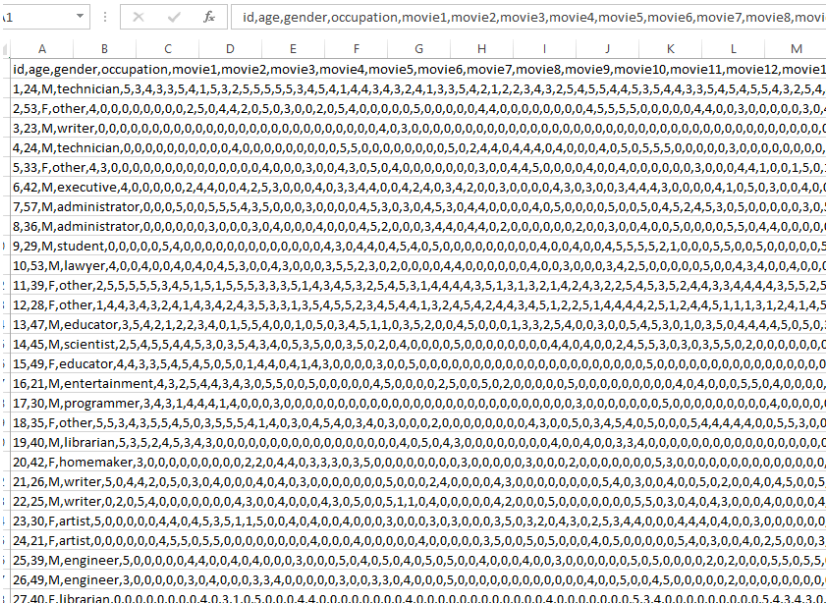
BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Penelitian

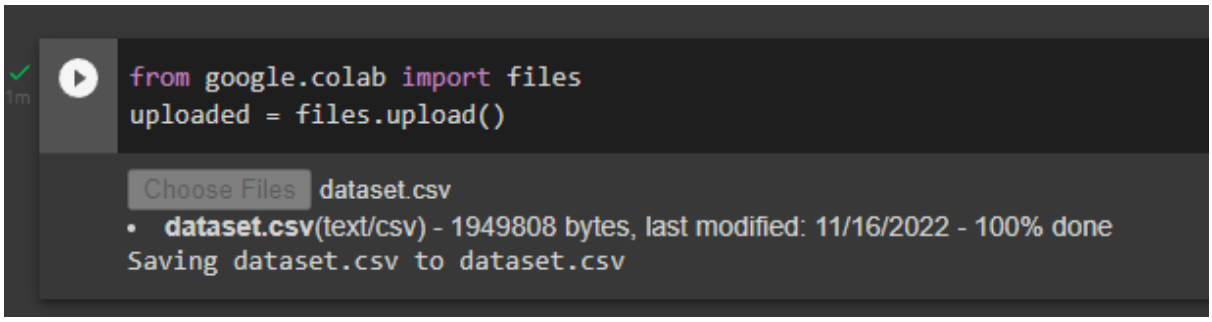
Hasil Penelitian merupakan hasil dari perimputasian dan metode pengerjaan. Berikut adalah hasil dari analisa yang telah dilakukan.

4.1.1 Persiapan Data

Mengubah dataset movielens 100k yang sudah di download menjadi ekstensi .csv (*comma separated values*) lalu mengimportnya ke notebook google colab seperti yang ditunjukkan pada gambar 4.1 dan 4.2.



Gambar 4.1 Data mentah



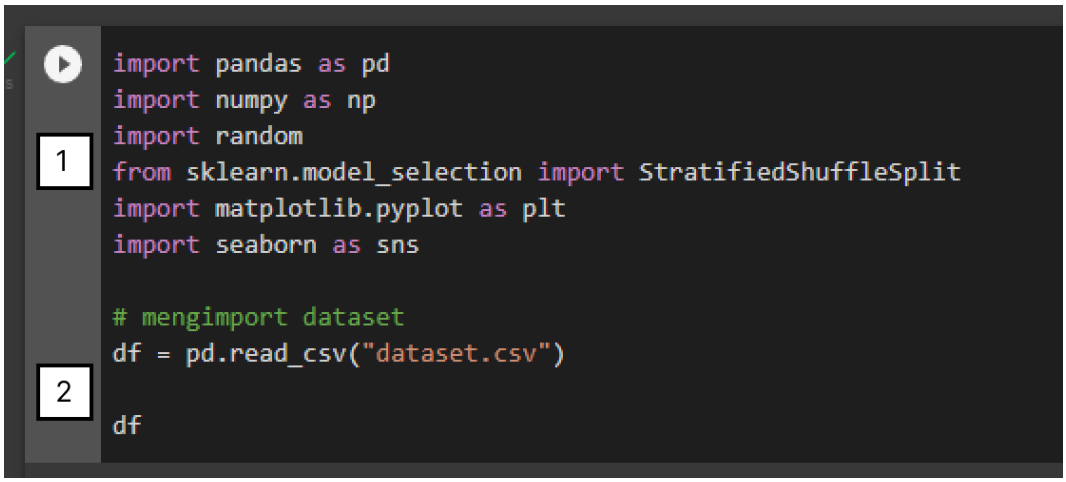
Gambar 4.2 Upload google colab

from google.colab import files : kode ini bertujuan untuk memakai module yang berfungsi untuk mengupload file data mentah ke google colab

uploaded = files.upload() : fungsi untuk memilih file yang ingin di upload dari file manager

4.1.2 Data wrangling

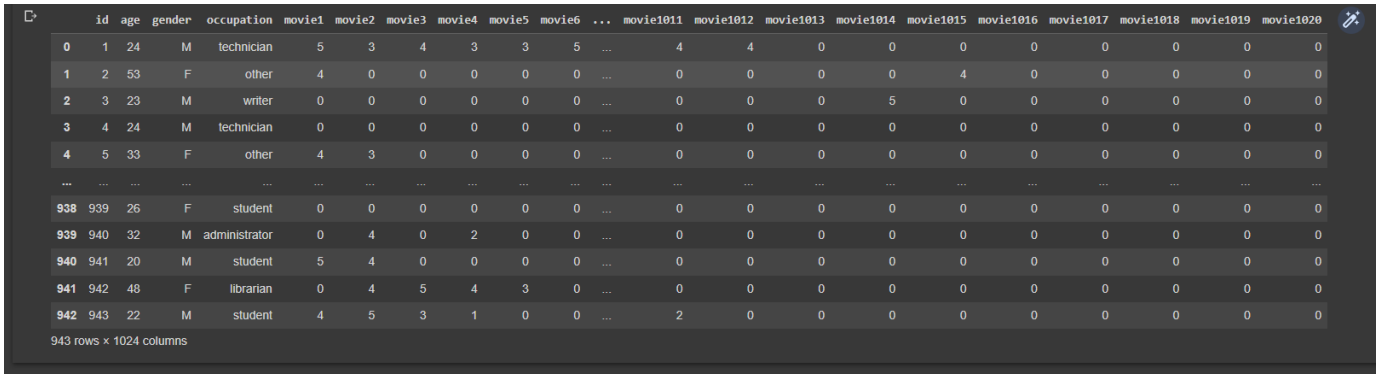
Sebelum Data wrangling harus dilakukan import dependencies yang diperlukan untuk mengimputasian seperti panda, numpy. Lalu memasukkan dataset kedalam variable df seperti pada gambar 4.3.



Gambar 4.3 Import dependency

1) Mengimport library yang dibutuhkan untuk penelitan seperti pandas, numpy, random, dan matplotlib

2) **df = pd.read_csv("dataset.csv")** : membuat variable bernama df lalu menggunakan library pandas (pd) untuk membaca file .csv. Dataset yang sudah masuk ke dalam variable df sudah dapat dipanggil dari dalam google colab seperti pada gambar 4.4.



	id	age	gender	occupation	movie1	movie2	movie3	movie4	movie5	movie6	...	movie1011	movie1012	movie1013	movie1014	movie1015	movie1016	movie1017	movie1018	movie1019	movie1020
0	1	24	M	technician	5	3	4	3	3	5	...	4	4	0	0	0	0	0	0	0	0
1	2	53	F	other	4	0	0	0	0	0	...	0	0	0	0	4	0	0	0	0	0
2	3	23	M	writer	0	0	0	0	0	0	...	0	0	0	5	0	0	0	0	0	0
3	4	24	M	technician	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	5	33	F	other	4	3	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
938	939	26	F	student	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
939	940	32	M	administrator	0	4	0	2	0	0	...	0	0	0	0	0	0	0	0	0	0
940	941	20	M	student	5	4	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
941	942	48	F	librarian	0	4	5	4	3	0	...	0	0	0	0	0	0	0	0	0	0
942	943	22	M	student	4	5	3	1	0	0	...	2	0	0	0	0	0	0	0	0	0

943 rows × 1024 columns

Gambar 4.4 Dataset pada variable df

Setelah itu memulai tahap cleaning dengan melakukan systematic sampling pada dataset dengan memfilter data, data yang diambil di mulai dari user 5 dan lanjut dengan interval 5, Dengan movies yang dipilih mulai dari movies 1 hingga movies 100. Ditunjukkan pada gambar 4.5 dan 4.6.

```
start_row = 4
interval_row = 5

n_rows = df.shape[0]

n_sample_row = min(500, n_rows) // interval_row

sampled_rows = []
sampled_cols = []

sampled_cols.extend(list(range(4)))

for i in range(n_sample_row):
    row_index = start_row + i*interval_row
    sampled_rows.append(row_index)

sampled_df = df.iloc[sampled_rows, :104]
sampled_df = sampled_df.reset_index(drop=True)

sampled_df
```

1 2 3 4 5

Gambar 4.5 Systematic sampling

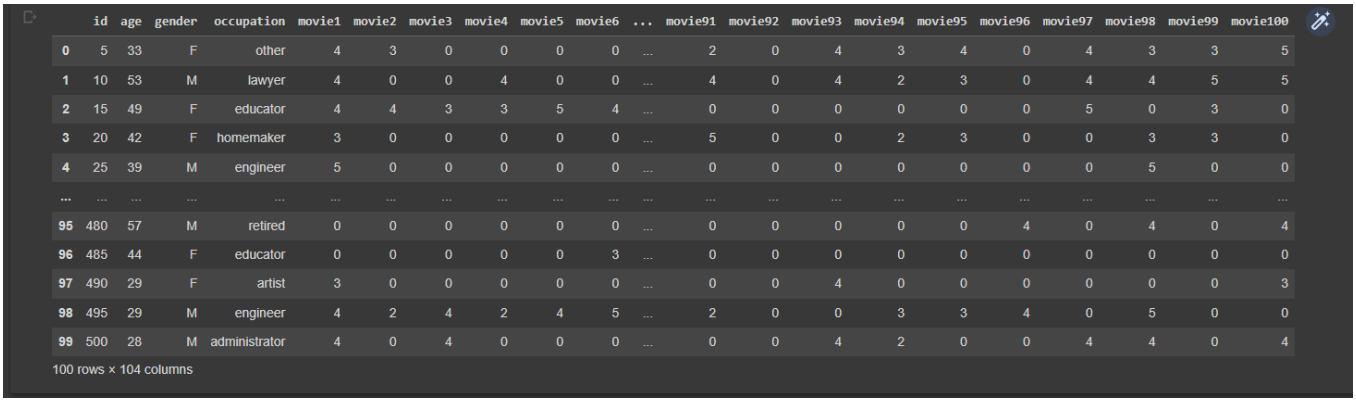
1) **Start_row = 4** bertujuan untuk memulai sampling pada baris ke 4, sehingga proses sampling dimulai dari user ke 5 dikarenakan kolom 0-3 berisikan kolom identitas user seperti id, umur, pekerjaan. Sehingga dilewati. **Interval_row = 5** menjelaskan bahwa baris akan melewati 5 dari setiap baris yang di sampel.

2) **n_rows = df.shape[0]** bertujuan untuk menghitung seluruh baris yang digunakan lalu memasukannya kedalam n_rows. **n_sample_row = min(500, n_rows) // interval_row** untuk membuat maksimum baris hingga 500 lalu dibagi oleh interval_row.

3) **sampled_rows = []** kode ini membuat array kosong untuk diisi oleh baris.**sampled_cols = []** juga membuat array kosong untuk diisi oleh kolom.**sampled_cols.extend(list(range(4)))** untuk melewati kolom 0-3 untuk tidak di sampling, dikarenakan kolom 0-3 berisikan kolom identitas user seperti id, umur, pekerjaan. Sehingga dilewati. **Interval_row = 5** menjelaskan bahwa baris akan melewati 5 dari setiap baris yang di sampel.

4) Langkah ini memulai perulangan dari semua kode diatas berdasarkan start_row dan interval_row, lalu mengatur ulang index agar diulang dari 0 .

5) Hasil sampling lalu disimpan ke dalam variable sampled_df DataFrame baru ini mencakup setiap kelima baris dimulai dari baris keempat, hingga maksimal 500 baris, dan 104 kolom pertama dari DataFrame asli.

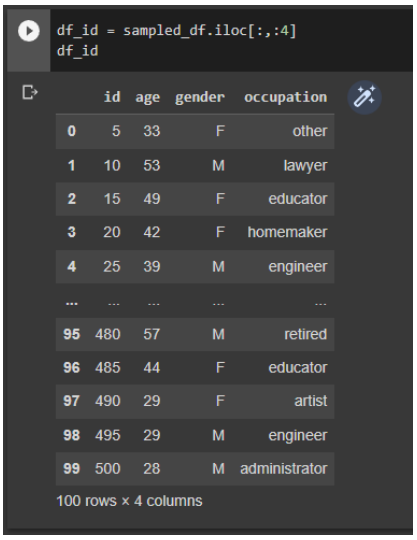


	id	age	gender	occupation	movie1	movie2	movie3	movie4	movie5	movie6	...	movie91	movie92	movie93	movie94	movie95	movie96	movie97	movie98	movie99	movie100
0	5	33	F	other	4	3	0	0	0	0	...	2	0	4	3	4	0	4	3	3	5
1	10	53	M	lawyer	4	0	0	4	0	0	...	4	0	4	2	3	0	4	4	5	5
2	15	49	F	educator	4	4	3	3	5	4	...	0	0	0	0	0	0	5	0	3	0
3	20	42	F	homemaker	3	0	0	0	0	0	...	5	0	0	2	3	0	0	3	3	0
4	25	39	M	engineer	5	0	0	0	0	0	...	0	0	0	0	0	0	0	5	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	480	57	M	retired	0	0	0	0	0	0	...	0	0	0	0	0	4	0	4	0	4
96	485	44	F	educator	0	0	0	0	0	3	...	0	0	0	0	0	0	0	0	0	0
97	490	29	F	artist	3	0	0	0	0	0	...	0	0	4	0	0	0	0	0	0	3
98	495	29	M	engineer	4	2	4	2	4	5	...	2	0	0	3	3	4	0	5	0	0
99	500	28	M	administrator	4	0	4	0	0	0	...	0	0	4	2	0	0	4	4	0	4

100 rows × 104 columns

Gambar 4.6 Hasil sampling

Selanjutnya memasukkan kolom kolom user seperti id, age, gender, dan occupation kedalam variable df_id. Lalu membuat variable df_gender berisikan kolom gender yang nanti akan digunakan pada imputasi hotdeck, dan variable df_movie berisi seluruh kolom movie agar imputasi tidak mencakup keseluruhan data melainkan hanya data yang memiliki missing value. Dapat dilihat pada gambar 4.7, 4.8, 4.9 .



```
df_id = sampled_df.iloc[:,:4]
df_id
```

	id	age	gender	occupation
0	5	33	F	other
1	10	53	M	lawyer
2	15	49	F	educator
3	20	42	F	homemaker
4	25	39	M	engineer
...	...	...	...	...
95	480	57	M	retired
96	485	44	F	educator
97	490	29	F	artist
98	495	29	M	engineer
99	500	28	M	administrator

100 rows × 4 columns

Gambar 4.7 df_id

```
df_gender = df_id.iloc[:,2]
df_gender
```

```
0     F
1     M
2     F
3     F
4     M
     ..
95    M
96    F
97    F
98    M
99    M
Name: gender, Length: 100, dtype: object
```

Gambar 4.8 df_gender

```
df_movie = sampled_df.iloc[:, 4:]
df_movie
```

	movie1	movie2	movie3	movie4	movie5	movie6	movie7	movie8	movie9	movie10	...	movie91	movie92	movie93	movie94	movie95	movie96	movie97	movie98	movie99	movie100
0	4	3	0	0	0	0	0	0	0	0	...	2	0	4	3	4	0	4	3	3	5
1	4	0	0	4	0	0	4	0	4	0	...	4	0	4	2	3	0	4	4	5	5
2	4	4	3	3	5	4	5	4	5	0	...	0	0	0	0	0	0	5	0	3	0
3	3	0	0	0	0	0	0	0	0	0	...	5	0	0	2	3	0	0	3	3	0
4	5	0	0	0	0	0	4	4	0	0	...	0	0	0	0	0	0	0	5	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	0	0	0	0	0	0	0	5	0	0	...	0	0	0	0	0	4	0	4	0	4
96	0	0	0	0	0	3	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
97	3	0	0	0	0	0	3	0	4	0	...	0	0	4	0	0	0	0	0	0	3
98	4	2	4	2	4	5	4	0	5	0	...	2	0	0	3	3	4	0	5	0	0
99	4	0	4	0	0	0	5	4	4	3	...	0	0	4	2	0	0	4	4	0	4

100 rows × 100 columns

Gambar 4.9 df_movie

4.1.3 Transforming data

Setelah cleaning data, tahap selanjutnya yang dilakukan adalah transformasi data. Pada dataset, data yang hilang (*missing value*) masih menjadi interger yaitu nilai 0, sehingga tipe data belum bisa untuk diimputasi. Maka harus dilakukan perubahan data menjadi data kosong (NaN) agar dapat diidentifikasi bahwa data tersebut data yang hilang. Dapat dilihat pada gambar 4.10.

```
df_NaN = df_movie.replace(0, np.NaN)
df_NaN
```

	movie1	movie2	movie3	movie4	movie5	movie6	movie7	movie8	movie9	movie10	...	movie91	movie92	movie93	movie94	movie95	movie96	movie97	movie98	movie99	movie100
0	4.0	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2.0	NaN	4.0	3.0	4.0	NaN	4.0	3.0	3.0	5.0
1	4.0	NaN	NaN	4.0	NaN	NaN	4.0	NaN	4.0	NaN	...	4.0	NaN	4.0	2.0	3.0	NaN	4.0	4.0	5.0	5.0
2	4.0	4.0	3.0	3.0	5.0	4.0	5.0	4.0	5.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	3.0	NaN
3	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	5.0	NaN	NaN	2.0	3.0	NaN	NaN	3.0	3.0	NaN
4	5.0	NaN	NaN	NaN	NaN	NaN	4.0	4.0	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	4.0	NaN	4.0	NaN	4.0
96	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
97	3.0	NaN	NaN	NaN	NaN	NaN	3.0	NaN	4.0	NaN	...	NaN	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	3.0
98	4.0	2.0	4.0	2.0	4.0	5.0	4.0	NaN	5.0	NaN	...	2.0	NaN	NaN	3.0	3.0	4.0	NaN	5.0	NaN	NaN
99	4.0	NaN	4.0	NaN	NaN	NaN	5.0	4.0	4.0	3.0	...	NaN	NaN	4.0	2.0	NaN	NaN	4.0	4.0	NaN	4.0

100 rows × 100 columns

Gambar 4.10 transformasi data

4.1.4 Imputasi data

Data yang sudah melalui tahap cleaning, preprocessing dan transforming menandakan data sudah siap untuk di olah, selanjutnya yang harus dilakukan adalah melakukan imputasi, metode yang digunakan merupakan metode mean dan hotdeck. Untuk imputasi mean data kosong akan diimputasi oleh keseluruhan rata-rata dari data yang ada pada masing masing kolom, lalu hasil dari imputasi akan dibulatkan mendekati angka terdekat. Ditunjukkan pada gambar 4.11 dan 4.12.

```
df_mean = df_NaN.fillna(df_NaN.mean())
df_mean
```

	movie1	movie2	movie3	movie4	movie5	movie6	movie7	movie8	movie9	movie10	...	movie91	movie92	movie93	movie94	movie95	movie96	movie97	movie98	movie99	movie100
0	4.000000	3.000000	3.26087	3.333333	3.4	4.28	3.688889	3.90625	3.685714	4.25	...	2.000000	3.9	4.000000	3.000000	4.000000	3.53125	4.00000	3.000000	3.00	5.000000
1	4.000000	3.185185	3.26087	4.000000	3.4	4.28	4.000000	3.90625	4.000000	4.25	...	4.000000	3.9	4.000000	2.000000	3.000000	3.53125	4.00000	4.000000	5.00	5.000000
2	4.000000	4.000000	3.00000	3.000000	5.0	4.00	5.000000	4.00000	5.000000	4.25	...	3.482759	3.9	3.894737	3.454545	3.689655	3.53125	5.00000	4.392157	3.00	4.075472
3	3.000000	3.185185	3.26087	3.333333	3.4	4.28	3.688889	3.90625	3.685714	4.25	...	5.000000	3.9	3.894737	2.000000	3.000000	3.53125	3.72973	3.000000	3.00	4.075472
4	5.000000	3.185185	3.26087	3.333333	3.4	4.28	4.000000	4.00000	3.685714	4.25	...	3.482759	3.9	3.894737	3.454545	3.689655	3.53125	3.72973	5.000000	3.72	4.075472
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	3.821429	3.185185	3.26087	3.333333	3.4	4.28	3.688889	5.00000	3.685714	4.25	...	3.482759	3.9	3.894737	3.454545	3.689655	4.00000	3.72973	4.000000	3.72	4.000000
96	3.821429	3.185185	3.26087	3.333333	3.4	3.00	3.688889	3.90625	3.685714	4.25	...	3.482759	3.9	3.894737	3.454545	3.689655	3.53125	3.72973	4.392157	3.72	4.075472
97	3.000000	3.185185	3.26087	3.333333	3.4	4.28	3.000000	3.90625	4.000000	4.25	...	3.482759	3.9	4.000000	3.454545	3.689655	3.53125	3.72973	4.392157	3.72	3.000000
98	4.000000	2.000000	4.00000	2.000000	4.0	5.00	4.000000	3.90625	5.000000	4.25	...	2.000000	3.9	3.894737	3.000000	3.000000	4.00000	3.72973	5.000000	3.72	4.075472
99	4.000000	3.185185	4.00000	3.333333	3.4	4.28	5.000000	4.00000	4.000000	3.00	...	3.482759	3.9	4.000000	2.000000	3.689655	3.53125	4.00000	4.000000	3.72	4.000000

100 rows × 100 columns

Gambar 4.11 imputasi mean

```
df_mean = pd.concat([df_id,df_rounded], axis=1)
df_mean
```

	id	age	gender	occupation	movie1	movie2	movie3	movie4	movie5	movie6	...	movie91	movie92	movie93	movie94	movie95	movie96	movie97	movie98	movie99	movie100
0	5	33	F	other	4.0	3.0	3.0	3.0	3.0	4.0	...	2.0	4.0	4.0	3.0	4.0	4.0	4.0	3.0	3.0	5.0
1	10	53	M	lawyer	4.0	3.0	3.0	4.0	3.0	4.0	...	4.0	4.0	4.0	2.0	3.0	4.0	4.0	4.0	5.0	5.0
2	15	49	F	educator	4.0	4.0	3.0	3.0	5.0	4.0	...	3.0	4.0	4.0	3.0	4.0	4.0	5.0	4.0	3.0	4.0
3	20	42	F	homemaker	3.0	3.0	3.0	3.0	3.0	4.0	...	5.0	4.0	4.0	2.0	3.0	4.0	4.0	3.0	3.0	4.0
4	25	39	M	engineer	5.0	3.0	3.0	3.0	3.0	4.0	...	3.0	4.0	4.0	3.0	4.0	4.0	4.0	5.0	4.0	4.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	480	57	M	retired	4.0	3.0	3.0	3.0	3.0	4.0	...	3.0	4.0	4.0	3.0	4.0	4.0	4.0	4.0	4.0	4.0
96	485	44	F	educator	4.0	3.0	3.0	3.0	3.0	3.0	...	3.0	4.0	4.0	3.0	4.0	4.0	4.0	4.0	4.0	4.0
97	490	29	F	artist	3.0	3.0	3.0	3.0	3.0	4.0	...	3.0	4.0	4.0	3.0	4.0	4.0	4.0	4.0	4.0	3.0
98	495	29	M	engineer	4.0	2.0	4.0	2.0	4.0	5.0	...	2.0	4.0	4.0	3.0	3.0	4.0	4.0	5.0	4.0	4.0
99	500	28	M	administrator	4.0	3.0	4.0	3.0	3.0	4.0	...	3.0	4.0	4.0	2.0	4.0	4.0	4.0	4.0	4.0	4.0

100 rows × 104 columns

Gambar 4.12 pembulatan hasil mean

Pada metode imputasi stokastik hotdeck, Penelitian ini menggunakan jenis kelamin sebagai observasi, data yang hilang pada suatu variabel pada baris data yang sama akan dicocokkan dengan data yang tersedia pada baris data lain yang memiliki nilai jenis kelamin yang sama. Sehingga, nilai yang hilang pada variabel yang dicari akan diisi dengan nilai yang sama pada variabel tersebut pada baris data yang cocok.

Data yang memiliki nilai terlebih dahulu masuk ke matriks kosong, lalu data yang masih hilang akan terisi dengan nilai dengan pembagian secara random.

Pengimputasian hotdeck dilakukan dua kali dengan jenis kelamin yang berbeda agar nilai dengan jenis kelamin laki laki maupun perempuan yang terproses tidak menyatu, lalu setelah data terimputasi maka keduanya akan digabungkan, seperti yang dilihat pada gambar 4.13.

```
1
def impute_missing_values(df_hotdeckn, variable, donor_pool, method='hotdeck'):
    for i in range(len(df_hotdeckn)):
        if df_hotdeckn.at[i, 'gender'] == 'F' and np.isnan(df_hotdeckn.at[i, variable]):
            donor = donor_pool[donor_pool[variable].notnull()].sample(n=1)
            df_hotdeckn.at[i, variable] = donor[variable].values[0]
    return df_hotdeckn

2
for column in df_hotdeckn.columns[1:]:
    donor_pool = df_hotdeckn[df_hotdeckn['gender'] == 'F']
    df_hotimput = impute_missing_values(df_hotdeckn, column, donor_pool, method='hotdeck')

df_hotimput
```

Gambar 4.13 imputasi hotdeck

1) **if df_hotdeckn.at[i, 'gender'] == 'F' and np.isnan(df_hotdeckn.at[i, variable]):: ** Baris ini mengecek apakah nilai kolom "gender" sama dengan "F" dan nilai kolom "variable" di baris tersebut bernilai NaN (missing value).

   donor = donor_pool[donor_pool[variable].notnull()].sample(n=1): Baris ini memilih secara acak satu data dari donor pool yang memiliki nilai yang tidak hilang (not null) pada kolom yang sama dengan "variable" yang sedang diimputasi.

   df_hotdeckn.at[i, variable] = donor[variable].values[0]: Baris ini mengganti nilai yang hilang pada kolom "variable" di baris tersebut dengan nilai yang telah dipilih dari donor pool.

   return df_hotdeckn: Baris ini mengembalikan DataFrame dengan nilai yang hilang pada kolom "variable" telah diimputasi dengan metode hot deck imputation.

2) Hasil imputasi untuk setiap kolom disimpan dalam DataFrame baru yang diberi nama **df_hotimput**. Setelah selesai melakukan iterasi pada semua kolom, DataFrame **df_hotimput** akan ditampilkan pada output akhir, dapat dilihat pada gambar 4.14.

	gender	movie1	movie2	movie3	movie4	movie5	movie6	movie7	movie8	movie9	...	movie91	movie92	movie93	movie94	movie95	movie96	movie97	movie98	movie99	movie100
0	F	4.0	3.0	3.0	3.0	4.0	3.0	3.0	5.0	5.0	...	2.0	3.0	4.0	3.0	4.0	4.0	4.0	3.0	3.0	5.0
1	M	4.0	5.0	3.0	4.0	4.0	4.0	4.0	3.0	4.0	...	4.0	5.0	4.0	2.0	3.0	4.0	4.0	4.0	5.0	5.0
2	F	4.0	4.0	3.0	3.0	5.0	4.0	5.0	4.0	5.0	...	2.0	3.0	4.0	2.0	3.0	3.0	5.0	5.0	3.0	4.0
3	F	3.0	3.0	2.0	3.0	4.0	4.0	4.0	5.0	4.0	...	5.0	4.0	4.0	2.0	3.0	4.0	5.0	3.0	3.0	2.0
4	M	5.0	2.0	4.0	2.0	2.0	4.0	4.0	4.0	4.0	...	3.0	4.0	5.0	4.0	2.0	4.0	5.0	5.0	4.0	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	M	3.0	4.0	4.0	3.0	4.0	5.0	4.0	5.0	3.0	...	5.0	5.0	4.0	3.0	3.0	4.0	3.0	4.0	3.0	4.0
96	F	4.0	5.0	4.0	3.0	4.0	3.0	3.0	4.0	4.0	...	3.0	4.0	4.0	2.0	5.0	4.0	5.0	4.0	3.0	5.0
97	F	3.0	5.0	3.0	3.0	3.0	4.0	3.0	5.0	4.0	...	2.0	4.0	4.0	2.0	5.0	3.0	5.0	3.0	3.0	3.0
98	M	4.0	2.0	4.0	2.0	4.0	5.0	4.0	3.0	5.0	...	2.0	5.0	3.0	3.0	3.0	4.0	3.0	5.0	1.0	3.0
99	M	4.0	4.0	4.0	2.0	4.0	2.0	5.0	4.0	4.0	...	2.0	5.0	4.0	2.0	2.0	3.0	4.0	4.0	5.0	4.0

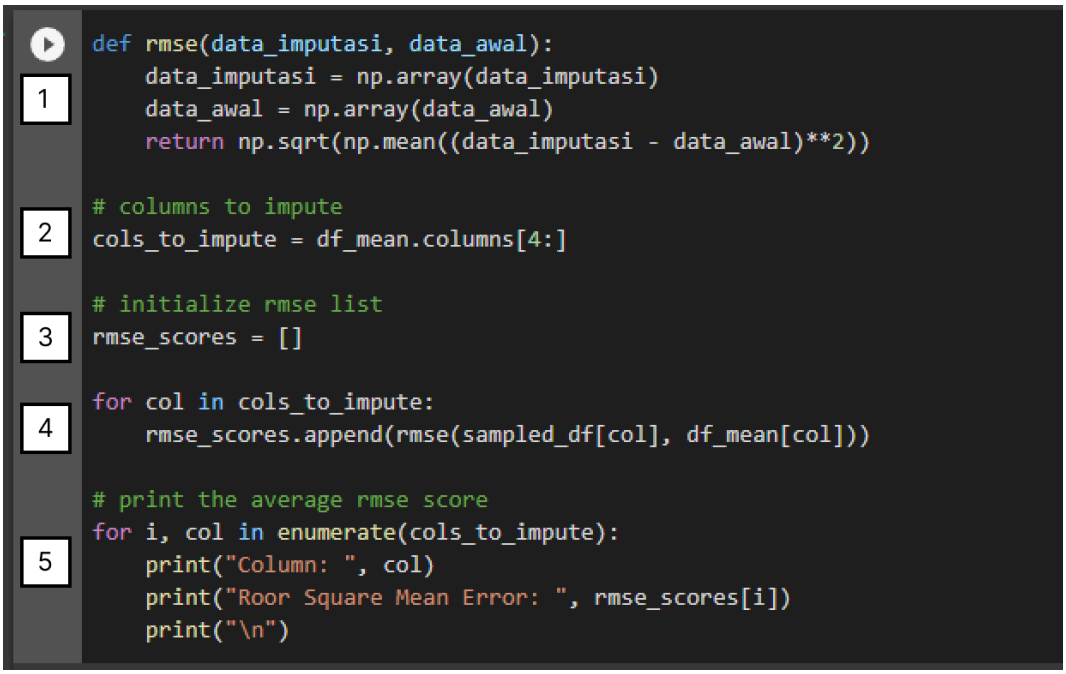
100 rows × 101 columns

Gambar 4.14 hasil imputasi hotdeck

4.2 Analisa Hasil Penelitian

Setelah imputasi dengan kedua metode sudah dilakukan, maka selanjutnya diberikan perbandingan antara metode imputasi mean dan hotdeck dengan evaluasi menggunakan RMSE dan MAE. RMSE dan MAE adalah dua metrik evaluasi yang digunakan untuk mengukur seberapa dekat data imputasi dengan data yang hilang asli. Dalam analisis ini, kinerja kedua metode diukur dengan cara menghitung RMSE dan MAE dari data imputasi yang dihasilkan.

RMSE mengukur akar rata-rata dari selisih kuadrat antara data imputasi dan data hilang asli.




```
def rmse(data_imputasi, data_awal):
    data_imputasi = np.array(data_imputasi)
    data_awal = np.array(data_awal)
    return np.sqrt(np.mean((data_imputasi - data_awal)**2))

# columns to impute
cols_to_impute = df_mean.columns[4:]

# initialize rmse list
rmse_scores = []

for col in cols_to_impute:
    rmse_scores.append(rmse(sampled_df[col], df_mean[col]))

# print the average rmse score
for i, col in enumerate(cols_to_impute):
    print("Column: ", col)
    print("Roor Square Mean Error: ", rmse_scores[i])
    print("\n")
```

Gambar 4.15 RMSE

1) Kode pada gambar 4.15 diatas bertujuan untuk menghitung nilai Root Mean Square Error (RMSE) untuk setiap kolom yang diimputasi menggunakan metode mean, dan memplot hasil RMSE untuk setiap kolom tersebut.

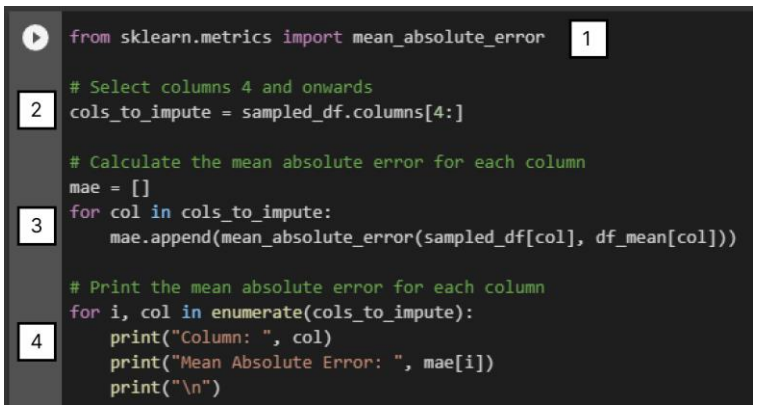
2) Kolom yang akan diimputasi diambil dari DataFrame **df_mean** dimulai dari indeks kolom ke-4 (kolom ke-0 sampai ke-3 kolom yang tidak perlu diimputasi).

3) Selanjutnya, dilakukan perulangan untuk menghitung RMSE pada setiap kolom yang diimputasi. Dalam perulangan tersebut, RMSE dihitung sebagai akar kuadrat dari mean squared error antara nilai imputasi pada kolom yang sedang diiterasi di DataFrame **df_mean** dan nilai yang ada pada kolom yang sama pada DataFrame **sampled_df** (yang merupakan subset dari df yang sudah diambil sebagai data sampel).

4) Hasil RMSE untuk setiap kolom disimpan dalam variabel rmse_list.

5) Pada baris terakhir, hasil RMSE untuk setiap kolom diplot ke dalam sebuah grafik menggunakan library **matplotlib**. Pada sumbu x, akan ditampilkan label nama kolom yang diimputasi setiap dua kolom sekali, sementara pada sumbu y akan ditampilkan nilai RMSE. Titik-titik pada grafik menunjukkan besarnya RMSE untuk setiap kolom yang diimputasi.

Sedangkan MAE mengukur rata-rata selisih absolut antara data imputasi dan data hilang asli, dapat dilihat pada gambar 4.16 dibawah ini.



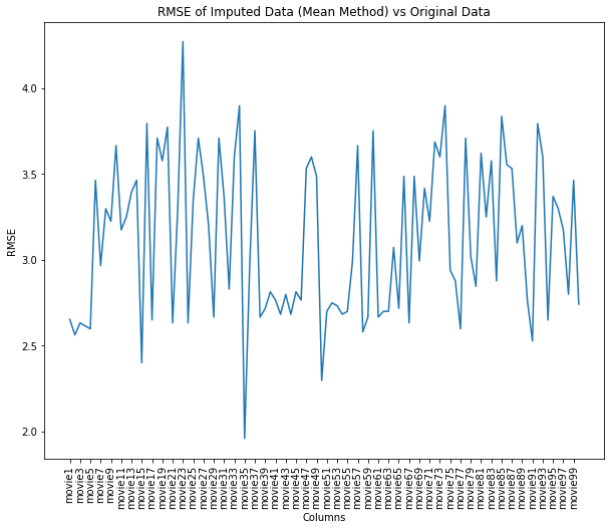
Gambar 4.16 MAE

1) Pada baris pertama, diimpor library **mean_absolute_error** dari **sklearn.metrics**
2) Selanjutnya, pada baris kedua, kolom yang akan diimputasi diambil dari DataFrame **sampled_df** dimulai dari indeks kolom ke-4 (kolom ke-0 sampai ke-3 sebagai kolom yang tidak perlu diimputasi).

3) Selanjutnya, pada baris ketiga, dilakukan perulangan untuk menghitung nilai MAE pada setiap kolom yang diimputasi. Dalam perulangan tersebut, nilai MAE dihitung dengan membandingkan nilai kolom pada DataFrame **sampled_df** (yang merupakan subset dari df yang sudah diambil sebagai data sampel) dengan nilai imputasi pada kolom yang sama di DataFrame **df_mean.** Semakin kecil RMSE atau MAE, semakin baik kinerja metode imputasi.

4.2.1 RMSE (*Root Mean Squared Error*)

Berikut ini adalah hasil RMSE dari kedua metode mean di gambar 4.17 dan 4.18 lalu hotdeck di gambar 4.19 dan 4.20.

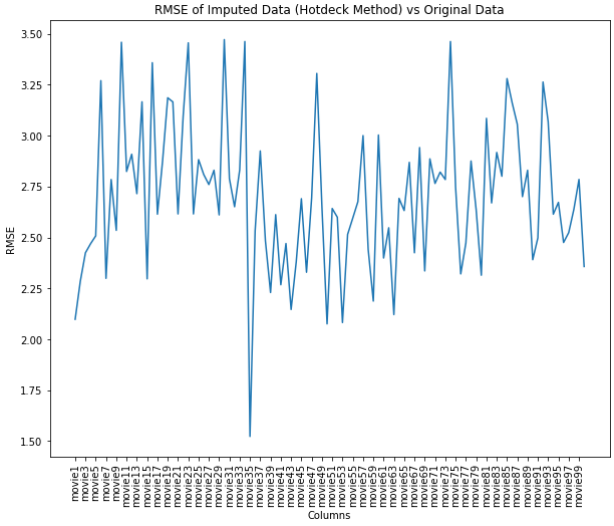


Gambar 4.17 RMSE imputasi mean

```
# Hasil keseluruhan rmse
print("RMSE:", np.mean(rmse_scores))

RMSE: 3.120696653236777
```

Gambar 4.18 Hasil RMSE imputasi mean

Sehingga, diketahui bahwa hasil RMSE (*Root Mean Square Error)* dari imputasi mean pada penelitian ini adalah : 3.120



Gambar 4.19 RMSE imputasi hotdeck

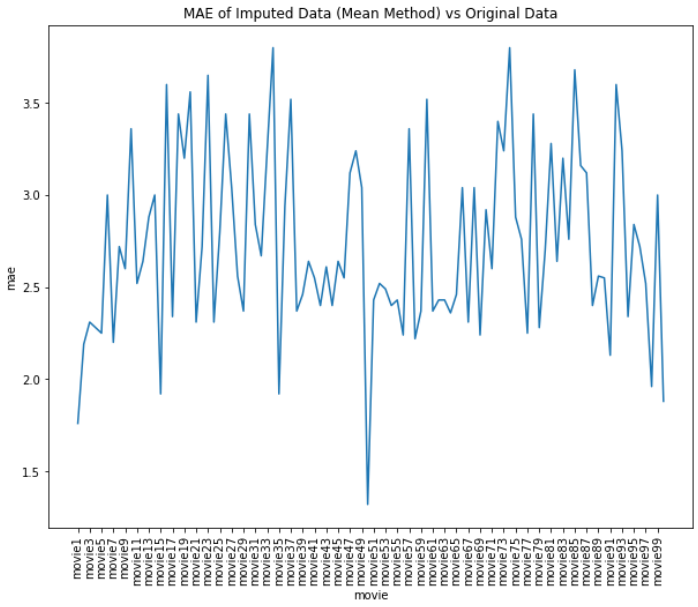
```
print("Hasil RMSE:", np.mean(rmse_scores))

Hasil RMSE: 2.7064309707581113
```

Gambar 4.20 Hasil RMSE imputasi hotdeck

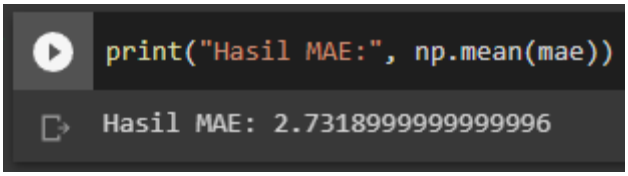
Sehingga, diketahui bahwa hasil RMSE (*Root Mean Squared Error)* dari imputasi hotdeck pada penelitian ini adalah : 2.706

4.2.2 MAE (*Mean Absolute Error*)

Berikut ini adalah hasil MAE dari kedua metode mean di gambar 4.21 dan 4.22 lalu hotdeck di gambar 4.23 dan 4.24.

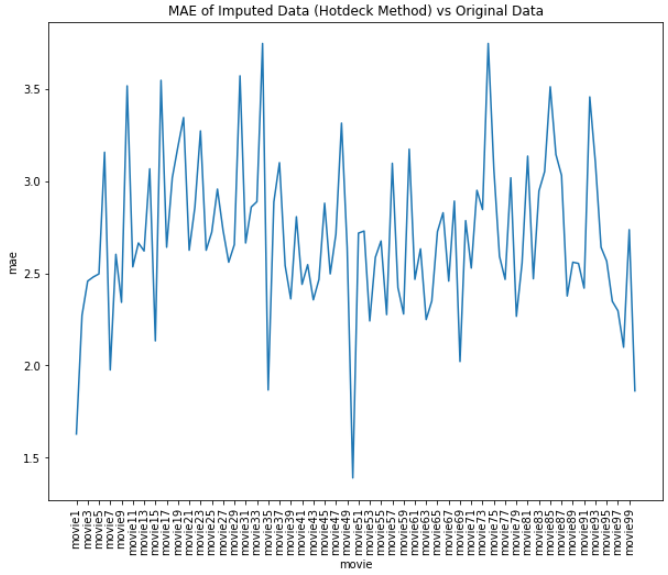


Gambar 4.21 MAE imputasi mean



Gambar 4.22 Hasil MAE imputasi mean

Sehingga, diketahui bahwa hasil MAE (*Mean Absolute error)* dari imputasi mean pada penelitian ini adalah : 2.731



Gambar 4.23 MAE imputasi hotdeck

```
print("Hasil MAE:", np.mean(mae))

Hasil MAE: 2.691477056634417
```

Gambar 4.23 Hasil MAE imputasi hotdeck

Sehingga, diketahui bahwa hasil MAE (*Mean Absolute error)* dari imputasi mean pada penelitian ini adalah : 2.691

Tabel 4.1 Hasil imputasi

Imputasi	Mean	Hotdeck
RMSE	3.120	2.706
MAE	2.731	2.691

Dari hasil analisis menggunakan RMSE dan MAE, metode imputasi hotdeck memberikan hasil yang lebih baik daripada metode imputasi mean pada dataset yang digunakan. Karena, semakin kecil RMSE atau MAE, semakin baik kinerja metode imputasi.