Recommender Systems

Prof. Dietmar Jannach
Alpen-Adria-Universität Klagenfurt, Summer Term 2021

Assignment 2

Task 1) Installing Pandas

- Download and install the *pandas* package using the *pip* or *conda* commands[1].
- You can find a helpful tutorial here: https://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’.

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.

Task 3) Analyzing a movie dataset

- Download the movie metadata dataset at
- https://www.kaggle.com/rounakbanik/the-movies-dataset/
- (If the link does not work, go to https://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-uninstalling-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.

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.

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.