Untitled5

June 15, 2023

[1]:
```
"""Filmempfehlungssystem"""
```

[1]:
```
'Filmempfehlungssystem'
```

[2]:
```
"""1. Nutzerbewertungen und 2. Nutzerdaten"""
```

[2]:
```
'1. Nutzerbewertungen und 2. Nutzerdaten'
```

[3]:
```
# wir setzen eine Version von KMeans Cluster (Nearest Neighbor) ein.
```

[4]:
```
# Daten Vorbereitung

import pandas as pd
import numpy as np
```

[6]:
```
movies_df = pd.read_csv('/Users/h4/desktop/movies.csv',
                        usecols=['movieId', 'title'], # information über Filme␣
 ↪und Id
                        dtype={'movieId':'int32', 'title':'str'}) # nur integer␣
 ↪von movie Id und strings von title
movies_df.head()
```

[6]:

	movieId	title
0	1	Toy Story (1995)
1	2	Jumanji (1995)
2	3	Grumpier Old Men (1995)
3	4	Waiting to Exhale (1995)
4	5	Father of the Bride Part II (1995)

[8]:
```
ratings_df = pd.read_csv('/Users/h4/desktop/ratings.csv',
                         usecols=['userId', 'movieId', 'rating', 'timestamp'],
                         dtype={'userId':'int32', 'movieId':'int32', 'rating':
 ↪'float32'})
ratings_df.head()
```

[8]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703

```
1       1        3     4.0  964981247
2       1        6     4.0  964982224
3       1       47     5.0  964983815
4       1       50     5.0  964982931
```

[10]:
```
# Prüfungsrelevant (PR)

# merge von 2 Dataframes: werden zusammen gebracht

movies_merged_df = movies_df.merge(ratings_df, on = 'movieId')

movies_merged_df.head()
```

[10]:
```
   movieId             title  userId  rating   timestamp
0        1  Toy Story (1995)       1     4.0   964982703
1        1  Toy Story (1995)       5     4.0   847434962
2        1  Toy Story (1995)       7     4.5  1106635946
3        1  Toy Story (1995)      15     2.5  1510577970
4        1  Toy Story (1995)      17     4.5  1305696483
```

[11]:
```
#PR

# löschung von NaN (löschung von nicht existierende Daten)

# dropna

movies_merged_df = movies_merged_df.dropna(axis=0, # axis = 0 sind Säulen
                                           subset = ['title'])

movies_merged_df.head()
```

[11]:
```
   movieId             title  userId  rating   timestamp
0        1  Toy Story (1995)       1     4.0   964982703
1        1  Toy Story (1995)       5     4.0   847434962
2        1  Toy Story (1995)       7     4.5  1106635946
3        1  Toy Story (1995)      15     2.5  1510577970
4        1  Toy Story (1995)      17     4.5  1305696483
```

[13]:
```
movies_average_rating = movies_merged_df.groupby('title')['rating'].mean().
↪sort_values(ascending=False).reset_index().rename(columns={'rating':'Average␣
↪Rating'})
movies_average_rating.head()
```

[13]:
```
                               title  Average Rating
0           Gena the Crocodile (1969)             5.0
1                 True Stories (1986)             5.0
2       Cosmic Scrat-tastrophe (2015)             5.0
```

```
3              Love and Pigeons (1985)            5.0
4  Red Sorghum (Hong gao liang) (1987)            5.0
```

[14]:
```
movies_rating_count=movies_merged_df.groupby('title')['rating'].count().
 ↪sort_values(ascending=True).reset_index().rename(columns={'rating':
 ↪'Rating␣Count'}) #ascending=False
movies_rating_count_avg=movies_rating_count.
 ↪merge(movies_average_rating,on='title')
movies_rating_count_avg.head()
```

[14]:
```
                                          title  Rating␣Count  \
0                                     '71 (2014)             1
1                                Latter Days (2003)             1
2                           Late Shift, The (1996)             1
3  Late Night with Conan O'Brien: The Best of Tri…             1
4                      Late Night Shopping (2001)             1

   Average Rating
0             4.0
1             3.5
2             2.5
3             2.0
4             4.5
```

[15]:
```
rating_with_RatingCount = movies_merged_df.merge(movies_rating_count, left_on␣
 ↪='title', right_on = 'title', how = 'left')
rating_with_RatingCount.head()
```

[15]:
```
   movieId             title  userId  rating   timestamp  Rating␣Count
0        1  Toy Story (1995)       1     4.0   964982703           215
1        1  Toy Story (1995)       5     4.0   847434962           215
2        1  Toy Story (1995)       7     4.5  1106635946           215
3        1  Toy Story (1995)      15     2.5  1510577970           215
4        1  Toy Story (1995)      17     4.5  1305696483           215
```

[19]:
```
# Erzeugung einer Pivottabelle mit den UserIds und die Movie Bewertungen

import os

movie_features_df = rating_with_RatingCount.pivot_table(index='title',␣
 ↪columns='userId', values='rating').fillna(0)

movie_features_df.head()
```

[19]:
```
userId                              1    2    3    4    5    6    7    \
title
'71 (2014)                        0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
'Hellboy': The Seeds of Creation (2004)  0.0  0.0  0.0  0.0  0.0  0.0  0.0
'Round Midnight (1986)                   0.0  0.0  0.0  0.0  0.0  0.0  0.0
'Salem's Lot (2004)                      0.0  0.0  0.0  0.0  0.0  0.0  0.0
'Til There Was You (1997)                0.0  0.0  0.0  0.0  0.0  0.0  0.0

userId                                   8    9    10   …  601  602  603  \
title                                                  …
'71 (2014)                               0.0  0.0  0.0  …  0.0  0.0  0.0
'Hellboy': The Seeds of Creation (2004)  0.0  0.0  0.0  …  0.0  0.0  0.0
'Round Midnight (1986)                   0.0  0.0  0.0  …  0.0  0.0  0.0
'Salem's Lot (2004)                      0.0  0.0  0.0  …  0.0  0.0  0.0
'Til There Was You (1997)                0.0  0.0  0.0  …  0.0  0.0  0.0

userId                                   604  605  606  607  608  609  610
title
'71 (2014)                               0.0  0.0  0.0  0.0  0.0  0.0  4.0
'Hellboy': The Seeds of Creation (2004)  0.0  0.0  0.0  0.0  0.0  0.0  0.0
'Round Midnight (1986)                   0.0  0.0  0.0  0.0  0.0  0.0  0.0
'Salem's Lot (2004)                      0.0  0.0  0.0  0.0  0.0  0.0  0.0
'Til There Was You (1997)                0.0  0.0  0.0  0.0  0.0  0.0  0.0

[5 rows x 610 columns]
```

[20]:
```
# Sparse Matrix

from scipy.sparse import csr_matrix

movie_features_df_matrix = csr_matrix(movie_features_df.values)
```

[23]:
```
# PR

from sklearn.neighbors import NearestNeighbors
model_knn = NearestNeighbors(metric='cosine', algorithm='brute')
model_knn.fit(movie_features_df_matrix)
```

[23]:
```
NearestNeighbors(algorithm='brute', metric='cosine')
```

[26]:
```
query_index = np.random.choice(movie_features_df.shape[0])
print(query_index)
distances, indices = model_knn.kneighbors(movie_features_df.iloc[query_index,:].
 ↪values.reshape(1, -1), n_neighbors = 6)

for i in range(0, len(distances.flatten())):
    if i == 0:
        print('Recommendations for {0}:\n'.format(movie_features_df.
 ↪index[query_index]))
    else:
```

```
        print('{0}: {1}, with distance of {2}:'.format(i, movie_features_df.
↪index[indices.flatten()[i]], distances.flatten()[i]))
```

```
3414
Recommendations for Ghost Town (2008):

1: Happy-Go-Lucky (2008), with distance of 0.010050535202026367:
2: Jalla! Jalla! (2000), with distance of 0.3481137156486511:
3: Mind Game (2004), with distance of 0.43230122327804565:
4: Merchant of Venice, The (2004), with distance of 0.5430727005004883:
5: Spiderwick Chronicles, The (2008), with distance of 0.5545454621315002:
```

[ ]:
```
a
```