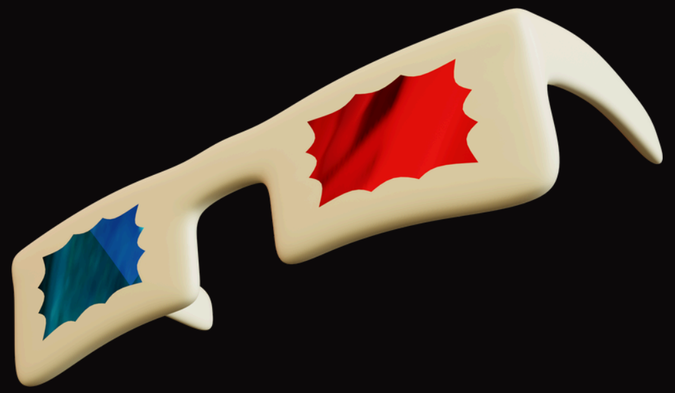
ITMGT 25.03 Final Project

CINEMATE

By Lyanna Kirsten Chee, Matt Giancarlo Molino, Jewelle Dayna Pua



Introducing CineMate

Whether you're a die-hard cinephile or a casual viewer who just wants a good flick after work, CineMate guarantees you'll never get stuck with "what to watch next" again.

The Objectives of CineMate

- Objective
  - To provide accurate and personalized movie and TV show recommendations.
  - To enhance user experience by offering a simple and intuitive interface.
  - To integrate data from multiple streaming platforms and create a comprehensive recommendation system.

The Datasets of CineMate

Amazon Prime

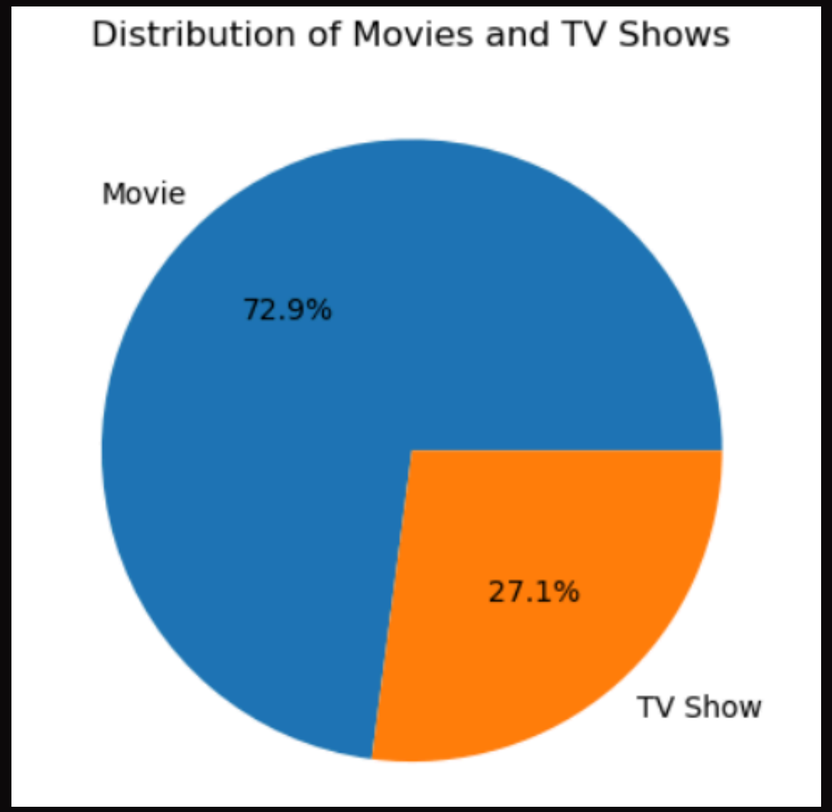
IMDb
Movies

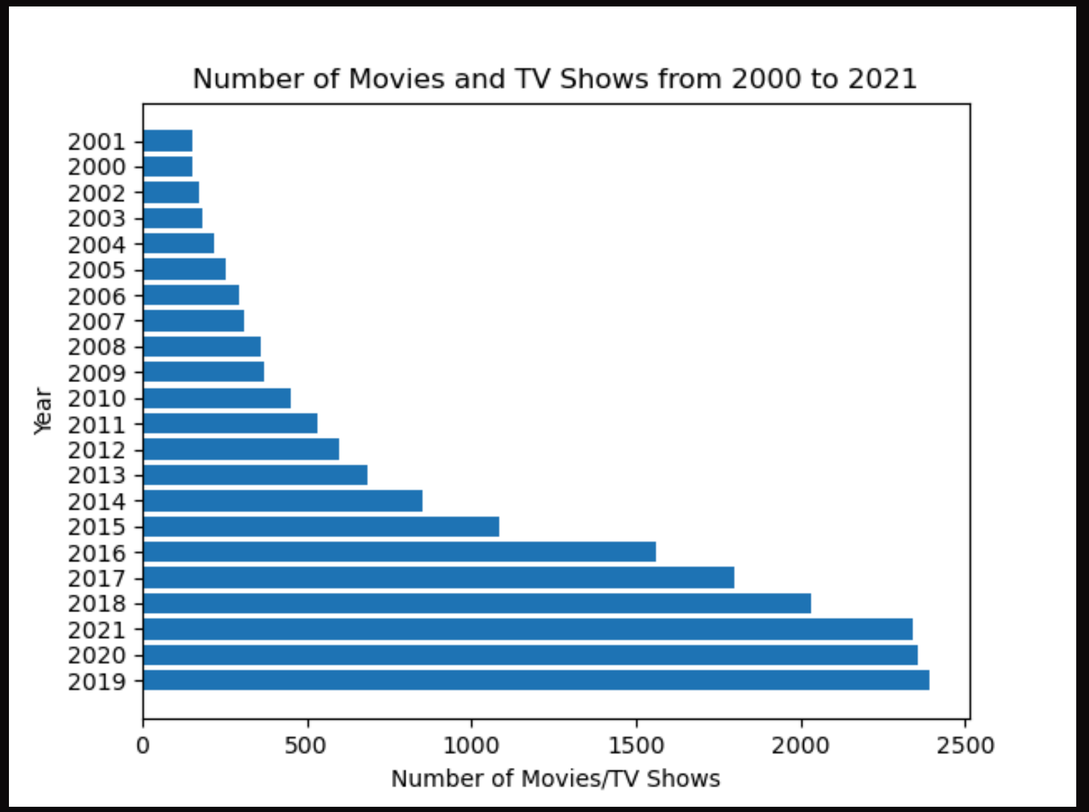
Netflix

Disney+

Hulu

Data Visualization





The Preprocessing of CineMate

```
#Merging Datasets
merge_repeat = 0
while merge_repeat != 2:
# merge all datasets
    merged_df = pd.concat([movie_df, netflix_df, amazon_df, disney_df, hulu_df], ignore_index = True)

# merging similar columns
    movie_df["type"] = "Movie"
    movie_df["title"] = movie_df["Series_Title"]
    movie_df["release_year"] = movie_df["Released_Year"]
    movie_df["description"] = movie_df["Overview"]
    netflix_df["Genre"] = netflix_df["listed_in"]
    amazon_df["Genre"] = amazon_df["listed_in"]
    disney_df["Genre"] = disney_df["listed_in"]
    hulu_df["Genre"] = hulu_df["listed_in"]

    merged_df = merged_df.rename(columns = {"type":"Type","title":"Title","release_year":"Release Year","description":"Description"})

# filtering columns from dataset + replace NaN with blank
    merged_df = merged_df[["Type","Title","Release Year","Genre","Description"]].fillna("")

# removing movie/show duplicates
    merged_df = merged_df.drop_duplicates(subset = ['Title'], keep = 'first')
```

The Types of Recommendations of CineMate

- **Genre-Based Recommendations:** Users can specify a genre and get a list of recommended movies or TV shows.
- **Specific Title Recommendations:** Users can input a specific movie or TV show they enjoyed, and CineMate will provide similar recommendations.

The Genre-Based Recommendations of CineMate

```
def system_option(x):
    exit = False
    if x.upper() == 'GENRE':
        kind = input('\nGreat! So what are you looking for? Recommendations for a movie, TV show, or both? ')
        genre = input('What genre of movie? (Action, Adventure, Comedy, Drama, Horror, Romance, Science Fiction, Fantasy, Historical, Crime) ')
        number = int(input('\nHow many suggestions do you want? (1 to 10) '))

        if kind.upper() == 'MOVIE':
            mov_df = merged_df.loc[(merged_df["Type"].str.upper() == "MOVIE") & (merged_df["Genre"].str.contains(genre, case = False))]
            suggestions = mov_df.sample(n = number) if not mov_df.empty else pd.DataFrame()
        elif kind.upper() == 'TV SHOW':
            tv_df = merged_df.loc[(merged_df["Type"].str.upper() == "TV SHOW") & (merged_df["Genre"].str.contains(genre, case = False))]
            suggestions = tv_df.sample(n = number) if not tv_df.empty else pd.DataFrame()
        elif kind.upper() == 'BOTH':
            both_df = merged_df.loc[merged_df["Genre"].str.contains(genre, case=False)]
            suggestions = both_df.sample(n = number) if not both_df.empty else pd.DataFrame()

        def recommendations():
            if not suggestions.empty:
                print("\nHere are your recommendations:\n")
                center("RECOMMENDATIONS")
                suggestions.index = pd.RangeIndex(start = 1, stop = number + 1)
                pd.set_option("max_colwidth", None)
                return display(suggestions)
            else:
                print("\nSorry, no suggestions found for your criteria.")

        return recommendations()
```

The Specific Title Recommendations of CineMate

```
elif x.upper() == 'SPECIFIC':
    print("In order to successfully find a match, you can check first if your title is present within our data!")
    while exit != True:
        title_check = str(input('\nKindly input your title here first to double check: ')).lower()
        if len(merged_df[merged_df["Title"].str.lower().str.contains(title_check, case=False)]) != 0:
            center("Database Search Results for "+title_check)
            search_df = pd.DataFrame(merged_df[merged_df["Title"].str.lower().str.contains(title_check, case=False)])
            display(search_df)
            exit_confirmation = input('\nWould you like to input another title? (Y/N)')
            if exit_confirmation.upper() == "Y":
                exit_confirmation = "Y"
            elif exit_confirmation.upper() == "N":
                exit_confirmation = "N"
                exit = True
                break
        else:
            print("Your search does not appear in our database! Please try another title!")
            exit_confirmation = input('\nWould you like to input another title? (Y/N)')
            if exit_confirmation.upper() == "Y":
                exit_confirmation = "Y"
            elif exit_confirmation.upper() == "N":
                exit_confirmation = "N"
                exit = True
                break
```

The Specific Title Recommendations of CineMate

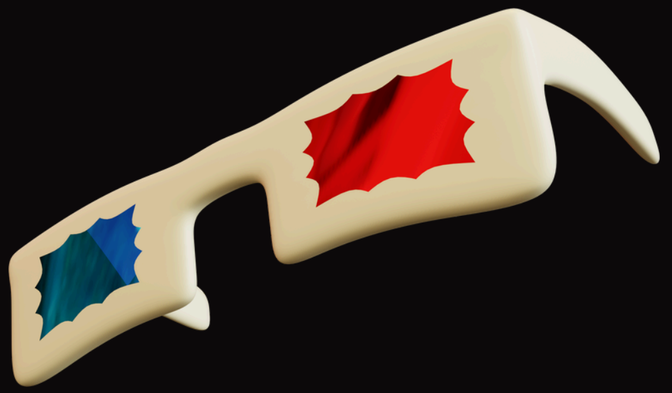
```
input_title = str(input('\nGreat! So what is the movie or TV show that you enjoyed and want to find similar of? ')).lower()
number = int(input('\nHow many suggestions do you want? (1 to 10) '))

dummy_df = merged_df.copy()
dummy_df["Genre"] = dummy_df["Genre"].str.replace(",", "")

def combine_features(row):
    return row["Title"] + " " + row["Genre"]

dummy_df["combined_features"] = dummy_df.apply(combine_features, axis=1)

cv = CountVectorizer()
count_matrix = cv.fit_transform(dummy_df["combined_features"])
cosine_sim = cosine_similarity(count_matrix)

indices = pd.Series(dummy_df.index, index = dummy_df["Title"].str.lower()).drop_duplicates()

def get_recommendations(title, cosine_sim = cosine_sim):
    pd.set_option("max_colwidth", None)

    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key = lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:number + 1]
    new_index = [i[0] for i in sim_scores]
```

The Specific Title Recommendations of CineMate

```
        def get_recommendations(title, cosine_sim = cosine_sim):
            pd.set_option("max_colwidth", None)

            idx = indices[title]
            sim_scores = list(enumerate(cosine_sim[idx]))
            sim_scores = sorted(sim_scores, key = lambda x: x[1], reverse=True)
            sim_scores = sim_scores[1:number + 1]
            new_index = [i[0] for i in sim_scores]
            titles_rec = merged_df["Title"].iloc[new_index]

            recs_df = pd.DataFrame(titles_rec)
            recommended_df = recs_df.merge(merged_df, on="Title")
            recommended_df = recommended_df[["Title", "Type", "Release Year", "Genre", "Description"]]
            recommended_df.index = pd.RangeIndex(start=1, stop=len(recommended_df) + 1)
            center("RECOMMENDATIONS")
            return display(recommended_df)

        return get_recommendations(input_title)

system_option(system)
center("Thank you for using CineMate, we hope to see you again!")
```

The User Interaction of CineMate

```
username = input('Hi! What is your name? ')
print('\nHi,', username, '! I am CineMate, your personalized companion for discovering the best movies and TV shows tailored to your unique tastes.')
print('Whether you are a movie buff or a casual viewer, CineMate ensures that your next watch is always a match.')
print('')
print('So to get this started: what exactly are you looking for?')
system = input('\nDo you want recommendations based on genre, or do you want recommendations based on a certain movie or TV show? (Genre, Specific) ')

def system_option(x):
    exit = False
    if x.upper() == 'GENRE':
        kind = input('\nGreat! So what are you looking for? Recommendations for a movie, TV show, or both? ')
        genre = input('What genre of movie? (Action, Adventure, Comedy, Drama, Horror, Romance, Science Fiction, Fantasy, Historical, Crime) ')
        number = int(input('\nHow many suggestions do you want? (1 to 10) '))
```



Code Walkthrough of CineMate

Thank you for listening!