Import Libraries

Objective

- Group titles based on genre, rating, and duration.
- Build a content based recommendation system using text similarity.
- Help users find similar shows or movies using data driven methods.

```
import pandas as pd
import requests
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pickle
```

Lord Data

```
import kagglehub
path = kagglehub.dataset_download("shivamb/netflix-shows")
```

```
Using Colab cache for faster access to the 'netflix-shows' dataset.
```

```
df = pd.read_csv(f'{path}/netflix_titles.csv')
```

```
TMDB_API_KEY = "f68d22a6e8fad653b5077015c5ca8fe3" # keep hidden
```

Data Cleaning

```

cols = ['title', 'type', 'director', 'cast', 'country', 'release_year']
df = df[cols].fillna('')
```

Genre Fetching Using TMDb

```
GENRE_MAP = {
28: 'Action', 12: 'Adventure', 16: 'Animation', 35: 'Comedy', 80: 'Crime',
99: 'Documentary', 18: 'Drama', 10751: 'Family', 14: 'Fantasy', 36: 'History',
27: 'Horror', 10402: 'Music', 9648: 'Mystery', 10749: 'Romance',
878: 'Science Fiction', 10770: 'TV Movie', 53: 'Thriller', 10752: 'War', 37: 'Western'
}
```

```
def fetch_genre(title, year):
    try:
        url = "https://api.themoviedb.org/3/search/movie"
        params = {"api_key": TMDB_API_KEY, "query": title, "year": year}
        r = requests.get(url, params=params).json()
        if r.get('results'):
            genre_ids = r['results'][0].get('genre_ids', [])
            genres = [GENRE_MAP.get(g) for g in genre_ids if g in GENRE_MAP]
            return ' '.join(genres)
    except:
        pass
    return ''
```

Fill genre and build combined features

```
df['genre'] = ''
for i in range(min(300, len(df))):
    df.at[i, 'genre'] = fetch_genre(df.at[i, 'title'], df.at[i, 'release_year'])
```

```
df['combined_features'] = (
df['director'] + ' ' +
df['cast'] + ' ' +
df['country'] + ' ' +
df['type'] + ' ' +
df['genre']
)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   title              8807 non-null   object
 1   type               8807 non-null   object
 2   director           8807 non-null   object
 3   cast               8807 non-null   object
 4   country            8807 non-null   object
 5   release_year       8807 non-null   int64 
 6   genre              8807 non-null   object
 7   combined_features  8807 non-null   object
dtypes: int64(1), object(7)
memory usage: 550.6+ KB
```

TF-IDF

```
vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = vectorizer.fit_transform(df['combined_features'])


# Save precomputed data for Streamlit app
with open('tfidf_matrix.pkl', 'wb') as f:
    pickle.dump(tfidf_matrix, f)
df.to_csv('netflix_enriched.csv', index=False)
```

```
# Cosine similarity
cosine_sim = cosine_similarity(tfidf_matrix)
```

Recommendation Function

```
def recommend_movie(
    movie_title,
    movie_type=None,
    genre=None,
    country=None,
    min_year=None,
    top_n=10
):
    movie_title = movie_title.lower().strip()
    matches = df[df['title'].str.lower().str.contains(movie_title)]


    if matches.empty:
        return "Movie not found"


    idx = matches.index[0]
    scores = list(enumerate(cosine_sim[idx]))
    scores = sorted(scores, key=lambda x: x[1], reverse=True)


    indices = [i[0] for i in scores[1:]]
    recs = df.iloc[indices]


    if movie_type:
        recs = recs[recs['type'].isin(movie_type)]
    if genre:
        recs = recs[recs['genre'].str.contains(genre, case=False)]
    if country:
        recs = recs[recs['country'].str.contains(country, case=False)]
    if min_year:
        recs = recs[recs['release_year'] >= min_year]


    return recs.head(top_n)
```

```
recommend_movie(
movie_title="Blood & Water",
movie_type=["TV Show"],
genre="Drama",
min_year=2020
)[['title','type','genre','country','release_year']]
```

	title	type	genre	country	release_year
32	Sex Education	TV Show	Drama Romance	United Kingdom	2020
15	Dear White People	TV Show	Drama Romance	United States	2021
243	Everything Will Be Fine	TV Show	Family Drama		2021
99	On the Verge	TV Show	Drama	France, United States	2021
225	Open Your Eyes	TV Show	Drama Thriller Science Fiction		2021
275	The Kingdom	TV Show	Thriller Action Drama	Argentina	2021

Interactive Recommendation Filters and Widgets

```
import ipywidgets as widgets
from IPython.display import display, clear_output
import requests
```

```
type_options = sorted(df['type'].unique()) + ['Both']

# Widgets
search_bar = widgets.Text(description="Search", placeholder="Type a movie or TV show name")
type_filter = widgets.Dropdown(options=type_options, description="Type")
genre_filter = widgets.Text(description="Genre", placeholder="Optional")
country_filter = widgets.Text(description="Country", placeholder="Optional")
year_filter = widgets.IntSlider(value=2000, min=1980, max=2022, description="Min Year")

button = widgets.Button(description="Recommend")
output = widgets.Output()

def on_button_click(b):
    with output:
        clear_output()

        selected_type = type_filter.value
        if selected_type == 'Both':
            types = list(df['type'].unique())
        else:
            types = [selected_type]

        result = recommend_movie(
            movie_title=search_bar.value,
            movie_type=types,
            genre=genre_filter.value,
            country=country_filter.value,
            min_year=year_filter.value
        )

        display(result[['title', 'type', 'genre', 'country', 'release_year']])

button.on_click(on_button_click)

display(search_bar, type_filter, genre_filter, country_filter, year_filter, button, output)
```

Search	breaking bad
Type	Both
Genre	drama
Country	Optional
Min Year	2016

Recommend

	title	type	genre	country	release_year
171	Same Kind of Different as Me	Movie	Drama	United States	2017
227	Really Love	Movie	Romance Drama	United States	2020
247	Sweet Girl	Movie	Action Thriller Drama	United States	2021
9	The Starling	Movie	Drama	United States	2021
162	Marshall	Movie	Drama	United States, China, Hong Kong	2017
38	Birth of the Dragon	Movie	Action Drama	China, Canada, United States	2017
99	On the Verge	TV Show	Drama	France, United States	2021
15	Dear White People	TV Show	Drama Romance	United States	2021
32	Sex Education	TV Show	Drama Romance	United Kingdom	2020
119	Here and There	Movie	Drama		2020

Saving and Downloading Dataset and TF-IDF Matrix

```
df.to_csv('netflix_enriched.csv', index=False)
```

```
import pickle

with open('tfidf_matrix.pkl', 'wb') as f:
    pickle.dump(tfidf_matrix, f)
```

```
from google.colab import files

files.download('netflix_enriched.csv')
files.download('tfidf_matrix.pkl')
```

Conclusion

- The recommendation system returned relevant similar titles using description text.
- The project shows how machine learning can enhance content discovery without relying on user ratings.