Конкурс Авито-2017

Рекомендательная система

Основные особенности решения

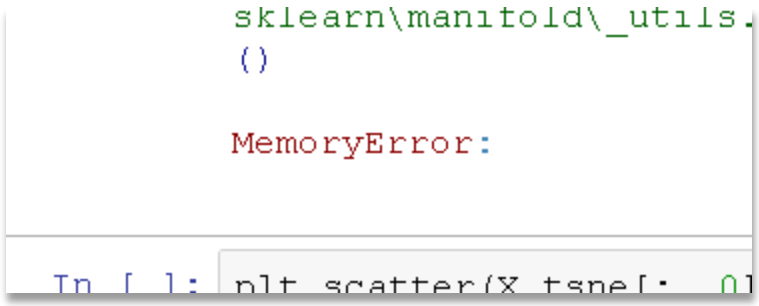
- Базовое решение, показавшее результат выше других решений (метод ближайших соседей, factorization machine)
- Использовались только Pandas и NumPy

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

Основные проблемы

- Большой объем данных



- Небольшое количество попыток в день

Изучение данных

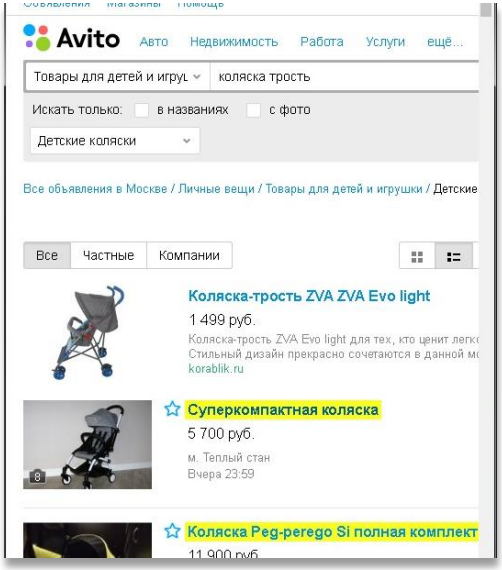
- Объединенные данные из train, item_data, user_searches, search_queries, item_titles помогли лучше понять поведение пользователей

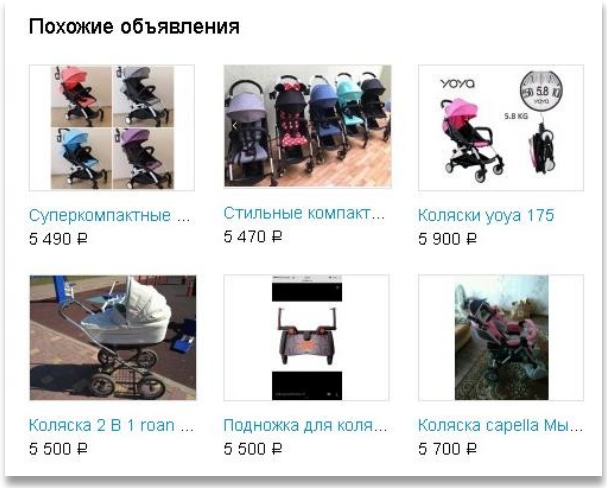
Out[17]:

	event_date	eventtype_id	item_id	location_id	microcat_id	query	user_id	title
10235812	2017-02-15 23:33:09	NaN	NaN	4294.0	6516.0	машина механическая	492549	NaN
10240133	2017-02-15 23:33:39	NaN	NaN	4294.0	6516.0	машина на аккумуляторе	492549	NaN
10240258	2017-02-15 23:33:40	57.0	448954.0	4294.0	503522.0	NaN	492549	Mazda 6, 2005
10241701	2017-02-15 23:33:50	NaN	NaN	4294.0	6516.0	тюль	492549	NaN
10241999	2017-02-15 23:33:52	57.0	2668422.0	4294.0	2715.0	NaN	492549	Тюль, нитяной занавес
15557719	2017-02-16 15:27:43	NaN	NaN	4294.0	6516.0	коляска трость	492549	NaN
15566380	2017-02-16 15:28:36	57.0	1909063.0	4294.0	259.0	NaN	492549	Коляска трость
15570918	2017-02-16 15:29:04	57.0	3066771.0	4294.0	259.0	NaN	492549	Коляска трость инфинити -китана

Основные предположения

- В основном, пользователи знают, что конкретно искать
- Рекомендации на сайте существенно влияют на поведение пользователей
- Объявления, для которых недавно были запрошены контакты продавца, имеют наибольшую значимость





Построение рекомендаций

- Объявления были отсортированы по нескольким признакам

```
In [4]: item_data.sort_values(by= ['microcat_id', 'location_id',
                                   'Param1', 'Param2', 'price',
                                   'ParamRE'], inplace= True)
```

- Объявлениям, с которыми пользователь совершал действия, задавались веса в зависимости от даты и типа действия

```
event_cost = {57: 1, 15: 2, 25: 2, 36: 5}
train_full['event_cost'] = train_full.eventtype_id.apply(lambda x: event_cost.get(int(x), 0))
train_full['user_item_cost'] = train_full.event_cost / train_full.days_passed
```

Построение рекомендаций

- Ближайшие по индексу в отсортированной item_data – наиболее близкие по признакам

```
for i in range(recomm_count):
    merge_user_item_cost['item+' + str(i + 1)] = merge_user_item_cost.item_index + i + 1
    merge_user_item_cost['item-' + str(i + 1)] = merge_user_item_cost.item_index - i - 1
```

- Актуальность рекомендации – количество дней с первого действия с рекомендованным объявлением

```
train_full.sort_values(by= 'event_date', ascending= False, inplace= True)
train_full.event_date = pd.to_datetime(train_full.event_date)
item_age = train_full.groupby(by= 'item_id').tail(1)
item_age['days'] = item_age.event_date.apply(lambda x: (pd.to_datetime('2017-02-22 23:59:59') - x) / np.timedelta64(1, 'D'))
```

Построение рекомендаций

- Выбираем наиболее актуальные рекомендации с большим весом, оставляем только 50

```
recom_df.sort_values(by= 'user_item_cost', ascending= False, inplace= True)
recom_df = recom_df.groupby(by= 'user_id')[['user_id', 'item_id']].head(50)
```

- ???
- PROFIT

СПАСИБО ЗА ВНИМАНИЕ!