 1s
30:     learn: 0.5103256        test: 0.4969519 best: 0.4969519 (30)    total: 1m 34s   remaining: 8m 34s
40:     learn: 0.5082598        test: 0.4947481 best: 0.4947481 (40)    total: 2m 7s    remaining: 8m 15s
50:     learn: 0.5059075        test: 0.4922657 best: 0.4922657 (50)    total: 2m 42s   remaining: 7m 53s
60:     learn: 0.5026170        test: 0.4871997 best: 0.4871997 (60)    total: 3m 17s   remaining: 7m 28s
70:     learn: 0.4994369        test: 0.4830421 best: 0.4830421 (70)    total: 3m 50s   remaining: 6m 59s
80:     learn: 0.4966135        test: 0.4841610 best: 0.4841610 (80)    total: 4m 26s   remaining: 6m 31s
90:     learn: 0.4939565        test: 0.4833236 best: 0.4833236 (90)    total: 5m 1s    remaining: 6m 1s
100:    learn: 0.4955960        test: 0.4886695 best: 0.4886695 (100)   total: 5m 42s   remaining: 5m 35s
110:    learn: 0.4940143        test: 0.4880722 best: 0.4880722 (110)   total: 6m 18s   remaining: 5m 3s
120:    learn: 0.4931790        test: 0.4782417 best: 0.4782417 (120)   total: 6m 53s   remaining: 4m 30s
130:    learn: 0.4910753        test: 0.4759818 best: 0.4759818 (130)   total: 7m 30s   remaining: 3m 57s
140:    learn: 0.4902610        test: 0.4752099 best: 0.4752099 (140)   total: 8m 3s    remaining: 3m 22s
150:    learn: 0.4890946        test: 0.4738706 best: 0.4738706 (150)   total: 8m 41s   remaining: 2m 49s
160:    learn: 0.4878633        test: 0.4725285 best: 0.4725285 (160)   total: 9m 17s   remaining: 2m 15s
170:    learn: 0.4868969        test: 0.4714926 best: 0.4714926 (170)   total: 9m 55s   remaining: 1m 40s
180:    learn: 0.4861926        test: 0.4707245 best: 0.4707245 (180)   total: 10m 28s  remaining: 1m 5s
190:    learn: 0.4855924        test: 0.4701238 best: 0.4701238 (190)   total: 11m 1s   remaining: 31.2s
199:    learn: 0.4851998        test: 0.4691682 best: 0.4691682 (199)   total: 11m 32s  remaining: 0us

bestTest = 0.4691681113
bestIteration = 199
```

```
from sklearn.metrics import classification_report

print(classification_report(y_test["gender"], cat.predict(X_test)))
```

	precision	recall	f1-score	support
0	0.74	0.66	0.70	1122361
1	0.73	0.80	0.77	1309122
accuracy			0.74	2431483
macro avg	0.74	0.73	0.73	2431483
weighted avg	0.74	0.74	0.74	2431483

ВОЗРАСТНЫЕ РЕЗУЛЬТАТЫ

```
from catboost import CatBoostRegressor

cat_cols1 = ["domain_top1", "domain_top2", "domain_top3", "domain_top4", "domain_top5", "device_type_top1", "device_type_top2", "brand_top1"]

X_train1, X_test1, y_train1, y_test1 = train_test_split(user_featured.drop(["gender", "age"], axis=1), user_featured[["gender", "age"]], train_size=0.8, stratify=user_featured["gender"])

for c in X_train1.columns:
    X_train1[c] = X_train1[c].astype("str")
    X_test1[c]  = X_test1[c].astype("str")

cat1 = CatBoostRegressor(iterations=100, learning_rate=0.5, depth=7, cat_features=cat_cols)

cat1.fit(X_train1, y_train1["age"], eval_set=(X_test1, y_test1["age"]), verbose=10)
print(i)
```

```
0:      learn: 13.2666071       test: 13.2504656        best: 13.2504656 (0)    total: 971ms    remaining: 1m 36s
10:     learn: 12.6320811       test: 12.6132240        best: 12.6132240 (10)   total: 8.02s    remaining: 1m 4s
20:     learn: 12.5584528       test: 12.5397816        best: 12.5397816 (20)   total: 14.3s    remaining: 53.7s
30:     learn: 12.5315257       test: 12.5134393        best: 12.5134393 (30)   total: 20.7s    remaining: 46s
40:     learn: 12.5098814       test: 12.4916957        best: 12.4916957 (40)   total: 27.1s    remaining: 39s
50:     learn: 12.4897634       test: 12.4719605        best: 12.4719605 (50)   total: 33.4s    remaining: 32.1s
60:     learn: 12.4729714       test: 12.4557422        best: 12.4557422 (60)   total: 39.9s    remaining: 25.5s
70:     learn: 12.4622822       test: 12.4454320        best: 12.4454320 (70)   total: 46.2s    remaining: 18.9s
80:     learn: 12.4494916       test: 12.4327391        best: 12.4327391 (80)   total: 52.4s    remaining: 12.3s
90:     learn: 12.4382217       test: 12.4218747        best: 12.4218747 (90)   total: 58.8s    remaining: 5.82s
99:     learn: 12.4272544       test: 12.4113459        best: 12.4113459 (99)   total: 1m 4s    remaining: 0us

bestTest = 12.41134592
bestIteration = 99
```

```
from sklearn.metrics import mean_absolute_percentage_error
print(mean_absolute_percentage_error(y_test["age"], cat1.predict(X_test)))
```

```
0.230530389027284 82
```

Спасибо

Грустный хомяк