WebSeries Analytics

**1.** import pandas as pd

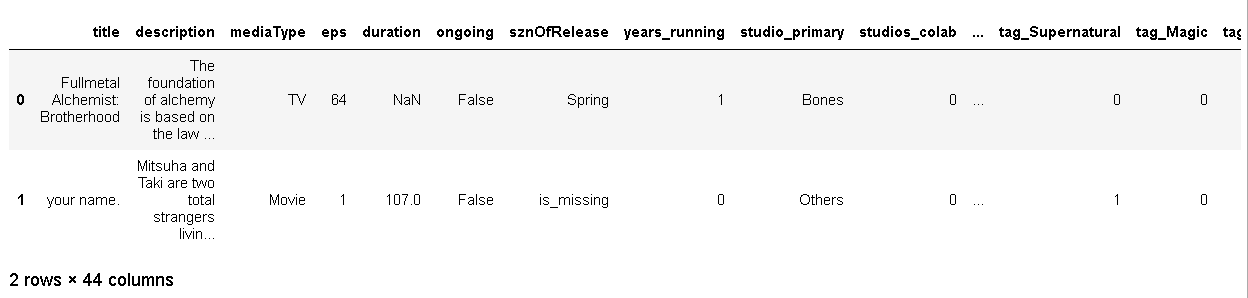
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns.

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

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

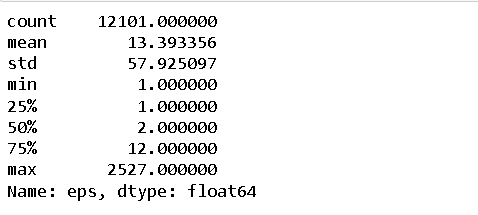


**4.dataset.shape**

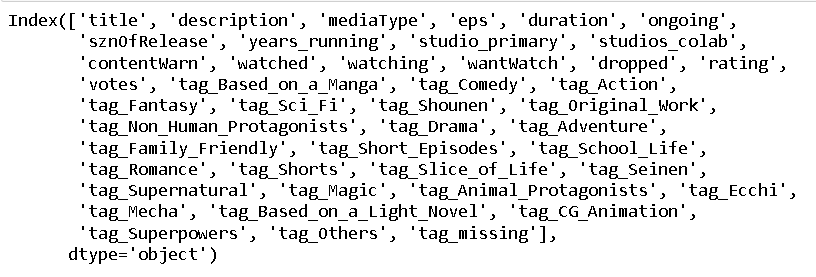
**Ans: (12101, 44)**

**In [5]:**

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

****

**6.dataset.columns**

****

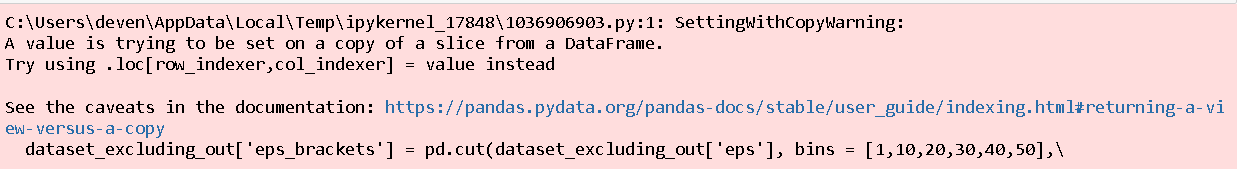
**7.dataset[(dataset['eps'] > 24) & (dataset.duration.isna())].shape**

**Ans: (1493, 44)**

**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],\**

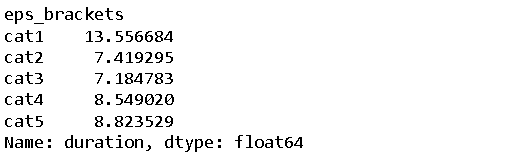
**labels = ['cat1','cat2','cat3','cat4','cat5'])**

****

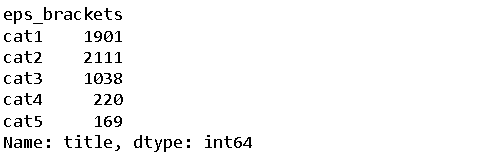
**10.dataset\_excluding\_out.shape**

**Ans:(11388, 45)**

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

****

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

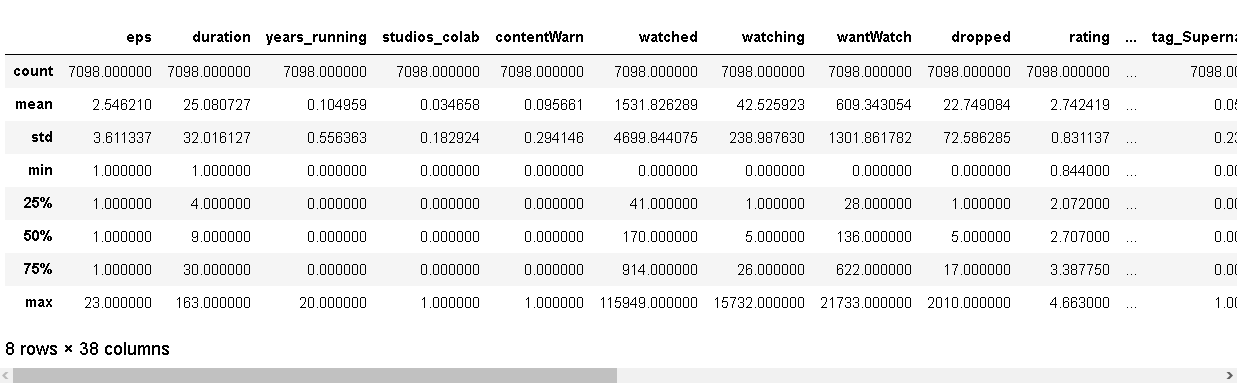
****

**13.dataset\_excluding\_out[dataset\_excluding\_out**

**['eps\_brackets'] == 'cat1'].shape.**

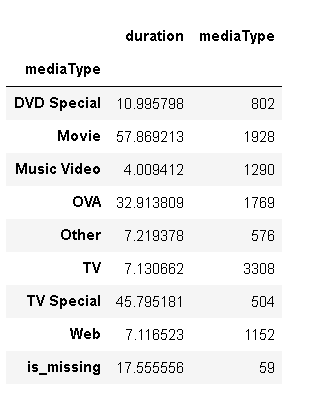
**Ans:(1901, 45)**

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

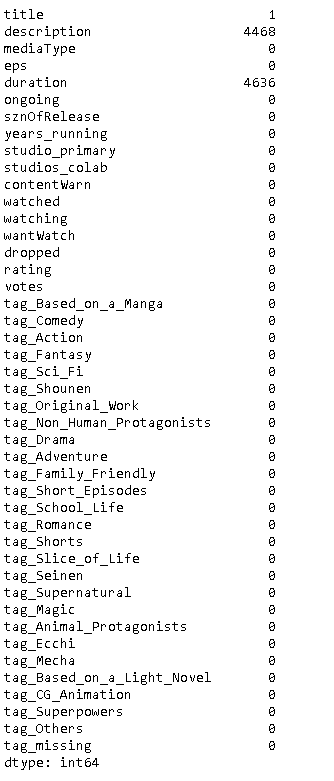
****

**15.dataset\_excluding\_out.groupby('mediaType')**

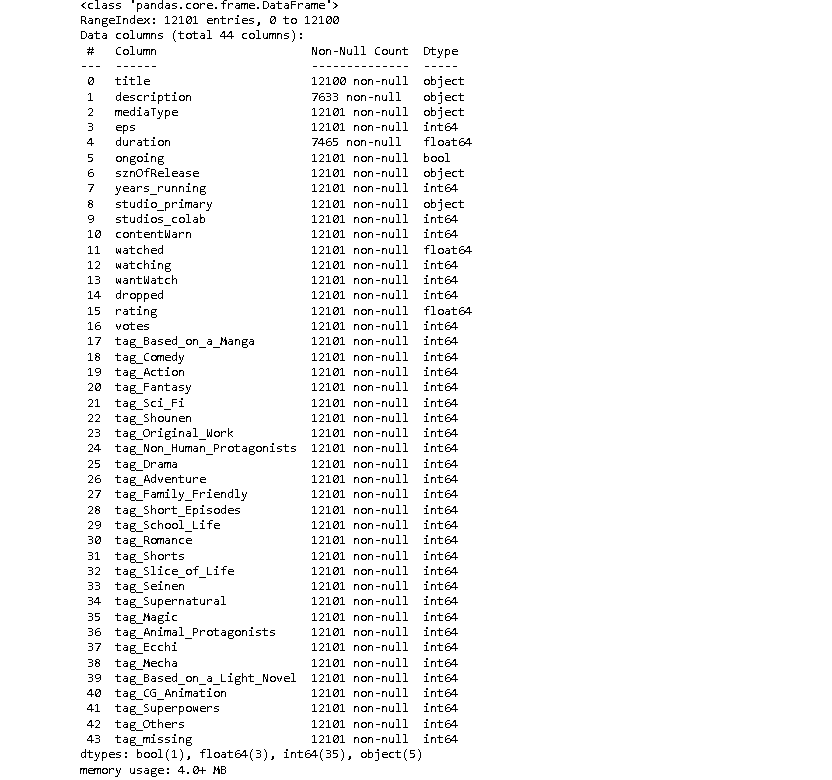
**.agg({'duration':'mean','mediaType':'count'})**

****

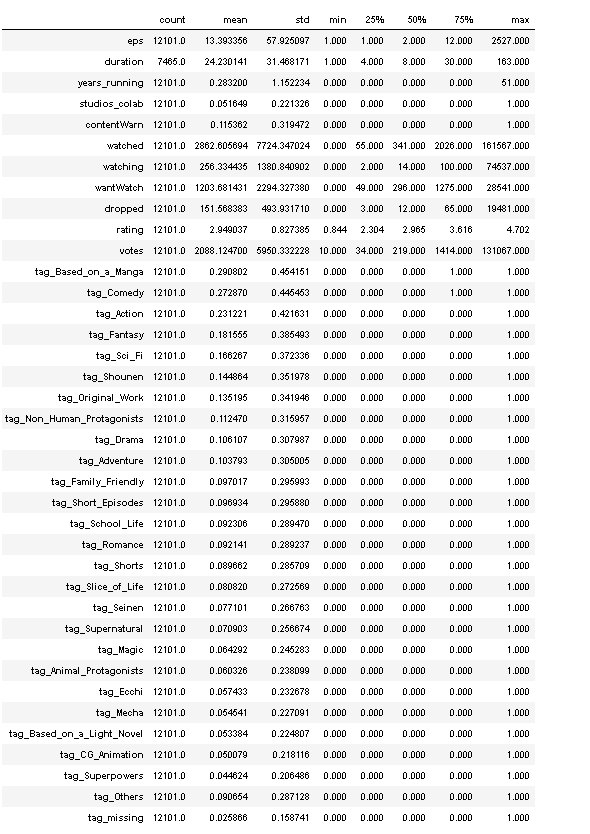
**16.dataset.isna().sum()**

****

**17.dataset.info()**

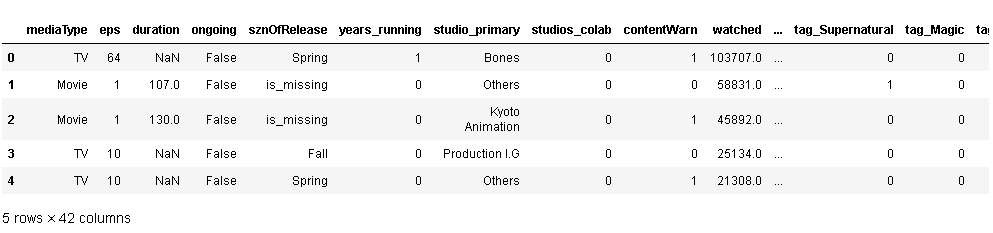
****

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

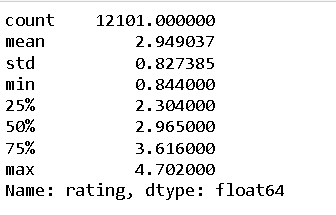
****

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

**20.dataset.head()**

****

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

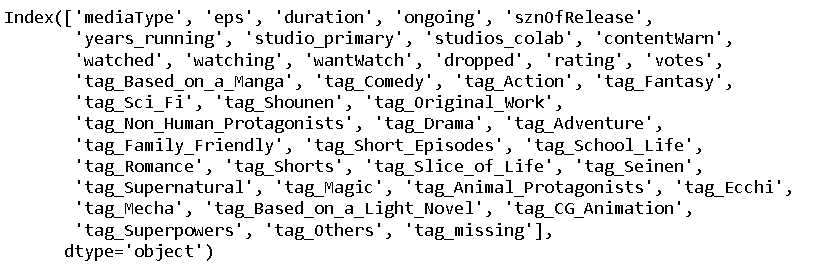
****

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

**dataset.shape**

**Ans:(7465, 42)**

**23.dataset.columns**

****

**24.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',**

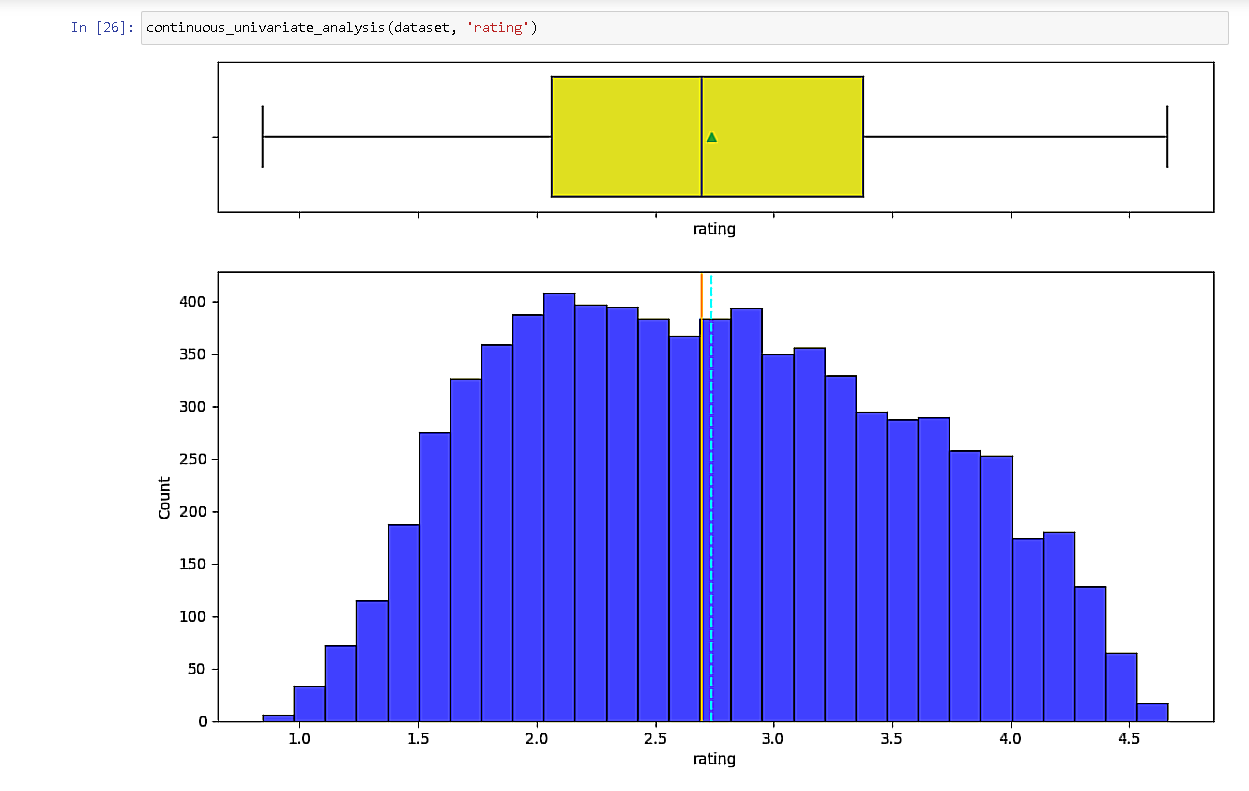
**bins=bins,**

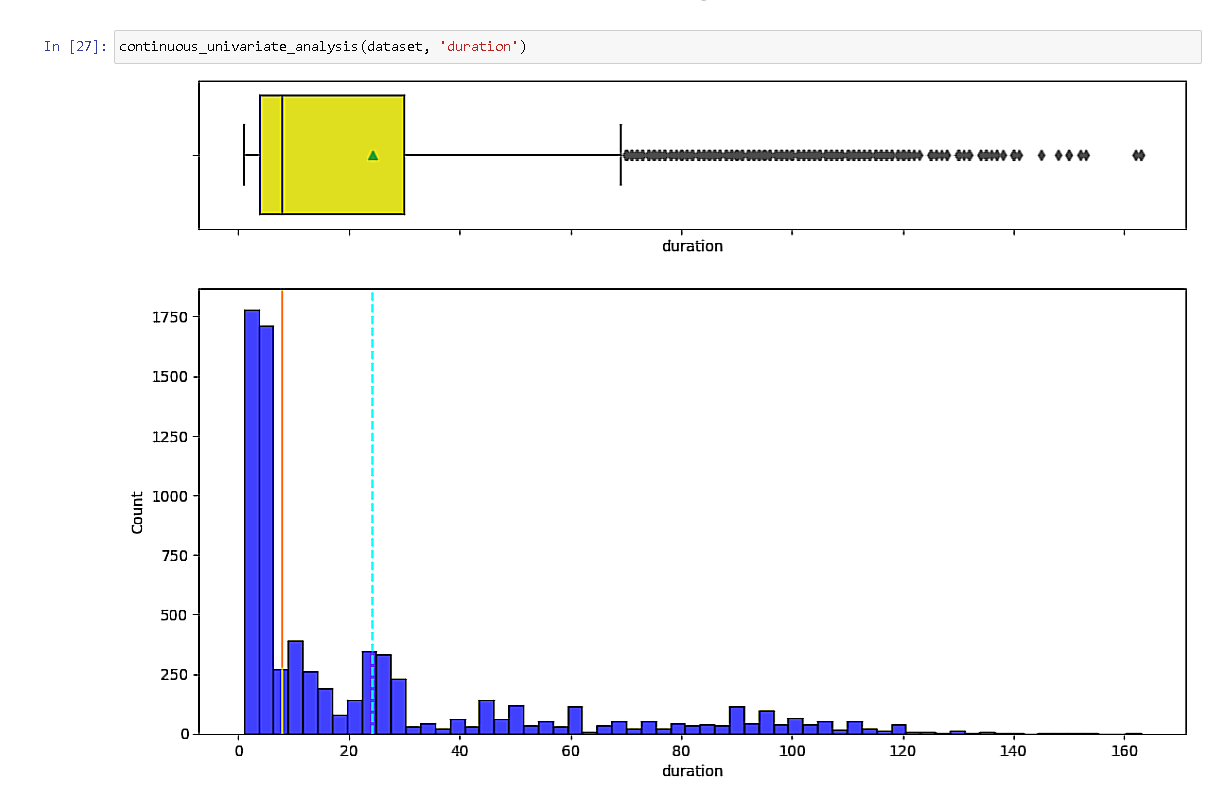
**kde=kde) if bins else sns.histplot(**

**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="-")**

****

****

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

**Ans:3.5694732254047326.**

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

**Ans:3.729269121813031.**

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

**Ans:3.7585191256830606.**

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

**Ans:2.7890469755469756.**

**29. 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",**

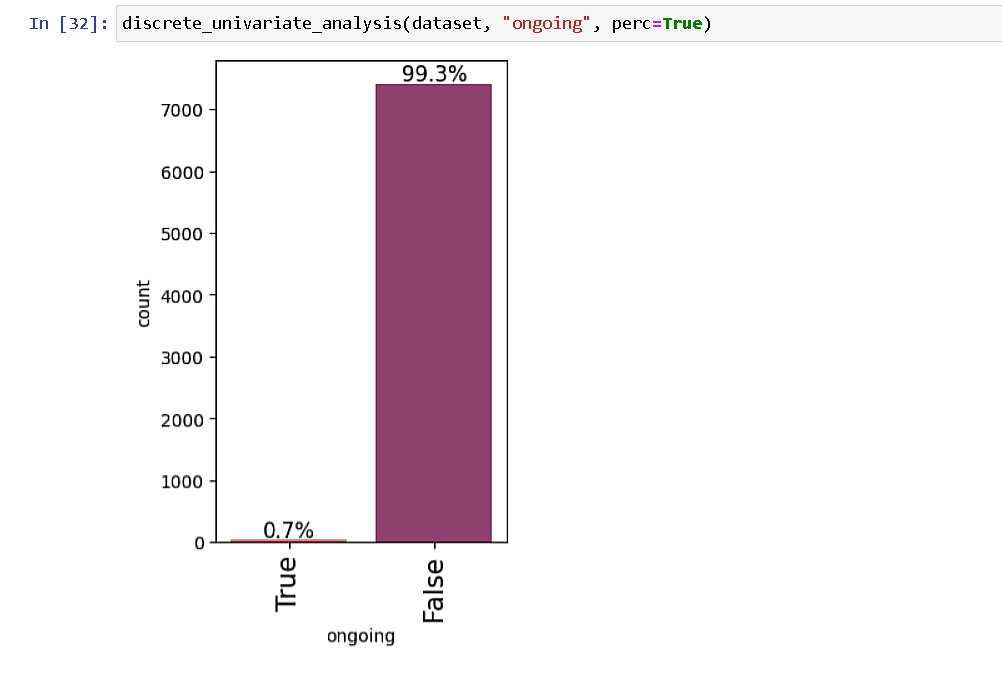
**va="center",**

**size=12,**

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

**textcoords="offset points")**

**plt.show()**

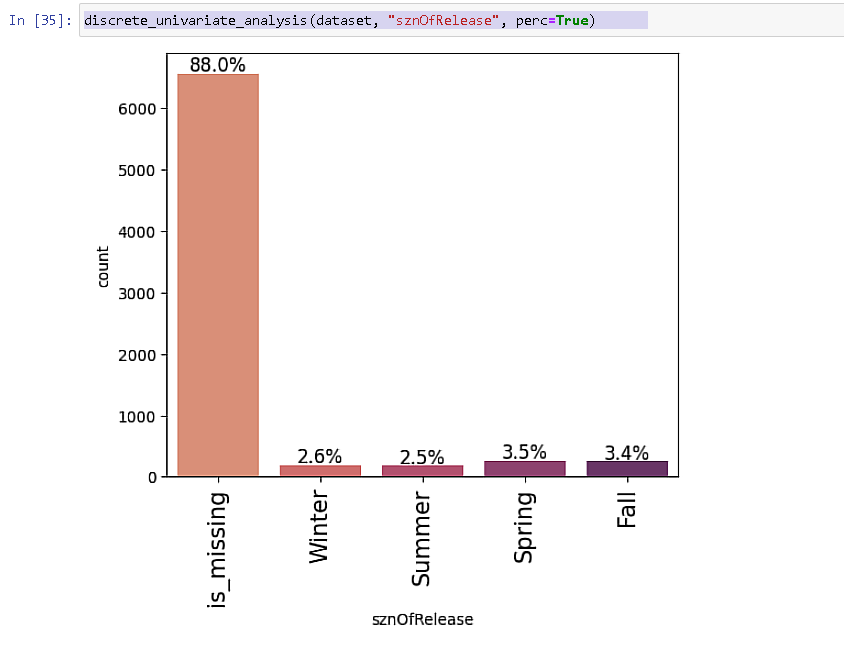
****

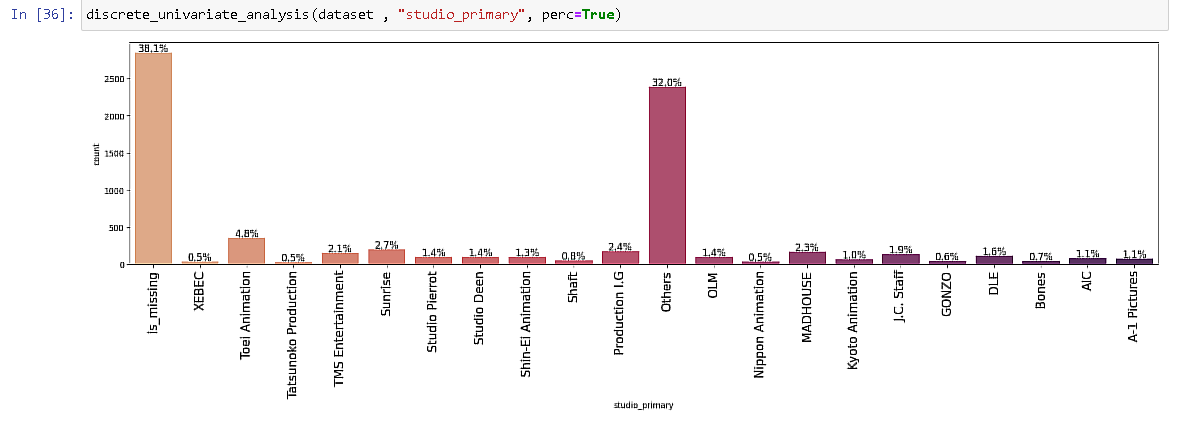
**30.dataset[dataset['ongoing'] == True]['rating'].mean()**

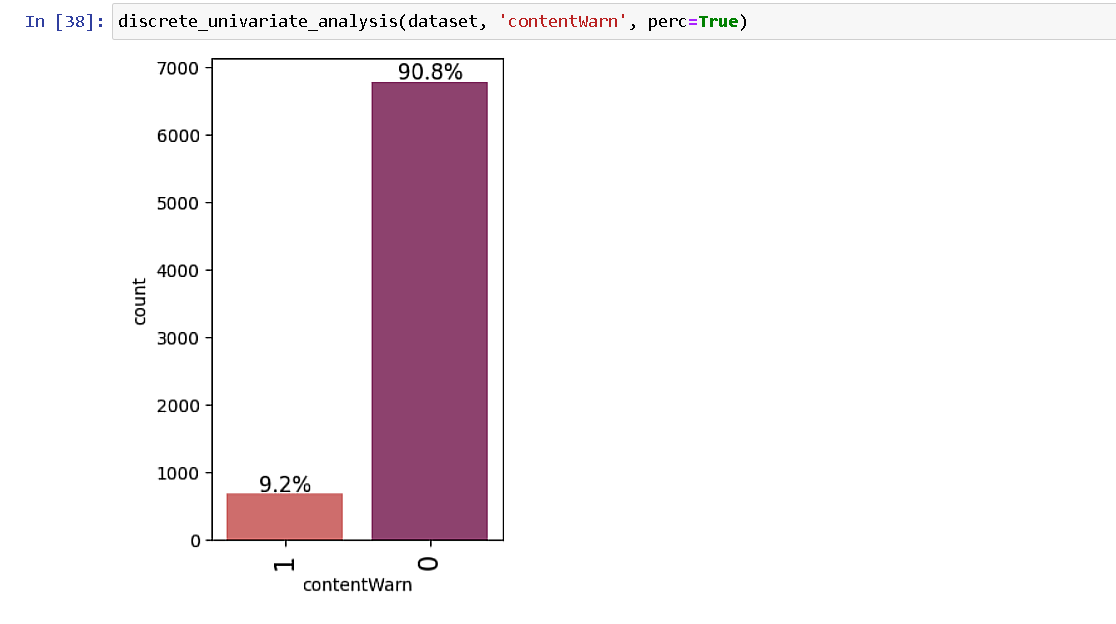
**Ans:3.1624600000000003**

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

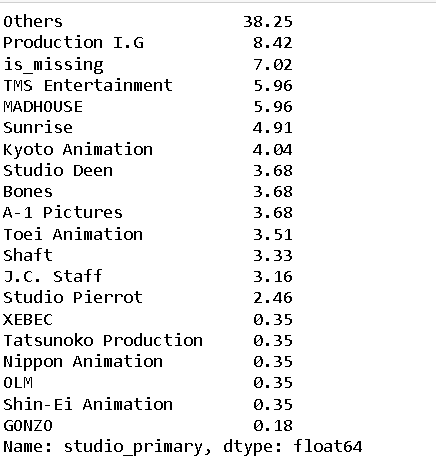
**Ans:8.94**

****

****

****

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

****

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

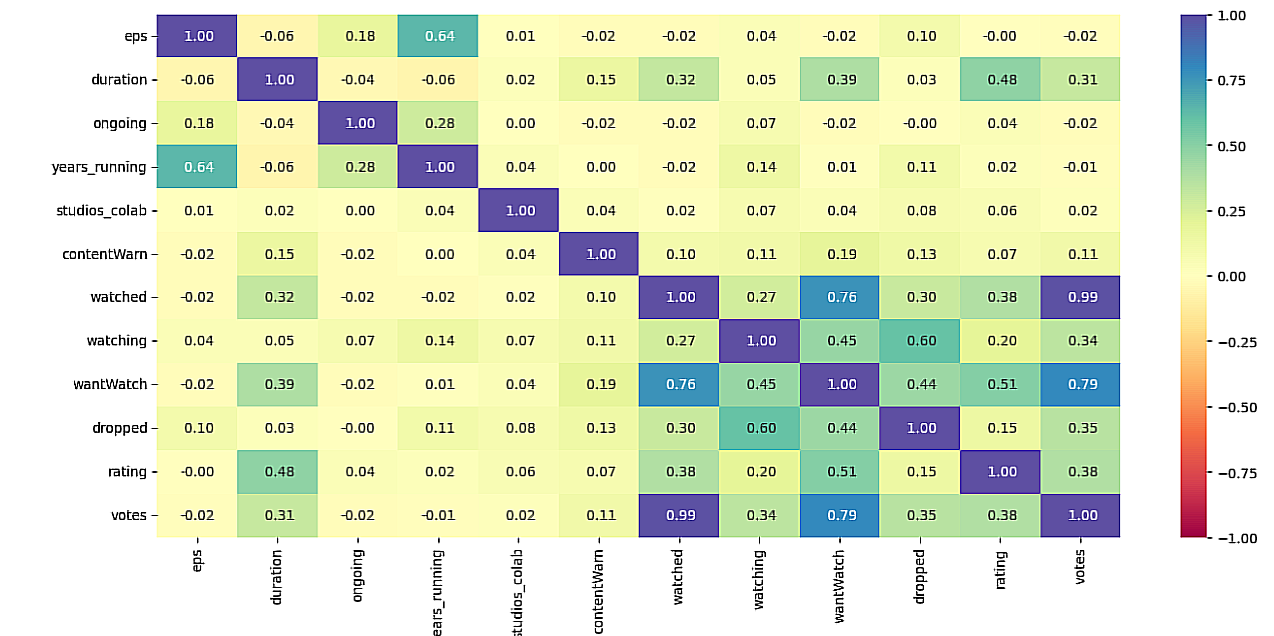
**34.corr\_cols//calling number 33 function**

****

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

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

**plt.show()**

****

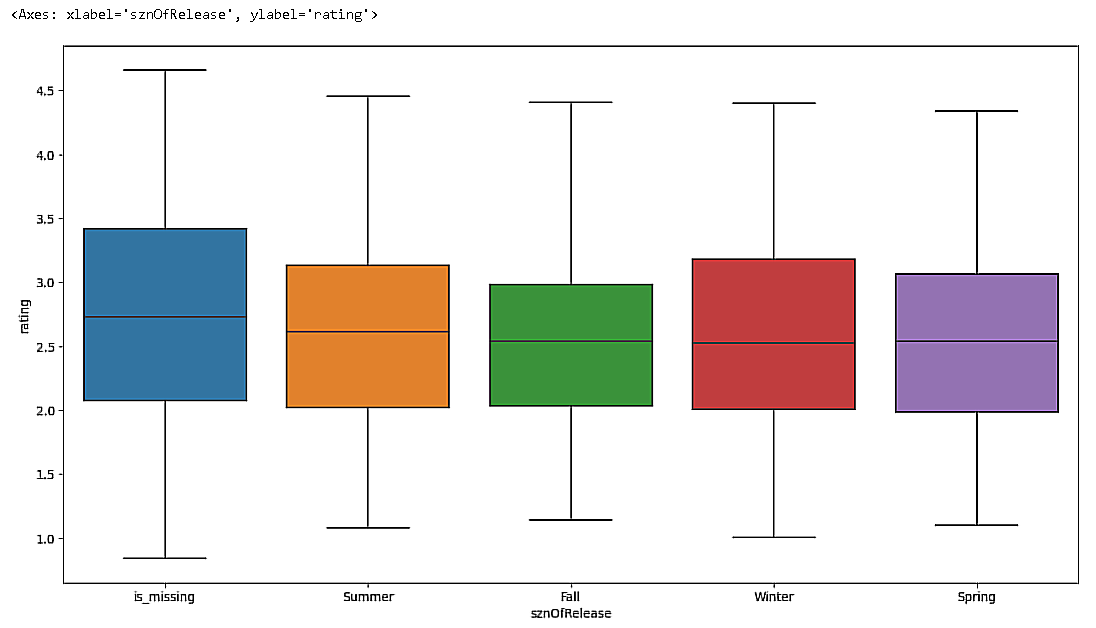
**36.dataset.drop(columns= ['eps', 'watched'], inplace=True)**

**37.dataset.shape**

**Ans:(7465, 40)**

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

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

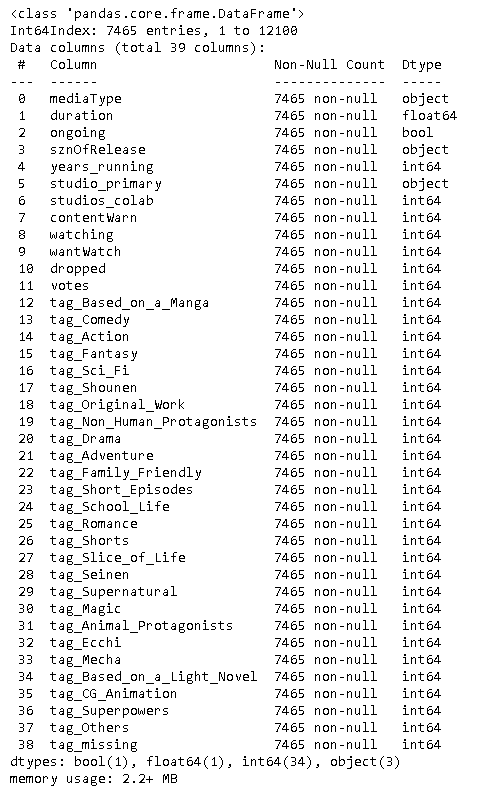
****

**##Model Building - Regression**

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

**y = dataset['rating']**

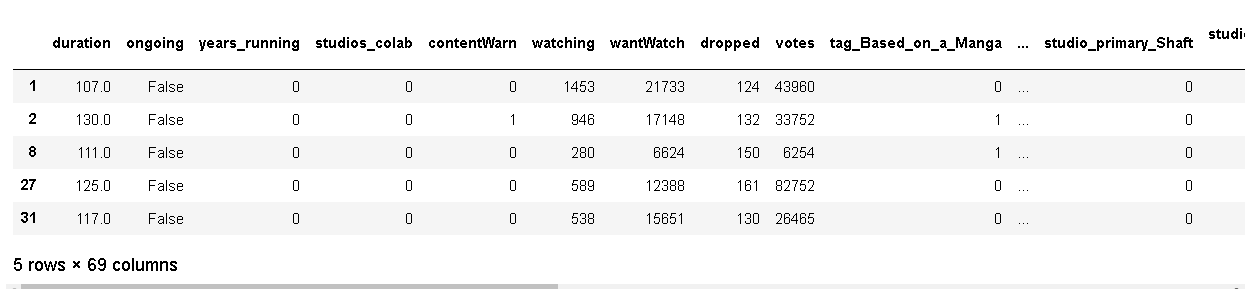
**40.x.info()**

****

**41.**

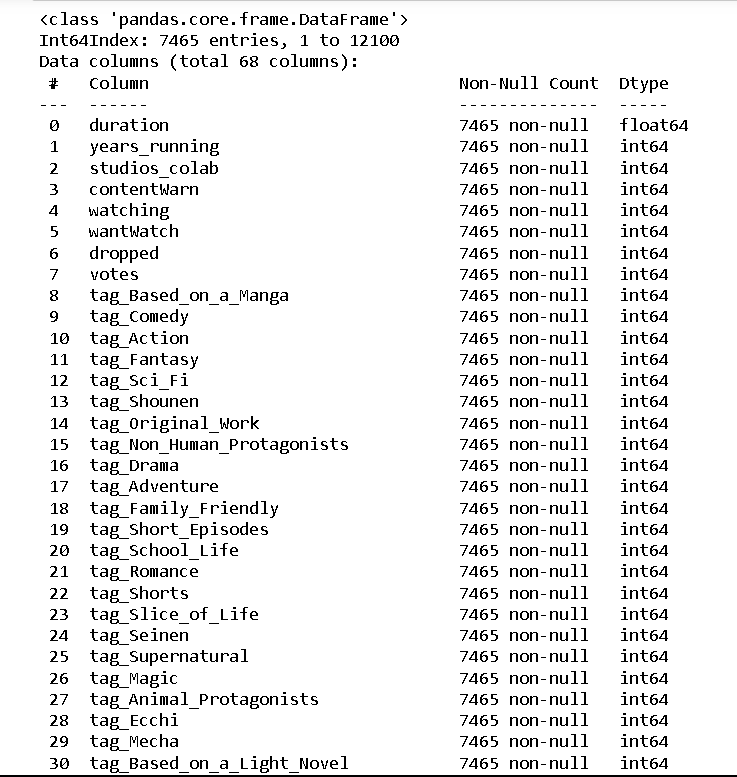
**x = pd.get\_dummies(x, columns=x.select\_dtypes(include=['object', 'category']).columns.tolist(), drop\_first=True)**

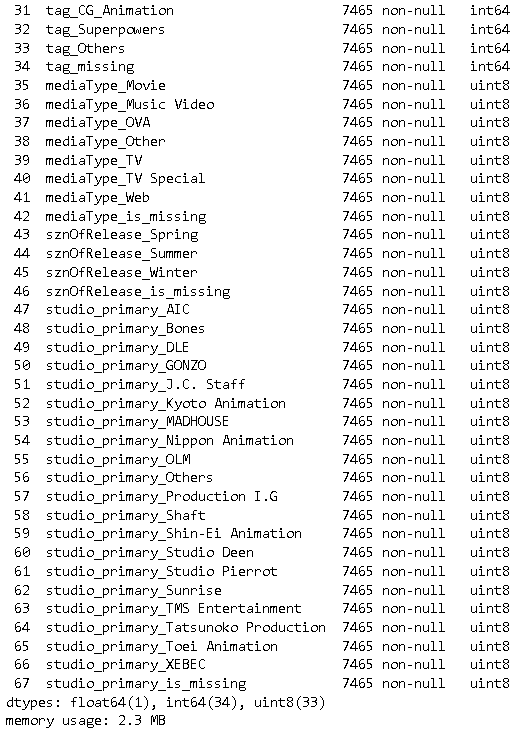
**x.head()**

****

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

**43.x.drop()**

****

****

**44.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**

**45. X\_train, X\_test, Y\_train, Y\_test = train\_test\_split**

**(x,y, test\_size=0.2, random\_state=1)**

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

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

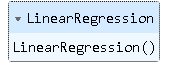
**Ans: Number of samples for train 5972**

**Number of samples for test 1493.**

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

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

**Ans:**

****

**48. 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]**

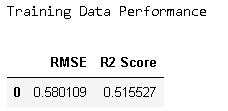
**)**

**return results**

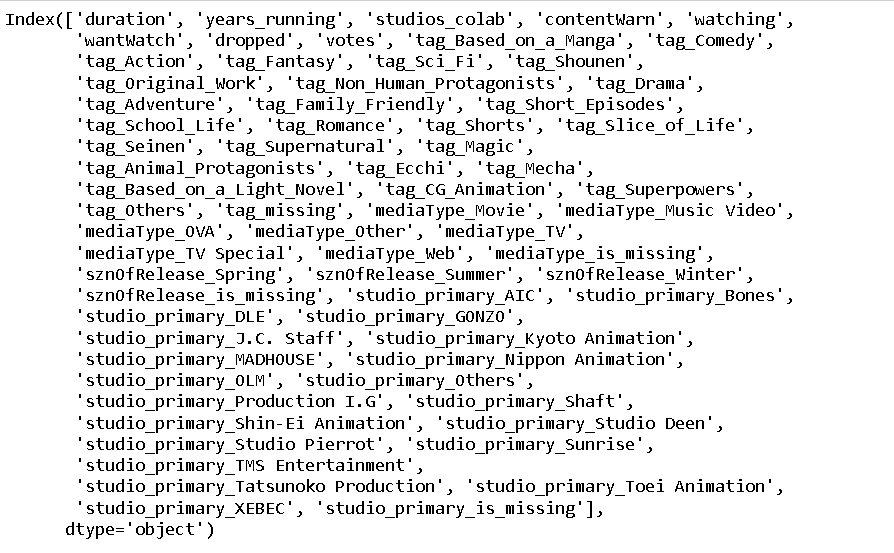
**49.print("Training Data Performance")**

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

**Lin\_model\_train**

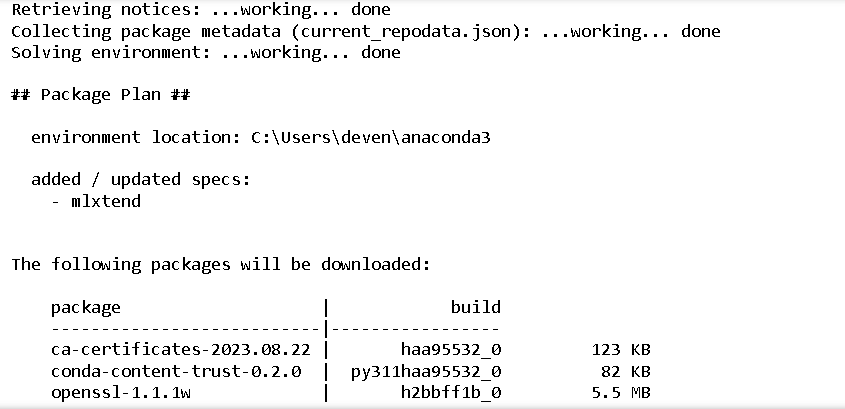
****

**50.x.columns**

****

**##feature selection technique**

**51.conda install mlxtend //downloading mlxtend**

****

**52.1.**

**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)**

**52.2.**

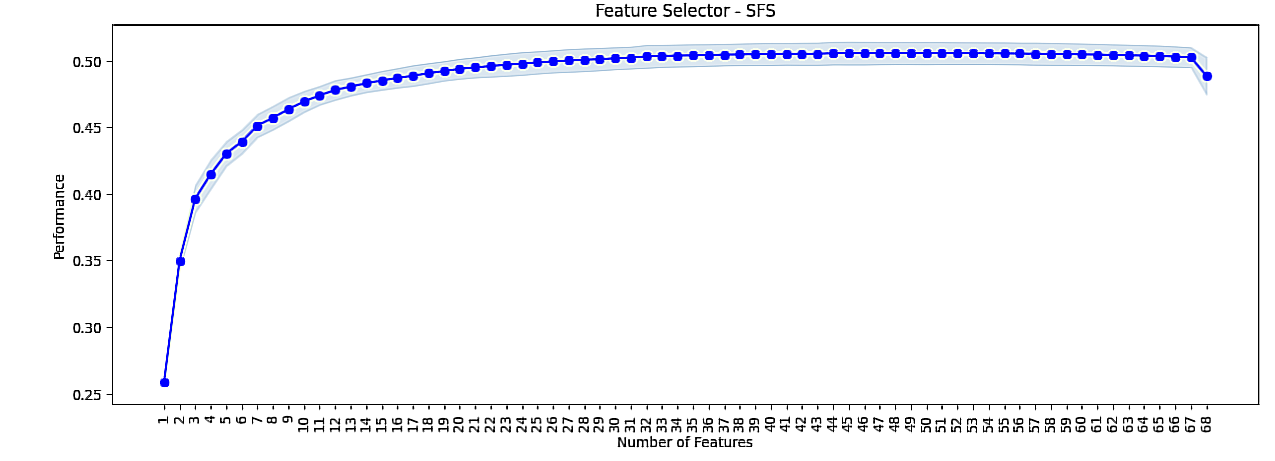
**from mlxtend.plotting import plot\_sequential\_feature\_selection as plot\_sfs**

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

**plt.title("Feature Selector - SFS")**

**plt.xticks(rotation=90)**

**plt.show()**

****

**53.1.**

**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)**

**53.2.**

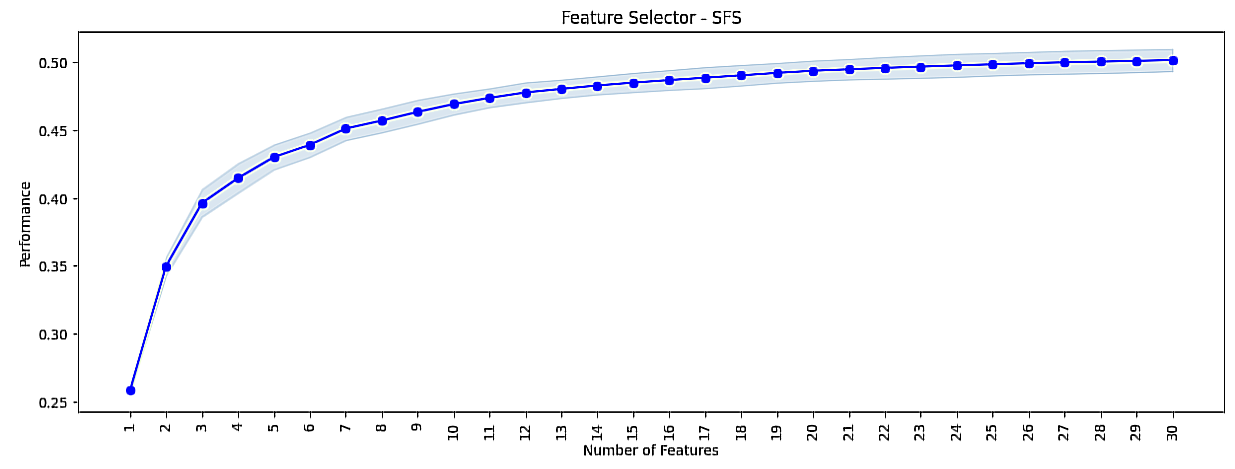
**from mlxtend.plotting import plot\_sequential\_feature\_selection as plot\_sfs**

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

**plt.title("Feature Selector - SFS")**

**plt.xticks(rotation=90)**

**plt.show()**

****

**54.1.**

**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)**

**54.2.**

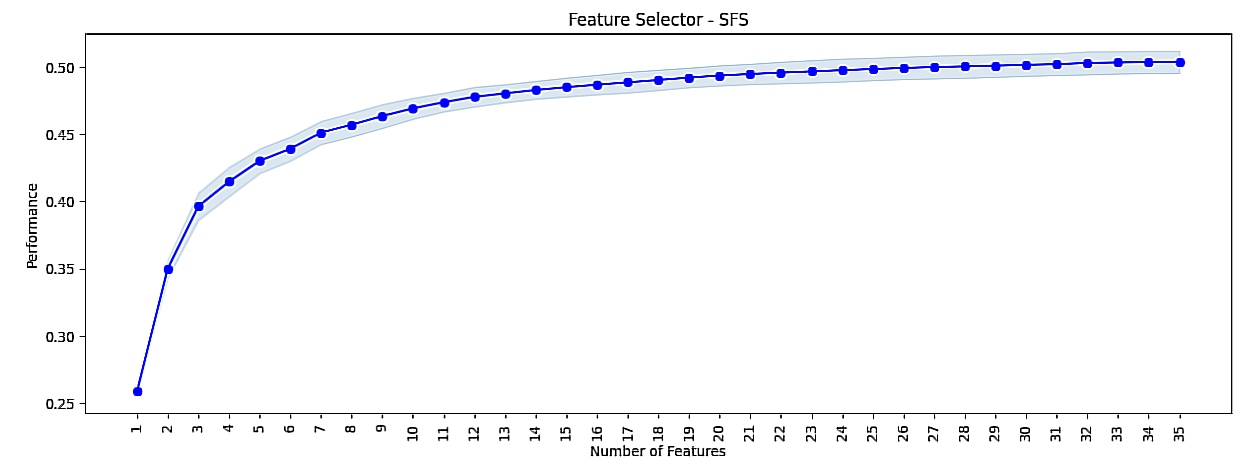
**from mlxtend.plotting import plot\_sequential\_feature\_selection as plot\_sfs**

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

**plt.title("Feature Selector - SFS")**

**plt.xticks(rotation=90)**

**plt.show()**

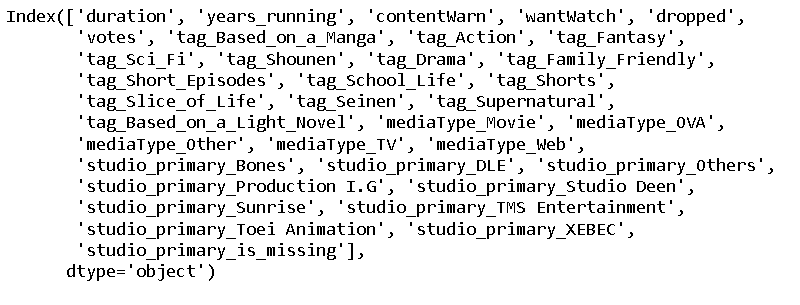
****

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

**print(feature\_index)**

****

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

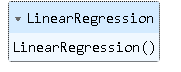
****

**57.X\_train\_final = X\_train[X\_train.columns[feature\_index]]**

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

**58.lin\_model\_v2 = LinearRegression()**

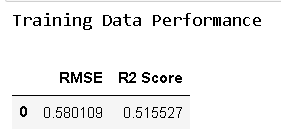
**lin\_model\_v2.fit(X\_train\_final, Y\_train)**

****

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

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

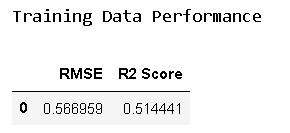
**lin\_model\_train**

****

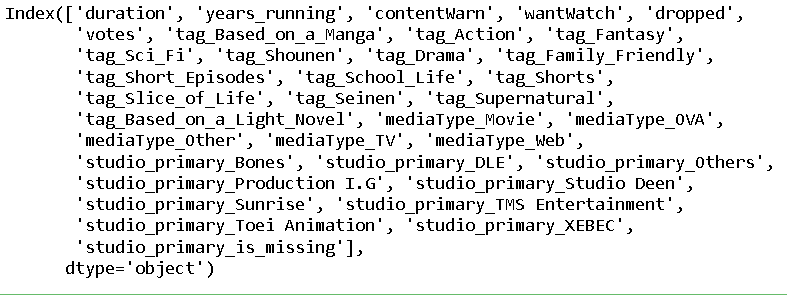
**60.print("Training Data Performance")**

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

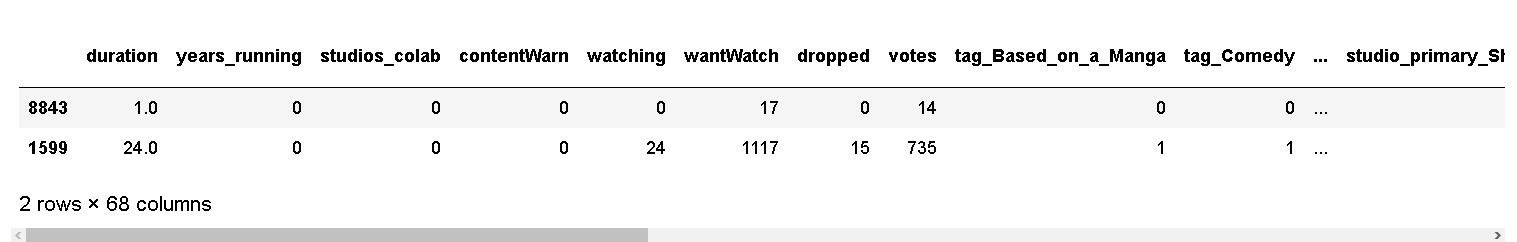
**Lin\_model\_train**

****

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

****

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

****

**Created by devendra ambre tybca sahyog college .**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***