1 Introduction

1.1 Background

A Recommender system (RS) is an intelligent computer-based technique that predicts on the basis of users' adoption and usage and helps them to pick items from a vast pool of online stuffs. Most internet users surely have happened upon an RS in some way. For instance, Facebook recommends us, prospective friends, YouTube recommends us the videos in accord, Glassdoor recommends us matching jobs.

Recommender system (RS) has emerged as a major research interest that aims to help users to find items online by providing suggestions that closely match their interests, and it's a subset of information filtering system. Typically, recommendation systems are designed and assessed using past user-item records. However.

1.2 Objectives

the majority of offline recommendation datasets are quite sparse and have a variety of biases. This impedes the assessment of policy recommendations Existing initiatives try to enhance data quality by gathering user preferences on randomly selected items, but one of the most problems in designing the recommendation systems is how should we evaluate the model accurately, and although that there are good methods but it's time and money-consuming and entails the risk of failure, also static recommendation systems have limitations regarding to capturing precious real-time user preferences, so our objectives here is to provide recommendations based on recorded information on the users' preferences with this fully observed dataset and evaluate our results accurately, also we want to demonstrate the efficacy and advantage of KuaiRec dataset.

for videos recommendation systems in general we have three important questions.

1. **What's the watch_ratio of video for each pair of user and video?**

   Answering to this question will help a lot of foundations like advertisement agents while the answer will tell them where they should put their advertisements and will help the content creator.

2. **Does the user like the video or dislike it?**

   Like the previous question Answering this question will help advertisement agents because the answer will tell them what's the most likes videos and where they should put their advertisement, also it will help the recommendation system engine to recommend videos to users like they pressed like button on them based on another features like tags.

3. **Are the shorter videos {less than 2 minutes} are more favorite than (more than) longer videos for the users?**

   Answering these questions will help the app founder to determine what's the best video duration that users like, and how each user will stay in the app, also they will recommend these videos to more people, and it'll appear at the top of the page to the users, also it will help content creators to choose the best video duration to attract more users to their videos.

2 Dataset

2.1 Dataset description

Dataset which used in this project based on [3] is obtained from the Kuaishou App, a well-known short-video platform with a lot of subscribers. Users may view a range of short-form videos from different categories such as dance, entertainment, and fitness/sports on this site. The videos are ordered by suggesting streaming, our dataset here is the first dataset generated from real-world recommendation logs with all users' preferences on items known and also it's not highly sparse dataset, this data is a fully-observed dataset with millions of users' interaction, the data collected here by the users themselves, every single user watch a video and left a feedback which means that we haven't almost no missing values here, based on [3] analysis, this dataset is dubbed as the small matrix for convenience, can be used for evaluation the trained model. And for the training purpose, they further collect a larger dataset, represented as big matrix, which contains additional interactions for the users and items in the small matrix, and based on one [3] analysis they found that user-item interactions in the small matrix are excluded from big matrix to separate the training and evaluation data, since the evaluation process in the recommendation system design is very important, also they collect side information of users and items each data in a separate file. for the user side, they collected the social relationship among these users. On the item side, we list each item's tags, i.e., a set of categories such as {Gaming, Sports}.

2.2 Files' structure

- KuaiRec
  - Data
    - Big_matrix.csv
    - Small_matrix.csv
    - Social_network.csv
    - Item_feat.csv

2.3 Big matrix and small matrix comparison

	#Users	#Items	#Observations	#Attributes of items	#Users who have friends	Density
small matrix	1,411	3,327	4,676,570	31	146	99.6%
big matrix	7,176	10,729	12,530,806	31	472	13.47%

Table 1: The statistics of the small matrix and big matrix in KuaiRec

2.4 Descriptions of the features in big_matrix.csv and small_matrix.csv

Feature	Description	Type	Example
user_id	The ID of the user	int64	5204
video_id	The ID of the viewed video	int64	3650
play_duration	Time of video viewing of this interaction (millisecond).	int64	13838
video_duration	Time of this video (millisecond)	int64	10867
time	Human-readable date for this interaction	str	"2020-07-05 00:08:23.438"
date	Date of this interaction	int64	20200705
timestamp	Unix timestamp	float64	1593878903.438
watch_ratio	video watching ratio (play_duration/video_duration)	float64	1.273397

Table 2: big_matrix.csv and small_matrix.csv features description

Data quality assurance: There aren't missing values in the **big_matrix.csv** dataset but it has **965819** consistent duplicated values, and for **small_matrix.csv** dataset, there are missing values in three features in this dataset but fortunately, there are no duplicated values in this dataset.

	Total missing	Missing %
user_id	0	0.0
video_id	0	0.0
play_duration	0	0.0
video_duration	0	0.0
time	0	0.0
date	0	0.0
timestamp	0	0.0
watch_ratio	0	0.0

Table 3: big_matrix.csv missing value analysis using pandas package

	Total missing	Missing %
time	181922	3.89157
date	181922	3.89157
timestamp	181922	3.89157
user_id	0	0.0
video_id	0	0.0
play_duration	0	0.0
video_duration	0	0.0
watch_ratio	0	0.0

Table 4: small_matrix.csv missing value analysis using pandas package

2.5 Descriptions of the features in social_network.csv

Feature	Description	Type	Example
user_id	The ID of the user	int64	5204
friend_list	the list of ID of the friends of this user.	list	[4202,7126]

Table 5: social_network.csv features description

You may want to note that there's a common feature between social_network.csv and the two main datasets which is user_id feature and we can join between these two datasets using this key to recommend the video to users' friends also. Note: There isn't neither missing values nor consistent duplicated values in social_network.csv dataset

Data quality assurance: There are neither missing values nor consistent duplicated values in social_network.csv dataset

	Total missing	Missing %
user_id	0	0.0
friend_list	0	0.0

Table 6: social_network.csv missing value analysis using pandas package

2.6 Descriptions of the features in item_feat.csv

Feature	Description	Type	Example
video_id	The ID of the video	int64	1
feat	the list of tags of this video	list	[27,9]

Table 7: item_feat.csv features description

You may want to note that there's a common feature between item_categories.csv and the two main datasets which is video_id feature and we can join between these two datasets using this key to get insights about the categories that's user likes. Note: There isn't neither missing values nor consistent duplicated values in item_feat.csv dataset

Data quality assurance: There are neither missing values nor consistent duplicated values in item_feat.csv dataset.

	Total missing	Missing %
video_id	0	0.0
feat	0	0.0

Table 8: item_feat.csv missing value analysis using pandas package

Outliers and noises analysis for all datasets: all numeric features in all datasets have outliers but there are not any dataset has noise datapoint and the outliers here we shouldn't drop them because there are have informative values like (play_duration) feature, so, we want to deal with them carefully.

3 Basic Data Exploration

3.1 Have a look on each file data

3.1.1 big_matrix.csv

	user_id	video_id	play_duration	video_duration	time	date	timestamp	watch_ratio
10293609	5874	4613	6825	6040	2020-08-10 09:30:09.088	20200810	1.597023e+09	1.129967
10253253	5850	9914	8410	9034	2020-08-01 14:14:32.156	20200801	1.596262e+09	0.930928
10950882	6263	6794	1724	26239	2020-07-12 14:25:12.511	20200712	1.594535e+09	0.065704
1747770	978	6553	19830	8399	2020-09-04 02:23:23.094	20200904	1.599157e+09	2.360995
3893706	2234	296	10076	12867	2020-08-04 18:07:29.246	20200804	1.596536e+09	0.783089
...	...	...	...	...	...	...	...	...
9742696	5548	5266	3296	9000	2020-07-05 07:21:19.978	20200705	1.593905e+09	0.366222
11575490	6622	6222	0	6271	2020-08-27 21:01:29.729	20200827	1.598533e+09	0.000000
7309401	4157	2094	6544	8534	2020-07-08 00:40:48.701	20200708	1.594140e+09	0.766815
6315130	3603	7280	7354	8034	2020-08-01 06:56:15.947	20200801	1.596236e+09	0.915360
10765569	6142	1410	8904	6421	2020-09-04 00:27:45.97	20200904	1.599150e+09	1.386700

Figure 1: some random samples from big_matrix.csv

3.1.2 small_matrix.csv

	user_id	video_id	play_duration	video_duration	time	date	timestamp	watch_ratio
3928128	5992	8563	10929	6734	2020-07-20 01:29:33.179	20200720.0	1.595180e+09	1.622958
589440	986	7671	11794	6270	2020-08-15 15:07:59.124	20200815.0	1.597475e+09	1.881021
3284190	5062	1416	7058	9867	2020-08-25 08:53:28.43	20200825.0	1.598317e+09	0.715314
4324394	6567	8547	3263	8217	2020-08-15 16:48:37.085	20200815.0	1.597481e+09	0.397104
4497919	6876	9558	5800	7300	2020-07-14 10:01:36.478	20200714.0	1.594692e+09	0.794521
...	...	...	...	...	...	...	...	...
809531	1308	585	5850	10311	2020-07-20 06:44:14.241	20200720.0	1.595199e+09	0.567355
1028471	1668	2303	10000	6167	2020-07-23 07:04:39.967	20200723.0	1.595459e+09	1.621534
2926381	4642	3267	4957	19321	2020-09-02 01:55:49.579	20200902.0	1.598983e+09	0.256560
2224276	3506	429	14476	8817	2020-07-11 09:15:16.227	20200711.0	1.594430e+09	1.641828
1282558	2017	3127	4788	11308	2020-08-26 19:33:08.066	20200826.0	1.598442e+09	0.423417

Figure 2: some random samples from small_matrix.csv

3.1.3 social_network.csv and item_feat.csv

	user_id	friend_list
158	6866	[2212, 4925]
268	476	[6229, 3694, 2187, 1128, 6164]
449	5547	[3354]
406	2650	[4356]
462	3312	[4636]
...	...	...
36	1485	[2361]
320	237	[5646]
228	2524	[5738]
76	990	[1102]
191	2870	[765]

	video_id	feat
5174	5174	[12]
871	871	[26]
8568	8568	[6]
3140	3140	[30, 11]
4791	4791	[11]
...	...	...
10341	10341	[11]
9212	9212	[6]
4220	4220	[6]
2392	2392	[26]
3224	3224	[19]

Figure 3: some random samples from social_network.csv
And item_feat.csv

3.2 Every dataset statistics

3.2.1 big_matrix.csv statistics

	user_id	video_id	play_duration	video_duration	date	timestamp	watch_ratio
count	12530810	12530810	12530810	12530810	12530810	12530810	12530810
mean	3574.377	5058.597	9027.027	1462.157	20200800	1596799000	0.9445059
std	2067.008	3090.082e	15473.43	19834.74	50.80192	1514698	1.674601
min	0.0	0.0	0.0	140.0	20200700	1592872000	0
25%	1788.0	2388.0	4218.000	7434.0	2.0200800	1596339000.0	0.3148246
50%	3578.0	4823.0	7277.0	9636.0	2.0200810	1596669000	0.7234710
75%	5343.750	7601.0	1035.0	1217.900	20200830	1.598502000	1.177644
max	7175.0	10728.0	999639.0	315072.0	20200900	1599694000	573.4571

Table 8: big_matrix.csv statistics

3.2.2 small_matrix.csv statistics

	user_id	video_id	play_duration	video_duration	date	timestamp	watch_ratio
count	4676570	4676570	4676570	4676570	4494578	4494578	4676570
mean	3631.649	4975.787	8612.637	14486.45e	20200770	1596241000	0.9070695
std	2043.873	3064.837	12236.61	20467.11	48.95180	1254444	1362324e
min	14.0	103.0000	0.0	3067.000	20200700	1593801000	0.0
25%	1834.0	2370.0	5811.000	7523.0	20200720	1595210000	0.4675769
50%	3687.0	4693.0	7549.0	9600.0	20200800	1596224000	0.7691666e
75%	5421.0	7475.0	9880.0	11934.0	20200810	1597121000	1.120590
max	7162.000	10596.00	7988155	315072.0	20200900	1599321000	571.5214

Table 9: small_matrix.csv statistics

3.2.3 social_network.csv and item_feat.csv statistics

Note: in this dataset statistics we don't care about the mean and the median since these statistics doesn't important and suitable for information like ID, and there are friend list and feat feature and because it's a list datatype, so I considered the length of each list and I construct another feature calledfriend_list_len for social_network.csv dataset and feat_len for item_feat.csv.

	user_id	friend_list_len
count	472.0	472.0
min	18.0	1.0
25%	1648.0	1.0
50%	3268.0	1.0
75%	5233.5	2.0
max	7174.0	5.0

Table 10: social_network.csv statistics

	video_id	friend_list_len
count	10729.0	10729.0
min	0.0	1.0
25%	2682.0	1.0
50%	5364.0	1.0
75%	8046.0	1.0
max	10728.0	4.0

Table 11: item_feat.csv statistics

3.2.4 some EDA about big_matrix.csv and small_matrix.csv

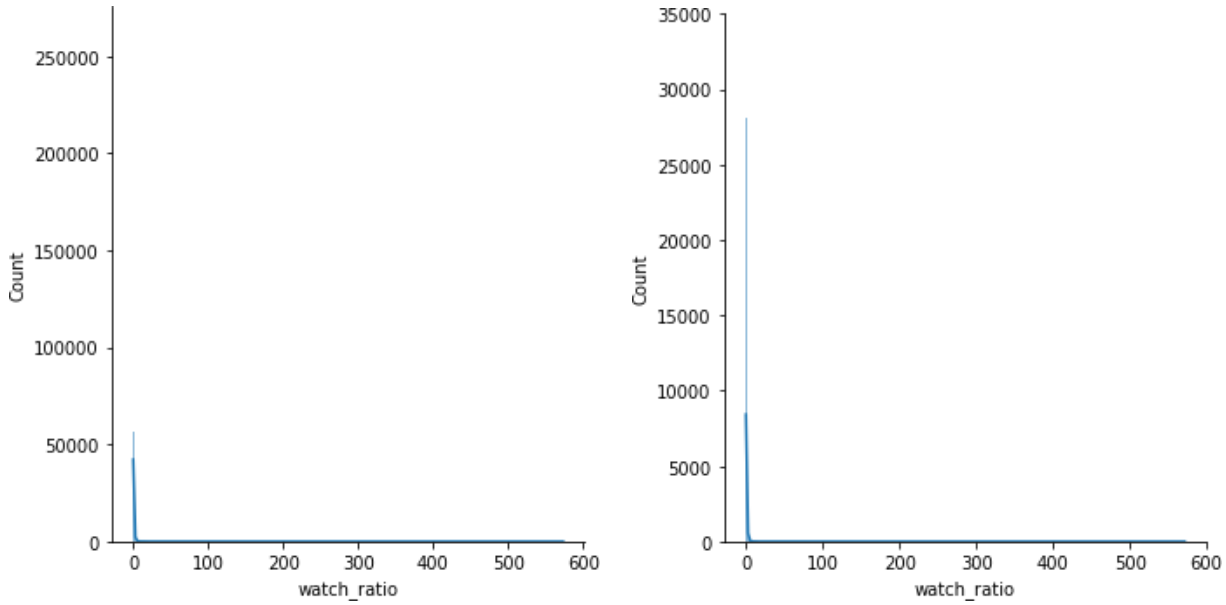

Figure 4: watch ratio data distribution in big_matrix.csv {left} and small_matrix.csv {right}

3.2.4.1 Distribution of watch_ratio in big matrix and in the small_matrix.csv Although that we can't determine the actual distribution from this figure, but we notice that the frequency starts to decrease from value 6 so we will decrease the range and make it from 1 to 6 to have a better overview.

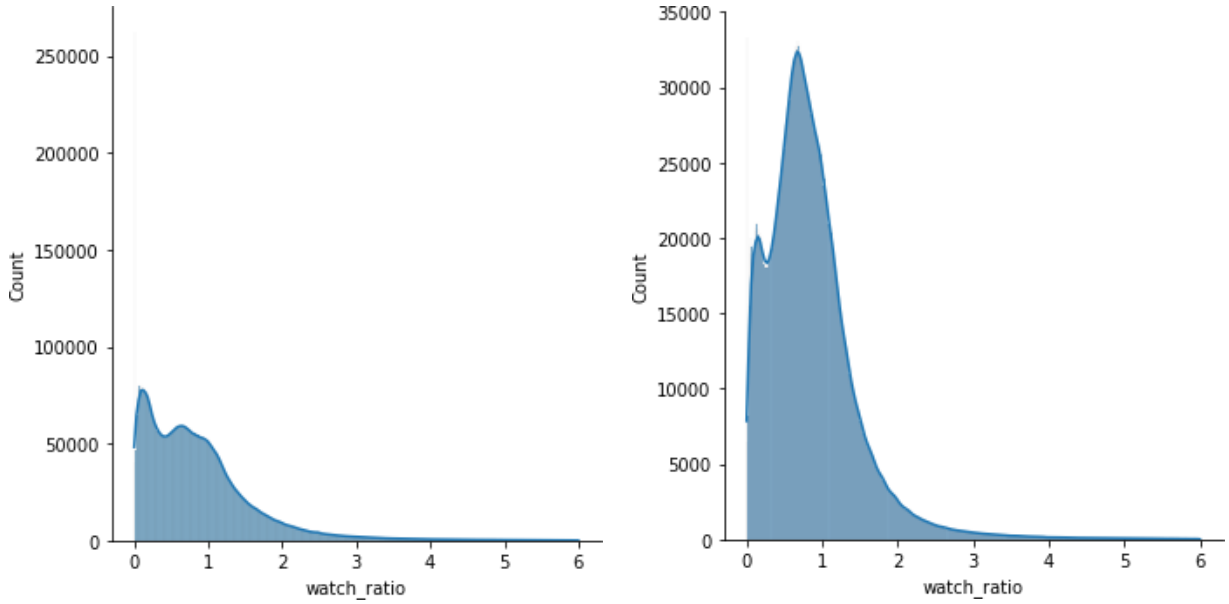

Figure 5: Ranged watch ratio data distribution in big_matrix.csv {left} and small_matrix.csv {right}

Now we noticed that the count of watch ratio decreased starting from 5 and they aren't normally distributed.

3.2.4.2 Distribution of video_duration in big matrix and in the small_matrix.csv Same as the watch_ratio feature, we can't determine the actual distribution from this figure, but we notice that the frequency starts to decrease from value 50000 so we will decrease the range and make it from 0 to 50000 to have a better overview.

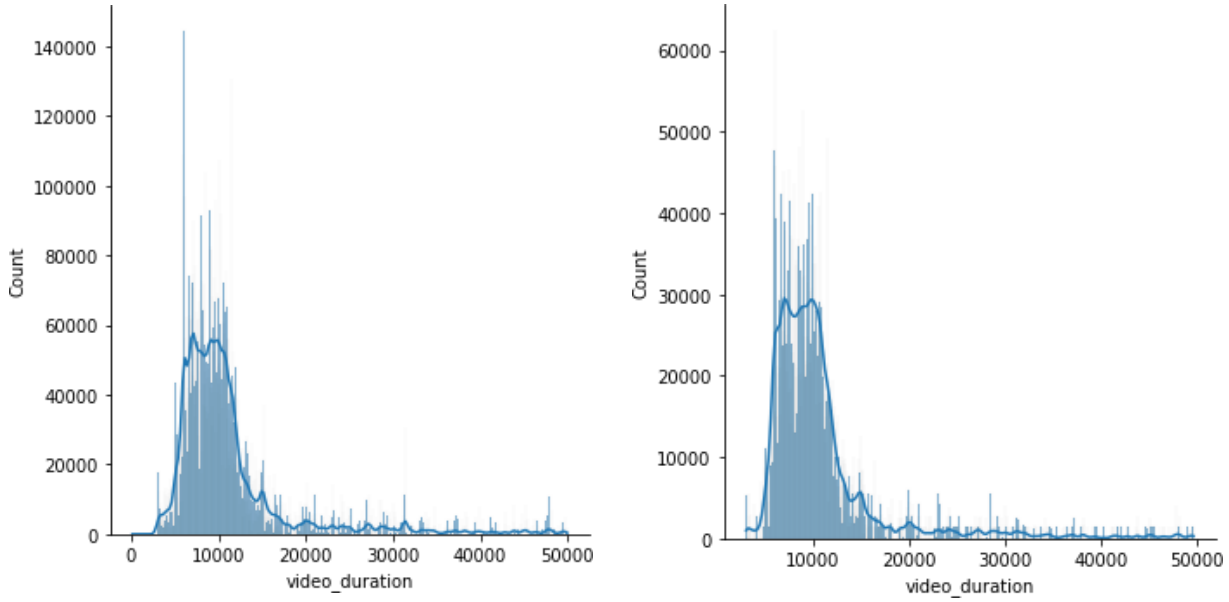

Figure 6: Ranged watch ratio data distribution in big_matrix.csv {left} and small_matrix.csv {right}

Now we noticed that the count of watch ratio decreased starting from 0 to 20000 is normally distributed.

3.2.5 some EDA about social_network.csv

In this dataset we care about how many friends does each user have?

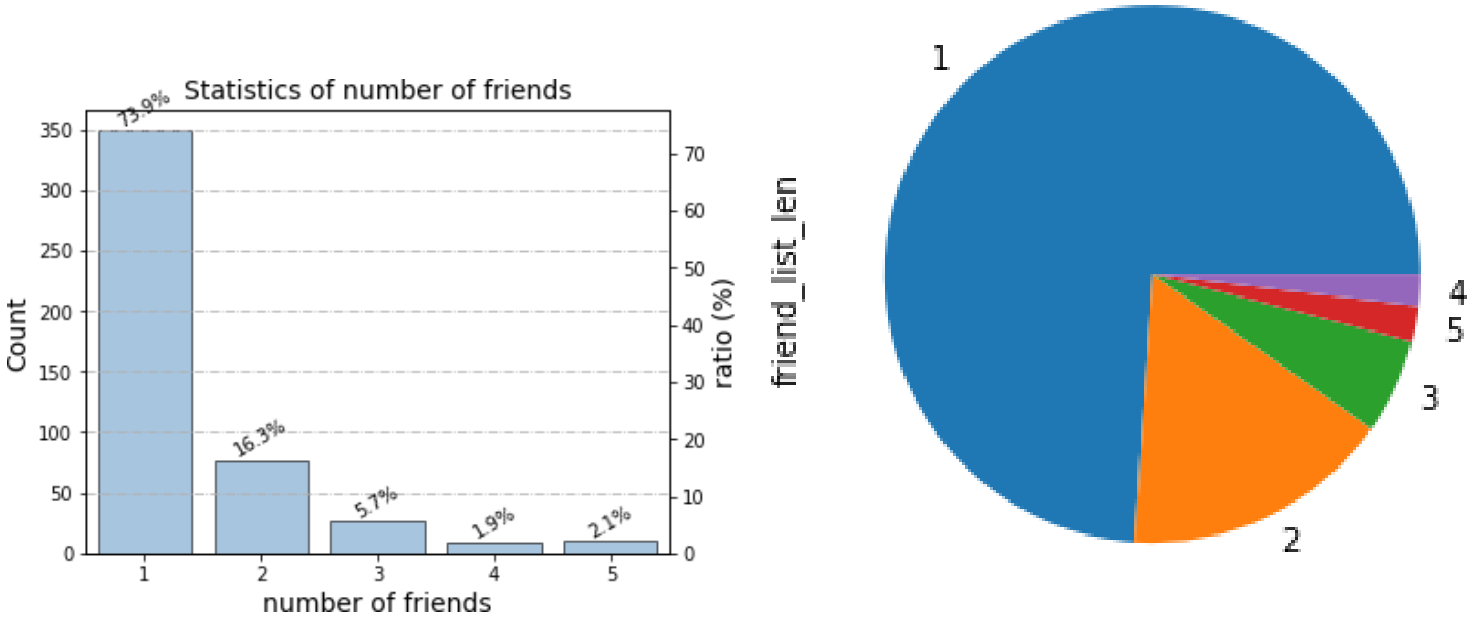

Figure 7: frequency of number of friends in each user

from above plots we see that the most users have only one {73.9% of the users} friend in thisplatform and there's not user have more than 5 friends.

3.2.6 some EDA about item_feat.csv

In this dataset we care about how many tags each video has because that may help us if the increasing in the number of tags make users happy or not, so we get the count of the tags in each video and we got the frequency of this counts to see which is the most frequent.

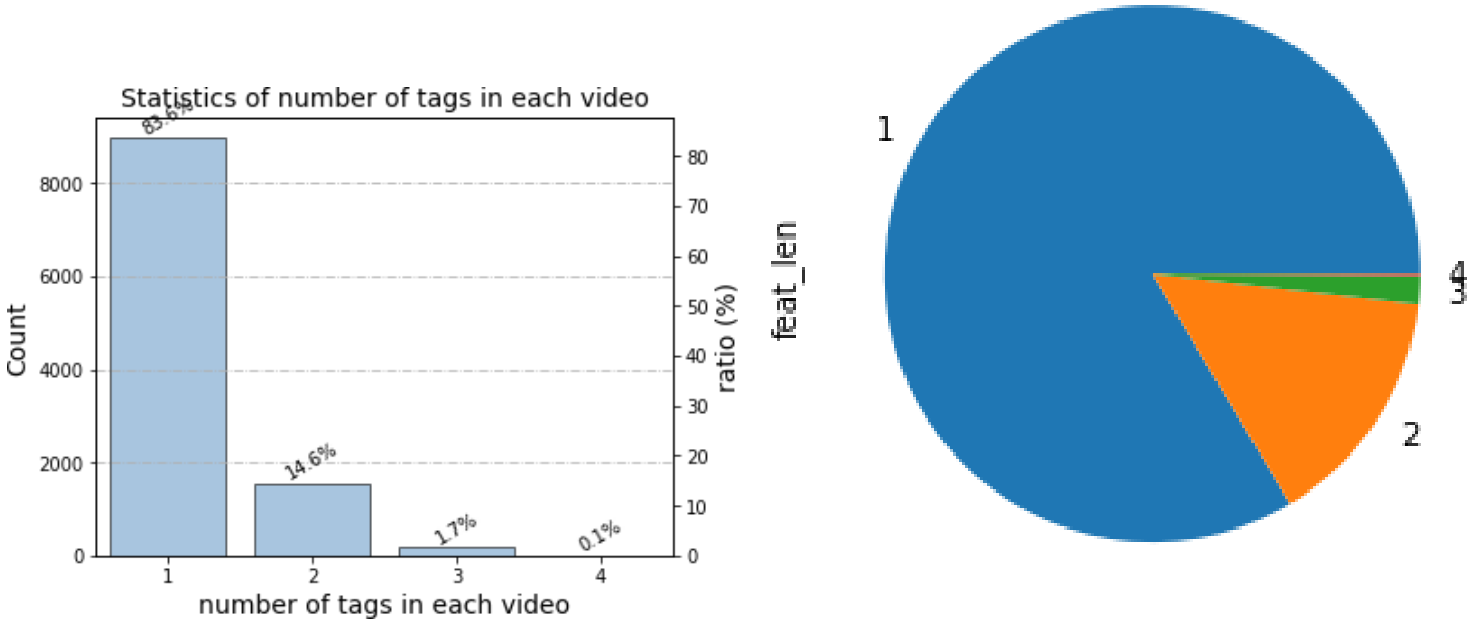

Figure 8: frequency of number of tags in each video

from above result we see that the videos that have only one tags is the most frequent videos. And now we want to see what's the most frequent tag, the which appears in most of the videos.

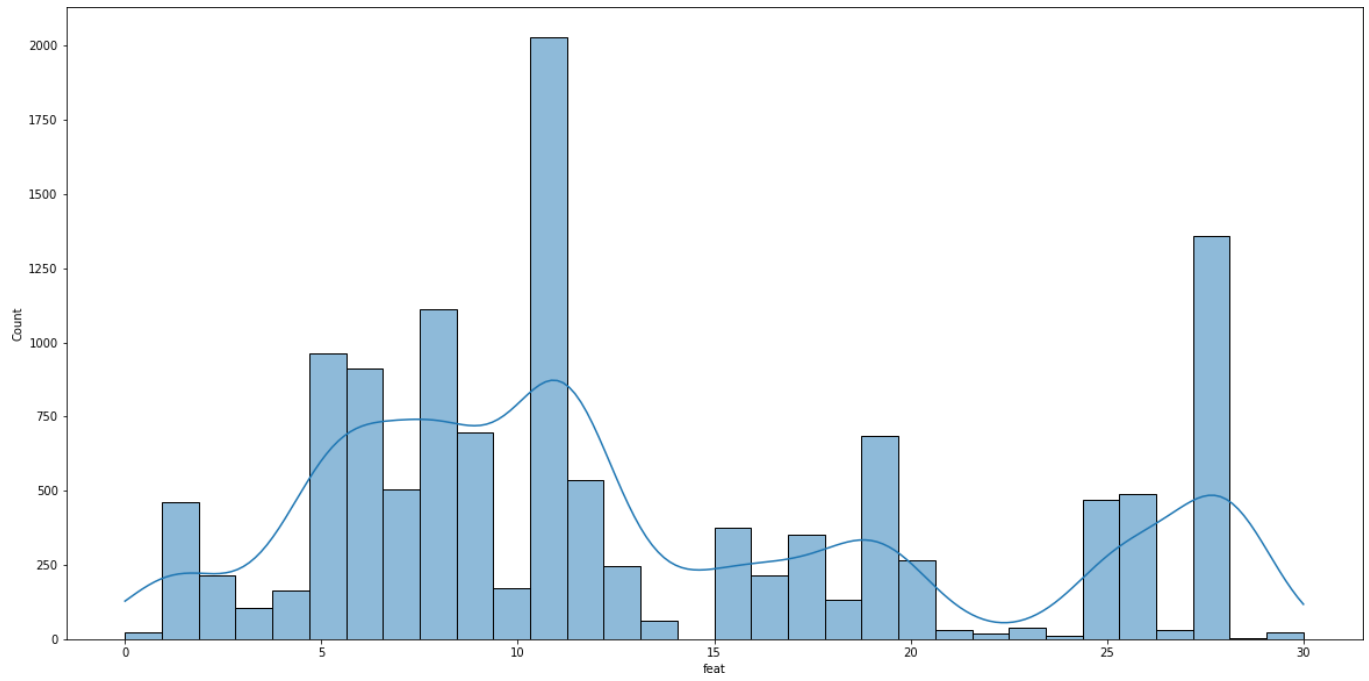

Figure 9: most frequented tags

from the above plot we see that the most frequent tag is the tag 11 and it's appeared more than 2000 times.

4 Data Analysis Pipeline

4.1 Plans and Methodology

- The future Methodology

1. **Data pre-processing and checking the quality for the data**

2. **Data pre-processing and checking the quality for the data**
   based on the basic data exploration there are some data quality issues either in the distributions or the format of the data, In this step I'll check for duplicated values and outliers but not missing values since they are fully observed datasets, I can drop consistent duplicated values since there isn't a same user can watch a same video at a same time, but the outliers here based on the distribution of the data and the box plot are very informative about the recommendation systems and data collection process, soin this case resisting the temptation to remove outliers inappropriately can be difficult,second this second is to handle the format of the data like the date feature or the friend_list feature because it's string data not list data.

3. **Do data exploration and analysis**
   We will do some data exploratory like pie and histograms and the distribution of each feature data.

4. **Do some feature engineering methods**
   We'll drop time feature in **big_matrix** and **small_matrix** datasets because we see that the time feature represented in the date feature but time has the time and the date of watching the video so we considered the time as redundant information, also we'll change the format of the date feature and we will transform it one-hot-encoded data, and in the social_network dataset feature I will create a feature that holds the number of friends that's each user has, and in the **item_feat.csv** I'll create a feature that holds the number of tags in each video, I think the **play_duration** feature is the most important feature since the increasing in its value means that the user is enjoying this video and I can recommend him video from the same category, and I think that the number of observations in my datasets are enough but we don't have that much of feature, we could have more features like number of likes on this video or number of comments that would help us a lot and would improve the performance, also we will check if the user watch the same video in different times (duplicated values in **user_id** and **video_id**) then we will aggregate the **play_duration** time of these duplicated records and then we will drop these duplicated records.

5. **Start building and training the model**
   on jupyter notebook and using python 3 programming language with some libraries like {numpy- matplotlib-pandas-seaborn-pyspark-recommenders}, we will apply one of those deep learning recommender algorithms either Alternating Least Squares (ALS) or eXtreme Deep Factorization Machines (xDeepFM), they are considering two of best filtering systems techniques.

6. **start evaluating our model**
   since they are many techniques to evaluate our performance, we'll choose among these offline metrics, based on [1],[2]

   - **A/B testing:**
   - **Non-accuracy-based metrics:**
     These do not compare predictions against ground truth but instead evaluate the following properties of the recommendations

* **Novelty:**
  measures of how novel recommendation items are by calculating their recommendation frequency among users.
* **Diversity:**
  measures of how different items in a set are with respect to each other
* **Serendipity:**
  measures of how surprising recommendations are to a specific user by comparing them to the items that the user has already interacted with
* **Coverage:**
  measures related to the distribution of items recommended by the system.
* **Ranking Metrics:**
  These are used to evaluate how relevant recommendations are for users.
  - Precision:
    this measures the proportion of recommended items that are relevant
  - **Recall:**
    this measures the proportion of relevant items that are recommended
  - **Normalized Discounted Cumulative Gain (NDCG):**
    evaluates how well the predicted items for a user are ranked based on relevance

So, we didn't determine which metrics should we select except for A/B testing since ourdata isn't highly sparse an **A/B testing** should be one of the suitable metrics.

4.2 potential challenges in deploying this analysis in production

1. **Cold start:**

   This issue arises when new users or new items are added to the system; a new item cannot be recommended to users when it is introduced to the recommendation system without any likes or reviews, making it difficult to predict the choice or interest of users, resulting in less accurate recommendations.

   A newly published video, for example, cannot be suggested to the user until it has received some ratings. A new user or item added-based challenge is tough to manage since obtaining a comparable user without knowing past interest or preferences is impossible.

2. **Sparsity:**

   It occurs frequently when most users do not provide likes or reviews for the things they watch, causing the rating model to become highly sparse, perhaps leading to data sparsity issues. This reduces the chances of discovering a group of users with comparable ratings or interests.

3. **Scalability:**

   The scalability of algorithms with real-world datasets under the recommendation system is a major challenge. A large amount of changing data is created by user-item interactions in the form of ratings and reviews, and hence scalability is a major worry for these datasets. Recommendation systems inefficiently interpret findings from huge datasets; certain advancedlarge-scaled algorithms are necessary to address this issue.