Recommender Systems

**Prof. Dietmar Jannach Alpen‐Adria‐Universität Klagenfurt**

**Assignment 2**

# Task 1) Installing Pandas

* Download and install the *pandas* package using the *pip* or *conda* commands1.
* You can find a helpful tutorial here: [https://www.tutorialspoint.com/python\_pandas/index.htm](http://www.tutorialspoint.com/python_pandas/index.htm)

# Task 2) Playing with Pandas Task 2.1) Getting used to Series

* Create a list of strings as follows:

data = [**'Toy Story'**,**'Jumanji'**,**'Grumpier Old Men'**]

* Create a Pandas Series from the list, then:
  + Print the first element
  + Print the first two elements
  + Print the last two elements
* Create a new series from the list with defined indexes: [‘a’, ’b’, ’c’].
* Print the element at index position ‘b’.

CODE:

import pandas as pd  
  
# Create a list of strings  
data = ['Toy Story', 'Jumanji', 'Grumpier Old Men']  
  
# Convert the list to a Pandas Series  
movie\_series = pd.Series(data)  
  
# Print the first element  
print("First Element:")  
print(movie\_series[0])  
  
# Print the first two elements  
print("\nFirst Two Elements:")  
print(movie\_series[:2])  
  
# Print the last two elements  
print("\nLast Two Elements:")  
print(movie\_series[-2:])  
  
# Create a new series with defined indexes  
indexed\_series = pd.Series(data, index=['a', 'b', 'c'])  
  
# Print the element at index 'b'  
print("\nElement at index 'b':")  
print(indexed\_series['b'])

# Task 2.2) Getting used to DataFrames

* Create a nested list as follows:

data = [[**'Toy Story'**,21.946943], [**'Jumanji'**,17.015539],[**'Grumpier Old Men'**,11.7129]]

* Create a DataFrame object from the nested list with column headings ‘title’ and ‘popularity’.
* Create a new DataFrame which has the entries sorted by popularity in ascending order.
* Print the popularity values.

Code:

import pandas as pd  
  
# Create a nested list with movie titles and their popularity  
data = [['Toy Story', 21.946943], ['Jumanji', 17.015539], ['Grumpier Old Men', 11.7129]]  
  
# Create a DataFrame from the nested list with specified column names  
movies\_df = pd.DataFrame(data, columns=['title', 'popularity'])  
  
# Print the initial DataFrame  
print("Initial DataFrame:")  
print(movies\_df)  
  
# Create a new DataFrame sorted by the 'popularity' column in ascending order  
sorted\_movies\_df = movies\_df.sort\_values(by='popularity', ascending=True)  
  
# Print the sorted DataFrame  
print("\nSorted DataFrame by Popularity (Ascending):")  
print(sorted\_movies\_df)  
  
# Print the popularity values from the sorted DataFrame  
print("\nPopularity Values from Sorted DataFrame:")  
print(sorted\_movies\_df['popularity'])

# Task 3) Analyzing a movie dataset

* Download the movie metadata dataset at
* [https://www.kaggle.com/r](http://www.kaggle.com/rounakbanik/the)o[unakbanik/](http://www.kaggle.com/rounakbanik/the)the‐movies‐dataset/
* (If the link does not work, go to https://[www.kaggle.com/rounakbanik](http://www.kaggle.com/rounakbanik) and navigate to the Movies Dataset)
* Write a program that does the following:
  + Read the CSV file “movies\_metadata” into a Pandas DataFrame.

1 You can also follow this instruction https://[www.jetbrains.com/help/pycharm/installing‐uni](http://www.jetbrains.com/help/pycharm/installing)nstalling‐and‐ upgrading‐packages.html

* + Use the *type* function to inspect the DataFrame, i.e, inspect the output of the command:

print(type(df))

* + Print the information about the first and the last movie in the dataset.
  + Show the information about the movie “Jumanji”.
* Create a smaller DataFrame called small\_df from the given one by considering only the following columns: 'title', 'release\_date', 'popularity', 'revenue', 'runtime' and 'genres',
* Create a function “to\_float” to convert the type of its input to float with following code:

**def** to\_float(x):

**try**:

x = float(x)

**except**:

x = numpy.nan

**return** x

* Next, add the following code to your program that adds a column names ‘release\_year’ to the DataFrame. Inspect how the lambda function is working.

small\_df = df[[**'title'**, **'release\_date'**, **'popularity'**, **'revenue'**, **'runtime'**, **'genres'**]].copy()

small\_df.loc[**'release\_date'**] = pd.to\_datetime(small\_df[**'release\_date'**], errors=**'coerce'**)

small\_df[**'release\_year'**] = small\_df[**'release\_date'**].apply

(**lambda** x: str(x).split(**'-'**)[0] **if** x != numpy.nan **else** numpy.nan) small\_df[**'release\_year'**] = small\_df[**'release\_year'**].apply(to\_float) small\_df[**'release\_year'**] = small\_df[**'release\_year'**].astype(**'float'**) small\_df = small\_df.drop(columns=**"release\_date"**)

* Now, print the titles of all movies that were released after the year 2010.

CODE:

import pandas as pd  
import numpy as np  
import zipfile  
  
# Path to the zip file and internal CSV file  
zip\_path = 'C:\\Users\\user\\Documents\\Uni\\Semester 2\\Information Search & Recommendation Systems\\HW\\archive.zip'  
csv\_file = 'movies\_metadata.csv'  
  
# Function to convert to float with error handling  
def to\_float(x):  
 try:  
 x = float(x)  
 except:  
 x = np.nan  
 return x  
  
# Extracting the dataset from the zip file  
with zipfile.ZipFile(zip\_path, 'r') as z:  
 with z.open(csv\_file) as f:  
 # Read the CSV file into a DataFrame  
 df = pd.read\_csv(f)  
  
 # Inspect the type of the DataFrame  
 print("Type of DataFrame:")  
 print(type(df))  
  
 # Print information about the first and the last movie in the dataset  
 print("\nFirst movie in the dataset:")  
 print(df.iloc[0])  
  
 print("\nLast movie in the dataset:")  
 print(df.iloc[-1])  
  
 # Show the information about the movie "Jumanji"  
 jumanji\_info = df[df['title'] == 'Jumanji']  
 print("\nInformation about Jumanji:")  
 print(jumanji\_info)  
  
# Creating a smaller DataFrame with specific columns  
small\_df = df[['title', 'release\_date', 'popularity', 'revenue', 'runtime', 'genres']].copy()  
  
# Processing release\_date to extract the year  
small\_df['release\_date'] = pd.to\_datetime(small\_df['release\_date'], errors='coerce')  
small\_df['release\_year'] = small\_df['release\_date'].apply(lambda x: str(x).split('-')[0] if pd.notnull(x) else np.nan)  
small\_df['release\_year'] = small\_df['release\_year'].apply(to\_float)  
small\_df['release\_year'] = small\_df['release\_year'].astype('float')  
small\_df = small\_df.drop(columns="release\_date")  
  
# Print the titles of all movies released after the year 2010  
movies\_after\_2010 = small\_df[small\_df['release\_year'] > 2010]  
print("\nMovies released after 2010:")  
print(movies\_after\_2010['title'])

# Task 4) Analyzing a rating dataset

* Read the file “ratings\_small.csv” into a Pandas DataFrame.
* Our goal is now to compute for every movie its mean and median rating value. For each movie, we would therefore like to print a dictionary like this:

{'id': 1, 'rating\_mean': 3.8724696356275303, 'rating\_median': 4.0}

* + Use the method “groupBy” to organize the DataFrame by “movieID”.
  + Iterate over the resulting grouped data structure
    - For each tuple, use the methods “mean()” and “median()” to determine the values for each movie.
    - Add a new dictionary entry to the resulting list
  + Print the list of dictionaries.

CODE:

import pandas as pd  
import numpy as np  
import zipfile  
  
# Path to the zip file and internal CSV file  
zip\_path = 'C:\\Users\\user\\Documents\\Uni\\Semester 2\\Information Search & Recommendation Systems\\HW\\archive.zip'  
csv\_file = 'movies\_metadata.csv'  
  
# Function to convert to float with error handling  
def to\_float(x):  
 try:  
 x = float(x)  
 except:  
 x = np.nan  
 return x  
  
# Extracting the dataset from the zip file  
with zipfile.ZipFile(zip\_path, 'r') as z:  
 with z.open(csv\_file) as f:  
 # Read the CSV file into a DataFrame  
 df = pd.read\_csv(f)  
  
 # Inspect the type of the DataFrame  
 print("Type of DataFrame:")  
 print(type(df))  
  
 # Print information about the first and the last movie in the dataset  
 print("\nFirst movie in the dataset:")  
 print(df.iloc[0])  
  
 print("\nLast movie in the dataset:")  
 print(df.iloc[-1])  
  
 # Show the information about the movie "Jumanji"  
 jumanji\_info = df[df['title'] == 'Jumanji']  
 print("\nInformation about Jumanji:")  
 print(jumanji\_info)  
  
# Creating a smaller DataFrame with specific columns  
small\_df = df[['title', 'release\_date', 'popularity', 'revenue', 'runtime', 'genres']].copy()  
  
# Processing release\_date to extract the year  
small\_df['release\_date'] = pd.to\_datetime(small\_df['release\_date'], errors='coerce')  
small\_df['release\_year'] = small\_df['release\_date'].apply(lambda x: str(x).split('-')[0] if pd.notnull(x) else np.nan)  
small\_df['release\_year'] = small\_df['release\_year'].apply(to\_float)  
small\_df['release\_year'] = small\_df['release\_year'].astype('float')  
small\_df = small\_df.drop(columns="release\_date")  
  
# Print the titles of all movies released after the year 2010  
movies\_after\_2010 = small\_df[small\_df['release\_year'] > 2010]  
print("\nMovies released after 2010:")  
print(movies\_after\_2010['title'])

# Task 5) Finding similar users

* Read the file “ratings\_small.csv” into a Pandas DataFrame.
* Determine the set of users watched by an arbitrary user A (e.g. the first one)
  + Group the DataFrame by “userID” (using the groupBy‐function)
  + Take the frist group/user (using “get\_group”), access the rated movies by ID and use the “set” function to create a set of the returned values.
  + Print the set of rated movies for user A.
* Given this set of movies, our goal is now to find other users who have rated at least three of the movies that user A has rated. Proceed as follows.
  + Iterate over the already grouped DataFrame
    - Access the rated movies
    - Use a set intersection operation to see if the current user has rated at least three of the movies that user A has rated.
    - If so, remember the user.
  + Print all users with an intersection of at least three movies.

CODE:

import pandas as pd  
import zipfile  
  
# Path to the zip file  
zip\_path = r'C:\Users\user\Documents\Uni\Semester 2\Information Search & Recommendation Systems\HW\archive.zip'  
csv\_file = 'ratings\_small.csv' # Assuming the file name inside the zip is correct  
  
# Use zipfile to open the archive and read the file  
with zipfile.ZipFile(zip\_path, 'r') as z:  
 with z.open(csv\_file) as f:  
 # Read the CSV file into a DataFrame  
 ratings\_df = pd.read\_csv(f)  
  
# Group the DataFrame by 'userId'  
grouped\_users = ratings\_df.groupby('userId')  
  
# Get the set of movies watched by the first user (User A)  
user\_a\_id = next(iter(grouped\_users.groups))  
user\_a\_movies = set(grouped\_users.get\_group(user\_a\_id)['movieId'])  
  
# Print the set of rated movies for user A  
print(f"Movies rated by User {user\_a\_id}: {user\_a\_movies}")  
  
# Find other users who have rated at least three of the movies that user A has rated  
similar\_users = []  
  
# Iterate over the grouped DataFrame  
for user\_id, group in grouped\_users:  
 # Skip the comparison with User A itself  
 if user\_id == user\_a\_id:  
 continue  
  
 # Access the rated movies by the current user  
 user\_movies = set(group['movieId'])  
  
 # Use set intersection to find common movies  
 common\_movies = user\_a\_movies.intersection(user\_movies)  
  
 # Check if the current user has rated at least three of the movies that user A has rated  
 if len(common\_movies) >= 3:  
 similar\_users.append(user\_id)  
  
# Print all users with an intersection of at least three movies  
print(f"Users with at least three common rated movies with User {user\_a\_id}: {similar\_users}")