**Movie Box Office Gross Prediction Using IBM Watson Machine Learning**

**S Ranjan 19BCE1675 Applied Data Science Smartbridge**

**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* –

Diagram, schematic

Description automatically generated

*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.

Text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Viewing the movie information

Table

Description automatically generated

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

Table

Description automatically generated

Checking for null values and their sum.

Table

Description automatically generated

Finding outliers and heatmap of all columns.

Chart, box and whisker chart

Description automatically generatedChart, box and whisker chart

Description automatically generated

Table

Description automatically generated A picture containing text, piano

Description automatically generated

Using word cloud to find the most frequent terms.

Text

Description automatically generated

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

**Flowchart**

Diagram

Description automatically generated

**Result**

Final output using Linear Regression algorithm.

Graphical user interface, text, application

Description automatically generated

Accuracy using metrics. Accuracy = 71%.

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

Output is box office collection.

**Advantages and Disadvantages**

Advantages

1. The predicted output using machine learning algorithm of box office revenue can help in understanding the performance of the movie after its release.
2. 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.
3. 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.
4. The reasons for failure of success can be analysed using the predicted output.

Disadvantages

1. The accuracy of the model is not very high, so predictions can vary sometimes.
2. The genre can be inconsistent sometimes since in a genre, even if many movies fail, some movies can become blockbuster.
3. 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.
4. 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 :-

1. 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
2. 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.
3. By audience, to check if the movie will be a blockbuster and plan to go to the movie or not.
4. 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','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()

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\_count','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

**HTML Code**

from flask import Flask, render\_template, request

import pickle

app = Flask(\_\_name\_\_)

model = pickle.load(open("model\_movies.pkl","rb"))

scalar = pickle.load(open("scalar\_movies.pkl","rb"))

@app.route('/')

def hello():

return render\_template("Demo2.html")

@app.route('/resultnew', methods = ['POST'])

def User():

b = request.form["bg"]

c = request.form["ge"]

d = request.form["pr"]

e = request.form["rt"]

f = request.form["va"]

g = request.form["vc"]

h = request.form["dc"]

i = request.form["rm"]

j = request.form["rd"]

t = [[float(b),float(c),float(d),float(e),float(f),float(g),float(h),float(i),float(j)]]

y = scalar.transform(t)

pred = model.predict(y)

return render\_template("resultnew.html",out="The revenue is"+str(pred))

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug = True)

**Demo2.html**

<html>

<head>

<style>

body {

background-image: url({{ url\_for('static', filename='css/12.jpg') }})

}

.btn {

background-color: #04AA6D;

color: white;

padding: 14px 18px;

border: none;

cursor: pointer;

width: 40%;

opacity: 0.9;

}

.container {

position: absolute;

right: 100;

margin: -100px;

max-width: 500px;

padding: 10px;

background-color: white;

}

</style>

</head>

<body>

<center>

<h1><b><p>Movie Box Office Gross Prediction</p></b></h1>

<form action = "/resultnew" class = "container" method = post>

<h2><b><p>Enter the credentials of the movie</p></b></h2>

<b><p>Budget</p></b>

<p><input type = "name" name = "bg"></p>

<b><p>Genres</p></b>

<p ><input type = "name" name = "ge"></p>

<b><p>Popularity</p></b>

<p><input type = "name" name = "pr"></p>

<b><p>Runtime</p></b>

<p><input type = "name" name = "rt"></p>

<b><p>Vote Average</p></b>

<p><input type = "name" name = "va"></p>

<b><p>Vote Count</p></b>

<p><input type = "name" name = "vc"></p>

<b><p>Director</p></b>

<p><input type = "name" name = "dc"></p>

<b><p>Release Month</p></b>

<p><input type = "name" name = "rm"></p>

<b><p>Release Day\_of\_Week</p></b>

<p><input type = "name" name = "rd"></p>

<p><input type = "submit" name = "Submit", class = "btn"></p>

</form>

</center>

</body>

</html>

**Resultnew.html**

<html>

<head>

<style>

body {

background-image: url({{ url\_for('static', filename='css/7.jpg') }})

}

.container {

margin: 0px;

max-width: 500px;

padding: 10px;

background-color: white;

}

</style>

</head>

<body>

<center>

<form action = "/resultnew" class = "container">

<h1 style="color:Black;"><center>Movies Box Office Gross Prediction</center></h1>

<b><h1 style="color:Black;"><center>{{ out }}</h1></center></b>

</form>

</center>

</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)

Diagram

Description automatically generated

Text, letter

Description automatically generated

YouTube Link : <https://youtu.be/lTlZP_7KbW4>