```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import metrics
from xgboost import XGBRegressor

import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('/content/boxoffice.csv',
                 encoding='latin-1')
df.head()
```

	title	domestic_revenue	world_revenue	distributor	opening_revenue	opening_theaters	
0	Star Wars: Episode VIII - The Last Jedi	$620,181,382	$1,332,539,889	Walt Disney Studios Motion Pictures	$220,009,584	4,232	$
1	The Fate of the Furious	$226,008,385	$1,236,005,118	Universal Pictures	$98,786,705	4,310	$
2	Wonder Woman	$412,563,408	$821,847,012	Warner Bros.	$103,251,471	4,165	$
3	Guardians of the Galaxy Vol. 2	$389,813,101	$863,756,051	Walt Disney Studios Motion Pictures	$146,510,104	4,347	$
4	Beauty and the Beast	$504,014,165	$1,263,521,126	Walt Disney Studios Motion Pictures	$174,750,616	4,210	$

```
df.shape
```

```
(2694, 10)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2694 entries, 0 to 2693
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   title             2694 non-null   object
 1   domestic_revenue  2694 non-null   object
 2   world_revenue     2694 non-null   object
 3   distributor       2694 non-null   object
 4   opening_revenue   2390 non-null   object
 5   opening_theaters  2383 non-null   object
 6   budget            397 non-null    object
 7   MPAA              1225 non-null   object
 8   genres            2655 non-null   object
 9   release_days      2694 non-null   object
dtypes: object(10)
memory usage: 210.6+ KB
```

```
df.describe().T
```

	count	unique	top	freq
title	2694	2468	A Beautiful Planet	3
domestic_revenue	2694	2495	$11,272,008	3
world_revenue	2694	2501	$25,681,505	3
distributor	2694	248	Fathom Events	292
opening_revenue	2390	2176	$4,696	3
opening_theaters	2383	732	1	503
budget	397	124	$40,000,000	14
MPAA	1225	8	R	568
genres	2655	567	Documentary	351

```
# We will be predicting only
# domestic_revenue in this article.

to_remove = ['world_revenue', 'opening_revenue']
df.drop(to_remove, axis=1, inplace=True)
```

```
df.isnull().sum() * 100 / df.shape[0]
```

```
title                0.000000
domestic_revenue     0.000000
distributor          0.000000
opening_theaters    11.544172
budget              85.263549
MPAA                54.528582
genres               1.447661
release_days         0.000000
dtype: float64
```

```
# Handling the null value columns
df.drop('budget', axis=1, inplace=True)

for col in ['MPAA', 'genres']:
    df[col] = df[col].fillna(df[col].mode()[0])

df.dropna(inplace=True)

df.isnull().sum().sum()
```

```
0
```

```
df['domestic_revenue'] = df['domestic_revenue'].str[1:]

for col in ['domestic_revenue', 'opening_theaters', 'release_days']:
    df[col] = df[col].str.replace(',', '')

    # Selecting rows with no null values
    # in the columns on which we are iterating.
    temp = (~df[col].isnull())
    df[temp][col] = df[temp][col].convert_dtypes(float)

    df[col] = pd.to_numeric(df[col], errors='coerce')
```

```
df['MPAA'].unique()
```

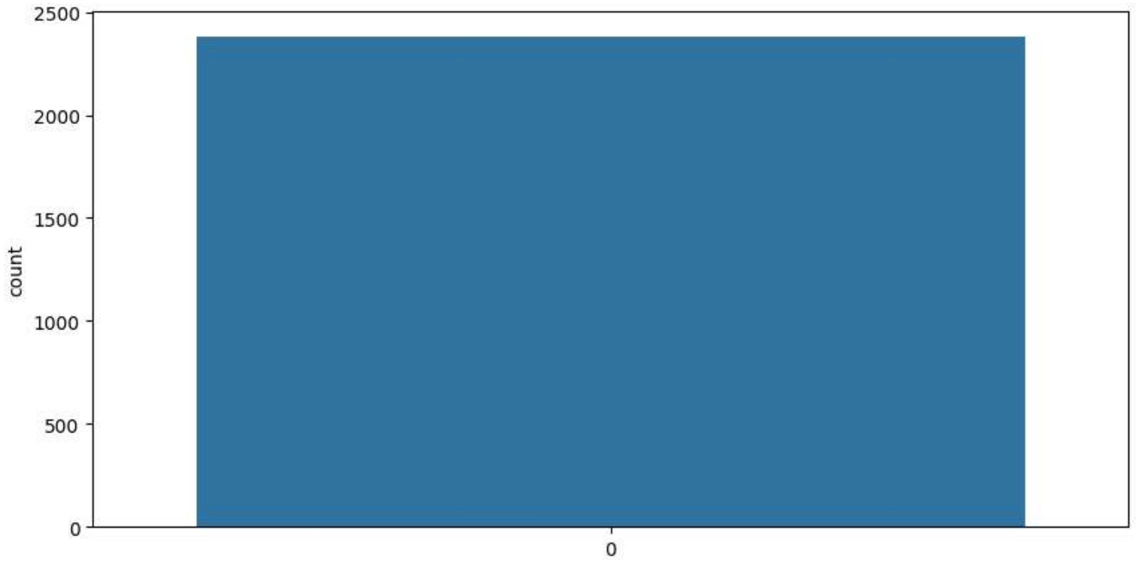
```
array(['PG-13', 'PG', 'R', 'Not Rated', 'G', 'NC-17', 'M/PG'],
      dtype=object)
```

```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows
# how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
```

```
# Encode labels in column 'species'.
df['MPAA']= label_encoder.fit_transform(df['MPAA'])

df['MPAA'].unique()
```

```
    array([5, 4, 6, 3, 0, 2, 1])
```

```
# Create count plot of MPAA ratings
plt.figure(figsize=(10, 5))
sb.countplot(df['MPAA'])
plt.show()
```



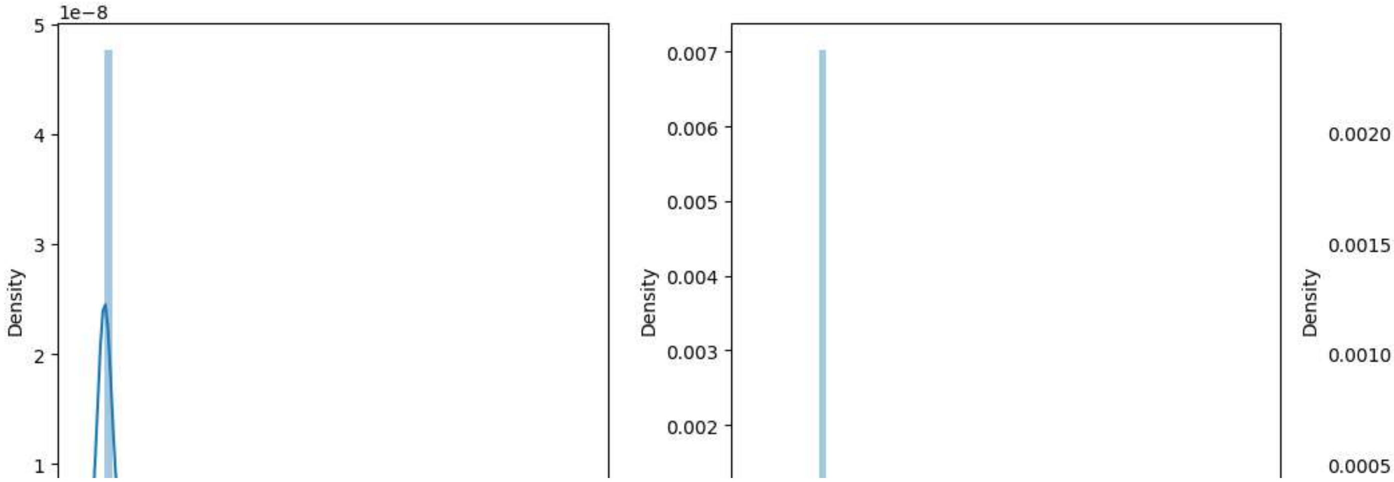
```
df.head()
```

	title	domestic_revenue	distributor	opening_theaters	MPAA	gen
0	Star Wars: Episode VIII - The Last Jedi	620181382	Walt Disney Studios Motion Pictures	4232	5	Action,Adventure,Fantasy,Sc
1	The Fate of the Furious	226008385	Universal Pictures	4310	5	Action,Adventure,Thr
2	Wonder Woman	412563408	Warner Bros.	4165	5	Action,Adventure,Fantasy,Sci-Fi,\
3	Guardians of the Galaxy Vol. 2	389813101	Walt Disney Studios Motion Pictures	4347	5	Action,Adventure,Comedy,Sc
4	Beauty and the Beast	504014165	Walt Disney Studios Motion Pictures	4210	4	Family,Fantasy,Musical,Roma

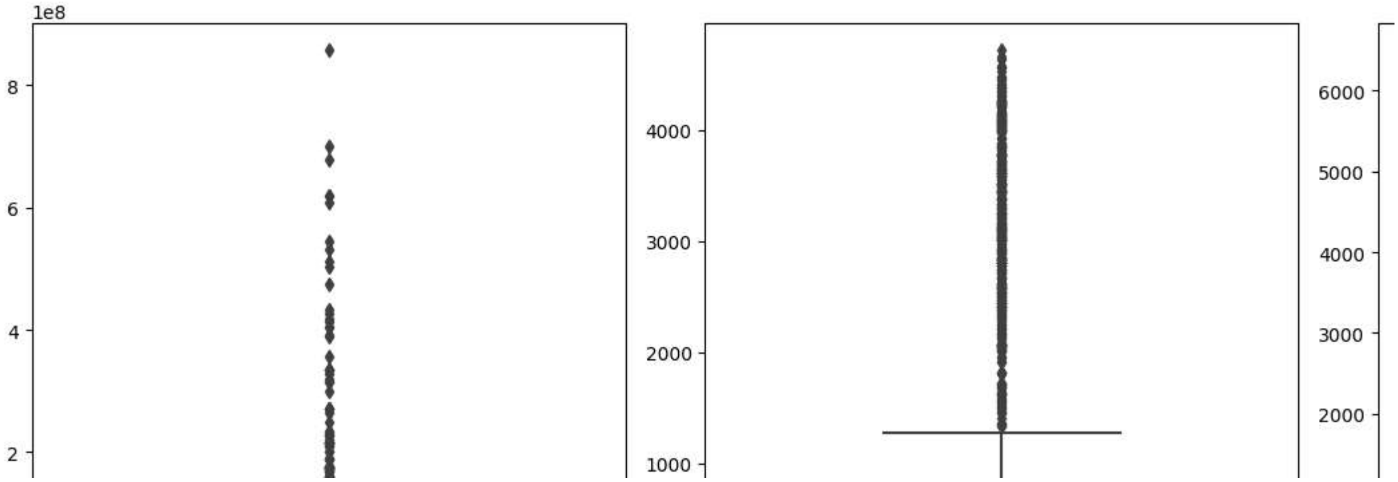
```
df.groupby('MPAA').mean()['domestic_revenue']
```

```
    MPAA
    0    3.539276e+07
    1    5.113500e+05
    2    1.368800e+04
    3    4.897703e+05
    4    5.379622e+07
    5    5.891966e+07
    6    6.591336e+06
    Name: domestic_revenue, dtype: float64
```

```
plt.subplots(figsize=(15, 5))

features = ['domestic_revenue', 'opening_theaters', 'release_days']
for i, col in enumerate(features):
    plt.subplot(1, 3, i+1)
    sb.distplot(df[col])
plt.tight_layout()
plt.show()
```



```
plt.subplots(figsize=(15, 5))
for i, col in enumerate(features):
    plt.subplot(1, 3, i+1)
    sb.boxplot(df[col])
plt.tight_layout()
plt.show()
```

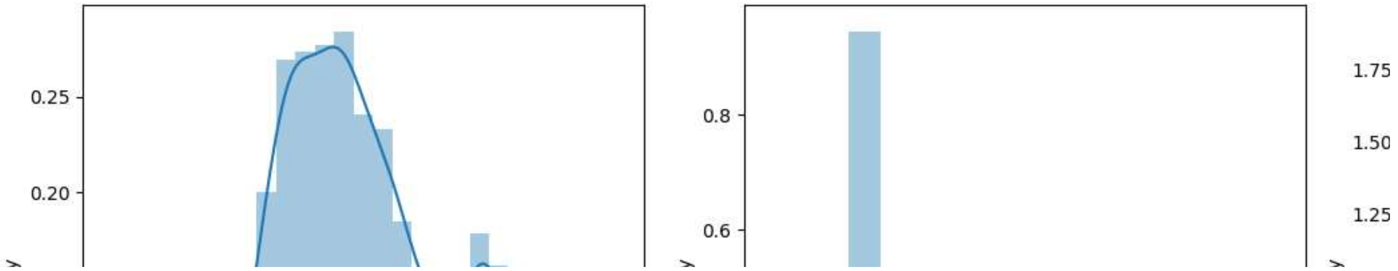


```
for col in features:
  df[col] = df[col].apply(lambda x: np.log10(x))
```

```
df['MPAA'].unique()
```

```
     array([5, 4, 6, 3, 0, 2, 1])
```

```
plt.subplots(figsize=(15, 5))
for i, col in enumerate(features):
    plt.subplot(1, 3, i+1)
    sb.distplot(df[col])
plt.tight_layout()
plt.show()
```



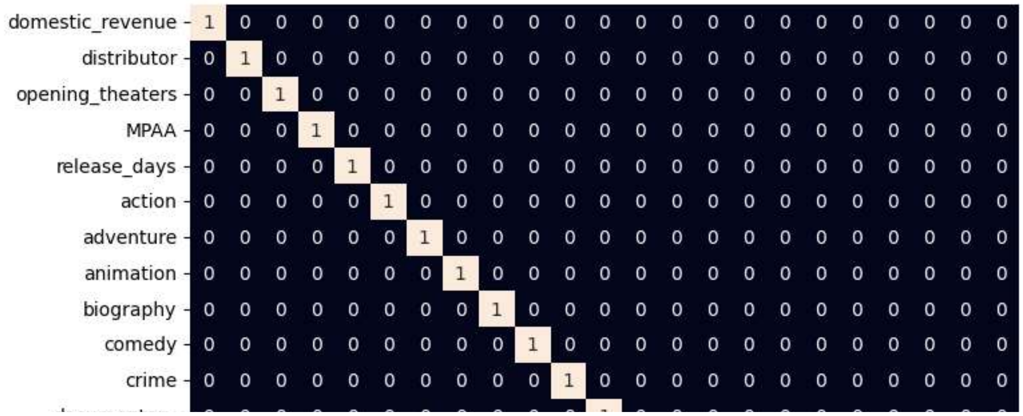
```
vectorizer = CountVectorizer()
vectorizer.fit(df['genres'])
features = vectorizer.transform(df['genres']).toarray()

genres = vectorizer.get_feature_names_out()
for i, name in enumerate(genres):
    df[name] = features[:, i]

df.drop('genres', axis=1, inplace=True)
```

```
removed = 0
for col in df.loc[:, 'action':'western'].columns:

    # Removing columns having more
    # than 95% of the values as zero.
    if (df[col] == 0).mean() > 0.95:
        removed += 1
        df.drop(col, axis=1, inplace=True)

print(removed)
print(df.shape)
```

```
11
(2383, 24)
```

```
for col in ['distributor', 'MPAA']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
```

```
plt.figure(figsize=(8, 8))
sb.heatmap(df.corr() > 0.8,
        annot=True,
        cbar=False)
plt.show()
```



```
df.head()
```

	title	domestic_revenue	distributor	opening_theaters	MPAA	release_days	action	adventure	animation	biography
0	Star Wars: Episode VIII - The Last Jedi	8.792519	217	3.626546	5	2.582063	1	1	0	0
1	The Fate of the Furious	8.354125	208	3.634477	5	2.418301	1	1	0	0
2	Wonder Woman	8.615491	218	3.619615	5	2.336460	1	1	0	0
3	Guardians of the Galaxy Vol. 2	8.590856	217	3.638190	5	2.382017	1	1	0	0
4	Beauty and the Beast	8.702443	217	3.624282	4	2.462398	0	0	0	0

5 rows × 24 columns

```
features = df.drop(['title', 'domestic_revenue','fi'], axis=1)
target = df['domestic_revenue'].values

X_train, X_val,\
    Y_train, Y_val = train_test_split(features, target,
                                      test_size=0.1,
                                      random_state=22)
X_train.shape, X_val.shape
```

```
((2144, 21), (239, 21))
```

```
# Normalizing the features for stable and fast training.
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
```

```
from sklearn.metrics import mean_absolute_error as mae
model = XGBRegressor()
model.fit(X_train, Y_train)
```

```
▾                                XGBRegressor

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
```

```
# Create and train an XGBoost model
model = XGBRegressor()
model.fit(X_train, Y_train)

# Make predictions on the validation set
val_preds = model.predict(X_val)

# Calculate the Mean Absolute Error on the validation set
validation_error = mean_absolute_error(Y_val, val_preds)
print(f'Validation Error: {validation_error}')
```

```
Validation Error: 0.4340367343796249
```