```
from google.colab import drive
```

```
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_excel(r"/content/gdrive/My Drive/content/2255872-anime_data.xlsx")
```

```
df.head(2)
```

	title	description	mediaType	eps	duration	ongoing	sznOfRelease	years_running	studio_primary	studios_colab	...	tag_Superr
0	Fullmetal Alchemist: Brotherhood	The foundation of alchemy is based on the law ...	TV	64	NaN	False	Spring	1	Bones	0	...	
1	your name.	Mitsuha and Taki are two total strangers livin...	Movie	1	107.0	False	is_missing	0	Others	0	...	

2 rows × 44 columns

```
df.shape
```

```
(12101, 44)
```

```
df.eps.describe()
```

```
count    12101.000000
mean        13.393356
std         57.925097
min          1.000000
25%          1.000000
50%          2.000000
75%         12.000000
max       2527.000000
Name: eps, dtype: float64
```

```
df.columns
```

```
Index(['title', 'description', 'mediaType', 'eps', 'duration', 'ongoing',
       'sznOfRelease', 'years_running', 'studio_primary', 'studios_colab',
       'contentWarn', 'watched', 'watching', 'wantWatch', 'dropped', 'rating',
       'votes', 'tag_Based_on_a_Manga', 'tag_Comedy', 'tag_Action',
       'tag_Fantasy', 'tag_Sci_Fi', 'tag_Shounen', 'tag_Original_Work',
       'tag_Non_Human_Protagonists', 'tag_Drama', 'tag_Adventure',
       'tag_Family_Friendly', 'tag_Short_Episodes', 'tag_School_Life',
       'tag_Romance', 'tag_Shorts', 'tag_Slice_of_Life', 'tag_Seinen',
       'tag_Supernatural', 'tag_Magic', 'tag_Animal_Protagonists', 'tag_Ecchi',
       'tag_Mecha', 'tag_Based_on_a_Light_Novel', 'tag_CG_Animation',
       'tag_Superpowers', 'tag_Others', 'tag_missing'],
      dtype='object')
```

```
df[(df['eps']>24 ) & (df.duration.isna())].shape
```

```
(1493, 44)
```

```
df_excluding_out = df[df['eps']<50]
```

```
df_excluding_out['eps_brackets'] = pd.cut(df_excluding_out['eps'], bins=[1, 10, 20, 30, 40, 50], \
labels=['cats1', 'cats2', 'cats3', 'cats4', 'cats5'])
```

```
<ipython-input-21-d06f3ec059a4>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df_excluding_out['eps_brackets'] = pd.cut(df_excluding_out['eps'], bins=[1, 10, 20, 30, 40, 50], \
```

```
df_excluding_out .shape
```

```
(11388, 45)
```

```
df_excluding_out.groupby(['eps_brackets']).duration.mean()
```

```
eps_brackets
cats1    13.556684
cats2     7.419295
cats3     7.184783
cats4     8.549020
cats5     8.823529
Name: duration, dtype: float64
```

```
df_excluding_out.groupby(['eps_brackets']).title.count()
```

```
eps_brackets
cats1    1901
cats2    2111
cats3    1038
cats4     220
cats5     169
Name: title, dtype: int64
```

```
df_excluding_out[df_excluding_out['eps_brackets'] =='cats1'].shape
```

```
(1901, 45)
```

```
df[(df['eps']<24) &(~df.duration.isna())].describe()
```

	eps	duration	years_running	studios_colab	contentWarn	watched	watching	wantWatch	dropped	ra
count	7098.000000	7098.000000	7098.000000	7098.000000	7098.000000	7098.000000	7098.000000	7098.000000	7098.000000	7098.00
mean	2.546210	25.080727	0.104959	0.034658	0.095661	1531.826289	42.525923	609.343054	22.749084	2.74
std	3.611337	32.016127	0.556363	0.182924	0.294146	4699.844075	238.987630	1301.861782	72.586285	0.83
min	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.84
25%	1.000000	4.000000	0.000000	0.000000	0.000000	41.000000	1.000000	28.000000	1.000000	2.07
50%	1.000000	9.000000	0.000000	0.000000	0.000000	170.000000	5.000000	136.000000	5.000000	2.70
75%	1.000000	30.000000	0.000000	0.000000	0.000000	914.000000	26.000000	622.000000	17.000000	3.38
max	23.000000	163.000000	20.000000	1.000000	1.000000	115949.000000	15732.000000	21733.000000	2010.000000	4.66

8 rows × 38 columns

```
df_excluding_out.groupby(['mediaType']).agg({'duration':'mean','mediaType':'count'})
```

mediaType	duration	mediaType
DVD Special	10.995798	802

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

```
title                          1
description                 4468
mediaType                      0
eps                            0
duration                    4636
ongoing                        0
sznOfRelease                   0
years_running                  0
studio_primary                 0
studios_colab                  0
contentWarn                    0
watched                        0
watching                       0
wantWatch                      0
dropped                        0
rating                         0
votes                          0
tag_Based_on_a_Manga           0
tag_Comedy                     0
tag_Action                     0
tag_Fantasy                    0
tag_Sci_Fi                     0
tag_Shounen                    0
tag_Original_Work              0
tag_Non_Human_Protagonists     0
tag_Drama                      0
tag_Adventure                  0
tag_Family_Friendly            0
tag_Short_Episodes             0
tag_School_Life                0
tag_Romance                    0
tag_Shorts                     0
tag_Slice_of_Life              0
tag_Seinen                     0
tag_Supernatural               0
tag_Magic                      0
tag_Animal_Protagonists        0
tag_Ecchi                      0
tag_Mecha                      0
tag_Based_on_a_Light_Novel     0
tag_CG_Animation               0
tag_Superpowers                0
tag_Others                     0
tag_missing                    0
dtype: int64
```

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

```
df.head()
```

	mediaType	eps	duration	ongoing	sznOfRelease	years_running	studio_primary	studios_colab	contentWarn	watched	...	tag_Superna
0	TV	64	NaN	False	Spring	1	Bones	0	1	103707.0	...	
1	Movie	1	107.0	False	is_missing	0	Others	0	0	58831.0	...	
2	Movie	1	130.0	False	is_missing	0	Kyoto Animation	0	1	45892.0	...	
3	TV	10	NaN	False	Fall	0	Production I.G	0	0	25134.0	...	
4	TV	10	NaN	False	Spring	0	Others	0	1	21308.0	...	

5 rows × 42 columns

```
df.rating.describe()
```

```
count    12101.000000
mean         2.949037
std          0.827385
min          0.844000
25%          2.304000
50%          2.965000
75%          3.616000
```

```
max          4.702000
Name: rating, dtype: float64
```

```
df.dropna(inplace=True)
df.shape
```

```
(7465, 42)
```

```
12000-7465
```

```
4535
```

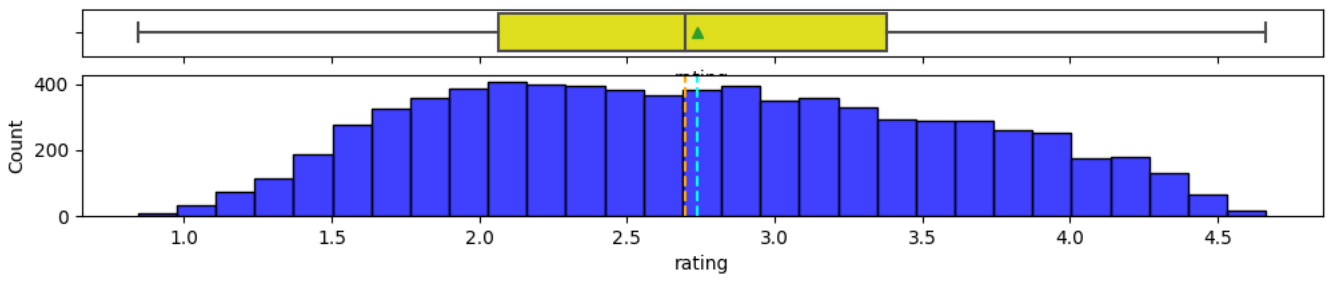
```
def continuous_univariate_analysis(data, feature, figsize=(12, 2), kde=False, bins=None):
    # Create subplots with shared x-axis
    f1, (ax_box, ax_hist) = plt.subplots(
        nrows=2,
        sharex=True,
        gridspec_kw={"height_ratios": (0.25, 0.75)},
        figsize=figsize
    )

    # Set color palette
    sns.color_palette("viridis", as_cmap=True)

    # Create a box plot
    sns.boxplot(data=data, x=feature, ax=ax_box, showmeans=True, color="yellow")

    # Create a histogram
    if bins:
        sns.histplot(data=data, x=feature, ax=ax_hist, showmeans=True, color="crest", bins=bins, kde=kde)
    else:
        sns.histplot(data=data, x=feature, ax=ax_hist, kde=kde, color="blue")

    # Add vertical lines for mean and median
    ax_hist.axvline(data[feature].mean(), color='cyan', linestyle='--')
    ax_hist.axvline(data[feature].median(), color='orange', linestyle='--')
```

```
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:
            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()
```

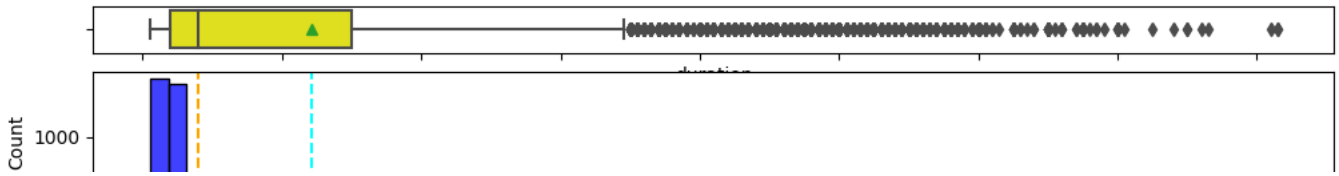
```
df.columns
```

```
Index(['mediaType', 'eps', 'duration', 'ongoing', 'sznOfRelease',
       'years_running', 'studio_primary', 'studios_colab', 'contentWarn',
       'watched', 'watching', 'wantWatch', 'dropped', 'rating', 'votes',
       'tag_Based_on_a_Manga', 'tag_Comedy', 'tag_Action', 'tag_Fantasy',
       'tag_Sci_Fi', 'tag_Shounen', 'tag_Original_Work',
       'tag_Non_Human_Protagonists', 'tag_Drama', 'tag_Adventure',
       'tag_Family_Friendly', 'tag_Short_Episodes', 'tag_School_Life',
       'tag_Romance', 'tag_Shorts', 'tag_Slice_of_Life', 'tag_Seinen',
       'tag_Supernatural', 'tag_Magic', 'tag_Animal_Protagonists', 'tag_Ecchi',
       'tag_Mecha', 'tag_Based_on_a_Light_Novel', 'tag_CG_Animation',
       'tag_Superpowers', 'tag_Others', 'tag_missing'],
      dtype='object')
```

```
continuous_univariate_analysis(df,'rating')
```




```
continuous_univariate_analysis(df,'duration')
```




```
df[df['duration']>=80]['rating'].mean()
```

```
3.5694732254047326
```

```
df[df['duration']>=100]['rating'].mean()
```

```
3.729269121813031
```

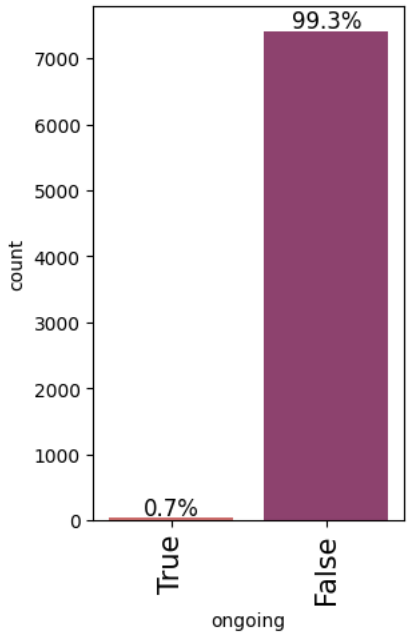
```
df[df['duration']>=110]['rating'].mean()
```

```
3.7585191256830606
```

```
df[(df['duration']>=5) & (df['duration']>=30)]['rating'].mean()
```

```
3.270219668626403
```

```
discrete_univariate_analysis(df,"ongoing",perc=True)
```



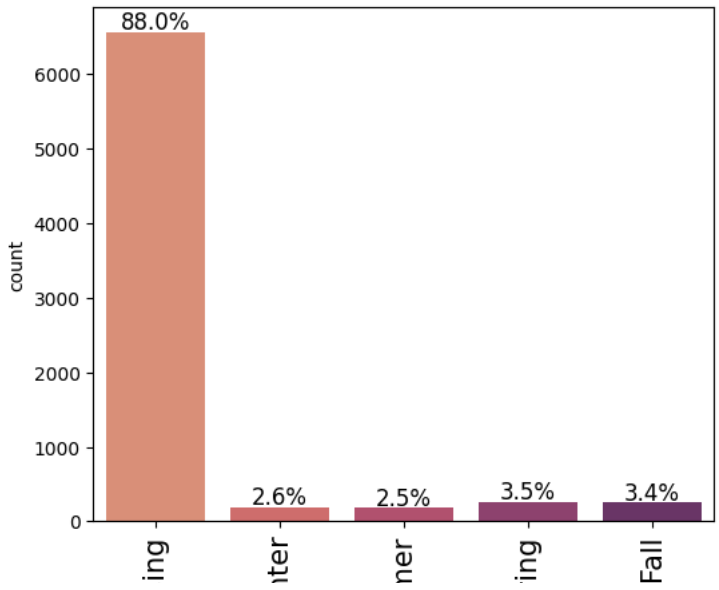

```
df[df['ongoing']== True]['rating'].mean()
```

```
3.1624600000000003
```

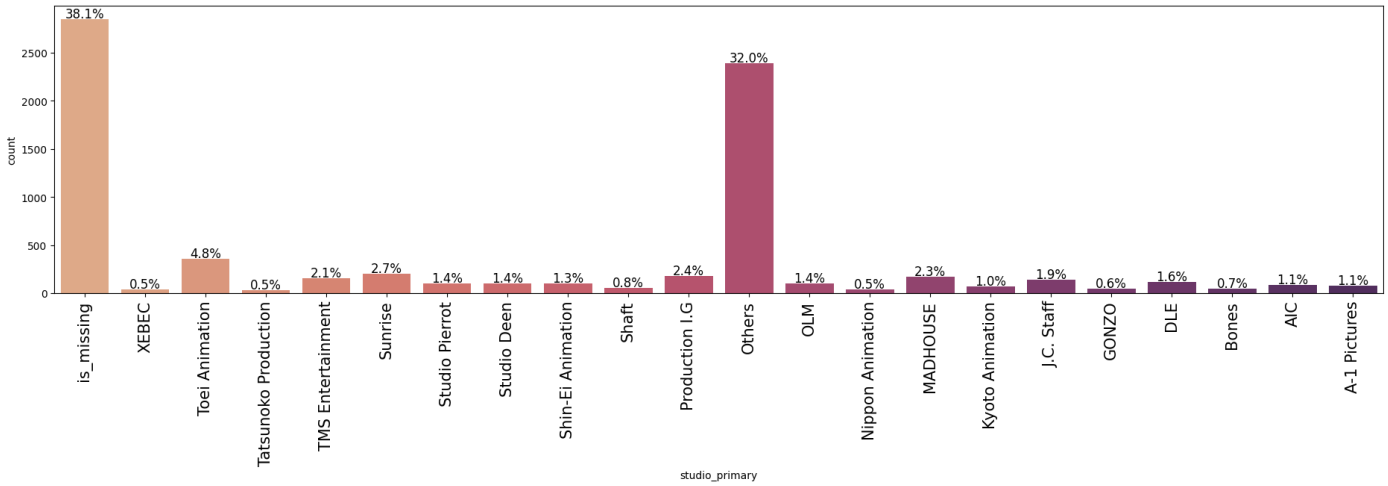
```
df[df['ongoing']== True]['duration'].mean()
```

```
8.94
```

```
discrete_univariate_analysis(df,"sznOfRelease",perc=True)
```



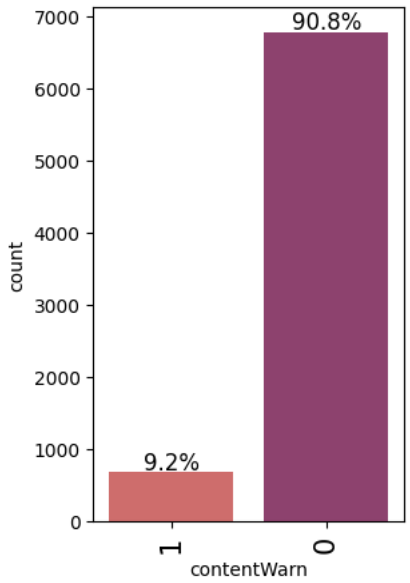

```
discrete_univariate_analysis(df,"studio_primary",perc=True)
```




```
df[df['rating']>4]['studio_primary'].value_counts(normalize = True).mul(100).round(2)
```

```
Others                38.25
Production I.G         8.42
is_missing             7.02
TMS Entertainment      5.96
MADHOUSE               5.96
Sunrise                4.91
Kyoto Animation        4.04
Studio Deen            3.68
Bones                  3.68
A-1 Pictures           3.68
Toei Animation         3.51
Shaft                  3.33
J.C. Staff             3.16
Studio Pierrot         2.46
XEBEC                  0.35
Tatsunoko Production   0.35
Nippon Animation       0.35
OLM                    0.35
Shin-Ei Animation      0.35
GONZO                  0.18
Name: studio_primary, dtype: float64
```

```
discrete_univariate_analysis(df,"contentWarn",perc=True)
```



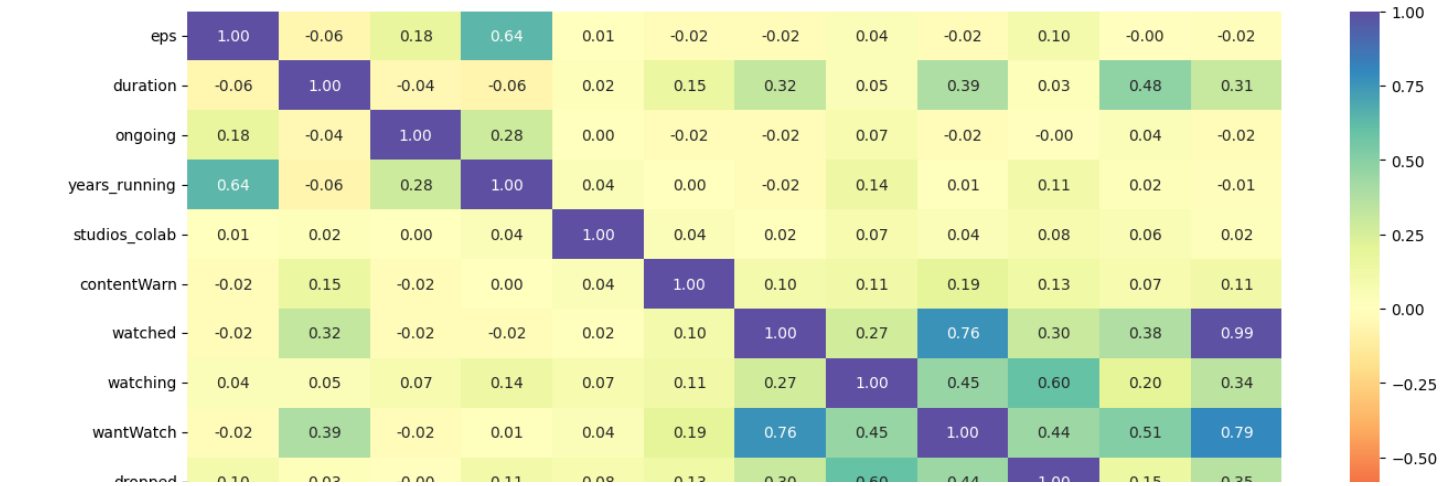

```
corr_cols =[item for item in df.columns if "tag" not in item]
```

```
corr_cols
```

```
['mediaType',
 'eps',
 'duration',
 'ongoing',
 'sznOfRelease',
 'years_running',
 'studio_primary',
 'studios_colab',
 'contentWarn',
 'watched',
 'watching',
 'wantWatch',
 'dropped',
 'rating',
 'votes']
```

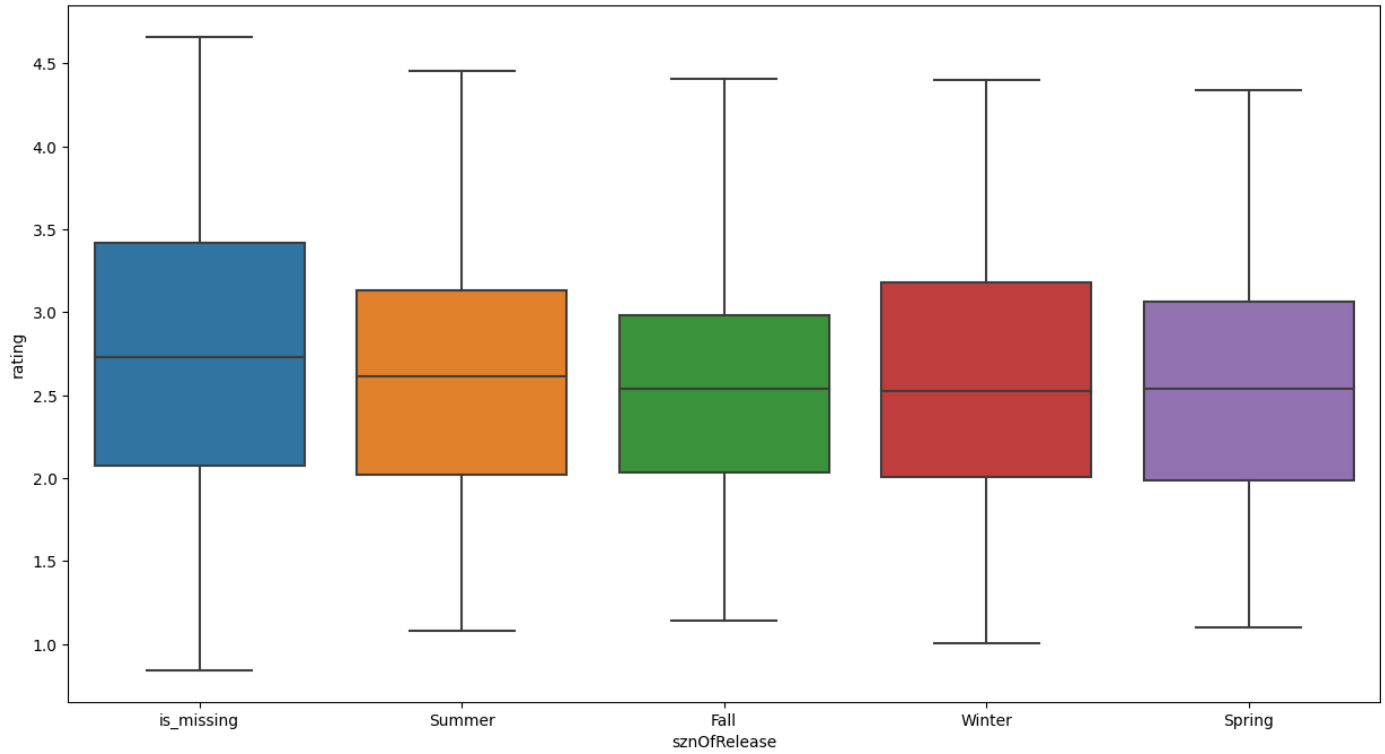
```
corr_cols = [col for col in corr_cols if pd.api.types.is_numeric_dtype(df[col])]
```

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

sns.heatmap(df[corr_cols].corr(), annot=True, vmin=-1, vmax =1, fmt='.2f', cmap='Spectral')

plt.show()
```




```
plt.figure(figsize=(15,8))
sns.boxplot(x='sznOfRelease', y='rating',data=df)
```

```
<Axes: xlabel='sznOfRelease', ylabel='rating'>
```



Model Building - Regression

```
x=df.drop(['rating'] ,axis=1)
y=df['rating']
```

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7465 entries, 1 to 12100
Data columns (total 41 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   mediaType                 7465 non-null   object
```

```
 1   eps                      7465 non-null   int64
 2   duration                 7465 non-null   float64
 3   ongoing                  7465 non-null   bool
 4   sznOfRelease             7465 non-null   object
 5   years_running            7465 non-null   int64
 6   studio_primary           7465 non-null   object
 7   studios_colab            7465 non-null   int64
 8   contentWarn              7465 non-null   int64
 9   watched                  7465 non-null   float64
 10  watching                 7465 non-null   int64
 11  wantWatch                7465 non-null   int64
 12  dropped                  7465 non-null   int64
 13  votes                    7465 non-null   int64
 14  tag_Based_on_a_Manga     7465 non-null   int64
 15  tag_Comedy               7465 non-null   int64
 16  tag_Action               7465 non-null   int64
 17  tag_Fantasy              7465 non-null   int64
 18  tag_Sci_Fi               7465 non-null   int64
 19  tag_Shounen              7465 non-null   int64
 20  tag_Original_Work        7465 non-null   int64
 21  tag_Non_Human_Protagonists  7465 non-null   int64
 22  tag_Drama                7465 non-null   int64
 23  tag_Adventure            7465 non-null   int64
 24  tag_Family_Friendly      7465 non-null   int64
 25  tag_Short_Episodes       7465 non-null   int64
 26  tag_School_Life          7465 non-null   int64
 27  tag_Romance              7465 non-null   int64
 28  tag_Shorts               7465 non-null   int64
 29  tag_Slice_of_Life        7465 non-null   int64
 30  tag_Seinen               7465 non-null   int64
 31  tag_Supernatural         7465 non-null   int64
 32  tag_Magic                7465 non-null   int64
 33  tag_Animal_Protagonists  7465 non-null   int64
 34  tag_Ecchi                7465 non-null   int64
 35  tag_Mecha                7465 non-null   int64
 36  tag_Based_on_a_Light_Novel  7465 non-null   int64
 37  tag_CG_Animation         7465 non-null   int64
 38  tag_Superpowers          7465 non-null   int64
 39  tag_Others               7465 non-null   int64
 40  tag_missing              7465 non-null   int64
dtypes: bool(1), float64(2), int64(35), object(3)
memory usage: 2.3+ MB
```

```
x = pd.get_dummies(x, columns=x.select_dtypes(include=['object', 'category']).columns.tolist(), drop_first=True)
x.head()
```

	eps	duration	ongoing	years_running	studios_colab	contentWarn	watched	watching	wantWatch	dropped	...	studio_primary_Shaft
1	1	107.0	False	0	0	0	58831.0	1453	21733	124	...	0
2	1	130.0	False	0	0	1	45892.0	946	17148	132	...	0
8	1	111.0	False	0	0	0	8454.0	280	6624	150	...	0
27	1	125.0	False	0	0	0	115949.0	589	12388	161	...	0
31	1	117.0	False	0	0	0	35896.0	538	15651	130	...	0

5 rows × 71 columns

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

```
x.info()
```

```
 28  tag_Magic                          7465 non-null   int64
 29  tag_Animal_Protagonists            7465 non-null   int64
 30  tag_Ecchi                          7465 non-null   int64
 31  tag_Mecha                          7465 non-null   int64
 32  tag_Based_on_a_Light_Novel         7465 non-null   int64
 33  tag_CG_Animation                   7465 non-null   int64
 34  tag_Superpowers                    7465 non-null   int64
 35  tag_Others                         7465 non-null   int64
 36  tag_missing                        7465 non-null   int64
 37  mediaType_Movie                    7465 non-null   uint8
 38  mediaType_Music Video              7465 non-null   uint8
 39  mediaType_OVA                      7465 non-null   uint8
 40  mediaType_Other                    7465 non-null   uint8
 41  mediaType_TV                       7465 non-null   uint8
 42  mediaType_TV Special               7465 non-null   uint8
 43  mediaType_Web                      7465 non-null   uint8
 44  mediaType_is_missing               7465 non-null   uint8
 45  sznOfRelease_Spring                7465 non-null   uint8
 46  sznOfRelease_Summer                7465 non-null   uint8
 47  sznOfRelease_Winter                7465 non-null   uint8
 48  sznOfRelease_is_missing            7465 non-null   uint8
 49  studio_primary_AIC                 7465 non-null   uint8
 50  studio_primary_Bones               7465 non-null   uint8
 51  studio_primary_DLE                 7465 non-null   uint8
 52  studio_primary_GONZO               7465 non-null   uint8
 53  studio_primary_J.C. Staff          7465 non-null   uint8
 54  studio_primary_Kyoto Animation     7465 non-null   uint8
 55  studio_primary_MADHOUSE            7465 non-null   uint8
 56  studio_primary_Nippon Animation    7465 non-null   uint8
 57  studio_primary_OLM                 7465 non-null   uint8
 58  studio_primary_Others              7465 non-null   uint8
 59  studio_primary_Production I.G      7465 non-null   uint8
 60  studio_primary_Shaft               7465 non-null   uint8
 61  studio_primary_Shin-Ei Animation   7465 non-null   uint8
 62  studio_primary_Studio Deen         7465 non-null   uint8
 63  studio_primary_Studio Pierrot      7465 non-null   uint8
 64  studio_primary_Sunrise             7465 non-null   uint8
 65  studio_primary_TMS Entertainment   7465 non-null   uint8
 66  studio_primary_Tatsunoko Production 7465 non-null  uint8
 67  studio_primary_Toei Animation      7465 non-null   uint8
 68  studio_primary_XEBEC               7465 non-null   uint8
 69  studio_primary_is_missing          7465 non-null   uint8
dtypes: float64(2), int64(35), uint8(33)
memory usage: 2.4 MB
```

```
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


X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2, random_state=1)
```

```
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
```

▾ LinearRegression
LinearRegression()

```
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


print("Training Data Performance")
```

```
lin_model_train = Model_performance(lin_model, X_train, Y_train)

print(lin_model_train)
```

```
     Training Data Performance
            RMSE  R2 Score
     0  0.578282  0.518574
```

```
print("Test Data Performance")

lin_model_test = Model_performance(lin_model, X_test, Y_test)

print(lin_model_test)
```

```
     Test Data Performance
            RMSE  R2 Score
     0  0.564057  0.519399
```

Feature Selection technique

1. Sequential Search - small amount of data
2. Grid Search - it is very expensive search algo
3. Baselan Search faster but it overfits

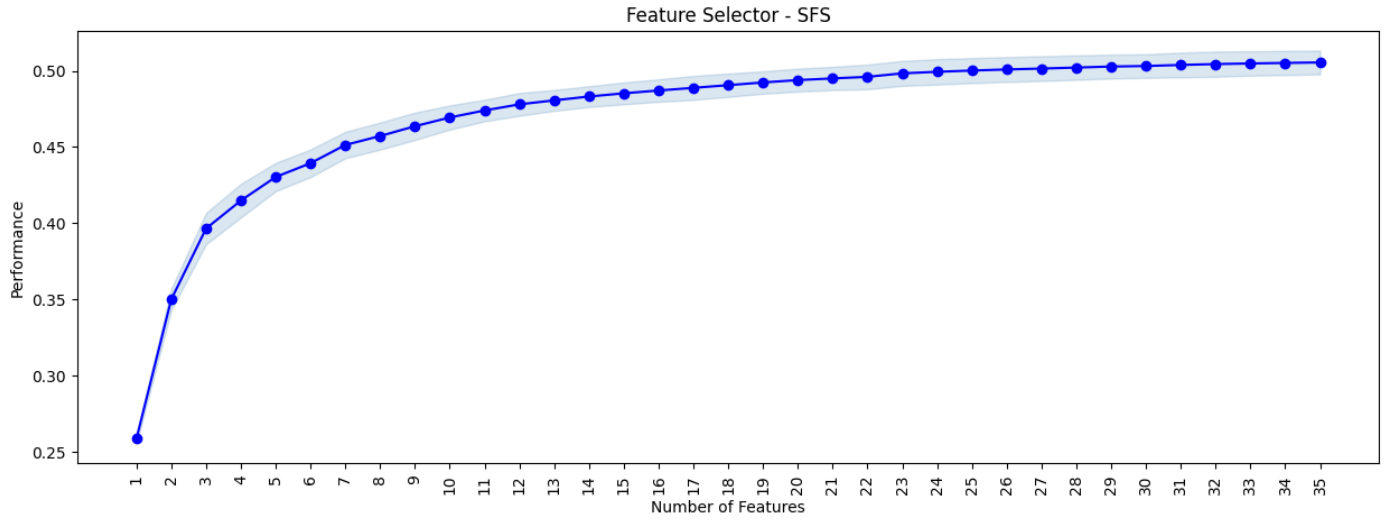
```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS

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)
```

```
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs

fig = plot_sfs(sfs.get_metric_dict(), kind='std_err', figsize=(15, 5))
plt.title('Feature Selector SFS')
plt.xticks(rotation=90)
plt.show()
```

Feature Selector SFS

```
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)
```

```
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs

fig = plot_sfs(sfs.get_metric_dict(), kind='std_err', figsize=(15, 5))
plt.title("Feature Selector - SFS")
plt.xticks(rotation=90)
plt.show()
```




```
feature_index = list(sfs.k_feature_idx_)
print(feature_index)
```

```
[1, 2, 4, 5, 7, 9, 10, 12, 13, 15, 18, 20, 21, 22, 24, 25, 26, 27, 30, 32, 37, 39, 40, 41, 43, 50, 51, 58, 59, 60, 62, 64, 65, 67, 69]
```

```
X_train.columns[feature_index]
```

```
Index(['duration', 'years_running', 'contentWarn', 'watched', 'wantWatch',
       'votes', 'tag_Based_on_a_Manga', 'tag_Action', 'tag_Fantasy',
       'tag_Shounen', 'tag_Drama', 'tag_Family_Friendly', 'tag_Short_Episodes',
       'tag_School_Life', 'tag_Shorts', 'tag_Slice_of_Life', 'tag_Seinen',
       'tag_Supernatural', 'tag_Ecchi', 'tag_Based_on_a_Light_Novel',
       'mediaType_Movie', 'mediaType_OVA', 'mediaType_Other', 'mediaType_TV',
       'mediaType_Web', 'studio_primary_Bones', 'studio_primary_DLE',
       'studio_primary_Others', 'studio_primary_Production I.G',
       'studio_primary_Shaft', 'studio_primary_Studio Deen',
       'studio_primary_Sunrise', 'studio_primary_TMS Entertainment',
       'studio_primary_Toei Animation', 'studio_primary_is_missing'],
      dtype='object')
```

```
X_train_final = X_train.iloc[:, feature_index]
```

```
X_train_final = X_train.iloc[:, feature_index]


X_test_final = X_test.iloc[:, feature_index]
lin_model_v2 = LinearRegression()
lin_model_v2.fit(X_train_final, Y_train)
```

▾ LinearRegression
LinearRegression()

```
print("Training Data Performance")

lin_model_train = Model_performance(lin_model, X_train, Y_train)

print(lin_model_train)
```

```
     Training Data Performance
            RMSE  R2 Score
     0  0.578282  0.518574
```

```
print("Training Data Performance")

lin_model_train = Model_performance(lin_model, X_test, Y_test)

print(lin_model_train)
```

```
     Training Data Performance
            RMSE  R2 Score
     0  0.564057  0.519399
```

```
X_train.columns[feature_index]
```

```
     Index(['duration', 'years_running', 'contentWarn', 'watched', 'wantWatch',
            'votes', 'tag_Based_on_a_Manga', 'tag_Action', 'tag_Fantasy',
            'tag_Shounen', 'tag_Drama', 'tag_Family_Friendly', 'tag_Short_Episodes',
            'tag_School_Life', 'tag_Shorts', 'tag_Slice_of_Life', 'tag_Seinen',
            'tag_Supernatural', 'tag_Ecchi', 'tag_Based_on_a_Light_Novel',
            'mediaType_Movie', 'mediaType_OVA', 'mediaType_Other', 'mediaType_TV',
            'mediaType_Web', 'studio_primary_Bones', 'studio_primary_DLE',
            'studio_primary_Others', 'studio_primary_Production I.G',
            'studio_primary_Shaft', 'studio_primary_Studio Deen',
            'studio_primary_Sunrise', 'studio_primary_TMS Entertainment',
```