ANALISIS SISTEM REKOMENDASI
LAYANAN MEDIA STREAMING DIGITAL

1. PENDAHULUAN

Streaming merupakan prosedur pengiriman suatu konten baik dalam bentuk suara atau video terkompres via internet yang diputar secara live oleh pengguna. Saat ini telah banyak layanan media streaming digital, seperti layanan media streaming yang menawarkan berbagai acara film, anime, documenter, dan masih banyak lagi. Layanan media streaming digital ini seringkali memberikan rekomendasi berbagai acara sesuai dengan kegemaran masing-masing penggunanya. Pada penelitian ini akan ditentukan sistem rekomendasi dari salah satu layanan media streaming digital.

2. METODE PENELITIAN

Data yang digunakan pada project ini yaitu data sekunder yang bersal dari salah satu layanan media streaming digital. Data tersebut berisikan Show ID, kategori, judul, sutradara, pemeran, negara, waktu rilis, rating, durasi, tipe, rata-rata rating *(averageRating)*, jumlah voting *(numVotes)*, dan deskripsi film. Dalam project ini analisis data dilakukan dengan bantuan *software* Jupyter Notebook dengan Bahasa pemrograman Python. Langkah-langkah yang dilakukan pada project ini yaitu:

1. Melihat lima data teratas dan lima data terbawah.
2. Melihat informasi-informasi terkait data.
3. Melakukan *data cleaning.*
4. Menentukan nilai C yang merupakan rata-rata dari jumlah votes seluruh semesta film.
5. Menentukan nilai m yang merupakan jumlah minimum votes yang diperlukan agar dapat masuk kedalam chart.
6. Menentukan IMDB (Internet Movie Database) dengan formula berikut:

$$Weigthed\ Rating\ (WR) = \left(\frac{v}{v+m}.R\right) + \left(\frac{m}{v+m}.C\right)$$

Dengan:
$v =$ jumlah votes untuk film tersebut
$m =$ jumlah minimum votes yang dibutuhkan supaya dapat masuk dalam chart
$R =$ rata-rata rating dari film tersebut
$C =$ rata-rata jumlah votes dari seluruh semesta film

7. Membangun sistem rekomendasi.

3. HASIL DAN PEMBAHASAN

1. Melihat lima data teratas dan lima data terbawah.

Dalam penelitian ini akan dibangun sebuah sistem rekomendasi dari salah satu layanan media streaming digital. Data yang digunakan adalah data sekunder dari salah satu layanan media streaming digital. Perintah yang digunakan untuk memanggil data adalah sebagai berikut:

```
#memanggil data Netflix

import pandas as pd

import numpy as np

movie_df=pd.read_csv('C:/Python/Netflix_Dataset.csv')

movie_df
```

Setelah data berhasil ditampilkan, maka selanjutnya akan dilihat lima data teratas dengan menggunakan perintah berikut:

```
#melihat 5 data teratas

movie_df.head()
```

diperoleh lima data teratas yaitu:

```
    Show_Id Category  Title           Director
0      s1  TV Show     3%               NaN
1      s2    Movie   7:19  Jorge Michel Grau
2      s3    Movie  23:59       Gilbert Chan
3      s4    Movie      9        Shane Acker
4      s5    Movie     21     Robert Luketic
```

Selain lima data teratas, perlu juga diperhatikan lima data terbawah. Memunculkan lima data terbawah dapat digunakan perintah berikut:

```
#melihat 5 data terbawah

movie_df.tail()
```

diperoleh lima data terbawah yaitu:

```
      Show_Id Category  Title                                    Director
7784   s7783    Movie   Zozo                                     Josef Fares
7785   s7784    Movie   Zubaan                                   Mozez Singh
7786   s7785    Movie   Zulu Man in Japan                        NaN
7787   s7786  TV Show  Zumbo's Just Desserts                     NaN
7788   s7787   Movie  ZZ TOP: THAT LITTLE OL' BAND FROM TEXAS Sam Dunn
```

2. Melihat informasi-informasi terkait data.

Langkah berikutnya adalah melihat informasi-informasi penting terkait data seperti melihat banyak baris dan kolom pada data, melihat banyak elemen pada data, melihat nama-nama kolom pada data, melihat tipe data masing-masing kolom, dan lain sebagainya. Berikut adalah perintah yang digunakan:

```
#melihat banyak baris dan kolom pada data

movie_df.shape

Output:

(7789, 13)

#melihat banyak elemen pada data

movie_df.size

Output:

101257

#melihat nama-nama kolom pada data

movie_df.columns

Output:

Index(['Show_Id', 'Category', 'Title', 'Director', 'Cast', 'Country',
       'Release_Date', 'Rating', 'Duration', 'Type', 'averageRating',
       'numVotes', 'Description'],

#melihat tipe data dari masing-masing kolom

movie_df.dtypes

Output:

Show_Id          object
Category         object
Title            object
Director         object
Cast             object
Country          object
Release_Date     object
Rating           object
Duration         object
Type             object
averageRating   float64
numVotes        float64
Description      object
dtype: object
```

```
#melihat tipe data dan info lainnya pada setiap kolom

movie_df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7789 entries, 0 to 7788
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Show_Id        7789 non-null   object
 1   Category       7789 non-null   object
 2   Title          7789 non-null   object
 3   Director       5401 non-null   object
 4   Cast           7071 non-null   object
 5   Country        7282 non-null   object
 6   Release_Date   7779 non-null   object
 7   Rating         7782 non-null   object
 8   Duration       7789 non-null   object
 9   Type           7789 non-null   object
 10  averageRating  7788 non-null   float64
 11  numVotes       7788 non-null   float64
 12  Description    7789 non-null   object
dtypes: float64(2), object(11)
```

3. Melakukan *data cleaning.*

Data cleaning yang dilakukan pada penelitian ini yaitu mengecek dan menghapus *duplicate record* pada data serta mengecek dan menghapus nilai kosong pada data. Pada data terdapat *duplicate record* atau data yang duplikat seperti pada kolom kategori, data-data tersebut tidak unik sehingga perlu dibersihkan dengan perintah berikut:

```
#memeriksa apakah terdapat duplicate record pada data, jika iya maka dihapus

movie_df[movie_df.duplicated()]

#menghapus data yang duplikat

movie_df.drop_duplicates(inplace=True)

#melihat banyak baris dan kolom pada data setelah duplicate record dihapus

movie_df.shape
```

Selanjutnya diperiksa apakah data memiliki data kosong (Null) atau tidak, jika iya maka data tersebut dihapus, menggunakan perintah:

```
#memeriksa apakah terdapat nilai Null pada data

movie_df.isnull()
```

```
ovie_df.isnull().sum()
```

Keluaran yang dihasilkan yaitu:

```
Show_Id             0
Category            0
Title               0
Director         2388
Cast              718
Country           507
Release_Date       10
Rating              7
Duration            0
Type                0
averageRating       1
numVotes            1
Description         0
dtype: int64
```

Sebelum menghapus nilai Null, nilai tersebut akan diubah terlebih dahulu dengan menggunakan perintah berikut:

```
movie_df['Director']=movie_df['Director'].replace('NaN',np.nan)

movie_df['Director']=movie_df['Director'].astype('str')

movie_df.Director.isnull().sum()



movie_df['Cast']=movie_df['Cast'].replace('NaN',np.nan)

movie_df['Cast']=movie_df['Cast'].astype('str')

movie_df.Cast.isnull().sum()


movie_df['Country']=movie_df['Country'].replace('NaN',np.nan)

movie_df['Country']=movie_df['Country'].astype('str')

movie_df.Country.isnull().sum()


movie_df['Release_Date']=movie_df['Release_Date'].replace('NaN',np.nan)

movie_df['Release_Date']=movie_df['Release_Date'].astype('str')

movie_df.Release_Date.isnull().sum()


movie_df['Rating']=movie_df['Rating'].replace('NaN',np.nan)

movie_df['Rating']=movie_df['Rating'].astype('str')
```

```
movie_df.Rating.isnull().sum()


movie_df['averageRating']=movie_df['averageRating'].replace('NaN',np.nan)

movie_df['averageRating']=movie_df['averageRating'].astype('float64')

movie_df.averageRating.isnull().sum()


#mengubah nilai 'NaN' pada numVotes menjadi np.nan dan cast colomnya menjadi
int64

movie_df['numVotes']=movie_df['numVotes'].replace('NaN',np.nan)

movie_df['numVotes']=movie_df['numVotes'].astype('float64')

movie_df.averageRating.isnull().sum()


#menghapus nilai NaN pada tabel yang berisi NaN

movie_df=movie_df.dropna(subset=['Director',      'Cast',      'Release_Date',
'averageRating', 'numVotes'])
```

untuk memastikan tidak ada lagi nilai Null pada data akan dilakukan pengecekan dengan menggunakan perintah berikut:

print(movie_df.info())

diperoleh keluaran sebagai berikut:

```
#   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Show_Id        7788 non-null   object
 1   Category       7788 non-null   object
 2   Title          7788 non-null   object
 3   Director       7788 non-null   object
 4   Cast           7788 non-null   object
 5   Country        7788 non-null   object
 6   Release_Date   7788 non-null   object
 7   Rating         7788 non-null   object
 8   Duration       7788 non-null   object
 9   Type           7788 non-null   object
 10  averageRating  7788 non-null   float64
 11  numVotes       7788 non-null   float64
 12  Description    7788 non-null   object
```

Hasil keluaran tersebut menunjukkan bahwa sudah tidak ada data Null.

4. Menentukan nilai C yang merupakan rata-rata dari jumlah votes seluruh semesta film.

Nilai C dapat ditentukan dengan menggunakan perintah berikut:

```
#mencari nilai C yang merupakan rata-rata dari averageRating

C= movie_df['averageRating'].mean()

C
```

Diperoleh nilai C sebagai berikut:

```
6.005688238315357
```

Sehingga diperoleh nilai rata-rata jumlah votes selurus semesta film adalah 6,006.

5. Menentukan nilai m yang merupakan jumlah minimum votes yang diperlukan agar dapat masuk kedalam chart.

Akan diambil contoh film dengan numVotes di atas 80% populasi, jadi populasi yang akan diambil hanya sebesar 20%. Digunakan perintah berikut:

```
m=movie_df['numVotes'].quantile(0.8)

m
```

Diperoleh nilai m sebagai berikut:

```
194.0
```

6. Menentukan IMDB (Internet Movie Database)

Selanjutnya ditentukan Internet Movie Database (IMDB) dengan menggunakan perintah berikut:

```
def imd_weighted_rating(df, var=0.8):
    v=df['numVotes']
    R=df['averageRating']
    C=df['averageRating'].mean()
    m=df['numVotes'].quantile(var)
    df['score']=(v/(m+v))*R+(m/(m+v))*C
    return df['score']

imd_weighted_rating(movie_df)

movie_df.head()
```

Menggunakan perintah tersebut diperoleh kolom baru yang berisi score untuk lima data teratas sebagai berikut:

```
    Show_Id Category  Title           Director  score
0      s1  TV Show     3%               NaN   5.643676
1      s2    Movie   7:19  Jorge Michel Grau   6.002822
2      s3    Movie  23:59       Gilbert Chan   6.435161
3      s4    Movie      9        Shane Acker   6.041916
4      s5    Movie     21     Robert Luketic   6.091846
```

7. Membangun sistem rekomendasi

Setelah diperoleh score untuk masing-masing data, maka data diurutkan dari score tertinggi ke score terendah dengan menggunakan perintah berikut:

```
def simple_recommender(df, top=100):
    df=df.loc[df['numVotes']>=m]
    df=df.sort_values(by='score', ascending=False)
    df=df[:top]
    return df
```

Dari task yang sudah dilakukan sebelumnya, dapat dilihat sekarang daftar film telah diurutkan dari score tertinggi ke terendah. Film dengan averageRating yang tinggi tidak selalu mendapat posisi yang lebih tinggi dibanding film dengan averageRating lebih rendah, hal ini disebabkan karena kita juga memperhitungkan faktor banyaknya votes. Sistem rekomendasi ini masih bisa ditingkatkan dengan menambah filter spesifik tentang tipe tayangan. Langkah selanjutnya yang akan dilakukan adalah membuat function untuk melakukan filter berdasarkan Type, dengan menggunakan perintah berikut:

```
df= movie_df. copy()

def user_prefer_recommender(df, ask_Type, top=100):

    #ask_Type = 'all' atau yang lain

    if ask_Type.lower()=='all':

        df=df

    else:

        def filter_Type(x):

            if ask_Type.lower() in str(x).lower():

                return True
```

```
            else:
                return False
        df=df.loc[df['Type'].apply(lambda x: filter_Type(x))]
    df=df.loc[df['numVotes']>=m]
    df=df.sort_values(by='score', ascending=False)
    df=df[:top]
    return df
user_prefer_recommender(df, ask_Type='Dramas')
```

Dari perintah tersebut diperoleh tayangan-tayangan yang direkomendasikan berdasarkan tipe data Dramas yang memiliki jumlah votes dan score tertinggi.