**LAB 4**

**RECOMMENDER SYSTEM**

<https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset?select=movie.csv>

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from scipy.sparse import csr\_matrix

from sklearn.neighbors import NearestNeighbors

from google.colab import files

uploaded = files.upload()

**Reading Data**

movies = pd.read\_csv("movie.csv")

tags = pd.read\_csv("tag.csv")

ratings = pd.read\_csv("rating.csv")

links = pd.read\_csv("link.csv")

movies = pd.read\_csv("movie.csv", index\_col="movieId")

print(movies.shape)

movies.head()

**Splitting genres**

movies["genres"] = movies["genres"].apply(lambda x : x.split("|"))

movies["genres"] = movies["genres"].apply(lambda x : ' '.join(word for word in x))

movies.head()

**Histogram for genres**

all\_genres = [genre for all\_genre in movies.genres.to\_numpy() for genre in all\_genre.split(" ")]

plt.figure(figsize=(20, 6))

\_ = sns.histplot(data=all\_genres)

\_ = plt.xticks(rotation=45)

**Pie chart for rating**

all\_rates, counts = np.unique(ratings.rating.to\_numpy(), return\_counts = True)

plt.figure(figsize=(10, 10))

\_=plt.pie(counts, labels=all\_rates, autopct='%1.1f%%', startangle=90)

**CONTENT BASED:**

**Recommend movies based on similar genres**

movies.info()

similarity\_feature = movies["genres"].to\_numpy()

similarity\_feature = np.array([genre.lower() for genre in similarity\_feature])

similarity\_feature[0]

from sklearn.feature\_extraction.text import CountVectorizer

count\_vecotr = CountVectorizer()

count\_vecotr.fit(similarity\_feature)

similarity\_feature = count\_vecotr.transform(similarity\_feature)

**Using Cosine similarity**

from sklearn.metrics.pairwise import cosine\_similarity

movies\_similarity = cosine\_similarity(similarity\_feature)

movies\_similarity = pd.DataFrame(movies\_similarity)

print(movies\_similarity.shape)

movies\_similarity.head()

**Get top10 most similar movies**

def get\_similar\_movies\_content(movie\_name):

  movie\_id = movies[movies.title == f"{movie\_name}"].index[0]

  all\_moveis = movies\_similarity[movie\_id - 1]

  k = 10

  idx = np.argpartition(all\_moveis, -k)

  most\_similar\_movies\_id = idx[-10:].values

  most\_similar\_movies\_id = np.delete(most\_similar\_movies\_id, np.where(most\_similar\_movies\_id == movie\_id - 1)[0])

  for movie\_id in most\_similar\_movies\_id :

    print(movies.iloc[movie\_id].title)

get\_similar\_movies\_content("Toy Story (1995)")

**COLLABORATIVE FILTERING RECOMMENDER SYSTEMS**

print(ratings.shape)

ratings.head()

**No of votes for each user**

user\_vote = pd.DataFrame(ratings.groupby("userId")["rating"].agg("count"))

top\_users\_index = user\_vote.rating.sort\_values(ascending=False)[:20].index

top\_users\_value = user\_vote.rating.sort\_values(ascending=False)[:20].values

plt.figure(figsize=(22, 8))

\_ = sns.barplot(x=top\_users\_index, y=top\_users\_value)

**Describing votes**

user\_vote.describe()

**No of votes for each movie**

def get\_movie\_name(movie\_id):

  return movies.iloc[movie\_id].title

movies\_votes = pd.DataFrame(ratings.groupby("movieId")["rating"].agg("count"))

top\_movies\_index = movies\_votes.rating.sort\_values(ascending=False)[:20].index

top\_movies\_value = movies\_votes.rating.sort\_values(ascending=False)[:20].values

movie\_names = [get\_movie\_name(id) for id in top\_movies\_index]

plt.figure(figsize=(22, 8))

\_ = sns.barplot(x=top\_movies\_index, y=top\_movies\_value)

print(movie\_names)

**Describing movie votes**

movies\_votes.describe()

**Users and movie in one pivot table**

data = pd.pivot(index = 'movieId',columns = 'userId', data = ratings, values ='rating')

print(data.shape)

data.head()

**Data cleaning**

**Getting rid of movies with less than 10 votes**

**Getting rid of users less than 170 votes**

data = data.dropna(thresh=10)

print(data.shape)

data.head()

data = data.dropna(axis = 1, thresh=170)

print(data.shape)

data.head()

**Filling NaN Values**

data = data.fillna(0)

data.head()

**Recommended movies**

from sklearn.neighbors import NearestNeighbors

knn = NearestNeighbors(metric='cosine', algorithm='brute', n\_neighbors=21)

knn.fit(data)

def get\_similar\_movies\_collaborative(movie\_name):

    movie\_id = movies[movies.title == f"{movie\_name}"].index[0]

    print("movie id is : ", movie\_id)

    movie\_vote\_data = data.loc[movie\_id].to\_numpy()

    movie\_vote\_data = movie\_vote\_data.reshape(1, -1)

    distances , indices = knn.kneighbors(movie\_vote\_data

                                     ,n\_neighbors=11)

    names = [get\_movie\_name(id) for id in indices]

    names = names[0].values.reshape(1, 11)

    for val in list(zip(distances[0], names[0])):

      print(val)

get\_similar\_movies\_collaborative("Father of the Bride Part II (1995)")