Турникеты 2.0: Чей это след?

Задача:

На тестовой выборке сопоставить словам id посетителей, которые проходили через турникеты, т.е. нужно научиться предсказывать id посетителей по их цифровому следу.

Александр Павлов

Импорт данных для разведочного анализа

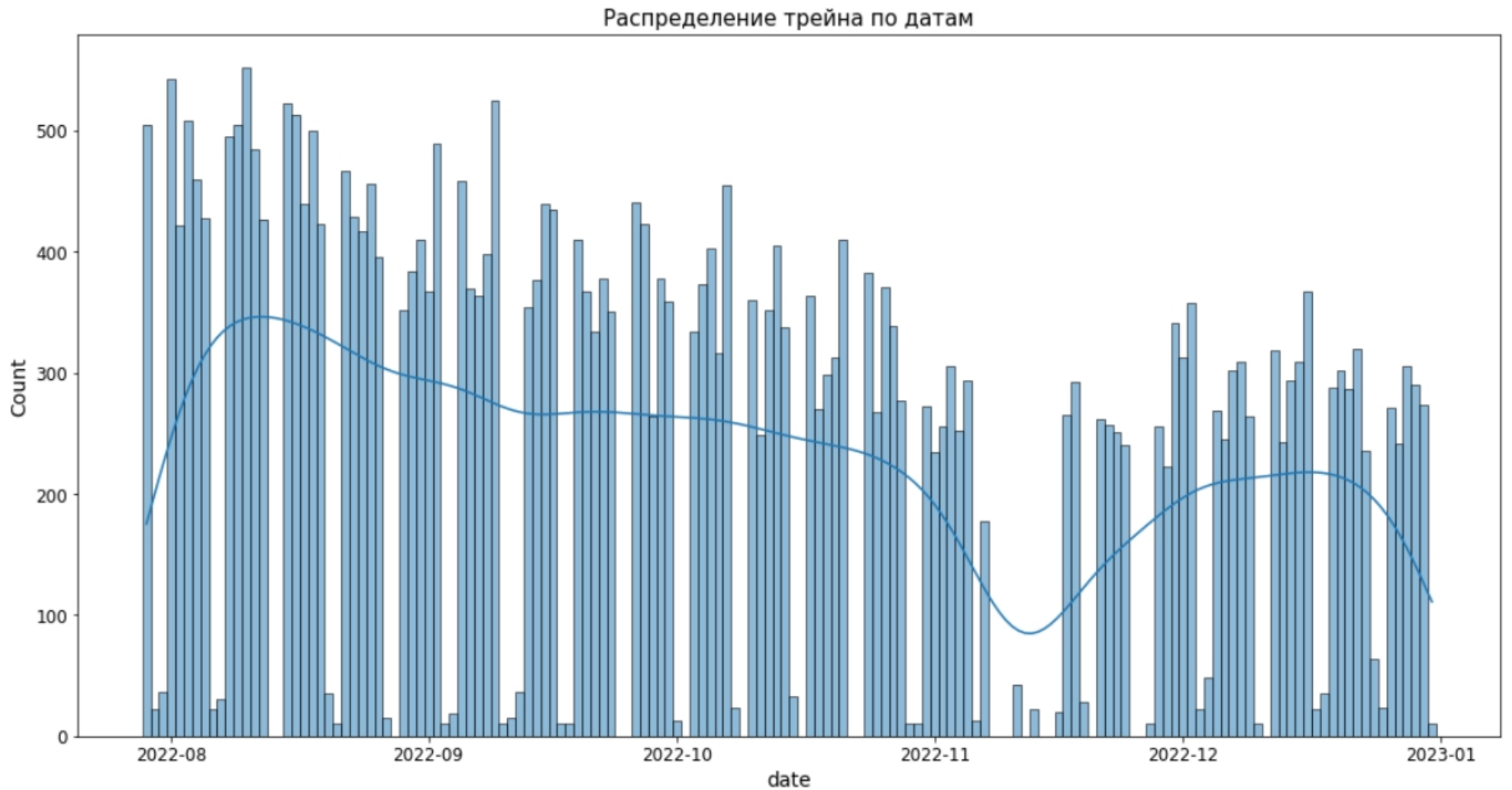
In [2]:
```
df = pd.read_csv('train.csv', parse_dates=['ts'], index_col=0)
test_df = pd.read_csv('test.csv', parse_dates=['ts'], index_col=0)
```

In [3]:
```
# добавление новых признаков

df['date'] = df['ts'].dt.date
df['day'] = df['ts'].dt.day
df['hour'] = df['ts'].dt.hour
df['minute'] = df['ts'].dt.minute
df['second'] = df['ts'].dt.second
df['month'] = df['ts'].dt.month

test_df['date'] = test_df['ts'].dt.date
test_df['day'] = test_df['ts'].dt.day
test_df['hour'] = test_df['ts'].dt.hour
test_df['minute'] = test_df['ts'].dt.minute
test_df['month'] = test_df['ts'].dt.month

df
```

Out[3]:

	user_id	ts	gate_id	date	day	hour	minute	second	month
0	18	2022-07-29 09:08:54	7	2022-07-29	29	9	8	54	7
1	18	2022-07-29 09:09:54	9	2022-07-29	29	9	9	54	7
2	18	2022-07-29 09:09:54	9	2022-07-29	29	9	9	54	7
3	18	2022-07-29 09:10:06	5	2022-07-29	29	9	10	6	7
4	18	2022-07-29 09:10:08	5	2022-07-29	29	9	10	8	7

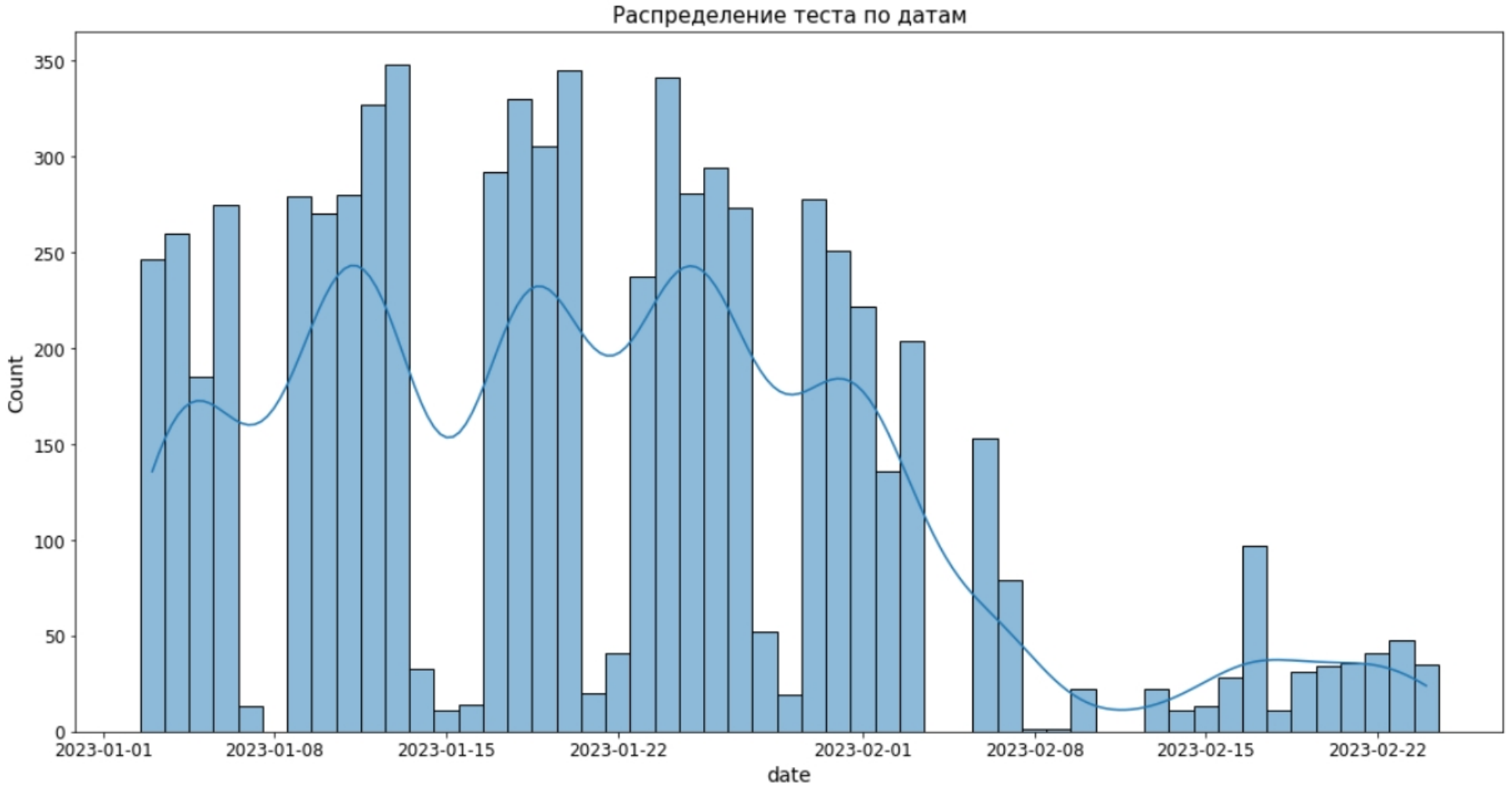
In [5]:
```
plt.figure(figsize=(18,9))
sns.histplot(data=df['date'], kde=True, bins=df['date'].nunique())
plt.title("Распределение трейна по датам")
plt.show()
```



В ноябре есть пропуски в данных

In [6]:

```
plt.figure(figsize=(18,9))
sns.histplot(data=test_df['date'], kde=True, bins=test_df['date'].nunique())
plt.title("Распределение теста по датам")
plt.show()
```



В феврале есть пропуски в данных

Посмотрим на количество посещений пользователей

In [13]:
```
grp_user = df.groupby('user_id').agg(
    counts=('ts', 'count'),
    date_unique=('date', lambda x: x.nunique()),
    date_min=('date', 'min'),
    date_max=('date', 'max'),
)
grp_user.sort_values(['counts', 'date_min']).head(7)
```

Out[13]:

user_id	counts	date_unique	date_min	date_max
4	2	1	2022-08-09	2022-08-09
51	3	2	2022-12-13	2022-12-16
44	4	1	2022-12-28	2022-12-28
52	5	1	2022-08-10	2022-08-10
21	5	2	2022-12-16	2022-12-28
5	10	2	2022-10-26	2022-11-30
30	10	2	2022-12-16	2022-12-27

Редких пользователей user_id 4, 51, 52 возможно убрать из трейна ?

Какие турникеты используются в трейне и тесте:

In [14]:
```
sorted(df.gate_id.unique())
```

Out[14]:
```
[-1, 0, 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
```

In [15]:
```
sorted(test_df.gate_id.unique())
```

Out[15]:
```
[-1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
```

In [16]:
```
# турникеты, которых нет в тесте
set(df.gate_id.unique()) - set(test_df.gate_id.unique())
```

Out[16]:
```
{0, 16}
```

Возможно эти турникеты следует убрать из обучающей выборки?

In [17]:
```
# кто и когда ходил через эти турникеты
df[df.gate_id.isin([0, 16])]
```

Out[17]:

	user_id	ts	gate_id	date	day	hour	minute	second	month
12652	25	2022-09-06 11:16:28	0	2022-09-06	6	11	16	28	9
12653	25	2022-09-06 11:16:36	0	2022-09-06	6	11	16	36	9
21309	25	2022-10-07 16:44:37	16	2022-10-07	7	16	44	37	10
21310	25	2022-10-07 16:44:38	16	2022-10-07	7	16	44	38	10
36798	56	2022-12-28 14:49:51	16	2022-12-28	28	14	49	51	12
36799	21	2022-12-28 14:49:54	16	2022-12-28	28	14	49	54	12

Посмотрим на распределение посещений по рабочим/выходным дням

In [33]:
```
flt_cols = ['date', 'weekday', 'beep_count', 'is_weekend']
tmp = all_df[flt_cols].drop_duplicates()
tmp["weekend"] = tmp["weekday"].map(lambda x: 1 if x in (5, 6) else 0)
tmp['cmp_weekends'] = tmp["weekend"] == tmp["is_weekend"]
grp_date = all_df.groupby(['date'], as_index=False).agg(date_cnt=('ts', 'count'))
```

In [34]:
```
tmp[tmp["weekend"] == 1].beep_count.describe()
```

Out[34]:
```
count     40.000000
mean      29.400000
std       44.799153
min       11.000000
25%       11.000000
50%       21.000000
75%       31.500000
max      294.000000
Name: beep_count, dtype: float64
```

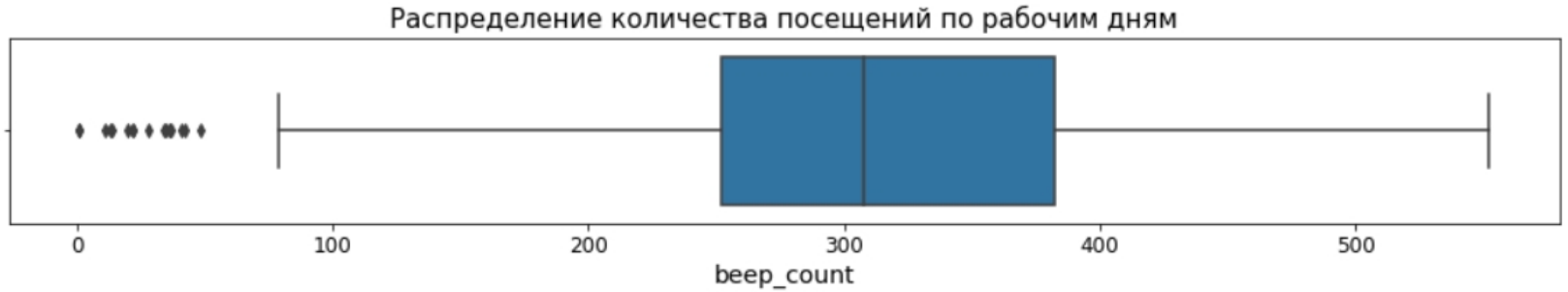
In [35]:
```
fig, ax = plt.subplots(figsize=(16, 2))
sns.boxplot(x='beep_count', data=tmp[tmp["weekend"] == 1], ax=ax)
plt.title('Распределение количества посещений по выходным дням')
plt.show()
```

Есть выбросы, нужно отметить эти выходные дни как рабочие

In [37]:
```
tmp[tmp["weekend"] == 0].beep_count.describe()
```

Out[37]:
```
count    144.000000
mean     301.854167
std      132.062232
min        1.000000
25%      251.750000
50%      307.500000
75%      382.500000
max      552.000000
Name: beep_count, dtype: float64
```

In [38]:
```
fig, ax = plt.subplots(figsize=(16, 2))
sns.boxplot(x='beep_count', data=tmp[tmp["weekend"] == 0], ax=ax)
plt.title('Распределение количества посещений по рабочим дням')
plt.show()
```



Есть выбросы в рабочих днях, нужно отмечать такие дни как выходные

In [39]:
```
beep_counts = tmp[tmp["weekend"] == 1].beep_count
beep_quantile = beep_counts.quantile(0.975)
beep_quantile_std = beep_counts.quantile(0.75) + beep_counts.std() * 1.5
print(beep_quantile, beep_quantile_std)
```

```
69.74999999999967 98.69872938359201
```

Попробовать обе стратегии по разграничению дней:

- порог 5%, т.е. 2.5% от максимального значения по выходным (или минимального значения по рабочим дням) = 69.7
- взять третий квантиль + полтора стандартных отклонения: как на графике с боксплотами = 98.7

Создание признаков

```
df["date"] = df['ts'].dt.date
df["time"] = df['ts'].dt.time
df["day"] = df['ts'].dt.day
df["hour"] = df['ts'].dt.hour
df["min"] = df['ts'].dt.minute
df["sec"] = df['ts'].dt.second
df['minutes'] = df["hour"] * 60 + df["min"]
df['seconds'] = df.minutes * 60 + df["sec"]
# 1-й день месяца
df["1day"] = df['ts'].dt.is_month_start.astype(int)
# 2-й день месяца
df["2day"] = (df.day == 2).astype(int)
# Предпоследний день месяца
df["last_day-1"] = (df.day == df.ts.dt.daysinmonth - 1).astype(int)
# Последний день месяца
df["last_day"] = df['ts'].dt.is_month_end.astype(int)
# День недели от 0 до 6
df["weekday"] = df['ts'].dt.dayofweek
# Метка выходного дня
df["is_weekend"] = df["weekday"].map(lambda x: 1 if x in (5, 6) else 0)
# Метки "график 2 через 2"  - второй со сдвигом на 1 день
df["DofY1"] = (df['ts'].dt.dayofyear % 4).apply(lambda x: int(x in (1, 2)))
df["DofY2"] = (df['ts'].dt.dayofyear % 4).apply(lambda x: int(x < 2))
```

Создание признаков

- Отметка рабочих дней как выходных и выходных как рабочих на основе порога по количеству посещений в день (69,7 и 98,7).
- Выделение шаблонов турникетов: длина 3-7 для одного пользователя и количеством от 4-х посещений по всем пользователям.
- Для каждой записи добавлен сдвиг по турникетам (номер и время прохода) 5 предыдущих и 5 следующих, т.е. 2 X 10 признаков. Для времени посчитана дельта между соседними турникетами:

row_id	user_id	ts	gate_id	g5	g4	g3	g2	g1	g0	g-1	g-2	g-3	g-4	g-5	t5	t4	t3	t2	t1	t0	t-1	t-2	t-3	t-4	t-5
0	18	29.07.2022 9:08	7	-9	-9	-9	-9	-9	7	9	9	5	5	10	0	0	0	0	0	0	60	0	12	2	26
1	18	29.07.2022 9:09	9	-9	-9	-9	-9	7	9	9	5	5	10	11	0	0	0	0	0	60	0	12	2	26	1333
2	18	29.07.2022 9:09	9	-9	-9	-9	7	9	9	5	5	10	11	4	0	0	0	0	60	0	12	2	26	1333	25
3	18	29.07.2022 9:10	5	-9	-9	7	9	9	5	5	10	11	4	4	0	0	0	60	0	12	2	26	1333	25	1
4	18	29.07.2022 9:10	5	-9	7	9	9	5	5	10	11	4	4	7	0	0	60	0	12	2	26	1333	25	1	3
5	18	29.07.2022 9:10	10	7	9	9	5	5	10	11	4	4	7	9	0	60	0	12	2	26	1333	25	1	3	7
6	18	29.07.2022 9:32	11	9	9	5	5	10	11	4	4	7	9	9	0	0	12	2	26	1333	25	1	3	7	0
7	18	29.07.2022 9:33	4	9	5	5	10	11	4	4	7	9	9	5	0	12	2	26	1333	25	1	3	7	0	18
8	18	29.07.2022 9:33	4	5	5	10	11	4	4	7	9	9	5	5	0	2	26	1333	25	1	3	7	0	18	1
9	1	29.07.2022 9:33	7	5	10	11	4	4	7	9	9	5	5	10	0	26	1333	25	1	3	7	0	18	1	22

Создание признаков

- Для каждого пользователя посчитано среднее время прохода между соседними турникетами.
- По шаблонам турникетов созданы бинарные признаки (около 150). Например, есть шаблон (9, 9,15) получаем признак G9_9_15, в котором 1/0 – есть/нет шаблон среди 11 последовательных турникетов, т.е. с учетом данных по предыдущим двум пунктам.
- Создание бинарных признаков на основе распределения дельт между соседними турникетами с такими диапазонами: $x = 1$, $x = 2$, $x <= 3$, $2 < x <= 5$, $5 < x <= 15$, $15 < x <= 25$, $25 < x <= 36$, $36 < x <= 49$, $49 < x <= 79$.
- Создание бинарных признаков из колонок: 'gate_id', 'weekday', 'hour‘ - применен one hot encoding.
- Ко всем не бинарным признакам применен StandardScaler().

Обучение модели

- Разбиение на обучающую и валидационную выборку выполнено train_test_split со стратификацией по user_id, т.к. разбиение по месяцам: обучение август-ноябрь, валидация – декабрь давало меньшую метрику.
- В качестве метрики выбраны: ROC_AUC и взвешенная мера F1.
- Экспериментальным путем подобраны: test_size=0.2, SEED=17.
- С помощь пакета optuna подобраны гиперпараметры: class_weight = None, multi_class = "ovr", solver = "newton-cholesky", penalty = 'l2', C = 5, max_iter = 13.

  Пробовал GridSearchCV – работает дольше.
- Обучена модель с помощью кросс-валидации на 4 фолдах (на 5 результат не отличался) с разбиением:

  skf = StratifiedKFold(n_splits=4, random_state=SEED, shuffle=True)

Формирование сабмита

- Получение user_id как самого частотного (как в baseline) для заданного слова не дает хорошего результата.
- Для каждого слова были посчитано процентное соотношение всех user_id. Можно назвать это вероятностью появления user_id, т.к. в сумме они = 1.
- Рассмотрим на примере user_id = 35: этот идентификатор присваивается тому слову, у которого максимальное значение p_value – это «binary».
- Для других слов 35 удаляется и выбирается следующий user_id из списка.
- Цикл повторяется до присвоения идентификаторов всем словам.

user_word	pred	p_value
aucroc	24	[(24, 0.7560975609756098), (36, 0.21951219512195122), (7, 0.024390243902439025)]
binary	35	[(35, 0.8549618320610687), (41, 0.0737913486005089), (43, 0.04834605597964377), (17, 0.010178117048346057), (42, 0.007633587786259542), (2
blue	23	[(23, 0.5), (40, 0.25), (25, 0.16666666666666666), (56, 0.08333333333333333)]
categorical	14	[(14, 0.6988416988416989), (43, 0.16988416988416988), (23, 0.05405405405405406), (35, 0.05019305019305019), (29, 0.019305019305019305), (9,
coefficient	35	[(35, 0.46153846153846156), (22, 0.3384615384615385), (27, 0.1076923076923077), (17, 0.046153846153846156), (34, 0.03076923076923077), (14,
collinear	35	[(35, 0.46200607902735563), (43, 0.2006079027355623), (17, 0.1337386018237082), (53, 0.1094224924012158), (27, 0.03951367781155015), (41, 0.
distributed	35	[(35, 0.6504065040650406), (17, 0.17886178861788618), (42, 0.06504065040650407), (34, 0.032520325203252036), (53, 0.032520325203252036), (2
epsilon	24	[(24, 0.5148247978436657), (26, 0.2668463611859838), (41, 0.07008086253369272), (17, 0.05660377358490566), (35, 0.04851752021563342), (34, 0
f1	41	[(41, 0.5703022339027596), (18, 0.2233902759526938), (35, 0.06307490144546649), (55, 0.06176084099868594), (53, 0.03942181340341656), (42, 0
fit	17	[(17, 0.34196891191709844), (35, 0.25906735751295334), (23, 0.21761658031088082), (43, 0.15544041450777202), (2, 0.015544041450777202), (34
gini	35	[(35, 0.4305555555555556), (17, 0.22916666666666666), (43, 0.11458333333333333), (34, 0.0625), (22, 0.052083333333333336), (28, 0.045138888
independent	24	[(24, 0.3858520900321543), (35, 0.3633440514469453), (17, 0.07395498392282958), (26, 0.06109324758842444), (2, 0.04501607717041801), (7, 0.0