Cleaning dataset

[6]:
```
#In the movies dataset,the ID column contains[] so replacing with 0
movies['id'].replace('[]',0,inplace=True)
```

[7]:
```
#Changing the datatype to numeric
movies['id'] = pd.to_numeric(movies['id'])
```

[8]:
```
#Merging movies and credits dataset with the common column 'id'
movies_credits = pd.merge(movies,credits,on='id')
```

[9]:
```
movies_credits.head(2)
```

[9]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	runtime	spo
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	81.0	[{' 'r
1	FALSE	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	104.0	[{' 'r

2 rows × 26 columns

10]:
```
#Displaying first 3 rows of links dataset
links.head(3)
```

10]:

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0

11]:
```
#Finding the datatypes of links
links.dtypes
```

11]:
```
movieId      int64
imdbId       int64
tmdbId     float64
dtype: object
```

12]:
```
#Checking the null values of links dataset
links.isnull().sum()
```

12]:
```
movieId      0
imdbId       0
tmdbId     219
dtype: int64
```

13]:
```
#As on the links dataset 'tmdbId' is replaced with 'id' (because all the id's has been entered on 'tmdbId')
links.rename(columns={'tmdbId':'id'},inplace=True)
```

14]:
```
#Dropping the 'imdb_id' column
```

Customer - Jupyter Notebook | Movie recommender systems (1)

lhost:8888/notebooks/Desktop/Movie%20recommender%20systems%20(1).ipynb#

ter Movie recommender systems (1) (autosaved)

Logo

dit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

13]:
```
#As on the links dataset 'tmdbId' is replaced with 'id' (because all the id's has been entered on 'tmdbId')
links.rename(columns={'tmdbId':'id'},inplace=True)
```

14]:
```
#Dropping the 'imdb_id' column
movies_credits = movies_credits.dropna(axis=0, subset=['imdb_id'])
```

15]:
```
#On 'imdb_id' we can see the values in 'tt' so lets remove it
movies_credits['imdb_id'] = movies_credits['imdb_id'].str.replace('tt''[0-7]','',regex=True)
```

16]:
```
#Changing the datatype
movies_credits['imdb_id'] = pd.to_numeric(movies_credits['imdb_id'])
```

17]:
```
#Rename the columns
links.rename(columns={'imdbId':'imdb_id'},inplace=True)
```

18]:
```
links
```

18]:

	movieId	imdb_id	id
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0
...	...	...	...
45838	176269	6209470	439050.0
45839	176271	2028550	111109.0

search

35°C Smoke

45841	176275	8536	227506.0
45842	176279	6980792	461257.0

45843 rows × 3 columns

19]:
```
links1 = links[['movieId','imdb_id']]
```

20]:
```
#Merging movies_credits and links1 with the common 'imdb_id' columns
movies_credits = pd.merge(movies_credits,links1,on='imdb_id')
```

21]:
```
#Displaying first 2 rows
movies_credits.head(2)
```

21]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	spoken_langua
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	[{'iso_639_1': 'name': 'Engl
1	FALSE	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	[{'iso_639_1': 'name': 'Engl {'i

2 rows × 27 columns

2 rows × 27 columns

In [22]:
```
#Reading the dataset
keywords = pd.read_csv(r"G:\rec_sys\keywords.csv")
```

In [23]:
```
#Displaying first 2 rows
keywords.head(2)
```

Out[23]:

	id	keywords
0	862	[{'id': 931, 'name': 'jealousy'}, {'id': 4290,...
1	8844	[{'id': 10090, 'name': 'board game'}, {'id': 1...

In [24]:
```
#Checking the datatypes
keywords.dtypes
```

Out[24]:
```
id           int64
keywords    object
dtype: object
```

In [25]:
```
#Merging movies_credits and keywords dataset based on 'id' column
movies_credits = pd.merge(movies_credits,keywords,on='id')
```

In [26]:
```
#Displaying first 2 rows
movies_credits.head(2)
```

Out[26]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	status
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection'	30000000	[{'id': 16, 'name':	http://toystory.disney.com/toy-	862	114709	en	Toy Story	Led by Woody, Andy's		Released

26]:
```
#Displaying first 2 rows
movies_credits.head(2)
```

26]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	status	
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	Released	
1	FALSE	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	Released	d exci

2 rows × 28 columns

27]:
```
#As in Genre columns we found all the unwanted data so we scrapped the genre section which was under 'name' and created a list or
def genre_name(obj):
    L = []
    for i in ast.literal_eval(obj):
        L.append(i['name'])
    return L
```

28]:
```
#Applying the function 'genre_name' on 'genres' column
movies_credits['genres'] = movies_credits['genres'].apply(genre_name)
```

29]:
```
#Displaying first 2 rows
movies_credits.head(2)
```

29]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	status	t
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[Animation, Comedy, Family]	http://toystory.disney.com/toy-story	862	114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	Released	
1	FALSE	NaN	65000000	[Adventure, Fantasy, Family]	NaN	8844	113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	Released	F di u excit

2 rows × 28 columns

30]:
```
#All the cast name was under 'name' section so scrapped the data and stored on list
def cast_name(obj):
    L = []
    counter = 0
    for i in ast.literal_eval(obj):
        if counter != 3:
            L.append(i['name'])
            counter+=1
        else:
            break
    return L
```

31]:
```
#Applying the function 'cast_name' on 'cast' column
movies_credits['cast'] = movies_credits['cast'].apply(cast_name)
```

32]:
```
movies_credits.head(2)
```

32]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	status	t
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[Animation, Comedy, Family]	http://toystory.disney.com/toy-story	862	114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	Released	
1	FALSE	NaN	65000000	[Adventure, Fantasy, Family]	NaN	8844	113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	Released	F di u excit

2 rows × 28 columns

33]:
```
#Director name is scrapped from 'crew' column
def director_name(obj):
    L = []
    for i in ast.literal_eval(obj):
        if i['job'] == 'Director':
            L.append(i['name'])
            break
    return L
```

36]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	status	t
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[Animation, Comedy, Family]	http://toystory.disney.com/toy-story	862	114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	Released	
1	FALSE	NaN	65000000	[Adventure, Fantasy, Family]	NaN	8844	113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	Released	F di u excit

2 rows × 28 columns

37]:

```
#We can see that there are integer values present in the language column.
#Let's take a look at it.
#Dropping the specific rows mentioned below
lang_drop = movies_credits[(movies_credits['original_language'] == '82.0') | (movies_credits['original_language'] == '68.0')| (mc

movies_credits.drop(index = lang_drop, inplace = True)
```

38]:

```
#Checking the null values
movies_credits['original_language'].isnull().sum()
```

38]: 9

39]:

```
movies_credits.dropna(subset = ['original_language'], inplace = True)# dropping the nan values
```

39]:
```
movies_credits.dropna(subset = ['original_language'], inplace = True)# dropping the nan values
```

40]:
```
lang_dec = {'en' : 'English', 'fr' : 'French', 'zh' : 'Chinese', 'it' : 'Italian', 'fa' : 'Farsi',
            'nl' : 'Dutch', 'de' : 'German', 'cn' : 'Chinese','ar' : 'Arabic', 'es' : 'Spanish',
            'ru' : 'Russian', 'sv' : 'Swedish', 'ja' : 'Japanese', 'ko' : 'Korean', 'sr' : 'Serbian',
            'bn' : 'Bengali', 'he' : 'Hebrew', 'pt' : 'Portuguese', 'wo' : 'Wolof', 'ro' : 'Romanian',
            'hu' : 'Hungarian', 'cy' : 'Welsh', 'vi' : 'Vietnamese', 'cs' : 'Czech', 'da' : 'Danish',
            'no' : 'Norwegian', 'nb' : 'Norwegian', 'pl' : 'Polish', 'el' : 'Greek', 'sh' : 'Serbo-Croatian',
            'mk' : 'Macedonian', 'bo' : 'Tibetan', 'ca' : 'Catalan', 'fi' : 'Finnish', 'th' : 'Thai',
            'sk' : 'Slovak', 'bs' : 'Bosnian', 'hi' : 'Hindi', 'tr' : 'Turkish', 'is' : 'Icelandic',
            'ps' : 'Pashto', 'ab' : 'Abkhazian','eo' : 'Esperanto','ka' : 'Georgian', 'mn' : 'Mongolian',
            'bm' : 'Bambara', 'zu' : 'Zulu', 'uk' : 'Ukrainian', 'af' : 'Afrikaans', 'la' : 'Latin',
            'et' : 'Estonian', 'ku' : 'Kurdish', 'fy' : 'Frisian', 'lv' : 'Latvian', 'ta' : 'Tamil',
            'sl' : 'Slovenian', 'tl' : 'Tagalog', 'ur' : 'Urdu', 'rw' : 'Kinyarwanda', 'id' : 'Indonesian',
            'bg' : 'Bulgarian', 'mr' : 'Marathi', 'lt' : 'Lithuanian', 'kk' : 'Kazakh', 'ms' : 'Malay',
            'sq' : 'Albanian', 'qu' : 'Quechua', 'te' : 'Telugu', 'am' : 'Amharic', 'jv' : 'Javanese',
            'tg' : 'Tajik', 'ml' : 'Malayalam','hr' : 'Croatian', 'lo' : 'Laothian', 'ay' : 'Aymara',
            'kn' : 'Kannada', 'eu' : 'Basque', 'ne' : 'Nepali', 'pa' : 'Punjabi', 'ky' : 'Kirghiz',
            'gl' : 'Galician', 'uz' : 'Uzbek', 'sm' : 'Samoan', 'mt' : 'Maltese', 'hy' : 'Armenian',
            'iu' : 'Inuktitut', 'lb' : 'Luxembourgish', 'si' : 'Sinhalese'

}


movies_credits['original_language'] = movies_credits['original_language'].map(lang_dec)
```

41]:
```
movies_credits['release_date'].isnull().sum()#Checking the null values
```

41]:
```
33
```

42]:
```
#Dropping the null values from 'release_date' column
```

Customer - Jupyter Notebook | Movie recommender systems (1)

lhost:8888/notebooks/Desktop/Movie%20recommender%20systems%20(1).ipynb#

ter Movie recommender systems (1) (autosaved)

Logo

dit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

41]:
```
movies_credits['release_date'].isnull().sum()#Checking the null values
```

41]:
```
33
```

42]:
```
#Dropping the null values from 'release_date' column
movies_credits.dropna(subset = ['release_date'], inplace = True)
```

43]:
```
#Applying 'release_date' into datetime format
movies_credits['release_date'] = movies_credits['release_date'].apply(pd.to_datetime)
```

```
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:1137: UserWarning: Parsing '30-10-1995' in DD/MM/YYY
Y format. Provide format or specify infer_datetime_format=True for consistent parsing.
  mapped = lib.map_infer(
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:1137: UserWarning: Parsing '15-12-1995' in DD/MM/YYY
Y format. Provide format or specify infer_datetime_format=True for consistent parsing.
  mapped = lib.map_infer(
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:1137: UserWarning: Parsing '22-12-1995' in DD/MM/YYY
Y format. Provide format or specify infer_datetime_format=True for consistent parsing.
  mapped = lib.map_infer(
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:1137: UserWarning: Parsing '16-11-1995' in DD/MM/YYY
Y format. Provide format or specify infer_datetime_format=True for consistent parsing.
  mapped = lib.map_infer(
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:1137: UserWarning: Parsing '17-11-1995' in DD/MM/YYY
Y format. Provide format or specify infer_datetime_format=True for consistent parsing.
  mapped = lib.map_infer(
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:1137: UserWarning: Parsing '22-11-1995' in DD/MM/YYY
Y format. Provide format or specify infer_datetime_format=True for consistent parsing.
  mapped = lib.map_infer(
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:1137: UserWarning: Parsing '13-12-1995' in DD/MM/YYY
```

44]:
```
#Fetching the year from 'release_year' columns
```

search

35°C Smoke

```
Y format. Provide format or specify infer_datetime_format=True for consistent parsing.
```

44]:
```
#Fetching the year from 'release_year' columns
movies_credits['release_year']  = movies_credits['release_date'].apply(lambda x:x.year)
```

45]:
```
#Displaying the 'release_year' column
movies_credits['release_year']
```

45]:
```
0        1995
1        1995
2        1995
3        1995
4        1995
         ...
31082    2000
31083    1995
31084    1991
31085    2003
31086    1917
Name: release_year, Length: 31045, dtype: int64
```

46]:
```
#Fetching the data,in which the runtime is less than 15
movies_credits[movies_credits['runtime'] <= 15].shape
```

46]:
```
(1469, 29)
```

47]:
```
median_runtime = movies_credits['runtime'].median() # getting the median value of runtime
movies_credits['runtime'] = movies_credits['runtime'].replace(list(range(0, 16)), median_runtime) #replacing runtime less than 10
```

48]:
```
movies_credits['runtime'].isnull().sum()#Checking the null values runtime column
```

49]:
```
movies_credits.dropna(subset = ['runtime'], inplace = True)#Dropping the rows which contains nan values in 'runtime' column
```

50]:
```
movies_credits.isnull().sum()#Checking the null values from movies_credits dataset
```

50]:
```
adult                      0
belongs_to_collection    27606
budget                       0
genres                       0
homepage                 28333
id                           0
imdb_id                      0
original_language           26
original_title               0
overview                   454
popularity                   0
poster_path                204
production_companies         0
production_countries         0
release_date                 0
revenue                      0
runtime                      0
spoken_languages             0
status                      50
tagline                  16193
title                        0
video                        0
vote_average                 0
vote_count                   0
cast                         0
crew                         0
movieId                      0
```

51]:
```
#Applying the function 'genre_name' on 'production_companies' column
movies_credits['production_companies'] = movies_credits['production_companies'].apply(genre_name)
```

92]:
```
#Applying the function 'genre_name' on 'production_countries' column
movies_credits['production_countries'] = movies_credits['production_countries'].apply(genre_name)
```

[1]:
```
from statistics import mode
```

53]:
```
#Filling nan with mode values
movies_credits['status'].fillna(movies_credits['status'].mode()[0],inplace=True)
```

54]:
```
#Displaying
movies_credits['status']
```

54]:
```
0        Released
1        Released
2        Released
3        Released
4        Released
           ...
31082    Released
31083    Released
31084    Released
31085    Released
31086    Released
Name: status, Length: 30902, dtype: object
```

55]:
```
#Filling nan values with mode values
movies_credits['original_language'].fillna(movies_credits['original_language'].mode()[0],inplace=True)
```

Customer - Jupyter Notebook | Movie recommender systems (1)

lhost:8888/notebooks/Desktop/Movie%20recommender%20systems%20(1).ipynb#

ter Movie recommender systems (1) (autosaved)

Logo

dit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

55]:
```
#Filling nan values with mode values
movies_credits['original_language'].fillna(movies_credits['original_language'].mode()[0],inplace=True)
```

56]:
```
#Dispalying
movies_credits['original_language']
```

56]:
```
0        English
1        English
2        English
3        English
4        English
         ...
31082    English
31083    English
31084    English
31085    English
31086    English
Name: original_language, Length: 30902, dtype: object
```

57]:
```
#Reading the dataset
ratings = pd.read_csv(r"G:\rec_sys\ratings_small.csv")
```

58]:
```
#Displaying
ratings.head()
```

58]:

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185

search

35°C Smoke

3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205

59]:
```
#Dropping the timestamp
ratings.drop('timestamp',axis=1,inplace=True)
```

60]:
```
ratings
```

60]:

	userId	movieId	rating
0	1	31	2.5
1	1	1029	3.0
2	1	1061	3.0
3	1	1129	2.0
4	1	1172	4.0
...	...	...	...
99999	671	6268	2.5
100000	671	6269	4.0
100001	671	6365	4.0
100002	671	6385	2.5
100003	671	6565	3.5

100004 rows × 3 columns

61]:
```
ratings = ratings[['movieId','rating']]
```

100004 rows × 3 columns

61]:
```
ratings = ratings[['movieId','rating']]
```

62]:
```
#Applying the groupby operation on 'movieId'
ratings1 = pd.DataFrame(ratings.groupby('movieId')['rating'].mean())
```

71]:
```
#Displaying
movies1 = pd.DataFrame(movies_credits[['title','movieId','release_year']])
```

65]:
```
#Merging the dataset 'movies1' and 'ratins1' with common column 'movieId'
ratings2 = pd.merge(movies1,ratings1,on='movieId')
```

66]:
```
#Displaying
ratings2
```

66]:

	title	movieId	release_year	rating
0	Toy Story	1	1995	3.872470
1	Jumanji	2	1995	3.401869
2	Grumpier Old Men	3	1995	3.161017
3	Waiting to Exhale	4	1995	2.384615
4	Father of the Bride Part II	5	1995	3.267857
...	...	...	...	...
7660	Michael Jackson's Thriller	152173	1983	3.000000
7661	Demons	152844	1971	4.000000
7662	The Video Dead	150462	1987	3.000000

7665 rows × 4 columns

74]:
```
#Replacing the column name
movies_credits['imdbid'] = movies['imdb_id']
```

94]:
```
#Displaying
movies_credits['production_countries'].head()
```

94]:
```
0    [United States of America]
1    [United States of America]
2    [United States of America]
3    [United States of America]
4    [United States of America]
Name: production_countries, dtype: object
```

80]:
```
#Checking the datatype
movies_credits.dtypes
```

80]:
```
adult                      object
belongs_to_collection      object
budget                      int64
genres                     object
homepage                   object
id                          int64
imdb_id                     int64
original_language          object
original_title             object
overview                   object
popularity                float64
poster_path                object
production_companies       object
production_countries       object
```

```
spoken_languages          object
status                    object
tagline                   object
title                     object
video                     object
vote_average             float64
vote_count               float64
cast                      object
crew                      object
movieId                    int64
keywords                  object
release_year               int64
imdbid                    object
dtype: object
```

77]:
```
#Changing the datatype
movies_credits['popularity'] = pd.to_numeric(movies_credits['popularity'])
```

79]:
```
#Changing the datatype
movies_credits['budget'] = pd.to_numeric(movies_credits['budget'])
```

81]:
```
#Checking the null values from  the respective columns
movies_credits.isnull().sum()
```

81]:
```
adult                     0
belongs_to_collection    27606
budget                    0
genres                    0
homepage                  28333
id                        0
imdb_id                   0
original_language         0
```

```
vote_count              0
cast                    0
crew                    0
movieId                 0
keywords                0
release_year            0
imdbid                 12
dtype: int64
```

91]:
```
#Fetching the data by using loc opeartion
movies_credits.loc[movies_credits['original_title']!=movies_credits['title']].head()
```

91]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	title	video	vote_ave
29	FALSE	NaN	18000000	[Fantasy, Science Fiction, Adventure]	NaN	902	112682	French	La Cité des Enfants Perdus	A scientist in a surrealist society kidnaps ch...	...	The City of Lost Children	False	
30	FALSE	NaN	0	[Drama, Crime]	NaN	37557	115012	Chinese	摇啊摇，摇到外婆桥	A provincial boy related to a Shanghai crime f...	...	Shanghai Triad	False	
33	FALSE	NaN	0	[Romance, Adventure]	NaN	78802	114952	French	Guillaumet, les ailes du courage	NaN	...	Wings of Courage	False	

Customer - Jupyter Notebook | Movie recommender systems (1)

lhost:8888/notebooks/Desktop/Movie%20recommender%20systems%20(1).ipynb#

ter Movie recommender systems (1) (autosaved)

Logo

dit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

95]:
```
#Dropping the unwanted columns
movies_credits.drop(['belongs_to_collection','homepage','adult','video','original_title','status'],axis=1,inplace=True)
```

96]:
```
##flatten list,join all tags overview etc
```

98]:
```
#Replacement
movies_credits['genres'] = movies_credits['genres'].apply(lambda x:[i.replace(' ','') for i in x])
movies_credits['keywords'] = movies_credits['keywords'].apply(lambda x:[i.replace(' ','') for i in x])
movies_credits['cast'] = movies_credits['cast'].apply(lambda x:[i.replace(' ','') for i in x])
movies_credits['crew'] = movies_credits['crew'].apply(lambda x:[i.replace(' ','') for i in x])
```

07]:
```
#Changing the datatype
movies_credits['overview'] = movies_credits['overview'].astype(str)
```

08]:
```
movies_credits['overview'] = movies_credits['overview'].apply(lambda x:x.split())
```

22]:
```
#Creating a feature column in which it carries multiple column('overview','genre','keywords','production_companies','production_c
movies_credits['features'] = movies_credits['overview']+movies_credits['genres']+movies_credits['keywords']+movies_credits['produ
```

18]:
```
movies_credits.columns
```

18]:
```
Index(['budget', 'genres', 'id', 'imdb_id', 'original_language', 'overview',
       'popularity', 'poster_path', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime', 'tagline',
       'title', 'vote_average', 'vote_count', 'cast', 'crew', 'movieId',
       'keywords', 'release_year', 'imdbid'],
      dtype='object')
```

search

35°C Smoke

```
       'keywords', 'release_year', 'imdbid'],
      dtype='object')
```

17]:
```
#Dropping the 'spoken_language' column
movies_credits.drop(columns='spoken_languages',axis=1,inplace=True)
```

Replacement and joining opeartion

48]:
```
movies_credits['features'].apply(lambda x:[i.replace(',','') for i in x ])
```

48]:
```
0        [Led, by, Woody, Andy's, toys, live, happily, ...
1        [When, siblings, Judy, and, Peter, discover, a...
2        [A, family, wedding, reignites, the, ancient, ...
3        [Cheated, on, mistreated, and, stepped, on, th...
4        [Just, when, George, Banks, has, recovered, fr...
                              ...
31082    [A, film, archivist, revisits, the, story, of,...
31083    [It's, the, year, 3000, AD., The, world's, mos...
31084    [Yet, another, version, of, the, classic, epic...
31085    [When, one, of, her, hits, goes, wrong, a, pro...
31086    [In, a, small, town, live, two, brothers, one,...
Name: features, Length: 30902, dtype: object
```

53]:
```
movies_credits['features'] = movies_credits['features'].apply(lambda x:' '.join(x))
```

55]:
```
 movies_credits['keywords'].apply(lambda x:' '.join(x))
```

55]:
```
0        jealousy toy boy friendship friends rivalry bo...
1        boardgame disappearance basedonchildren'sbook ...
2           fishing bestfriend duringcreditsstinger oldmen
```

48]:
```
movies_credits['features'].apply(lambda x:[i.replace(',','') for i in x ])
```

48]:
```
0        [Led, by, Woody, Andy's, toys, live, happily, ...
1        [When, siblings, Judy, and, Peter, discover, a...
2        [A, family, wedding, reignites, the, ancient, ...
3        [Cheated, on, mistreated, and, stepped, on, th...
4        [Just, when, George, Banks, has, recovered, fr...
                              ...
31082    [A, film, archivist, revisits, the, story, of,...
31083    [It's, the, year, 3000, AD., The, world's, mos...
31084    [Yet, another, version, of, the, classic, epic...
31085    [When, one, of, her, hits, goes, wrong, a, pro...
31086    [In, a, small, town, live, two, brothers, one,...
Name: features, Length: 30902, dtype: object
```

53]:
```
movies_credits['features'] = movies_credits['features'].apply(lambda x:' '.join(x))
```

55]:
```
 movies_credits['keywords'].apply(lambda x:' '.join(x))
```

55]:
```
0        jealousy toy boy friendship friends rivalry bo...
1        boardgame disappearance basedonchildren'sbook ...
2           fishing bestfriend duringcreditsstinger oldmen
3        basedonnovel interracialrelationship singlemot...
4        baby midlifecrisis confidence aging daughter m...
                              ...
31082     witch mythology legend serialkiller mockumentary
31083
31084
31085
31086
Name: keywords, Length: 30902, dtype: object
```

57]:
```
movies_credits['cast'] = movies_credits['cast'].apply(lambda x:' '.join(x))
movies_credits['crew'] = movies_credits['crew'].apply(lambda x:' '.join(x))
movies_credits['genres'] = movies_credits['genres'].apply(lambda x:' '.join(x))
movies_credits['overview'] = movies_credits['overview'].apply(lambda x:' '.join(x))
movies_credits['production_companies'] = movies_credits['production_companies'].apply(lambda x:' '.join(x))
movies_credits['production_countries'] = movies_credits['production_countries'].apply(lambda x:' '.join(x))
movies_credits['keywords'] = movies_credits['keywords'].apply(lambda x:' '.join(x))
```

70]:
```
movies_credits.head(1)
```

70]:

	budget	genres	id	imdb_id	original_language	overview	popularity	poster_path	production_companies	production_countries
0	30000000	Animation Comedy Family	862	114709	English	Led by Woody, Andy's toys live happily in his ...	21.946943	/rhIRbceoE9lR4veEXuwCC2wARtG.jpg	Pixar Animation Studios	United States of America

1 rows × 24 columns

71]:
```
movies_credits['features'] = movies_credits['features'].apply(lambda x:x.lower())
```

75]:
```
#Reading the dataset
movies_credits.to_csv(r"G:\rec_sys\movies_credits.csv")
```

76]:
```
#Saving the dataset in csv format
ratings2.to_csv(r"G:\rec_sys\ratings2.csv")
```

77]:
```
movies_credits.isnull().sum()
```

71]:
```
movies_credits['features'] = movies_credits['features'].apply(lambda x:x.lower())
```

75]:
```
#Reading the dataset
movies_credits.to_csv(r"G:\rec_sys\movies_credits.csv")
```

76]:
```
#Saving the dataset in csv format
ratings2.to_csv(r"G:\rec_sys\ratings2.csv")
```

77]:
```
movies_credits.isnull().sum()
```

77]:
```
budget                    0
genres                    0
id                        0
imdb_id                   0
original_language         0
overview                  0
popularity                0
poster_path             204
production_companies      0
production_countries      0
release_date              0
revenue                   0
runtime                   0
tagline               16193
title                     0
vote_average              0
vote_count                0
cast                      0
crew                      0
movieId                   0
keywords                  0
```

Customer - Jupyter Notebook | Movie recommender systems (1)

lhost:8888/notebooks/Desktop/Movie%20recommender%20systems%20(1).ipynb#

ter Movie recommender systems (1) (autosaved)

Logo

dit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run | Markdown

```
features                    0
dtype: int64
```

06]:
```
df = pd.read_csv(r"G:\rec_sys\movies_credits.csv")
```

87]:
```
df
```

87]:

	Unnamed: 0	budget	genres	id	imdb_id	original_language	overview	popularity	poster_path	production_compa
0	0	30000000	Animation Comedy Family	862	114709	English	Led by Woody, Andy's toys live happily in his ...	21.946943	/rhIRbceoE9lR4veEXuwCC2wARtG.jpg	Pixar Animation St
1	1	65000000	Adventure Fantasy Family	8844	113497	English	When siblings Judy and Peter discover an encha...	17.015539	/vzmL6fP7aPKNKPRTFnZmiUfciyV.jpg	TriStar Pictures T Film Inters Com
2	2	0	Romance Comedy	15602	113228	English	A family wedding reignites the ancient feud be...	11.712900	/6ksm1sjKMFLbO7UY2i6G1ju9SML.jpg	Warner Bros. Lanc
3	3	16000000	Comedy Drama Romance	31357	114885	English	Cheated on, mistreated and stepped on, the wom...	3.859495	/16XOMpEaLWkrcPqSQqhTmeJuqQl.jpg	Twentieth Centur Film Corpor
4	4	0	Comedy	11862	113041	English	Just when George Banks has recovered	8.387519	/e64sOI48hQXyru7naBFyssKFxVd.jpg	Sandollar Produc Touchstone Pic

search

35°C Smoke

Customer - Jupyter Notebook | Movie recommender systems (1)

lhost:8888/notebooks/Desktop/Movie%20recommender%20systems%20(1).ipynb#

ter Movie recommender systems (1) (autosaved)

dit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run | Markdown

86]: movies_credits

86]:

	budget	genres	id	imdb_id	original_language	overview	popularity	poster_path	production_companies	produ
0	30000000	Animation Comedy Family	862	114709	English	Led by Woody, Andy's toys live happily in his ...	21.946943	/rhIRbceoE9lR4veEXuwCC2wARtG.jpg	Pixar Animation Studios	
1	65000000	Adventure Fantasy Family	8844	113497	English	When siblings Judy and Peter discover an encha...	17.015539	/vzmL6fP7aPKNKPRTFnZmiUfciyV.jpg	TriStar Pictures Teitler Film Interscope Commu...	
2	0	Romance Comedy	15602	113228	English	A family wedding reignites the ancient feud be...	11.712900	/6ksm1sjKMFLbO7UY2i6G1ju9SML.jpg	Warner Bros. Lancaster Gate	
3	16000000	Comedy Drama Romance	31357	114885	English	Cheated on, mistreated and stepped on, the wom...	3.859495	/16XOMpEaLWkrcPqSQqhTmeJuqQl.jpg	Twentieth Century Fox Film Corporation	
4	0	Comedy	11862	113041	English	Just when George Banks has recovered from his ...	8.387519	/e64sOI48hQXyru7naBFyssKFxVd.jpg	Sandollar Productions Touchstone Pictures	
...	...	...	...	...	...	...	...	...	...	...
						A film archivist			Neptune Salad	

search 35°C Smoke

Customer - Jupyter Notebook | Movie recommender systems (1)

lhost:8888/notebooks/Desktop/Movie%20recommender%20systems%20(1).ipynb#

ter Movie recommender systems (1) (autosaved)

Logo

dit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

10]:
```
df.dropna(subset='features',inplace=True)#drop the nan values
```

11]:
```
df['features'].isnull().sum()
```

11]:
```
0
```

17]:
```
df[df['production_countries'].isnull()]
```

84	84	0	NaN	188588	113612	English	Filmed entirely on location in East Hampton, L...	0.531159	/pfgpkDNcwSi1x4jVzeLqvxTwX5a.jpg
107	108	0	Documentary	89333	112646	English	A documentary following Christy Turlington and...	0.976707	/yhFcbTCnsWjg3nH3PLL6RoltjqS.jpg
108	109	0	Crime	96357	113276	English	An ex-con holds a group of people hostage in a...	0.001346	/tQ6HEWNxvbeF2WklTVEp3su446F.jpg
127	128	0	NaN	290157	110217	English	Michel Negroponte, a documentary filmmaker, me...	0.001178	/uUi23HjvDFYGfuVlCBGozUY1Ab4.jpg
128	129	0	Comedy Romance	110972	114131	English	Pie in the Sky is a 1996 American romantic com...	0.699066	/6KO5jsPOr80J8PkKXPE8fym5I2R.jpg
138	139	0	NaN	124639	114618	English	A subtle yet violent commentary on feudal lords	0.001205	/z0ezqAFMeGYd5mLEWaN8jC9eczF.jpg

16]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

search

35°C Smoke

Customer - Jupyter Notebook | Movie recommender systems (1)

lhost:8888/notebooks/Desktop/Movie%20recommender%20systems%20(1).ipynb#

ter Movie recommender systems (1) (autosaved)

Logo

dit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

16]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

[ ]:
```
#Filling the nan with mode values from 'production_countries' and 'production_companies' columns
```

26]:
```
df['production_countries'].replace(np.NaN,df['production_countries'].mode()[0],inplace=True)
```

41]:
```
df.dropna(subset='poster_path',inplace=True)
```

[ ]:
```
df.dropna(subset='poster_path',inplace=True)
```

43]:
```
df['production_companies'].replace(np.NaN,df['production_companies'].mode()[0],inplace=True)
```

45]:
```
df.to_csv(r"G:\rec_sys\movies_credits1.csv")
```

46]:
```
df1 = pd.read_csv(r"G:\rec_sys\ratings2.csv")
```

47]:
```
df1.isnull().sum()
```

47]:
```
Unnamed: 0      0
title           0
movieId         0
release_year    0
rating          0
dtype: int64
```

48]:
```
df1.dtypes
```

search

35°C Smoke

```
movieId          0
release_year     0
rating           0
dtype: int64
```

48]:
```
df1.dtypes
```

48]:
```
Unnamed: 0        int64
title            object
movieId           int64
release_year      int64
rating          float64
dtype: object
```

52]:
```
df.dtypes
```

52]:
```
Unnamed: 0                    int64
budget                        int64
genres                       object
id                            int64
imdb_id                       int64
original_language            object
overview                     object
popularity                  float64
poster_path                  object
production_companies         object
production_countries         object
release_date          datetime64[ns]
revenue                     float64
runtime                     float64
tagline                      object
title                        object
vote_average                float64
```

```
runtime                 float64
tagline                  object
title                    object
vote_average            float64
vote_count              float64
cast                     object
crew                     object
movieId                   int64
keywords                 object
release_year              int64
imdbid                   object
features                 object
dtype: object
```

51]:
```
df['release_date'] = pd.to_datetime(df['release_date'])
```

Saved the cleaned dataset in csv format

55]:
```
df.to_csv(r"G:\rec_sys\movies.csv")
```

[ ]:

Movie Recommender system

Importing Libraries

[1]:
```
import pandas as pd
import numpy as np
import ast
```

Reading the dataset

73]:
```
movies = pd.read_csv(r"G:\rec_sys\movies_metadata.csv")
credits = pd.read_csv(r"G:\rec_sys\credits.csv")
links = pd.read_csv(r"G:\rec_sys\links.csv")
```

```
C:\Users\lokit\AppData\Local\Temp\ipykernel_26964\798319844.py:1: DtypeWarning: Columns (10) have mixed types. Specify dtype option on import or set low_memory=False.
  movies = pd.read_csv(r"G:\rec_sys\movies_metadata.csv")
```

[5]:
```
#Checking the datatype of links dataset
links.dtypes
```

[5]:
```
movieId      int64
imdbId       int64
tmdbId     float64
dtype: object
```