• The platform you used to implement your system :

we used Jupyter Notebook to implement the model
and Streamlit framework to design the GUI

• All the steps you took in order to design and implement your system

First we implement the model in Jupyter Notebook using
the surprise library to build a recommender system then we
transfer the important information using pickle module in python to the source code of GUI page in Visual Studio Code editor

• Description of your ML models including all the used hyperparameters :

We choose Singular value decomposition (SVD) algorithm in surprise library which is a matrix factorization method that generalizes the eigendecomposition of a square matrix (n x n) to any matrix (n x m).
He give the minimum RMSE than other algorithm in surprise library and best time predict.

Hyperparameters:

n_epochs=14 ---------- The number of iteration of the SGD procedure.

,lr_all=0.002, ---------- The learning rate for all parameters

reg_all=0.1---------- The regularization term for all parameters
n_factors=5 ---------- The number of factors

• Description of all data preprocessing you applied to the chosen dataset :

We only marge two dataset from two different csv file, into one big dataset then

We see the dataset it don’t have any missing value so we go to implement the model

• Description of the roles and responsibilities of each student in the project (if the team consists of more than one student) :

abdulkream alqasem :

I do the implementations of model in Jupyter and two function in source code of GUI page which is: get_top_n(predictions, n=5) , top_n_recs(user_id, top_n) .

Yousef Alanazi :

I do the transfer important information using pickle module in python and
I do the design of the GUI page with best component.

• Evaluation results of your model methodology and report the following results:

Mean Absolute Error (MAE): 0.686401234

Root Mean Square Error (RMSE): 0.888376256

o All the source code :

```
import pandas as pd
from surprise import Dataset
from surprise import Reader
from surprise import SVD
from surprise.model_selection import cross_validate
from collections import defaultdict


# we load the first part of the dataset whice contain four column ["userId","movieId","rating","timestamp"]
# row : 100836, col : 4
rating_data_set=pd.read_csv("ratings.csv")
```

```
# we load the second part of the dataset whice contain four column ["movieId","title","genres"]
# row : 9742, col : 3
movie_data_set=pd.read_csv("movies.csv")
```

```
#then we merge betwwen them in movieId column to get final dataset
# row : 100836, col : 6
final_dataset_org=pd.merge(rating_data_set,movie_data_set,on="movieId")
```

```
# we see here thre are no missing data in dataset
final_dataset_org.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100836 entries, 0 to 100835
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   userId     100836 non-null  int64  
 1   movieId    100836 non-null  int64  
 2   rating     100836 non-null  float64
 3   timestamp  100836 non-null  int64  
 4   title      100836 non-null  object 
 5   genres     100836 non-null  object 
dtypes: float64(1), int64(3), object(2)
memory usage: 5.4+ MB
```

```
# here i will no how many user id in my data so when do recommendation we ask
#the costumer to enter the number of id of user to get recommendation of that user
final_dataset_org["userId"].tail()
```

```
100831    610
100832    610
100833    610
100834    610
100835    610
Name: userId, dtype: int64
```

```
rating_data_set.shape
```

```
(100836, 4)
```

```
#we will use Surprise lib for recommender systems he provide
# so the dataset must procese to get inside this lib
reader = Reader(rating_scale=(1, 5))

#we drop ["timestamp","title","genres"] becouse the lib only git It must have three columns,
#corresponding to the (raw) user ids, the item ids, and the ratings, in this order.
final_dataset_norg=final_dataset_org.drop(["timestamp","title","genres"],axis=1)

data_set = Dataset.load_from_df(final_dataset_norg, reader)
```

```
# we choose SVD in surprise it give small rmse
svd = SVD(n_epochs=14,lr_all=0.002,reg_all=0.1,n_factors=5)

train_set = data_set.build_full_trainset()
svd_fit=svd.fit(train_set)

testset = train_set.build_testset()
predictions = svd.test(testset)

cross_validate(svd, data_set, measures=['RMSE', 'MAE'],cv=5)
```

```
{'test_rmse': array([0.88560129, 0.88833111, 0.8861241 , 0.89565596, 0.88616882]),
 'test_mae': array([0.68384459, 0.68934474, 0.68304561, 0.69086042, 0.68491081]),
 'fit_time': (1.4789798259735107,
  1.4854631423950195,
  1.3521897792816162,
  1.2402839660644531,
  1.507220983505249),
 'test_time': (0.16042208671569824,
  0.2620048522949219,
  0.19508886337280273,
  0.14255213737487793,
  0.31708693504333496)}
```

```
links_data_set=pd.read_csv("links.csv")

final_dataset_for_link=pd.merge(final_dataset_org,links_data_set,on="movieId")
final_dataset_for_link=final_dataset_for_link.drop(["userId","rating","timestamp","imdbId"],axis=1)

final_dataset_org=final_dataset_org.drop("timestamp",axis=1)
```

```
import pickle
data= {"model":svd,"final_data_org":final_dataset_org,
       "final_data_norg":final_dataset_norg,"predictions":predictions,
       "final_dataset_for_link":final_dataset_for_link}
with open("saved_steps.pkl","wb") as file:
    pickle.dump(data,file)
```

Code of GUI page

app_movie.py > ...

```python
import streamlit as st
import pickle
import numpy as np
import pandas as pd
from surprise import Dataset,Reader,SVD
from surprise.model_selection import cross_validate
from collections import defaultdict


with open("saved_steps.pkl","rb") as file:
    data=pickle.load(file)


my_model=data["model"]
my_final_dataset_org=data["final_data_org"]
final_dataset_norg=data["final_data_norg"]
final_dataset_for_link=data["final_dataset_for_link"]
predictions=data["predictions"]


st.title("movie recommendation system")
st.write("## the user enter the number user id he want then he give him his top N recommendation system ")

x=st.slider("select a number of id user from 1 to 610",1,610)
st.write("you selected ",x)

st.write("### user id of number {} and his rating list ".format(x))
st.write(my_final_dataset_org[my_final_dataset_org["userId"] == x])


reader = Reader(rating_scale=(1, 5))
data_set = Dataset.load_from_df(final_dataset_norg, reader)

train_set = data_set.build_full_trainset()
my_model.fit(train_set)

testset = train_set.build_testset()
predictions = my_model.test(testset)

#this function i take from https://surprise.readthedocs.io/en/stable/FAQ.html
#from surprise lib to get top 5 from same lib
def get_top_n(predictions, n=5):
    """Return the top-N recommendation for each user from a set of predictions.

    Args:
        predictions(list of Prediction objects): The list of predictions, as
            returned by the test method of an algorithm.
        n(int): The number of recommendation to output for each user. Default
            is 5.
```

```
    Returns:
    A dict where keys are user (raw) ids and values are lists of tuples:
        [(raw item id, rating estimation), ...] of size n.
    """

    # First map the predictions to each user.
    top_n = defaultdict(list)
    for uid, iid, true_r, est, _ in predictions:
        top_n[uid].append((iid, est))

    # Then sort the predictions for each user and retrieve the k highest ones.
    for uid, user_ratings in top_n.items():
        user_ratings.sort(key=lambda x: x[1], reverse=True)
        top_n[uid] = user_ratings[:n]

    return top_n

def top_n_recs(user_id, top_n):
    top_n = get_top_n(predictions, n=top_n)
    return pd.DataFrame(top_n[user_id], columns=["movieId", "rating"])

top_N=st.slider("select the number of N in top N to see",1,20)

st.write("### \nTop %d recommendations for user %d" % (top_N, x))

st.write("--------------------------")

top_5_df=top_n_recs(x,top_N)
for i in range(top_N):
    movie=final_dataset_for_link.loc[top_5_df["movieId"].loc[i]]
    st.write("title is : ",movie["title"])
    st.write("genres : ",movie["genres"])
    st.write("the {title} in TMDP is : https://www.themoviedb.org/movie/{number_of_movie}"
    .format(title=movie["title"],number_of_movie=int(movie["tmdbId"])))
    st.write("--------------------------")
```

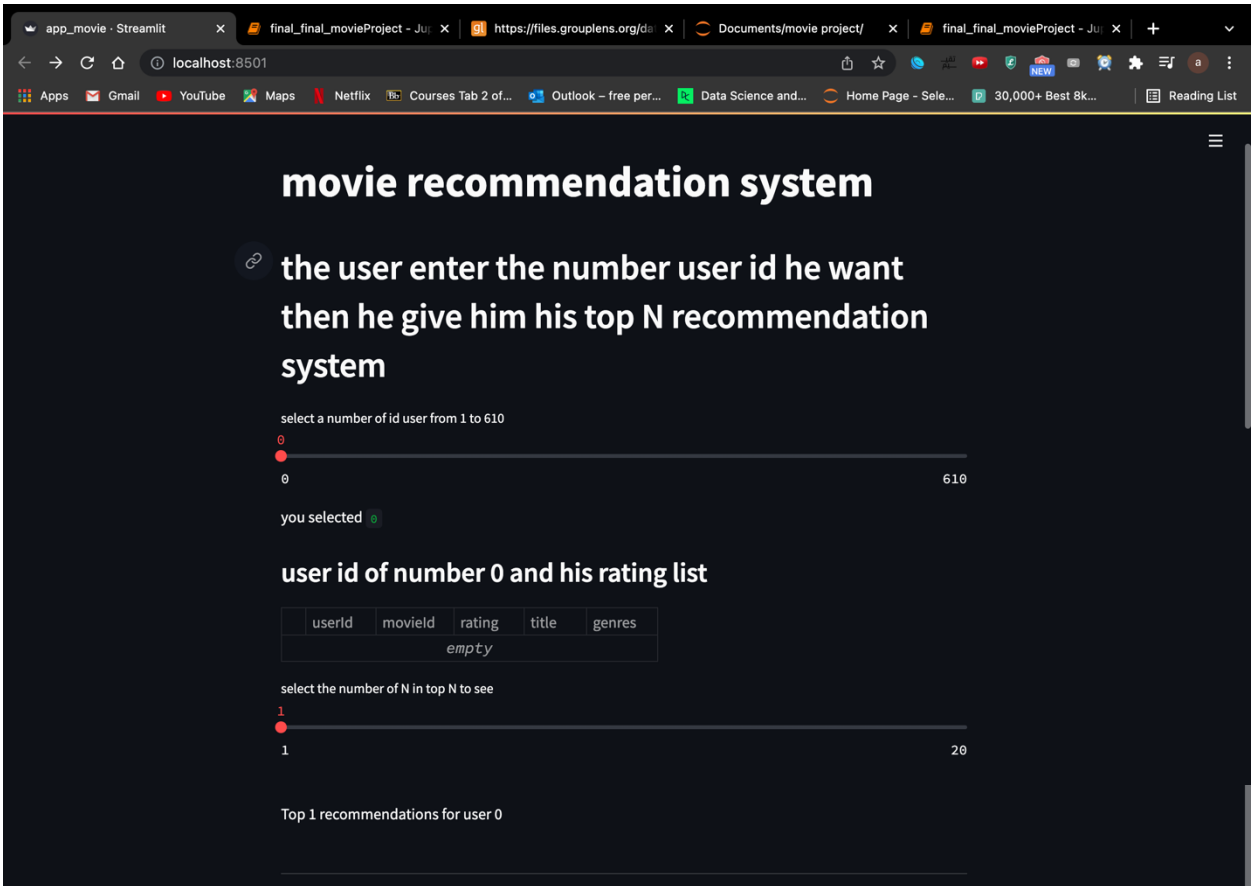
o Evaluation results of your models

```
cross_validate(svd, data_set, measures=['RMSE', 'MAE'],cv=5)
```

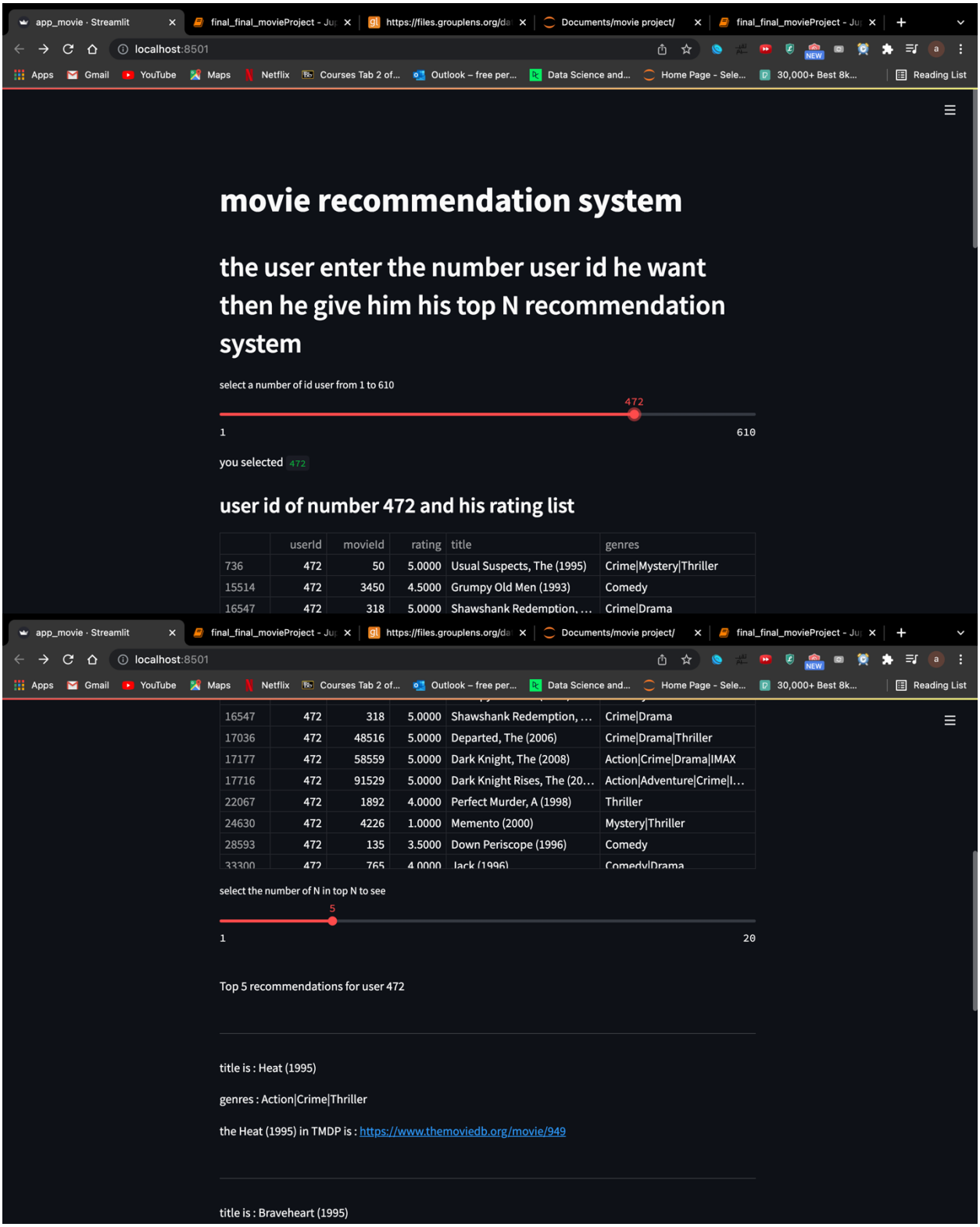
```
{'test_rmse': array([0.88560129, 0.88833111, 0.8861241 , 0.89565596, 0.88616882]),
 'test_mae': array([0.68384459, 0.68934474, 0.68304561, 0.69086042, 0.68491081]),
 'fit_time': (1.4789798259735107,
  1.4854631423950195,
  1.3521897792816162,
  1.2402839660644531,
  1.507220983505249),
 'test_time': (0.16042208671569824,
  0.2620048522949219,
  0.19508886337280273,
  0.14255213737487793,
  0.31708693504333496)}
```

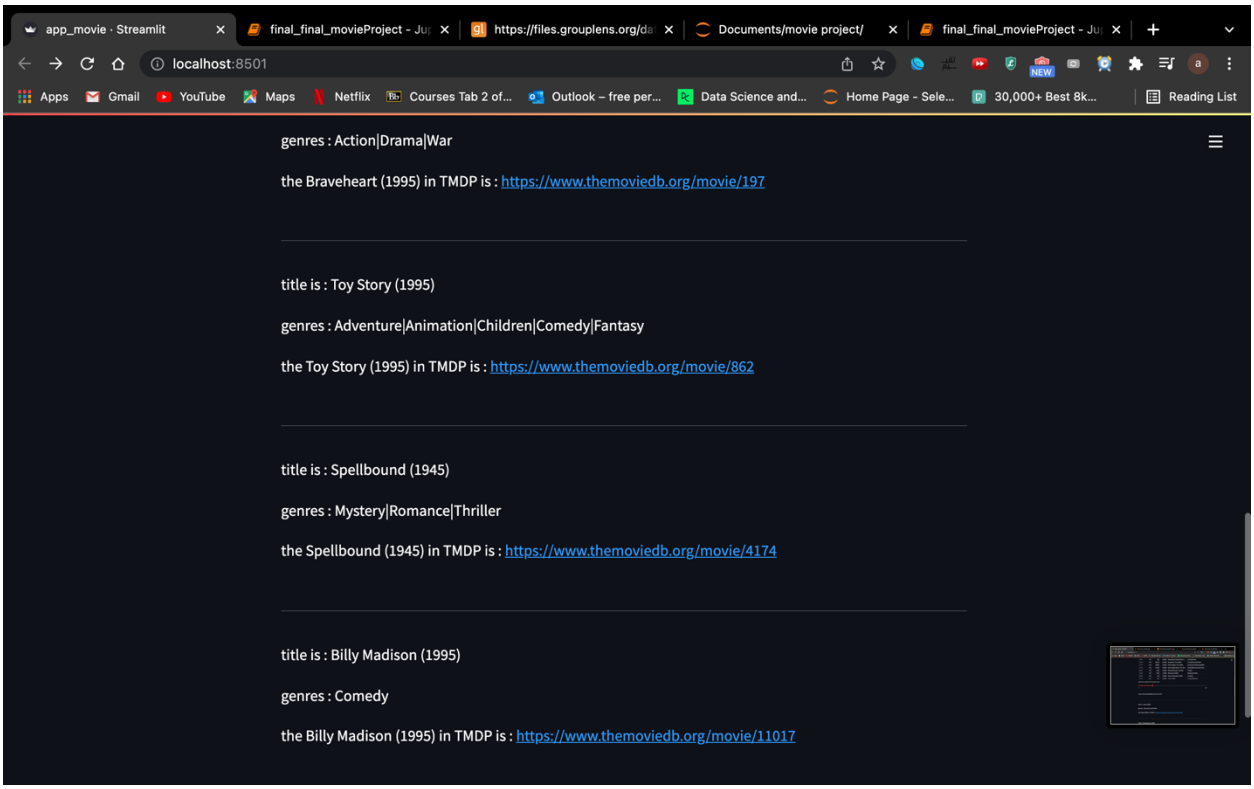
We sum all five RMSE and divide it by 5 to get RMSE which is : 0.888376256
We sum all five MAE and divide it by 5 to get MAE which is : 0.686401234

Demo of the GUI before



after prediction




app_movie · Streamlit
final_final_movieProject - Jup
https://files.grouplens.org/da
Documents/movie project/
final_final_movieProject - Jup
localhost:8501
Apps
Gmail
YouTube
Maps
Netflix
Courses Tab 2 of...
Outlook – free per...
Data Science and...
Home Page - Sele...
30,000+ Best 8k...
Reading List
genres : Action|Drama|War
the Braveheart (1995) in TMDP is : https://www.themoviedb.org/movie/197
title is : Toy Story (1995)
genres : Adventure|Animation|Children|Comedy|Fantasy
the Toy Story (1995) in TMDP is : https://www.themoviedb.org/movie/862
title is : Spellbound (1945)
genres : Mystery|Romance|Thriller
the Spellbound (1945) in TMDP is : https://www.themoviedb.org/movie/4174
title is : Billy Madison (1995)
genres : Comedy
the Billy Madison (1995) in TMDP is : https://www.themoviedb.org/movie/11017