Movie Box Office Gross Prediction Using Machine Learning

1.introduction

The movie industry is a massive sector for investment but larger business sectors have more complexity, and it is hard to choose how to invest. Furthermore, significant investments come with more significant risks. The CEO of Motion Picture Association of America (MPAA), J. Valenti mentioned that “No one can tell you how a movie is going to do in the marketplace. Not until the film opens in darkened theatre and sparks fly up between the screen and the audience” . As movie industry is growing too fast day by day, there are now a considerable amount of data available on the internet, which makes it an exciting field for data analysis. Predicting a movie's box office success is a very complicated task to do. The definition of success of a film is relative, some videos are called successful based on its worldwide gross income, and some movies may not shine in business part but can be called successful for good critics' review and popularity. In this paper, we consider a movie's box office success based on its profit only. Researchers show that almost 25% of movie revenue comes within the first or second week of its release. So it is hard to predict a movie's box-office success before its release date. In our proposed model, we use two types of features called pre-released features and post-released features. Only pre-released features are considered to predict the success of an upcoming movie. After releasing a film, both pre-released and post-released features will be available for further speculation. There are six pre-released features and nine post-released features. Instead of predicting only flop or blockbuster movies , we instead choose to classify a film based on its box office profit into one of five categories ranging from flop to blockbuster.

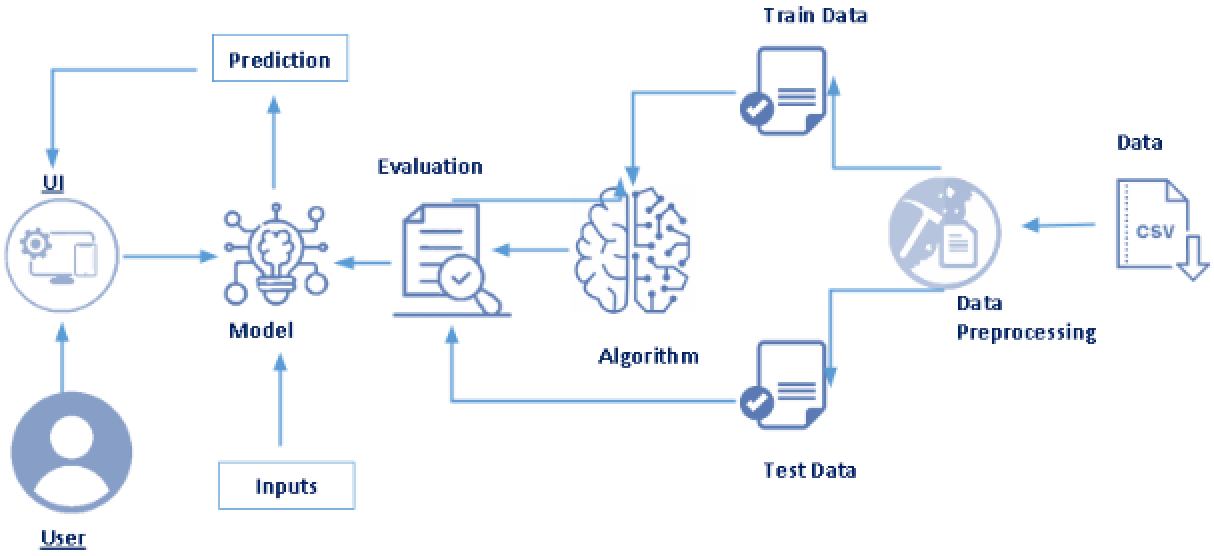
predicting society's reaction to a new product in the sense of popularity and adaption rate has become an emerging field of data analysis. The motion picture industry is a multi-billion-dollar business, and there is a massive amount of data related to movies is available over the internet. This study proposes a decision support system for movie investment sector using machine learning techniques. This research helps investors associated with this business for avoiding investment risks. The system predicts an approximate success rate of a movie based on its profitability by analyzing historical data from different sources like IMDb, Rotten Tomatoes, Box Office Mojo and Metacritic. Using Support Vector Machine (SVM), Neural Network and Natural Language Processing the system predicts a movie box office profit based on some pre-released features and post-released features. This paper shows Neural Network gives an accuracy of 84.1% for pre-released features and 89.27% for all features while SVM has 83.44% and 88.87% accuracy for pre-released features and all features respectively when one away prediction is considered. Moreover, we figure out that budget, IMDb votes and no. of screens are the most important features which play a vital role while predicting a movie's box-office success.

2.LITERATURE SURVEY

Matt Vitelli [1] has designed a model for the movie revenue prediction by observing the relationships by the combination between the features using graphs such as actor-actor relationship graphs, actor-movie relationship graphs, and movie-movie relationship graphs and concluded that by combining these features and observing the relationships using the actor-actor relationship graphs and actor movie relationship graphs, they were able to design more accurate model than using the features in the Dataset alone.

Prediction of movie box office success based on the Wikipedia Activity Big Data used a simple model based only on few variables, but the efficiency could be enhanced using statistical computations and also more on the content related parameters[4][5], for example, the controversial measure in the article. There can be a diverse number of changes that can be made to the defined model in this paper.

Technical Architecture:



EXPERIMENTAL INVESTIGATION :

Market information, epidemic information, national micro-data and other information are introduced to study the influencing factors on the box office. In terms of market information, data such as the number and average box office of films participating in the past of film production companies, manufacturing companies, distribution companies, integrated marketing companies, and new media marketing companies are collated and calculated, and the Baidu search index data of film titles are introduced. In terms of epidemic information, a variety of data including newly confirmed cases on the day of screening, existing confirmed cases, newly asymptomatic infected persons, cumulative cured and discharged cases, cumulative deaths, cumulative reported confirmed cases, and existing suspected cases were introduced, as well as their derivative data. In terms of national micro-data, the polynomial regression equation is established by taking the number of screens, the number of cinemas, GDP, Internet penetration rate, and other microdata over the years as dependent variables and time as the independent variable. According to the release date of the movie, we predict the number of screens and theaters when the movie is released, as a prediction of the data characteristics of the research box office. In this paper, different from the existing single-model evaluation, the enhanced stacking algorithm is used to integrate multiple models to build a movie box office prediction model. The movie box office prediction model is trained and tested on the dataset of the top 2000 films used in this paper, and the final MAPE is 14.49%. It is superior to XGBoost, LightGBM, and other models. The experimental results show that the enhanced stacking model used in this paper has a certain application value, but there is still room for further research and improvement in algorithm improvement. The data volume of box office prediction used in this paper is not large enough, and it can be further expanded in the future to establish a more effective box office prediction model.

Prerequisites

To complete this project you should have the following softwares and packages

Anaconda Navigator :

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like

JupyterLab, Jupyter Notebook,QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupiter notebook and spyder

Anaconda navigator

- Refer the link below to download anaconda navigator
- Link :- https://youtu.be/5mDYijMfSzs

○ **Python packages:**

- Open anaconda prompt as administrator
- Type“pipinstallnumpy”andclick enter.
- Type“pipinstallpandas”andclickenter.
- Type“pipinstallscikit-learn”andclickenter.
- Type”pipinstall matplotlib”andclickenter.
- Type”pipinstallscipy”andclickenter.
- Type”pipinstallpickle-mixin”andclickenter.
- Type”pipinstallseaborn”andclickenter.
- Type“pipinstallFlask”andclickenter.

To build Machine learning models you must require the following packages

Sklearn: Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms.

NumPy: NumPy is a Python package that stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object

Pandas: pandas is a fast, powerful, flexible, and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language.

Matplotlib: It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits

Wordcloud:

A word cloud is an image made of words that together resemble a cloudy shape. The size of a word shows how important it is e.g. how often it appears in a text.

Watch the below video to download the necessary Packages

Prior Knowledge

To complete this project you must have prior knowledge of the Following Topics

Supervised and unsupervised learning:

Link:- https://youtu.be/kE5QZ8G_78c

Regression Classification and Clustering :

Link:- https://youtu.be/6za9_mh3uTE

LinearRegression :

Link:- https://youtu.be/1-OGRohmH2s

Flask :

Link:- https://youtu.be/lj4I_CvBnt0

Project Objectives

By the end of this project:

- You’ll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
- You will able to analyze or get insights of data through visualization.
- Applying different algorithms according to the dataset and based on visualization.
- You will able to know how to find the accuracy of the model.
- You will be able to know how to build a web application using the Flask framework.

Project Work Flow

Project Work Flow:

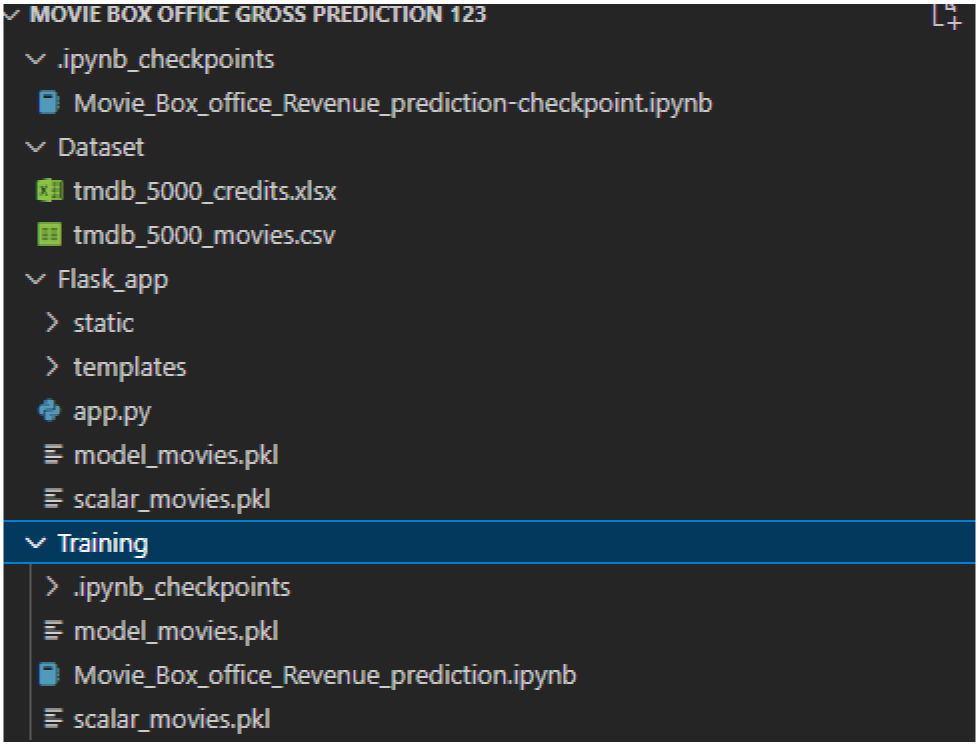
- The user interacts with the UI (User Interface) to upload the input features.
- Uploaded features/input is analyzed by the model which is integrated.
- Once the model analyses the uploaded inputs, the prediction is showcased on the UI.

To accomplish this, we have to complete all the activities and tasks listed below Tasks:

- Data Collection.
    - Collect the dataset or Create the dataset
- Data Preprocessing.
    - Import the Libraries.
    - Reading the dataset.
    - Exploratory Data Analysis
    - Converting json objects to strings
    - Checking for Null Values.
    - Data Visualization.
    - Dropping the columns
    - Label Encoding
    - Splitting the Dataset into Dependent and Independent variable.
    - Feature scaling
    - Splitting Data into Train and Test.
- Model Building
    - Training and testing the model
    - Evaluation of Model
    - Save the model
    - Predicting the output using the model
- Application Building
    - Create an HTML file
    - Build a Python Code
    - Run the app

Project Structure:

Create the Project folder which contains files as shown below



```
MOVIE BOX OFFICE GROSS PREDICTION 123
  .ipynb_checkpoints
    Movie_Box_office_Revenue_prediction-checkpoint.ipynb
  Dataset
    tmdb_5000_credits.xlsx
    tmdb_5000_movies.csv
  Flask_app
    static
    templates
    app.py
    model_movies.pkl
    scalar_movies.pkl
  Training
    .ipynb_checkpoints
    model_movies.pkl
    Movie_Box_office_Revenue_prediction.ipynb
    scalar_movies.pkl
```

- We have three folders dataset, Flask_app, and model_building
- **A python file called app.py for server-side scripting.**
- **We need the model which is saved and the saved model in this content is (model_movies.pkl and scalar_movies.pkl).**
- **Templates folder which contains Demo2.HTML and resultnew.html files.**
- The static folder which contains CSS folder which contains styles.css.

Milestone 1: Data Collection

Now that you have set up your environment , installed necessary packages, and gained knowledge let's start the project building

The first step towards your project building is Data Collection

ML depends heavily on data, without data, it is impossible for an “AI” to learn. It is the most crucial aspect that makes algorithm training possible. In Machine

Learning projects, we need a training **data set.** It is the actual **data set** used to train the model for performing various actions.

Activity 1: Download the dataset

The dataset used for this project was obtained from Kaggle . Please refer to the link given below to download the data set and to know about the dataset

Our base data consists of two CSV files containing information about 5000 movies and the list of their cast and crew members

- tmdb_5000_movies.csv: Contains information like the score, title, date_of_release, genres, production companies, budget, revenue, genres etc.
- tmdb_5000_credits.csv: Contains information of the cast and crew for each movie.
-

Milestone 2: Visualizing and analysing the data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analysing techniques.

- **Note: There is n number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques**

Data Pre-Processing

The data collected will be raw data and that is to be cleaned. this activity lets you clean your collected data. you need to follow the below steps to clean your data

- Import the Libraries.
- Reading the dataset.

- Exploratory Data Analysis
- Converting json objects to strings
- Checking for Null Values.
- Data Visualization.
- Dropping the columns
- Label Encoding
- Splitting the Dataset into Dependent and Independent variable.
- Feature scaling
- Splitting Data into Train and Test.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image

Pandas: It is a python library mainly used for data manipulation.
NumPy: This python library is used for numerical analysis.
Counter: Python Counter is a container that will hold the count of each of the elements present in the container.
Matplotlib and Seaborn: Both are the data visualization library used for plotting graphs which will help us for understanding the data.
Accuracy score: used in classification type problem and for finding accuracy it is used.
R2 Score: Coefficient of Determination or R² is another metric used for evaluating the performance of a regression model. The metric helps us to compare our current model with a constant baseline and tells us how much our model is better.
Literal_eval: we can use ast.literal_eval() to evaluate the string as a python expression.
Pickle: to serialize your machine learning algorithms and save the serialized format to a file.
Word cloud: to create visualizations with text data

Activity 2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of csv file.

credits=pd.read_csv(r"C:\Users\myaka\Downloads\tmdb_5000_credits.csv")

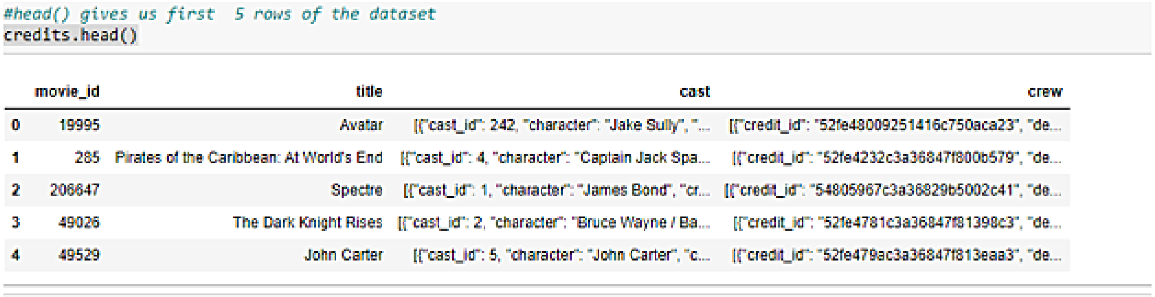
C:\Users\myaka\Downloads\Movie box office gross prediction-20220906T101041Z-001 (2)\Movie box

movies_df=pd.read_csv(r"C:\Users\myaka\Downloads\tmdb_5000_movies (2).csv")

Exploratory Data Analysis

Exploratory data analysis is an approach to analyzing data sets to summarize their main characteristics, often with visual methods and used for determining how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis or check assumptions.

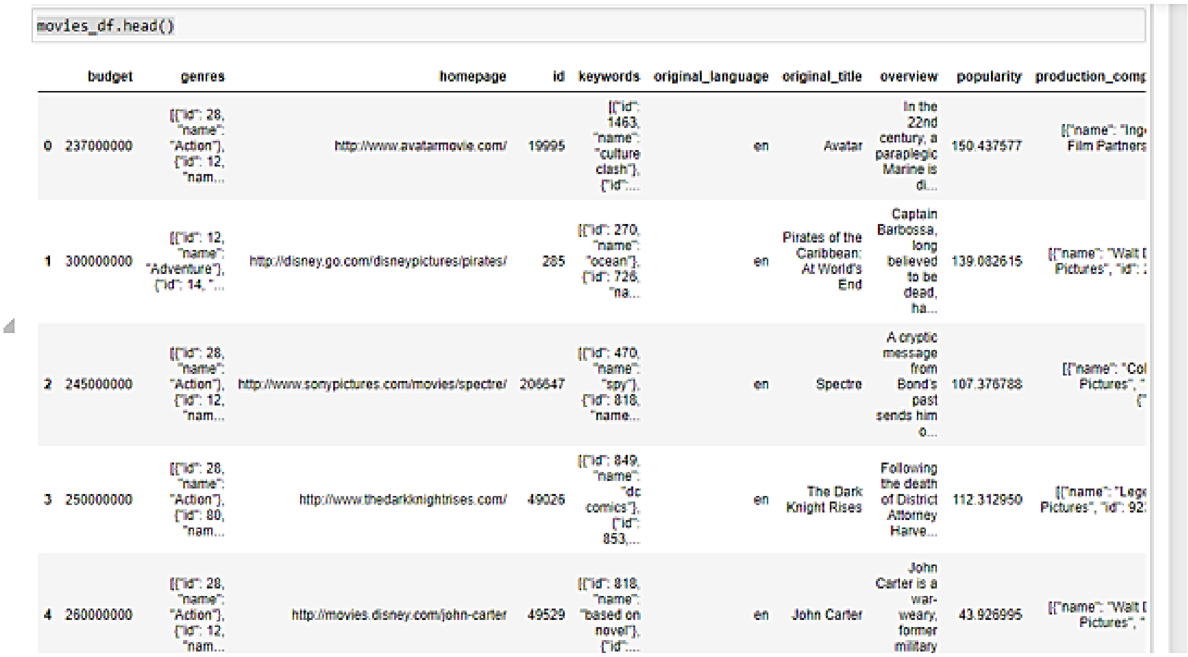
head() : To check first five rows of dataset, we have a function call head( ).



```
#head() gives us first  5 rows of the dataset
credits.head()
```

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

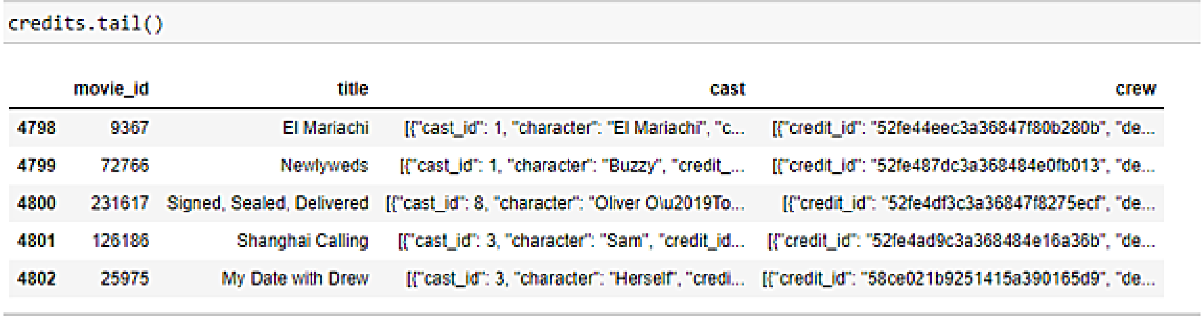
movies_df.head()



```
movies_df.head()
```

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_comp
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ing Film Partners
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt [Pictures", "id": :
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[{"name": "Col Pictures", " {"
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950	[{"name": "Lege Pictures", "id": 92:
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	John Carter	John Carter is a war-weary, former military	43.926995	[{"name": "Walt [Pictures", "

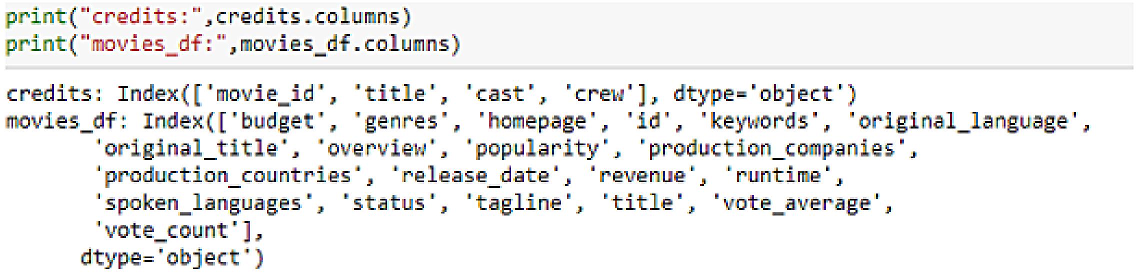
Tail(): To check last five rows of dataset, we have a function call tail().



```
credits.tail()
```

	movie_id	title	cast	crew
4798	9367	El Mariachi	[{"cast_id": 1, "character": "El Mariachi", "c...	[{"credit_id": "52fe44eec3a36847f80b280b", "de...
4799	72766	Newlyweds	[{"cast_id": 1, "character": "Buzzy", "credit_...	[{"credit_id": "52fe487dc3a368484e0fb013", "de...
4800	231617	Signed, Sealed, Delivered	[{"cast_id": 8, "character": "Oliver O\u2019To...	[{"credit_id": "52fe4df3c3a36847f8275ecf", "de...
4801	126186	Shanghai Calling	[{"cast_id": 3, "character": "Sam", "credit_id...	[{"credit_id": "52fe4ad9c3a368484e16a36b", "de...
4802	25975	My Date with Drew	[{"cast_id": 3, "character": "Herself", "credi...	[{"credit_id": "58ce021b9251415a390165d9", "de...

columns:For finding the names of the columns present in the dataset we make use of columns

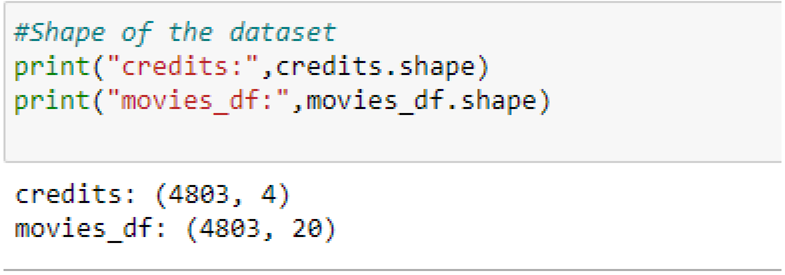


```
print("credits:",credits.columns)
print("movies_df:",movies_df.columns)
```

```
credits: Index(['movie_id', 'title', 'cast', 'crew'], dtype='object')
movies_df: Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'original_language',
       'original_title', 'overview', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'vote_average',
       'vote_count'],
      dtype='object')
```

data.columns will return you all the column names which are present in your data .

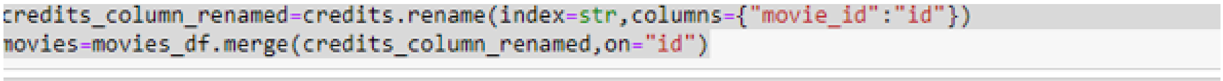
shape:shape function is used to know number of columns and rows in the dataset



```
#Shape of the dataset
print("credits:",credits.shape)
print("movies_df:",movies_df.shape)
```

```
credits: (4803, 4)
movies_df: (4803, 20)
```

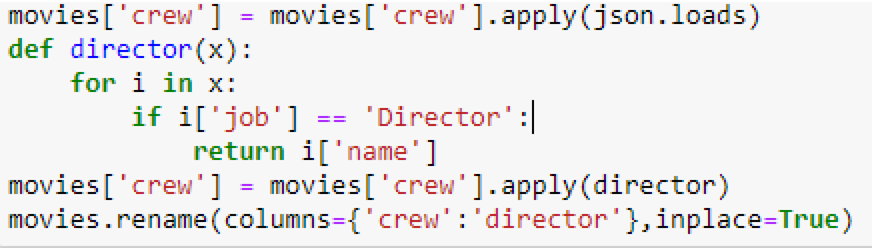
Merging credits and movies_df dataset by using common key-id
We notice that id column in movies_df is similar to the movie_id in the credits dataset. Let us merge these two datasets, movie_df, and credits using a common key. Therefore rename the column movie_id in credits to id.



```
credits_column_renamed=credits.rename(index=str,columns={"movie_id":"id"})
movies=movies_df.merge(credits_column_renamed,on="id")
```

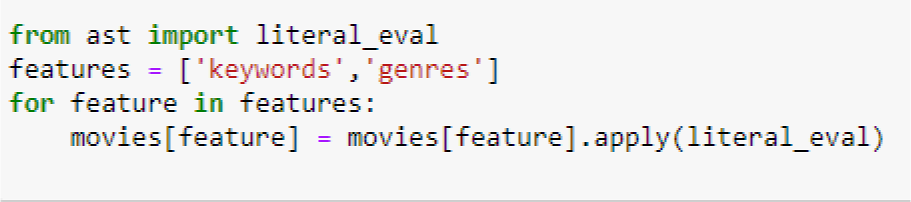
Converting Json Objects To Strings

Crew column



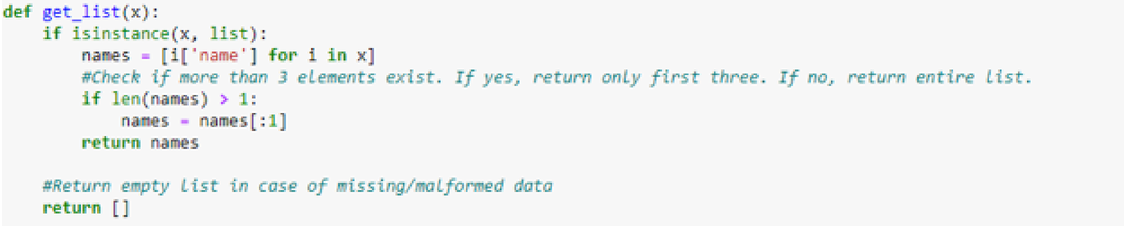
```
movies['crew'] = movies['crew'].apply(json.loads)
def director(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
movies['crew'] = movies['crew'].apply(director)
movies.rename(columns={'crew':'director'},inplace=True)
```

Keywords and genre column:



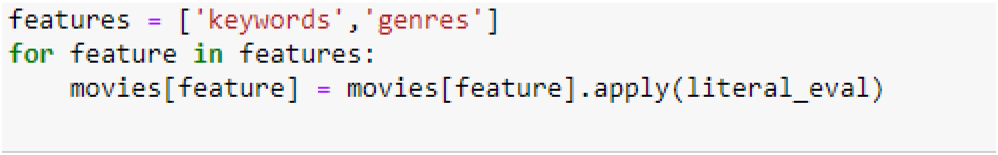
```
from ast import literal_eval
features = ['keywords','genres']
for feature in features:
    movies[feature] = movies[feature].apply(literal_eval)
```

Let us define a list to get top 1 element from the genres and keyword columns



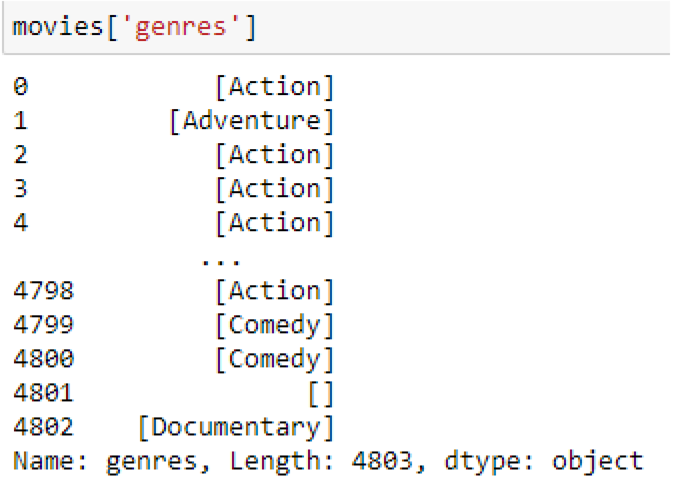
```
def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]
        #Check if more than 3 elements exist. If yes, return only first three. If no, return entire list.
        if len(names) > 1:
            names = names[:1]
        return names

    #Return empty list in case of missing/malformed data
    return []
```

Applying the defined function get_list(x) to required features keywords and genres



```
features = ['keywords','genres']
for feature in features:
    movies[feature] = movies[feature].apply(literal_eval)
```

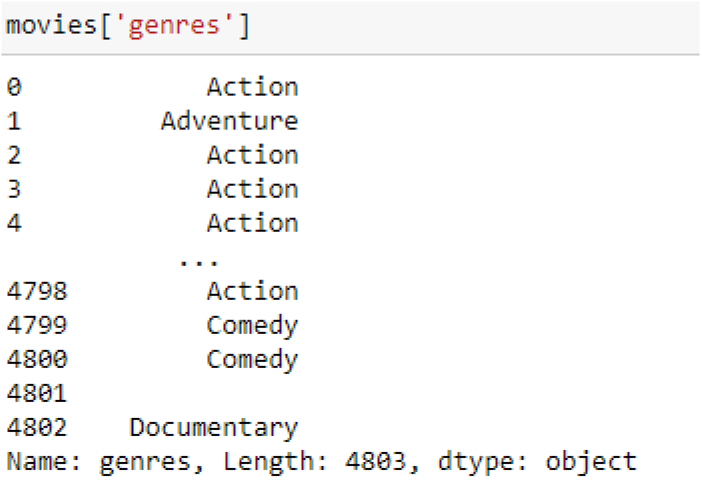
Now Let us see the column genres



```
movies['genres']
```

```
0              [Action]
1           [Adventure]
2              [Action]
3              [Action]
4              [Action]
              ...
4798           [Action]
4799           [Comedy]
4800           [Comedy]
4801                 []
4802      [Documentary]
Name: genres, Length: 4803, dtype: object
```

We notice that the genres is in [].Let us strip [] from our column genres..

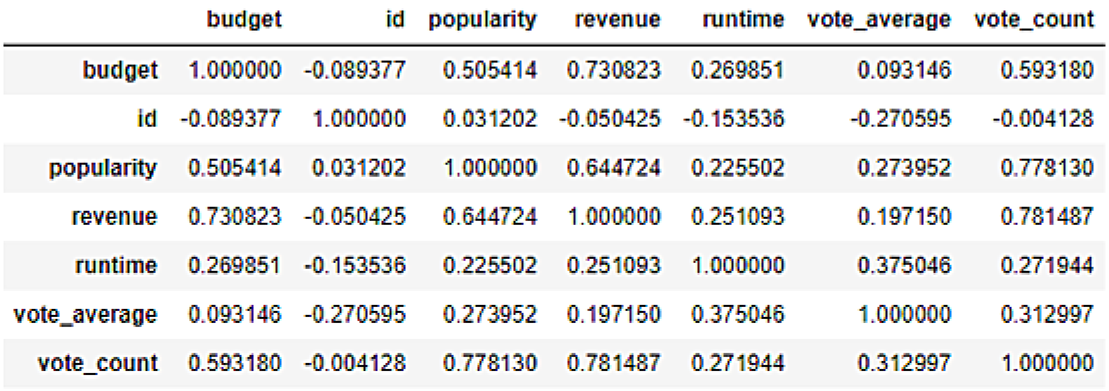


```
movies['genres']
```

```
0              Action
1           Adventure
2              Action
3              Action
4              Action
              ...
4798           Action
4799           Comedy
4800           Comedy
4801
4802      Documentary
Name: genres, Length: 4803, dtype: object
```

Finding the Correlation

Correlation is a statistic that measures the degree to which two variables move in relation to each other.



	budget	id	popularity	revenue	runtime	vote_average	vote_count
budget	1.000000	-0.089377	0.505414	0.730823	0.269851	0.093146	0.593180
id	-0.089377	1.000000	0.031202	-0.050425	-0.153536	-0.270595	-0.004128
popularity	0.505414	0.031202	1.000000	0.644724	0.225502	0.273952	0.778130
revenue	0.730823	-0.050425	0.644724	1.000000	0.251093	0.197150	0.781487
runtime	0.269851	-0.153536	0.225502	0.251093	1.000000	0.375046	0.271944
vote_average	0.093146	-0.270595	0.273952	0.197150	0.375046	1.000000	0.312997
vote_count	0.593180	-0.004128	0.778130	0.781487	0.271944	0.312997	1.000000

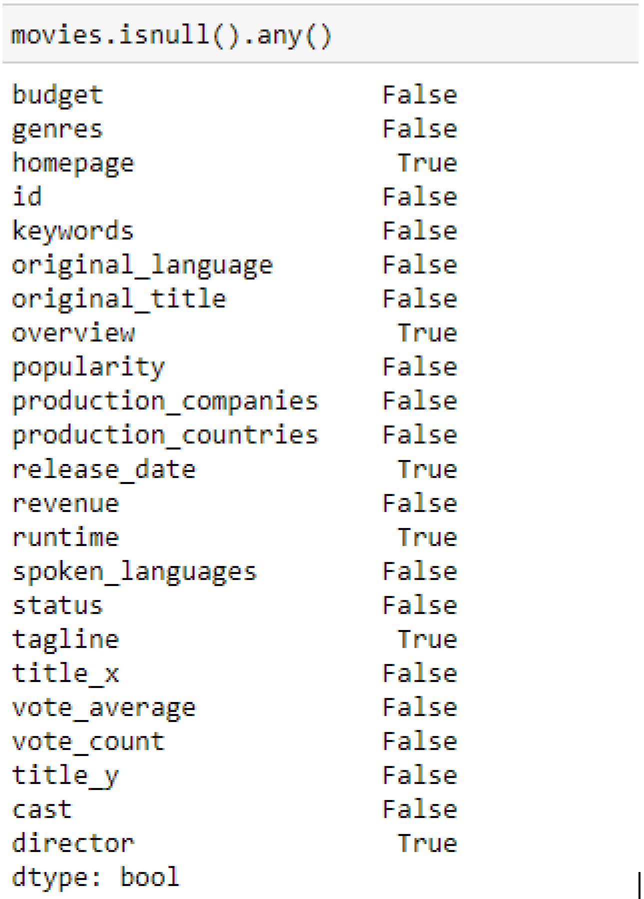
Checking For Null Values

After loading it is important to check the complete information of data as it can indicate many of the hidden information such as null values in a column or a row

Check whether any null values are there or not. if it is present then the following can be done,

- Imputing data using the Imputation method in sklearn
- Filling NaN values with mean, median and mode using fillna() method.

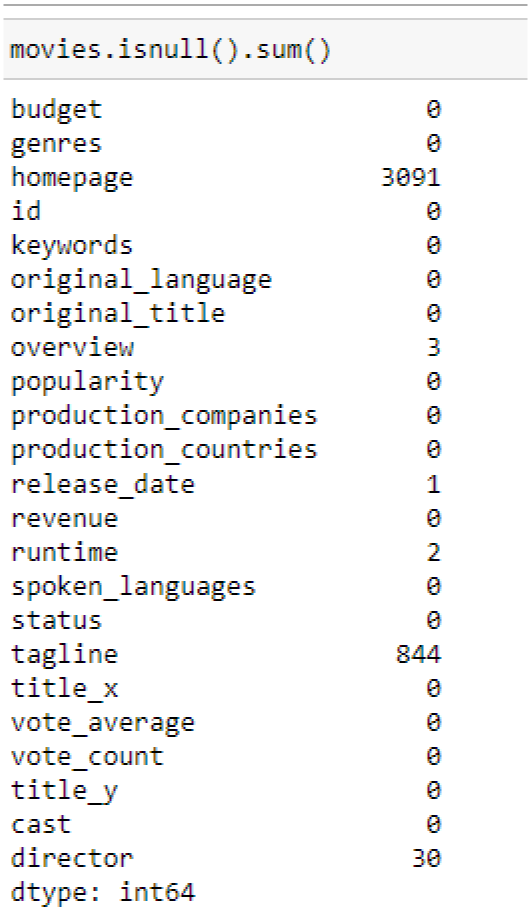
We will be using isnull().any() method to see which column has missing values.



```
movies.isnull().any()
```

```
budget                  False
genres                  False
homepage                 True
id                      False
keywords                False
original_language       False
original_title          False
overview                 True
popularity              False
production_companies    False
production_countries    False
release_date             True
revenue                 False
runtime                  True
spoken_languages        False
status                  False
tagline                  True
title_x                 False
vote_average            False
vote_count              False
title_y                 False
cast                    False
director                 True
dtype: bool
```

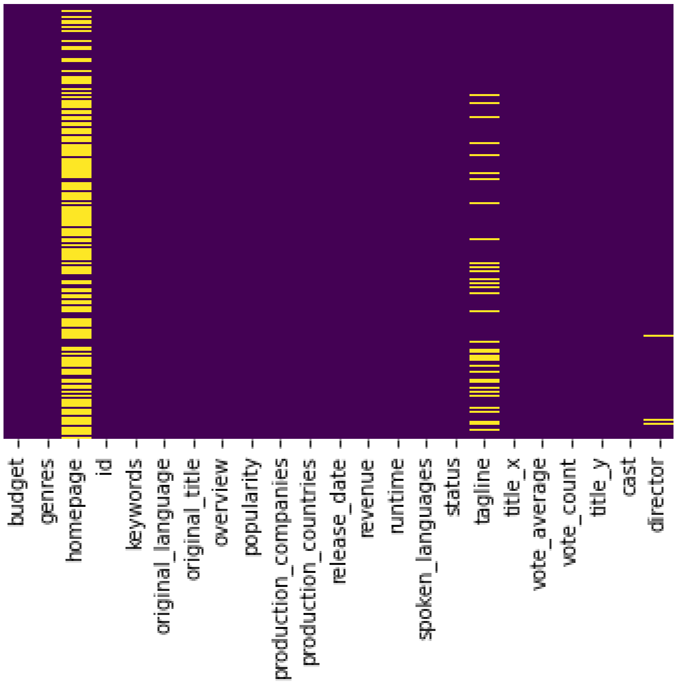
“True” indicates that the particular column has missing values, we can also see the count of null values in each column by using isnull().sum().



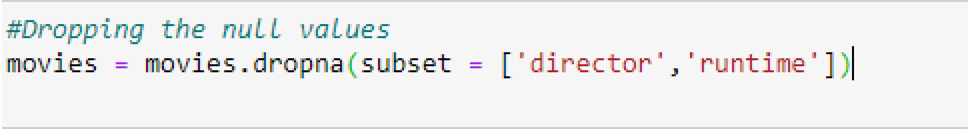
```
movies.isnull().sum()
```

```
budget                     0
genres                     0
homepage                3091
id                         0
keywords                   0
original_language          0
original_title             0
overview                   3
popularity                 0
production_companies       0
production_countries       0
release_date               1
revenue                    0
runtime                    2
spoken_languages           0
status                     0
tagline                  844
title_x                    0
vote_average               0
vote_count                 0
title_y                    0
cast                       0
director                  30
dtype: int64
```

As our data is having null values so we have to handle the null values . Let us visualize the null values using the seaborn heatmap libraries .

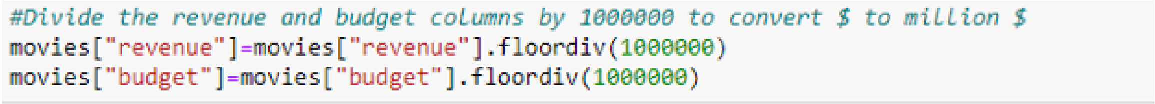


We can see,home page and tagline have null values,but they have least importance and can be ignored.But ,as director and run time are important features and has only 30 nulls and 2 nulls,Dropping the rows with null values



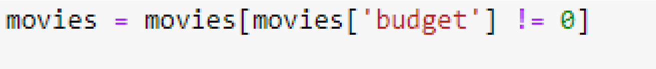
```
#Dropping the null values
movies = movies.dropna(subset = ['director','runtime'])
```

The numbers in the budget and revenue are too big, compromising its readability. Let's convert the unit of the budget and revenue columns from ???????????????????????????????????? first



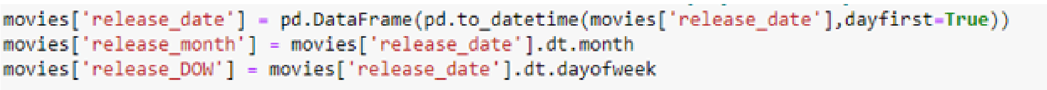
```
#Divide the revenue and budget columns by 1000000 to convert $ to million $
movies["revenue"]=movies["revenue"].floordiv(1000000)
movies["budget"]=movies["budget"].floordiv(1000000)
```

As there cannot be any movie with budget as 0,let us remove the rows with budget as 0



```
movies = movies[movies['budget'] != 0]
```

Let us extract the date, month, and day of the week from release_date column



```
movies['release_date'] = pd.DataFrame(pd.to_datetime(movies['release_date'],dayfirst=True))
movies['release_month'] = movies['release_date'].dt.month
movies['release_DOW'] = movies['release_date'].dt.dayofweek
```

Data Visualization

- Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends, and correlations that might go undetected in text-based data.
- Understanding your data and the relationship presents within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that wasn't visualized and understood properly.
- To visualize the dataset we need libraries called Matplotlib and Seaborn.
- The Matplotlib library is a Python 2D plotting library that allows you to generate plots, scatter plots, histograms, bar charts etc.

Let's visualize our data using Matplotlib and seaborn library.

Before diving into the code, let's look at some of the basic properties we will be using when plotting.

xlabel: Set the label for the x-axis.

ylabel: Set the label for the y-axis.

title: Set a title for the axes.

Legend: Place a legend on the axes.

Let’s visualize our data using Matplotlib and seaborn library.

Univariate Analysis

Univariate analysis is the simplest form of data analysis where the data being analyzed contains only one variable.

Bivariate Analysis

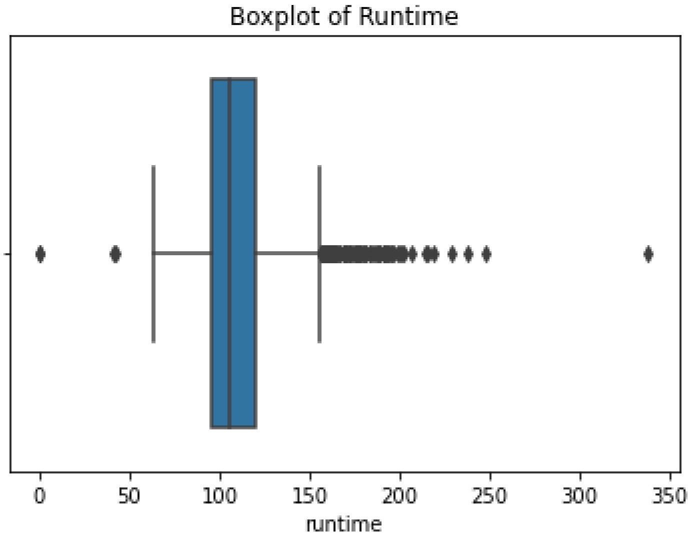
It involves the analysis of two variables (often denoted as X, Y), for the purpose of determining the empirical relationship between them.

1.Outliers Detection using boxplot

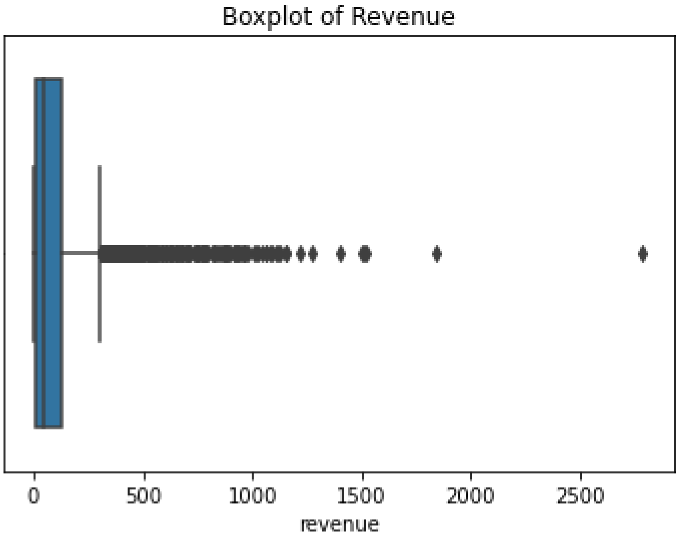
Outliers are observations in a dataset that don't fit in some way or you can say those values are of no use and also may effect our results.

So with the help of boxplot we can visualize and check whether the data contains any outliers or not.

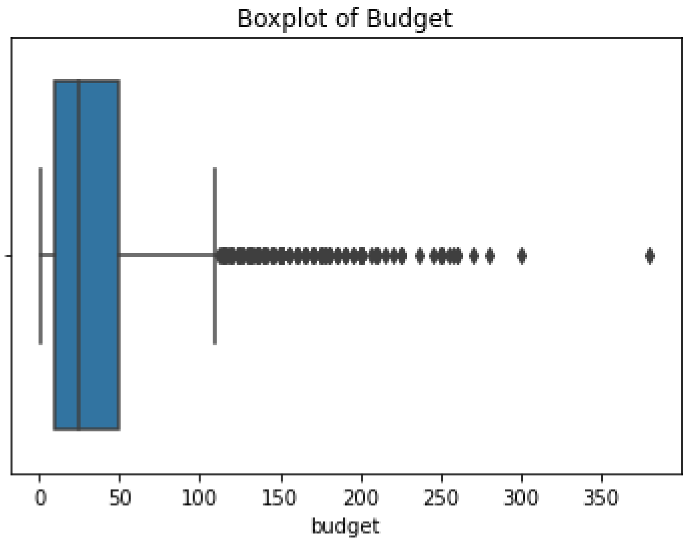
Box plot for runtime column



Box plot for revenue column



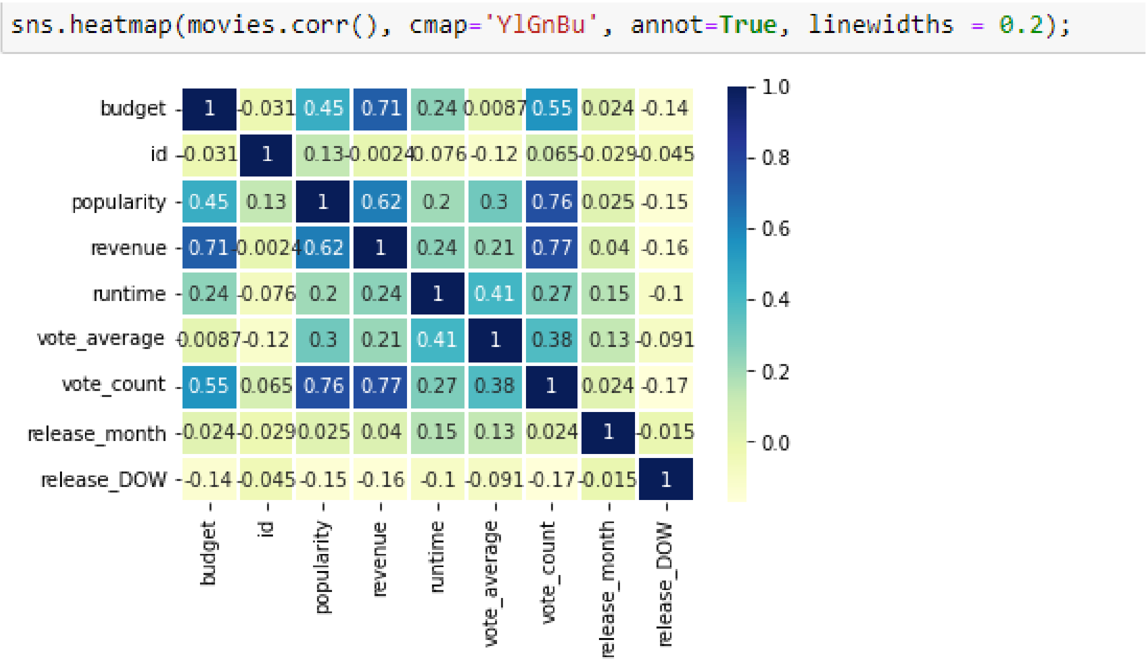
Box plot for budget column



2.Finding correlation between the independent Columns

Correlation is a statistical relationship between two variables and it could be positive, meaning both variables move in the same direction, or negative, meaning that when one variable's value increases, the other variables' values decrease.

With the help of seaborn heatmap we will be plotting the heatmap and for finding the correlation between variable we have corr() available.



```
sns.heatmap(movies.corr(), cmap='YlGnBu', annot=True, linewidths = 0.2);
```

	budget	id	popularity	revenue	runtime	vote_average	vote_count	release_month	release_DOW
budget	1	-0.031	0.45	0.71	0.24	0.0087	0.55	0.024	-0.14
id	-0.031	1	0.13	-0.0024	-0.076	-0.12	0.065	-0.029	-0.045
popularity	0.45	0.13	1	0.62	0.2	0.3	0.76	0.025	-0.15
revenue	0.71	-0.0024	0.62	1	0.24	0.21	0.77	0.04	-0.16
runtime	0.24	-0.076	0.2	0.24	1	0.41	0.27	0.15	-0.1
vote_average	0.0087	-0.12	0.3	0.21	0.41	1	0.38	0.13	-0.091
vote_count	0.55	0.065	0.76	0.77	0.27	0.38	1	0.024	-0.17
release_month	0.024	-0.029	0.025	0.04	0.15	0.13	0.024	1	-0.015
release_DOW	-0.14	-0.045	-0.15	-0.16	-0.1	-0.091	-0.17	-0.015	1

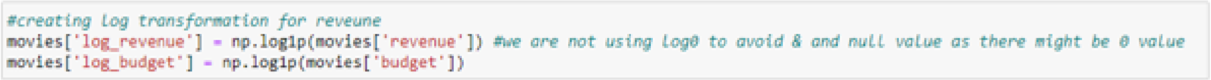
we see that the output to be predicted "Revenue" depends more on budget,popularity and vote_count columns as they have correlation >=0.5

Well, this explains that revenue is strongly correlated with the budget of the movies, and runtime is least correlated with revenue. So we can say that Budget of the movie is directly related for the revenue generated.

3.Relationship between budget and revenue:

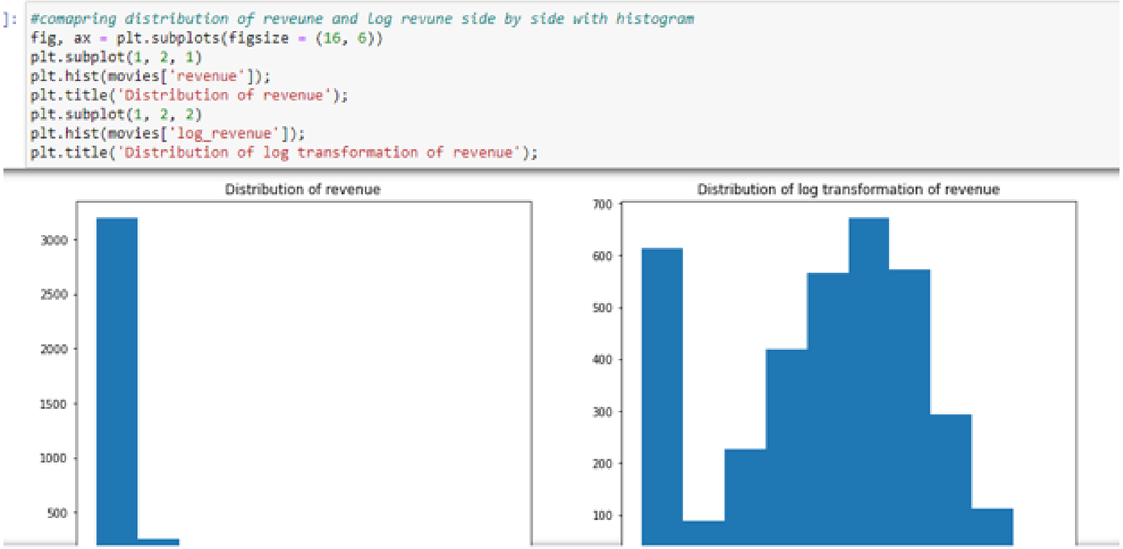
Let's plot the distribution of revenue and budget.

The data in the revenue column is very skewed and therefore it is difficult to draw a conclusion from this graph. we need to normalize this data. Therefore we will be normalizing it using log transformation.



```
#creating log transformation for reveune
movies['log_revenue'] = np.log1p(movies['revenue']) #we are not using log0 to avoid & and null value as there might be 0 value
movies['log_budget'] = np.log1p(movies['budget'])
```

Let’s compare our value Revenue and log transformation of revenue using the seaborn library.

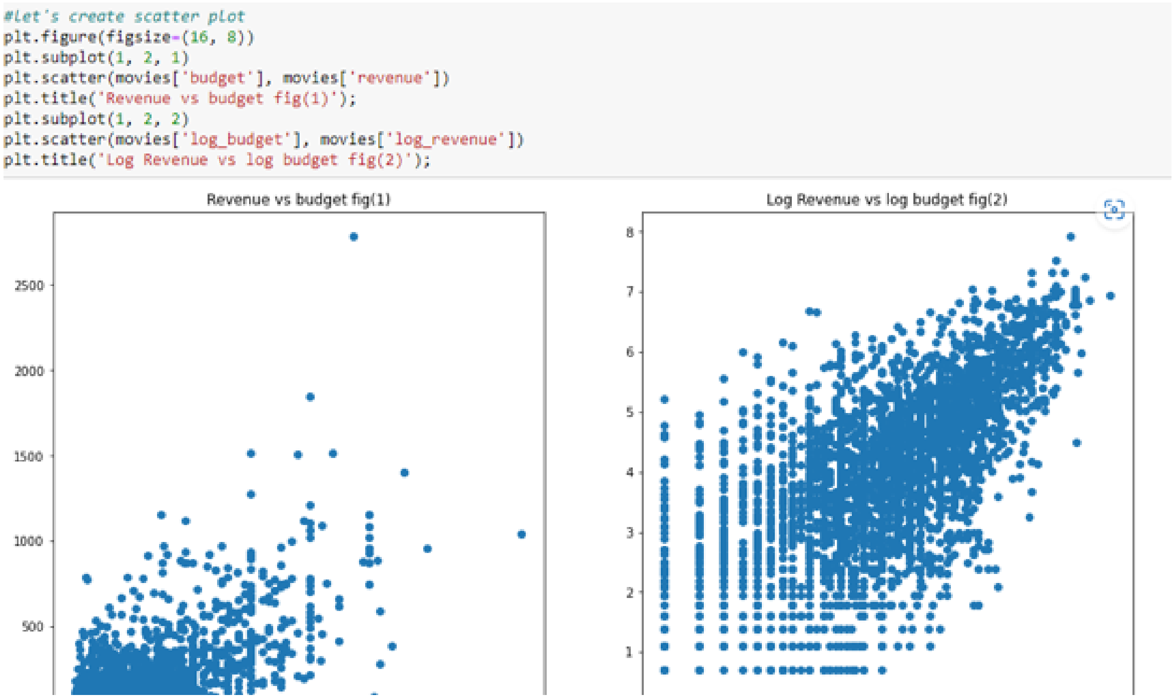


```
]: #comapring distribution of reveune and log revune side by side with histogram
   fig, ax = plt.subplots(figsize = (16, 6))
   plt.subplot(1, 2, 1)
   plt.hist(movies['revenue']);
   plt.title('Distribution of revenue');
   plt.subplot(1, 2, 2)
   plt.hist(movies['log_revenue']);
   plt.title('Distribution of log transformation of revenue');
```

We can see that original distribution i.e (one without log) is extremely skewed. We used the log transformation method and made data normally distributed which has less skewness and kurtosis.

4.Relationship between Film Revenue and Budget.

Let’s examine the relationship using a scatter plot.



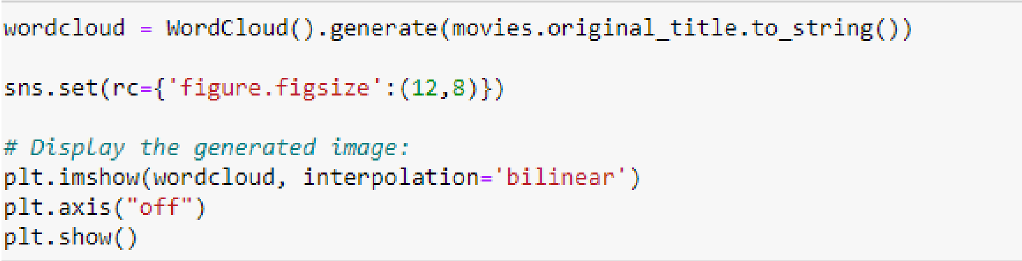
```
#let's create scatter plot
plt.figure(figsize=(16, 8))
plt.subplot(1, 2, 1)
plt.scatter(movies['budget'], movies['revenue'])
plt.title('Revenue vs budget fig(1)');
plt.subplot(1, 2, 2)
plt.scatter(movies['log_budget'], movies['log_revenue'])
plt.title('Log Revenue vs log budget fig(2)');
```

Fig(1, left side): we can see that they are a somewhat correlation between budget and revenue, but we are not clear.

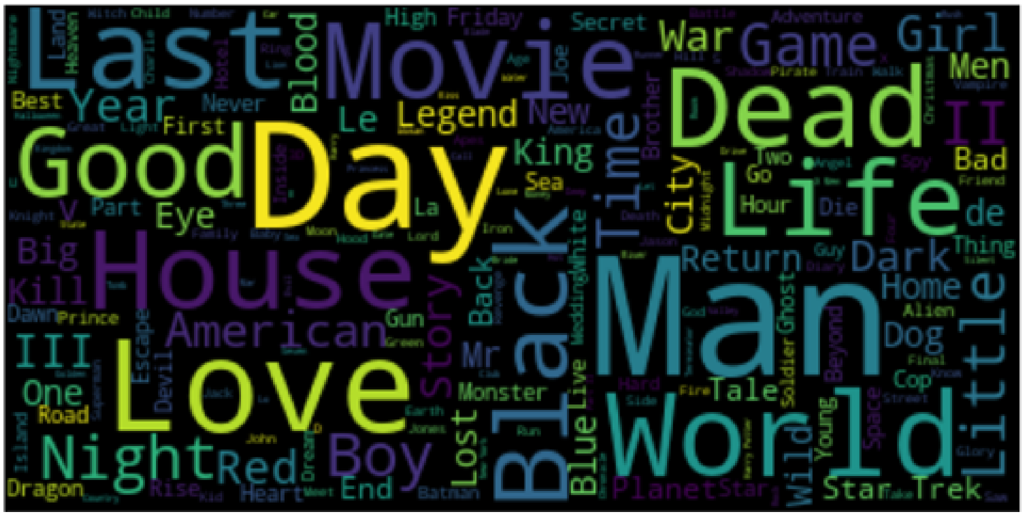
Fig(2, right side): however this plot indicates that there is a correlation between both variables that are log transformation of revenue and log transformation of budget.

5.What are the frequent Words in Film Titles:

Word cloud is a data visualization technique used for the representation of text data in which the size of each word indicates its frequency or importance.

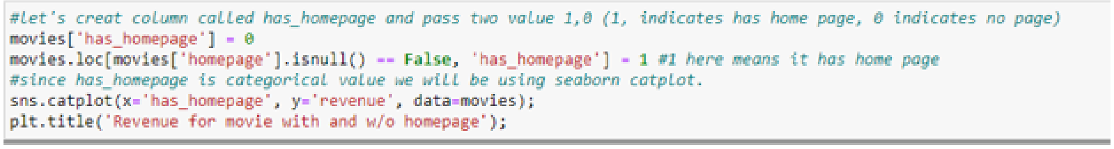


```
wordcloud = WordCloud().generate(movies.original_title.to_string())

sns.set(rc={'figure.figsize':(12,8)})

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

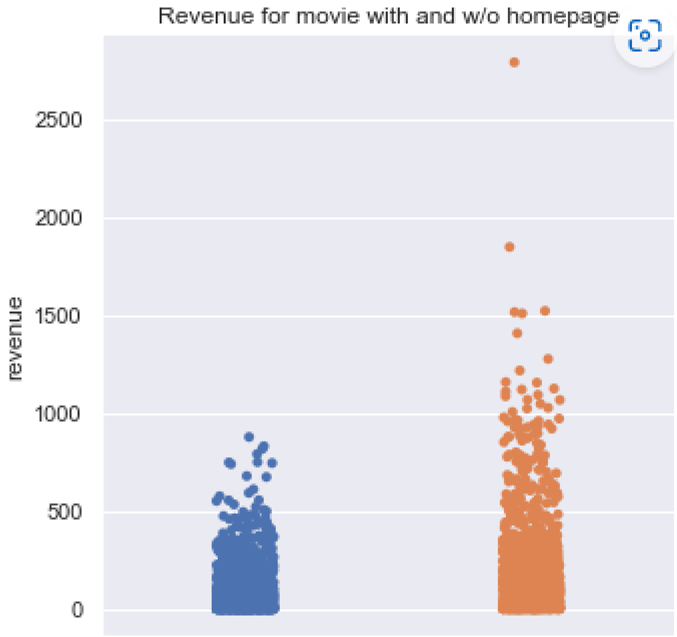


6.Relationship between the home page and Revenue

Let’s find if having a home page affects revenue or not. And for that lets first find out movies have a home page or not.
let's create column called has_homepage and pass two value 1,0 (1, indicates has home page, 0 indicates no page)
Let us use seaborn catplot for plotting this graph as home page is a categorical variable

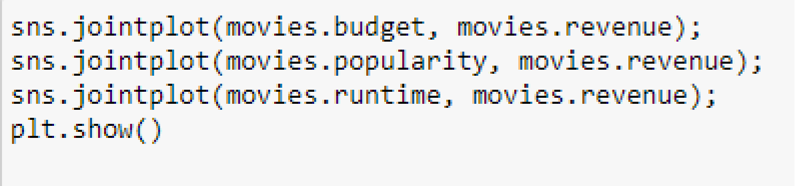


```
#let's creat column called has_homepage and pass two value 1,0 (1, indicates has home page, 0 indicates no page)
movies['has_homepage'] = 0
movies.loc[movies['homepage'].isnull() == False, 'has_homepage'] = 1 #1 here means it has home page
#since has_homepage is categorical value we will be using seaborn catplot.
sns.catplot(x='has_homepage', y='revenue', data=movies);
plt.title('Revenue for movie with and w/o homepage');
```



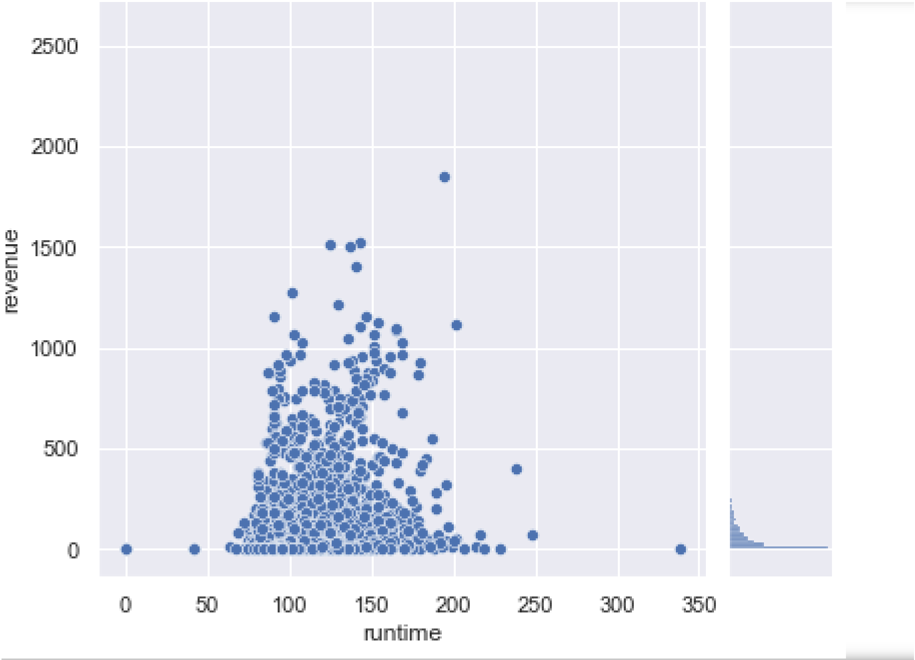
7.Relationship between release_month and revenue

Lets plot the revenue generated each month, and find if it affects movie generated revenue or not. Let us plot a join tplot between release_month and revenue.



```
sns.jointplot(movies.budget, movies.revenue);
sns.jointplot(movies.popularity, movies.revenue);
sns.jointplot(movies.runtime, movies.revenue);
plt.show()
```

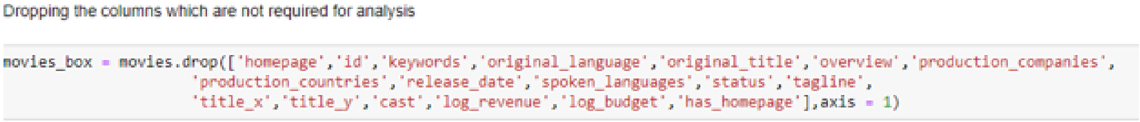
The output of the above code is



Dropping The Columns

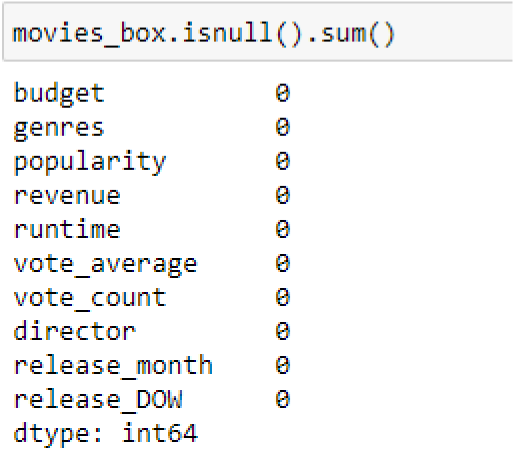
Let us drop the column ,which are not required for predicting the output column revenue.

Check for null values in the new dataset movies_box



Dropping the columns which are not required for analysis

```
movies_box = movies.drop(['homepage','id','keywords','original_language','original_title','overview','production_companies',
                        'production_countries','release_date','spoken_languages','status','tagline',
                        'title_x','title_y','cast','log_revenue','log_budget','has_homepage'],axis = 1)
```



```
movies_box.isnull().sum()
```

```
budget           0
genres           0
popularity       0
revenue          0
runtime          0
vote_average     0
vote_count       0
director         0
release_month    0
release_DOW      0
dtype: int64
```

We can see that now our data is free from null values.

Label Encoding

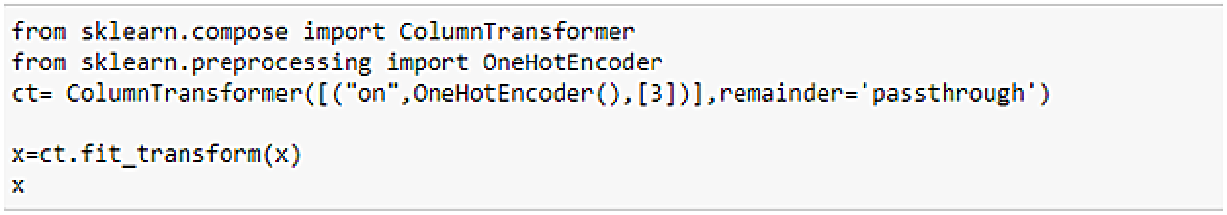
Typically, any structured dataset includes multiple columns with combination of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text. That’s essentially the case with Machine Learning algorithms too.We need to convert each text category to numbers in order for the machine to process those using mathematical equations.

How should we handle categorical variables? There are Multiple way to handle, but will see one of it is LabelEncoding.

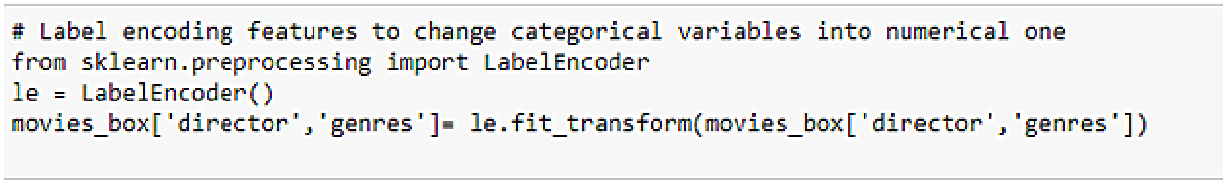
- Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.

Let’s see how to implement label encoding in Python using the scikit-learn library.

As we have to convert only the text class category columns, we first select it then we will implement Label Encoding to it.



```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct= ColumnTransformer([("on",OneHotEncoder(),[3])],remainder='passthrough')

x=ct.fit_transform(x)
x
```

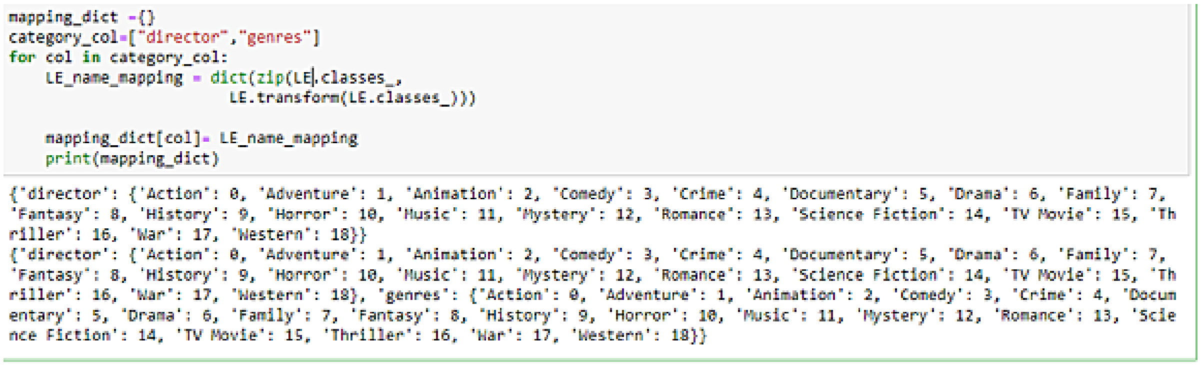


```
# Label encoding features to change categorical variables into numerical one
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
movies_box['director','genres']= le.fit_transform(movies_box['director','genres'])
```

In the above code we are looping through all the selected text class categorical columns and performing label encoding.

We notice the output of the above code, after performing label encoding alphabetical classes is converted to numeric.

-



```
mapping_dict ={}
category_col=["director","genres"]
for col in category_col:
    LE_name_mapping = dict(zip(LE.classes_,
                        LE.transform(LE.classes_)))

    mapping_dict[col]= LE_name_mapping
    print(mapping_dict)
```

```
{'director': {'Action': 0, 'Adventure': 1, 'Animation': 2, 'Comedy': 3, 'Crime': 4, 'Documentary': 5, 'Drama': 6, 'Family': 7,
'Fantasy': 8, 'History': 9, 'Horror': 10, 'Music': 11, 'Mystery': 12, 'Romance': 13, 'Science Fiction': 14, 'TV Movie': 15, 'Th
riller': 16, 'War': 17, 'Western': 18}}
{'director': {'Action': 0, 'Adventure': 1, 'Animation': 2, 'Comedy': 3, 'Crime': 4, 'Documentary': 5, 'Drama': 6, 'Family': 7,
'Fantasy': 8, 'History': 9, 'Horror': 10, 'Music': 11, 'Mystery': 12, 'Romance': 13, 'Science Fiction': 14, 'TV Movie': 15, 'Th
riller': 16, 'War': 17, 'Western': 18}, 'genres': {'Action': 0, 'Adventure': 1, 'Animation': 2, 'Comedy': 3, 'Crime': 4, 'Docum
entary': 5, 'Drama': 6, 'Family': 7, 'Fantasy': 8, 'History': 9, 'Horror': 10, 'Music': 11, 'Mystery': 12, 'Romance': 13, 'Scie
nce Fiction': 14, 'TV Movie': 15, 'Thriller': 16, 'War': 17, 'Western': 18}}
```

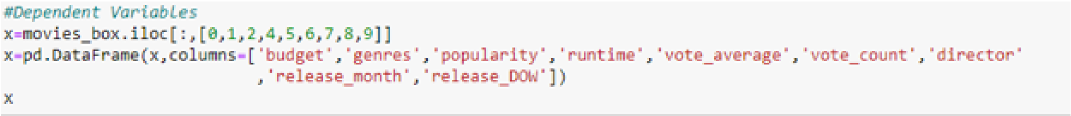
Splitting The Dataset Into Dependent And Independent Variable.

Let’s split our dataset into independent and dependent variables.

1. The independent variable in the dataset would be considered as 'x' and the 'budget','genres','popularity','runtime','vote_average','vote_count','director','release_month','release_DOW' columns would be considered as independent variable.
2. The dependent variable in the dataset would be considered as 'y' and the 'revenue' column is considered as dependent variable.

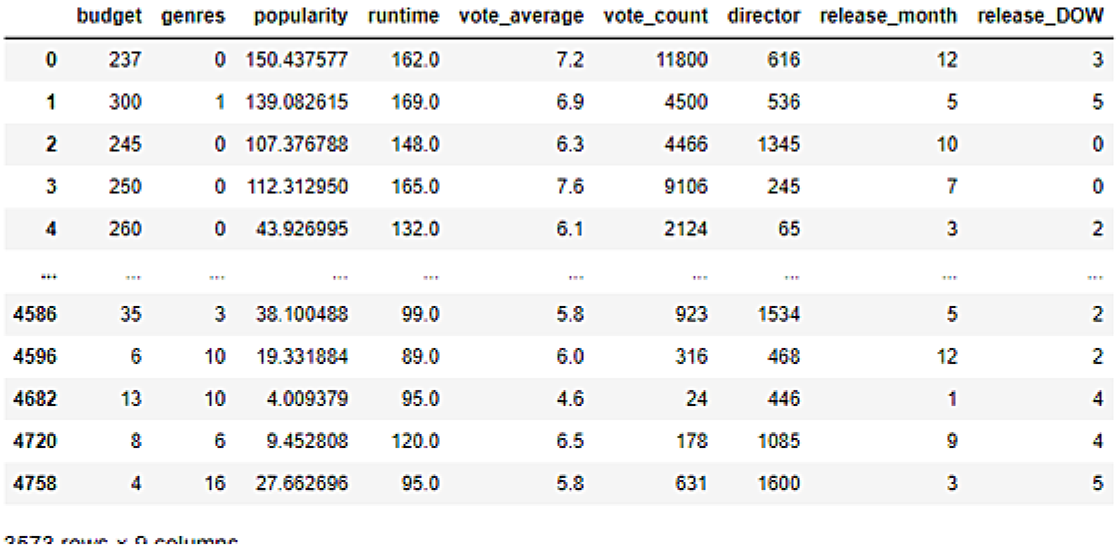
Now we will split the data of independent variables,

Independent_variables



```
#Dependent Variables
x=movies_box.iloc[:,[0,1,2,4,5,6,7,8,9]]
x=pd.DataFrame(x,columns=['budget','genres','popularity','runtime','vote_average','vote_count','director'
                    ,'release_month','release_DOW'])
x
```

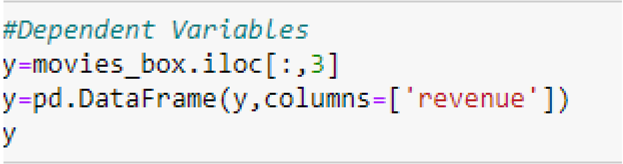
The input features selected are



	budget	genres	popularity	runtime	vote_average	vote_count	director	release_month	release_DOW
0	237	0	150.437577	162.0	7.2	11800	616	12	3
1	300	1	139.082615	169.0	6.9	4500	536	5	5
2	245	0	107.376788	148.0	6.3	4466	1345	10	0
3	250	0	112.312950	165.0	7.6	9106	245	7	0
4	260	0	43.926995	132.0	6.1	2124	65	3	2
...	...	...	...	...	...	...	...	...	...
4586	35	3	38.100488	99.0	5.8	923	1534	5	2
4596	6	10	19.331884	89.0	6.0	316	468	12	2
4682	13	10	4.009379	95.0	4.6	24	446	1	4
4720	8	6	9.452808	120.0	6.5	178	1085	9	4
4758	4	16	27.662696	95.0	5.8	631	1600	3	5

3573 rows × 9 columns

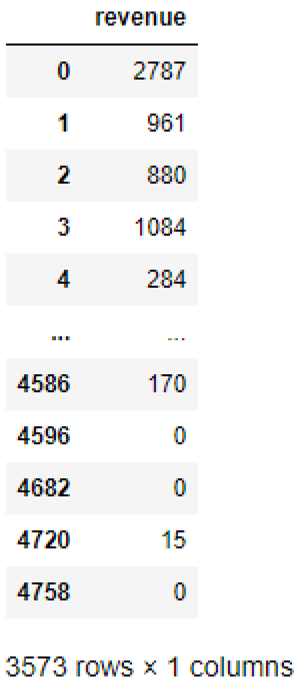
Dependent_variables:



```
#Dependent Variables
y=movies_box.iloc[:,3]
y=pd.DataFrame(y,columns=['revenue'])
y
```

In the above code we are creating DataFrame of the independent variable **x** with our selected columns and for dependent variable **y** we are only taking the **revenue** column.

The output feature selected is



	revenue
0	2787
1	961
2	880
3	1084
4	284
...	...
4586	170
4596	0
4682	0
4720	15
4758	0

3573 rows × 1 columns

Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data.

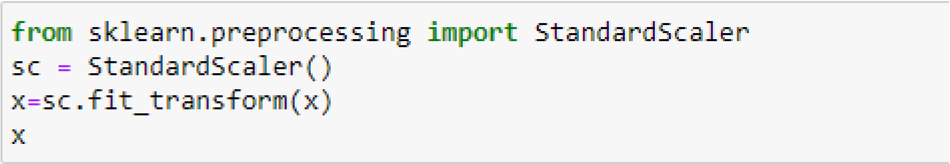
Many machine learning algorithms perform better when numerical input features are scaled to a standard range.

There are two types of scaling..They are

1.Standard Scaling: Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.

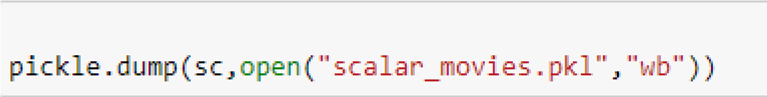
2. Min-Max Scaling: Min-Max is rescaling of the data from the original range so that all values are within the new range of 0 and 1.
Let us scale our data to improve performance. Create a object sc as instance

of the class Standard scalar and fit the same to our data frame



```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x=sc.fit_transform(x)
x
```

Save the scalar object sc to reuse it in our application building by using pickle library



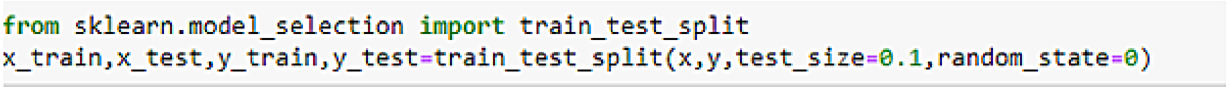
```
pickle.dump(sc,open("scalar_movies.pkl","wb"))
```

Split The Dataset Into Train Set And Test Set

- When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.
- But the question is, how do you split the data? You can’t possibly manually split the dataset into two sets. And you also have to make sure you split the data in a random manner. To help us with this task, the Scikit library provides a tool, called the Model Selection library. There is a class in the library which is,**‘train_test_split.’** Using this we can easily split the dataset into the training and the testing datasets in various

proportions.

- The train-test split is a technique for evaluating the performance of a machine learning algorithm.
- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.
- In general you can allocate 80% of the dataset to training set and the remaining 20% to test set.We will create 4 sets— X_train (training part of the matrix of features), X_test (test part of the matrix of features), Y_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices), Y_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices.
- There are a few other parameters that we need to understand before we use the class:
- **test_size** — this parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset
- **train_size** — you have to specify this parameter only if you're not specifying the test_size. This is the same as test_size, but instead you tell the class what percent of the dataset you want to split as the training set.
- **random_state** — here you pass an integer, which will act as the seed for the random number generator during the split. Or, you can also pass an instance of the Random_state class, which will become the number generator. If you don't pass anything, the Random_state instance used by np.random will be used instead.
- Now split our dataset into train set and test using train_test_split class from scikit learn library.



```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=0)
```

Model Building

Now that our data is cleaned let's build the model

Predictive modeling is a mathematical approach to create a statistical model to forecast future behavior based on input test data.

Steps involved in predictive modeling:

Algorithm Selection:

When we have the structured dataset, and we want to estimate the continuous or categorical outcome then we use supervised machine learning methodologies like regression and classification techniques. When we have unstructured data and want to predict the clusters of items to which a particular input test sample belongs, we use unsupervised algorithms. An actual data scientist applies multiple algorithms to get a more accurate model.

Train Model:

After assigning the algorithm and getting the data handy, we train our model using the input data applying the preferred algorithm. It is an action to determine the correspondence between independent variables, and the prediction targets.

Model Prediction:

We make predictions by giving the input test data to the trained model. We measure the accuracy by using a cross-validation strategy or ROC curve which performs well to derive model output for test data.

Model building includes the following main tasks

1. Training and testing the model
2. Evaluation of Model
3. Save the model

4. Predicting the output using the model

Application Building

This milestone lets you create Web applications

The Application building involves the following steps :

1. Create an HTML file
2. Build a Python Code
3. Run the app

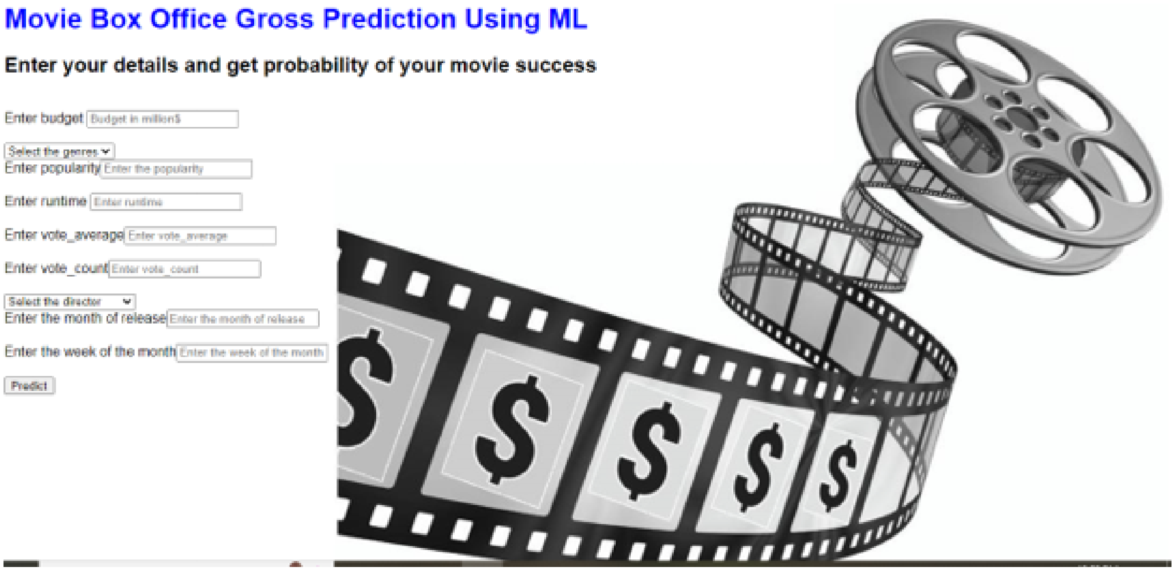
Create An HTML File

We use HTML to create the front end part of the web page.

- Here, we created 2 html pages- Demo2.html, resultnew.html.
- Demo2.html displays the home page which accepts the values from the user
- resultnew.html displays the prediction.

You can refrer to the html pages from the downloaded files.

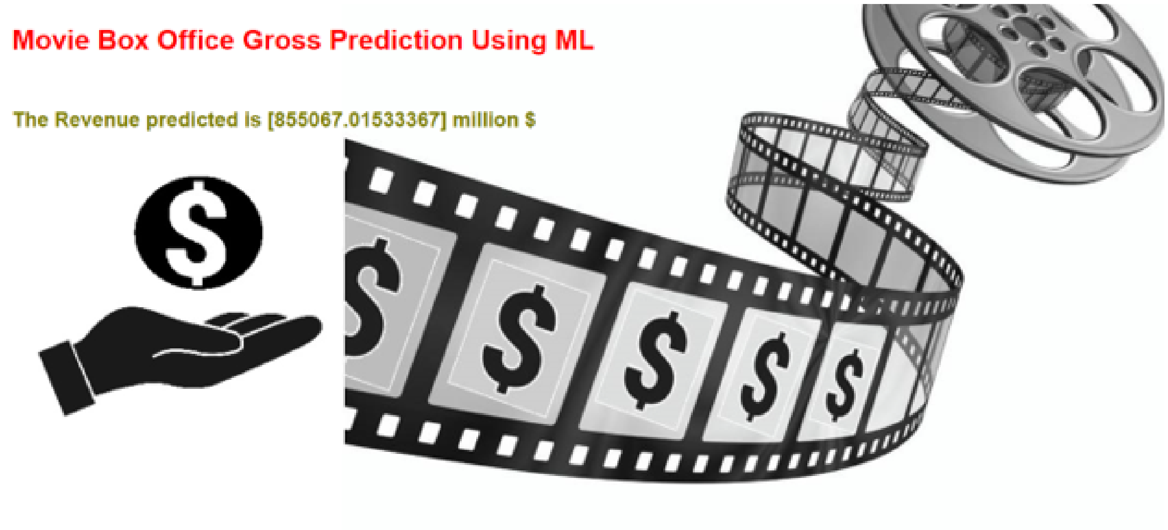
Your Demo2.HTML page looks like below



Movie Box Office Gross Prediction Using ML

Enter your details and get probability of your movie success

THE OUTPUT:-



Movie Box Office Gross Prediction Using ML

The Revenue predicted is [855067.01533367] million $