AIWR ASSIGNMENT -2

Team Members:

V NIMISH BHASU PES2UG20CS479
K VASANTH KARTHIK PES2UG20CS515

SURAJ P BHADANIA PES2UG20CS470

ANUP D SIRSIKAR PES2UG20CS494

CORPUS DETAILS:

Used Dataset: **anime.csv, rating.csv**

Dataset Description:

The datasets used in this project consist of a list of animes and their associated ratings, as well as information about their respective genres and types.

Importing all the packages

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from collections import defaultdict
from scipy.sparse import csr_matrix
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
import re
from surprise import SVD,Dataset,Reader
```

Preprocessing and EDA

The code reads in two CSV files and checks for null values. It drops null values in the "anime" DataFrame and replaces -1 with NaN in the "rating" DataFrame. The two DataFrames are merged on "anime_id" to create "df", where "rating_user" is renamed to "user_rating". The head(10) function displays the first 10 rows of "df".

Preprocessing data and EDA

```python
anime=pd.read_csv('./anime.csv')
rating=pd.read_csv('./rating.csv')
```

```python
anime
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266
...	...	...	...	...	...	...	...
12289	9316	Toushindai My Lover: Minami tai Mecha-Minami	Hentai	OVA	1	4.15	211
12290	5543	Under World	Hentai	OVA	1	4.28	183
12291	5621	Violence Gekiga David no Hoshi	Hentai	OVA	4	4.88	219
12292	6133	Violence Gekiga Shin David no Hoshi: Inma Dens...	Hentai	OVA	1	4.98	175
12293	26081	Yasuji no Pornorama: Yacchimae!!	Hentai	Movie	1	5.46	142

12294 rows × 7 columns

```python
rating
```

	user_id	anime_id	rating
0	1	20	-1
1	1	24	-1
2	1	79	-1
3	1	226	-1
4	1	241	-1
...	...	...	...
7813732	73515	16512	7
7813733	73515	17187	9
7813734	73515	22145	10
7813735	73516	790	9
7813736	73516	8074	9

7813737 rows × 3 columns

```python
print(anime.isnull().sum())
print("----------------------")
print(rating.isnull().sum())
```

```
anime_id      0
name          0
genre        62
type         25
episodes      0
rating      230
members       0
dtype: int64
----------------------
user_id    0
anime_id   0
rating     0
dtype: int64
```

```python
#droping the null values in anime df
#replacing ratings which are -1 to nan in rating column in rating df
animeanime.dropna()
rating.rating.replace(-1, np.NaN,inplace=True)
```

```python
for text in anime['name']:
    text = re.sub(r'&quot;', '', text)
    text = re.sub(r'.hack//', '', text)
    text = re.sub(r'&#039;', '', text)
    text = re.sub(r'A&#039;', '', text)
    text = re.sub(r'I&#039;', 'I\'', text)
    text = re.sub(r'&amp;', 'and', text)
```

```python
#merging anime datafram and ratign df
dfcpi.merge(anime,rating,on='anime_id',suffixes= ['', '_user'])
df = df.rename(columns={'rating_user': 'user_rating'})
df.head(10)
```

	anime_id	name	genre	type	episodes	rating	members	user_id	user_rating
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	99	5.0
1	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	152	10.0
2	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	244	10.0
3	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	271	10.0
4	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	278	NaN
5	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	322	10.0
6	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	398	10.0
7	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	482	8.0
8	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	490	10.0
9	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	548	10.0

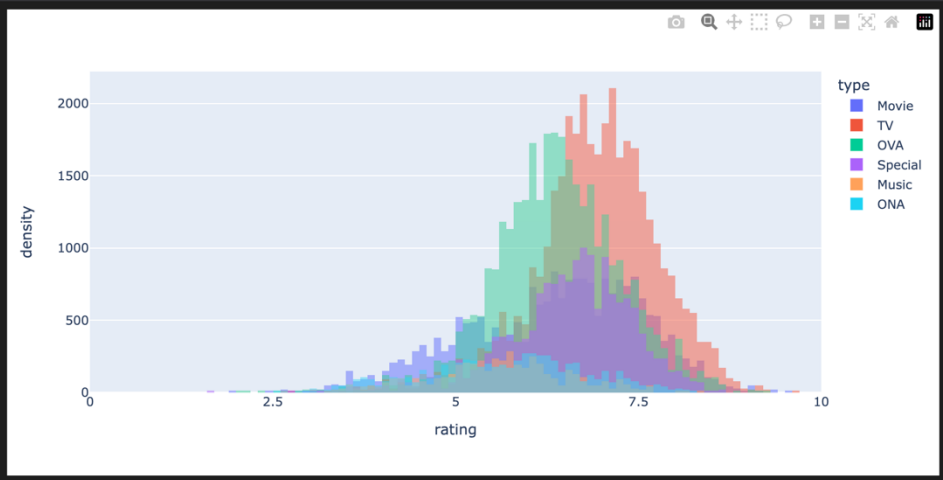
Variations of ratings

The code creates a histogram plot of the anime ratings data using Plotly Express library. The plot shows the distribution of anime ratings by type. The dropna() function is used to remove any null values from the "anime" DataFrame. The update_layout() function is used to customize the layout of the plot. The resulting plot provides a visual representation of the distribution of anime ratings, which can be helpful in analyzing and understanding the data.

variations of ratings by anime type

```
fig = px.histogram(anime.dropna(subset=['rating']), x='rating', color='type',
                   histnorm='density', nbins=100, opacity=0.5,
                   barmode='overlay', range_x=[0, 10])
fig.update_layout(xaxis=dict(dtick=2.5))


fig.show()
```



How many animes do people rate

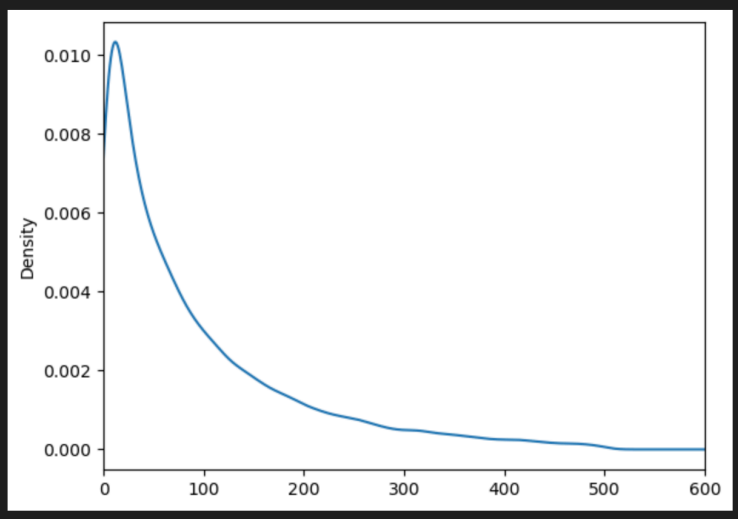
This code creates a density plot to show how many anime ratings each user submitted. It groups the anime ratings data by user ID and counts the number of ratings submitted by each user. Then it filters the data to only include users who submitted less than 500 ratings. The density plot is created using Matplotlib library and shows the distribution of rating submissions by user. The x-axis limits are set to a range of 0 to 600 to show the range of number of rating submissions.

How many animes does people rate

```
g = rating.groupby('user_id').size().reset_index(name='n')
g = g[g['n'] < 500]

# Density plot using matplotlib
fig, ax = plt.subplots()
ax = g['n'].plot.density()
ax.set_xlim(0, 600)
```

```
(0.0, 600.0)
```



- we can see that not many users rate the anime

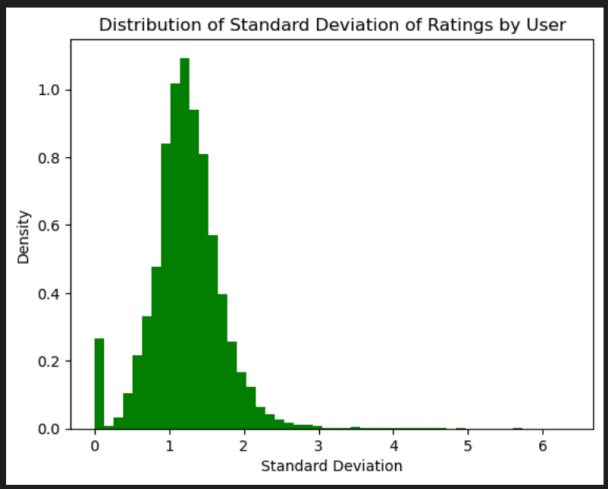
Consistency

This code creates a histogram to show the distribution of the standard deviation of anime ratings submitted by each user. It groups the anime ratings data by user ID and calculates the standard deviation of the ratings submitted by each user. The resulting DataFrame is used to create a histogram plot using Matplotlib library. The x-axis represents the standard deviation of ratings, and the y-axis represents the density of the data. The plot provides a visual representation of the distribution of standard deviation of ratings by user, which can be helpful in analyzing the consistency of user ratings.

How consistently does people rate

```
g = rating.groupby('user_id')['rating'].std().reset_index(name='sd')
# fig = px.histogram(g, x='sd', nbins=50)
# fig.show()
plt.hist(g['sd'], bins=50, density=True,color="green")
plt.xlabel('Standard Deviation')
plt.ylabel('Density')
plt.title('Distribution of Standard Deviation of Ratings by User')

# Display the plot
plt.show()
```



- Some people have only a single rating (likely the spike at 0)
- A lot of people shuffle their ratings 1-2 points.
- Some people have huge variance.

Animes based on rating

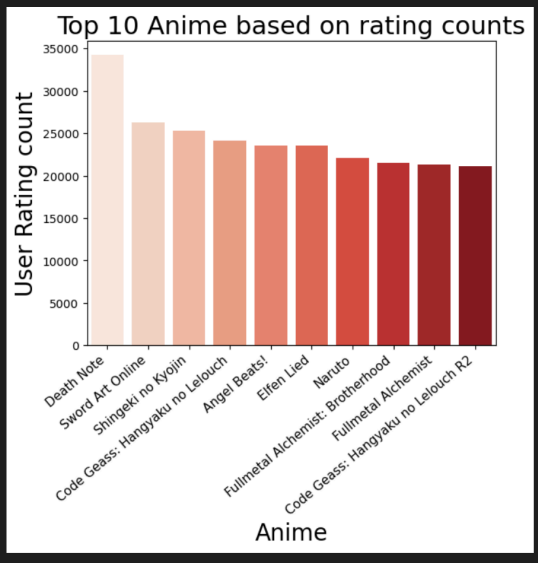
This code creates a horizontal bar plot using Seaborn library to show the top 10 animes based on the number of user ratings. It drops rows with missing values in the 'name' column, groups the data by anime name, and calculates the number of user ratings for each anime. The resulting DataFrame is then sorted in descending order based on the number of user ratings, and the top 10 animes are selected. The horizontal bar plot is created using Seaborn's barplot function and shows the anime names on the y-axis and the number of user ratings on the x-axis. The plot includes labels for the x-axis, y-axis, and the plot title, all with custom font sizes.

Top animes based on rating

```
combine_anime_rating = df.dropna(axis = 0, subset = ['name'])
anime_ratingCount = (combine_anime_rating.
    groupby(by = ['name'])['user_rating'].
    count().
    reset_index().rename(columns = {'rating': 'totalRatingCount'})
    [['name', 'user_rating']]
    )


top10_animerating=anime_ratingCount[['name', 'user_rating']].sort_values(by = 'user_rating',ascending = False).head(10)
ax=sns.barplot(x="name", y="user_rating", data=top10_animerating, palette="Reds")
ax.set_xticklabels(ax.get_xticklabels(), fontsize=11, rotation=40, ha="right")
ax.set_title('Top 10 Anime based on rating counts',fontsize = 22)
ax.set_xlabel('Anime',fontsize = 20)
ax.set_ylabel('User Rating count', fontsize = 20)
```

```
Text(0, 0.5, 'User Rating count')
```



Popularity of Genres

This code counts the frequency of anime genres in the dataset and creates a bar chart using Matplotlib. It first creates an empty dictionary called all_genres using the defaultdict method from the collections module. It then loops through each row in the 'genre' column of the DataFrame, splits the genres by comma, and counts the frequency of each genre. The resulting dictionary is sorted by the frequency of each genre in ascending order, and a bar plot is created using Matplotlib's bar function. The plot includes labels for the x-axis, y-axis, and the plot title, and the x-axis labels are rotated by 60 degrees for better visibility.

Popularity of genres

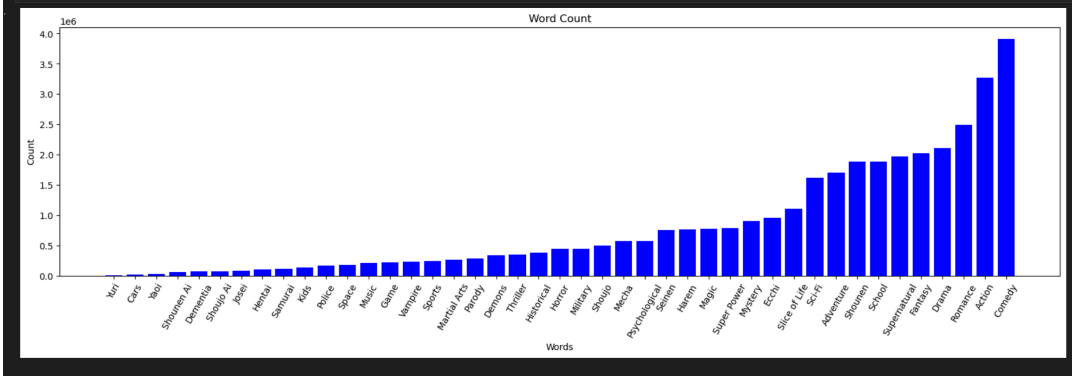
```python
all_genres = all_genres = defaultdict(int)
for i in df['genre']:
    for j in i.split(','):
        all_genres[j.strip()] += 1
all_genres=dict(all_genres)
print(all_genres)
print(type(all_genres))
all_genres = dict(sorted(all_genres.items(), key=lambda x: x[1]))
```

Python

```
{'Drama': 2101320, 'Romance': 2487902, 'School': 1884897, 'Supernatural': 1964112, 'Action': 3263393, 'Adventure': 1695965, 'Fantasy': 2021336, 'Magic': 769540, 'Military': 440336, 'Shounen': 1882227, 'Comedy': 3908785, 'Historical': 375688, 'Parody': 27
<class 'dict'>
```

```python
keys = list(all_genres.keys())
values = list(all_genres.values())
plt.figure(figsize=(20, 5))
plt.bar(keys, values, color='blue')
plt.title("Word Count")
plt.xlabel("Words")
plt.ylabel("Count")
plt.xticks(rotation=60)
plt.show()
```

Python



Collaborative Filtering

This code is preparing a dataset for collaborative filtering and using it to recommend anime shows similar to 'Death Note'. It drops rows with missing values and limits the dataset to user_id values of 20000 or less. Then it creates a pivot table and normalizes it. The pivot table is then converted into a sparse matrix, which is used to calculate item and user similarity matrices. Finally, a function is defined that recommends top-n similar anime shows to a given anime, 'Death Note' in this case.

Collaborative Filtering

Collaborative filtering can be done using creating a pivot table, since the data is huge it takes lot of time to create the pi

```python
df_collab=df[['user_id', 'name', 'user_rating']]
df_collab=df_collab.dropna()
df_collab= df_collab[df_collab.user_id <= 20000]
```

```python
df_collab=df[['user_id', 'name', 'user_rating']]
# df_collab=df_collab.dropna()
df_collab= df_collab[df_collab.user_id <= 20000]
#creating pivot table
pivot_table= df_collab.pivot_table(index=['user_id'], columns=['name'], values='user_rating')

#normalizing
normalised_pivot_table = pivot_table.apply(lambda x: (x-np.mean(x))/(np.max(x)-np.min(x)), axis=1)
# drop all columns containing only zeros representing users who did not rate
normalised_pivot_table.fillna(0, inplace=True)
normalised_pivot_table = normalised_pivot_table.T
normalised_pivot_table = normalised_pivot_table.loc[:, (normalised_pivot_table != 0).any(axis=0)]
#converting pivot table into sparse matrix
sparse_matrix = csr_matrix(normalised_pivot_table.values)
#getting item_similarity anf user similarity
item_similarity = cosine_similarity(sparse_matrix)
user_similarity = cosine_similarity(sparse_matrix.T)

#make item and user similarity matrices into df
item_sim_df = pd.DataFrame(item_similarity, index = normalised_pivot_table.index, columns = normalised_pivot_table.index)
user_sim_df = pd.DataFrame(user_similarity, index = normalised_pivot_table.columns, columns = normalised_pivot_table.columns)
```

```python
def collaborative_filtering_recommendation(anime_name,top_n):
    count = 1
    print('Similar shows to {} include:\n'.format(anime_name))
    for item in item_sim_df.sort_values(by = anime_name, ascending = False).index[1:top_n+1]:
        print('No. {}: {}'.format(count, item))
        count +=1
```

```python
collaborative_filtering_recommendation('Death Note',10)
```

```
Similar shows to Death Note include:

No. 1: Code Geass: Hangyaku no Lelouch R2
No. 2: Code Geass: Hangyaku no Lelouch
No. 3: Fullmetal Alchemist: Brotherhood
No. 4: Steins;Gate
No. 5: Shingeki no Kyojin
No. 6: Clannad: After Story
No. 7: One Punch Man
No. 8: Kiseijuu: Sei no Kakuritsu
No. 9: Hunter x Hunter (2011)
No. 10: Fate/Zero 2nd Season
```

Content Based Recommendation

The code is performing content-based recommendation using TF-IDF and cosine similarity. It is first processing the anime data to extract features such as genre, type, and episodes. Then, it applies TF-IDF to these features to create a matrix of feature vectors. Using cosine similarity, it generates similarity scores between each anime based on their feature vectors. Finally, it defines a function that takes an anime name and returns a list of top k recommended anime based on their similarity scores with the given anime.

Content Based recomendation

```
anime_features=anime[['genre', 'type', 'episodes']].astype('str')
anime_features["genre"]
anime_features["genre"]=anime_features["genre"].str.split(',').str.join(' ')
anime_features
```

	genre	type	episodes
0	Drama Romance School Supernatural	Movie	1
1	Action Adventure Drama Fantasy Magic Mili...	TV	64
2	Action Comedy Historical Parody Samurai S...	TV	51
3	Sci-Fi Thriller	TV	24
4	Action Comedy Historical Parody Samurai S...	TV	51
...	...	...	...
12289	Hentai	OVA	1
12290	Hentai	OVA	1
12291	Hentai	OVA	4
12292	Hentai	OVA	1
12293	Hentai	Movie	1

12017 rows × 3 columns

```
#creating tf-idf
tf_idf = TfidfVectorizer(stop_words='english')
anime_matrix = tf_idf.fit_transform(anime_features.apply(lambda x: ' '.join(x), axis=1))
cosine_similarities = cosine_similarity(anime_matrix)  #creating cosine similarity
#creating a list which contains all anime names
anime_names=anime["name"]
indices = pd.Series(anime.index, index=anime['name'])
```

```
def get_recommendations(anime_name,k):
    idx = indices[anime_name]
    # Get the pairwise similarity scores
    sim_scores = list(enumerate(cosine_similarities[idx]))
    # Sort the anime based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    # Get the scores of the k most similar anime
    sim_scores = sim_scores[1:k+1]
    # Get the anime indices
    anime_indices = [i[0] for i in sim_scores]
    # Return the top k most similar anime
    return anime_names.iloc[anime_indices]
```

```
get_recommendations("High School DxD New",10)
```

```
1036                          High School DxD BorN
1057                                 High School DxD
3002                     Shinmai Maou no Testament
2687                                 Omamori Himari
1163        High School DxD BorN: Yonigeranai Fushichou
1893                     Saenai Heroine no Sodatekata
3181   Ore ga Ojousama Gakkou ni &quot;Shomin Sample&...
3250                          Hoshizora e Kakaru Hoshi
4289       Oniichan dakedo Ai Sae Areba Kamhainai yo ne!
4535                                              R-15
Name: name, dtype: object
```