Геоаналитика
Разбор задачи

2024.02.29
DataFusion Contest 2024

Задача

- 8157 локаций. В 1657 есть банкоматы. В 8154 совершались оплаты.
- Даны транзакции клиента (кроме снятия наличных)
- Нужно для каждой из 1657 локаций предсказать вероятность снятия наличных клиентом

Целевая переменная

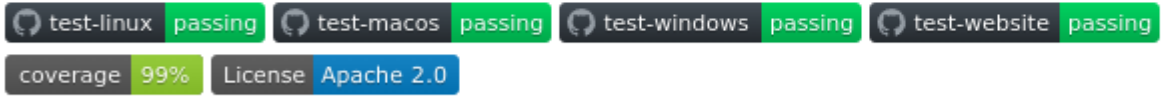
- h3_09 — локация использования банкомата (string)
- customer_id — код клиента (int)

```
target = pd.read_parquet(data_root / 'target.parquet')
target.head()
```

	h3_09	customer_id
0	8911aa6ac3bffff	23172
1	8911aa7a857ffff	95640
2	8911aa70b97ffff	60350
3	8911aa70b97ffff	69521
4	891181b69abffff	29437

https://github.com/uber/h3

H3: A Hexagonal Hierarchical Geospatial Indexing System

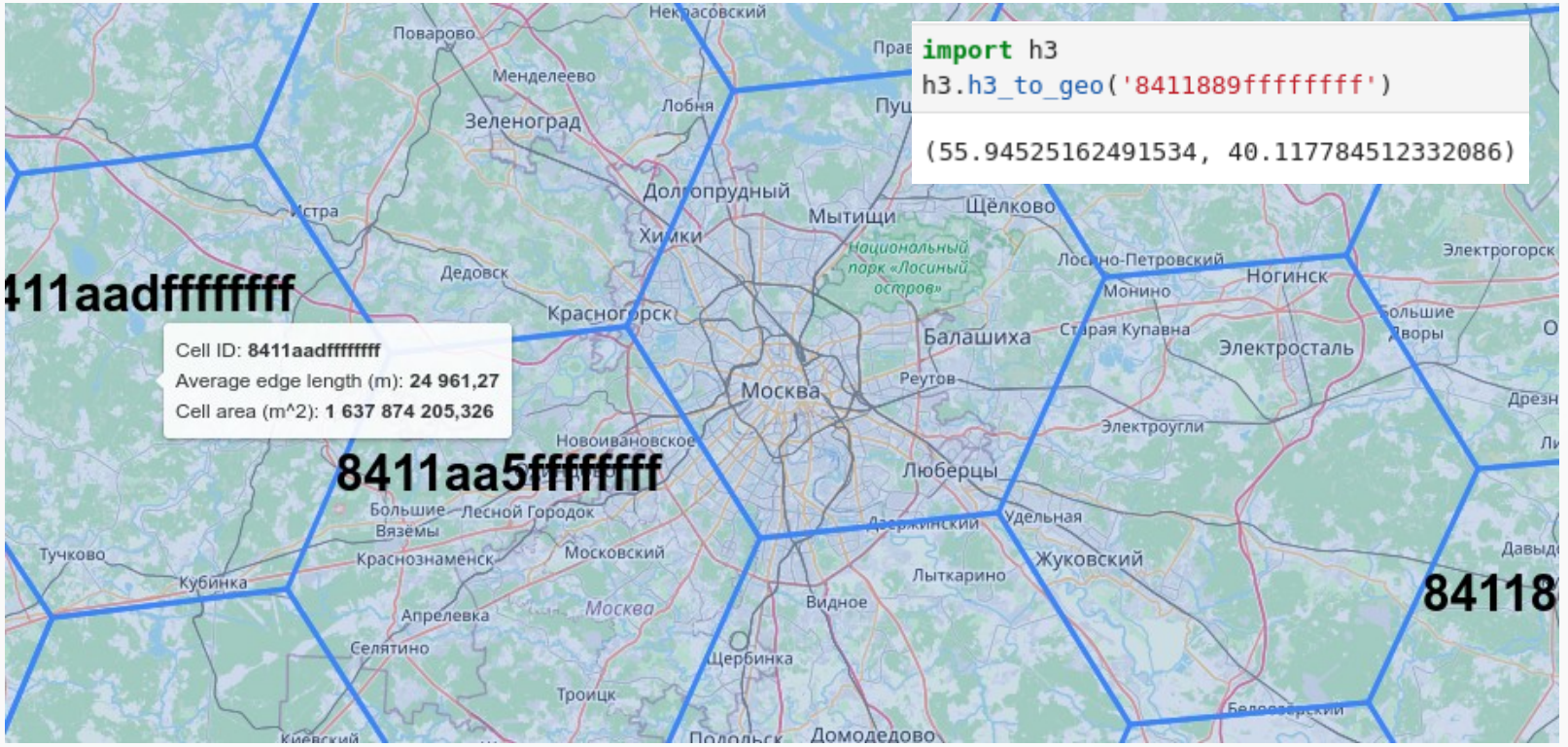


H3 is a geospatial indexing system using a hexagonal grid that can be (approximately) subdivided into finer and finer hexagonal grids, combining the benefits of a hexagonal grid with S2's hierarchical subdivisions.

Documentation is available at https://h3geo.org/. Developer documentation in Markdown format is available under the dev-docs directory.

- Post **bug reports or feature requests** to the GitHub Issues page
- Ask **questions** by posting to the H3 tag on StackOverflow
- There is also an H3 Slack workspace

Вложенные шестиугольники



https://wolf-h3-viewer.glitch.me/

Чтение данных

```
with open(data_root / "hexses_target.lst", "r") as f:
    hexses_target = [x.strip() for x in f.readlines()]

with open(data_root / "hexses_data.lst", "r") as f:
    hexses_data = [x.strip() for x in f.readlines()]

transactions = pd.read_parquet(data_root / 'transactions.parquet')
target = pd.read_parquet(data_root / 'target.parquet')
```

Как выглядит таргет

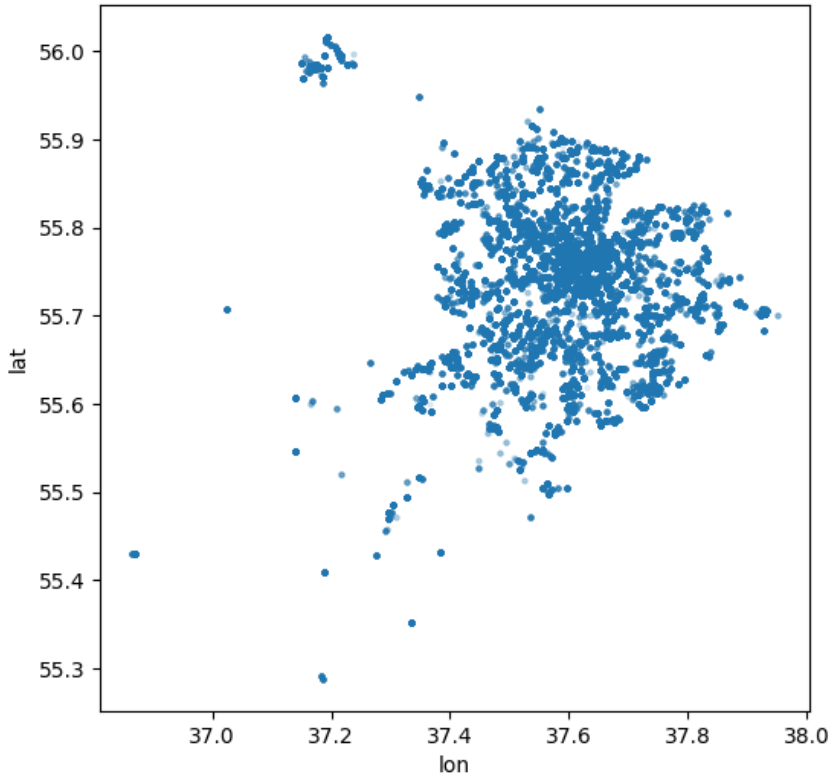
```
coord_target = target.assign(
    lat = target['h3_09'].apply(lambda x: h3.h3_to_geo(x)[0]),
    lon = target['h3_09'].apply(lambda x: h3.h3_to_geo(x)[1]),
)
```

```
coord_target.head()
```

	h3_09	customer_id	lat	lon
0	8911aa6ac3bffff	23172	55.753226	37.820995
1	8911aa7a857ffff	95640	55.758045	37.595189
2	8911aa70b97ffff	60350	55.884192	37.605146
3	8911aa70b97ffff	69521	55.884192	37.605146
4	891181b69abffff	29437	55.646191	37.718121

Расположение банкоматов

```
coord_target.plot(x='lon', y='lat', kind='scatter', marker='.', alpha=0.1, figsize=(6, 6));
```



Как выглядят транзакции

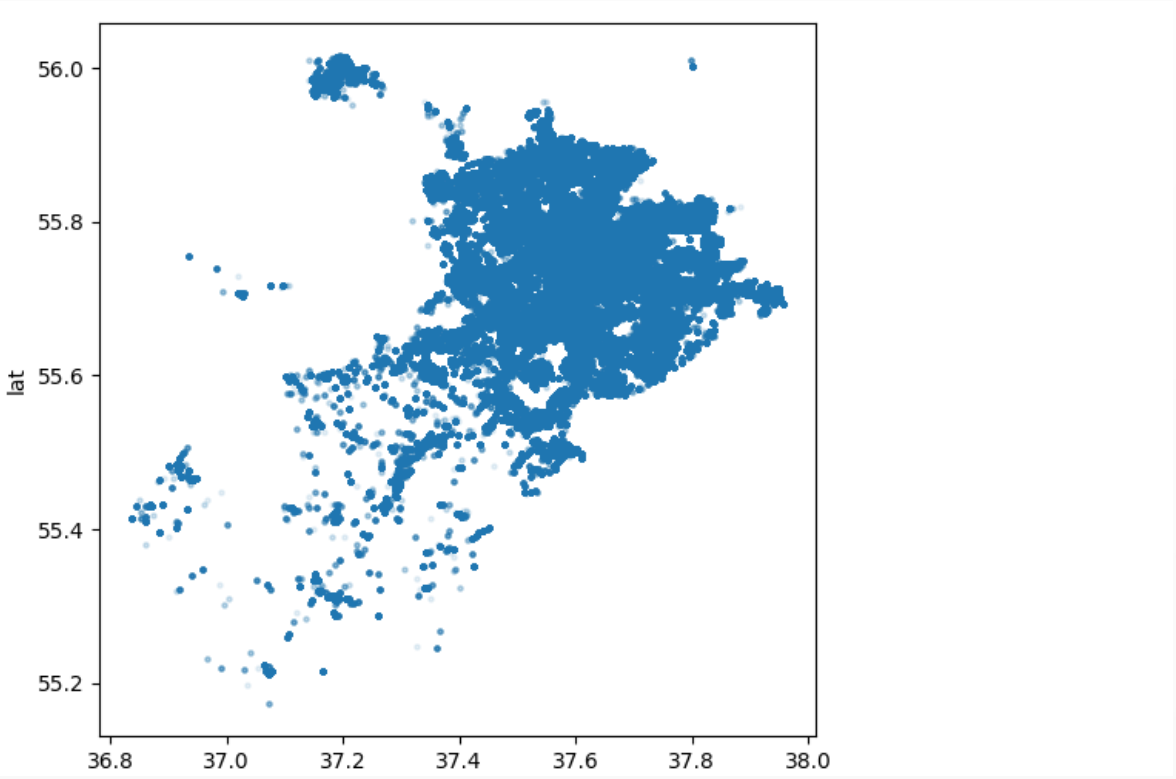
```
coord_transactions = transactions.assign(
    lat = transactions['h3_09'].apply(lambda x: h3.h3_to_geo(x)[0]),
    lon = transactions['h3_09'].apply(lambda x: h3.h3_to_geo(x)[1]),
)
```

```
coord_transactions.head()
```

	h3_09	customer_id	datetime_id	count	sum	avg	min	max	std	count_distinct	mcc_code	lat	lon
0	8911aa4c62fffff	1	3	1	3346.65	3346.650	3346.65	3346.65	NaN	1	13	55.565522	37.445024
1	8911aa7b5b3ffff	4	3	1	450.00	450.000	450.00	450.00	NaN	1	8	55.699940	37.502864
2	8911aa63623ffff	5	3	10	11035.69	1103.569	59.00	3620.18	1190.530333	6	13	55.784319	37.665680
3	8911aa48577ffff	9	2	2	628.00	314.000	295.00	333.00	26.870058	2	5	55.472146	37.294992
4	8911aa78297ffff	11	2	1	4155.00	4155.000	4155.00	4155.00	NaN	1	10	55.691497	37.602534

Транзакции на карте

```
coord_transactions.plot(x='lon', y='lat', kind='scatter', marker='.', alpha=0.1, figsize=(6, 6));
```

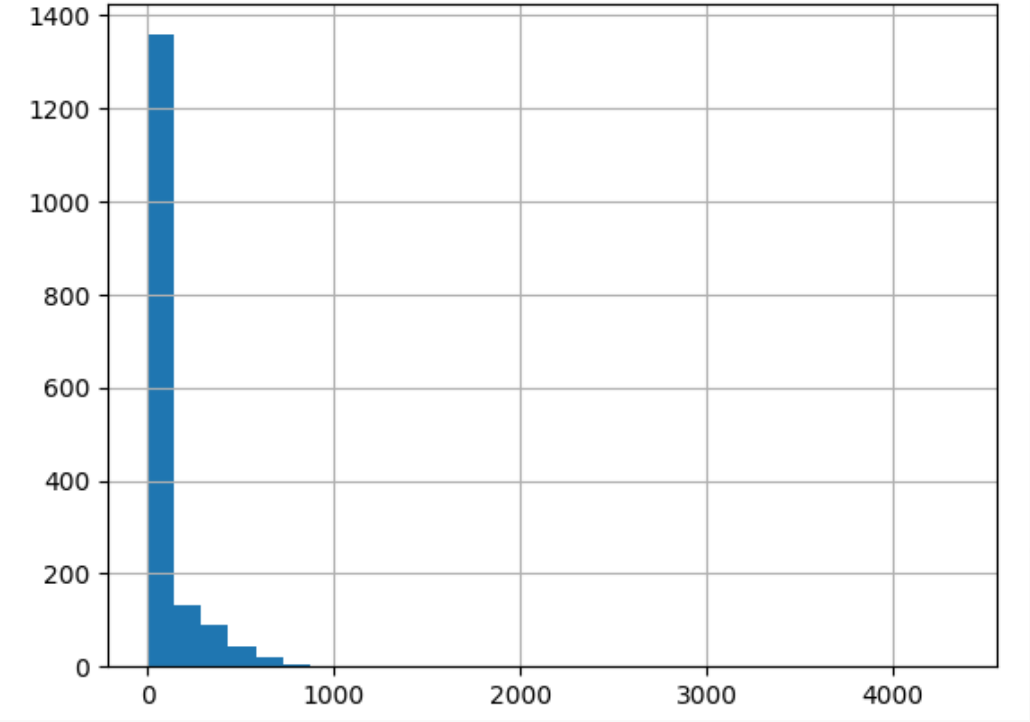


Локальная валиадция

- Кросс-валидация на 3-х фолдах
- Как нам равномерно распределить клиентов?
- Чем они вообще отличаются?
  - Местами, где бывают
  - Количеством банкоматов, в которых снимают

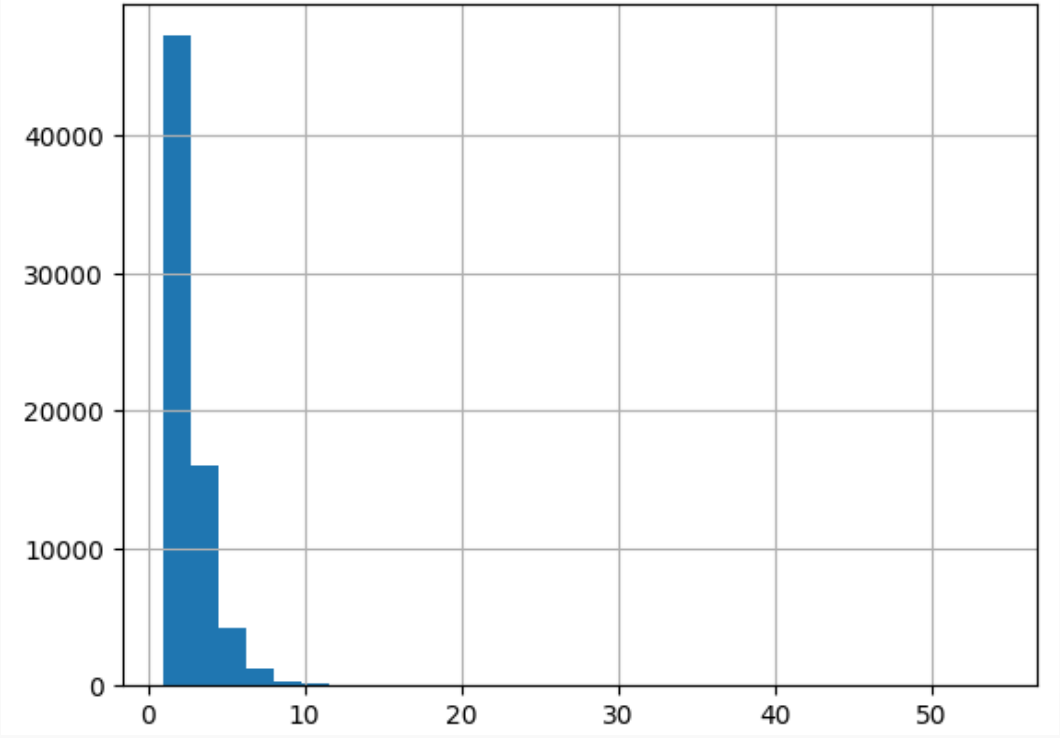
Разнообразие мест

```
target.groupby(by='h3_09').size().hist(bins=30);
```



Разнообразие банкоматов

```
target.groupby(by='customer_id').size().hist(bins=30);
```



Разбиение на фолды

- Один пользователь снимал в 54 банкоматах
- Большинство — в двух или трех
- Равномерно распределим пользователей с учетом количества банкоматов, которыми они пользуются
- Поделим количество багнкоматов на квантили
- StratifiedKFold по квантилям

Заготовка валидации

```
num_split = 3
num_quantiles = 10
customers = (
    pd.DataFrame({"cnt": target.groupby(by='customer_id').size()})
    .pipe(lambda x: x.assign(bin=pd.qcut(x.cnt, num_quantiles, duplicates='drop')))
)
for ids_test, ids_train in tqdm(
        StratifiedKFold(n_splits=num_split, shuffle=True, random_state=20240225)
        .split(customers.index, pd.Categorical(customers.bin).codes),
        total=num_split
):
    transactions_tran = transactions.loc[transactions.customer_id.isin(ids_train)]
    target_tran = target.loc[target.customer_id.isin(ids_train)]
```

Самая простая модель

- Посчитаем, как часто в среднем пользовались банкоматом
- Предскажем всем среднее значение
- Получим на валидации
  - 15.595103709664691
  - 15.605737138344905
  - 15.693970284502669
- Зато быстро

Лосс по строкам и столбцам

```
row_score = (
    -np.log(predict.clip(1e-8, 1 - 1e-8)) * labels
    -np.log(1-predict.clip(1e-8, 1 - 1e-8)) * (1 - labels)
).sum(axis=1)
col_score = (
    -np.log(predict.clip(1e-8, 1 - 1e-8)) * labels
    -np.log(1-predict.clip(1e-8, 1 - 1e-8)) * (1 - labels)
).sum(axis=0)

score = row_score.mean()
```

Неравномерный

```
col_score.describe()
```

```
count    1657.000000
mean      309.662102
std       468.967378
min         0.000329
25%        73.683052
50%       130.224162
75%       303.044738
max      7934.145782
dtype: float64
```

```
row_score.describe()
```

```
count    32902.000000
mean        15.595104
std          9.869358
min          5.041200
25%          8.157561
50%         12.988631
75%         19.312626
max        292.072820
dtype: float64
```

Метрики для одного из фолдов

Ошибки и находки

```
col_score.sort_values(ascending=False)[:10].index
```

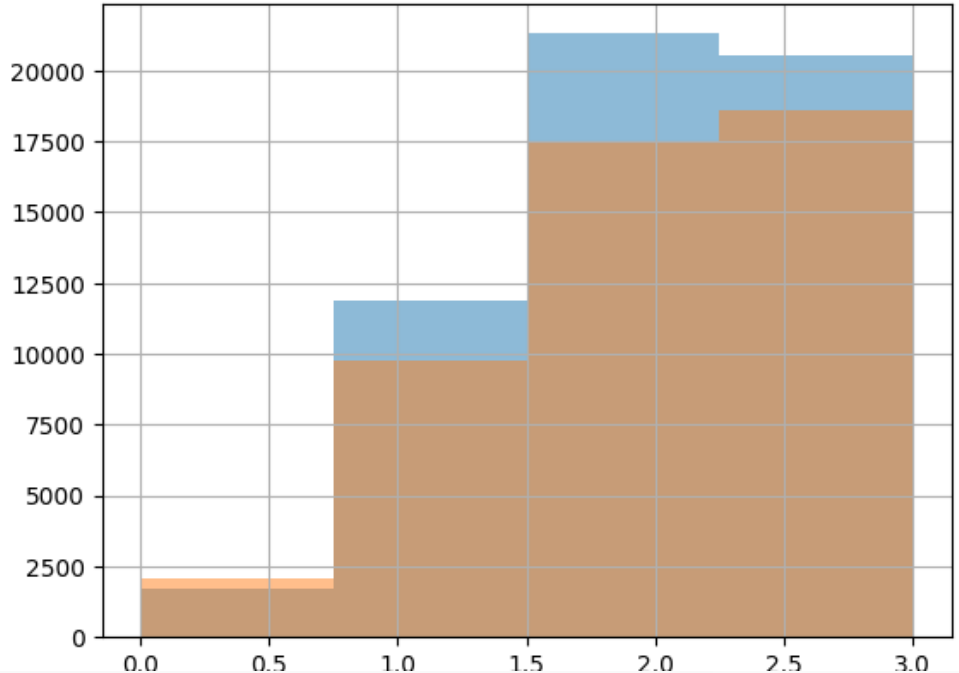
```
Index(['8911aa7abd3ffff', '8911aa7a117ffff', '8911aa7ab43ffff',
       '8911aa7aaaffffff', '8911aa7a967ffff', '8911aa7ae2bffff',
       '8911aa7aa0fffff', '8911aa6360bffff', '891181b6a37ffff',
       '8911aa78d8bffff'],
      dtype='object')
```

```
raw_score.sort_values(ascending=False)[:10].index
```

```
Int64Index([20772, 22639, 54861, 57045, 28064, 46170, 33689, 9652, 27369,
            19659],
           dtype='int64')
```

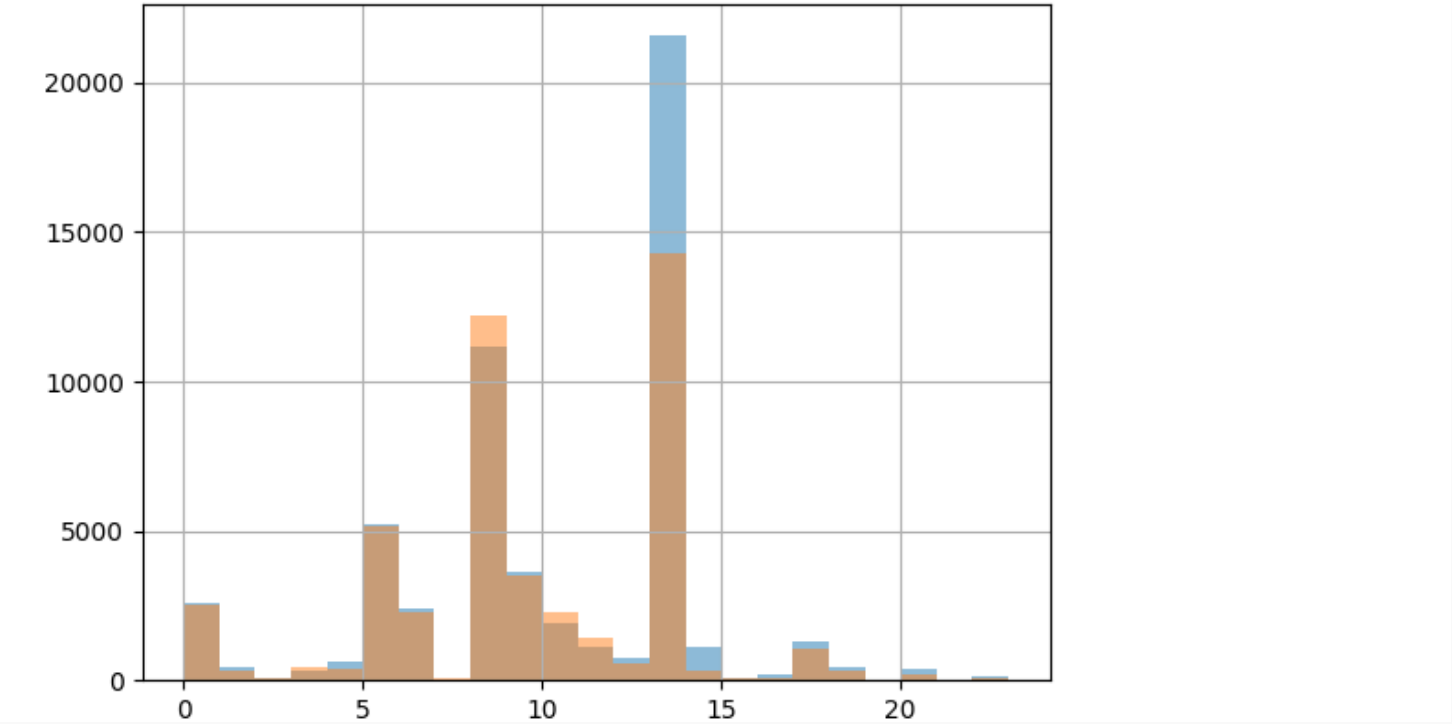
Отличается режим дня

```
transactions[transactions.customer_id.isin(bad_customers)]['datetime_id'].hist(bins=4, alpha=0.5);
transactions[transactions.customer_id.isin(good_customers)]['datetime_id'].hist(bins=4, alpha=0.5);
```

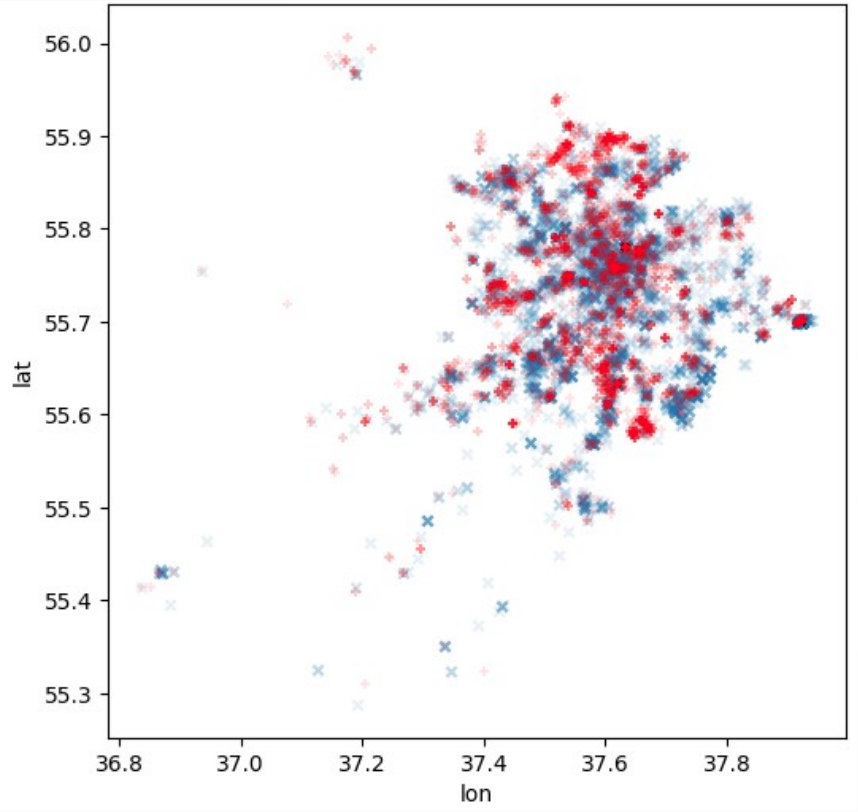


Отличаются покупки

```
transactions[transactions.customer_id.isin(bad_customers)]['mcc_code'].hist(bins=23, alpha=0.5);
transactions[transactions.customer_id.isin(good_customers)]['mcc_code'].hist(bins=23, alpha=0.5);
```



Локации с ошибками



Наблюдения

- Клиенты совершают мелкие покупки у дома
- Клиенты снимают деньги в «шумных местах»
- Есть клиенты, ориентированные на наличные

См SimpleFeaturesTransform

```
t = SimpleFeaturesTransform()
t.fit(transactions_tran)
X = t.transform(transactions_tran)
X.head()
```

	dt_0	dt_1	dt_2	dt_3	hex_8911818610bffff	hex_89118195133ffff	hex_8911819513bffff	hex_891181b2827ffff	hex_8
5	0.0	187.0	297.0	473.0	0.0	0.0	0.0	0.0	
23	0.0	99.0	308.0	209.0	0.0	0.0	0.0	0.0	
31	11.0	110.0	231.0	143.0	0.0	0.0	0.0	0.0	
33	11.0	231.0	253.0	176.0	0.0	0.0	0.0	0.0	
42	0.0	99.0	132.0	242.0	0.0	0.0	0.0	0.0	

5 rows × 1685 columns