RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

BOX OFFICE REVENUE PREDECTION USING

LINEAR REGRESSION IN MACHINE LEARNING

A MINI PROJECT REPORT

SUBMITTED BY

THARUN S	**231801506**
VARUN V	**231801185**

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM CHENNAI-602105

2024-2025

BONAFIDE CERTIFICATE

CERTIFIED THAT THIS PROJECT REPORT "**BOX OFFICE REVENUE PREDICTION USING LINEAR REGRESSION IN MACHINE LEARNING**" IS THE BONAFIDE WORK OF **"THARUN S[231801506] VARUN V[231801185]"** WHO CARRIED OUT THE PROJECT WORK UNDER MY SUPERVISION.

Submitted for the Practical Examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Abstract:

This mini project focuses on predicting box office revenue using linear regression, a key machine learning technique. The goal is to create a model that can forecast a movie's financial success based on various factors, including production budget, genre, star power, director, and marketing efforts. By analyzing historical data of past movie releases, this project aims to establish a predictive relationship between these features and the resulting box office revenue.

The dataset used in this study includes essential attributes such as budget, genre, cast, director, and movie ratings. The data is preprocessed to handle missing values and ensure consistency. A linear regression model is then trained on this data to understand the underlying patterns and relationships between the predictors and the target variable, which is the box office revenue. The model's performance is evaluated using metrics like mean squared error (MSE) and R-squared value.

The results show how well the linear regression model can generalize and predict box office revenue for new movies. The insights gained from this project can help stakeholders, including film producers, marketers, and investors, make data-driven decisions in budgeting, casting, and marketing strategies to improve a movie's chances of success at the box office.

TABLE OF CONTENTS

1. INTRODUCTION

When a movie is produced then the director would certainly like to maximize his/her movie's revenue. But can we predict what will be the revenue of a movie by using its genre or budget information. We will learn how to implement a machine learning algorithm that can predict box office revenue by using the genre of the movie and other related features.

1.1 Purpose

1. Business Decision Making: Movie studios and distributors can use revenue predictions to make informed decisions about release strategies, marketing budgets, and distribution plans.

2. Risk Assessment: Linear regression can help assess the potential financial risks associated with a movie's production and release, aiding investors and studios in managing their resources.

3. Marketing Optimization: It can assist in optimizing marketing campaigns by identifying the key factors (e.g., genre, star cast, release date) that influence a movie's box office performance.

4. Revenue Maximization: Studios can aim to maximize their profits by fine-tuning various factors, such as pricing, release timing, and audience targeting, based on the predictions.

5. Performance Evaluation: After a movie's release, these predictions can be compared to actual revenue to evaluate the model's accuracy and improve future predictions.

1.2 Scope

Predicting box office revenue using linear regression in machine learning helps

Movie makers, investors, and studios make smarter decisions about how to create, market, and release films. It's like using math to guess how much money a movie will make, so people can plan better and reduce financial risks. This method is also useful for understanding what factors, like the type of movie or famous actors, make a movie successful at the box office.

2. SOFTWARE AND HARDWARE SPECIFICATIONS

2.1 Software requirements

Software	:	i) Anaconda – Jupyter Notebook (Latest Version) ii) Python IDLE 3.11+
User interface	:	PYTHON, JUPYTER
Dataset	:	Ms Excel (CSV)
Documentation Tool	:	Ms Word

2.2 Hardware requirements

Processor	:	i3 or better processor
Operating System	:	Windows 8.1, 10, 11
RAM	:	6 GB RAM or more
Hard Disk	:	100 GB or more
Monitor	:	LED color monitor
Keyboard	:	Standard keyboard
Mouse	:	Standard mouse
Internet	:	High Speed Network

3. PROBLEM DEFINITION

3.1 Importing Libraries

3.2

Python libraries make it easy for us to handle the data and perform typical and complex tasks with a single line of code.

- **Pandas –** This library helps to load the data frame in a 2D array format and has multiple functions to perform analysis tasks in one go.
- **Numpy –** Numpy arrays are very fast and can perform large computations in a very short time.
- **Matplotlib/Seaborn –** This library is used to draw visualizations.
- **Sklearn –** This module contains multiple libraries are having pre-implemented functions to perform tasks from data preprocessing to model development and evaluation.
- **XGBoost –** This contains the eXtreme Gradient Boosting machine learning algorithm which is one of the algorithms which helps us to achieve high accuracy on predictions.

3.3 Importing Dataset

The Dataset is in the format of CSV named as "boxoffice.csv"

The Dataset contains,

1) **title –** The name of Movie. (Eg: Avengers Endgame, Cars 3, Leo)
2) **domestic_revenue –** The revenue collection of particular regions. (Eg: Salem - Tamil Nadu)
3) **world_revenue –** The revenue collection of worldwide areas. (Eg: India, USA)
4) **distributor –** A film distributor is responsible for the marketing of a film. The distribution company may be the same with, or different from, the production company. Distribution deals are an important part of financing a film.
   (Eg: Sony Pictures, Walt Disney, Sun Pictures)
5) **opening revenue –** The first day release collection.
6) **budget –** The amount is invested by production company.
7) **opening_theaters –** The allocated theatres.
8) **MPAA –** This is a Film rating system given by filming authorities. (Eg: U, U/A, A)
9) **genres –** A film genre is a stylistic or thematic category for motion pictures based on similarities either in the narrative elements, aesthetic approach, or the emotional response to the film.

- Action.
- Comedy.
- Drama.
- Fantasy.
- Horror.
- Mystery.
- Romance.
- Thriller.

10) **release_days –** After release, In how many days the revenue collection is calculated.

3.4 Data Cleaning

There are times when we need to clean the data because the raw data contains lots of noise and irregularities and we cannot train an ML model on such data. Hence, data cleaning is an important part of any machine-learning pipeline.

3.5 Exploratory Data Analysis

EDA is an approach to analyzing the data using visual techniques. It is used to discover trends, and patterns, or to check assumptions with the help of statistical summaries and graphical representations.

3.6 Model Development

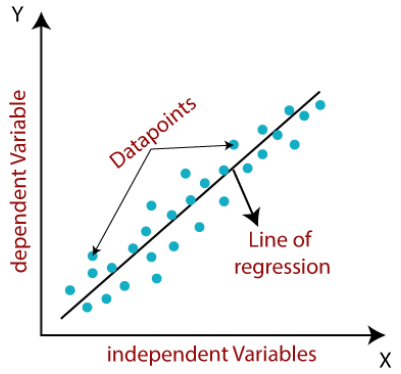
Now we will separate the features and target variables and split them into training and the testing data by using which we will select the model which is performing best on the validation data.

XGBoost library models help to achieve state-of-the-art results most of the time so, we will also train this model to get better results.

3.7 Algorithm –

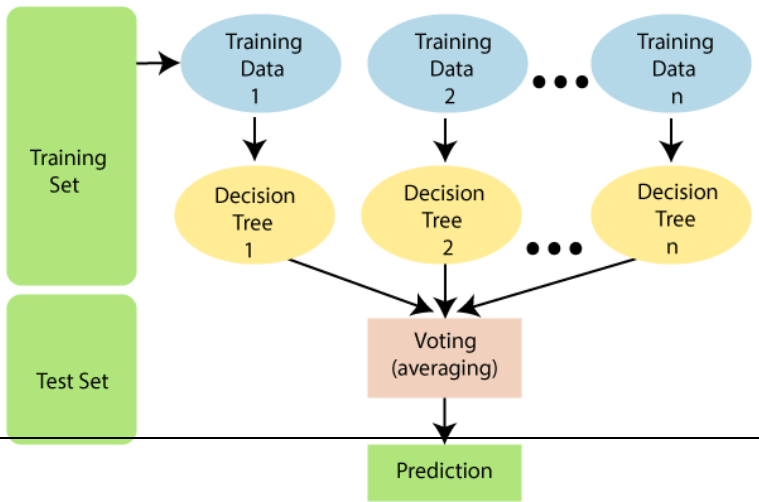
i) Linear Regression

Linear Regression is the supervised Machine Learning model in which the model finds the best fit linear line between the independent and dependent variable i.e it finds the linear relationship between the dependent and independent variable.



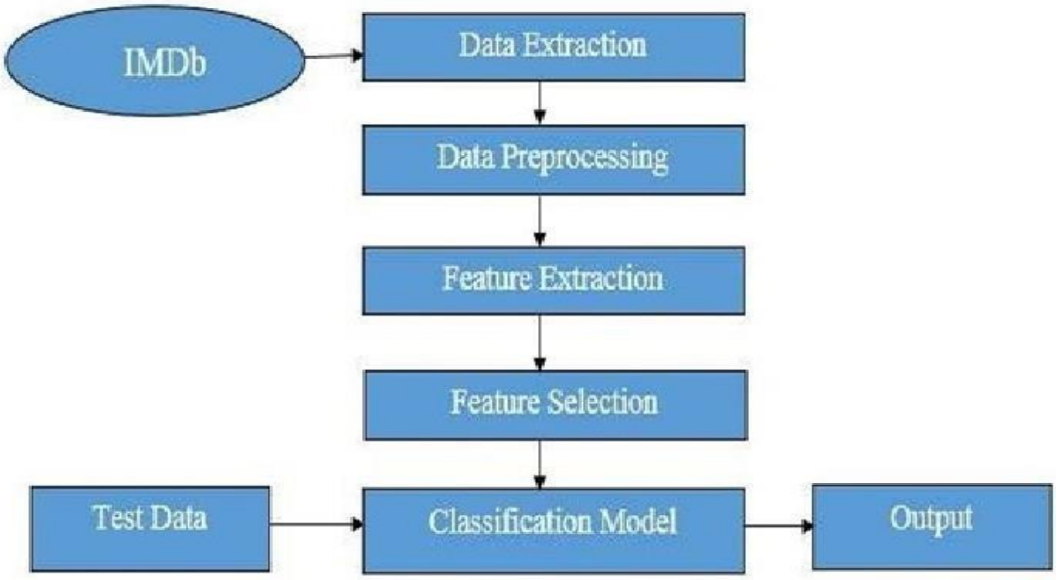
ii) Random Forest Algorithm

Random Forest is a classifier that contains a number of decision trees on various subsets of



the given dataset and takes the average to improve the predictive accuracy of that dataset.

3.8 Proposed System



SCREEN SHOTS (Jupyter Notebook)

jupyter box Last Checkpoint: 14 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Importing Libraries and Dataset

In [1]:
```
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.ensemble import RandomForestRegressor
from sklearn import metrics
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression ,Lasso ,Ridge

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

Loading Dataset

In [2]:
```
df = pd.read_csv('boxoffice.csv',
                 encoding='latin-1')
df.head()
```

Out[2]:

	title	domestic_revenue	world_revenue	distributor	opening_revenue	opening_theaters	budget	MPAA	genres
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	$317,000,000	PG-13	Action,Adventure,Fantasy,Sci-Fi
1	The Fate of the Furious	$226,008,385	$1,236,005,118	Universal Pictures	$98,786,705	4,310	$250,000,000	PG-13	Action,Adventure,Thriller
2	Wonder Woman	$412,563,408	$821,847,012	Warner Bros.	$103,251,471	4,165	$149,000,000	PG-13	Action,Adventure,Fantasy,Sci-Fi,War
3	Guardians of the Galaxy Vol. 2	$389,813,101	$863,756,051	Walt Disney Studios Motion Pictures	$146,510,104	4,347	$200,000,000	PG-13	Action,Adventure,Comedy,Sci-Fi
4	Beauty and the Beast	$504,014,165	$1,263,521,126	Walt Disney Studios Motion Pictures	$174,750,616	4,210	$160,000,000	PG	Family,Fantasy,Musical,Romance

In [3]:
```
df.shape
```

Out[3]: (2694, 10)

jupyter box Last Checkpoint: 14 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [4]:
```
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
```

In [5]:
```
df.describe().T
```

Out[5]:

	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
release_days	2694	457	347	35

Data Cleaning

In [6]:
```
# We will be predicting only
# domestic_revenue in this article.

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

In [7]:
```
df.isnull().sum() * 100 / df.shape[0]
```

Out[7]:
```
title               0.000000
domestic_revenue    0.000000
distributor         0.000000
opening_theaters   11.544172
budget             85.263549
```

jupyter box Last Checkpoint: 14 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Data Cleaning

In [6]:
```
# We will be predicting only
# domestic_revenue in this article.

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

In [7]:
```
df.isnull().sum() * 100 / df.shape[0]
```

Out[7]:
```
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
```

In [8]:
```
# 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()
```

Out[8]:
```
0
```

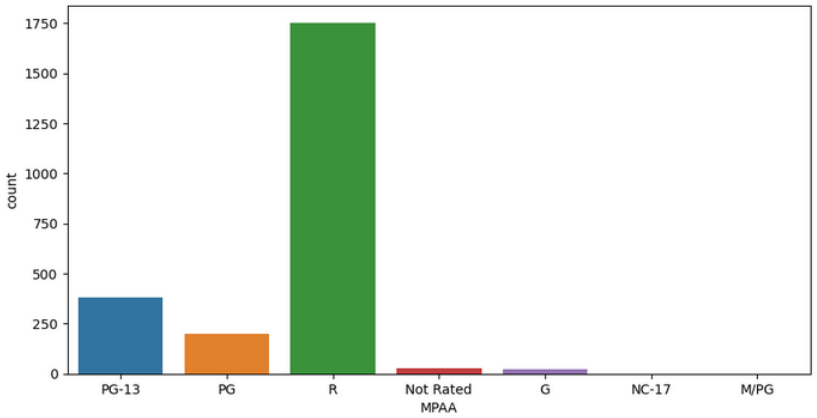
In [9]:
```
df['domestic_revenue'] = df['domestic_revenue'].str[1:]

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')
```

jupyter box Last Checkpoint: 15 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Exploratory Data Analysis

In [10]:
```
plt.figure(figsize=(10, 5))
sb.countplot(df['MPAA'])
plt.show()
```



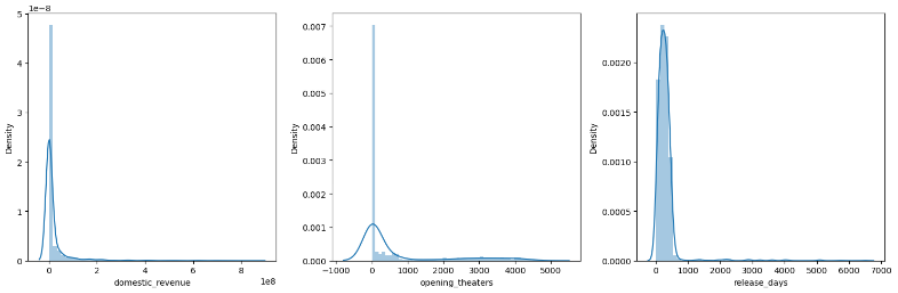
In [11]:
```
df.groupby('MPAA').mean()['domestic_revenue']
```

Out[11]:
```
MPAA
G            3.539276e+07
M/PG         5.113500e+05
NC-17        1.368800e+04
Not Rated    4.897703e+05
PG           5.379622e+07
PG-13        5.891966e+07
R            6.591336e+06
Name: domestic_revenue, dtype: float64
```

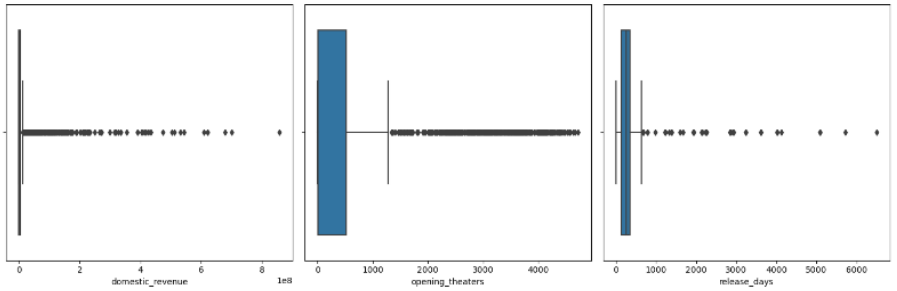
jupyter box Last Checkpoint: 15 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [12]:
```
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()
```



In [13]:
```
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()
```

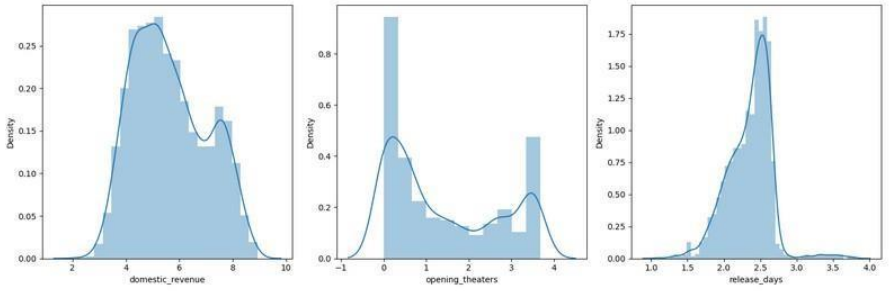


jupyter box Last Checkpoint: 15 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [14]:
```
for col in features:
    df[col] = df[col].apply(lambda x: np.log10(x))
```

In [15]:
```
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()
```



In [16]:
```
vectorizer = CountVectorizer()
vectorizer.fit(df['genres'])
features = vectorizer.transform(df['genres']).toarray()

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

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

In [17]:
```
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)
```

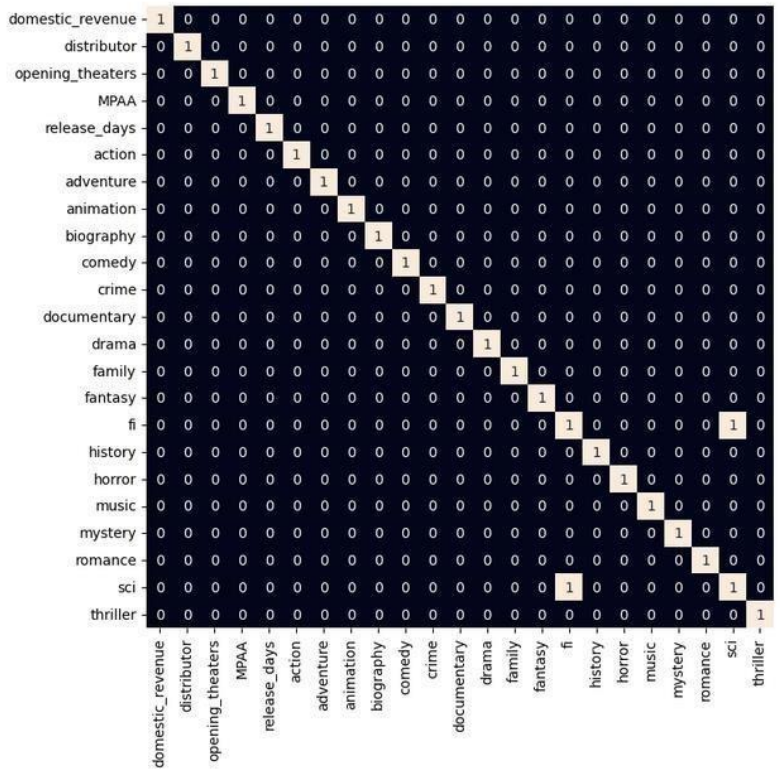
```
11
(2383, 24)
```

jupyter box Last Checkpoint: 15 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [18]:
```
for col in ['distributor', 'MPAA']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
```

In [19]:
```
plt.figure(figsize=(8, 8))
sb.heatmap(df.corr() > 0.8,
           annot=True,
           cbar=False)
plt.show()
```



jupyter box Last Checkpoint: 15 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Model Devlopement

In [20]:
```
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
```

Out[20]: ((2144, 21), (239, 21))

In [21]:
```
# Normalizing the features for stable and fast training.
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
```

In [22]:
```
from sklearn.metrics import mean_absolute_error as mae
model = XGBRegressor()
model.fit(X_train, Y_train)
```

Out[22]:
```
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,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

jupyter box Last Checkpoint: 15 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Final Output

In [23]:
```
train_preds = model.predict(X_train)
print('Training Error : ', mae(Y_train, train_preds))

val_preds = model.predict(X_val)
print('Validation Error : ', mae(Y_val, val_preds))
print()
```

```
Training Error :  0.1311083568380254
Validation Error :  0.4340367343796249
```

In [24]:
```
from sklearn.metrics import mean_absolute_error as mae
models = [LinearRegression(),
    Lasso(), RandomForestRegressor(), Ridge(), XGBRegressor(),]

for i in range(5):
    models[i].fit(X_train, Y_train)

    print(f'{models[i]} : ')

    train_preds = models[i].predict(X_train)
    print('Training Error : ', mae(Y_train, train_preds))

    val_preds = models[i].predict(X_val)
    print('Validation Error : ', mae(Y_val, val_preds))
    print()
```

```
LinearRegression() :
Training Error :  0.6346133108283997
Validation Error :  0.656118317140736

Lasso() :
Training Error :  1.1323906831239086
Validation Error :  1.1868408850364023

RandomForestRegressor() :
Training Error :  0.16184048873946683
Validation Error :  0.45321036673871545

Ridge() :
Training Error :  0.6346326594921088
Validation Error :  0.6561942489041211

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,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...) :
Training Error :  0.1311083568380254
Validation Error :  0.4340367343796249
```

4. CONCLUSION

To sum it up, using linear regression in machine learning for predicting box office revenue helps movie folks make better choices about how to create and sell their movies. It's like a financial crystal ball, giving insights into what makes a movie successful and how to make more money in the film business.

5. REFERENCES

- GeeksforGeeks (https://www.geeksforgeeks.org/box-office-revenue-prediction-using-linear-regression-in-ml/)
- Machine Learning, Anuradha Srinivasa Raghavan and Vincy Joseph
- Guru99 (https://www.guru99.com/)
- Towards Data Science (https://towardsdatascience.com/)