Детекция "перебросов"

Ноутбук по нахождению "перебросов" велосипедов с одной станции на другую сотрудниками компании

Загрузка датасета

In [31]:
```
import scipy as sps
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from tqdm import tqdm_notebook
import warnings
from sklearn.preprocessing import LabelEncoder

warnings.filterwarnings('ignore')
```

In [32]:
```
trips = pd.read_csv('cycle-share-dataset/trip.csv', error_bad_lines=False)

trips.starttime = trips.starttime.map(lambda x: pd.to_datetime(x))
trips.stoptime = trips.stoptime.map(lambda x: pd.to_datetime(x))

print(trips.shape)
```

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

(286857, 12)
```

In [33]:
```
trips.head()
```

Out[33]:

	trip_id	starttime	stoptime	bikeid	tripduration	from_station_name	to_station_name	from_s
0	431	2014-10-13 10:31:00	2014-10-13 10:48:00	SEA00298	985.935	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washing...	
1	432	2014-10-13 10:32:00	2014-10-13 10:48:00	SEA00195	926.375	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washing...	
2	433	2014-10-13 10:33:00	2014-10-13 10:48:00	SEA00486	883.831	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washing...	
3	434	2014-10-13 10:34:00	2014-10-13 10:48:00	SEA00333	865.937	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washing...	
4	435	2014-10-13 10:34:00	2014-10-13 10:49:00	SEA00202	923.923	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washing...	

In [34]:
```
trips.dtypes
```

Out[34]:
```
trip_id                     int64
starttime           datetime64[ns]
stoptime            datetime64[ns]
bikeid                      object
tripduration               float64
from_station_name           object
to_station_name             object
from_station_id             object
to_station_id               object
usertype                    object
gender                      object
birthyear                  float64
dtype: object
```

Поиск "перебросов"

1) Для каждого велосипеда будем хранить список индексы всех его поездок в хронологическом порядке

2) Проходясь по списку поездок будем смотреть на пары (откуда, куда). Если встречается две подряд идущих пары (a, b) (c, d), причем b≠c, то со станции b на станцию c был осуществлен переброс.

3) Сохраним все перебросы в отдельном pandas датафрейме

In [35]:
```
# проверим, совпадают ли уникальные значения станций в столбцах "откуда" и "куда
set(np.unique(np.array(trips.from_station_name))) == set(np.unique(np.array(trips
```

Out[35]:
```
True
```

In [36]:
```
%%time

# label encoding станций
le = LabelEncoder()
trips.from_station_name = le.fit_transform(trips.from_station_name)
trips.to_station_name = le.transform(trips.to_station_name)
```

```
Wall time: 96.9 ms
```

In [37]:
```
# ключ - bikeid, значение - список поездок
d = {}

X = np.array(trips)
for i, trip in tqdm_notebook(enumerate(X)):
    bike_id = trip[3]

    if bike_id in d:
        d[bike_id].append(i)
    else:
        d[bike_id] = [i]
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [38]:
```
# по списку поездок выдаём список перебросов
def check_bike_trips(arr):
    transfers = []

    trips_list = []
    for i in arr:
        from_station = X[i][5]
        to_station = X[i][6]
        trips_list.append((from_station, to_station))

    for i in range(1, len(arr)):
        cur_trip = trips_list[i]
        prev_trip = trips_list[i-1]
        if cur_trip[0] != prev_trip[1]:
            prev_trip_id = arr[i-1]
            cur_trip_id = arr[i]
            transfers.append([
                X[prev_trip_id][2], # время конца предыдущей поездки
                X[cur_trip_id][1], # время начала текущей поездки
                le.inverse_transform([X[prev_trip_id][6]])[0], # на какой станци
                le.inverse_transform([X[cur_trip_id][5]])[0] # на какой оказался
            ])

    return transfers
```

In [39]:
```
transfers_dict = {}

for key in d:
    transfers_dict[key] = check_bike_trips(d[key])
```

In [40]:
```
transfers = []

for key in transfers_dict:
    for transfer in transfers_dict[key]:
        transfers.append([key] + transfer)
```

In [43]:
```
print(f'Найдено перебросов: {len(transfers)}')
print(f'Отношение кол-ва поездок к перебросам: {len(X) / len(transfers)}')
```

```
Найдено перебросов: 68413
Отношение кол-ва поездок к перебросам: 4.193018870682472
```

Получилось, что перебросов довольно много: на каждый 4 поездки в среднем приходится один переброс. Наверняка компания может изменить конфигурации станций, чтобы таких перебросов было меньше.

Чтобы сохранить перебросы в отдельный датасет, получим еще словарь, который мэтчит названия станций с их id.

In [50]:
```
name_id = {}
stations = pd.read_csv('cycle-share-dataset/station.csv', error_bad_lines=False)

for id_, name in zip(np.array(stations.station_id), np.array(stations.name)):
    if name in name_id:
        continue
    else:
        name_id[name] = id_
```

Добавим айдишники в список перебросок

In [68]:
```
transfers_with_id = []

for transfer in tqdm_notebook(transfers):
    # исключаем магазины, нас интересуют станции
    if transfer[3][:11] == 'Pronto shop' or transfer[4][:11] == 'Pronto shop':
        continue
    transfers_with_id.append(transfer + \
                             [name_id[transfer[3]], name_id[transfer[4]]])
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [70]:
```
df = pd.DataFrame(transfers_with_id)
df.columns = ['bikeid', 'starttime', 'stoptime',
              'from_station_name', 'to_station_name',
              'from_station_id', 'to_station_id']
df.head()
```

Out[70]:

	bikeid	starttime	stoptime	from_station_name	to_station_name	from_station_id	to_station_
0	SEA00298	2014-10-13 12:05:00	2014-10-13 20:03:00	1st Ave & Marion St	2nd Ave & Pine St	CBD-05	CBD-
1	SEA00298	2014-10-15 10:42:00	2014-10-16 00:10:00	Eastlake Ave E & E Allison St	Cal Anderson Park / 11th Ave & Pine St	EL-05	CH-(
2	SEA00298	2014-10-16 08:52:00	2014-10-20 11:04:00	2nd Ave & Pine St	Pier 69 / Alaskan Way & Clay St	CBD-13	WF-(
3	SEA00298	2014-10-20 11:55:00	2014-10-23 16:28:00	3rd Ave & Broad St	PATH / 9th Ave & Westlake Ave	BT-01	SLU-(
4	SEA00298	2014-10-26 02:31:00	2014-10-29 07:55:00	6th Ave S & S King St	12th Ave & E Denny Way	ID-04	CH-(

In [72]:
```
df.to_csv('cycle-share-dataset/transfers.csv')
```