**1. import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**2. dataset = pd.read\_excel("2255872-anime\_data.xlsx")**

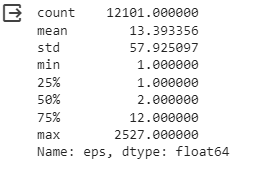
**3. dataset.head(2)**

****

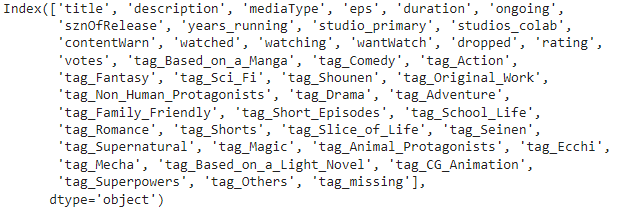
**4. dataset.shape**

****

**5. dataset.eps.describe()**

****

**6. dataset.columns**

****

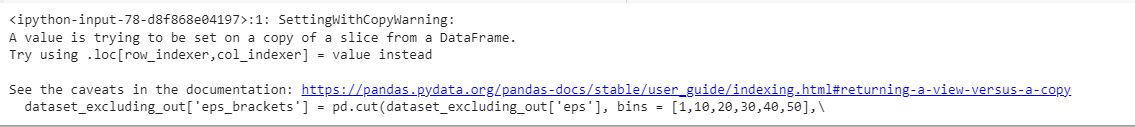
**7. dataset[(dataset['eps'] > 24) & (dataset.duration.isna())].shape**

****

**8. dataset\_excluding\_out = dataset[dataset['eps'] < 50]**

**9.dataset\_excluding\_out['eps\_brackets'] = pd.cut(dataset\_excluding\_out['eps'], bins = [1,10,20,30,40,50],\**

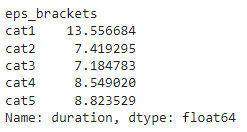
**labels = ['cat1','cat2','cat3','cat4','cat5'])**

****

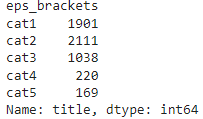
**10. dataset\_excluding\_out.shape**

****

**11. dataset\_excluding\_out.groupby(['eps\_brackets']).duration.mean()**

****

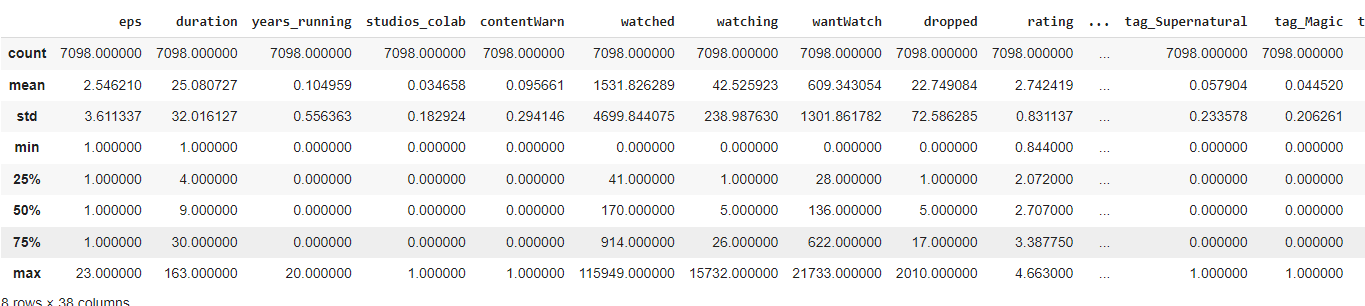
**12. dataset\_excluding\_out.groupby(['eps\_brackets']).title.count()**

****

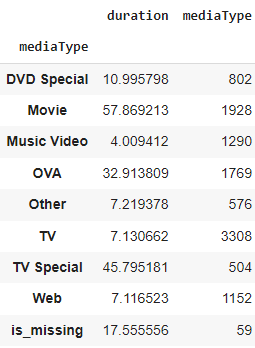
**13. dataset\_excluding\_out[dataset\_excluding\_out['eps\_brackets'] == 'cat1'].shape**

****

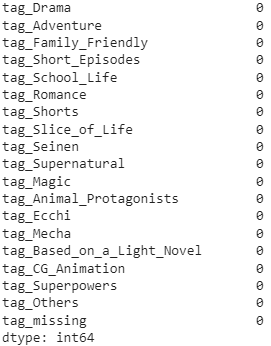
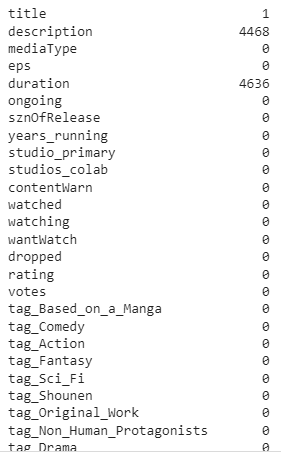
**14. dataset[(dataset['eps'] < 24) & (~dataset.duration.isna())].describe()**

****

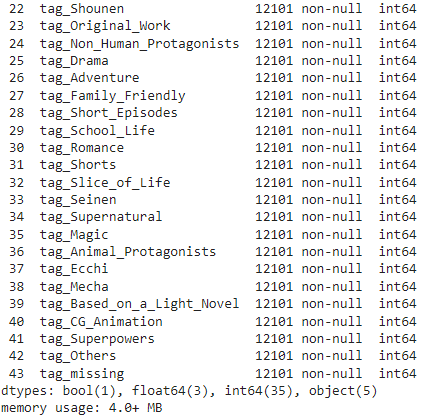
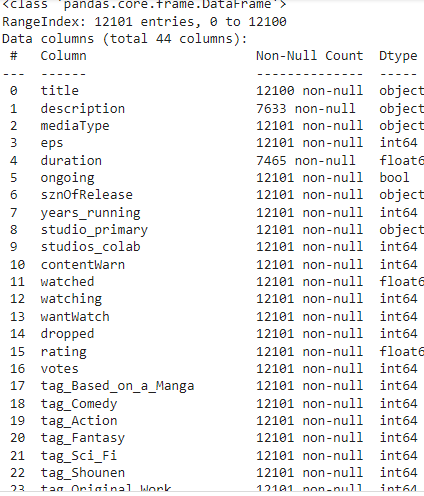
**15.dataset\_excluding\_out.groupby('mediaType').agg({'duration':'mean','mediaType':'count'})**

****

**16.dataset.isna().sum()**

****

**17. dataset.info()**

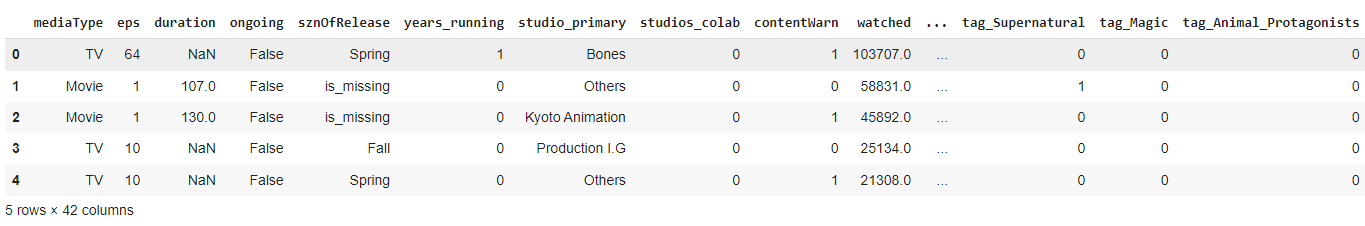
****

**18. dataset.describe().T**

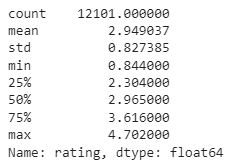
****

**19. dataset.drop(columns = ['title','description'], axis=1, inplace=True)**

**20. dataset.head()**

****

**21. dataset.rating.describe()**

****

**22. dataset.dropna(inplace=True)**

**dataset.shape**

****

**23. def continuous\_univariate\_analysis(data,**

**feature,**

**figsize=(12,8),**

**kde=False,**

**bins=None):**

**f1, (ax\_box,**

**ax\_hist) = plt.subplots(nrows=2,**

**sharex=True,**

**gridspec\_kw={'height\_ratios': (0.25,0.75)},**

**figsize=figsize)**

**sns.color\_palette("viridis", as\_cmap=True)**

**sns.boxplot(data=data,**

**x=feature,**

**ax=ax\_box,**

**showmeans=True,**

**color='yellow')**

**sns.histplot(data=data,**

**x=feature,**

**ax=ax\_hist,**

**showmeans=True,**

**color='crest',**

**bins=bins,**

**kde=kde) if bins else sns.histplot(**

**data=data, x=feature, ax=ax\_hist, kde=kde, color='blue')**

**ax\_hist.axvline(data[feature].mean(), color='cyan', linestyle='--')**

**ax\_hist.axvline(data[feature].median(), color='orange', linestyle="-")**

**24.**

**def discrete\_univariate\_analysis(data, feature, perc=False, n=None):**

**total = len(data[feature])**

**count = data[feature].nunique()**

**if n is None:**

**plt.figure(figsize=(count + 1, 5))**

**else:**

**plt.figure(figsize=(n + 1, 5))**

**plt.xticks(rotation=90, fontsize=15)**

**ax = sns.countplot(**

**data=data,**

**x=feature,**

**palette="flare",**

**order=data[feature].value\_counts().index[:n].sort\_values(ascending=False))**

**for p in ax.patches:**

**if perc == True:**

**label = "{:.1f}%".format(100 \* p.get\_height() / total)**

**else:**

**label = p.get\_height()**

**x = p.get\_x() + p.get\_width() / 2**

**y = p.get\_height()**

**ax.annotate(label, (x, y),**

**ha="center",**

**va="center",**

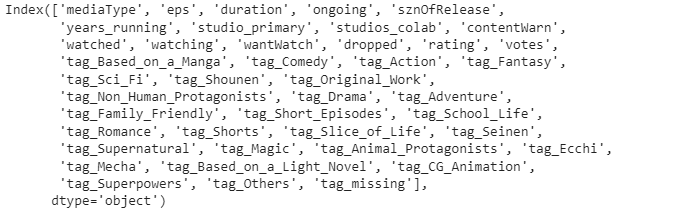
**size=12,**

**xytext=(0, 5),**

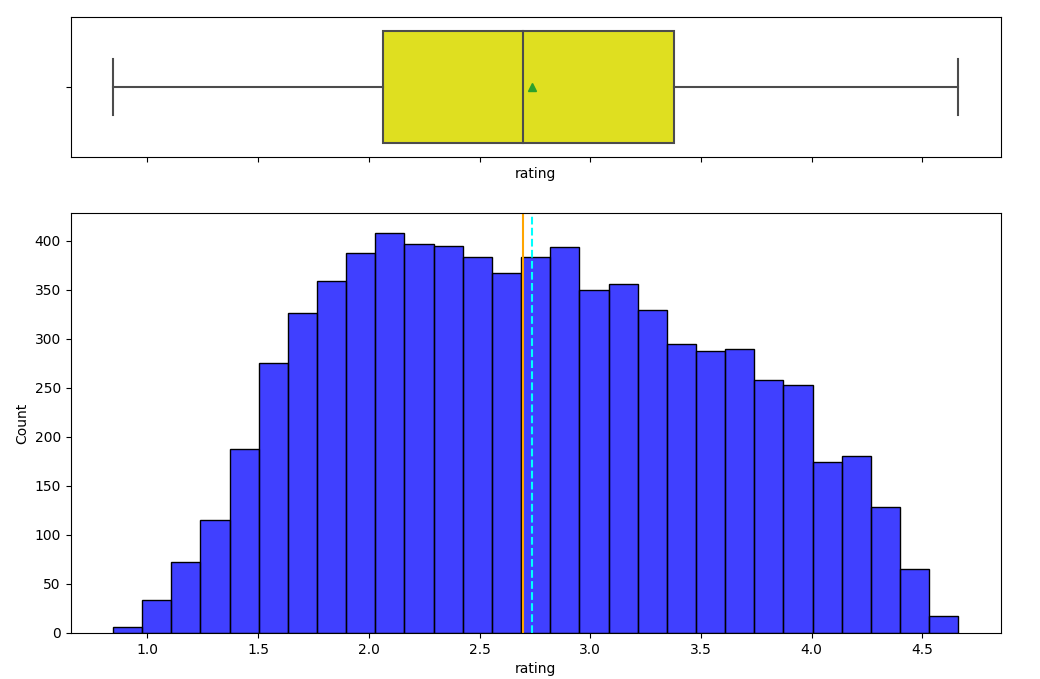
**textcoords="offset points")**

**plt.show()**

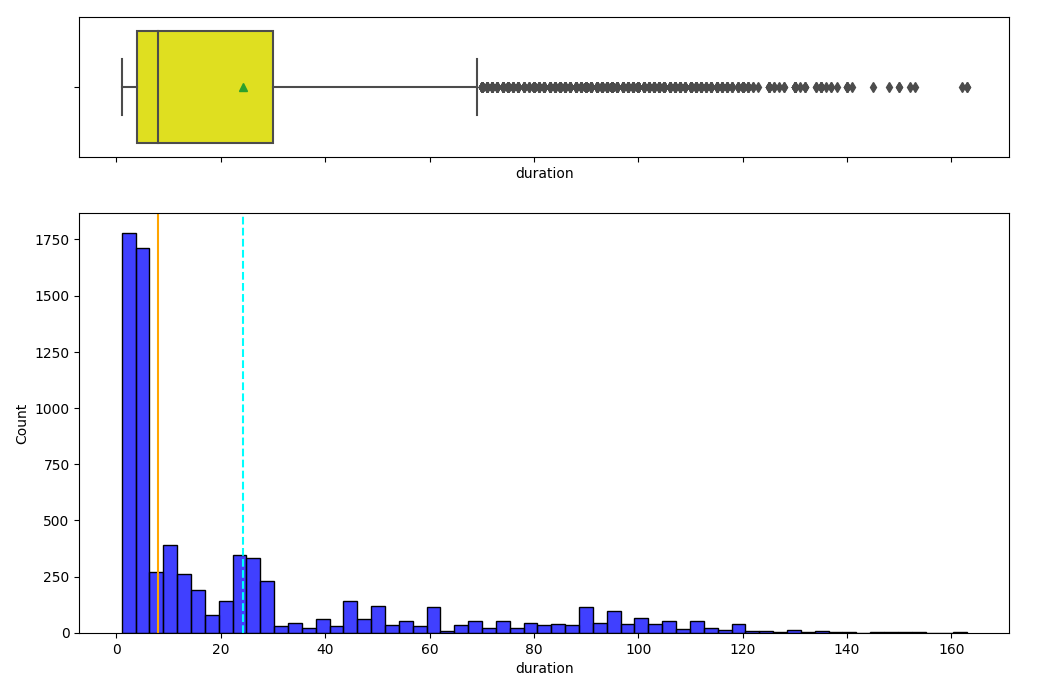
**25. dataset.columns**

****

**26. continuous\_univariate\_analysis(dataset, 'rating')**

****

**27. continuous\_univariate\_analysis(dataset, 'duration')**

****

**28. dataset[dataset['duration'] >=80]['rating'].mean()**

****

**29. dataset[dataset['duration'] >=100]['rating'].mean()**

****

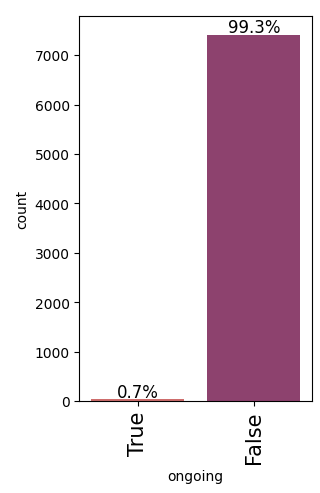
**30. dataset[dataset['duration'] >=110]['rating'].mean()**

****

**31. dataset[(dataset['duration'] >=5) & (dataset['duration']<=30)]['rating'].mean()**

****

**32. discrete\_univariate\_analysis(dataset, "ongoing", perc=True)**

****

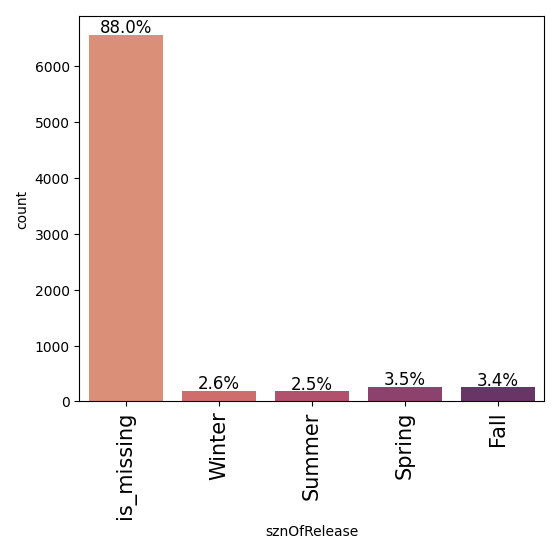
**33. dataset[dataset['ongoing'] == True]['rating'].mean()**

****

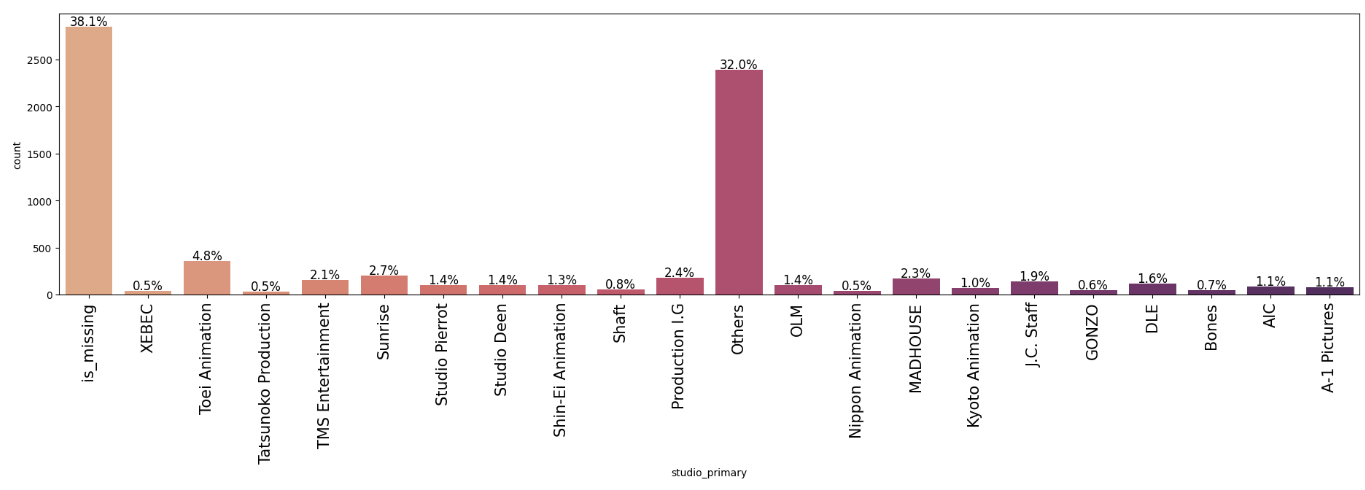
**34. dataset[dataset['ongoing'] == True]['duration'].mean()**

****

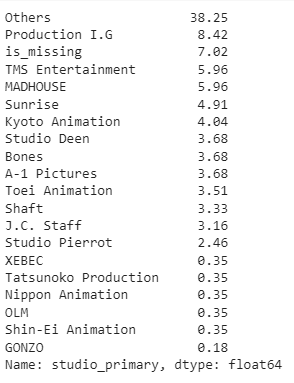
**35. discrete\_univariate\_analysis(dataset, "sznOfRelease", perc=True)**

****

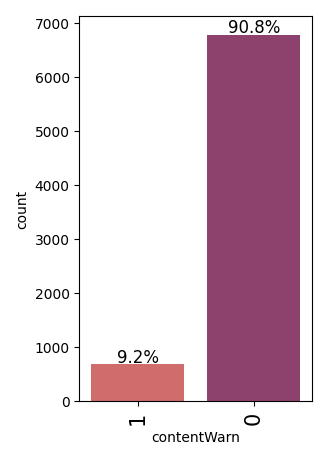
**36. discrete\_univariate\_analysis(dataset , "studio\_primary", perc=True)**

****

**37. dataset[dataset['rating'] > 4]['studio\_primary'].value\_counts(normalize=True).mul(100).round(2)**

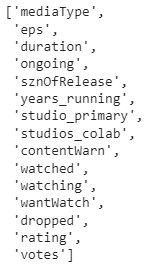
****

**38. discrete\_univariate\_analysis(dataset, 'contentWarn', perc=True)**

****

**39. corr\_cols = [item for item in dataset.columns if "tag" not in item]**

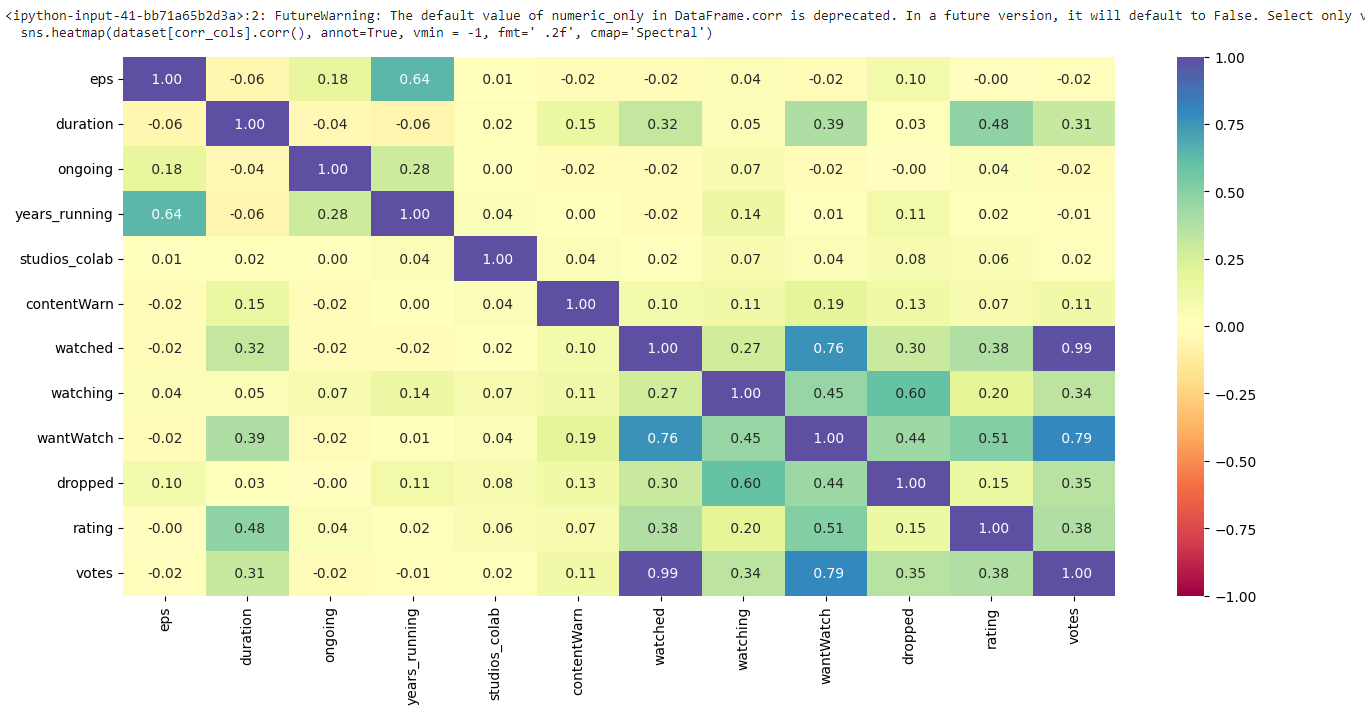
**40. corr\_cols**

****

**41. plt.figure(figsize=(16,7))**

**sns.heatmap(dataset[corr\_cols].corr(), annot=True, vmin = -1, fmt=' .2f', cmap='Spectral')**

**plt.show()**

****

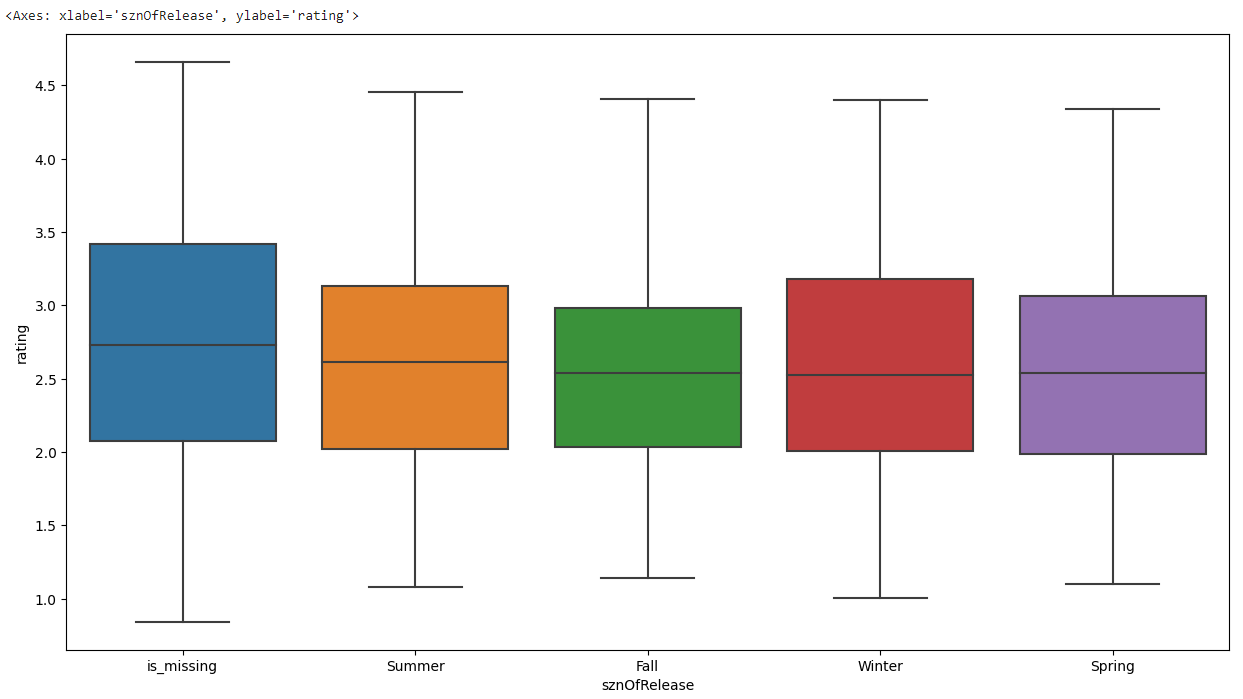
**42. dataset.drop(columns= ['eps', 'watched'], inplace=True)**

**43. dataset.shape**

****

**44.** plt.figure(figsize=(15,8))

sns.boxplot(x = 'sznOfRelease', y='rating', data=dataset)

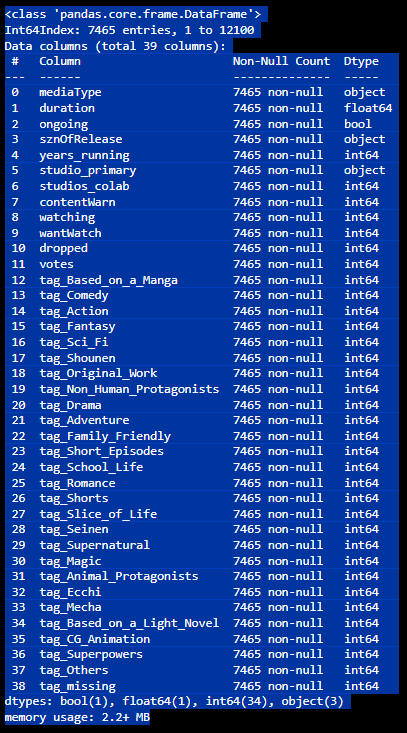


##Model Building - Regression

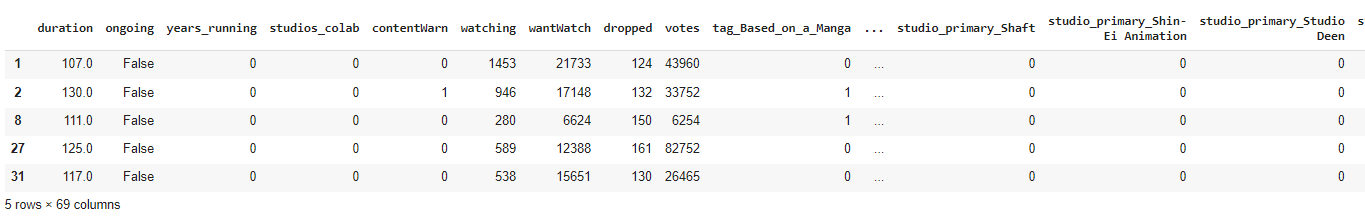
**45.** x = dataset.drop(['rating'], axis=1)

y = dataset['rating']

**46.** x.info()

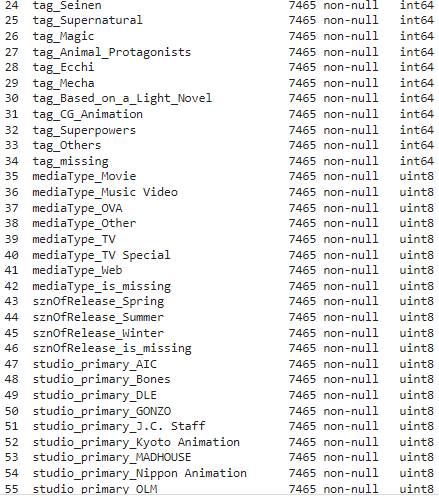
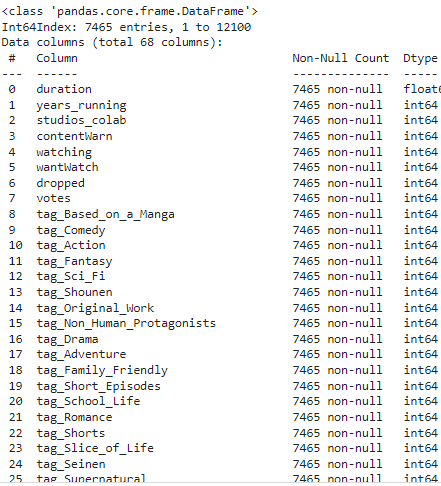
****

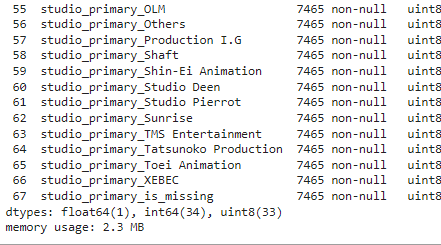
**47.** x = pd.get\_dummies(x, columns=x.select\_dtypes(include=['object', 'category']).columns.tolist(), drop\_first=True)

x.head()

**48.** x.drop(columns='ongoing', inplace=True)

**49.** x.info()

****

****

**50.** from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error

**51.** X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(x,y, test\_size=0.2, random\_state=1)

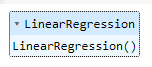
**52.** X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(x,y, test\_size=0.2, random\_state=1)

**53.** print("Number of samples for train", X\_train.shape[0])

print("Number of samples for test", X\_test.shape[0])

**54.** lin\_model = LinearRegression()

lin\_model.fit(X\_train, Y\_train)



**55.** def Model\_performance(model, predictor, target):

    pred = model.predict(predictor)

    r2 = r2\_score(target, pred)

    rmse = np.sqrt(mean\_squared\_error(target, pred))

    results= pd.DataFrame({

        "RMSE":rmse,

        "R2 Score":r2

    }, index=[0]

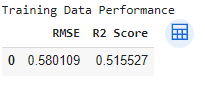
    )

    return results

**56.** print("Training Data Performance")

lin\_model\_train = Model\_performance(lin\_model, X\_train, Y\_train)

lin\_model\_train

****

**57.** x.columns

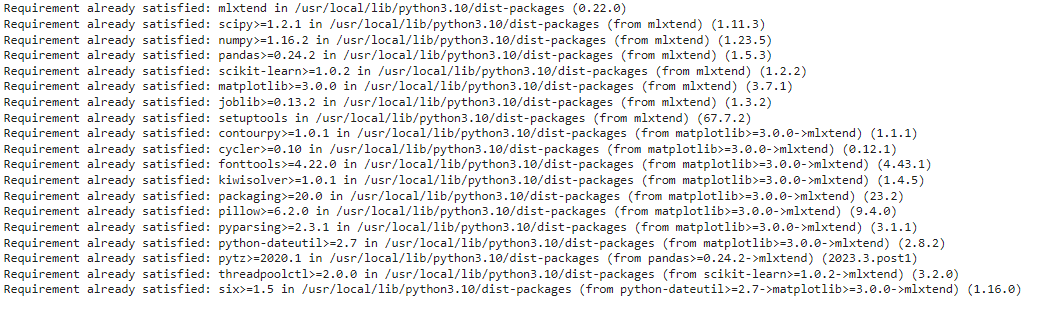
****

##feature selection technique

**58.** ##conda install mlxtend

!pip install mlxtend

import mlxtend



**59.** from mlxtend.feature\_selection import SequentialFeatureSelector as SFS

reg = LinearRegression()

sfs = SFS(reg, k\_features = X\_train.shape[1],

forward = True, floating=False, scoring='r2', n\_jobs= -1, cv=5)

sfs = sfs.fit(X\_train, Y\_train)

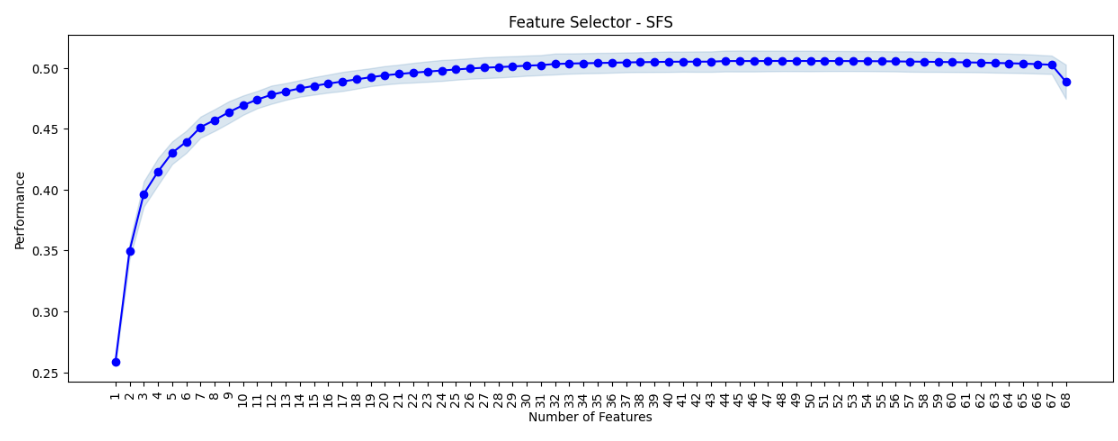
**60.** from mlxtend.plotting import plot\_sequential\_feature\_selection as plot\_sfs

fig1 = plot\_sfs(sfs.get\_metric\_dict(), kind='std\_err', figsize=(15,5))

plt.title("Feature Selector - SFS")

plt.xticks(rotation=90)

plt.show()



**61.** from mlxtend.feature\_selection import SequentialFeatureSelector as SFS

reg = LinearRegression()

sfs = SFS(reg, k\_features = 30,

         forward = True, floating=False, scoring='r2', n\_jobs= -1, cv=5)

sfs = sfs.fit(X\_train, Y\_train)

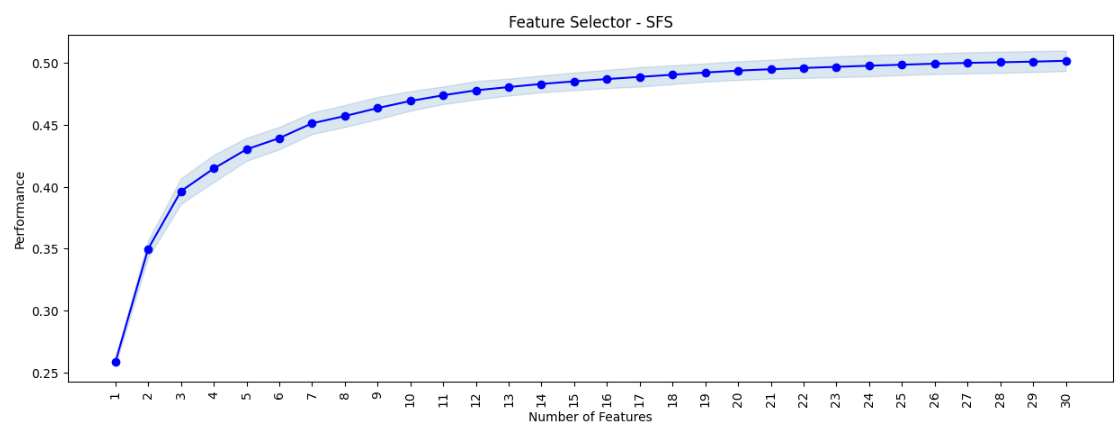
**62.** from mlxtend.plotting import plot\_sequential\_feature\_selection as plot\_sfs

fig1 = plot\_sfs(sfs.get\_metric\_dict(), kind='std\_err', figsize=(15,5))

plt.title("Feature Selector - SFS")

plt.xticks(rotation=90)

plt.show()



**63.** from mlxtend.feature\_selection import SequentialFeatureSelector as SFS

reg = LinearRegression()

sfs = SFS(reg, k\_features = 35,

       forward = True, floating=False, scoring='r2', n\_jobs= -1, cv=5)

sfs = sfs.fit(X\_train, Y\_train)

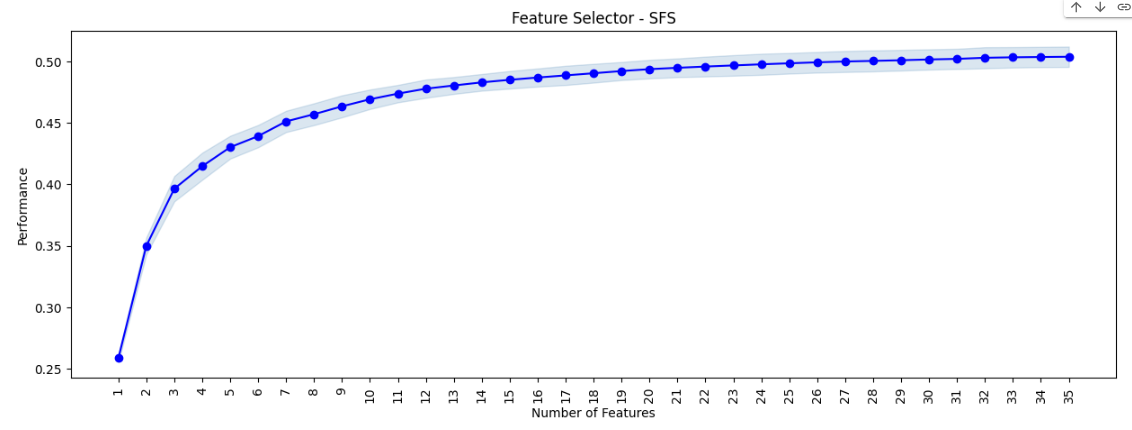
**64.** from mlxtend.plotting import plot\_sequential\_feature\_selection as plot\_sfs

fig1 = plot\_sfs(sfs.get\_metric\_dict(), kind='std\_err', figsize=(15,5))

plt.title("Feature Selector - SFS")

plt.xticks(rotation=90)

plt.show()

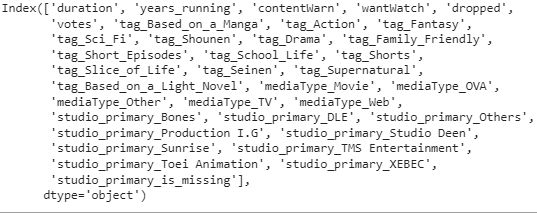


**65.** feature\_index = list(sfs.k\_feature\_idx\_)

print(feature\_index)

****

**66.** X\_train.columns[feature\_index]

****

**67.** print("Training Data Performance")

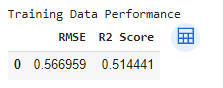
lin\_model\_train = Model\_performance(lin\_model, X\_train, Y\_train)

lin\_model\_train

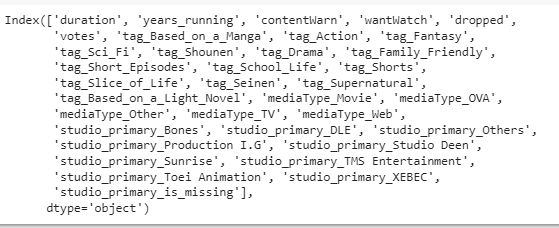
**68.** print("Training Data Performance")

lin\_model\_train = Model\_performance(lin\_model, X\_test, Y\_test)

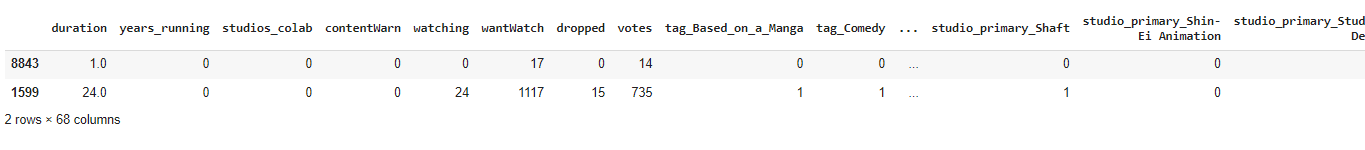
lin\_model\_train

****

**69.** X\_train.columns[feature\_index]

****

**70.** X\_train.head(2)



**Name –** Mithilesh maurya

**Batch-**DS with ML and Python

Certificate Code- TCRIB4R140