Movie Box Office Gross Prediction Using IBM Watson Machine Learning

NAZARIA NAZAR

Introduction

Overview – Movies are very common throughout the world. It has been a form of entertainment for decades from the 1930s. They are also a common platform for wealth and fame of many actors. It pays out 2.2 million jobs and contributed 1% of Gross Domestic Product(GDP). For the common audience, movies provide entertainment, information, and message for the society in general. So, in this project the box office gross prediction will be predicted using IBM Watson Studio and Machine Learning algorithms.

Purpose – The prediction of box office gross can help in finding the genre most suitable for audience. The most popular, least popular, trending, successful genres can be found. Factors that directly affect such as actors, directors, budget, genres, popularity or indirectly factors such as vote average, vote count, release month, release date or week affect the box office collections are also analysed. Whether a movie will be successful in the box office also can be predicted.

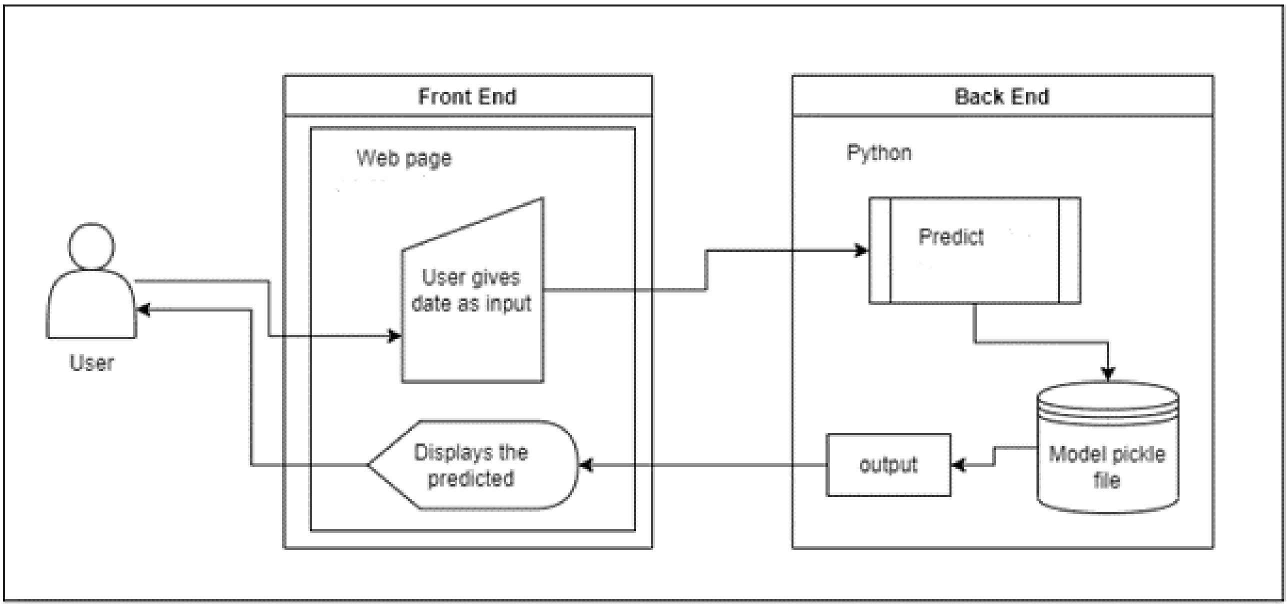
Literature Survey

Existing Problem – The accuracy of the prediction should be high since the revenue of the movie is very important and should be precise. But the prediction of box office depends on many internal and external factors. Factors like actors, directors, plot plays a role but other factors such as pulse of audience, trending genre and other movies releasing during the same time, festive season etc must also be considered. So, the proper factors along with suitable Machine Learning algorithm is mandatory.

Proposed Solution – To find the revenue with high accuracy, in this project important factors are taken from movie and audience point of view. The factors are budget which can lead to better quality of movie, genres which can interest people, popularity impacts more audience, runtime since many prefer short but sweet movies, vote average and vote counts since more votes means many diverse opinions, directors such as James Cameron always provide blockbusters, release month, date and week because of festive seasons and more audience on weekends etc.

Theoretical Analysis

Block Diagram –



Hardware/Software designing –

Hardware Requirements :-

- System with 2- core processor

Software Requirements :-

- Spyder
- Anaconda
- Jupyter Notebook
- IBM- Watson Studio
- Any browser ex: Google Chrome

Experimental Investigations

Analysing the columns of 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.head()
```

budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies
7000000	[{"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": "Ingenious Film Partners", "id": 289...
0000000	[{"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 Disney Pictures", "id": 2}, {"...

Viewing the movie information

```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4803 entries, 0 to 4802
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   budget                4803 non-null   int64
 1   genres                4803 non-null   object
 2   homepage              1712 non-null   object
 3   id                    4803 non-null   int64
 4   keywords              4803 non-null   object
 5   original_language     4803 non-null   object
 6   original_title        4803 non-null   object
 7   overview              4800 non-null   object
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object
 10  production_countries  4803 non-null   object
 11  release_date          4802 non-null   object
 12  revenue               4803 non-null   int64
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object
 15  status                4803 non-null   object
 16  tagline               3959 non-null   object
 17  title_x               4803 non-null   object
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64
 20  title_y               4803 non-null   object
 21  cast                  4803 non-null   object
 22  crew                  4803 non-null   object
dtypes: float64(3), int64(4), object(16)
memory usage: 900.6+ KB
```

Description of the columns such as count, mean, min, max, standard deviation.

```
movies.describe()
```

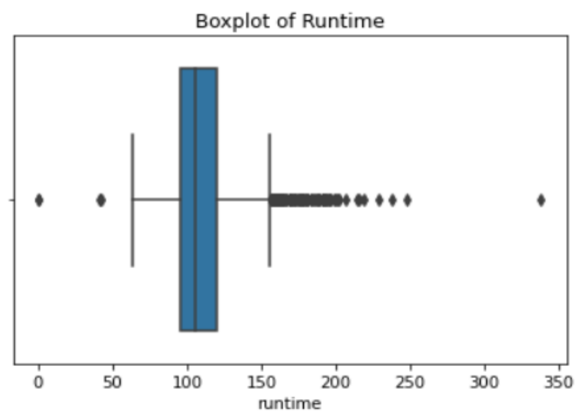
	budget	id	popularity	revenue	runtime	vote_average	vote_count
count	4.803000e+03	4803.000000	4803.000000	4.803000e+03	4801.000000	4803.000000	4803.000000
mean	2.904504e+07	57165.484281	21.492301	8.226064e+07	106.875859	6.092172	690.217989
std	4.072239e+07	88694.614033	31.816650	1.628571e+08	22.611935	1.194612	1234.585891
min	0.000000e+00	5.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000
25%	7.900000e+05	9014.500000	4.668070	0.000000e+00	94.000000	5.600000	54.000000
50%	1.500000e+07	14629.000000	12.921594	1.917000e+07	103.000000	6.200000	235.000000
75%	4.000000e+07	58610.500000	28.313505	9.291719e+07	118.000000	6.800000	737.000000
max	3.800000e+08	459488.000000	875.581305	2.787965e+09	338.000000	10.000000	13752.000000

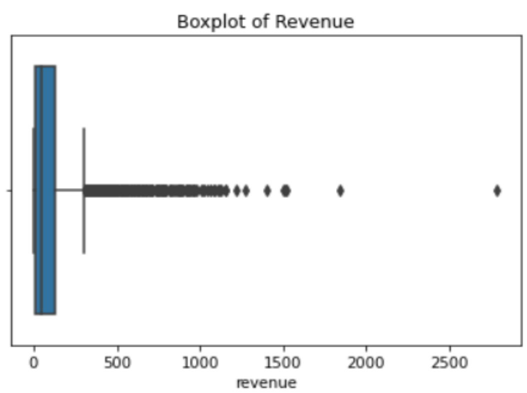
Checking for null values and their sum.

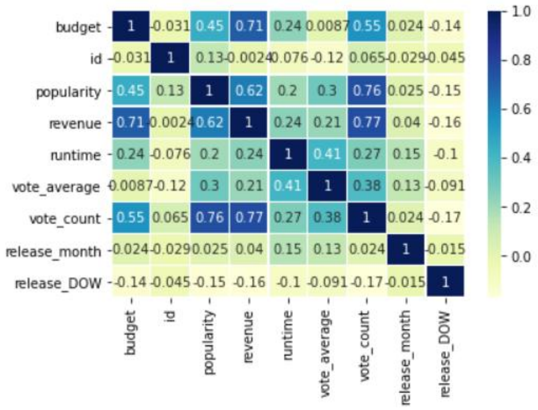
```
movies.isnull().any()
```

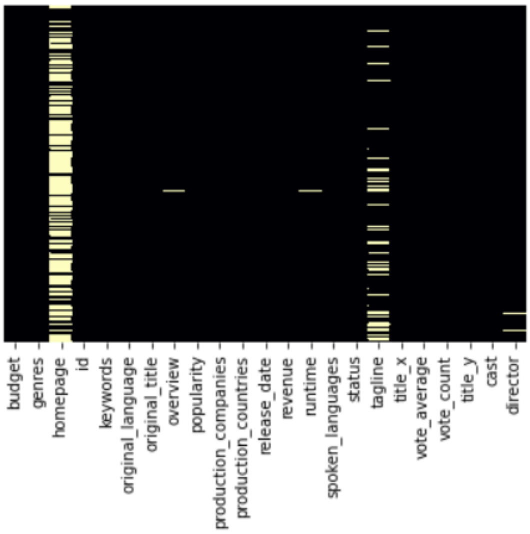
```
budget                   False
genres                   False
homepage                  True
id                       False
keywords                 False
original_language        False
original_title           False
overview                  True
popularity               False
production_companies     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
```

Finding outliers and heatmap of all columns.





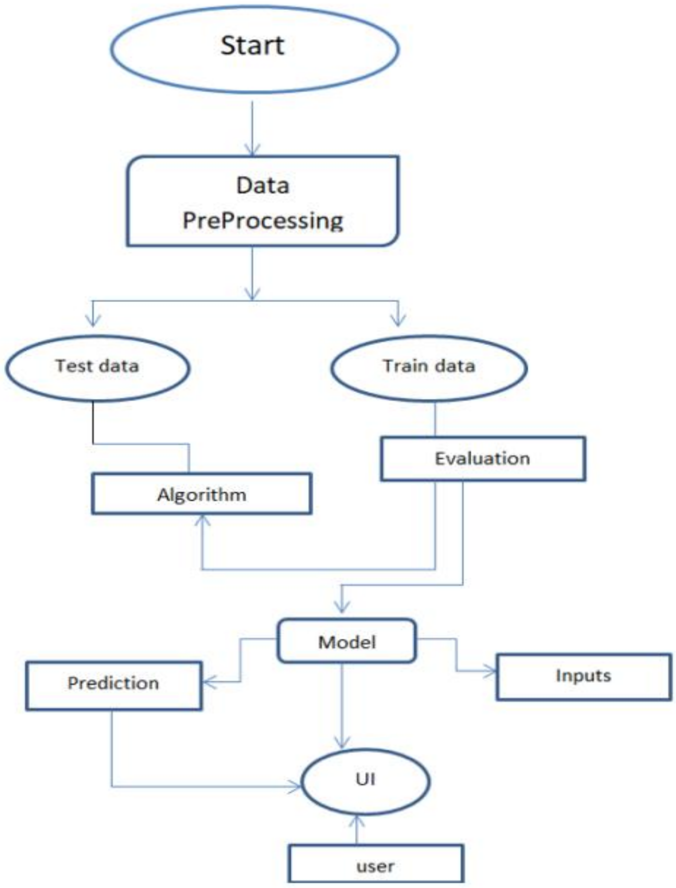




Using word cloud to find the most frequent terms.

Hence, these are the analysis and investigations made during the project implementation.

Flowchart



Result

Final output using Linear Regression algorithm.

```
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)
```

```
from sklearn.linear_model import LinearRegression
mr=LinearRegression()
mr.fit(x_train,y_train)
```

```
LinearRegression()
```

Accuracy using metrics. Accuracy = 71%.

```
from sklearn import metrics
print("MAE:",metrics.mean_absolute_error(y_test,y_pred_mr))
print("RMSE:",np.sqrt(metrics.mean_absolute_error(y_test,y_pred_mr)))
```

```
MAE: 56.52764663167956
RMSE: 7.518486990856575
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_mr)
```

```
0.7174505906933415
```

The inputs are Budget, Genres, Popularity, Runtime, Vote Average, Vote Count, Director, Release Month, and release day of week.

```
input=[[50,8,20.239061,88,5,366,719,7,3]]
input=scalar.transform(input)
prediction = model.predict(input)
prediction
```

```
array([[88.42348926]])
```

Output is box office collection.

Advantages and Disadvantages

Advantages

(i) The predicted output using machine learning algorithm of box office revenue can help in understanding the performance of the movie after its release.

(ii) From the producer's point of view, the revenue can be helpful in knowing if the movie will succeed or fail in the box office performance.

(iii) From the audience's point of view, they can find out if a movie will succeed or fail using the past performance of similar movies and can plan accordingly.

(iv) The reasons for failure of success can be analysed using the predicted output.

Disadvantages

(i) The accuracy of the model is not very high, so predictions can vary sometimes.

(ii) The genre can be inconsistent sometimes since in a genre, even if many movies fail, some movies can become blockbuster.

(iii) Some movies even if the vote average can be less can be blockbuster due to brand values Ex: Avengers, even if the storyline is not good it will be a blockbuster due to its fans.

(iv) The release of other movies or other unavoidable reasons such as COVID pandemic can also play a major role. So, those kinds of factors cannot be measured, and it can impact the box office collections.

Applications

The model can be used in various areas such as :-

(i) Producers or stakeholders can use this model to find if the invested money can be earned back by using the various factors and they can decide whether to invest or not

(ii) Directors, actors, or others associated with the production of movie can analyse if the genre the movie is based on can lead to success, or whether popularity can help in the movie's revenue.

(iii) By audience, to check if the movie will be a blockbuster and plan to go to the movie or not.

(iv) Film critics or reviewers can review the movie using model to check the success ,storyline etc.

Conclusions

Hence, using Multiple Linear Regression machine learning algorithm , the accuracy of the model is 71 %. The model takes into consideration many factors and predicts the revenue. Many advantages and disadvantages are discussed. Areas of application is also briefed. Some findings are genres such as action are generally more successful, the budget is directly proportional to revenue, popularity of famous franchise such as Avengers, Star Wars play a major role, vote average is the most common and easiest way to judge a movie, vote count is high for more watched movies, few directors such as James Cameron, Christopher Nolan are highly successful regardless of other factor. Release month and week, date is also vital in a movie's success. Hence, the reason for the implementation of these factors in model.

Future Scope

Number of movies are increasing year by year, so the future scope in movies is good. The accuracy rate of the can be increased using some other machine learning or deep learning models. Other factors that can play an important role in movies can be analysed and implemented in future models for better predictions. Hence, the best enhancements to be implemented in a model is a better and more accurate model that has an accuracy rate of almost 100% could potentially revolutionize the movie industry.

Bibliography

[1] Liu, T., Ding, X., Chen, Y., Chen, H., & Guo, M. (2016). Predicting movie box-office revenues by exploiting large-scale social media content. Multimedia Tools and Applications, 75(3), 1509-1528.

[2] Wang, Z., Zhang, J., Ji, S., Meng, C., Li, T., & Zheng, Y. (2020). Predicting and ranking box office revenue of movies based on big data. Information Fusion, 60, 25-40.

[3] Zhou, Y., Zhang, L., & Yi, Z. (2019). Predicting movie box-office revenues using deep neural networks. Neural Computing and Applications, 31(6), 1855-1865.

[4] Quader, N., Gani, M. O., Chaki, D., & Ali, M. H. (2017, December). A machine learning approach to predict movie box-office success. In 2017 20th International Conference of Computer and Information Technology (ICCIT) (pp. 1-7). IEEE.

[5] Kim, T., Hong, J., & Kang, P. (2015). Box office forecasting using machine learning algorithms based on SNS data. International Journal of Forecasting, 31(2), 364-390.

[6] Quader, N., Gani, M. O., & Chaki, D. (2017, December). Performance evaluation of seven machine learning classification techniques for movie box office success prediction. In 2017 3rd International Conference on Electrical Information and Communication Technology (EICT) (pp. 1-6). IEEE.

Appendix

IPYNB File Code

```
import numpy as np

import pandas as pd

import seaborn as sns

import json

import matplotlib.pyplot as plt

import warnings

import pickle

from collections import Counter

from sklearn.metrics import r2_score
```

```
from ast import literal_eval
from wordcloud import WordCloud, STOPWORDS
credits = pd.read_csv("C:/Users/Ranjan/Desktop/Extern/tmdb_5000_credits.csv")
movies = pd.read_csv("C:/Users/Ranjan/Desktop/Extern/tmdb_5000_movies.csv")
credits.head()
credits.tail()
movies.head()
movies.tail()
credits.columns
movies.columns
credits.shape
movies.shape
credits.columns = ['id','title','cast','crew']
movies = movies.merge(credits,on='id')
movies.shape
movies.info()
movies.describe()

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)

from ast import literal_eval
features = ['keywords','genres']
for feature in features:
```

```
    movies[feature] = movies[feature].apply(literal_eval)

def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]
        if len(names) > 1:
            names = names[:1]
        return names
    return []

print (type(movies.loc[0, 'genres']))

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

movies['genres']  = movies['genres'] .str.join(', ')
movies['genres']

movies.head()
print("movies:",movies.shape)
movies.corr()
movies.isnull().any()
movies.isnull().sum()
sns.heatmap(movies.isnull(),yticklabels=False,cbar=False,cmap='magma')
movies = movies.dropna(subset = ['director','runtime'])
movies.isnull().sum()
movies["revenue"]=movies["revenue"].floordiv(1000000)
```

```
movies["budget"]=movies["budget"].floordiv(1000000)

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

movies.info()

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


sns.boxplot(x=movies['runtime'])

plt.title('Boxplot of Runtime')

sns.boxplot(x=movies['revenue'])

plt.title('Boxplot of Revenue')

sns.boxplot(x=movies['budget'])

plt.title('Boxplot of Budget')

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

movies['log_revenue'] = np.log1p(movies['revenue'])

movies['log_budget'] = np.log1p(movies['budget'])

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');

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)');

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

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

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis("off")

plt.show()

movies['has_homepage'] = 0

movies.loc[movies['homepage'].isnull() == False, 'has_homepage'] = 1

sns.catplot(x='has_homepage', y='revenue', data=movies);

plt.title('Revenue for movie with and w/o homepage');


sns.jointplot(movies.budget, movies.revenue);

sns.jointplot(movies.popularity, movies.revenue);

sns.jointplot(movies.runtime, movies.revenue);

plt.show()


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

sns.jointplot(movies.release_month, movies.revenue);

plt.xticks(rotation=90)

plt.xlabel('Months')

plt.title('revenue')

movies.info()


movies_box =
movies.drop(['homepage','id','keywords','original_language','original_title','overview','produ
ction_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()
```

```
movies_box.dtypes

movies_box.head()


from sklearn.preprocessing import LabelEncoder

from collections import Counter as c

cat=['director','genres']

for i in movies_box[cat]:

   print("LABEL ENCODING OF:",i)

   LE = LabelEncoder()

   print(c(movies_box[i]))

   movies_box[i] = LE.fit_transform(movies_box[i])

   print(c(movies_box[i]))


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)


movies_box.head()

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_co
unt','director'

                    ,'release_month','release_DOW'])

x

y=movies_box.iloc[:,3]

y=pd.DataFrame(y,columns=['revenue'])
```

```
y
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x=sc.fit_transform(x)
x
pickle.dump(sc,open("scalar_movies.pkl","wb"))
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)
from sklearn.linear_model import LinearRegression
mr=LinearRegression()
mr.fit(x_train,y_train)
x_test
y_test[0:5]
y_pred_mr=mr.predict(x_test)
y_pred_mr[0:5]
y_test
from sklearn import metrics
print("MAE:",metrics.mean_absolute_error(y_test,y_pred_mr))
print("RMSE:",np.sqrt(metrics.mean_absolute_error(y_test,y_pred_mr)))
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_mr)
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_jobs = -1, random_state = 42)
rf.fit(x_train, y_train)
y_pred_mr=mr.predict(x_test)
r2_score(y_test,y_pred_mr)

import pickle
pickle.dump(mr,open("model_movies.pkl","wb"))
```

```
model=pickle.load(open("model_movies.pkl","rb"))
scalar=pickle.load(open("scalar_movies.pkl","rb"))
input=[[50,8,20.239061,88,5,366,719,7,3]]
input=scalar.transform(input)
prediction = model.predict(input)
prediction
mr.score(x_test,y_test)*100
```

Python Code

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
import pandas as pd

app = Flask(__name__,template_folder="template")
model=pickle.load(open("model_movies.pkl","rb"))
scalar=pickle.load(open("scalar_movies.pkl","rb"))

@app.route('/')
def home():
    return render_template('Demo2.html')

@app.route('/y_predict',methods=['POST'])
def y_predict():
    input_features=[float(x) for x in request.form.values() ]
    features_values=[np.array(input_features)]

feature_name=['budget','genres','popularity','runtime','vote_average','vote_count','director
','release_month','release_DOW']
    x_df=pd.DataFrame(features_values,columns=feature_name)
```

```
    x=scalar.transform(x_df)
    prediction=model.predict(x)
    print("Prediction is:",prediction)
    return render_template("resultnew.html",prediction_text=prediction[0])
if __name__=="__main__":
    app.run(debug=False)
```

Demo2.html

```
<html>
<style>
    body {
        background-image: url("../static/css/12.jpg");
        background-repeat: no-repeat;


       background-position: center;
        font-family:sans-serif;
        background-size:cover;
    }
</style>
<body>
<div class="login">
<h1>Movie Box Office Gross Prediction Using ML <span class="label label-
default"></span></h1>
<h2>Enter your details and get probability of your movie success <span class="label label-
default"></span></h2><br>
<style>
h1   {color: blue;}
h2    {color: red;}
</style>
```

```
<form action="{{ url_for('y_predict')}}" method="post">


Enter budget <input type="text" name="budget" placeholder="Budget in million$"
required="required"/><br><br>

<select  id="genres" name="genres" >

   <option>Select the genres</option>

   <option value="6">Drama</option>

   <option value="3">Comedy</option>

<option value="0">Action</option>

   <option value="1">Adventure</option>

<option value="10">Horror</option>

   <option value="4">Crime</option>

<option value="16">Thriller</option>

   <option value="2">Animation</option>

<option value="8">Fantasy</option>

   <option value="14">Science Fiction</option>

<option value="13">Romance</option>

   <option value="7">Family</option>

<option value="12">Mystery</option>

   <option value="5">Documentary</option>

<option value="18">Western</option>

   <option value="17">War</option>

<option value="9">History</option>

   <option value="15">TV Movie</option>

<option value="11">Music</option>


  </select><br>
```

```
Enter  popularity<input type="text" name="popularity" placeholder="Enter the popularity" required="required"/><br><br>

Enter  runtime <input type="text" name="runtime" placeholder="Enter runtime" required="required"/><br><br>

Enter  vote_average<input type="text" name="vote_average" placeholder="Enter vote_average" required="required"/><br><br>

Enter  vote_count<input type="text" name="vote_count" placeholder="Enter vote_count" required="required"/><br><br>


<select  id="director" name="director" >

   <option>Select the director</option>

   <option value="2108">Steven Spielberg</option>

   <option value="2323">Woody Allen</option>

<option value="1431">Martin Scorsese</option>

   <option value="377">Clint Eastwood</option>

<option value="1851">Ridley Scott</option>

   <option value="1894">Robert Rodriguez</option>

<option value="2051">Spike Lee</option>

   <option value="2107">Steven Soderbergh</option>

<option value="1810">Renny Harlin</option>

   <option value="2169">Tim Burton</option>

<option value="1654">Oliver Stone</option>

   <option value="1904">Robert Zemeckis</option>

<option value="1930">Ron Howard</option>

   <option value="1034">Joel Schumacher</option>

<option value="156">Barry Levinson</option>

   <option value="1480">Michael Bay</option>

<option value="2234">Tony Scott</option>

   <option value="245">Brian De Palma</option>

<option value="667">Francis Ford Coppola</option>

   <option value="1256">Kevin Smith</option>
```

```
<option value="1973">Sam Raimi</option>
   <option value="2025">Shawn Levy</option>
<option value="1823">Richard Donner</option>
   <option value="320">Chris Columbus</option>
  </select><br>

Enter the month of release<input type="text" name="release_month" placeholder="Enter the month of release" required="required"/><br><br>
Enter the week of the month<input type="text" name="release_DOW" placeholder="Enter the week of the month" required="required"/><br><br>
 <button type="submit" class="btn btn-default">Predict</button>
</form>
{{prediction_text}}
</div>
</body>
</html>
```

Resultnew.html

```
<html>
<style>
.idiv{
border-radius:10px;

}

   body {
      background-image: url("../static/css/7.jpg");
      background-repeat: no-repeat;

      background-position: center;
      font-family:sans-serif;
```

```
      background-size:cover;
    }

input{
font-size:1.3em;
width:80%;
text-align:center;
}
input placeholder{
text-align:center;
}
button{
outline:0;
border:0;
background-color:darkred;
color:white;
width:100px;
height:40px;
}
button:hover{
background-color:brown;
border:solid 1px black;
}
h1{
color:red;
}
h2{
color:olive;
}
```

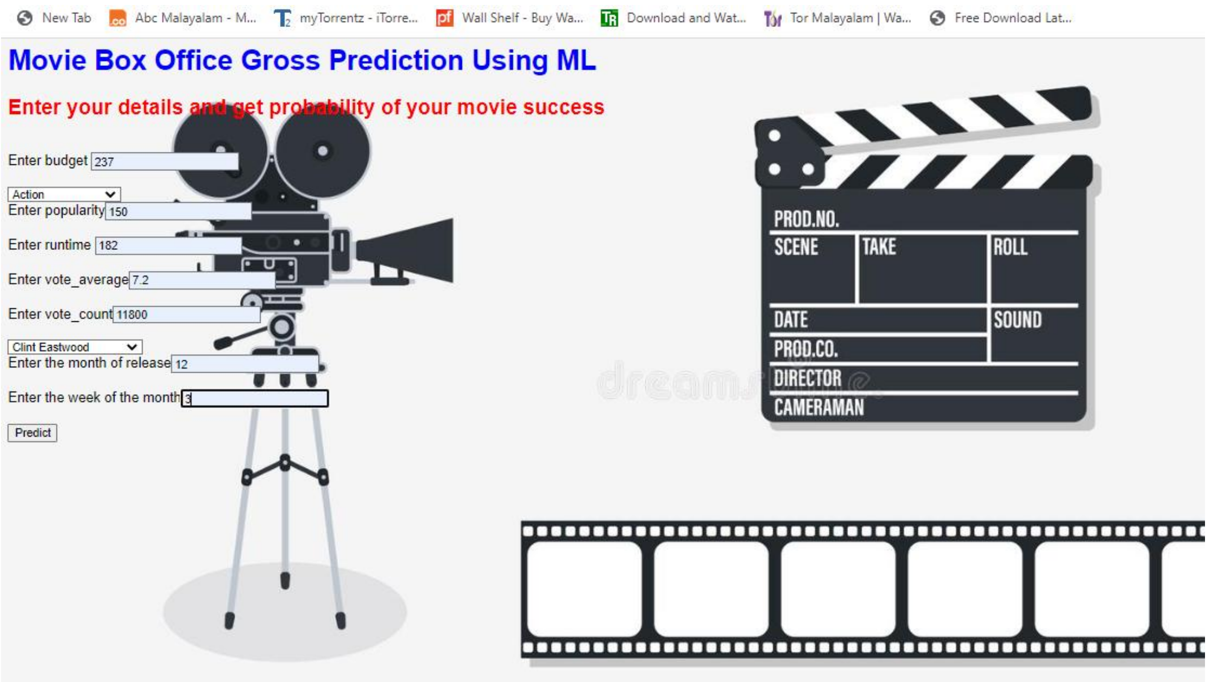
```
</style>

<head>

<title > Movie Box Office Gross Prediction Using ML</title>

</head>

<body>

<div class='idiv'>

<br/>

<h1>Movie Box Office Gross Prediction Using ML</h1>

<br/>

<h2>The Revenue predicted is {{prediction_text}} million $ </h2>

<br/>

<br/>

<br/>

</div>

</body>

</html>
```

IBM Deployment Code

```
import requests


# NOTE: you must manually set API_KEY below using information retrieved from your IBM
Cloud account.

API_KEY = "A4M0hSoy-nfCNTQ7VtiP7MLHcTRJHKlDCMbVjkX3Ygqz"

token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":

 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})

mltoken = token_response.json()["access_token"]


header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}


# NOTE: manually define and pass the array(s) of values to be scored in the next line
```

```
payload_scoring = {"input_data": [{"fields":[["f0","f1","f2","f3","f4","f5","f6","f7","f8"]],
"values": [[  ]] }]}


response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/41f82949-cdb6-4c3f-b453-
97afce0fcdd9/predictions?version=2022-06-01', json=payload_scoring,
 headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
pred = response_scoring.json()
output = pred['predictions'][0]['values'][0][0][0]
print(output)
```

Output Screenshots




Movie Box Office Gross Prediction
The Revenue predicted is [1204.03604194] million $
BOX OFFICE