

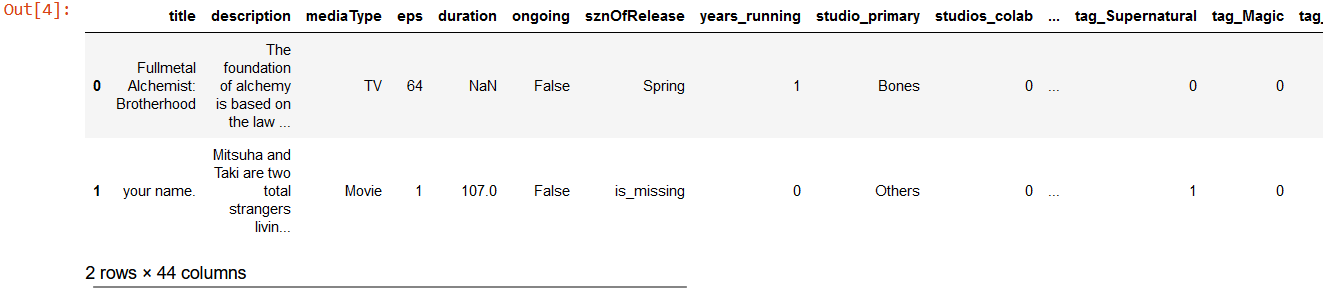
1. import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

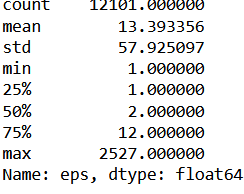
1. dataset = pd.read\_csv('NewWeb.csv')
2. dataset.head(2)



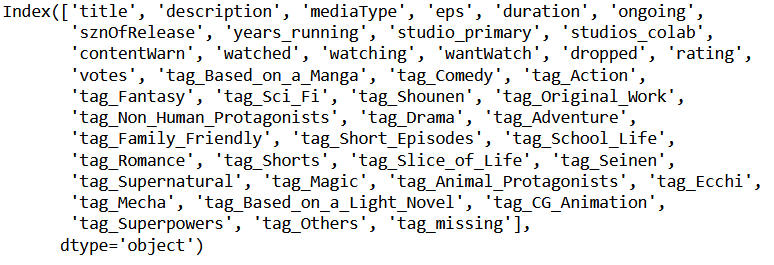
1. dataset.shape



5. dataset.eps.describe()



6.dataset.columns



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



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

10. dataset\_excluding\_out['eps\_brackets']= pd.cut(dataset\_excluding\_out['eps'],

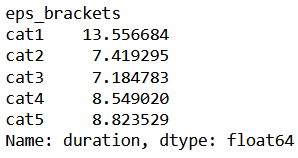
bins=[1,10,20,30,40,50] ,\labels = ['cat1','cat2','cat3','cat4','cat5'])

11. import warnings warnings.filterwarnings("ignore")

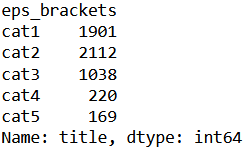
12. dataset\_excluding\_out.shape



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



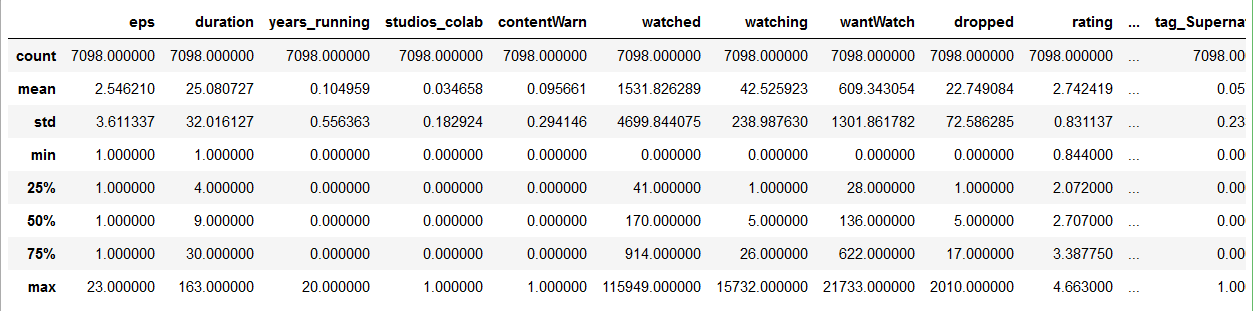
14. dataset\_excluding\_out.groupby(['eps\_brackets']).title.count()



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

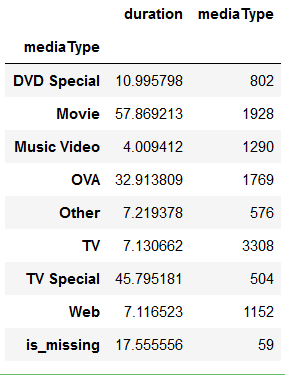


16. dataset[(dataset['eps']<24) & (~dataset.duration.isna())].describe()

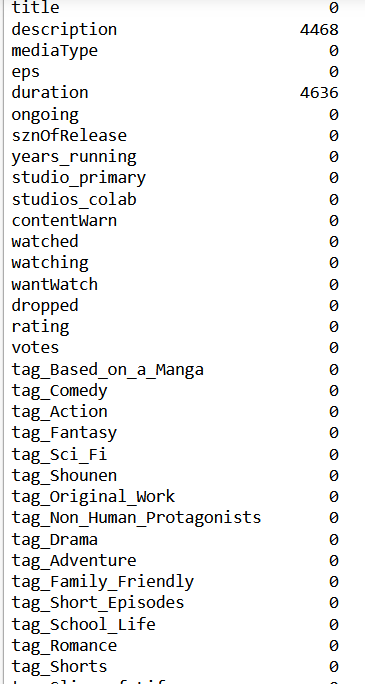


17. dataset\_excluding\_out.groupby('mediaType').agg({'duration'

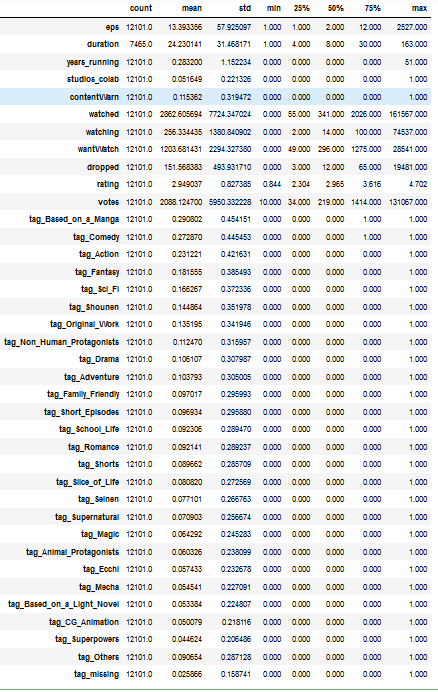
'mean','mediaType':'count'})



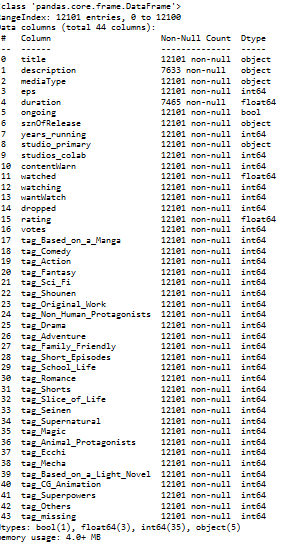
18. dataset.isna().sum()



19. dataset.describe().T

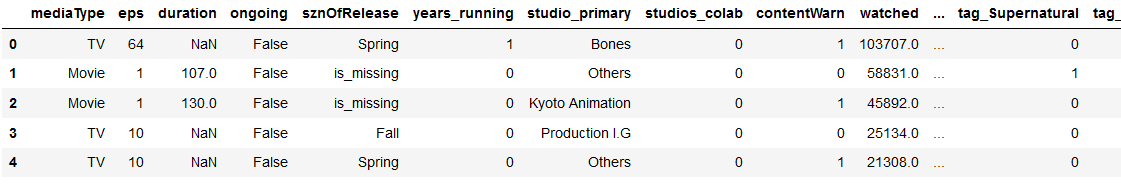


20. dataset.info()

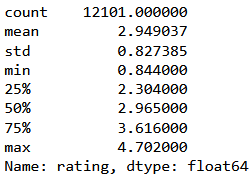


21. dataset.drop(columns = ['title','description'] , axis=1,inplace=True)

22. dataset.head()



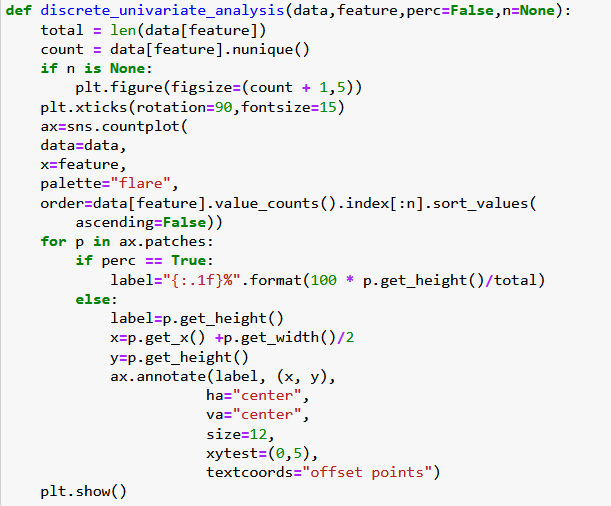
23. dataset.rating.describe()



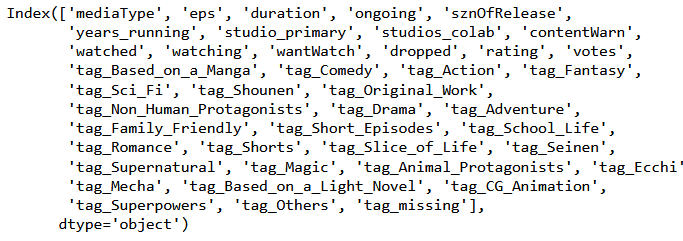
24. dataset.dropna(inplace=True) dataset.shape



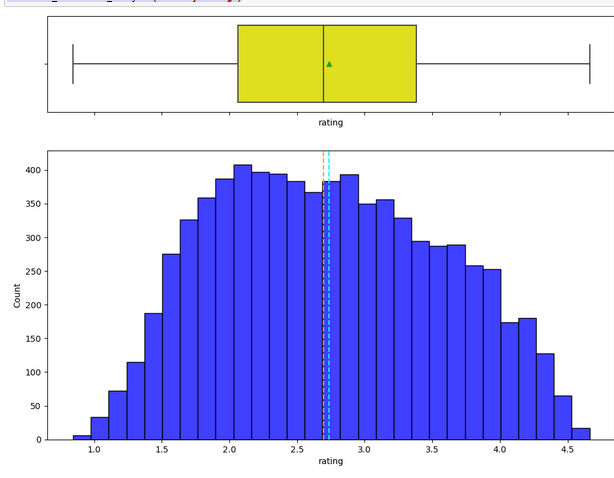




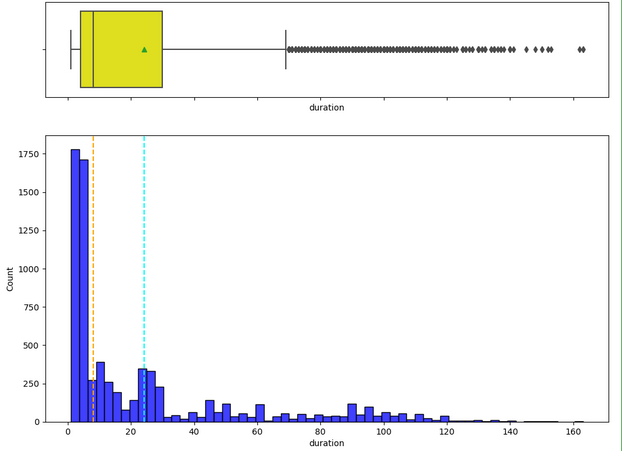
25. dataset.columns

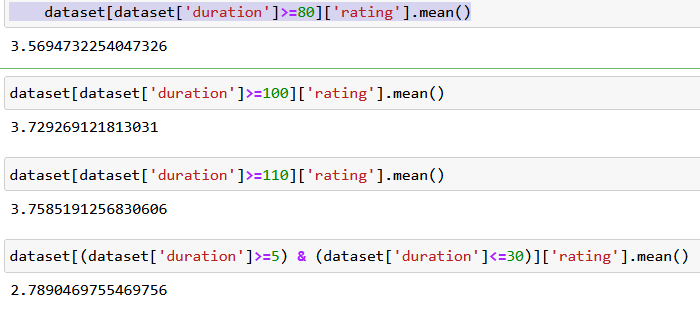


26. continous\_univariate\_analysis(dataset,'rating')

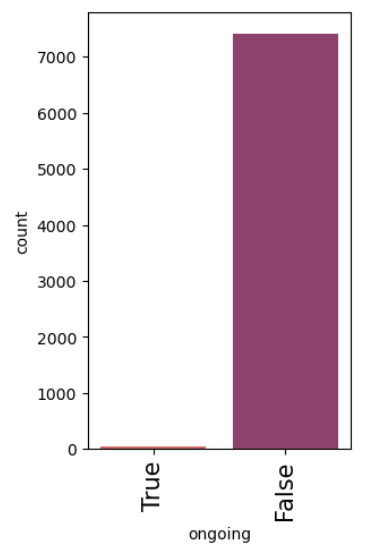


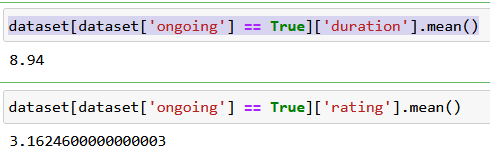
27.continous\_univariate\_analysis(dataset,'duration')



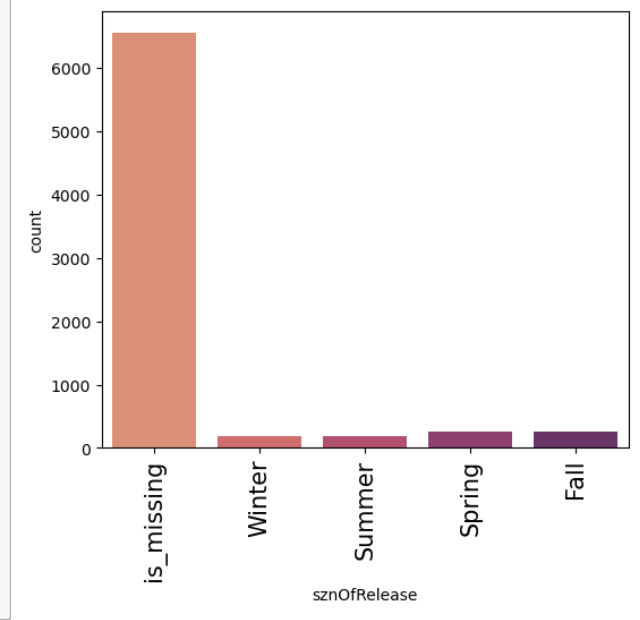


28.discrete\_univariate\_analysis(dataset,'ongoing',perc=True)

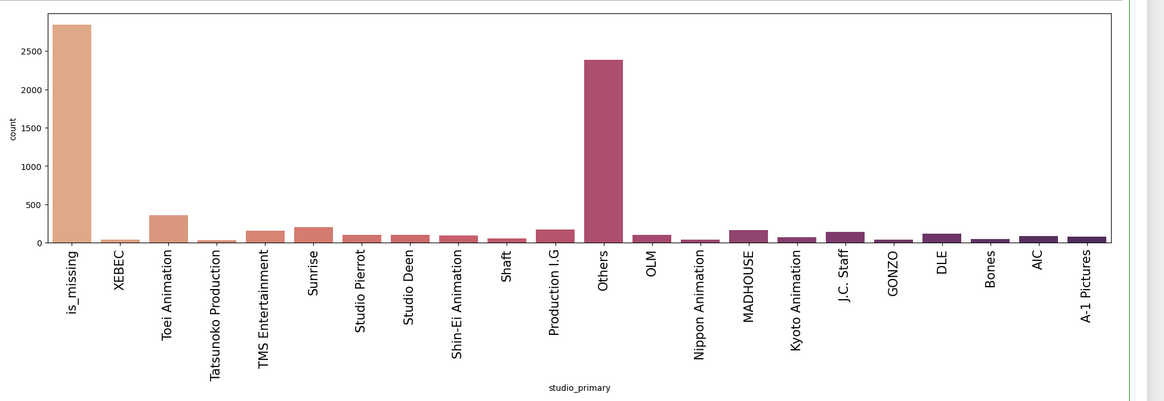




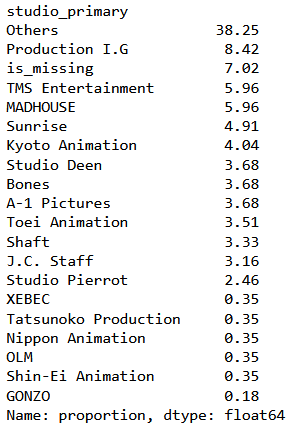
29.discrete\_univariate\_analysis(dataset,'sznOfRelease',perc=True)



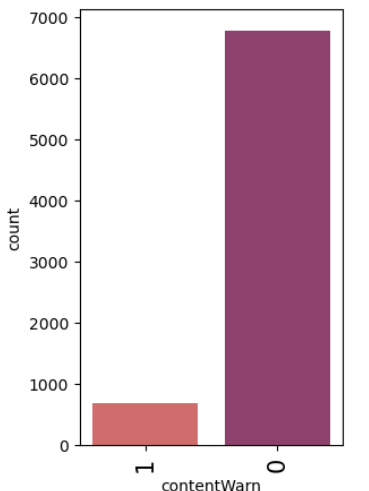
30.discrete\_univariate\_analysis(dataset,'studio\_primary',perc=True)



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

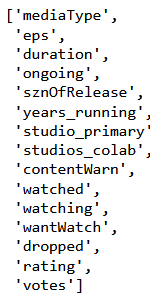


32.discrete\_univariate\_analysis(dataset,'contentWarn',perc=True)



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

corr\_cols

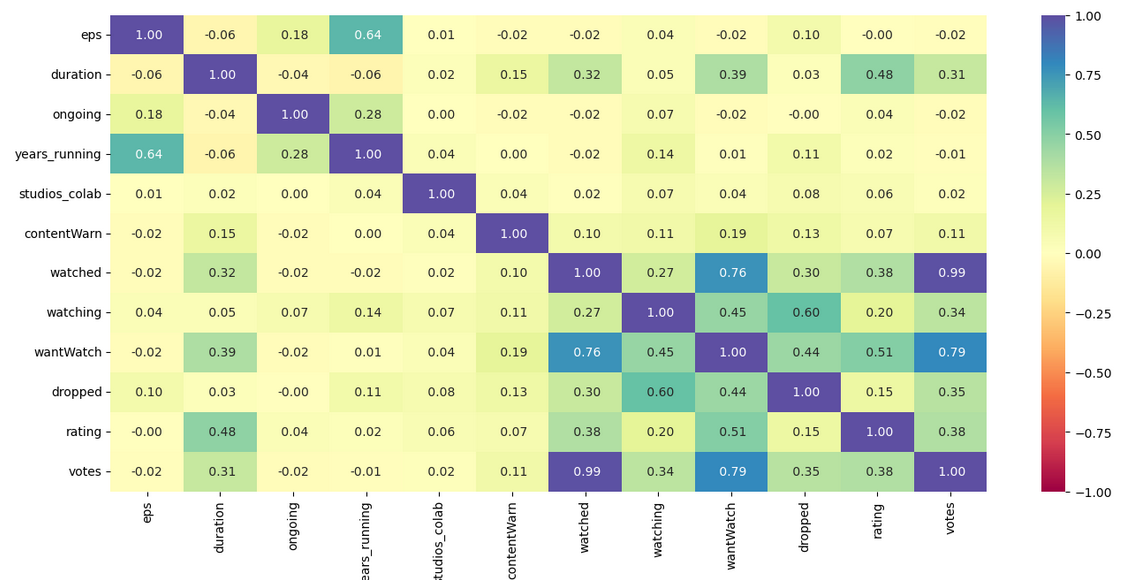


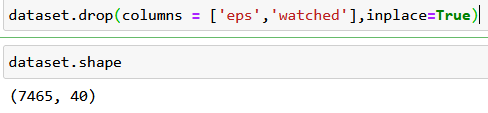
35.dataset['ongoing'] = dataset['ongoing'].astype(int)

36.corr\_cols = ['eps', 'duration', 'ongoing', 'years\_running', 'studios\_colab', 'contentWarn', 'watched', 'watching', 'wantWatch', 'dropped', 'rating', 'votes']

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

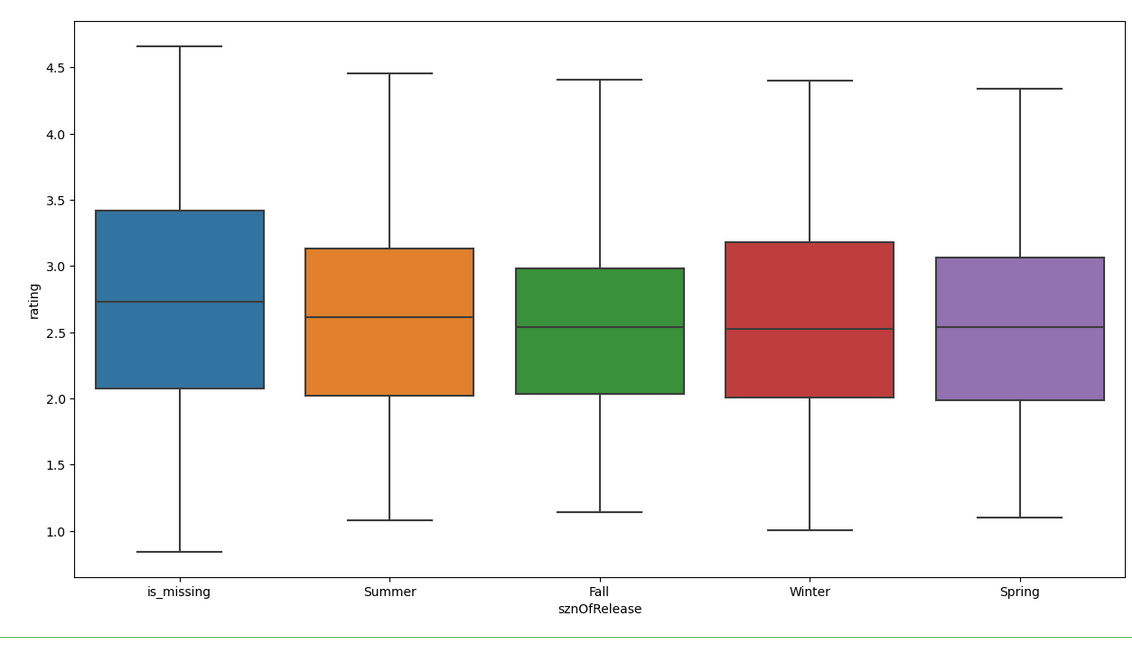
37.sns.heatmap(dataset[corr\_cols].corr(), annot=True, vmin = -1, fmt=' .2f', cmap='Spectral')

plt.show()



38.plt.figure(figsize=(15,8))

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

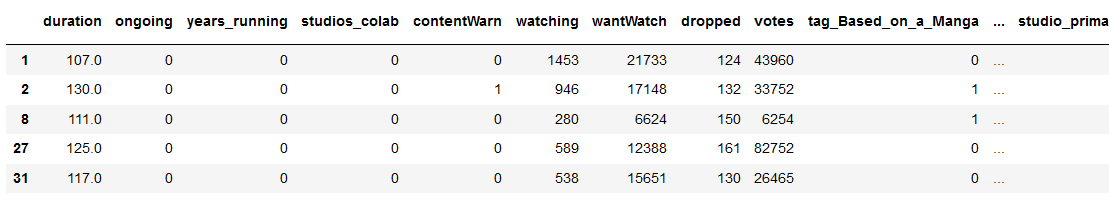


39.x=dataset.drop(['rating'],axis=1)

40.y=dataset['rating']

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

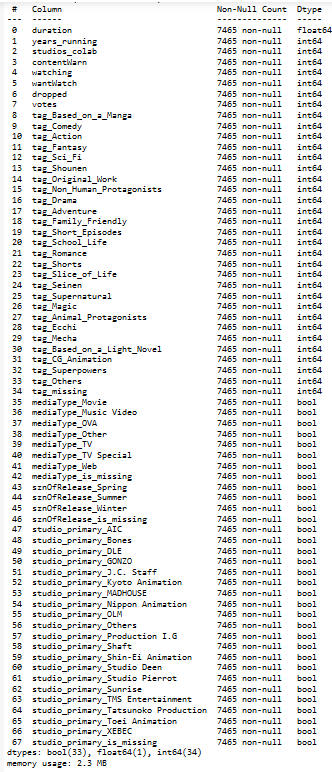
x.head()



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

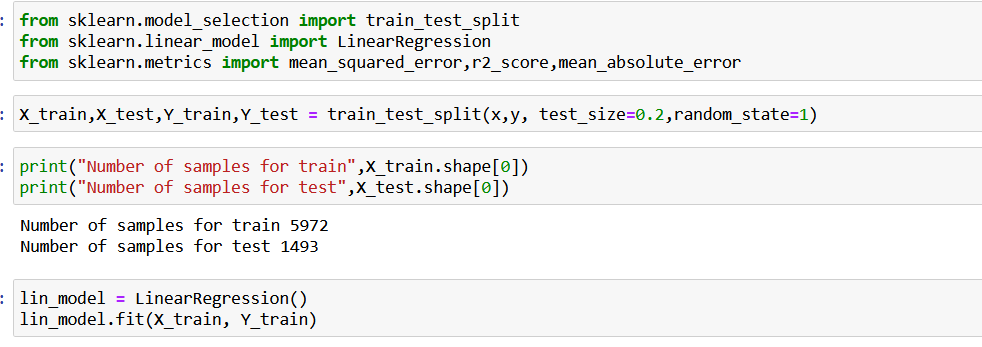
43.x.drop(columns='ongoing',inplace=True)

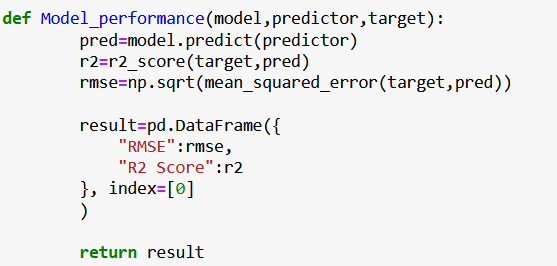
x.info()



43.x.columns



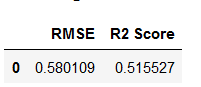




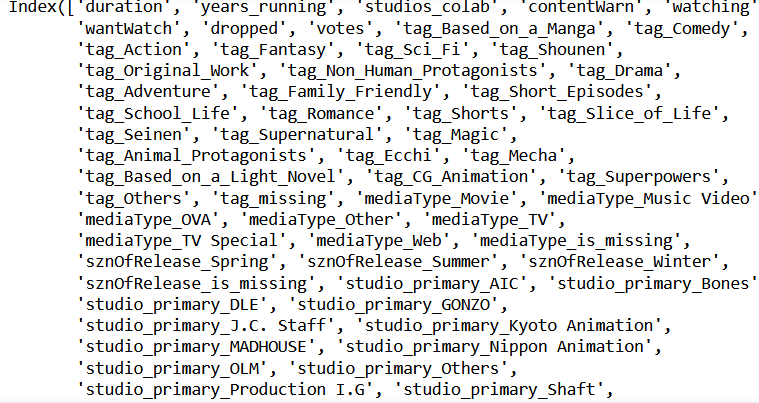
44.print("Training Data Performance")

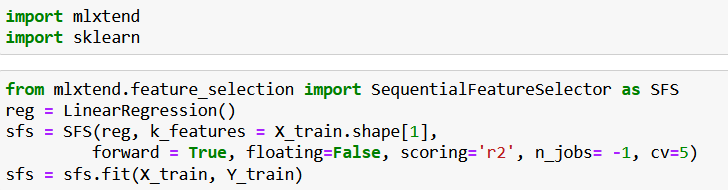
lin\_model\_train = Model\_performance(lin\_model,X\_train,Y\_train)

lin\_model\_train



45.x.columns





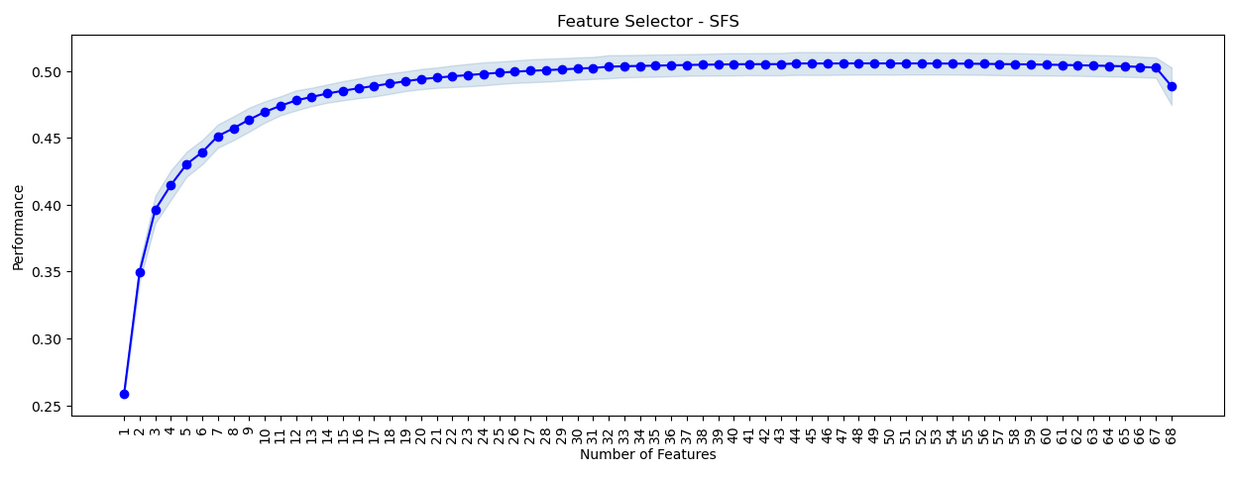
46.from mlxtend.plotting import plot\_sequential\_feature\_selection as plot\_sfs

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

plt.title("Feature Selector - SFS")

plt.xticks(rotation=90)

plt.show()

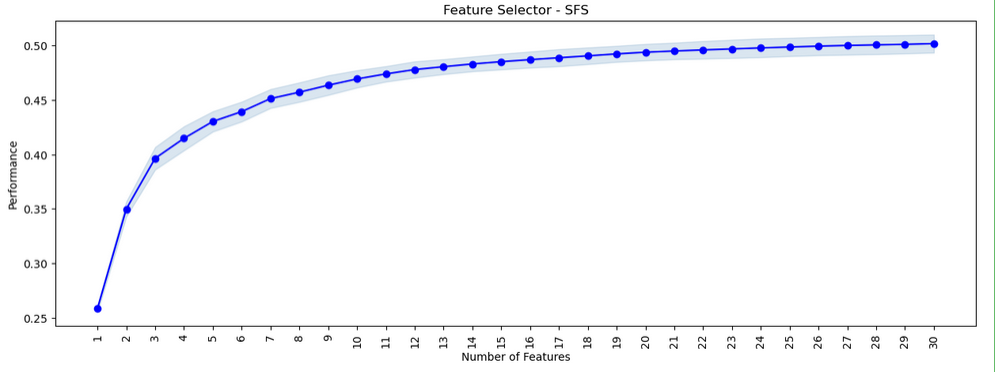


47.from mlxtend.feature\_selection import SequentialFeatureSelector as SFS

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)

48.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()



49.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)

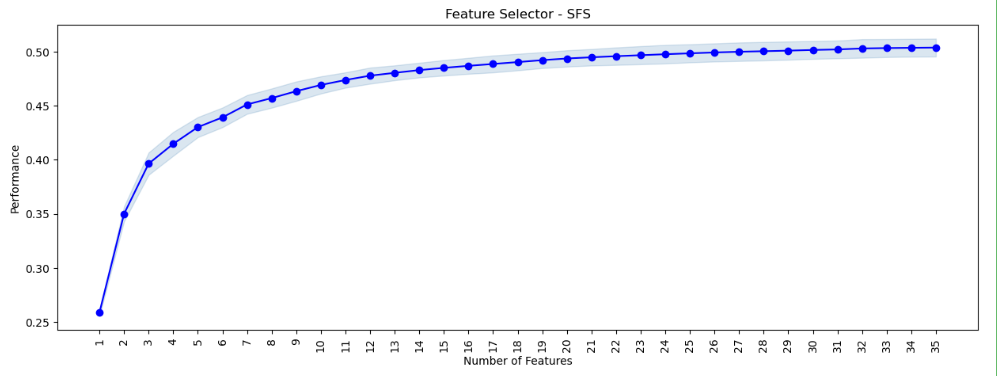
50.from mlxtend.plotting import plot\_sequential\_feature\_selection as plot\_sfs

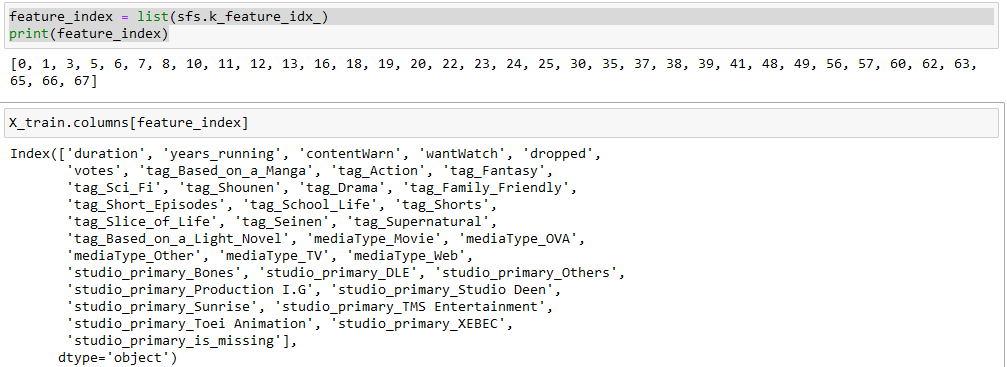
fig1 = plot\_sfs(sfs.get\_metric\_dict(), kind='std\_err', figsize=(15,5))

plt.title("Feature Selector - SFS")

plt.xticks(rotation=90)

plt.show()



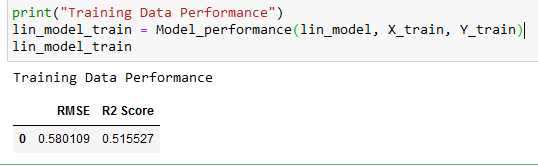


51.X\_train\_final = X\_train[X\_train.columns[feature\_index]]

52.X\_test\_final = X\_test[X\_test.columns[feature\_index]]

53.lin\_model\_v2 = LinearRegression()

54.lin\_model\_v2.fit(X\_train\_final, Y\_train)



55.lin\_model\_v2 = LinearRegression()

56.lin\_model\_v2.fit(X\_train\_final, Y\_train)

