National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Metehan Bereketoglu
x22161872

School of Computing
National College of Ireland

Supervisor: Musfira Jilani

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Metehan Bereketoglu
Student ID:	x22161872
Programme:	Data Analytics
Year:	2023
Module:	Msc Research Project
Supervisor:	Musfira Jilani
Submission Due Date:	14/12/2023
Project Title:	Configuration Manuel
Word Count:	68
Page Count:	7

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Metehan Bereketoglu
Date:	14/12/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	X
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	X
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	X

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Metehan Bereketoglu
x22161872

1 Introduction

Libraries, hardware and software configurations, and essential code fragments for every stage of implementation are all included in the setup manual. (Mane; 2017).

2 System Requirement

2.1 Device Requirement

Processor: Intel® CoreTM i3 (minimum processor needed) RAM: 4GB These setups are the very minimum needed to meet hardware specifications.

2.2 Software Requirement

This project's implementation makes use of Python's abilities, an interpreted programming language that is well-known for its dynamism, high level capabilities, and adaptability. Python is a great option for developing applications since it supports object-oriented programming and provides a wide range of high-level data structures in a user-friendly and easy-to-learn environment.

Python is a particularly appealing language for application development because of its strong power and flexibility along with its ease of learning.

The project made use of the Google Colab tool during the coding phase. Google Colab offers an interactive environment that is similar to a lab notebook and is powered by Jupyter Notebook, an open-source and publicly available platform created by the Jupyter Project. In this setting, editable code, comments, and data coexist in harmony.

3 Packages and the Library

I used a variety of libraries and packages in my project, such as math, os, pandas, matplotlib.pyplot, seaborn, numpy, and time, to make tasks like data manipulation, visualisation, and numerical operations and time-related functions easier.

4 Research Implementation

4.1 Importing Necessary Libraries

Along with The Surprise library for collaborative filtering-based recommendation models, numpy for numerical operations, scikit-learn for machine learning algorithms, time and math for time-related functionalities, and pandas for data manipulation are among the essential libraries for data analysis and machine learning that this code imports. Other modules required for model evaluation and cross-validation are also imported.

```
# import necessary libraries
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
import time
import math
from surprise import Dataset
from surprise import Reader
from surprise import SVD, NMF, SlopeOne, KNNBasic, KNNWithMeans, KNNBaseline, CoClustering, BaselineOnly, NormalPred
from surprise.model_selection import cross_validate
from surprise.model_selection import KFold
from surprise.model_selection import GridSearchCV
from surprise import accuracy, Dataset, Reader, SVD
from surprise.model_selection import PredefinedKFold
from surprise.model_selection import  train_test_split as tts
```

Figure 1: Importing Libraries

4.2 Loading Data and Printing

This code creates DataFrames for movies, ratings, user information, genre information, profession information, and separate sets for training and testing the recommendation model. It does this by loading movie- and user-related data from specified files using pandas.

```
# load data
data_path = '/content/drive/MyDrive/Movie_Recommendation/Data'

movies = pd.read_csv(f'{data_path}/u.item', sep='|', encoding='latin-1', header=None, names=['movie_id', 'movie_titl
ratings = pd.read_csv(f'{data_path}/u.data', sep='\t', header=None, names=['user_id', 'movie_id', 'rating', 'timesta

user_info = pd.read_csv(f'{data_path}/u.user', sep='|', header=None, names=['user_id', 'age', 'gender', 'occupation'
genre_info = pd.read_csv(f'{data_path}/u.genre', sep='|', header=None, names=['genre_id', 'genre_name'])
occupation_info = pd.read_csv(f'{data_path}/u.occupation', sep='|', header=None, names=['occupation'])

u_base = pd.read_csv(f'{data_path}/u1.base', sep='\t', header=None, names=['user_id', 'movie_id', 'rating', 'timesta
u_test = pd.read_csv(f'{data_path}/u1.test', sep='\t', header=None, names=['user_id', 'movie_id', 'rating', 'timesta
```

```
print('Movie information shape: ', ratings.shape)
movies.head()
```

```
Movie information shape:  (100000, 4)
```

	movie_id	movie_title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)

Figure 2: Loading Data and Displaying

4.3 Data Analysing

In order to comprehend the overall properties of the dataset, statistical metrics were computed. This information proved invaluable for making well-informed decisions during the recommendation system's development.(Ronald E. Walpole and Ye; 2012)

```
movie_ratings = pd.merge(ratings, movies, on='movie_id')
print('Movie_Ratings dataset shape: ', movie_ratings.shape)
movie_ratings.head()
```

```
Movie_Ratings dataset shape:  (100000, 5)
```

	user_id	movie_id	rating	timestamp	movie_title
0	196	242	3	881250949	Kolya (1996)
1	63	242	3	875747190	Kolya (1996)
2	226	242	5	883888671	Kolya (1996)
3	154	242	3	879138235	Kolya (1996)
4	306	242	5	876503793	Kolya (1996)

Figure 3: Data Analysing-1

```
x = movie_ratings_users.groupby('user_id')['rating'].count()
plt.hist(x , bins = 20)
plt.xlabel("Number of rating")
plt.ylabel("Number of users")
plt.title("Distebution of rating for users");
```

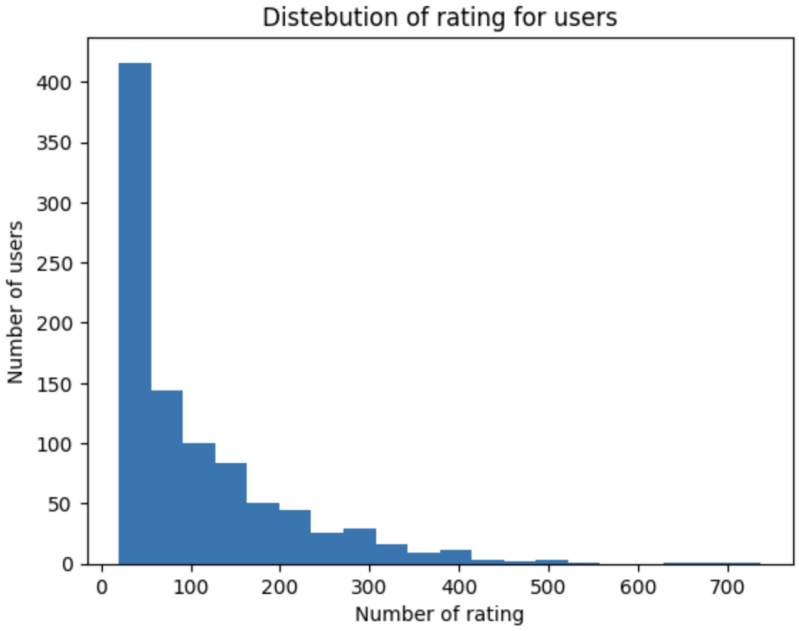

Figure 4: Distebution of Rating for Users

```
movie_ratings_users = pd.merge(movie_ratings, user_info, on='user_id')
print('Movie_Ratings_user dataset shape: ', movie_ratings_users.shape)
movie_ratings_users.head()
```

```
Movie_Ratings_user dataset shape:  (100000, 9)
```

	user_id	movie_id	rating	timestamp	movie_title	age	gender	occupation	zip_code
0	196	242	3	881250949	Kolya (1996)	49	M	writer	55105
1	196	257	2	881251577	Men in Black (1997)	49	M	writer	55105
2	196	111	4	881251793	Truth About Cats & Dogs, The (1996)	49	M	writer	55105
3	196	25	4	881251955	Birdcage, The (1996)	49	M	writer	55105
4	196	382	4	881251843	Adventures of Priscilla, Queen of the Desert, ...	49	M	writer	55105

Figure 5: Movie Rating Dataset Shape

```
plt.figure(figsize = (15, 5))
sns.countplot(movie_ratings_users, x = 'occupation', hue = 'rating')
plt.xticks(rotation = 30)
plt.title('Distribution of occupation per rating');
```

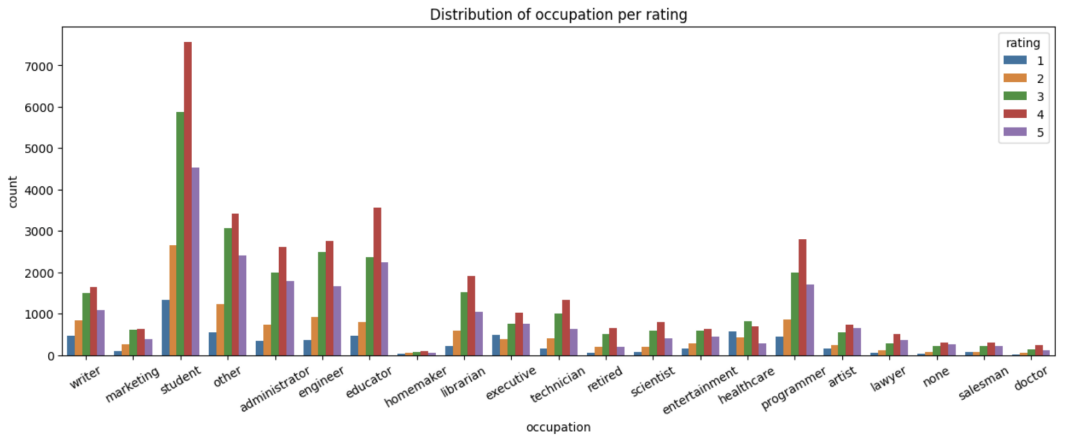

Figure 6: Distebution of Occupation Per Rating

5 Model Evaluation

This method records the mean absolute error (MAE) and root mean squared error (RMSE) for each recommendation model and outputs the results in a DataFrame. It does this by using cross-validation to assess the performance of several recommendation models (SVD, NMF, SlopeOne, KNNWithMeans, and KNNBaseline) on the training data.

```python
# try different models on train data to find the best model
Results = pd.DataFrame(columns = ['Model','RMSE','MAE','Duration'])
models = (SVD, NMF, SlopeOne, KNNWithMeans, KNNBaseline)
models_name = ['SVD', 'NMF', 'SlopeOne', 'KNNWithMeans', 'KNNBaseline']
# Set number of itteration while doing cross validation
kf = KFold(3, random_state=0)
# itterate in models list and run all the model and calculate the perormance of each model
for model, name in zip(models, models_name):
    t0 = time.time()
    output = cross_validate(model(), train_df, ['rmse', 'mae'], kf)
    mean_rmse = '%.3f' % np.mean(output['test_rmse'])
    mean_mae = '%.3f' % np.mean(output['test_mae'])
    duration = round((time.time() - t0 ) / 60, 2)
    Results = Results.append({'Model': name, 'RMSE': mean_rmse, 'MAE': mean_mae, 'Duration': duration}, ignore_index
# print the results
print("\nThe results of models:\n")
Results
```

Figure 7: To determine which model is best, test several models on train data.

	Model	RMSE	MAE	Duration
0	SVD	0.944	0.745	0.11
1	NMF	0.972	0.764	0.08
2	SlopeOne	0.941	0.740	0.15
3	KNNWithMeans	0.949	0.749	0.20
4	KNNBaseline	0.931	0.735	0.25

Figure 8: The Result of the Models

6 Prediction

The user is prompted to provide their ID and the desired amount of movie suggestions by this user interface code, which then calls the get-recommendation method to retrieve and show the suggested movies for the designated user.

```python
def get_recommendation(user_id , n):
  '''
  This function use to get the recommendation movies for a specific user
  :param
    user_id: the identification code of user
    n : number of movie to recommend
  return: list of n movies that system recommend
  '''
  # filter the movies watched by user before
  user_movies_list = final_data[final_data['user_id']== user_id]

  # load the user data in surprise format
  reader = Reader(rating_scale=(1,5))
  data = Dataset.load_from_df(final_data[['user_id', 'movie_id', 'rating']], reader)

  # train the model
  bsl_options = {
    'method': 'als',
    'n_epochs': 15  }
  sim_options= {
      'name':'pearson_baseline',
      'shrinkage':  5 }

  model = KNNBaseline(sim_options=sim_options, bsl_options=bsl_options, verbose= False)

  train_data = data.build_full_trainset()
  model.fit(train_data)

  # get top n recommendation
  movies_id = movie_ratings['movie_id'].unique().tolist()
  for movie_id in user_movies_list['movie_id'].to_list():
    if movie_id in movies_id :
      movies_id.remove(movie_id)

  # predict rating for user
  predictions = []
  for movie_id in movies_id:
    predictions.append((movie_id, model.predict(int(user_id), int(movie_id)).est))

  #predictions = model.test(test_data)
  # prepare recommendation output
  top_n = sorted(predictions, key = lambda x: x[1], reverse = True)
  result = pd.DataFrame(top_n, columns = ['movie_id', 'rating'])
  top_n movies = [x[0] for x in top_n][:n]
```

Figure 9: Prediction-1

```python
# user interface
user_id = int(input('please enter user id: '))
n = int(input('please enter number on movie recommendation: '))
recommend_movie_list = get_recommendation(user_id, n)
print(f'\nRecommended movies for user id {user_id} is: ')
for name in  recommend_movie_list:
    print("\n", name[0])
```

```
please enter user id: 196
please enter number on movie recommendation: 5

Recommended movies for user id 196 is:

 North by Northwest (1959)

 12 Angry Men (1957)

 Citizen Kane (1941)

 Wallace & Gromit: The Best of Aardman Animation (1996)

 One Flew Over the Cuckoo's Nest (1975)
```

Figure 10: Prediction-2

References

Mane, P. (2017). Predictive accuracy of recommendation algorithms, *Doctoral dissertation, Purdue University* .

Ronald E. Walpole, Raymond H. Myers, S. L. M. and Ye, K. (2012). Probability statistics for engineers scientists, *Pearson* .