КУРСОВ ПРОЕКТ

Препоръчваща система за състезанието Dressipi Recsys2022

Факултет по Математика и Информатика
Студент: Борислав Стоянов Марков
Факултетен номер: 0MI3400048
Учебен план: Изкуствен Интелект (редовно, магистър)
Курс: **Курс 1**; Група: **Група 1**
Активен период: 2021/2022 летен, магистри
Дисциплина: Препоръчващи системи

1. Съдържание

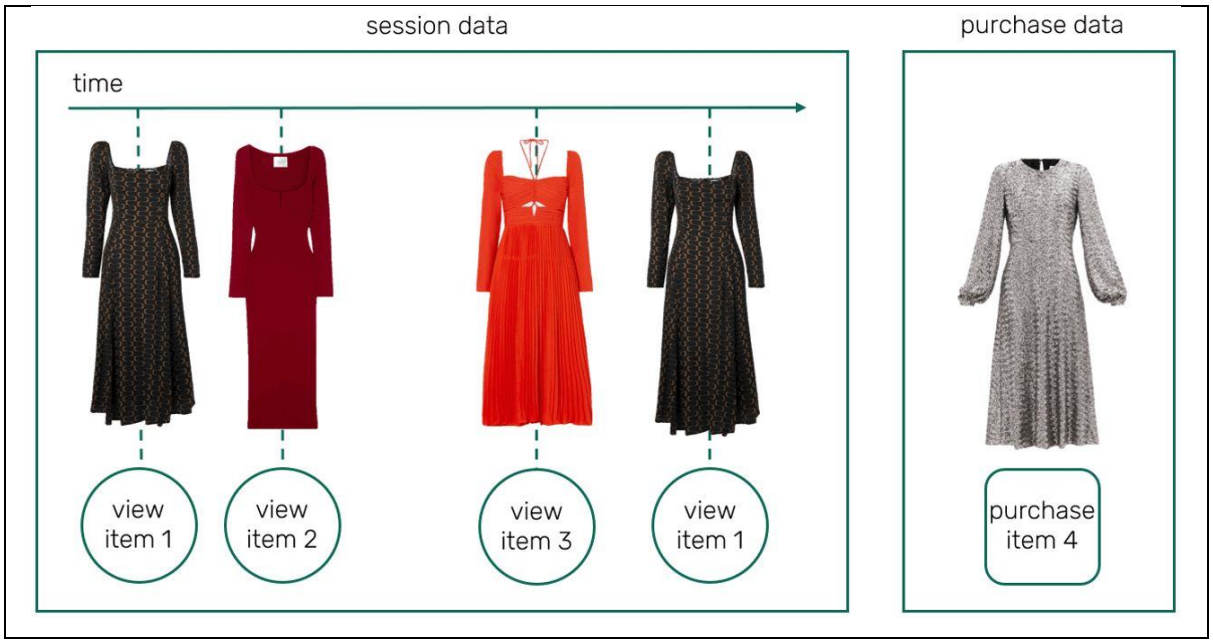
2. Увод

Проектът има за цел да се реализират получените от курса по препоръчващи системи знания. Dressipi е състезание от експерти по изкуствен интелект с участието на ETH университета в Цюрих. Ние като студенти имаме възможността да сравним резултатите си със световни лидери по темата като тези резултати ще бъдат показани публично.

2.1 За състезанието Dressipi 2022

Дрессипи(Dressipi) е организация от експерти в областта на изкуствения интелект, която продава софтуер на онлайн магазини(ритейл мрежа). Те вече имат необходимия софтуер и препоръчваща система. Целта им е да подобрят софтуера, като организират подобно състезание всяка година.

Таз-годишното състезание е да се подобри препоръчващия механизъм за модни онлайн сайтове. Когато даден потребител разглежда няколко стоки, алгоритъмът на участниците трябва да предвиди акуратно какво е купил този човек накрая на потребителската си сесия. Потребителите са анонимизирани и са представени само като сесии от по максиум един ден. Съдържанието на данните са проверени от модни експерти и данните са от високо качество. На фигура 2.1.1 виждаме примерна сесия на потребител, който е разгледал няколко рокли и е купил подобна рокля.



Фигура 2.1.1

Особености на задачата са:

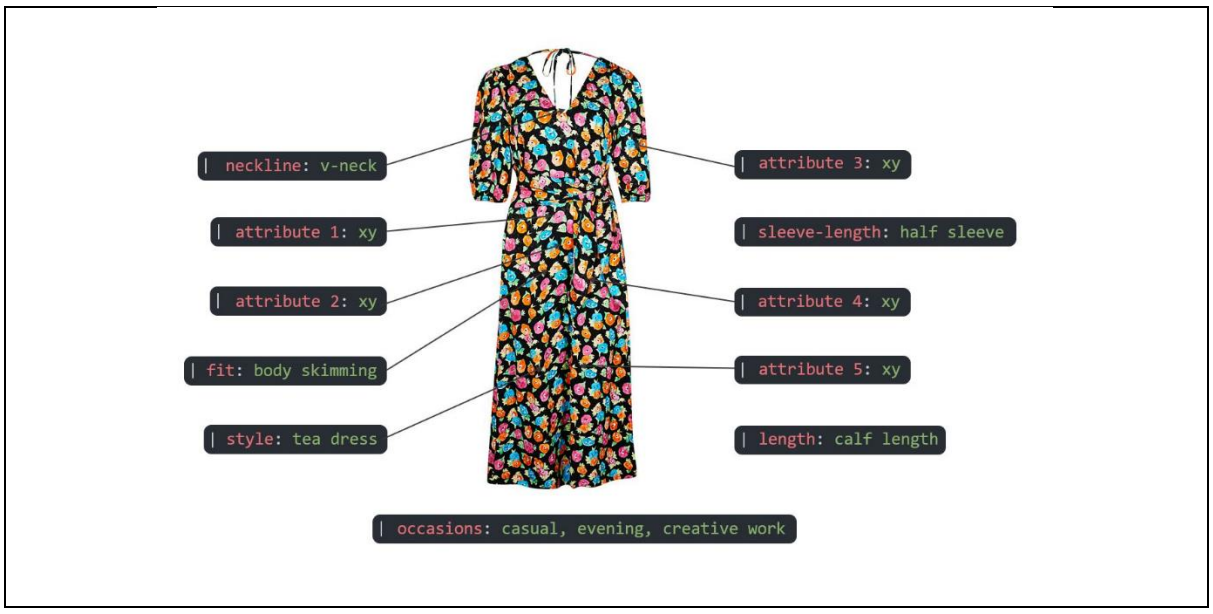
- потребителите са 51% нови и няма исторически данни и предложените стоки трябва да се основават само на текущата сесия
- за тези потребители, за които има исторически данни, знаем, че си сменят вкуса и предпочитанията много рязко и това се обяснява със спецификата на домейна – модата.

- Сесиите са твърде кратки и трябват акуратни предложения за потребителя, преди да е напуснал сайта

Като част от задачата Дрессипи са реализирали набор от данни от 1.1 милиона ритейл сесии, които са приключили с поръчка. Всички стоки са етикирани по категории. Данните са стоките ще наричаме фийчъри на стоките. Всички данни са анонимизирани.

- Сесия: стоките, които са прегледани от потребител, максимално за един ден.
- Поръчка: Поръчка, която е станала накрая на сесията, максимално е дадена една стока за поръчка.
- Фийчър на стоката: Етикета на някой атрибут на стоката, например : „цвят:зелен“, „деколте: V-образно деколте“ и т.н.

Примерна атрибутирана стока е показана в фигура 2.1.2



Фигура 2.1.1

Повече информация за състезанието и правилата можете да намерите в [1]. Сега да се спрем на данните и тяхното подробно описание.

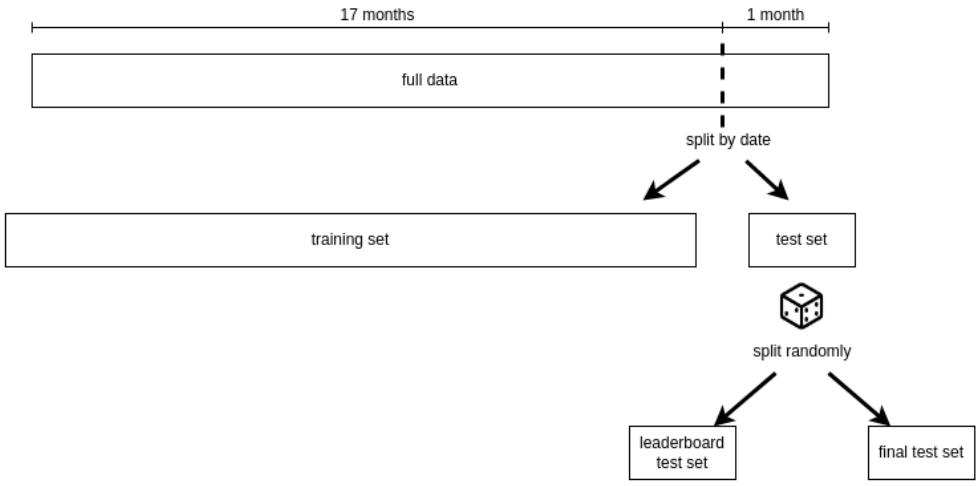
3 Корпус с данни

Описание на данните може е описано на сайта на Dressipi [2]. Данните са събрани от 1.1 милиона сесии от ритейл магазини за период от 18 месеца. Всички те са сесии в които

има поръчка накрая. Артикулите, които се разглеждат, са облекло и обувки. Целта е да се предвиди какво е купено накрая на сесията. Най-общо можем да кажем какво означават сесия, поръчка и атрибут на стоката:

- Сесия: стоките, които са прегледани от потребител, максимално за един ден.
- Поръчка: Поръчка, която е станала накрая на сесията, максимално е дадена една стока за поръчка.
- Атрибут(фийчър) на стоката: етикета на някой атрибут на стоката, например : „цвят:зелен“, „деколте: V-образно деколте“ и т.н.

Разделянето на тестово и обучаемо множество става като тестовото множесто е от 17 месеца и данните от последният месец се ползва за тестово множество, като се разделя още на две: стена на лидерите, и финални данни. Задачата е да се направят 100 предположения за всяка сесия. Тренировъчните сесии са 1 милион а тестовите „стена на лидерите“ и „финално множество“ са по 50 хиляди сесии.



Фигура 3.1

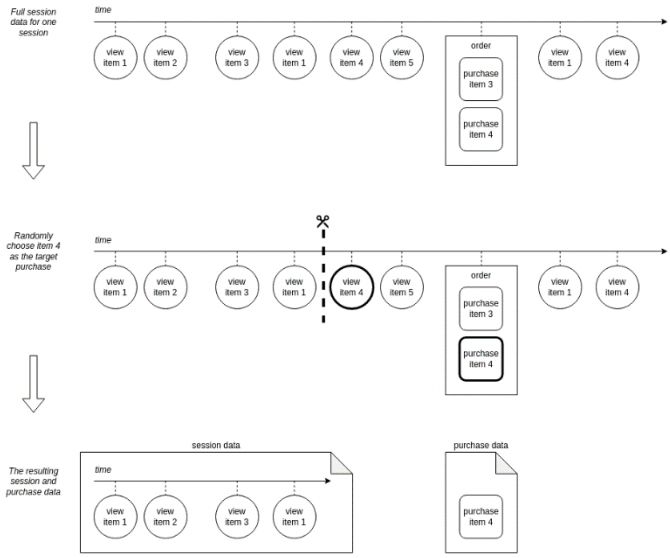
Подаването на решение се изисква във формат описан в таблица 3.2

```
session_id,item_id,rank
1,100,1
1,105,2
1,107,3
...
1,101,100
2,108,1
2,107,2
...
```

Таблица 3.2

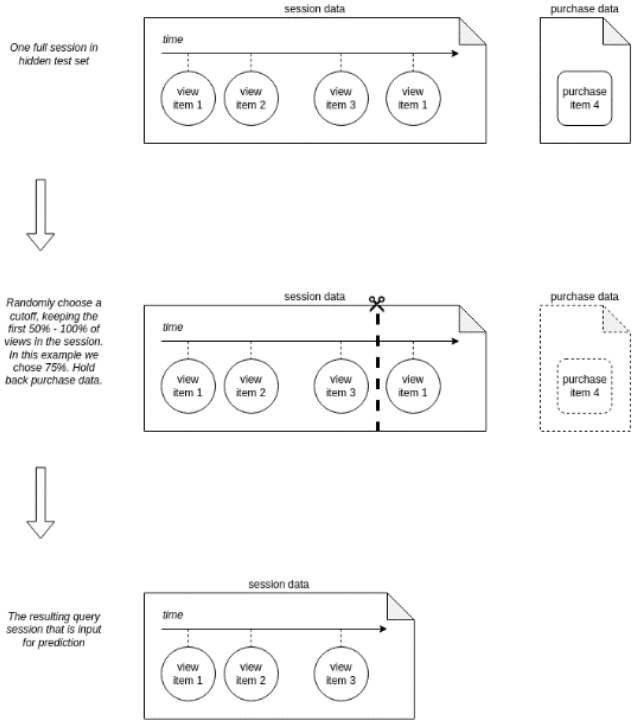
Метриката за оценяване е MRR (среден реципрочен ранк) Повече може да се прочете в [3].

Характеристиките на данните можем да кажем с няколко изречения. Данните са анонимизирани. При покупка на няколко стоки се взема предвид само първата. Много от сесиите имат само по едно две преглеждания преди покупка, но това е поради реалността и поради особеността как е конструирано множеството от данни. А именно при поръчка от няколко неща и продължаване на разглеждане, бъдещите интеракции не влизат в множеството за състезанието. А също и ако има разглеждане на артикула, който впоследствие е купен, той не влиза в данните за обучение. На следващата фигура е показана схема как се получават данните за сесиите и данните за покупките.



Фигура 3.3

Конструирането на данните за тестовите сесии се събира по следния начин. За всяка сесия се определя случаен процент x% между 50% и 100% и сесията се отрязва до там. Нарочно не се включва първият преглед на артикула, който е закупен накрая. Смисълът на това е да се предложи на потребителя стоки, които той би си купил и това предложение да е на по-ранен етап. Описаният алгоритъм за получаване на тестови сесии е показан на фигура 3.4.



Фигура 3.4

Сваляне на архива с данните за състезанието става само след регистрация. В архива, който се сваля има csv файлове и те изглеждат по този начин:

train_sessions.csv

	session_id	item_id	date
1	3	9655	2020-12-18 21:25:00.373
2	3	9655	2020-12-18 21:19:48.093
3	13	15654	2020-03-13 19:35:27.136
4	18	18316	2020-08-26 19:18:30.833

train_purchases.csv

	session_id	item_id	date
1	3	15085	2020-12-18 21:26:47.986
2	13	18626	2020-03-13 19:36:15.507
3	18	24911	2020-08-26 19:20:32.849
4	19	12534	2020-11-02 17:16:45.92

candidate_items.csv

	C1
1	item_id
2	4
3	8
4	9
5	19

item_features.csv

	item_id	feature_category_id	feature_value_id
1	2	56	365
2	2	62	801
3	2	68	351
4	2	33	802

test_final_sessions.csv

	session_id	item_id	date
1	61	27088	2021-06-01 08:12:39.664
2	96	11693	2021-06-19 17:48:05.227
3	96	18298	2021-06-19 17:49:08.589

test_leaderboard_sessions.csv

	session_id	item_id	date
1	26	19185	2021-06-16 09:53:54.158
2	200	17089	2021-06-25 12:23:40.811
3	200	17089	2021-06-25 12:24:36.631

Таблица 3.5

4. Анализ на данните

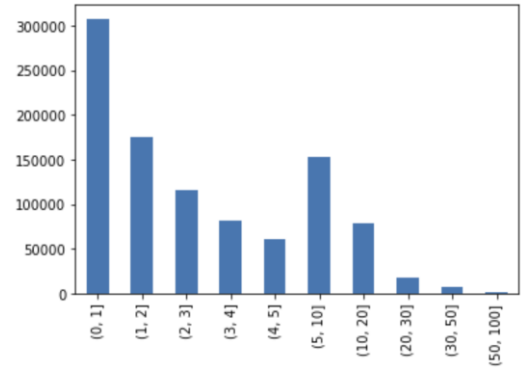
Анализът е направен с помощта на python и Jupyter Notebook средата. Можем един по един да покажем какво представляват CSV файловете, след като ги заредим в pandas.DataFrame, използвайки функцията “df.info()”. Ще покажем само само за по-важните „train_purchases.csv” и “ train_sessions.csv” в Таблица 4.1

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 3 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   session_id  1000000 non-null  int64
 1   item_id     1000000 non-null  int64
 2   date        1000000 non-null  datetime64[ns]
dtypes: datetime64[ns](1), int64(2)
memory usage: 22.9 MB
train_purchases_df.info() : None
```

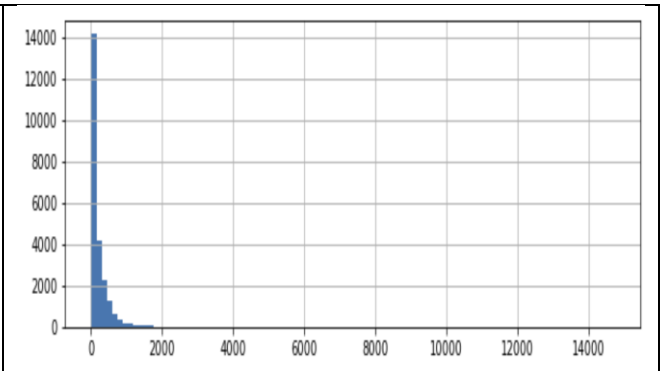
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4743820 entries, 0 to 4743819
Data columns (total 3 columns):
 #   Column      Dtype
---  ------      -----
 0   session_id  int64
 1   item_id     int64
 2   date        datetime64[ns]
dtypes: datetime64[ns](1), int64(2)
memory usage: 108.6 MB
train_sessions_df.info() : None
```

Таблица 4.1

Сега следва да видим как са разпределени прегледите по сесии и преглежданията по артикули – таблица 4.2.



а) интеракции по сесии



б) брой прегледи по артикул

Таблица 4.2

Средно имаме 4.74 интеракции във всяка сесия. От подточка а) виждаме, че най-много сесии има с по един преглед. От подточка б) Виждаме, че имаме някои много популярни артикули, докато други са преглеждани минимално и не са търсени. Можем да видим и как са прегледите по артикули:

```
train_sessions_df['item_id'].value_counts().describe()
count    23496.000000
mean       201.899047
std        362.441815
min          1.000000
25%         14.000000
50%         95.000000
75%        263.000000
max      14714.000000
Name: item_id, dtype: float64
```

Таблица 4.3

Прегледите по месеци и покупките по месеци са показани в таблица 4.4.

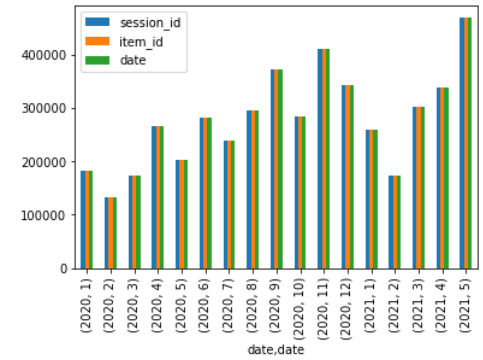
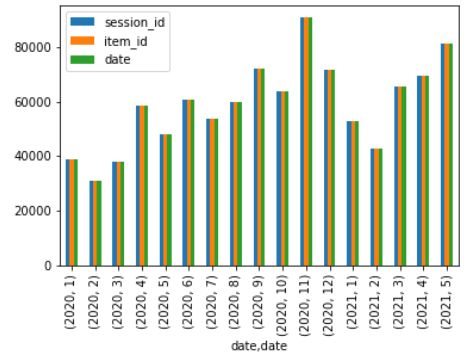
`# session views per month` `train_sessions_df.groupby([train_sessions_df["date"].dt.year, train_sessions_df["date"].dt.month]).count().plot(kind="bar")`	`# Purchases per month` `train_purchases_df.groupby([train_purchases_df["date"].dt.year, train_purchases_df["date"].dt.month]).count().plot(kind="bar")`
	

Таблица 4.4

Прегледите по година и час се виждат на следващата таблица. Силните часове за 2020г и 2021г са 19,20 и 21 часа. Часовете между 9 и 17 изглеждат еднакво натоварени. Останлите часове са слаби на преглеждания.

```
# session views per year and hour
train_sessions_df.groupby([train_sessions_df["date"].dt.year,
train_sessions_df["date"].dt.hour]).count().plot(kind="bar")
```

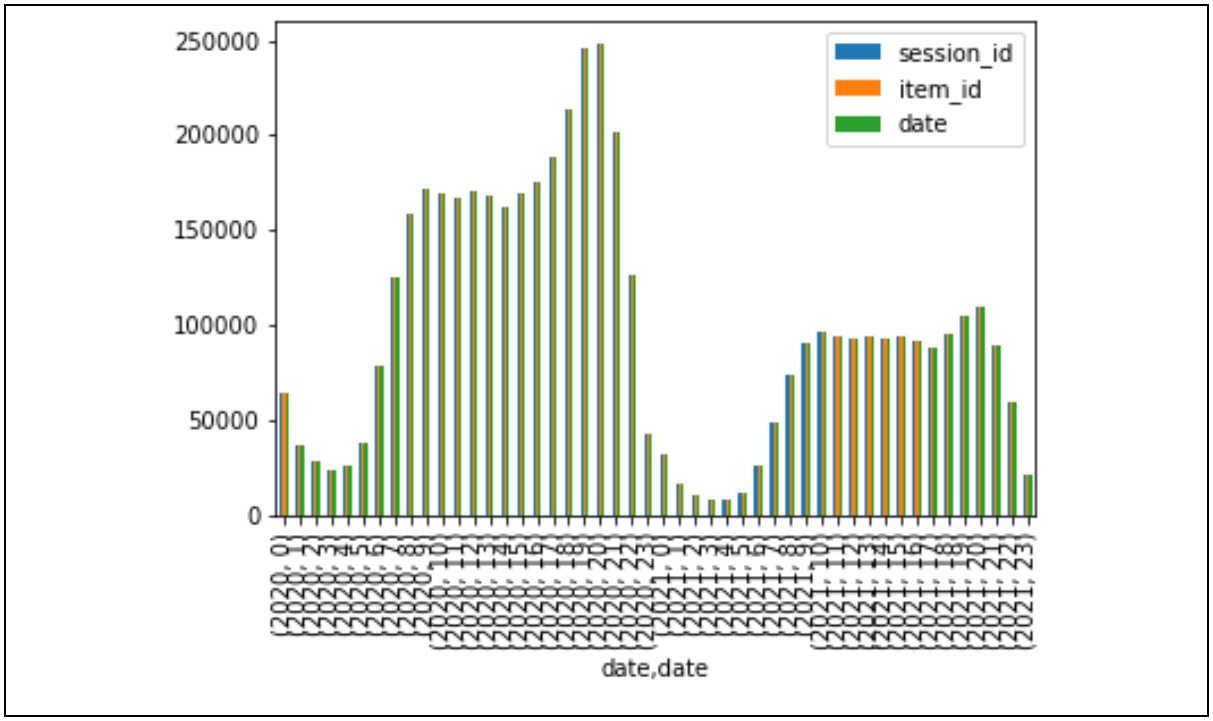


Таблица 4.5

Анализът на данни може да се види от „notebooks/EDA.ipynb“ в Приложения.

4.1 Отправна точка(baseline)

Като отправна точка за всеки алгоритъм е да се сравни с някой друг и това става като например даваме случайни предложения и измерим MRR (среден реципрочен ранк). За целта е разработена специална ноутбук папка „notebooks/Baseline.ipynb“ и там е реализиран случаен алгоритъм и замерване на 5000(поради ограничената RAM памет) елемента на случайно подбрана 5% извадка от поръчките. Вижда се, че MRR граничи около 0.01, което е и минимума, който е описан в състезанието.

Направен е и втори алгоритъм за базов, който ни препоръчва 100-те най-купувани стоки подредени в съответния ред. Това ни дава резултат **MRR=0.027**, което ще ни служи за отправна точка. Това е направено в „notebooks/Baseline_top100.ipynb“

5. Реализация с LightFM

LightFM е библиотека на python за хибридни препоръчващи системи. Тя работи добре при студен старт, както имаме в текущите данни, тъй като нямаме информация за потребителите и техните вкусове. Ноутбук папката с разработката е

„notebooks/lightfm.ipynb“ и там е кодът реализиращ препоръчваща система. Като цяло библиотеката работи с атрибути на артикулите(item_features), атрибути на потребителите(user_features) и потребителски взаимодействия(user_interactions).

5.1 Алгоритъм

Тъй като нямаме потребители, то тогава идентификаторите на сесиите ще са потребителите. Разделяме множеството с поръчките на трениращо и тестово множество.

```
train_p_df, test_p_df = train_test_split(train_purchases_df,
test_size=0.02)
len(train_p_df)=980000 train purchases
len(test_p_df)=20000   test purchases
```

Сесиите винаги са трениращо множество, защото там няма как да се тества. За атрибутиране на потребителите бихме могли да сложим месеца и часа на сесията, например “m6, h23”, или “m1, h11, h12”.

```
train_sessions_df['month']='m' +
train_sessions_df['date'].dt.month.apply(str)
train_sessions_df['hour']='h' +
train_sessions_df['date'].dt.hour.apply(str)

# Aggregate by sessionId and add hour and month as list of features
user_features = train_sessions_df.groupby('session_id').agg(
    session_id=pd.NamedAgg(column="session_id", aggfunc="min"),
    month=pd.NamedAgg(column="month", aggfunc=set),
    hour=pd.NamedAgg(column="hour", aggfunc=set),
).reset_index(drop=True).apply( \
        lambda row: (row['session_id'], list(row['month']) +
(list(row['hour']))), axis = 1)

user_features.head(2)
```

След направени експерименти се вижда, че слагане на фийчъри на потребителя **не носи добър резултат и те са премахнати**.

За стоките, имаме дадени категории и техните стойности, бихе могли да дадем атрибути на всяка стока нейните категории, конкатенирани със стойностите на категории. Например стока „3567“ ще има атрибути [“56-365”, “50-317”, “42-409”…] и т.н.

```
item_features_df['cat_val'] =
item_features_df['feature_category_id'].apply(str)+ '-' +
item_features_df['feature_value_id'].apply(str)

# Aggregate by item_id and add category and value as list of features
item_features = item_features_df.groupby('item_id').agg(
    item_id=pd.NamedAgg(column="item_id", aggfunc="min"),
    cat_val=pd.NamedAgg(column="cat_val", aggfunc=list),
).reset_index(drop=True).progress_apply( \
```

```
    lambda row: (row['item_id'], most_valuable(row['cat_val'])), axis =
1)

item_features.head(15)
```

При направените експерименти се вижда, че **слагайки атрибути това не дава добър резултат и затова ги махаме от експеримента**. Направени са следните експерименти.

Екс-пери-мент №	Фийчъри на стоката	Пример	Резултат MRR
1	(item_id,[cat1-val1, cat2-val2..]	(2, [56-365, 62-801, 68-351, 33-802, 72-75, 29... (3, [56-365, 69-592, 68-14, 17-378, 32-902, 11...	0.03
2	(item_id,[cat1-val1, cat2-val2..] Където cat-val са само измежду популярни категории	(2, [56-365, 62-801, 68-351, 33-802, 72-75, 29... (3, [56-365, 69-592, 68-14, 17-378, 32-902, 11...	0.03
3	(item_id,[cat1, cat2..] Използвани са само ИД на категориите без стойностите Където cat-val са само измежду популярни категории	(2, [56, 62, 68, 33, 72, 29... (3, [56, 69, 68, 17, 32, 11...	0.03
4	(item_id,[]) Използва се празен масив за фийчъри на стоката	(2, []) (3, [])	0.08

Таблица 5.1.1

Виждаме, че слагането на категории не носи резултат и затова всички фийчър вектори са премахнати (експеримент 4). Нямам обяснение за това, вероятно е замърсяване на модела с ненужна информация, защото при ръчно направен анализ се вижда, че измежду прегледаните стоки и закупената почти няма прилика, само една-две категории измежду 20 имат същата стойност.

LightFM изисква и масив на интеракциите. Ще получим този масив като използваме сесиите и поръчките и ги конкатенираме. Всяка интеракция трябва да получи тегло. Емпирично можем да кажем, че при преглед теглото ще е по-малко от 1, а при покупка ще е 1. Бихме могли да предположим, че потребителят се приближава все повече до желаната стока и следователно с всяко ново преглеждане можем да даваме повече точки за тази сесия и този артикул. Емпирично съм тествал няколко формули, които имат различен резултат:

Експеримент №	Формула	Резултат MRR
1	`train_sessions_df['weight'] = train_sessions_df['score'].apply(lambda x: np.tanh(x/4))`	0.04
2	`train_sessions_df['weight'] = train_sessions_df['score'].apply(lambda x: np.tanh(x/7))`	0.0759
3	`train_sessions_df['weight'] = train_sessions_df.progress_apply(lambda x: x['view_ord']/(x['views']+4), axis=1)`	0.0802

Таблица 5.1.2

При последният експеримент, вече не се ползва tanh а наслагването е линейно и пропорционално спрямо всички преглеждания за конкретната сесия.

Получените масиви се зареждат в LightFM модела и той се обучава.

```
model = LightFM(
    no_components=60,
    learning_rate=0.02,
    loss='warp',
    random_state=42)

model.fit(
    trn_interactions_ds,
    item_features=item_features_ds,
    user_features=user_features_ds,
    epochs=25, num_threads=8, verbose=True)

(user_map, feature_map, item_map, item_feature_map) = ds.mapping()
```

Отново са направени различни експерименти с размера на компонентите „no_components“.

Експеримент №	Компоненти	Резултат MRR
1	`no_components = 10`	0.02
2	`no_components = 30`	0.06
3	`no_components = 60`	0.08
4	`no_components = 70`	0.07

Таблица 5.1.3

Виждаме, че моделът започва да деградира след 60, затова остава на тази стойност.

За изчисляване на резултатите избираме 2500 измежду най-продаваните стоки, тъй като ако сметнем всички възможни стоки в комбинация със всички възможни сесии, то не би ни стигнала RAM паметта, но това е само за тест, при предаване за Дъската на лидерите няма да има този проблем.

```
# Recommend from most sold items
items_to_predict_array = \
     list(train_sessions_df['item_id'].value_counts()[0:2500].keys())
```

Функцията predict_ratings има за цел да получи готов DataFrame във вид за публикуване на резултатите. Реално се подготвят масиви с идентификатори на LightFM и после трябва да се обърнат във идентификатори, които ние ползваме.

```
def predict_ratings(model, sessions_or_purchase_df, items_to_predict) :
    session_ids =
pd.Series(sessions_or_purchase_df['session_id'].unique())
    all_users = session_ids \
        .apply(lambda x: user_map[x]).to_numpy()
    all_available_items = items_to_predict.apply(lambda x:
item_map[x]).to_numpy()
    users =[]
    items =[]
    for user_item_tuple in tqdm(product(all_users, all_available_items)):
        users.append(user_item_tuple[0])
        items.append(user_item_tuple[1])
    preds = model.predict(np.array(users), np.array(items))

    session_ids_expanded = []
    item_ids_expanded = []
    for tup in tqdm(product(session_ids, items_to_predict)):
        session_ids_expanded.append(tup[0])
        item_ids_expanded.append(tup[1])


    df_score = pd.DataFrame({'session_id':
np.array(session_ids_expanded), \
```

```
                        'item_id':np.array(item_ids_expanded) , \
                        'score':np.array(preds)})
    return df_score
```

Тук времеемките цикли са от декартовото произведение на **product(all_users, all_available_items)** и подаването към LightFM модела.

Смятането на MRR е със следната функция, като по подразбиране ранка е 100 ако стоката не е в първите 100 предложения.

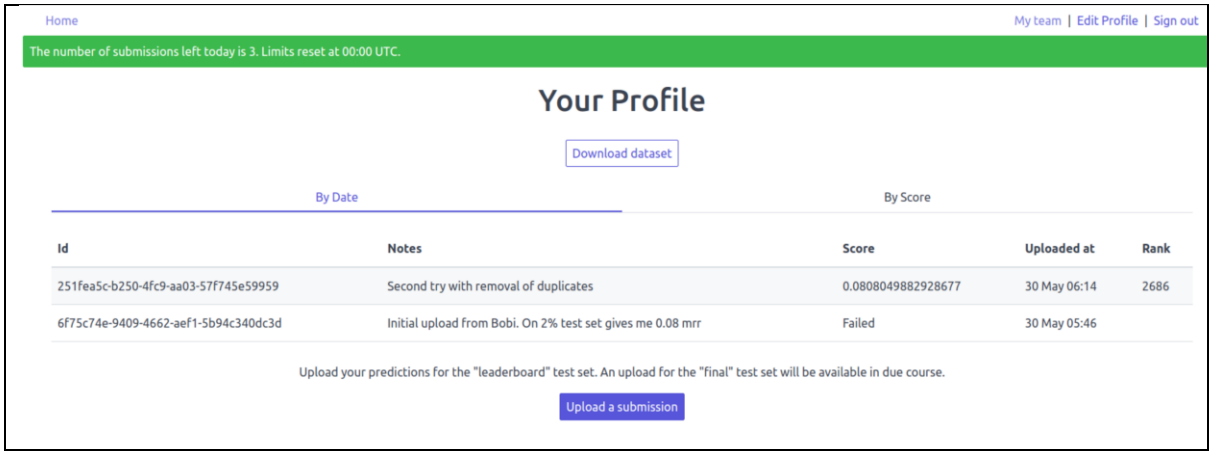
```
# Calculate Mrr (Mean reciprocal rank)
def calc_mrr(result_df, test_df):
    mrr = 0
    # Iterate all sessions
    for sess_id in tqdm(test_df['session_id']):
        # Make view for only this session with all ranked
        ranked =
result_df[result_df['session_id']==sess_id]['item_id'].reset_index(drop=T
rue)
        real_item_id =
test_df[test_df['session_id']==sess_id]['item_id'].reset_index(drop=True)
[0]
        first_rank = 100
        found_t = ranked[ranked == real_item_id]
        if len(found_t)!=0 :
            first_rank = found_t.index[0]+1

        mrr =mrr+ 1/first_rank

    mrr = mrr / test_df['session_id'].nunique()
    return mrr
```

След пресмятане на MRR над тестовото множество имаме резултат 0.0811.

Само за проба тестваме как се държи модела върху обучаваните данни и виждаме, че резултата над едно избрано множество от 10000 сесии от обучаваните данни ни дава резултат MRR=0.179, което е очаквано, тъй като това са виждани данни, но е едно потвърждение, че модела работи правилно, макар и не давайки добри резултати.

5.1.1 Предаване на резултата към Дъска на Лидерите

Предаването на резултат изисква да се обработи още един масив с интеракции „test_leaderboard_sessions.csv“ и да се премахне разделянето на обучаемо и тестово множество, защото всичко дадено е обучаемо множество и нямаме достъп до тестово множество. Всичко това е направено в отделен ноутбук „lightfm_for_leaderboard.ipynb“.

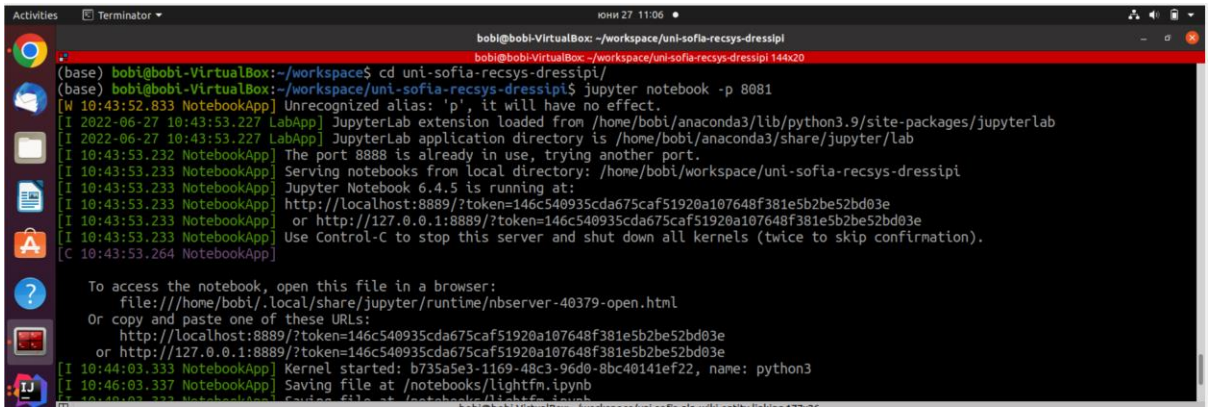


Фигура 5.1.1.1

Виждаме, че след предаване на резултата има същият MRR , както и при тестовете, което е още една гаранция, че няма допуснати технически грешки, и че данните са наистина добре подбрани и истински.

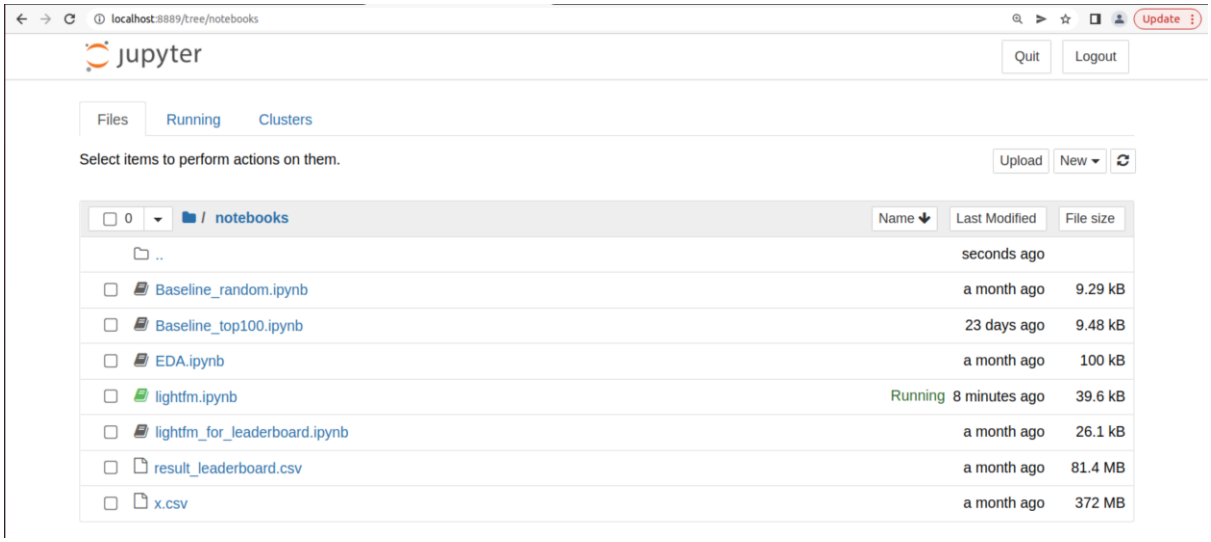
5.2 Използване на приложението

Приложението се използва като локално на Линукс се инсталира jupyter notebook, който идва по подразбиране със средата Anaconda.



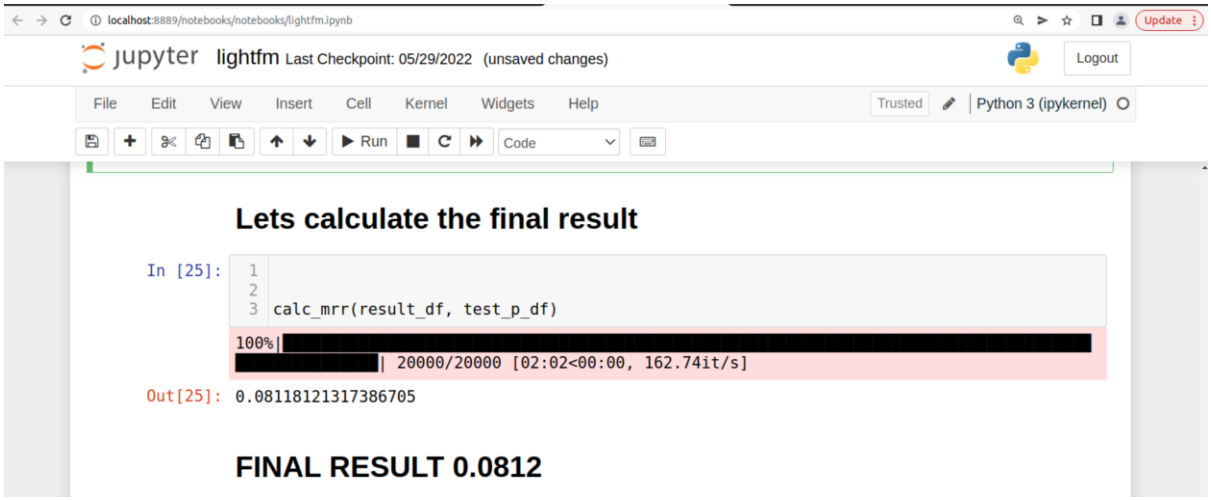
Фигура 5.2.1

След командата ‚jupyter notebook -p 8081’, което ни стартира сървър на порт 8081, отваряме браузър в текущата пака и можем да стартираме ноутбука.



Фигура 5.2.2

Отваряме например **lightfm.ipynb** и можем да стартираме всичко от началото, но след като свалим ZIP файла с данните, който не е публикуван в github поради лицензни съображения.



Фигура 5.2.3

5.3 Оценка на резултатите

Резултатите не са впечатляващи, защото **MRR=0.0812** не се класира след водещите резултати. Стигнахме до извода, че подаването към лидерборда също е с MRR=0.08, което ни дава увереност, че правилно сме предвидили какви точки ще имаме и данните са наистина добре подбрани от специалисти. Направихме оценка на тестовите данни и там имаме MRR=0.179, което означава, че правилно провеждаме експериментите, тъй

като това са данните за трениране и е нормално да е повече MRR. Библиотеката LightFM не работи добре с добавени фийчър вектори на стоките и потребителите, което е озадачаващо, вероятно има някакъв проблем с имплементацията ѝ.

8. Недостатъци и подобрения

С направените експерименти не можахме да постигнем добри резултати с LightFM. Може да се изпробва друга библиотека за подобен вид задачи. Като подобрения, можем да пробваме ансамблови алгоритми като Boosting или Stacking, което вероятно ще подобри резултата.

9. Източници и използвана литература

[1] The RecSys Challenge 2022, Dressipi, Bruce Ferwerda (Jönköping University, Sweden), Saikishore Kalloori (ETH Zürich, Switzerland), and Abhishek Srivastava (IIM Jammu, India).
http://www.recsyschallenge.com/2022/#participation

[2] RecSys Challenge 2022 Dataset,
http://www.recsyschallenge.com/2022/dataset.html

[3] Mean Reciprocal Rank, Wikipedia,
https://en.wikipedia.org/wiki/Mean_reciprocal_rank

Приложения

1. Сорс код (Source code)

Кодът е публичен и качен в платформата Гитхъб.

https://github.com/borkox/uni-sofia-recsys-dressipi