

Vivian Joseph

20BCE0777

Machine Learning LAB DA-1

Pre-processing of Data

Q1) Create your own dataset - 9 column/ features and minimum 30 samples/rows.

movieID	name	year	Director	budget	marketing	boxOffice	studio	genre	imdb
1	Endgame	2018	Russo Brothers	350	150	3000	Marvel	team	7.6
2	Infinity War	2017	Russo Brothers	250	170	2400	Marvel	team	8.1
3	Iron Man	2008	Jon Favreau	150	100	760	Marvel	Sci-fi	8.1
4	Iron Man 2	2010	Jon Favreau	151	101	861	Marvel	Sci-fi	7.1
5	Iron Man 3	2012	Jon Favreau	200	120	862	Marvel	Sci-fi	6.8
6	Captain America	2011	Joe Johnston	120	50	640	Marvel	Action	7.1
7	Winter Soldier	2014	Russo Brothers	170	120	940	Marvel	Political Thriller	8.7
8	Civil War	2016	Russo Brothers	270	200	1840	Marvel	team	8.7
9	Thor	2010	Kenneth Branagh	100	80	760	Marvel	Mythological	7.6
10	Dark World	2013	Kenneth Branagh	201	120	1200	Marvel	Mythological	6.4
11	Ragnarok	2018	Taika Wititi	180	121	976	Marvel	Mythological	7.9
12	Love and thunder	2022	Taika Wititi	181	122	1077	Marvel	Mythological	6.2
13	Homecoming	2017	Jon Watts	220	100	970	Sony	Teen comedy	7.6
14	Far from home	2019	Jon Watts	240	120	1120	Sony	Teen comedy	7.4
15	No way home	2021	Jon Watts	260	140	2170	Sony	Teen comedy	7.9
16	Venom	2018		170	-23		Sony	Alien	6.9
17	X-men	2000	Simon Kinberg	94		560	Fox	team	7.1
18	Deadpool	2016	Tim Miller	120	70	720	Fox	Adult comedy	7.9
19	Deadpool 2	2019	Tim Miller	200	71	1024	Fox	Adult comedy	7.2
20	Woolverine	2004	James Mangold	140		640	Fox	Action origin	11
21	GOTG	2014	James Gunn	200	40	870	Marvel	Space Comedy	7.8
22	GOTG Vol. 2	2017	James Gunn	250	60	990	Marvel	Space Comedy	7.2
23	GOTG Vol. 3	2023	James Gunn	270			Marvel	Space Comedy	
24	Antman	2015	Peyton Reed	180	50	870	Marvel	Action Comedy	7.1
25	Antman and Wasp	2019	Peyton Reed	200	60	945	Marvel	Action Comedy	7
26	A&W: QuantamMania	2023	Peyton Reed	250			Marvel	Action Comedy	
27	Dr. Strange	2014	Scott Derickson	230	35	780	Marvel	Magic	7

Importing the data into the notebook

```
import numpy as np
import pandas as pd
✓ 6.8s

df=pd.read_csv("vvn-movies.csv")
✓ 0.1s

df.head()
✓ 0.2s
```

	movieID	name	year	Director	budget	marketing	boxOffice	studio	genre	imdb
0	1	Endgame	2018	Russo Brothers	350	150.0	3000.0	Marvel	team	7.6
1	2	Infinity War	2017	Russo Brothers	250	170.0	2400.0	Marvel	team	8.1
2	3	Iron Man	2008	Jon Favreau	150	100.0	760.0	Marvel	Sci-fi	8.1
3	4	Iron Man 2	2010	Jon Favreau	151	101.0	861.0	Marvel	Sci-fi	7.1
4	5	Iron Man 3	2012	Jon Favreau	200	120.0	862.0	Marvel	Sci-fi	6.8

Q2) Apply Data Manipulation Operations like:

a. Insert new sample

to insert new row:

```
df=df.append({'movieID':30,'name':'Black Panther','year':2018,'Director':'Ryan Coogler','imdb':8.1},ignore_index=True)
```

output:

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_14444\527112898.py:1: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
df=df.append({'movieId':31,'name':'Black Panther','year':2018,'Director':'Ryan Coogler','imdb':8.1},ignore_index=True)
```

```
df.tail()
```

✓ 0.1s

	movieID	name	year	Director	budget	marketing	boxOffice	studio	genre	imdb	r
26	27.0	Dr. Strange	2014	Scott Derickson	230.0	35.0	780.0	Marvel	Magic	7.0	
27	28.0	Dr. Strange: MOM	2022	Sam Raima	270.0	55.0	1200.0	Marvel	Magic	6.6	
28	29.0	Eternals	2022	Chloe Zhao	240.0	38.0	834.0	Marvel	History Action	7.0	
29	30.0	Shang Chi	2022	Destin Daniel Cretton	210.0	45.0	1005.0	Marvel	Fantasy	7.8	
30	30.0	Black Panther	2018	Ryan Coogler	NaN	NaN	NaN	NaN	NaN	8.1	

to insert a column:

```
df.insert(10,"Country","America")
```

output:

```
df.head()
```

✓ 0.8s

	movieID	name	year	Director	budget	marketing	boxOffice	studio	genre	imdb	Country
0	1.0	Endgame	2018	Russo Brothers	350.0	150.0	3000.0	Marvel	team	7.6	America
1	2.0	Infinity War	2017	Russo Brothers	250.0	170.0	2400.0	Marvel	team	8.1	America
2	3.0	Iron Man	2008	Jon Favreau	150.0	100.0	760.0	Marvel	Sci-fi	8.1	America
3	4.0	Iron Man 2	2010	Jon Favreau	151.0	101.0	861.0	Marvel	Sci-fi	7.1	America
4	5.0	Iron Man 3	2012	Jon Favreau	200.0	120.0	862.0	Marvel	Sci-fi	6.8	America

b. Delete particular sample

dropping row by row id:

```
df.drop(29, axis=0, inplace=True)
```

output:

```
df.tail()
```

✓ 0.1s

	movieID	name	year	Director	budget	marketing	boxOffice	studio	genre	imdb	Country
26	27.0	Dr. Strange	2014	Scott Derickson	230.0	35.0	780.0	Marvel	Magic	7.0	America
27	28.0	Dr. Strange: MOM	2022	Sam Raima	270.0	55.0	1200.0	Marvel	Magic	6.6	America
28	29.0	Eternals	2022	Chloe Zhao	240.0	38.0	834.0	Marvel	History Action	7.0	America
29	30.0	Shang Chi	2022	Destin Daniel Cretton	210.0	45.0	1005.0	Marvel	Fantasy	7.8	America
30	30.0	Black Panther	2018	Ryan Coogler	NaN	NaN	NaN	NaN	NaN	8.1	America

dropping column:

```
df.drop('movieID', inplace=True, axis=1)
```

```
df.head()
```

✓ 0.1s

	name	year	Director	budget	marketing	boxOffice	studio	genre	imdb	Country
0	Endgame	2018	Russo Brothers	350.0	150.0	3000.0	Marvel	team	7.6	America
1	Infinity War	2017	Russo Brothers	250.0	170.0	2400.0	Marvel	team	8.1	America
2	Iron Man	2008	Jon Favreau	150.0	100.0	760.0	Marvel	Sci-fi	8.1	America
3	Iron Man 2	2010	Jon Favreau	151.0	101.0	861.0	Marvel	Sci-fi	7.1	America
4	Iron Man 3	2012	Jon Favreau	200.0	120.0	862.0	Marvel	Sci-fi	6.8	America

c. Update particular sample

To update values at specific sample

```
df.at[17, 'Country'] = "Canada"
```

output:

```
df[15:20]
```

✓ 0.8s

	name	year	Director	budget	marketing	boxOffice	studio	genre	imdb	Country
15	Venom	2018	NaN	170.0	-23.0	NaN	Sony	Alien	6.9	America
16	X-men	2000	Simon Kinberg	94.0	NaN	560.0	Fox	team	7.1	America
17	Deadpool	2016	Tim Miller	120.0	70.0	720.0	Fox	Adult comedy	7.9	Canada
18	Deadpool 2	2019	Tim Miller	200.0	71.0	1024.0	Fox	Adult comedy	7.2	America
19	Woolverine	2004	James Mangold	140.0	NaN	640.0	Fox	Action origin	11.0	America

Replacing given values across the data

```
df.replace("America", "USA", inplace=True)
```

output:

```
df[15:20]
```

✓ 0.1s

	name	year	Director	budget	marketing	boxOffice	studio	genre	imdb	Country
15	Venom	2018	NaN	170.0	-23.0	NaN	Sony	Alien	6.9	USA
16	X-men	2000	Simon Kinberg	94.0	NaN	560.0	Fox	team	7.1	USA
17	Deadpool	2016	Tim Miller	120.0	70.0	720.0	Fox	Adult comedy	7.9	Canada
18	Deadpool 2	2019	Tim Miller	200.0	71.0	1024.0	Fox	Adult comedy	7.2	USA
19	Wolverine	2004	James Mangold	140.0	NaN	640.0	Fox	Action origin	11.0	USA

Q3) Apply Data pre-processing like:

a. Find no. of missing values

Find nan values in each column:

```
df.isna().sum()
```

output:

```
name      0
year      0
Director   1
budget     1
marketing   5
boxOffice  4
studio     1
genre      1
imdb       2
Country    0
dtype: int64
```

Find total nan values in df

```
df.isna().sum().sum()
```

output:

```
15
```

b. Replace missing values by mean, median and mode operations.

```
mean_value=df['marketing'].mean()
df['marketing'].fillna(value=mean_value, inplace=True)
median_value=df['boxOffice'].median()
df['boxOffice'].fillna(value=median_value, inplace=True)
mode_value=df['imdb'].mode()
df['imdb'].fillna(value=int(mode_value), inplace=True)
print(df)
```

output:

	name	year	Director	budget	marketing	\	boxOffice	studio	genre	imdb	Country
0	Endgame	2018	Russo Brothers	350.0	150.000000	0	3000.0	Marvel	team	7.6	USA
1	Infinity War	2017	Russo Brothers	250.0	170.000000	1	2400.0	Marvel	team	8.1	USA
2	Iron Man	2008	Jon Favreau	150.0	100.000000	2	760.0	Marvel	Sci-fi	8.1	USA
3	Iron Man 2	2010	Jon Favreau	151.0	101.000000	3	861.0	Marvel	Sci-fi	7.1	USA
4	Iron Man 3	2012	Jon Favreau	200.0	120.000000	4	862.0	Marvel	Sci-fi	6.8	USA
5	Captain America	2011	Joe Johnston	120.0	50.000000	5	640.0	Marvel	Action	7.1	USA
6	Winter Soldier	2014	Russo Brothers	170.0	120.000000	6	940.0	Marvel	Political Thriller	8.7	USA
7	Civil War	2016	Russo Brothers	270.0	200.000000	7	1840.0	Marvel	team	8.7	USA
8	Thor	2010	Kenneth Branagh	100.0	80.000000	8	760.0	Marvel	Mythological	7.6	USA
9	Dark World	2013	Kenneth Branagh	201.0	120.000000	9	1200.0	Marvel	Mythological	6.4	USA
10	Ragnarok	2018	Taika Wititi	180.0	121.000000	10	976.0	Marvel	Mythological	7.9	USA
11	Love and thunder	2022	Taika Wititi	181.0	122.000000	11	1077.0	Marvel	Mythological	6.2	USA
12	Homecoming	2017	Jon Watts	220.0	100.000000	12	970.0	Sony	Teen comedy	7.6	USA
13	Far from home	2019	Jon Watts	240.0	120.000000	13	1120.0	Sony	Teen comedy	7.4	USA
14	No way home	2021	Jon Watts	260.0	140.000000	14	2170.0	Sony	Teen comedy	7.9	USA
15	Venom	2018	NaN	170.0	-23.000000	15	945.0	Sony	Alien	6.9	USA
16	X-men	2000	Simon Kinberg	94.0	89.038462	16	560.0	Fox	team	7.1	USA
17	Deadpool	2016	Tim Miller	120.0	70.000000	17	720.0	Fox	Adult comedy	7.9	Canada
18	Deadpool 2	2019	Tim Miller	200.0	71.000000	18	1024.0	Fox	Adult comedy	7.2	USA
19	Woolverine	2004	James Mangold	140.0	89.038462	19	640.0	Fox	Action origin	11.0	USA
20	GOTG	2014	James Gunn	200.0	40.000000	20	870.0	Marvel	Space Comedy	7.8	USA
21	GOTG Vol. 2	2017	James Gunn	250.0	60.000000	21	990.0	Marvel	Space Comedy	7.2	USA
22	GOTG Vol. 3	2023	James Gunn	270.0	89.038462	22	945.0	Marvel	Space Comedy	7.0	USA
23	Antman	2015	Peyton Reed	180.0	50.000000	23	870.0	Marvel	Action Comedy	7.1	USA
24	Antman and Wasp	2019	Peyton Reed	200.0	60.000000	24	945.0	Marvel	Action Comedy	7.0	USA
25	A&W: QuantamMania	2023	Peyton Reed	250.0	89.038462	25	945.0	Marvel	Action Comedy	7.0	USA
26	Dr. Strange	2014	Scott Derickson	230.0	35.000000	26	780.0	Marvel	Magic	7.0	USA
27	Dr. Strange: MOM	2022	Sam Raima	270.0	55.000000	27	1200.0	Marvel	Magic	6.6	USA
28	Eternals	2022	Chloe Zhao	240.0	38.000000	28	834.0	Marvel	History Action	7.0	USA
29	Shang Chi	2022	Destin Daniel Cretton	210.0	45.000000	29	1005.0	Marvel	Fantasy	7.8	USA
30	Black Panther	2018	Ryan Coogler	NaN	89.038462	30	945.0	NaN	NaN	8.1	USA

replaced by mean

replaced by median

replaced by mode

c. Apply encoding techniques like

i. ordinal to categorical

```
values=['fresh','rotten']
conditions=[(df["imdb"]>8),(df["imdb"]<=8)]
arr=np.select(conditions,values)
print(arr)
df.insert(9,"score",arr)
```

output:

```
['rotten' 'fresh' 'fresh' 'rotten' 'rotten' 'rotten' 'fresh' 'fresh'
 'rotten' 'rotten' 'rotten' 'rotten' 'rotten' 'rotten' 'rotten' 'rotten'
 'rotten' 'rotten' 'rotten' 'fresh' 'rotten' 'rotten' 'rotten' 'rotten'
 'rotten' 'rotten' 'rotten' 'rotten' 'rotten' 'rotten']
```

score	imdb
rotten	7.6
fresh	8.1
fresh	8.1
rotten	7.1
rotten	6.8
rotten	7.1
fresh	8.7
fresh	8.7
rotten	7.6
rotten	6.4
rotten	7.9
rotten	6.2
rotten	7.6
rotten	7.4
rotten	7.9
rotten	6.9
rotten	7.1
rotten	7.9
rotten	7.2
fresh	11.0
rotten	7.8
rotten	7.2
rotten	7.1
rotten	7.1
rotten	7.0
rotten	7.1
rotten	7.0
rotten	6.6
rotten	7.0
rotten	7.8

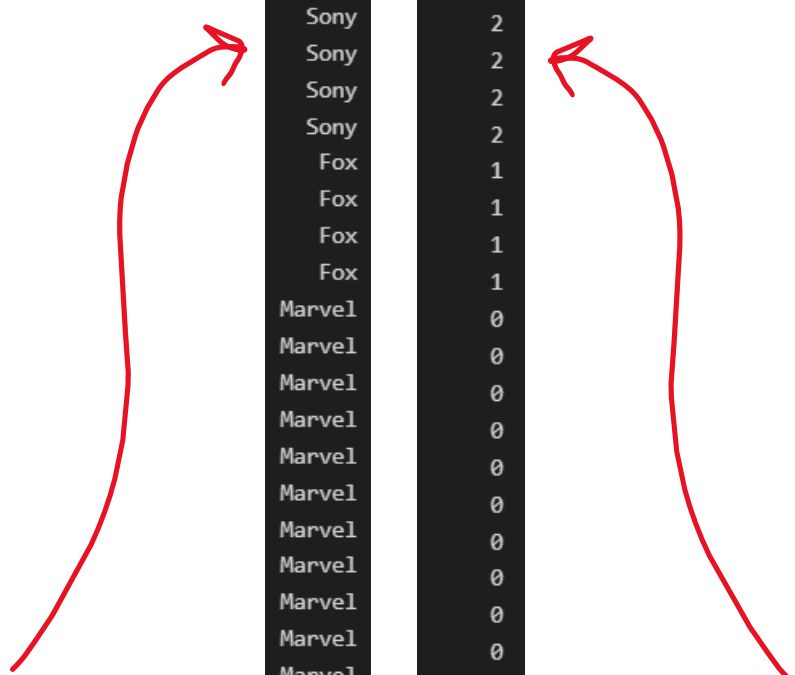
Categorical data

Original numeric data (no categorical data available in dataset)

ii. categorical (text) to categorical (numeric)

```
df['Country'].replace([0,1], ['USA', 'Canada'], inplace=True)
```

Output:



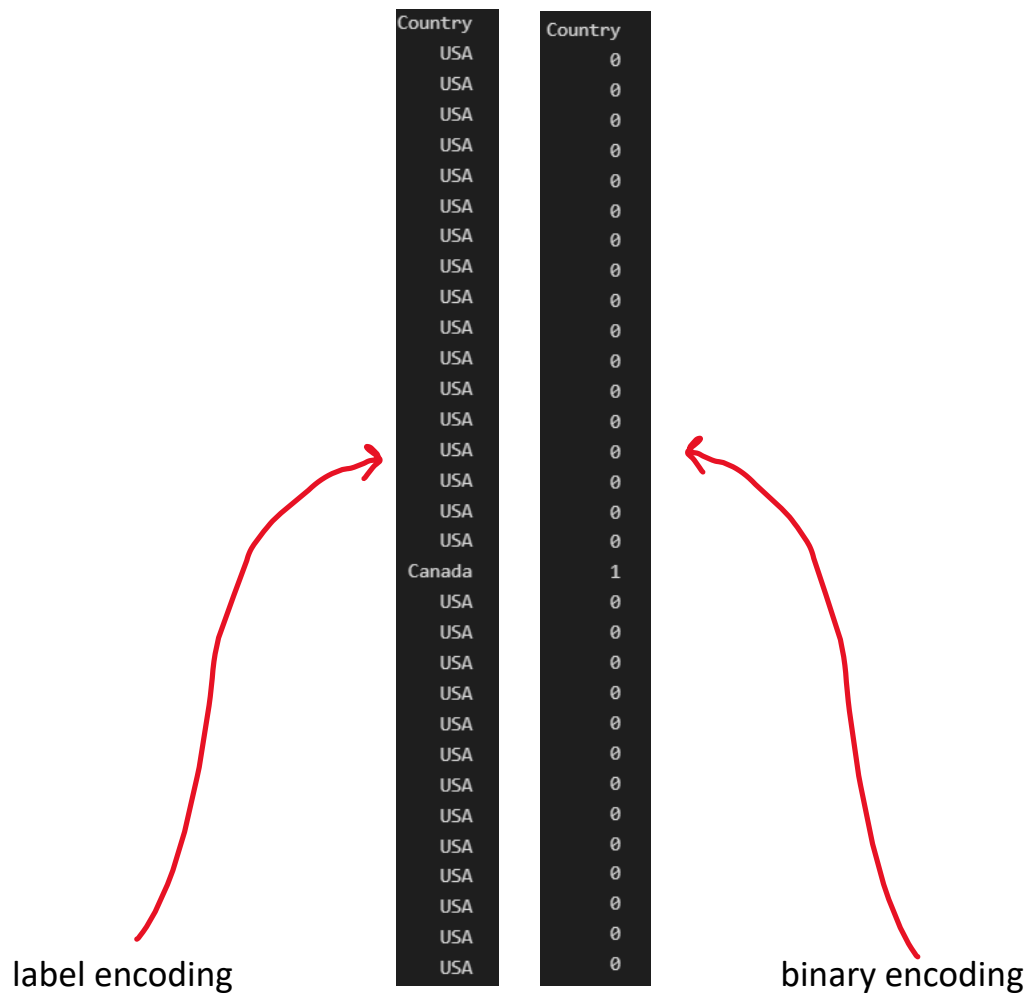
studio	studio
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Sony	2
Sony	2
Sony	2
Sony	2
Fox	1
Fox	1
Fox	1
Fox	1
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0
Marvel	0

Categorical (text) Data

Categorical (numeric) Data

iii. label encoding to binary encoding

```
df['studio'].replace(['Marvel', 'Fox', 'Sony'], [0,1,2], inplace=True)
```



Q4) Apply Normalization techniques

a.Minmax

```
df_min_max_scaled = df[['imdb','boxOffice','marketing','budget']]

for column in df_min_max_scaled.columns:
    df_min_max_scaled[column] = (df_min_max_scaled[column] -
df_min_max_scaled[column].min()) / (df_min_max_scaled[column].max() -
df_min_max_scaled[column].min())

print(df_min_max_scaled)
```

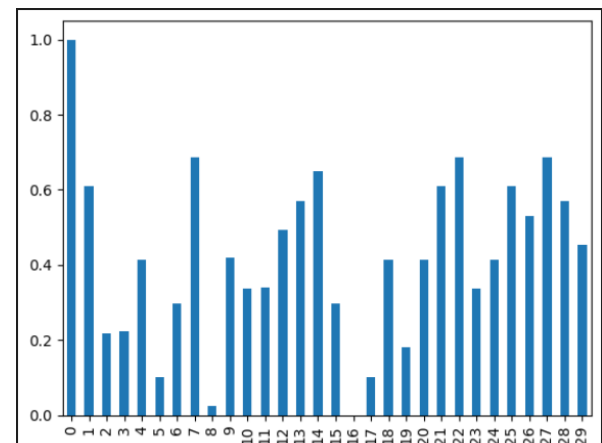
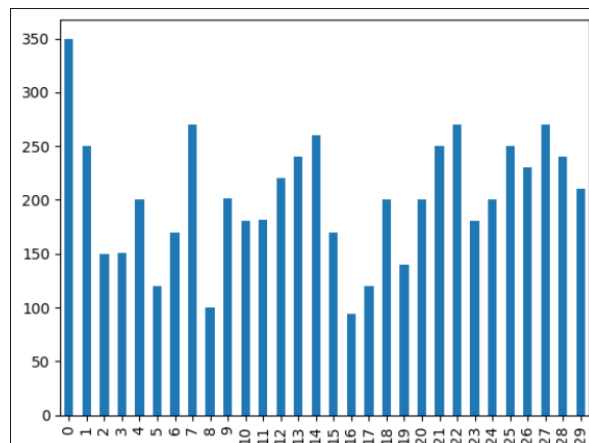
Output:

Before

	imdb	boxOffice	marketing	budget
0	7.6	3000.0	150.000000	350
1	8.1	2400.0	170.000000	250
2	8.1	760.0	100.000000	150
3	7.1	861.0	101.000000	151
4	6.8	862.0	120.000000	200
5	7.1	640.0	50.000000	120
6	8.7	940.0	120.000000	170
7	8.7	1840.0	200.000000	270
8	7.6	760.0	80.000000	100
9	6.4	1200.0	120.000000	201
10	7.9	976.0	121.000000	180
11	6.2	1077.0	122.000000	181
12	7.6	970.0	100.000000	220
13	7.4	1120.0	120.000000	240
14	7.9	2170.0	140.000000	260
15	6.9	945.0	-23.000000	170
16	7.1	560.0	89.038462	94
17	7.9	720.0	70.000000	120
18	7.2	1024.0	71.000000	200
19	11.0	640.0	89.038462	140
20	7.8	870.0	40.000000	200
21	7.2	990.0	60.000000	250
22	7.1	945.0	89.038462	270
23	7.1	870.0	50.000000	180
24	7.0	945.0	60.000000	200
25	7.1	945.0	89.038462	250
26	7.0	780.0	35.000000	230
27	6.6	1200.0	55.000000	270
28	7.0	834.0	38.000000	240
29	7.8	1005.0	45.000000	210

After MinMax Normalization

	imdb	boxOffice	marketing	budget
0	0.291667	1.000000	0.775785	1.000000
1	0.395833	0.754098	0.865471	0.609375
2	0.395833	0.081967	0.551570	0.218750
3	0.187500	0.123361	0.556054	0.222656
4	0.125000	0.123770	0.641256	0.414062
5	0.187500	0.032787	0.327354	0.101562
6	0.520833	0.155738	0.641256	0.296875
7	0.520833	0.524590	1.000000	0.687500
8	0.291667	0.081967	0.461883	0.023438
9	0.041667	0.262295	0.641256	0.417969
10	0.354167	0.170492	0.645740	0.335938
11	0.000000	0.211885	0.650224	0.339844
12	0.291667	0.168033	0.551570	0.492188
13	0.250000	0.229508	0.641256	0.570312
14	0.354167	0.659836	0.730942	0.648438
15	0.145833	0.157787	0.000000	0.296875
16	0.187500	0.000000	0.502415	0.000000
17	0.354167	0.065574	0.417040	0.101562
18	0.208333	0.190164	0.421525	0.414062
19	1.000000	0.032787	0.502415	0.179688
20	0.333333	0.127049	0.282511	0.414062
21	0.208333	0.176230	0.372197	0.609375
22	0.187500	0.157787	0.502415	0.687500
23	0.187500	0.127049	0.327354	0.335938
24	0.166667	0.157787	0.372197	0.414062
25	0.187500	0.157787	0.502415	0.609375
26	0.166667	0.090164	0.260090	0.531250
27	0.083333	0.262295	0.349776	0.687500
28	0.166667	0.112295	0.273543	0.570312
29	0.333333	0.182377	0.304933	0.453125



Budget column bar graph (before and after)

b. Standard scaling (according to mean)

```
df_standard_scaled = df[['imdb','boxOffice','marketing','budget']]

for column in df_standard_scaled.columns:
    df_standard_scaled[column] = (df_standard_scaled[column] -
df_standard_scaled[column].mean()) / (df_standard_scaled[column].max() -
df_standard_scaled[column].min())

print(df_standard_scaled)
```

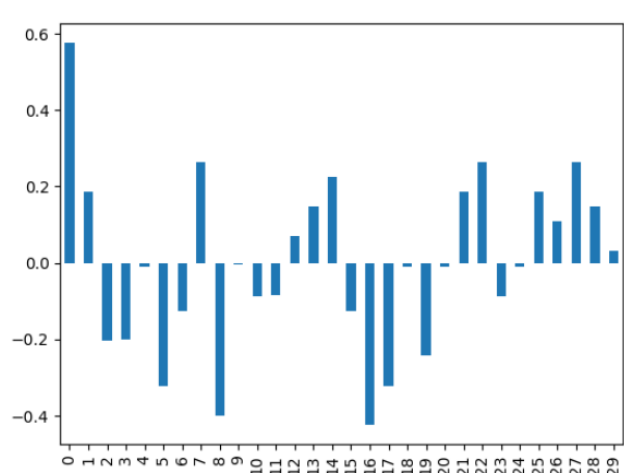
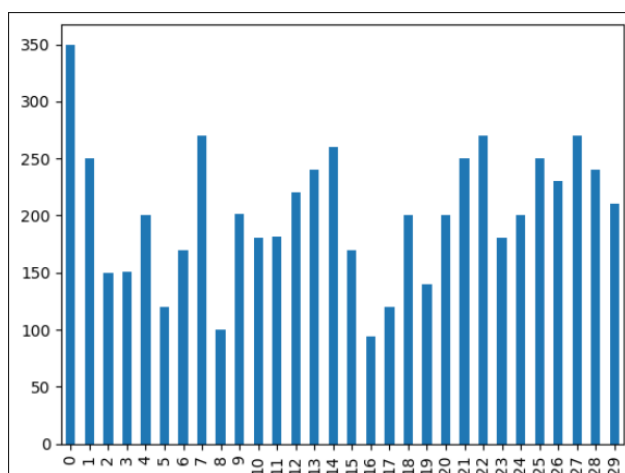
Output:

Before

	imdb	boxOffice	marketing	budget
0	7.6	3000.0	150.000000	350
1	8.1	2400.0	170.000000	250
2	8.1	760.0	100.000000	150
3	7.1	861.0	101.000000	151
4	6.8	862.0	120.000000	200
5	7.1	640.0	50.000000	120
6	8.7	940.0	120.000000	170
7	8.7	1840.0	200.000000	270
8	7.6	760.0	80.000000	100
9	6.4	1200.0	120.000000	201
10	7.9	976.0	121.000000	180
11	6.2	1077.0	122.000000	181
12	7.6	970.0	100.000000	220
13	7.4	1120.0	120.000000	240
14	7.9	2170.0	140.000000	260
15	6.9	945.0	-23.000000	170
16	7.1	560.0	89.038462	94
17	7.9	720.0	70.000000	120
18	7.2	1024.0	71.000000	200
19	11.0	640.0	89.038462	140
20	7.8	870.0	40.000000	200
21	7.2	990.0	60.000000	250
22	7.1	945.0	89.038462	270
23	7.1	870.0	50.000000	180
24	7.0	945.0	60.000000	200
25	7.1	945.0	89.038462	250
26	7.0	780.0	35.000000	230
27	6.6	1200.0	55.000000	270
28	7.0	834.0	38.000000	240
29	7.8	1005.0	45.000000	210

After

	imdb	boxOffice	marketing	budget
0	0.020833	0.780751	0.273370	0.577214
1	0.125000	0.534850	0.363056	0.186589
2	0.125000	-0.137281	0.049155	-0.204036
3	-0.083333	-0.095888	0.053639	-0.200130
4	-0.145833	-0.095478	0.138841	-0.008724
5	-0.083333	-0.186462	-0.175060	-0.321224
6	0.250000	-0.063511	0.138841	-0.125911
7	0.250000	0.305342	0.497585	0.264714
8	0.020833	-0.137281	-0.040531	-0.399349
9	-0.229167	0.043046	0.138841	-0.004818
10	0.083333	-0.048757	0.143325	-0.086849
11	-0.270833	-0.007363	0.147810	-0.082943
12	0.020833	-0.051216	0.049155	0.069401
13	-0.020833	0.010260	0.138841	0.147526
14	0.083333	0.440587	0.228527	0.225651
15	-0.125000	-0.061462	-0.502415	-0.125911
16	-0.083333	-0.219249	0.000000	-0.422786
17	0.083333	-0.153675	-0.085374	-0.321224
18	-0.062500	-0.029085	-0.080890	-0.008724
19	0.729167	-0.186462	0.000000	-0.243099
20	0.062500	-0.092199	-0.219903	-0.008724
21	-0.062500	-0.043019	-0.130217	0.186589
22	-0.083333	-0.061462	0.000000	0.264714
23	-0.083333	-0.092199	-0.175060	-0.086849
24	-0.104167	-0.061462	-0.130217	-0.008724
25	-0.083333	-0.061462	0.000000	0.186589
26	-0.104167	-0.129085	-0.242325	0.108464
27	-0.187500	0.043046	-0.152639	0.264714
28	-0.104167	-0.106954	-0.228872	0.147526
29	0.062500	-0.036872	-0.197482	0.030339



Budget column bar graph (before and after)