



Movie Recommendation System

```
EXPLORER
...
MINI...
  > ml-100k
    app.py
    movie_recommender...

movie_recommender.py > ...
1 import pandas as pd
2 from sklearn.metrics.pairwise import cosine_similarity
3 import streamlit as st
4
5 # Step 1: Genre columns
6 genre_cols = ['unknown', 'Action', 'Adventure', 'Animation', "Children's", 'Comedy',
7               'Crime', 'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror',
8               'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']
9
10 # Step 2: Load data
11 @st.cache_data
12 def load_data():
13     ratings = pd.read_csv('ml-100k/u.data', sep='\t',
14                           names=['userId', 'movieId', 'rating', 'timestamp'])
15
16     movies = pd.read_csv('ml-100k/u.item', sep='|', encoding='latin-1', header=None,
17                           names=['movieId', 'title', 'release_date', 'video_release_date', 'IMDb_URL'] + genre_cols)
18
19     movies_simple = movies[['movieId', 'title']]
20     data = pd.merge(ratings, movies_simple, on='movieId')
21
22     # Collaborative filtering
23     user_movie_matrix = data.pivot_table(index='userId', columns='title', values='rating')
24     user_movie_matrix.fillna(0, inplace=True)
25     collab_similarity = cosine_similarity(user_movie_matrix.T)
26     similarity_df = pd.DataFrame(collab_similarity, index=user_movie_matrix.columns, columns=user_movie_matrix.columns)
27
28     # Content-based filtering
29     genre_features = movies[genre_cols]
30     genre_similarity = cosine_similarity(genre_features)
31     genre_sim_df = pd.DataFrame(genre_similarity, index=movies['title'], columns=movies['title'])
32
33     return similarity_df, genre_sim_df
34
35 # Step 3: Hybrid Recommendation
36 def hybrid_recommend(movie_name, similarity_df, genre_sim_df, top_n=5):
37     if movie_name not in similarity_df.columns or movie_name not in genre_sim_df.columns:
```

```
File Edit Selection View Go Run Terminal Help
mini_project

EXPLORER
...
MINI_PROJECT
  > ml-100k
    app.py
    movie_recommender...

movie_recommender.py > load_data
33 return similarity_df, genre_sim_df
34
35 # Step 3: Hybrid Recommendation
36 def hybrid_recommend(movie_name, similarity_df, genre_sim_df, top_n=5):
37     if movie_name not in similarity_df.columns or movie_name not in genre_sim_df.columns:
38         return ["Movie not found. Try another."]
39
40     collab_scores = similarity_df[movie_name]
41     genre_scores = genre_sim_df[movie_name]
42
43     final_scores = (collab_scores + genre_scores) / 2
44     final_scores = final_scores.sort_values(ascending=False)
45
46     return final_scores.drop(movie_name).head(top_n).index.tolist()
47
48 # Step 4: Streamlit UI
49 st.set_page_config(page_title="🎬 Movie Recommender", page_icon="🎬")
50
51 st.title("🎬 Hybrid Movie Recommendation System")
52 st.markdown("Get recommendations based on what you like!")
53
54 similarity_df, genre_sim_df = load_data()
55 movie_list = sorted(similarity_df.columns)
56
57 selected_movie = st.selectbox("Select a movie:", movie_list)
58
59 if st.button("Recommend"):
60     recommendations = hybrid_recommend(selected_movie, similarity_df, genre_sim_df)
61     st.subheader("Top 5 Recommendations:")
62     for i, rec in enumerate(recommendations, 1):
63         st.write(f"{i}. {rec}")
```

```
EXPLORER
> MINI...
> ml-100k
  app.py
  movie_recommender...

Welcome
movie_recommender.py
app.py x

app.py > ...
1 import pandas as pd
2 from sklearn.metrics.pairwise import cosine_similarity
3 import streamlit as st
4
5 # Step 1: Genre columns
6 genre_cols = ['unknown', 'Action', 'Adventure', 'Animation', "Children's", 'Comedy',
7               'Crime', 'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror',
8               'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']
9
10 # Step 2: Load data with caching
11 @st.cache_data
12 def load_data():
13     ratings = pd.read_csv('ml-100k/u.data', sep='\t',
14                           names=['userId', 'movieId', 'rating', 'timestamp'])
15
16     movies = pd.read_csv('ml-100k/u.item', sep='|', encoding='latin-1', header=None,
17                           names=['movieId', 'title', 'release_date', 'video_release_date', 'IMDb_URL'] + genre_cols)
18
19     movies_simple = movies[['movieId', 'title']]
20     data = pd.merge(ratings, movies_simple, on='movieId')
21
22     # Collaborative filtering similarity matrix
23     user_movie_matrix = data.pivot_table(index='userId', columns='title', values='rating')
24     user_movie_matrix.fillna(0, inplace=True)
25     collab_similarity = cosine_similarity(user_movie_matrix.T)
26     similarity_df = pd.DataFrame(collab_similarity, index=user_movie_matrix.columns, columns=user_movie_matrix.columns)
27
28     # Content-based similarity matrix
29     genre_features = movies[genre_cols]
30     genre_similarity = cosine_similarity(genre_features)
31     genre_sim_df = pd.DataFrame(genre_similarity, index=movies['title'], columns=movies['title'])
32
33     return similarity_df, genre_sim_df
34
35 # Step 3: Hybrid Recommendation Logic
36 def hybrid_recommend(movie_name, similarity_df, genre_sim_df, top_n=5):
37     if movie_name not in similarity_df.columns or movie_name not in genre_sim_df.columns:
38         return ["Movie not found. Try another."]
39
40     collab_scores = similarity_df[movie_name]
41     genre_scores = genre_sim_df[movie_name]
42
43     final_scores = (collab_scores + genre_scores) / 2
44     final_scores = final_scores.sort_values(ascending=False)
45
46     return final_scores.drop(movie_name).head(top_n).index.tolist()
47
48 # Step 4: Streamlit UI
49 st.set_page_config(page_title="🎬 Movie Recommender", page_icon="🎬", layout="centered")
50
51 st.markdown("<h1 style='text-align: center;'>🎬 Hybrid Movie Recommendation System</h1>", unsafe_allow_html=True)
52 st.markdown("<p style='text-align: center; font-size:18px;'>Get personalized movie suggestions based on your favorite film 🎬</p>", unsafe_allow_html=True)
53
54 st.markdown("----")
55
56 similarity_df, genre_sim_df = load_data()
57 movie_list = sorted(similarity_df.columns)
58
59 # Centered layout
60 col1, col2, col3 = st.columns([1, 2, 1])
61 with col2:
62     selected_movie = st.selectbox("🎬 Choose a movie you like:", movie_list)
63
64     if st.button("🔍 Recommend Movies"):
65         recommendations = hybrid_recommend(selected_movie, similarity_df, genre_sim_df)
66
67         st.markdown("----")
68         st.subheader("🌟 Top 5 Movie Recommendations for You:")
69         for i, rec in enumerate(recommendations, 1):
70             st.markdown(f"**{i}. 🎬 {rec}**")
```

```
59 # Centered layout
60 col1, col2, col3 = st.columns([1, 2, 1])
61 with col2:
62     selected_movie = st.selectbox("🎬 Choose a movie you like:", movie_list)
63
64     if st.button("🔍 Recommend Movies"):
65         recommendations = hybrid_recommend(selected_movie, similarity_df, genre_sim_df)
66
67         st.markdown("----")
68         st.subheader("🌟 Top 5 Movie Recommendations for You:")
69         for i, rec in enumerate(recommendations, 1):
70             st.markdown(f"**{i}. 🎬 {rec}**")
71
```



Hybrid Movie Recommendation System

Get personalized movie suggestions based on your favorite film 🎬

👤 Choose a movie you like:

101 Dalmatians (1996) ▼

🔍 Recommend Movies

🌟 Top 5 Movie Recommendations for You:

1. 🎬 First Kid (1996)
2. 🎬 Santa Clause, The (1994)
3. 🎬 Home Alone (1990)
4. 🎬 Matilda (1996)