from google.colab import files

uploaded=files.upload()

Choose Files No file chosen Cancel upload

movies\_metadata.csv: The main Movies Metadata le. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

#sets the backend of matplotlib to the inline backend

%matplotlib inline

warnings.filterwarnings('ignore')

import statsmodels.api as sm

# Loading the data

df = pd.read\_csv("movies\_dataset.csv", encoding= 'unicode\_escape')

df.head(5)

**adult belongs\_to\_collection budget genres homepage id imdb\_id** [{'id': 16, 

**0** False {'id': 10194, 'name': 'Toy

Story Collection', ... 30000000

print(f'Number of rows: {df.shape[0]}') print(f'Number of columns: {df.shape[1]}')

**1** False NaN 65000000 #summary of a DataFrame

df.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 45466 entries, 0 to 45465 Data columns (total 24 columns):

{'id': 119050, 'name':

'name':

'Animation'}, {'id': 35, '...

[{'id': 12,

'name':

'Adventure'}, {'id': 14, '...

[{'id': 10749,

http://toystory.disney.com/toy

story 862 tt0114709

NaN 8844 tt0113497 

# Column Non-Null Count Dtype  

**2** False

'Grumpy Old Men

0

'name':

NaN 15602 tt0113228 

--- ------ -------------- -----  

Collect...

'Romance'}, 

0 adult 45466 non-null object

{'id': 35, ...

 1 belongs\_to\_collection 4494 non-null object   2 budget 45466 non-null object   3 genres 45466 non-null object   4 homepage 7782 non-null object

[{'id': 35,

 5 id 45466 non-null object

'name':

 6 imdb\_id 45449 non-null object  

**3** False NaN 16000000

'Comedy'},

NaN 31357 tt0114885 

7 original\_language 45455 non-null object

{'id': 18,

 8 original\_title 45466 non-null object

'nam...

 9 overview 44512 non-null object   10 popularity 45461 non-null object   11 poster\_path 45080 non-null object  [{'id': 35,

 12 production\_companies 45463 non-null object  **4** False {'id': 96871, 'name': 

'Father of the Bride Col... 0

'name':

NaN 11862 tt0113041 

13 production\_countries 45463 non-null object  'Comedy'}]

 14 release\_date 45379 non-null object   15 revenue 45460 non-null float64  16 runtime 45203 non-null float64  17 spoken\_languages 45460 non-null object   18 status 45379 non-null object   19 tagline 20412 non-null object   20 title 45460 non-null object   21 video 45460 non-null object   22 vote\_average 45460 non-null float64  23 vote\_count 45460 non-null float64 dtypes: float64(4), object(20)

memory usage: 8.3+ MB

#summary statistic of the numerical variables df.describe()

**revenue runtime vote\_average vote\_count**

**count** 4.546000e+04 45203.000000 45460.000000 45460.000000

**mean** 1.120935e+07 94.128199 5.618207 109.897338

**std** 6.433225e+07 38.407810 1.924216 491.310374

**min** 0.000000e+00 0.000000 0.000000 0.000000

#checking datatype for columns

df.dtypes

**25%** 0.000000e+00 85.000000 5.000000 3.000000

adult object

**50%** 0.000000e+00 95.000000 6.000000 10.000000

belongs\_to\_collection object

**75%** 0.000000e+00 107.000000 6.800000 34.000000

budget object

genres object

**max** 2.787965e+09 1256.000000 10.000000 14075.000000

homepage object

id object

imdb\_id object

original\_language object

original\_title object

overview object

popularity object

poster\_path object

production\_companies object

production\_countries object

release\_date object

revenue float64

runtime float64

spoken\_languages object

status object

tagline object

title object

video object

vote\_average float64

vote\_count float64

dtype: object

**DATA RELATION**

#getting independent values from dataset

x = df.iloc[:, :-1].values

print(x)

[['False'

 "{'id': 10194, 'name': 'Toy Story Collection', 'poster\_path': '/7G9915LfUQ2lVfwMEEhDsn3kT4B.jpg  '30000000' ... 'Toy Story' False 7.7]

 ['False' nan '65000000' ... 'Jumanji' False 6.9]

 ['False'

 "{'id': 119050, 'name': 'Grumpy Old Men Collection', 'poster\_path': '/nLvUdqgPgm3F85NMCii9gVFUce '0' ... 'Grumpier Old Men' False 6.5]

 ...

 ['False' nan '0' ... 'Betrayal' False 3.8]

 ['False' nan '0' ... 'Satan Triumphant' False 0.0]

 ['False' nan '0' ... 'Queerama' False 0.0]]

#getting dependent values from dataset

=-

print(y)

[5415. 2413. 92. ... 6. 0. 0.]

**DATA PREPROCESSING**

#checking for missing values

df.isnull()

**adult belongs\_to\_collection budget genres homepage id imdb\_id original\_language 0** False False False False False False False False **1** False True False False True False False False **2** False False False False True False False False **3** False True False False True False False False **4** False False False False True False False False **...** ... ... ... ... ... ... ... ...

**45461** False True False False False False False False **45462** False True False False True False False False **45463** False True False False True False False False **45464** False True False False True False False False **45465** False True False False True False False False 45466 rows × 24 columns

#count for missing values

print(df.isnull().sum())

adult 0

belongs\_to\_collection 40972

budget 0

genres 0

homepage 37684

id 0

imdb\_id 17

original\_language 11

original\_title 0

overview 954

popularity 5

poster\_path 386

production\_companies 3

production\_countries 3

release\_date 87

revenue 6

runtime 263

spoken\_languages 6

status 87

tagline 25054

title 6

video 6

vote\_average 6

vote\_count 6

dtype: int64

# After dropping the values

print(df.isnull().sum())

adult 0

belongs\_to\_collection 40972

budget 0

genres 0

homepage 37684

id 0

imdb\_id 17

original\_language 11

original\_title 0

overview 954

popularity 5

poster\_path 386

production\_companies 3

production\_countries 3

release\_date 87

revenue 6

runtime 263

spoken\_languages 6

status 87

tagline 25054

title 6

video 6

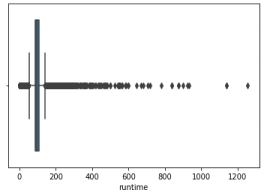
vote\_average 6

vote\_count 6

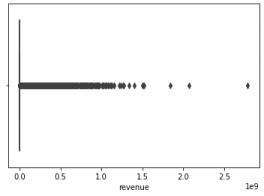
dtype: int64

#detecting outliers for budget

sns.boxplot(x = df['runtime'])

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f6a20322b70> 

sns.boxplot(x = df['revenue'])

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f6a1d793a58> 

sns.boxplot(x = df['vote\_count'])

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f6a1d766198> 

#Transformation - Binning

data = df['vote\_count']

data = data[:30] #taking only first 30 data items data=np.sort(data)

print('Data:')

print(data)

print('')

#create three different matrices having 10 rows and 3 columns b1=np.zeros((10,3))

b2=np.zeros((10,3))

b3=np.zeros((10,3))

#binning by mean

for i in range (0,30,3):

 k=int(i/3)

 mean=(data[i] + data[i+1] + data[i+2] )/3

 for j in range(3):

 b1[k,j]=mean

print("----Binning by mean:---- \n",b1)

print('')

#binning by median

for i in range (0,30,3):

 k=int(i/3)

 for j in range (3):

 b2[k,j]=data[i+1]

print("----Binning by median:---- \n",b2)

print('')

#binning by boundary

for i in range (0,30,3):

 k=int(i/3)

 for j in range (3):

 if (data[i+j]-data[i]) < (data[i+2]-data[i+j]):

 b3[k,j]=data[i]

 else:

 b3[k,j]=data[i+2]

print("----Binning by boundary:----\n",b3)

Data:

[ 17. 33. 34. 36. 45. 72. 91. 92. 137. 141. 143. 173.  174. 199. 199. 210. 224. 305. 308. 364. 365. 394. 423. 539.  1128. 1194. 1343. 1886. 2413. 5415.]

----Binning by mean:----

 [[ 28. 28. 28. ]

 [ 51. 51. 51. ]

 [ 106.66666667 106.66666667 106.66666667]

 [ 152.33333333 152.33333333 152.33333333]

 [ 190.66666667 190.66666667 190.66666667]

 [ 246.33333333 246.33333333 246.33333333]

 [ 345.66666667 345.66666667 345.66666667]

 [ 452. 452. 452. ]

 [1221.66666667 1221.66666667 1221.66666667]

 [3238. 3238. 3238. ]]

----Binning by median:----

 [[ 33. 33. 33.]

 [ 45. 45. 45.]

 [ 92. 92. 92.]

 [ 143. 143. 143.]

 [ 199. 199. 199.]

 [ 224. 224. 224.]

 [ 364. 364. 364.]

 [ 423. 423. 423.]

 [1194. 1194. 1194.]

 [2413. 2413. 2413.]]

----Binning by boundary:----

 [[ 17. 34. 34.]

 [ 36. 36. 72.]

 [ 91. 91. 137.]

 [ 141. 141. 173.]

 [ 174. 199. 199.]

 [ 210. 210. 305.]

 [ 308. 365. 365.]

 [ 394. 394. 539.]

 [1128. 1128. 1343.]

 [1886. 1886. 5415.]]

#Transformation - Normalization

import statistics

from sklearn import preprocessing

#min-max normalization

def minMaxNor(num,list):

 minNum=int(input("Enter Minimun Setting:\t"))

 maxNum = int(input("Enter Maximum Setting:\t"))

 ans=round(((num-min(list))/(max(list)-min(list))\*(maxNum-minNum))+minNum,2)  return ans

data = df['vote\_count']

data = data[:10] #taking only first 10 data items

data=np.sort(data)

print(data)

#z-score normalization

def zNor (num,mean,stdDv):

 return round((num-mean)/stdDv,2)

#modified z-score normalization

def zNorMAD (num,mean,abMeanDiv):

 return round((num-mean)/abMeanDiv,2)

#decimal-scaling normalization

def decNor(num,maxNum):

 digit=len(str(maxNum))

 div=pow(10,digit)

 return num/div

num=float(input("Enter an item from data : \t"))

if num in data:

 print("Calculating min-max normalization")

 print("After doing min-max normalization :",minMaxNor(num,data))

 print("\nCalculating z-score normalization")

 print("After doing z-score normalization : \t", zNor(num,statistics.mean(data),statistics.stdev(data) print("\nCalculating Modified z-score normalization")

 df = pd.DataFrame(data)

 print("After doing Modified z-score normalization : \t", zNorMAD(num,statistics.mean(data),df.mad())) print("\nCalculating decimal scaling normalization")

 print("After doing decimal scaling normalization : \t", decNor(num,max(data))) else:

 print("Item entered is not present!!")

 print("Can't perform normalization on the selected item!")

[ 34. 45. 92. 141. 173. 174. 1194. 1886. 2413. 5415.]

Enter an item from data : 45

Calculating min-max normalization

Enter Minimun Setting: 0

Enter Maximum Setting: 1

After doing min-max normalization : 0.0

Calculating z-score normalization

After doing z-score normalization : -0.64

Calculating Modified z-score normalization

After doing Modified z-score normalization : 0 -0.88

dtype: float64

Calculating decimal scaling normalization

After doing decimal scaling normalization : 4.5e-05

**DATA VISUALIZATION**

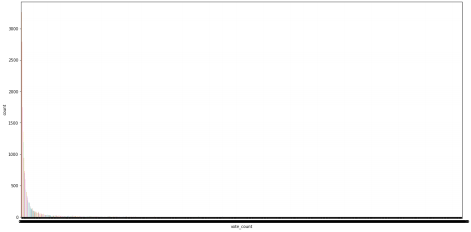
#finding no. of movies who got a particular vote

plt.figure(figsize=(20,10))

sns.countplot(x='vote\_count',data=df,palette="pastel")

plt.xticks(fontsize=9)

plt.show()



plt.figure(figsize=(20,10))

sns.countplot(x='release\_date',data=df,palette="pastel")

plt.xticks(fontsize=9)

plt.show()