

В **этом ноутбуке** анализируется, как велопрокатом пользуются владельцы различных абонементов

```
1 import pandas as pd
2 import numpy as np
3 from sklearn import preprocessing
4 import scipy.stats as sps
5 from tqdm import tqdm
6 import datetime
7 from statsmodels.stats.multitest import multipletests
8 import matplotlib.pyplot as plt
9
10 import plotly.graph_objects as go
11 import plotly.express as px
12 import plotly.offline
13
14
15 from mpl_toolkits.basemap import Basemap
```

## ▼ 0. Загрузка данных

```
1 trips = pd.read_csv('cycle-share-dataset/trip.csv', error_bad_lines=False,
2                     parse_dates=[1, 2])
```

 b'Skipping line 50794: expected 12 fields, saw 20\n'

label encoding и парсинг временных признаков


```
1 le = preprocessing.LabelEncoder()
2 le_trips = trips.copy()
3 le_trips.bikeid = le.fit_transform(le_trips.bikeid)
4 le_trips.from_station_id = le.fit_transform(le_trips.from_station_id)
5 le_trips.to_station_id = le.transform(le_trips.to_station_id)
6 le_trips.usertype = (le_trips.usertype.values == 'Member').astype(int)
7 le_trips.gender = (le_trips.gender == 'Male').astype(int)
8
9 le_trips['start_year'] = le_trips['starttime'].dt.year
10 le_trips['start_month'] = le_trips['starttime'].dt.month
11 le_trips['start_day'] = le_trips['starttime'].dt.day
12 le_trips['start_date'] = le_trips['starttime'].dt.date
13 le_trips['start_time'] = le_trips['starttime'].dt.time
14 le_trips['start_weekday'] = le_trips['starttime'].dt.dayofweek
15
16 le_trips['stop_year'] = le_trips['stoptime'].dt.year
17 le_trips['stop_month'] = le_trips['stoptime'].dt.month
18 le_trips['stop_day'] = le_trips['stoptime'].dt.day
19 le_trips['stop_date'] = le_trips['stoptime'].dt.date
20
```

```

21 le_trips['stop_time'] = le_trips['stoptime'].dt.time
22 le_trips['stop_weekday'] = le_trips['stoptime'].dt.dayofweek
23
24 le_trips.fillna(-1, inplace=True)

```


```
1 le_trips.head()
```



	trip_id	starttime	stoptime	bikeid	tripduration	from_station_name	to_station_name
0	431	2014-10-13 10:31:00	2014-10-13 10:48:00	289	985.935	2nd Ave & Spring St	Occidental Parl Occidental Ave & S Washing
1	432	2014-10-13 10:32:00	2014-10-13 10:48:00	186	926.375	2nd Ave & Spring St	Occidental Parl Occidental Ave & S Washing
2	433	2014-10-13 10:33:00	2014-10-13 10:48:00	477	883.831	2nd Ave & Spring St	Occidental Parl Occidental Ave & S Washing
3	434	2014-10-13 10:34:00	2014-10-13 10:48:00	324	865.937	2nd Ave & Spring St	Occidental Parl Occidental Ave & S Washing
4	435	2014-10-13 10:34:00	2014-10-13 10:49:00	193	923.923	2nd Ave & Spring St	Occidental Parl Occidental Ave & S Washing

5 rows × 24 columns

```
1 le_trips.columns
```



```

Index(['trip_id', 'starttime', 'stoptime', 'bikeid', 'tripduration',
      'from_station_name', 'to_station_name', 'from_station_id',
      'to_station_id', 'usertype', 'gender', 'birthyear', 'start_year',
      'start_month', 'start_day', 'start_date', 'start_time', 'start_weekday',
      'stop_year', 'stop_month', 'stop_day', 'stop_date', 'stop_time',
      'stop_weekday'],
      dtype='object')

```

## ▼ 1. Анализ зависимостей

Посмотрим, как разные признаки влияют на наличие абонеента.

Анализ будем осуществлять с помощью двух критериев:

1) Для категориальных признаков классический хи-квадрат.

2) Для вещественных признаков критерий Манна-Уитни: в зависимости от наличия абонемен выборки и проверять гипотезу о равенстве их распределений. В случае отвержения гипотезы

наличие абонемента.

## Категориальные признаки:

```
1 pvalues = []
2
3 cat_features = ['from_station_id', 'to_station_id', 'gender',
4                 'birthyear',
5                 'start_year', 'start_month', 'start_weekday']
6
7 for feat in cat_features:
8     print(f'Признак {feat}')
9
10    obs = pd.crosstab(le_trips[feat],
11                      le_trips['usertype'])
12    chi2, p, dof, expected = sps.chi2_contingency(obs)
13    if (expected < 5).mean() < 0.2:
14        print(f'p-value: {p}')
15        pvalues.append(p)
16    else:
17        print('Критерий неприменим')
18        pvalues.append(-1)
19    print()
```



Признак from\_station\_id  
p-value: 0.0

Признак to\_station\_id  
p-value: 0.0

Признак gender  
p-value: 0.0

Признак birthyear  
p-value: 0.0

Признак start\_year  
p-value: 1.4820725743292082e-296

Признак start\_month  
p-value: 0.0

Признак start\_weekday  
p-value: 0.0

## Вещественные признаки

Вещественный признак здесь только один - tripduration.

```
1 st, p = sps.mannwhitneyu(le_trips[le_trips.usertype == 1]['tripduration'],
2                           le_trips[le_trips.usertype == 0]['tripduration'])
```

```

2         group_trips[group_trips.usertype == 0]['tripduration']
3
4 pvalues.append(p)

```

## Итог:

```

1 multipletests(pvalues)

```

```

(
  array([ True,  True,  True,  True,  True,  True,  True,  True]),
  array([0., 0., 0., 0., 0., 0., 0., 0.]),
  0.006391150954545011,
  0.00625)

```

Для всех признаков гипотеза о независимости отверглась, а значит все они влияют на налич

## ▼ 2. Абонементы у участников групповых поездок

```

1 group_trips = pd.read_csv('cycle-share-dataset/trips_w_groups.csv')
2 group_trips = group_trips.drop(columns=['Unnamed: 0'])
3
4 group_count = pd.DataFrame(group_trips.groupby(by = 'group_id').
5                             count()['trip_id'])
6
7 group_count.columns = ['members_num']
8 group_count.reset_index(inplace=True)
9
10 group_trips = group_trips.merge(group_count,
11                                how='left', on='group_id')

```

```

1 np.unique(group_trips.members_num)

```

```

array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 12, 14, 16, 20, 24])

```

```

1 trips = group_trips.copy()
2 single_trips = group_trips[group_trips.members_num == 1]
3 group_trips = group_trips[group_trips.members_num > 1]

```

Посмотрим на распределение наличия абонементов у тех, кто катается один и у тех, кто катае

```

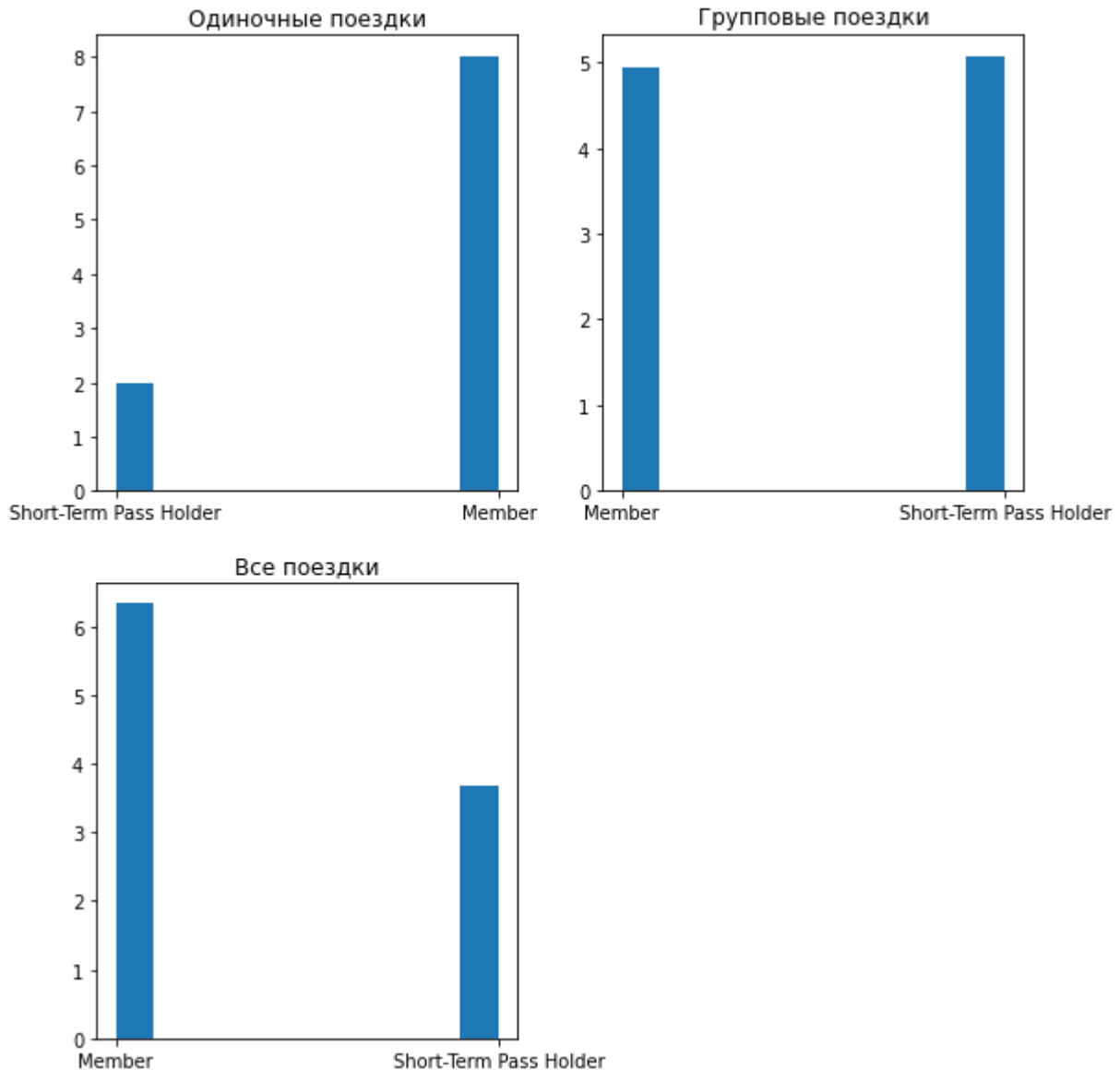
1 plt.figure(figsize=(9, 10))
2
3 plt.subplot(221)
4 plt.title('Одиночные поездки')
5 plt.hist(single_trips.usertype.values, density=True)
6
7 plt.subplot(222)
8 plt.title('Групповые поездки')

```

```

9 plt.hist(group_trips.usertype.values, density=True)
10
11 plt.subplot(223)
12 plt.title('Все поездки')
13 plt.hist(trips.usertype.values, density=True)
14
15 plt.show()

```



**Наблюдение:** удивительно, но больше половины участников групповых поездок не владеют и катаются по одиночке, в подавляющем большинстве владеют абонементом.

При этом в ноутбуке по исследованию групповых поездок мы выяснили, что они составляют поэтому можно посоветовать компании придумать способ рекламы абонементов участникам, например скидки по акции "приведи друга".

### ▼ 3. Визуализация маршрутов владельцев абонементов

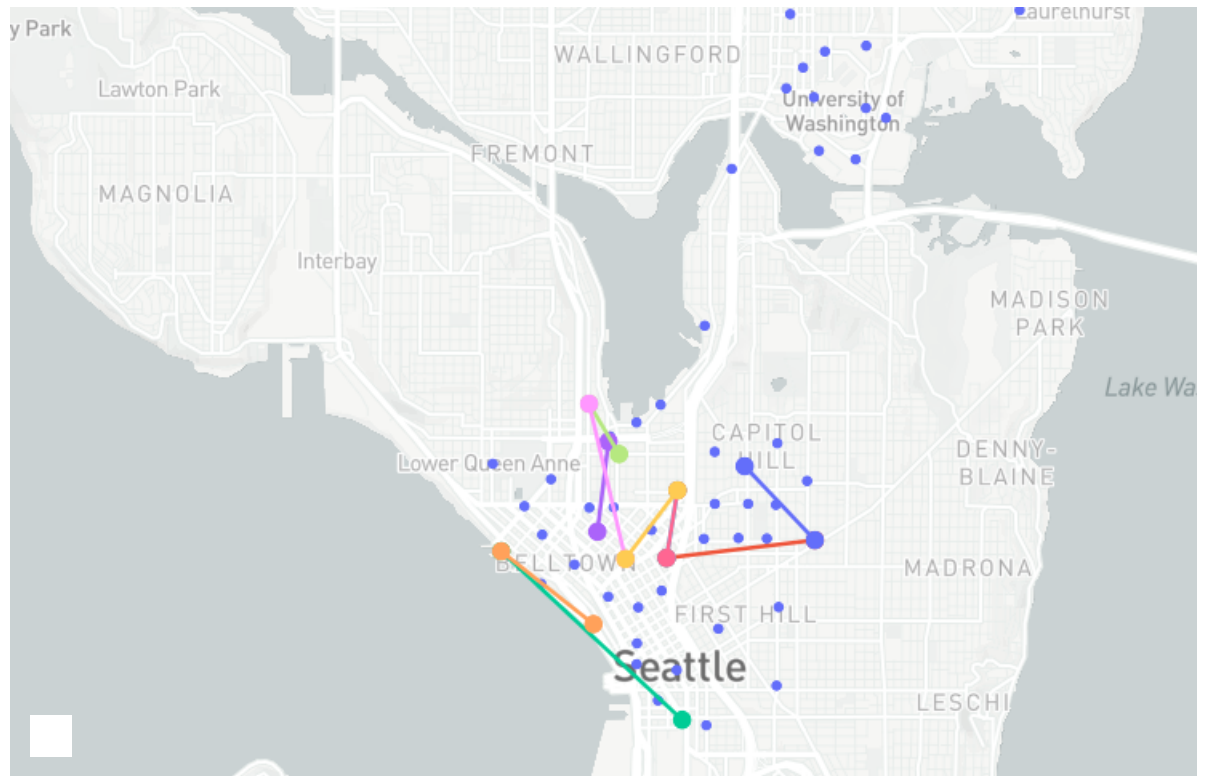
Посмотрим, по каким маршрутам обычно катаются люди без абонементов и люди с абоне

```
1 stations = pd.read_csv('cycle-share-dataset/station.csv')

1 def most_popular(trip):
2     d = {}
3
4     for route in trip.values:
5         from_to = (route[7], route[8])
6         if from_to in d:
7             d[from_to] += 1
8         else:
9             d[from_to] = 1
10
11     d = {k: v for k, v in sorted(d.items(), key=lambda item: item[1])[:-1]}
12
13     return d
14
15
16 def plot_popular(popular_dict, n=10):
17     px.set_mapbox_access_token(open("public_key").read())
18     fig = px.scatter_mapbox(stations, lat="lat", lon="lon",
19                             zoom=11)
20     top = list(popular_dict.keys())[:n]
21
22     for route in top:
23         st_from = route[0]
24         st_to = route[1]
25         from_lon = stations[stations.station_id == st_from].lon.values[0]
26         from_lat = stations[stations.station_id == st_from].lat.values[0]
27         to_lon = stations[stations.station_id == st_to].lon.values[0]
28         to_lat = stations[stations.station_id == st_to].lat.values[0]
29
30         fig.add_trace(go.Scattermapbox(mode = "markers+lines",
31                                         lon = [from_lon, to_lon],
32                                         lat = [from_lat, to_lat],
33                                         marker = {'size': 10}))
34
35     fig.show()

1 member_popular = most_popular(trips[trips.usertype == 'Member'])
2 plot_popular(member_popular, 10)
```





**Наблюдение:** владельцы абонементов часто катаются и в "деловой", и в "развлекательной" ча

```
1 not_member_popular = most_popular(trips[trips.usertype != 'Member'])
2 plot_popular(not_member_popular, 10)
```







