

МГТУ им. Н. Э. Баумана, кафедра ИУ5  
курс “Методы машинного обучения”

Лабораторная работа №6  
«Создание рекомендательной модели»

ВЫПОЛНИЛ:

Лычагин Д.А.

Группа: ИУ5-24М

ПРОВЕРИЛ:

Гапанюк Ю.Е.

Москва 2022

**Цель лабораторной работы:** обучение работе с предварительной обработкой графовых типов данных и обучением нейронных сетей на графовых данных.

**Задание:**

1. Подготовить датасет графовых данных
2. Подобрать модель и гиперпараметры обучения для получения качества  $AUC > 0.65$
3. Сформировать отчет и разместить его в своем репозитории на github.

**Выполнение работы:**

# Лабораторная работа №6:

## "Разработка системы предсказания поведения на основании графовых моделей"

**Цель:** обучение работе с графовым типом данных и графовыми нейронными сетями.

**Задача:** подготовить графовый датасет из базы данных о покупках и построить модель предсказания совершения покупки.

## Графовые нейронные сети

**Графовые нейронные сети** - тип нейронной сети, которая напрямую работает со структурой графа. Типичными применениями GNN являются:

- Классификация узлов;
- Предсказание связей;
- Графовая классификация;
- Распознавание движений;
- Рекомендательные системы.

В данной лабораторной работе будет происходить работа над **графовыми сверточными сетями**. Отличаются они от сверточных нейронных сетей нефиксированной структурой, функция свертки не является .

Подробнее можно прочитать тут: <https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bdfb7aba7b>

Тут можно почитать современные подходы к использованию графовых сверточных сетей <https://paperswithcode.com/method/gcn>

## Датасет

В качестве базы данных предлагаем использовать датасет о покупках пользователей в одном магазине товаров RecSys Challenge 2015 (<https://www.kaggle.com/datasets/chadgostopp/recsys-challenge-2015>).

Скачать датасет можно отсюда: <https://drive.google.com/drive/folders/1gtAeXPTj-c0RwVOKreMrZ3bfSmCwl2y-?usp=sharing> (lite-версия является облегченной версией исходного датасета, рекомендуем использовать её)

Также рекомендуем загружать данные в виде архива и распаковывать через пакет zipfile или/и скачивать датасет в собственный Google Drive и примонтировать его в колаб.

## Установка библиотек, выгрузка исходных датасетов

```
In [5]: # Slow method of installing pytorch geometric
# !pip install torch_geometric
# !pip install torch_sparse
# !pip install torch_scatter

# Install pytorch geometric
!pip install torch-sparse -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-cluster -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-spline-conv -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-geometric -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-scatter==2.0.9 -f https://data.pyg.org/whl/torch-1.11.0%2Bcu113.html

Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-sparse in /usr/local/lib/python3.7/dist-packages (0.6.13)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from torch-sparse) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scipy->torch-sparse) (1.21.6)
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-cluster in /usr/local/lib/python3.7/dist-packages (1.6.0)
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-spline-conv in /usr/local/lib/python3.7/dist-packages (1.2.1)
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-geometric in /usr/local/lib/python3.7/dist-packages (2.0.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (1.21.6)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (1.4.1)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (3.0.9)
```

Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (2.11.3)  
 Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (4.64.0)  
 Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (2.23.0)  
 Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (1.3.5)  
 Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (1.0.2)  
 Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from jinja2->torch-geometric) (2.0.1)  
 Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->torch-geometric) (2022.1)  
 Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->torch-geometric) (2.8.2)  
 Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas->torch-geometric) (1.15.0)  
 Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->torch-geometric) (2021.10.8)  
 Requirement already satisfied: urllib3!=1.25.0,!<1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->torch-geometric) (1.24.3)  
 Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->torch-geometric) (3.0.4)  
 Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->torch-geometric) (2.10)  
 Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->torch-geometric) (1.1.0)  
 Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->torch-geometric) (3.1.0)  
 Looking in links: https://data.pyg.org/whl/torch-1.11.0%2Bcu113.html  
 Requirement already satisfied: torch-scatter==2.0.9 in /usr/local/lib/python3.7/dist-packages (2.0.9)

```
In [6]: import numpy as np
import pandas as pd
import pickle
import csv
import os

from sklearn.preprocessing import LabelEncoder

import torch

# PyG - PyTorch Geometric
from torch_geometric.data import Data, DataLoader, InMemoryDataset

from tqdm import tqdm

RANDOM_SEED = 17 #@param { type: "integer" }
BASE_DIR = '/content/' #@param { type: "string" }
np.random.seed(RANDOM_SEED)
```

```
In [7]: # Check if CUDA is available for colab
torch.cuda.is_available
```

```
Out[7]: <function torch.cuda.is_available>
```

```
In [8]: # Unpack files from zip-file
import zipfile
with zipfile.ZipFile(BASE_DIR + 'yoochoose-data-lite.zip', 'r') as zip_ref:
    zip_ref.extractall(BASE_DIR)
```

## Анализ исходных данных

```
In [9]: # Read dataset of items in store
df = pd.read_csv(BASE_DIR + 'yoochoose-clicks-lite.dat')
# df.columns = ['session_id', 'timestamp', 'item_id', 'category']
df.head()
```

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: DtypeWarning: Columns (3) have mixed types.Specify dtype option on import or set low\_memory=False.  
 exec(code\_obj, self.user\_global\_ns, self.user\_ns)

```
Out[9]:
```

	session_id	timestamp	item_id	category
0	9	2014-04-06T11:26:24.127Z	214576500	0
1	9	2014-04-06T11:28:54.654Z	214576500	0
2	9	2014-04-06T11:29:13.479Z	214576500	0

3	19	2014-04-01T20:52:12.357Z	214561790	0
4	19	2014-04-01T20:52:13.758Z	214561790	0

```
In [10]: # Read dataset of purchases
buy_df = pd.read_csv(BASE_DIR + 'yoochoose-buys-lite.dat')
# buy df.columns = ['session_id', 'timestamp', 'item_id', 'price', 'quantity']
buy_df.head()
```

```
Out[10]:
```

	session_id	timestamp	item_id	price	quantity
0	420374	2014-04-06T18:44:58.314Z	214537888	12462	1
1	420374	2014-04-06T18:44:58.325Z	214537850	10471	1
2	489758	2014-04-06T09:59:52.422Z	214826955	1360	2
3	489758	2014-04-06T09:59:52.476Z	214826715	732	2
4	489758	2014-04-06T09:59:52.578Z	214827026	1046	1

```
In [11]: # Filter out item session with length < 2
df['valid_session'] = df.session_id.map(df.groupby('session_id')['item_id'].size() > 2)
df = df.loc[df.valid_session].drop('valid_session',axis=1)
df.nunique()
```

```
Out[11]: session_id    1000000
timestamp    5557758
item_id       37644
category       275
dtype: int64
```

```
In [12]: # Randomly sample a couple of them
NUM_SESSIONS = 60000 #@param { type: "integer" }
sampled_session_id = np.random.choice(df.session_id.unique(), NUM_SESSIONS, replace=False)
df = df.loc[df.session_id.isin(sampled_session_id)]
df.nunique()
```

```
Out[12]: session_id    60000
timestamp    334990
item_id       20043
category       103
dtype: int64
```

```
In [13]: # Average length of session
df.groupby('session_id')['item_id'].size().mean()
```

```
Out[13]: 5.5834166666666665
```

```
In [14]: # Encode item and category id in item dataset so that ids will be in range (0,len(df.item.unique()))
item_encoder = LabelEncoder()
category_encoder = LabelEncoder()
df['item_id'] = item_encoder.fit_transform(df.item_id)
df['category'] = category_encoder.fit_transform(df.category.apply(str))
df.head()
```

```
Out[14]:
```

	session_id	timestamp	item_id	category
91	131	2014-04-03T04:46:08.891Z	13649	0
92	131	2014-04-03T04:46:53.499Z	13445	0
93	131	2014-04-03T04:47:32.085Z	13585	0
177	309	2014-04-06T07:59:23.727Z	14064	0
178	309	2014-04-06T08:02:02.034Z	15547	0

```
In [15]: # Encode item and category id in purchase dataset
buy_df = buy_df.loc[buy_df.session_id.isin(df.session_id)]
buy_df['item_id'] = item_encoder.transform(buy_df.item_id)
```

```
buy_df.head()
```

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:3: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
This is separate from the ipykernel package so we can avoid doing imports until

```
Out[15]:
```

	session_id	timestamp	item_id	price	quantity
5	70427	2014-04-02T15:54:07.144Z	13729	3769	1
25	140964	2014-04-04T07:02:02.655Z	10268	2408	1
62	489671	2014-04-03T15:48:37.392Z	13710	4188	1
63	489671	2014-04-03T15:59:35.495Z	13710	4188	1
64	489671	2014-04-03T16:00:06.917Z	13710	4188	1

```
In [16]: # Get item dictionary with grouping by session
buy_item_dict = dict(buy_df.groupby('session_id')['item_id'].apply(list))
buy_item_dict
```

```
Out[16]: {714: [16129, 16324, 16326, 3323],
3517: [11939, 13381],
4832: [12191, 12191, 12191],
5002: [12217],
5942: [16913, 14322, 14040, 14040, 14040, 14040, 14322, 16913],
7054: [14021],
7173: [13549],
8628: [280, 281],
9292: [13783, 4280],
9702: [15763, 15756, 14250, 3096, 9004, 13708, 11207, 14092],
10879: [2311, 2362],
12017: [5219],
12282: [13864,
13866,
13862,
9463,
13866,
13864,
13862,
9463,
13866,
13864,
13862,
9463],
13073: [12213, 12213],
14227: [11645, 11646],
14314: [12227, 13658, 13585, 13773],
16053: [4365, 13874, 13872],
16946: [8864],
19576: [13862, 13862],
21812: [1049],
23934: [14142],
24703: [14079, 14079],
28668: [14065],
35699: [1758],
38117: [8306,
12079,
13907,
13907,
13618,
13775,
13775,
13625,
13625,
12959,
12960,
14764],
39247: [13862, 13866],
40176: [4277],
40209: [15763, 15762, 15756, 10293],
40717: [15058],
40827: [16129, 16129],
42536: [14040, 14322],
44097: [13520],
44714: [1468, 13523, 14321, 13523],
45836: [10705],
46132: [14021, 2584],
46288: [1361],
50567: [13864, 14065],
```

51751: [13865, 14040, 1601, 10089],  
54207: [14024, 14024, 14025, 14027],  
56987: [5175, 4484],  
58252: [12227, 12212],  
58836: [2278, 2272],  
59756: [7433],  
60358: [14145, 14133, 14982, 8161, 12167],  
63126: [14040, 13870],  
64182: [12227],  
64714: [12714, 13667],  
66977: [12996],  
69207: [14064, 14039, 14031, 952, 14022],  
70427: [13729],  
72638: [13862],  
72834: [247, 12765],  
74101: [5276, 15871],  
74252: [16355, 9429],  
75101: [14766, 8418, 8418, 14766],  
75528: [9180, 9501, 13888, 9056, 13888, 9180, 9056, 9501],  
76012: [17012],  
82692: [356, 356],  
82791: [6001],  
82911: [14079, 14092, 14030],  
83071: [10473],  
83258: [16461],  
84198: [2484],  
85449: [3584],  
86657: [14040, 7783],  
87451: [8373, 1688],  
89379: [12996, 12965, 12953],  
89924: [14143, 2365],  
92574: [8079],  
92676: [16129],  
93444: [14536, 12195],  
93471: [11389, 11390, 14021],  
94408: [14071, 14022],  
94548: [12193],  
101456: [8864],  
102253: [14119,  
14031,  
14119,  
13673,  
14045,  
14045,  
13673,  
14119,  
14119,  
14031,  
13673,  
14045,  
14119,  
14119,  
14031],  
102539: [16129, 13585],  
103406: [15883],  
104669: [3817],  
105938: [12192],  
107928: [14068],  
110581: [9426, 16482, 9439],  
110994: [8940, 5496],  
115778: [14025, 14027, 14080, 14087, 14026, 13887],  
116371: [14766, 13850, 1833],  
118554: [13873, 13874],  
118898: [14068],  
119611: [11179],  
121434: [13585, 12832],  
122484: [4078, 12250, 14086, 12520],  
123711: [4150],  
123789: [6742],  
124553: [4260, 5707, 12125, 4260, 12125, 5707],  
125016: [11368, 11385],  
125163: [3844, 6249],  
125408: [13771, 2488, 14041],  
126612: [11032],  
127304: [12467, 2484],  
127836: [9401, 16474],  
128226: [12217, 12217, 6760, 12217, 12217, 12217, 12217, 6760],  
129148: [2002, 2002],  
129686: [483],  
130063: [13731, 5511, 2846, 14064, 14039, 13861, 13719, 15313],  
130951: [14068],  
131342: [15058],  
134431: [3496],  
134681: [14079],  
135543: [13862, 13865, 13866],  
136192: [14135, 14107, 14133],  
138576: [4560],  
140964: [10268],  
144424: [7873],

145959: [14107, 14982, 14133, 14133],  
150667: [12195],  
151019: [13778, 12823],  
152571: [4281],  
152948: [868],  
157764: [13516],  
160621: [16128],  
162559: [7051, 5397],  
163316: [13549, 13549],  
163502: [16129],  
163997: [13862],  
164334: [54],  
164566: [13795],  
164849: [14045, 14079],  
166859: [13864, 14065],  
169128: [13555, 13897, 9056, 13897, 9056, 13555],  
171421: [10386, 11181],  
171899: [3190],  
175897: [12214],  
177488: [12952, 12996],  
179087: [5175, 5175, 14065, 13864, 14065, 13864],  
182813: [14065, 12124, 14045, 5192],  
182997: [17127],  
187011: [12994, 12996],  
187684: [12237, 14321, 13852, 14766],  
187714: [13854, 13848, 13610, 13854, 13848, 13610],  
189126: [11183, 13721, 10855],  
192272: [3994, 8474],  
192824: [13858, 13784],  
195093: [154],  
199746: [4024],  
200463: [13862],  
201316: [13570],  
201449: [16128, 4023, 14024, 13615, 14028],  
201584: [2216, 719, 13888, 2210, 13874],  
201967: [16930, 14312, 5192, 16915, 9481, 16510, 16926, 16728, 16527, 9506],  
204747: [6031],  
209453: [14093, 12326],  
212159: [5320, 11386, 11520],  
212724: [13731, 13719],  
213507: [232, 4127, 217, 1224],  
216349: [8985, 11368, 11392],  
221231: [14041, 14022, 14041, 14022],  
221828: [14107, 14135, 14136, 14133, 14982],  
223586: [16128, 13585],  
224413: [5192],  
224762: [12228],  
225078: [11181, 9004, 14030],  
225209: [1848, 1850, 14040],  
227211: [14321, 13610, 13699],  
228174: [15831, 15831],  
235922: [5192, 5193],  
239003: [13674, 14098],  
249438: [16420],  
251197: [12952],  
251419: [12952, 12952],  
251738: [14136, 14107, 14133, 14982],  
255903: [12196, 12196],  
256794: [12468, 12320, 13766, 13906],  
257821: [13731],  
257991: [320, 14093],  
260177: [13865, 13783, 13862],  
260656: [17],  
265052: [8986],  
267811: [4842, 14079, 14021, 14021, 14079, 4842],  
270061: [1945],  
270271: [12789],  
273056: [14071, 14022],  
274366: [4741],  
275441: [11164, 11165],  
276829: [12218, 13559],  
278609: [15999, 9490, 14039],  
278614: [14065],  
281122: [12218, 4842, 4842, 12218],  
283323: [6695, 14692, 14693, 14716, 8, 14717, 14714],  
284841: [12245],  
285851: [14045],  
286078: [13862, 13862],  
286656: [13862],  
286821: [5676],  
292252: [10027],  
293619: [14767],  
297264: [14041, 14063, 14040],  
299776: [7661, 3299, 982],  
300343: [8985],  
303848: [5195],  
303904: [8864],  
304811: [14041, 14022, 16420],  
307402: [12218, 12200, 12201],



308516: [16375, 12965, 12995],  
309659: [4499],  
309692: [12190],  
310214: [4231],  
312523: [14974],  
314311: [10542, 1833, 12227, 12227],  
314784: [12193, 12193, 281],  
315631: [15268, 4508],  
316214: [14394, 11181],  
321253: [16551],  
321651: [4231],  
321933: [14021, 14031, 13703],  
324233: [14031, 13864],  
325144: [3932],  
328188: [14119, 14079, 14021],  
328904: [13844, 13865],  
331212: [12948, 12995],  
332869: [16355, 11951, 12227, 16354],  
334417: [14394, 14144, 8988, 13862],  
335958: [14119, 14065],  
336242: [16352],  
336492: [12228, 12191, 10960],  
337048: [13718],  
337909: [13862],  
341318: [11217, 11217],  
341457: [5183, 5183],  
343079: [2155, 2153],  
344132: [8864],  
344402: [2128, 54],  
344839: [14119, 14079, 14029],  
346517: [14327],  
346742: [12949, 12996, 12949, 12996],  
349657: [14119],  
357839: [14077, 14093, 14094],  
361071: [13863, 14031],  
361231: [13864, 14031, 14031, 13864, 14031, 13864],  
364756: [12473, 12453],  
365203: [5424, 16591, 16906, 16905, 17020],  
366704: [14077, 11383, 14094],  
367684: [13793, 14096, 16325],  
371187: [14162, 151, 12567],  
373084: [14766, 14767, 13831],  
377464: [14321, 12468, 4491],  
378782: [13870, 14100],  
381579: [14119, 14031, 13863, 14079, 14045],  
383181: [13779, 13549],  
386796: [14119, 14031, 14045],  
389843: [2269, 8972],  
392542: [10033],  
396189: [13897, 13555],  
397833: [14031, 14119],  
402278: [14077, 14082],  
403286: [11181, 6749],  
404276: [8054],  
404849: [281, 12995],  
405347: [1789],  
408613: [13593],  
409608: [5784, 5876],  
409989: [13549, 13779],  
410391: [6023],  
413417: [14982, 14136],  
421304: [9101, 1903],  
421377: [12228, 12228],  
422816: [4486, 15154, 4566, 16258, 16258, 15154, 4486, 4566],  
423532: [13729],  
425544: [11907, 12198, 14796, 12852, 4402],  
425851: [14119, 14079, 11406, 11392, 11368, 11366],  
431642: [13731, 13719, 13719],  
433573: [12950, 12951, 12949],  
433688: [12227, 12227],  
434007: [13775, 13775, 13831],  
434387: [14109, 14092],  
435192: [1257],  
435912: [12951, 12996],  
441428: [15182],  
444494: [13555, 14021, 15043],  
445679: [14068],  
445898: [14136, 14087, 14982, 14132, 14250, 14109, 16026, 15763, 14133],  
446758: [14093],  
447123: [12734, 1559],  
450707: [12964],  
450926: [12949],  
451081: [8939, 3428, 13800, 12424, 12468],  
455769: [12327, 11253, 11383, 13854, 7933],  
457513: [12193, 12193],  
459111: [14071, 13865, 14022, 13866, 13729],  
461578: [14981],  
462557: [14766],  
463878: [181, 16368],

466456: [9440],  
467932: [15394],  
468484: [14064, 13862],  
470727: [460],  
471707: [13862],  
471824: [2200, 1257, 2743],  
473777: [12191],  
474203: [644],  
478114: [12457, 14767],  
478278: [14767, 12457, 13848],  
478921: [16573, 16573],  
486031: [16460],  
489671: [13710, 13710, 13710],  
494374: [5314],  
494853: [13862, 13866, 13863],  
495012: [14135, 14136],  
495133: [13668, 13650],  
496003: [12321],  
499437: [6806, 1332],  
501207: [16254, 411],  
505106: [13862, 13862],  
506317: [14041, 14068],  
508572: [11508],  
509182: [11540],  
519472: [14087, 14087, 14085],  
522888: [13871, 13887],  
522939: [16048],  
524829: [942],  
525857: [14327],  
527728: [14085, 14025],  
529402: [12800],  
529696: [12952, 4756, 9888],  
531273: [40, 40, 40],  
532336: [5260],  
532936: [12218, 12212, 12212],  
535934: [13862],  
537506: [10705],  
540172: [942, 14974, 13864],  
543594: [6678, 3719],  
543766: [11519, 12471],  
543943: [2465],  
546959: [6391],  
547283: [4127, 4127],  
547334: [12949, 12996, 12951],  
547464: [3874],  
548517: [13874, 13887],  
551414: [13876],  
554879: [13708, 11183, 11181, 13704, 9056],  
554976: [16661, 16532],  
557162: [11939, 3988],  
557367: [13871],  
558221: [13870, 14132],  
560828: [13876],  
562401: [14025, 14028],  
569293: [16129, 16326, 16325, 899, 888],  
571414: [14100, 14085, 14080, 12292],  
573456: [13862],  
574017: [12158, 12157],  
575682: [14138],  
576964: [8599],  
578953: [5437],  
579696: [5731],  
580002: [14041],  
580484: [13838, 633, 197],  
583516: [14065],  
583804: [6710],  
588491: [16129],  
588579: [14767],  
592249: [1062, 642],  
594661: [13205, 11418],  
596594: [3217, 11037, 2177],  
596902: [5700, 12473, 14137, 10626],  
597919: [14094],  
600116: [10930],  
607852: [2880],  
609391: [11222],  
610056: [13585, 2154],  
615278: [11541],  
618418: [14137, 11386],  
619386: [1015],  
627312: [10680],  
635691: [14028, 14026, 14025],  
636074: [13205],  
636153: [16129],  
639221: [12953, 16129, 16325, 16326],  
639429: [126],  
640408: [13719, 13731, 13719, 13731],  
640533: [16129, 16325],  
641032: [15044, 13874],

641808: [4024],  
642357: [942],  
648612: [14058],  
648793: [971],  
650204: [5700, 14137],  
652067: [13866, 13865, 13866],  
653517: [12952],  
654817: [14067],  
658774: [179],  
659171: [14273],  
661486: [16129],  
662513: [13872, 13911],  
664249: [13876, 12523],  
665391: [15763, 15756],  
665661: [16129, 16325, 16326],  
666289: [8599],  
667286: [13725, 13725],  
668991: [5864, 10402],  
671061: [2800, 2272],  
671599: [13704, 13708],  
672104: [7874],  
674939: [14326],  
675221: [13774, 12505, 12503, 13776],  
676373: [5755],  
676692: [10726],  
684492: [413, 11958],  
684543: [13871],  
685528: [8940, 8940],  
686913: [13677],  
687613: [14027, 15764],  
687696: [14136],  
14250,  
14102,  
14145,  
14107,  
3848,  
11927,  
14135,  
14133,  
14982,  
14982,  
14136,  
14250,  
14102,  
14145,  
14133,  
14107,  
14982,  
14982,  
3848,  
11927,  
14135],  
689288: [13872],  
691938: [16375, 3135, 6549],  
692364: [11886],  
693221: [14329],  
694637: [11015, 12728, 12458],  
694958: [14102, 5175],  
696186: [12950],  
696691: [14820],  
700394: [2107, 14326, 623, 12945],  
704016: [12475],  
705319: [13871],  
709196: [12192, 12227],  
710323: [17087],  
711198: [14137, 14138, 14139],  
712441: [5175, 5175],  
718772: [13871],  
721918: [13872, 17154, 14253, 13873],  
721932: [13663, 1184, 14041, 16078],  
725763: [14082],  
726546: [14028, 12185, 14080],  
729638: [6393],  
730913: [16461],  
731117: [12949],  
732071: [14132, 14025, 14102, 14025, 14027, 14028, 12823, 14110, 14100],  
732131: [14086],  
733484: [13359],  
734488: [2513],  
735486: [589],  
737403: [14145, 14107, 14135, 14133],  
739408: [14053],  
740263: [11520, 11519],  
742962: [13836],  
744207: [13872, 13873],  
744519: [5572],  
745038: [5700],  
746527: [12517],  
747581: [6025, 2137],

748523: [14137, 12995, 11390, 14093, 14077],  
749562: [14139, 14138, 14041],  
750127: [5193, 5193],  
751702: [11939, 11507, 9911, 12219, 3984, 12219],  
753556: [12465],  
754163: [6678],  
754531: [3844],  
756294: [14142],  
757819: [8090],  
764059: [12194],  
766444: [14028, 14025, 14026, 14027],  
767532: [261, 292],  
767639: [14326, 16026],  
770473: [12950, 12995, 12994],  
771316: [13868, 13593],  
775433: [16129],  
775712: [11390, 14082],  
778843: [10746, 10747],  
779677: [14145, 9002, 14250, 12823, 12823, 13549, 13667],  
779987: [14100, 13871, 14100, 13871],  
780342: [14327, 7581, 98],  
780781: [13515, 7139],  
783148: [13874, 13871],  
783158: [13872, 13871],  
786279: [13864, 2356],  
787382: [437],  
789111: [9056],  
789687: [14108, 14109, 14110],  
790019: [5700],  
790176: [13866, 14031],  
794276: [16461],  
794999: [14326, 13870],  
795574: [6102],  
798224: [14835, 15262],  
800001: [12158],  
800719: [12950, 11888],  
802002: [14102, 14250, 9960],  
802672: [14138],  
805538: [14085, 5680, 14087],  
806698: [40],  
808287: [14974, 4250],  
810357: [151],  
812219: [13871, 13876],  
813497: [2568],  
816386: [5700, 5700],  
818381: [14764, 13909],  
818841: [13877, 13872, 13877, 13872],  
819671: [12191, 11605, 12227, 12227],  
820353: [10919],  
820468: [14108, 12823],  
821093: [12194],  
821987: [10146],  
827732: [13776, 13549, 4364],  
830713: [4019],  
832337: [13555],  
832993: [14080, 13478, 13911, 14100],  
834797: [12194, 12193],  
835178: [5382, 13870, 14025, 14064, 14027, 11547],  
837023: [1491, 11917, 12334, 11390],  
838723: [13729],  
839867: [4812],  
841407: [13520],  
842286: [10473, 13907, 13618],  
844364: [13554, 13656],  
845384: [14142],  
846032: [13862, 13865, 2576, 14143],  
846156: [13877, 13872],  
846761: [14118],  
851353: [16461, 5192],  
851384: [1976],  
851993: [15923],  
852964: [17409],  
853229: [14250, 14102],  
854146: [13672],  
856862: [14137],  
857101: [659, 13585, 10182, 12963, 600, 7131],  
860113: [14107, 14133, 14135, 14982],  
865328: [12527, 12632, 1250, 1228, 1252],  
865821: [1040],  
866566: [14041],  
866862: [14137, 14139],  
870307: [15744],  
871781: [16247],  
871917: [13708],  
872589: [4247],  
873302: [10780, 9067, 434, 16223, 7895],  
876467: [15756, 14027, 13776, 14026, 15764, 15765],  
879887: [8598, 11917],  
880909: [8599],

885063: [14093, 14077],  
885776: [14974, 942, 4250, 4246],  
887191: [12212],  
892571: [7678, 17],  
899332: [12935, 14030, 13772, 12325, 8071, 12315],  
900139: [13718],  
902066: [17464],  
904382: [431, 441, 11000, 597, 10352],  
905146: [2826],  
908893: [14250, 14102, 9002, 13873],  
911682: [14175, 14329, 3032],  
912647: [13872, 13874],  
914297: [14028, 14026],  
914931: [2107, 2266],  
915972: [14109, 14110, 14026, 14027],  
916651: [9101, 9056, 14250, 2977, 12091, 11383],  
919806: [16129, 16325],  
920208: [720, 13872],  
926509: [16129, 16326, 16325],  
928243: [14065, 12391, 14393, 9820, 5683, 3493],  
929629: [15209],  
929956: [14085, 14087],  
933173: [14145, 14135, 14982, 14102, 14250, 11927, 9002, 3467],  
933603: [4419],  
933669: [12194],  
934768: [13668],  
935256: [3846, 3846],  
936354: [12193],  
937514: [8979],  
937756: [12471, 2814, 12466],  
937962: [942, 4546, 4540],  
938106: [5664],  
939603: [9450, 9454],  
942777: [5529, 5524],  
944311: [4020, 14251, 4021, 14363],  
945021: [4546],  
949326: [13872, 13887],  
950594: [12346, 6713, 14251],  
953068: [14025],  
956526: [14399, 14253, 14398],  
957212: [13872, 13872],  
957621: [14273, 14268],  
958112: [167, 4002],  
961856: [13877],  
964362: [4250],  
964368: [14974],  
965519: [11540],  
966976: [5680, 1510],  
967432: [12342, 12728],  
968386: [13828, 11918],  
969256: [9, 10748, 13205, 13205, 10748, 9],  
970713: [4746],  
972051: [11513, 3989],  
972233: [4251, 14335],  
973291: [14988, 14245, 13915, 13711, 14106, 14130],  
976244: [13871, 1833, 13871, 1833],  
980617: [4250, 4250],  
982527: [11540, 5851, 14045],  
984226: [14329, 16186, 13829, 13828, 14336],  
984729: [13872],  
985866: [5292],  
986979: [16129, 16324],  
988443: [14399],  
997288: [4203],  
997548: [14023, 13916],  
997911: [14781, 12228],  
998446: [14242],  
1000683: [5266, 11939],  
1004773: [11286, 5615, 5645, 5603],  
1008569: [14138, 11321, 14137, 11321, 14138, 14137],  
1011276: [12227],  
1014437: [4010, 6809, 6809, 4010, 6809, 4010],  
1020822: [2213],  
1023394: [12227, 12227, 12191, 12192],  
1024072: [5103],  
1024232: [12195, 5693],  
1027499: [13871],  
1039884: [14331, 14332],  
1041829: [1510, 16323],  
1042824: [9028, 9056, 9056, 9028],  
1044241: [14274, 14335],  
1045572: [4345, 13873, 2210],  
1046209: [17168, 17188, 13887, 13887, 17188, 17168, 17188, 13887, 17168],  
1050998: [12801],  
1051854: [3241],  
1052313: [13856, 13412],  
1055351: [14271, 2883],  
1056763: [14331, 14331, 14331, 2272, 14331, 2272],  
1057419: [16461],

1058554: [4555, 4559],  
1059771: [14057, 11386, 14137, 14057, 14137, 11386],  
1064338: [1833, 1833],  
1065002: [2356, 9026, 14326],  
1066239: [11419, 13988],  
1068709: [15913],  
1074288: [4126],  
1076964: [2272, 2277],  
1077998: [12250, 14965, 14135],  
1080149: [16129, 15620, 4044],  
1081158: [14256],  
1081821: [13872, 13874, 13872, 13874],  
1083153: [14941],  
1087122: [4481, 14258],  
1087189: [14253],  
1089248: [14137],  
1089733: [4546, 16461],  
1089762: [14252, 14293],  
1093901: [3431],  
1094614: [12858, 3984, 11939],  
1097381: [5219],  
1099254: [1231, 1248, 1238],  
1100316: [13872, 13871],  
1101143: [1468],  
1102354: [14253, 13483],  
1102487: [15978, 16485, 16485, 15978],  
1103767: [14144, 182],  
1104398: [11385, 11366],  
1104461: [12346],  
1106904: [4512],  
1107004: [12416, 13848, 13633, 4044],  
1107747: [12948, 12995],  
1108122: [16711, 4481],  
1111629: [17180, 17180],  
1112633: [8940, 4365, 2212, 5676],  
1114146: [4246],  
1116163: [14835, 14257],  
1118104: [14253, 1241, 1248],  
1122811: [3496],  
1123957: [3588],  
1125834: [5219, 14326, 1833, 5219, 14326, 1833],  
1126498: [7260],  
1126713: [11014, 13863],  
1126833: [9948],  
1128034: [12466],  
1129937: [1901],  
1130157: [13872],  
1130429: [14253, 13572, 13548, 12331],  
1131734: [13828, 1257],  
1132059: [14256],  
1133141: [13708, 13708],  
1139747: [13874],  
1144541: [3123],  
1144616: [11025],  
1144864: [17135, 17019],  
1146953: [10389],  
1146991: [11903, 2479],  
1148314: [12471],  
1149267: [5180],  
1153691: [4126, 1186],  
1154671: [14244, 8915],  
1157302: [12400, 16375],  
1158068: [5192, 5193],  
1158407: [13878],  
1159062: [13607, 2793, 3211, 2834, 7783, 8938],  
1159404: [13871],  
1163769: [14251],  
1169468: [5192, 16710],  
1175021: [8940],  
1175436: [8598],  
1176523: [14137, 16131, 17115],  
1177008: [3385],  
1177998: [13871],  
1178791: [12129],  
1178847: [12824, 12948],  
1182557: [8263],  
1182673: [14329],  
1187083: [14775],  
1188177: [8016, 12157],  
1189894: [9948],  
1193796: [5700],  
1194322: [11371, 15773, 14990],  
1194989: [14833, 14251],  
1197376: [14975],  
1198247: [14334],  
1199078: [13721],  
1199561: [14858],  
1205866: [13874, 13888],  
1209648: [13872, 13874],

1214621: [14252, 14328, 14252, 14328],  
1217786: [9998, 15196, 12570],  
1218303: [14333],  
1219567: [11925],  
1221396: [14329],  
1224047: [14351],  
1224953: [13520, 13575, 13522],  
1225777: [14137],  
1226527: [13887, 13877],  
1227514: [13829, 13830],  
1234702: [6811, 3136],  
1234771: [10621],  
1240491: [10146],  
1247423: [10560, 14086, 13910, 12462],  
1248963: [13838],  
1251838: [6184, 12317],  
1253343: [14989, 14965, 14023],  
1255379: [12196, 13584, 13584, 12196],  
1255914: [14256, 14268, 14273, 14835],  
1257224: [14137, 11386, 6808],  
1258937: [13716, 14289],  
1259426: [13878, 14023, 14989, 14988, 14988],  
1261996: [13887, 17154],  
1262164: [14327],  
1263007: [460, 14312],  
1264417: [13908, 13908, 13828, 14118],  
1266998: [12734],  
1267443: [13876],  
1269672: [2216, 8870],  
1271254: [13381],  
1273864: [14253, 13828],  
1273981: [2695, 14339, 918],  
1274234: [4544, 1468],  
1274612: [2272, 7328],  
1277346: [1557],  
1278647: [13784, 14144, 14144, 13784, 14144, 14144, 14144, 13784, 14144],  
1279418: [16551, 6248],  
1280668: [12471, 11520],  
1281797: [8868],  
1282037: [14737, 14738],  
1286224: [14335, 17118],  
1289384: [13865],  
1291221: [12215, 12215],  
1291888: [17415, 14288, 17048],  
1292324: [17046, 17048],  
1298274: [5676, 17179],  
1299948: [14293, 14293, 15262, 15262],  
1300549: [13873, 13872, 13871],  
1300726: [13874, 13871, 13872],  
1300789: [12562],  
1304107: [12421, 14568, 14485],  
1304826: [5358],  
1305331: [67],  
1306361: [14339, 14988, 14345, 14023],  
1306908: [14066, 14070, 11425, 13867, 14066, 14070, 11425, 13867],  
1312403: [14988, 14989],  
1314418: [7288],  
1315059: [14386],  
1315237: [14339],  
1316929: [14251],  
1321763: [2210],  
1322374: [14274, 14335],  
1323157: [13520, 15333, 13567],  
1326153: [14268, 14273],  
1327659: [14988, 14023, 14102],  
1327939: [14023, 14988, 14345, 14020],  
1328983: [14041, 14022],  
1335459: [14256],  
1335638: [7775],  
1335704: [7685, 1941, 11223],  
1342307: [12823, 14989],  
1343713: [14989, 11520, 11519],  
1344283: [12852, 16466, 9438],  
1346352: [13916, 11383, 14020, 13915],  
1349402: [12298],  
1349922: [16573],  
1351226: [14989, 14988, 14015],  
1353556: [1145],  
1362118: [11341, 14867],  
1363093: [15013, 12142],  
1363889: [9492, 9501, 16509, 9490],  
1364022: [9387, 9492],  
1365446: [14042],  
1365487: [10704, 10704],  
1366209: [13851],  
1366828: [14986, 13878],  
1378983: [13866],  
1380237: [14439, 14441],  
1380511: [14023, 14989, 14989, 11519],

1382401: [13351],  
1382469: [14989, 14023],  
1384059: [11386],  
1384711: [13572, 5233],  
1385663: [14989, 14988, 14023, 14345],  
1385773: [6476],  
1391357: [14349, 9810, 4465, 14817, 14269, 14764, 14766, 14366],  
1393968: [14240, 1979, 16454, 13879],  
1395197: [14023],  
1398773: [14965, 14988],  
1399539: [14988, 14339, 14345],  
1401503: [14070, 10077],  
1402908: [14023, 14989, 13915, 13916],  
1403437: [13898, 2503, 2862, 2617],  
1403776: [13862],  
1404683: [11230],  
1415274: [14988, 14989, 14023],  
1415289: [11520, 12454],  
1415362: [3331],  
1415617: [12727, 14020, 10851],  
1416463: [1118],  
1419311: [12419],  
1420472: [13916, 14988, 13650],  
1424849: [13671, 14989, 14023, 13885],  
1425353: [14065, 14065],  
1429098: [14331, 14334, 14332],  
1430221: [15354],  
1431002: [14988, 14965, 14023],  
1432123: [13872, 13871],  
1435339: [14989, 14023],  
1436348: [14251],  
1437197: [10872],  
1439529: [13916, 13915],  
1440943: [14251],  
1442851: [14289, 13716],  
1443998: [14870, 14870],  
1449601: [16421],  
1450702: [14242],  
1451157: [11520, 13885],  
1454773: [14026, 14027],  
1457728: [352],  
1458234: [13909, 14038],  
1459251: [14339, 13916, 14015, 15978],  
1460323: [5676,  
12720,  
13679,  
14077,  
14119,  
7329,  
8984,  
11969,  
7161,  
13606,  
7160,  
11967,  
11966,  
167],  
1461142: [14256],  
1463069: [14262, 14331],  
1465012: [14737, 14064],  
1465318: [9501, 16509, 16526],  
1466243: [14987, 16356, 14237],  
1467011: [14339, 14020, 14023, 14988],  
1468393: [1056],  
1470474: [14766],  
1471044: [14066, 2565, 14989, 14023],  
1471534: [14938, 14131, 14937, 14989, 17168],  
1474082: [14339],  
1475723: [17039],  
1476162: [13916],  
1476496: [14988, 14023, 14965, 14345, 14339],  
1481656: [643],  
1486097: [6417, 1011],  
1486574: [13764],  
1487836: [13864, 14065, 11232],  
1489009: [12964, 12964],  
1489531: [14023, 14989],  
1490732: [14988, 14023, 14345, 14989, 14339],  
1491167: [13862, 13827, 5382, 13830],  
1491563: [14911],  
1494023: [16254, 10781],  
1494109: [4577, 3733],  
1495382: [14935, 14933, 14935, 14933],  
1497361: [13914],  
1499409: [14989, 14023],  
1500126: [13886, 13886],  
1501409: [12075, 11538],  
1503214: [14251, 14251],  
1504319: [11520, 11383, 13915, 14989],



1504453: [13867, 14363],  
1513343: [14273, 14268],  
1513671: [16926],  
1516247: [16343, 5846, 8598],  
1517898: [14987, 15035],  
1518142: [14041],  
1519176: [14023, 14989],  
1520152: [14087, 12324],  
1524154: [16898, 17120, 16660, 17020, 16675, 9387, 15980],  
1525113: [14341, 14043, 12867],  
1528797: [12727],  
1529394: [14989, 14345],  
1535879: [14989, 13898],  
1535931: [14098],  
1539847: [15030],  
1540248: [440],  
1540937: [13865, 13866],  
1546673: [14988, 14989, 14345, 14023, 14339, 12793, 14245],  
1548616: [3460, 899],  
1553861: [14399],  
1557519: [14106, 17190],  
1558297: [4312],  
1560669: [14439],  
1561776: [14989, 10851],  
1563752: [14401, 14108, 14110],  
1564707: [10900, 10900],  
1565731: [13403, 11633, 7909],  
1566699: [3673],  
1567263: [176],  
1569714: [14989, 14023],  
1572986: [14401],  
1575674: [4481, 13774],  
1580358: [6586, 15764],  
1581886: [4126],  
1582931: [14247, 13870, 13774, 14247, 13870, 13774],  
1584569: [17167],  
1586853: [14249, 4481, 13607, 14100, 4247],  
1588817: [12728],  
1589821: [14989],  
1594278: [14766, 14766, 13886, 13610],  
1596107: [1376],  
1598719: [13668, 13667],  
1600537: [2436],  
1601312: [5301],  
1605984: [14989, 14023, 14965, 14111, 14129],  
1614023: [14023, 13910, 14988, 4465, 14349, 14341, 14086],  
1614231: [4481, 14023, 14988, 14965, 14345, 16186],  
1616066: [1015],  
1616956: [14989, 14339, 14965],  
1617323: [13779, 4481, 13849],  
1620243: [14988, 14989, 14988, 14989],  
1620856: [5192, 5193],  
1621062: [14287],  
1621151: [13915, 14989, 14988, 14023],  
1622166: [12471],  
1623404: [16858, 16857],  
1623667: [13874],  
1624443: [12949],  
1625746: [14237, 13916, 14989],  
1626118: [4299],  
1630847: [14989, 14988, 9065],  
1631302: [14988, 14989, 14023, 14339],  
1633681: [5266],  
1634722: [4465, 4465],  
1636883: [2212, 15031],  
1643328: [9375, 9426, 9385, 16875, 9429],  
1648444: [369],  
1649417: [1833],  
1650347: [14397],  
1658949: [14042, 11263, 14066, 15033],  
1662624: [709],  
1670517: [13725],  
1671832: [15763, 14367, 3582, 14817],  
1675126: [10900],  
1675762: [11939, 14251],  
1677669: [14020, 14989, 13916, 14986],  
1677874: [14767, 14339, 13851],  
1679296: [14987, 13885],  
1679728: [16680, 16690],  
1683484: [11520, 14988, 14989],  
1686752: [12964],  
1687056: [13889],  
1690711: [10930, 10930],  
1692087: [13844, 14293],  
1694197: [14989, 14023],  
1695187: [14904, 14905, 14905, 14904, 14905, 14904, 14905, 14904],  
1696008: [5461, 5461, 5461],  
1699464: [11386, 6807, 10040],  
1700194: [16516, 12732],

```

1702749: [12271],
1708104: [2486],
1711249: [16710],
1714647: [4465],
1715477: [8557],
1716782: [11230],
1719213: [14988,
10877,
14108,
14020,
14989,
13915,
13831,
13552,
14833,
13552,
2834,
2834,
14833,
13831],
1721921: [465, 3874],
1723401: [3984, 3984],
1723441: [3992],
1728434: [14346],
1733991: [14256],
1737034: [14287],
1737112: [14339],
1738199: [3294],
1738611: [14099],
1745309: [14988, 14988, 14023],
1745534: [14289],
1745982: [14102, 14339],
1746101: [12727, 12727, 12727, 12727],
1746348: [369, 369],
1748501: [11546],
1749123: [14339, 14989, 13879, 14988],
1749378: [5384, 5384, 5384, 5384],
1750134: [15999, 3182, 13908],
1753811: [4825],
1756284: [14986, 13879, 14401],
1757251: [14023, 14345, 14020, 14989],
1761344: [13916],
1769273: [14289],
1770074: [54],
1771186: [13519, 2272],
...}

```

## Сборка выборки для обучения

```

In [17]: # Transform df into tensor data
def transform_dataset(df, buy_item_dict):
    data_list = []

    # Group by session
    grouped = df.groupby('session_id')
    for session_id, group in tqdm(grouped):
        le = LabelEncoder()
        sess_item_id = le.fit_transform(group.item_id)
        group = group.reset_index(drop=True)
        group['sess_item_id'] = sess_item_id

        #get input features
        node_features = group.loc[group.session_id==session_id,
                                ['sess_item_id', 'item_id', 'category']].sort_values('sess_item_id')[['item_id',
                                'category']]
        node_features = torch.LongTensor(node_features).unsqueeze(1)
        target_nodes = group.sess_item_id.values[1:]
        source_nodes = group.sess_item_id.values[:-1]

        edge_index = torch.tensor([source_nodes,
                                    target_nodes], dtype=torch.long)
        x = node_features

        #get result
        if session_id in buy_item_dict:
            positive_indices = le.transform(buy_item_dict[session_id])
            label = np.zeros(len(node_features))
            label[positive_indices] = 1
        else:
            label = [0] * len(node_features)

        y = torch.FloatTensor(label)

        data = Data(x=x, edge_index=edge_index, y=y)

```

```

        data_list.append(data)

    return data_list

# Pytorch class for creating datasets
class YooChooseDataset(InMemoryDataset):
    def __init__(self, root, transform=None, pre_transform=None):
        super(YooChooseDataset, self).__init__(root, transform, pre_transform)
        self.data, self.slices = torch.load(self.processed_paths[0])

    @property
    def raw_file_names(self):
        return []

    @property
    def processed_file_names(self):
        return [BASE_DIR+'yoochoose_click_binary_100000_sess.dataset']

    def download(self):
        pass

    def process(self):
        data_list = transform_dataset(df, buy_item_dict)

        data, slices = self.collate(data_list)
        torch.save((data, slices), self.processed_paths[0])

```

```

In [18]: # Prepare dataset
dataset = YooChooseDataset('./')

```

Processing...

```

0%|          | 0/60000 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:21: UserWarning: Creating a tensor from a list of numpy.ndarrays is extremely slow. Please consider converting the list to a single numpy.ndarray with numpy.array() before converting to a tensor. (Triggered internally at ../torch/csrc/uts/ils/tensor_new.cpp:210.)
100%|██████████| 60000/60000 [05:05<00:00, 196.46it/s]
Done!

```

## Разделение выборки

```

In [19]: # train_test_split
dataset = dataset.shuffle()
one_tenth_length = int(len(dataset) * 0.1)
train_dataset = dataset[:one_tenth_length * 8]
val_dataset = dataset[one_tenth_length*8:one_tenth_length * 9]
test_dataset = dataset[one_tenth_length*9:]
len(train_dataset), len(val_dataset), len(test_dataset)

```

```

Out[19]: (48000, 6000, 6000)

```

```

In [20]: # Load dataset into PyG loaders
batch_size= 512
train_loader = DataLoader(train_dataset, batch_size=batch_size)
val_loader = DataLoader(val_dataset, batch_size=batch_size)
test_loader = DataLoader(test_dataset, batch_size=batch_size)

```

```

/usr/local/lib/python3.7/dist-packages/torch_geometric/deprecation.py:12: UserWarning: 'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
warnings.warn(out)

```

```

In [21]: # Load dataset into PyG loaders
num_items = df.item_id.max() +1
num_categories = df.category.max()+1
num_items , num_categories

```

```

Out[21]: (20043, 102)

```

## Настройка модели для обучения

```

In [22]: embed_dim = 128
from torch_geometric.nn import GraphConv, TopKPooling, GatedGraphConv, SAGEConv, SGConv

```

```

from torch_geometric.nn import global_mean_pool as gap, global_max_pool as gmp
import torch.nn.functional as F

class Net(torch.nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # Model Structure
        self.conv1 = GraphConv(embed_dim * 2, 128)
        self.pool1 = TopKPooling(128, ratio=0.9)
        self.conv2 = GraphConv(128, 128)
        self.pool2 = TopKPooling(128, ratio=0.9)
        self.conv3 = GraphConv(128, 128)
        self.pool3 = TopKPooling(128, ratio=0.9)
        self.item_embedding = torch.nn.Embedding(num_embeddings=num_items, embedding_dim=embed_dim)
        self.category_embedding = torch.nn.Embedding(num_embeddings=num_categories, embedding_dim=embed_dim)
        self.lin1 = torch.nn.Linear(256, 256)
        self.bn1 = torch.nn.BatchNorm1d(128)
        self.bn2 = torch.nn.BatchNorm1d(64)
        self.act1 = torch.nn.ReLU()
        self.act2 = torch.nn.ReLU()

    # Forward step of a model
    def forward(self, data):
        x, edge_index, batch = data.x, data.edge_index, data.batch

        item_id = x[:, :, 0]
        category = x[:, :, 1]

        emb_item = self.item_embedding(item_id).squeeze(1)
        emb_category = self.category_embedding(category).squeeze(1)

        x = torch.cat([emb_item, emb_category], dim=1)
        # print(x.shape)
        x = F.relu(self.conv1(x, edge_index))
        # print(x.shape)
        r = self.pool1(x, edge_index, None, batch)
        # print(r)
        x, edge_index, _, batch, _, _ = self.pool1(x, edge_index, None, batch)
        x1 = torch.cat([gmp(x, batch), gap(x, batch)], dim=1)

        x = F.relu(self.conv2(x, edge_index))

        x, edge_index, _, batch, _, _ = self.pool2(x, edge_index, None, batch)
        x2 = torch.cat([gmp(x, batch), gap(x, batch)], dim=1)

        x = F.relu(self.conv3(x, edge_index))

        x, edge_index, _, batch, _, _ = self.pool3(x, edge_index, None, batch)
        x3 = torch.cat([gmp(x, batch), gap(x, batch)], dim=1)

        x = x1 + x2 + x3

        x = self.lin1(x)
        x = self.act1(x)
        x = self.lin2(x)
        x = F.dropout(x, p=0.5, training=self.training)
        x = self.act2(x)

        outputs = []
        for i in range(x.size(0)):
            output = torch.matmul(emb_item[data.batch == i], x[i, :])

            outputs.append(output)

        x = torch.cat(outputs, dim=0)
        x = torch.sigmoid(x)

        return x

```

## Обучение нейронной сверточной сети

```

In [23]: # Enable CUDA computing
device = torch.device('cuda')
model = Net().to(device)
# Choose optimizer and criterion for learning
optimizer = torch.optim.Adam(model.parameters(), lr=0.002)
crit = torch.nn.BCELoss()

```

```

In [24]: # Train function
def train():
    model.train()

```

```

loss_all = 0
for data in train_loader:
    data = data.to(device)
    optimizer.zero_grad()
    output = model(data)

    label = data.y.to(device)
    loss = crit(output, label)
    loss.backward()
    loss_all += data.num_graphs * loss.item()
    optimizer.step()
return loss_all / len(train_dataset)

```

In [25]:

```

# Evaluate result of a model
from sklearn.metrics import roc_auc_score
def evaluate(loader):
    model.eval()

    predictions = []
    labels = []

    with torch.no_grad():
        for data in loader:

            data = data.to(device)
            pred = model(data).detach().cpu().numpy()

            label = data.y.detach().cpu().numpy()
            predictions.append(pred)
            labels.append(label)

    predictions = np.hstack(predictions)
    labels = np.hstack(labels)

    return roc_auc_score(labels, predictions)

```

In [26]:

```

# Train a model
NUM_EPOCHS = 10 #@param { type: "integer" }
for epoch in tqdm(range(NUM_EPOCHS)):
    loss = train()
    train_acc = evaluate(train_loader)
    val_acc = evaluate(val_loader)
    test_acc = evaluate(test_loader)
    print('Epoch: {:03d}, Loss: {:.5f}, Train Auc: {:.5f}, Val Auc: {:.5f}, Test Auc: {:.5f}'.
          format(epoch, loss, train_acc, val_acc, test_acc))

```

10%|██████| 1/10 [01:23<12:35, 83.94s/it]

Epoch: 000, Loss: 0.64618, Train Auc: 0.54206, Val Auc: 0.54058, Test Auc: 0.53312

20%|██████| 2/10 [02:41<10:39, 79.99s/it]

Epoch: 001, Loss: 0.45930, Train Auc: 0.58907, Val Auc: 0.55887, Test Auc: 0.55532

30%|██████| 3/10 [03:57<09:08, 78.35s/it]

Epoch: 002, Loss: 0.40134, Train Auc: 0.63227, Val Auc: 0.57910, Test Auc: 0.57072

40%|██████| 4/10 [05:13<07:45, 77.51s/it]

Epoch: 003, Loss: 0.36662, Train Auc: 0.67441, Val Auc: 0.59295, Test Auc: 0.59049

50%|██████| 5/10 [06:28<06:23, 76.66s/it]

Epoch: 004, Loss: 0.34265, Train Auc: 0.71249, Val Auc: 0.61112, Test Auc: 0.61277

60%|██████| 6/10 [07:44<05:04, 76.17s/it]

Epoch: 005, Loss: 0.33223, Train Auc: 0.74366, Val Auc: 0.62061, Test Auc: 0.62761

70%|██████| 7/10 [09:00<03:48, 76.07s/it]

Epoch: 006, Loss: 0.30576, Train Auc: 0.78091, Val Auc: 0.63387, Test Auc: 0.63352

80%|██████| 8/10 [10:15<02:31, 75.91s/it]

Epoch: 007, Loss: 0.28174, Train Auc: 0.81766, Val Auc: 0.64153, Test Auc: 0.64433

90%|██████| 9/10 [11:30<01:15, 75.63s/it]

Epoch: 008, Loss: 0.26621, Train Auc: 0.85218, Val Auc: 0.65124, Test Auc: 0.65225

100%|██████| 10/10 [12:45<00:00, 76.55s/it]

Epoch: 009, Loss: 0.24922, Train Auc: 0.88251, Val Auc: 0.65665, Test Auc: 0.66046

## Проверка результата с помощью примеров

```
In [46]: # Подход №1 - из датасета
evaluate(DataLoader(test_dataset[25:45], batch_size=10))
```

```
/usr/local/lib/python3.7/dist-packages/torch_geometric/deprecation.py:12: UserWarning: 'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
warnings.warn(out)
```

```
Out[46]: 0.7247191011235956
```

```
In [28]: # Подход №2 - через создание сессии покупок
test_df = pd.DataFrame([
    [-1, 15219, 0],
    [-1, 15431, 0],
    [-1, 14371, 0],
    [-1, 15745, 0],
    [-2, 14594, 0],
    [-2, 16972, 11],
    [-2, 16943, 0],
    [-3, 17284, 0]
], columns=['session_id', 'item_id', 'category'])

test_data = transform_dataset(test_df, buy_item_dict)
test_data = DataLoader(test_data, batch_size=1)

with torch.no_grad():
    model.eval()
    for data in test_data:
        data = data.to(device)
        pred = model(data).detach().cpu().numpy()

        print(data, pred)
```

```
100%|██████████| 3/3 [00:00<00:00, 183.19it/s]
```

```
DataBatch(x=[1, 1, 2], edge_index=[2, 0], y=[1], batch=[1], ptr=[2]) [5.360081e-06]
```

```
DataBatch(x=[3, 1, 2], edge_index=[2, 2], y=[3], batch=[3], ptr=[2]) [0.00379266 0.05972052 0.01434517]
```

```
DataBatch(x=[4, 1, 2], edge_index=[2, 3], y=[4], batch=[4], ptr=[2]) [4.1785872e-05 2.6933427e-04 1.6458357e-03 2.7660092e-03]
```

```
/usr/local/lib/python3.7/dist-packages/torch_geometric/deprecation.py:12: UserWarning: 'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
warnings.warn(out)
```

Как видно из результатов, значение метрики  $AUC = 72.5\%$

В ходе работы были изменены следующие гиперпараметры: количество эпох (5->10), скорость обучения (0.001->0.002), количество сессий (50000->60000)