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

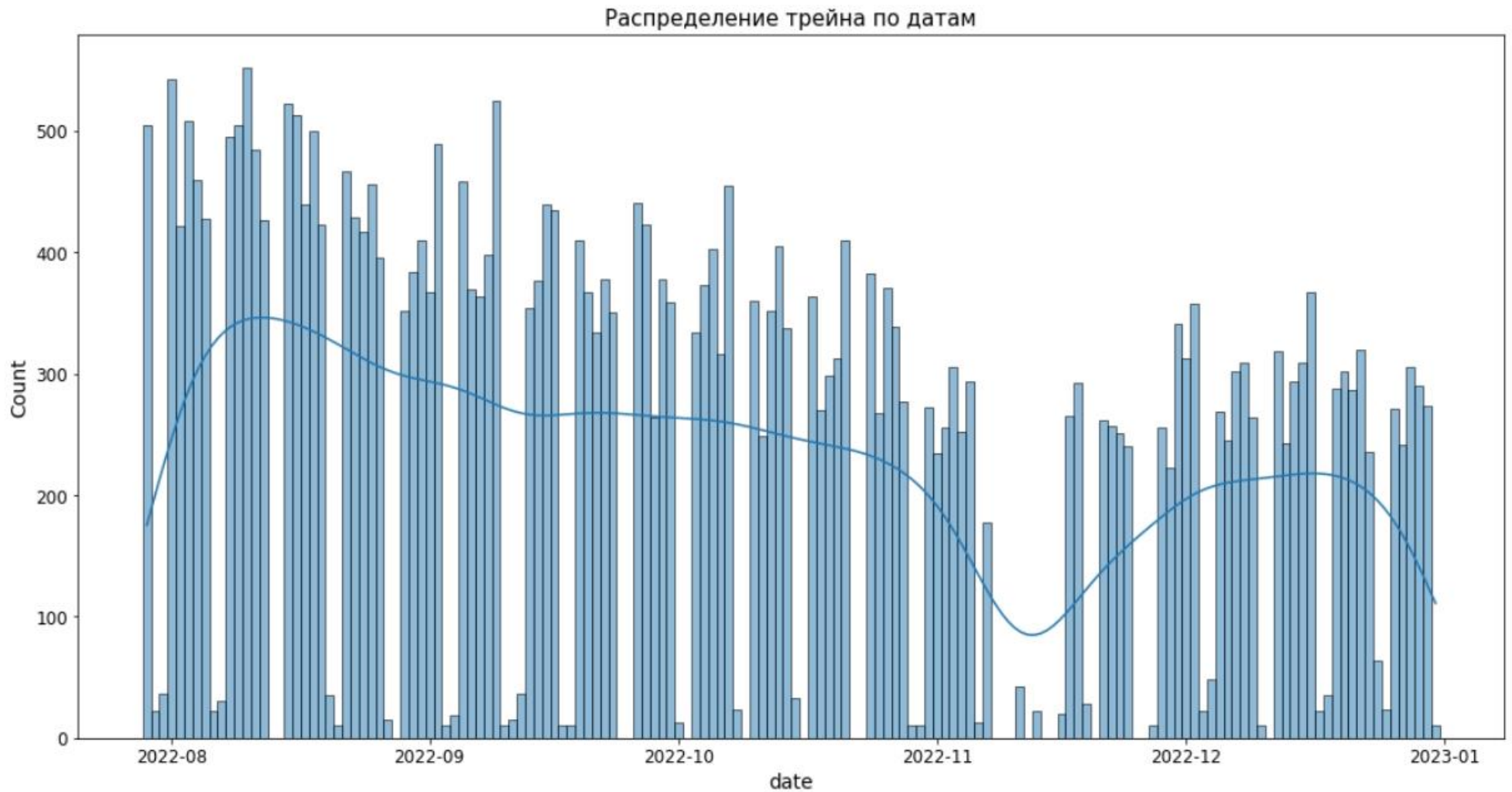
In [2]:
```
df = pd.read_csv('train.csv', parse_dates=['ts'], index_col=0)
test_df = pd.read_csv('test.csv', parse_dates=['ts'], index_col=0)
```

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['second'] = test_df['ts'].dt.second
test_df['month'] = test_df['ts'].dt.month
df
```

Out[3]:

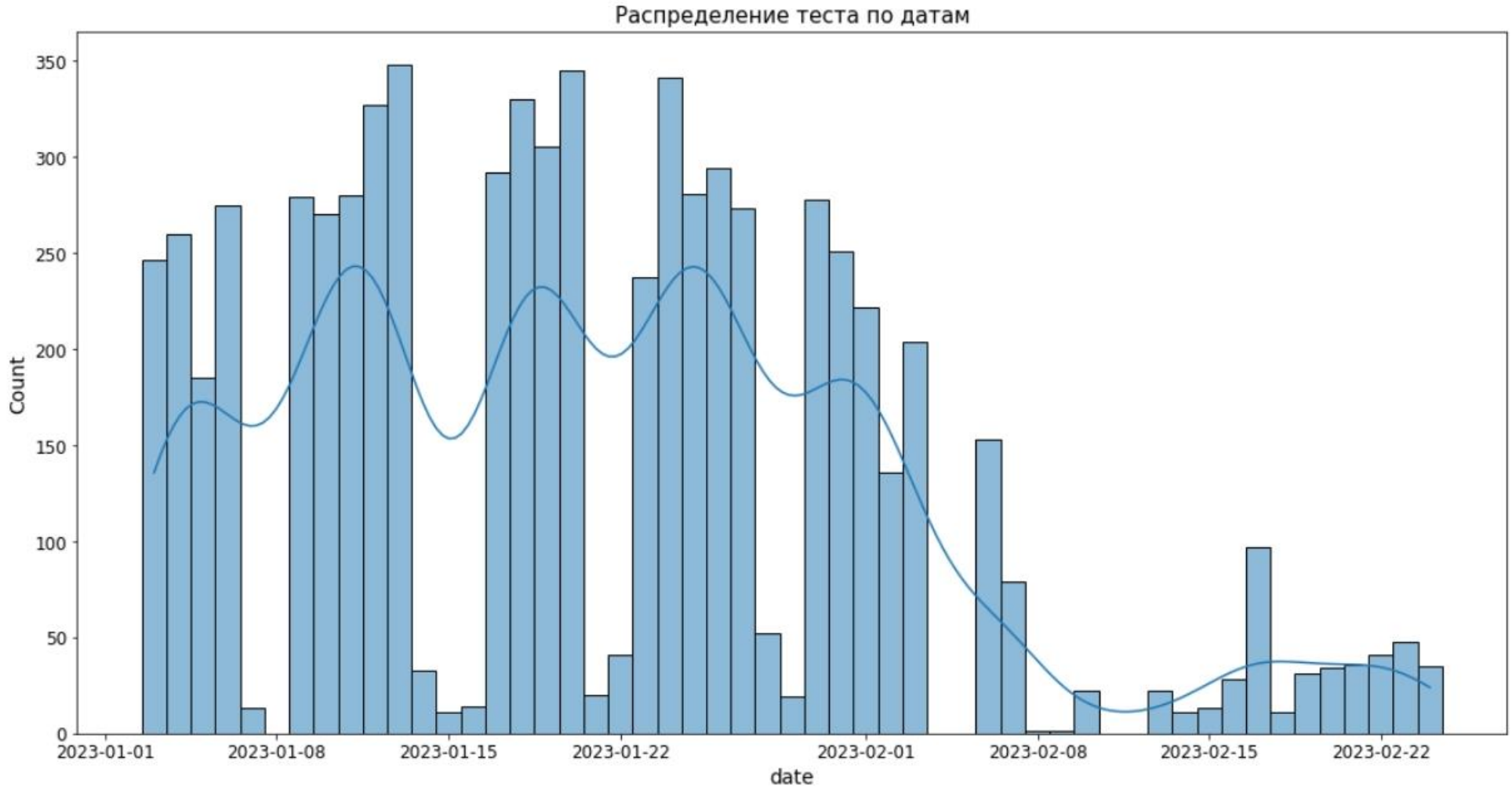
	user_id	ts	gate_id	date	day	hour	minute	second	month
0	18	2022-07-29 09:08:54	7	2022-07-29	29	9	8	54	7
1	18	2022-07-29 09:09:54	9	2022-07-29	29	9	9	54	7
2	18	2022-07-29 09:09:54	9	2022-07-29	29	9	9	54	7
3	18	2022-07-29 09:10:06	5	2022-07-29	29	9	10	6	7
4	18	2022-07-29 09:10:08	5	2022-07-29	29	9	10	8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

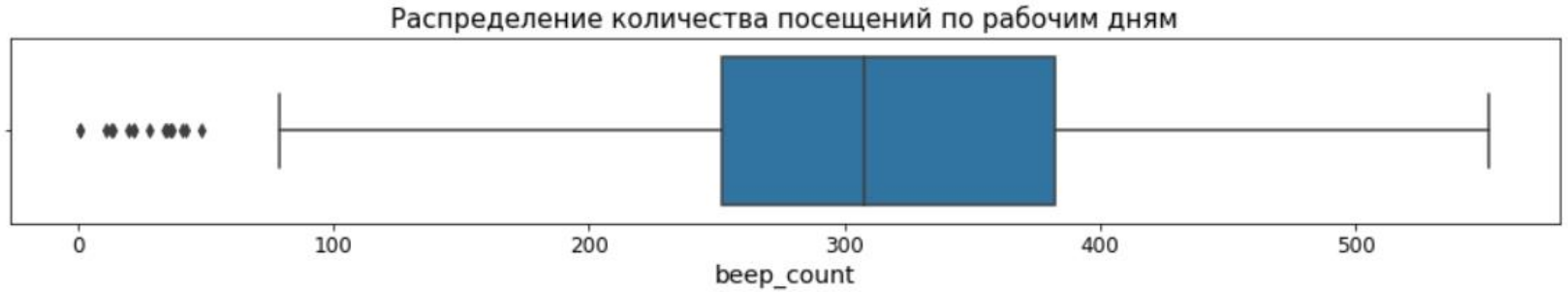
In [35]:
```
fig, ax = plt.subplots(figsize=(16, 2))
sns.boxplot(x='beep_count', data=tmp[tmp["weekend"] == 1], ax=ax)
plt.title('Распределение количества посещений по выходным дням')
plt.show()
```

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 = 0, 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.

Создание признаков

- Для каждой строки в датасете группировкой по пользователю и дате добавлены такие статистики:

1. Последовательность турникетов, которую он прошел за день;
2. Количество пройденных турникетов за день, время прихода, время ухода, длительность нахождения в офисе;
3. По этим статистикам посчитаны: среднее, медиана и стандартное отклонение.

- Пользу принесли: количество турникетов и среднее количество, среднее время прихода и стандартное отклонение времени прихода.
- Опробованы CountVectorizer и TfidfVectorizer для последовательности турникетов. Сработал CountVectorizer с параметрами: 32 турникета, ngram_range = (1, 4), max_features = 256.
- Всего создано 512 признаков из них 256 от CountVectorizer.
- Ко всем не бинарным признакам применен StandardScaler().

Обучение модели

- Разбиение на обучающую и валидационную выборку выполнено по месяцам: обучение август-ноябрь, валидация – декабрь, т.к. разбиение train_test_split со стратификацией по user_id на моих данных приводило к переобучению даже на глубине дерева = 1 из-за статистик по пользователю.
- В качестве метрики выбраны: ROC_AUC и взвешенная мера F1.
- С помощь GridSearchCV подобраны параметры модели: RandomForestClassifier (SEED=17 (остался от линейной модели), class_weight='balanced’, criterion='entropy’, max_depth=21, n_estimators=100) – это количество деревьев было по-умолчанию.
- Модели с бустингом LightGBM и CatBoost не завел из-за того, что в декабре появились новые пользователи, а их нет в обучающей выборке. Разбиение train_test_split со стратификацией приводило к переобучению.

Обучение модели

Важность признаков				
Feature	**Importance**	**...**	**Feature**	**Importance**
time_start_mean	0,089337564		G13_13_4_4	0,000000170
time_start_std	0,077227103		hour_6	0,000000157
user_day_beep_mean	0,072809007		gate_id_14	0,000000000
user_day_beep	0,028052747		G7_3_3_4	0,000000000
beep_count	0,025737895		G10_10_11_10	0,000000000
vct_156	0,019281510		G13_6_6_9	0,000000000
vct_142	0,016433036		G5_5_5_12	0,000000000
vct_000	0,016282236		gate_id_2	0,000000000
vct_129	0,016131517		hour_0	0,000000000
vct_001	0,014201590		hour_1	0,000000000
vct_192	0,013666012		hour_3	0,000000000

- В качестве эксперимента были убраны «неважные» признаки = 0 → скор уменьшился на 100 баллов.
- После подбора гиперпараметров модель была обучена на всем трейне, для тестовой выборки получены предсказания по user_id.

Формирование сабмита

- Получение user_id как самого частотного (как в baseline) для заданного слова не дало хорошего сабмита → результат **230**.
- Для каждого слова было посчитано процентное соотношение всех user_id. Можно назвать это вероятностью появления user_id, т.к. в сумме они = 1. При этом из предсказаний были удалены user_id, которых не было в декабре и редкие user_id, определенные на этапе EDA (4, 51, 52).
- Из списка вероятностей для слова берется первый кортеж: предсказанный user_id (поле «pred»), вероятность «p_value» и количество user_id (поле «p_count»).
- Таблица сортируется по убыванию p_value и возрастанию pred.

user_word	pred	p_value	p_count	p_values
binary	12	1,00000	291	[(12, 1.0, 291)]
logistic	36	1,00000	5	[(36, 1.0, 5)]
aucroc	24	0,95122	39	[(24, 0.95122, 39), (25, 0.04878, 2)]
sigmoid	55	0,90807	405	[(55, 0.90807, 405), (0, 0.04709, 21), (11, 0.03587, 16), (46, 0.00897, 4)]
categorical	14	0,81818	207	[(14, 0.81818, 207), (23, 0.25253, 31), (15, 0.02767, 7), (1, 0.01581, 4), (54, 0.01581, 4)]

Формирование сабмита

user_word	pred	p_value	p_count	p_values
binary	12	1,00000	291	[(12, 1.0, 291)]
logistic	36	1,00000	5	[(36, 1.0, 5)]
aucroc	24	0,95122	39	[(24, 0.95122, 39), (25, 0.04878, 2)]
sigmoid	55	0,90807	405	[(55, 0.90807, 405), (0, 0.04709, 21), (11, 0.03587, 16), (46, 0.00897, 4)]
categorical	14	0,81818	207	[(14, 0.81818, 207), (23, 0.25253, 31), (15, 0.02767, 7), (1, 0.01581, 4), (54, 0.01581, 4)]
loss	19	0,80488	396	[(19, 0.80488, 396), (6, 0.06911, 34), (25, 0.06098, 30), (12, 0.03049, 15), (24, 0.02033, 10), (37, 0.01423, 7)]
f1	6	0,79539	552	[(6, 0.79539, 552), (37, 0.15994, 111), (19, 0.02161, 15), (11, 0.01297, 9), (33, 0.01009, 7)]
recall	3	0,74874	149	[(3, 0.74874, 149), (35, 0.12563, 25), (39, 0.07538, 15), (9, 0.03015, 6), (50, 0.01508, 3),
ols	11	0,72414	105	[(11, 0.72414, 105), (32, 0.11724, 17), (37, 0.06897, 10), (29, 0.04138, 6), (54, 0.04138, 6),
mse	43	0,68687	68	[(43, 0.68687, 68), (23, 0.25121, 12), (35, 0.15152, 15), (25, 0.0404, 4)]
distributed	0	0,65421	70	[(0, 0.65421, 70), (1, 0.1215, 13), (11, 0.11215, 12), (35, 0.11215, 12)]
precision	12	0,64815	35	[(12, 0.64815, 35), (0, 0.35185, 19)]
pvalue	32	0,60510	95	[(32, 0.6051, 95), (50, 0.17197, 27), (43, 0.09554, 15), (54, 0.0828, 13), (46, 0.04459, 7)]
blue	14	0,58333	7	[(14, 0.58333, 7), (56, 0.41667, 5)]

- Рассмотрим на примере p_value = 1, binary – предсказанный user_id = 12.
- Для других слов 12 удаляется и выбирается следующий кортеж из списка, т.е. для precision данные заполняется такими значениями (0, 0.35185, 19).
- Таблица сортируется, цикл повторяется пока есть «pred», отличные от -999. На одном из этапов цикла для precision не будет значений в списке, т.к. pred = 0 используется у distributed и такому слову проставляется -999.
- Аналогично происходит для blue значения заменяются на (56, 0.41667, 5).

Формирование сабмита

Что делать со словами у которых pred = -999 ???

- Ранее были посчитаны статистики по каждому пользователю:

1. Количество пройденных турникетов за день;
2. Время прихода, время ухода, длительность нахождения в офисе;
3. Среднее, медиана и стандартное отклонение.

- На сцену выходит функция **cdist** из пакета scipy.spatial.distance для расчета расстояний между векторами:

- Для оставшихся слов и нераспределенными user_id были посчитаны расстояния между векторами слова и user_id с использованием метрик, которые выдавали вменяемые результаты: 'euclidean', 'cosine', 'chebyshev', 'correlation', 'minkowski’, 'cityblock', 'canberra', 'braycurtis', 'matching’. По минимальному расстоянию определяется user_id и берется самый частотный.

Формирование сабмита

Что делать со словами у которых pred = -999 ???

- Во время экспериментов с признаками и параметрами модели количество слов с -999 получалось от 2 до 5.
- В качестве эксперимента все слова были предсказаны по такому алгоритму и был получен результат = 36 (3 слова: categorical, gini, target). Т.е. без машинного обучения, только на статистиках пользователя.
- На этапе, когда случайный лес стал выдавать результат лучше линейной регрессии (более 300), был создан сабмит на основе предсказаний 7 лучших моделей, обученных на разных данных, это аналог комитета классификаторов, который выдал результат 447.
- На основе этого сабмита выбирались признаки и настройки для деревянной модели. Результат = 414, т.е. модель не сопоставила 4 слова: 2 существующих пользователя independent и residual (+33 за слово) и 2 пасхалки, которые нашел Александр: regression и y (+40 за слово).