MOVIES BOX OFFICE GROSS PREDICTION

1. INTRODUCTION

1.1 Overview

In a movie industry and its related stakeholders use a forecasting method for the prediction of revenue that a new movie can generate based on a few given input attributes like budget, runtime, released year, popularity, and so on. It 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 Online rating, Director, Budget, Pre Release business, Genre, etc.. .

1.2 Purpose

Predicting society's reaction to a new product in the sense of popularity and adoption rate has become an emerging field of data analysis, and such kind of analysis can help the movie industry to take appropriate decisions. This study marks as a decision support system for the movie investment sector using machine learning techniques.

2. LITURATURE SURVAY

2.1 Existing problem

According to others using this dataset, some of the values for the movies are incorrect, meaning that some of our predictions will be off by a large amount, but we shouldn’t always trust the listed value.

It is a very time-consuming task to predict a new release movie revenue by actual testing. For one car it could take up to week of testing and paper work to do it properly.

Using a machine learning model, we can do the same in seconds.

2.1Proposed solution

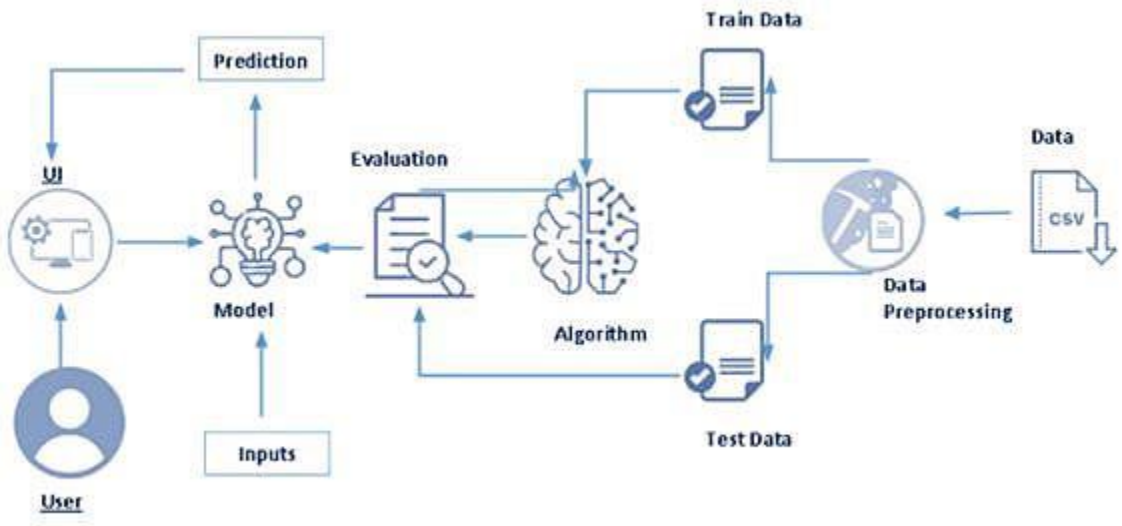
Using advanced machine learning model which is trained using the verified dataset of which movie have more success and its various attributer the model can be trained to predict revenue of a new movie provided necessary values given.

The model can predict the performance with an accuracy of 75% given the fact that the result can be seen in seconds the model is reliable.

Anyone with prior knowledge of using a web browser can operate the application easily. Generally, a model is only as good as the data passed into it, and the data preprocessing we do ensure that the model has as accurate a dataset as possible

3. Theoratical Analysis

3.1 Block diagram



3.2 Hardware and Software Designing:

- IBM Watson Studio

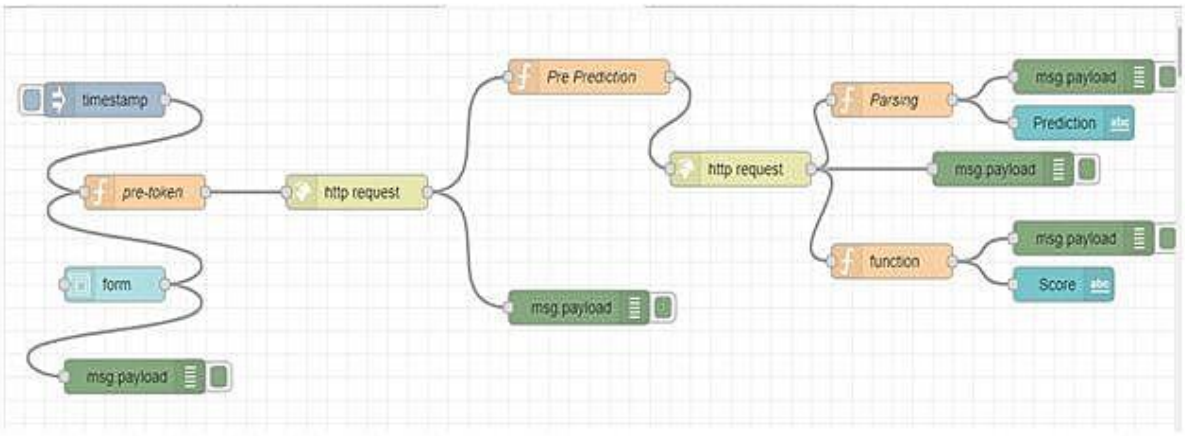
Watson Studio accelerates the machine and deep learning workflows required to infuse AI into your business to drive innovation. It provides a suite of tools for data scientists, application developers and subject matter experts, allowing them to collaboratively connect to data, wrangle that data and use it to build, train and deploy models at scale. Successful AI projects require a combination of algorithms + data + team, and a very powerful compute infrastructure.

- IBM Watson Machine Learning
- IBM Cloud Object Storage

4. Experimental Investigation

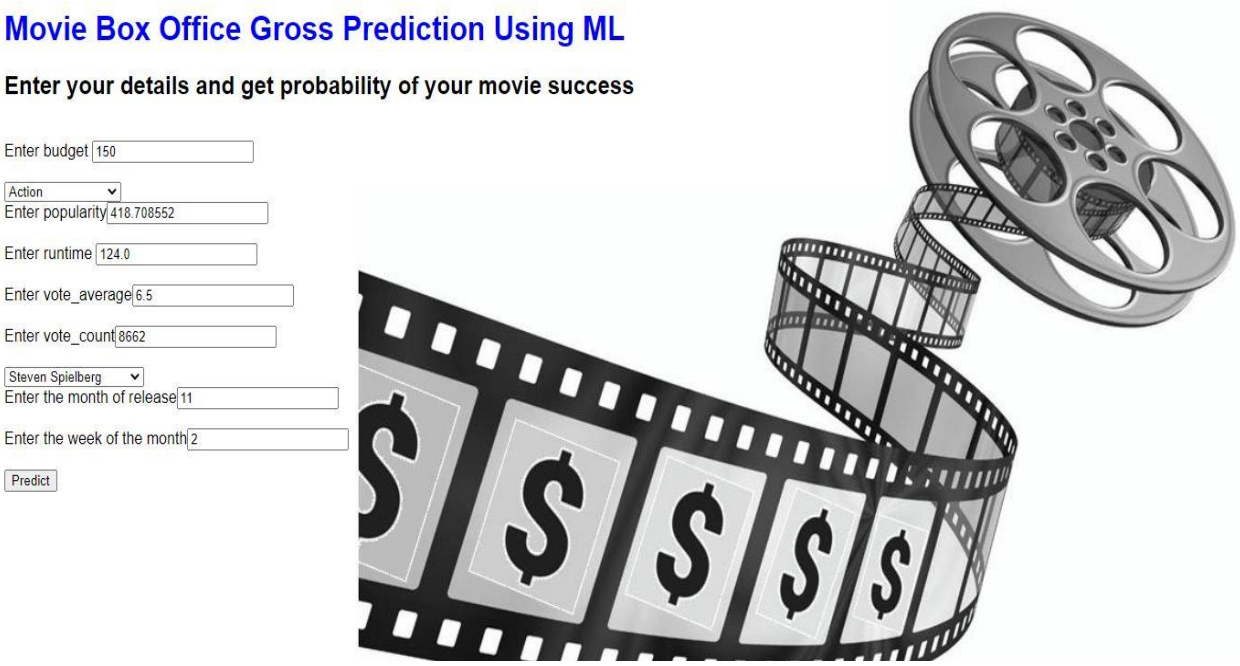
- In this section, we will be creating and training our model for predicting the success rate of a new released movie. Since there are multiple algorithms, we can use to build our model, we will compare the accuracy scores after testing and pick the most accurate algorithm.
- From this list, we are using Random Forest and Linear Regression to perform our predictions. We then see which algorithm produces the highest accuracy and select it as our algorithm of choice for future use.
- On the results of the following algorithms, we have done the conclusion the Linear Regression model is the most accurate out of all the models which we have tested.

5.FLOWCHART



6. RESULT

The final result of the project Movies Box office gross prediction . The output would be in this format:



7.ADVANTAGES

The main advantage the proposed model is it that it can predict whether certain attributes like budget, genres, director, release_month, etc. have any effect on the movies gross which movie might get more profit.

Therefore, the movie investor can avoid the risk of losing their investment.

DISADVANTAGES

- Need more datasets, to increase the accuracy of the algorithms.
- The accuracy of the application depends on the dataset used to train the model
- The proposed application is Web-based, hence cannot be used in Mobile devices.
- The result of the application depends upon the accuracy of the algorithms

8. APPLICATION

Film production Companies

It includes filmmakers, distribution aspects, as well as film study

Consumer

The audience, as well as user of the on Demand platform.

9. CONCLUSION

The film industry is an unpredictable business either the producer gain higher profits or they might get into huge loss, so it is difficult for a human to predict that the movie box-office prior to the release. Although it is very important for production studios to be able to predict the movie box office revenues before they are released, the prediction of box office revenue is still classified as an art rather than a science because most experts predict revenue based on their own rules of thumb, hunches, and their experience. This project helps the production studios to predict box office revenues that can be used to decide for planning the production and the movie distribution stages.

10. FUTURE SCOPE

The Global box office revenue had hit a record of $42.5 billion in 2019. According to this, Movie Industries has become one of the major industries in the world which includes a lot of Budget, Crew, and Cast. So the prediction of movie revenue makes a great deal. Predicted revenues can be used to decide on planning the production and the distribution stages. For an instance, the projected revenue will be helpful to decide the remuneration of actors, members, and also the costs to sell the copyrights. Movie success or failure can depend mainly on the Stars that are acting in the film, release day, Budget, and also the Genre of the film. For an instance, the Adventure Genre films usually get more Gross revenue compared to other Genre. These all the factors have an

impact on the Movie's revenue. We can also redesign the web application to follows the latest trends and also support different languages in the future.

11. BIBLIOGRAPHY

Data Science for Beginners

www.wikipedia.org

www.google.com

www.github.org

11 .1 Appendix

Movie Box Office Gross prediction.ipynb

```
#!/usr/bin/env python
# coding: utf-8
# In[8]:


import pandas as pd
import numpy as np
import seaborn as sns
import json
import matplotlib.pyplot as plt
import pickle
from wordcloud import WordCloud
from ast import literal_eval
import os
credits=pd.read_csv("tmdb_5000_credits.csv")
```

```
movies_df=pd.read_csv("tmdb_5000_movies.csv")


credits.head()
movies_df.head()
print("credits:",credits.columns)
print("movies_df:",movies_df.columns)
print("credits:",credits.shape)
print("movies_df:",movies_df.shape)


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


#Convert JSON into string


# changing the crew column from json to string
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)
# Returns the top 1 element or entire list; whichever is more.
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 []
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']

#corr() is to find the relationship between the columns
movies.corr()

#Checking for null value
```

```
movies.isnull().any()
movies.isnull().sum()
sns.heatmap(movies.isnull(),yticklabels=False,cbar=False,cmap='viridis')

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



#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 o,let us remove the rows with budget as 0
movies = movies[movies['budget'] != 0]
movies.info()

### Data visualization

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')
```

```
#removing outliers

bq_low = movies['budget'].quantile(0.01)

bq_hi  = movies['budget'].quantile(0.99)

rq_low = movies['runtime'].quantile(0.01)

rq_hi  = movies['runtime'].quantile(0.99)

movies = movies[(movies['budget'] < bq_hi) & (movies['budget'] > bq_low) &
(movies['runtime'] < rq_hi) & (movies['runtime'] > rq_low)]

movies.shape

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

plt.title('Boxplot of Runtime(Outliers Removed)')

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

plt.title('Boxplot of Budget(Outliers Removed)')

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



#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'])


#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');
```

Relationship between Film Revenue and Budget.

```
#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)');
```

```
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()
```

Relationship between the home page and Revenue.

#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');


#### Relationship between release_month and revenue


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

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

plt.xticks(rotation=90)

plt.xlabel('Months')

plt.title('revenue')


    ### Label Encoding


    # Label encoding features to change categorical variables into numerical one

    from sklearn.preprocessing import LabelEncoder

    from collections import Counter as c

    cat=['director','genres']

    for i in movies_box[cat]:#looping through all the categorical columns

        print("LABEL ENCODING OF:",i)

        LE = LabelEncoder()#creating an object of LabelEncoder

        print(c(movies_box[i])) #getting the classes values before transformation

        movies_box[i] = LE.fit_transform(movies_box[i]) # trannsforming our text classes to numerical values
```

```
    print(c(movies_box[i])) #getting the classes values after transformation
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)

### Splitting the Dataset into Dependent and Independent variable
#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
#Dependent Variables
y=movies_box.iloc[:,3]
y=pd.DataFrame(y,columns=['revenue'])
y

### Feature Scaling

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

```
x=sc.fit_transform(x)
x

## splitting the data to train and test

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

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]
3.76955224*100000000
y_test

### Model_Evaluation

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)

### Saving the model

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)
```

App_IBM.py

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

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

API_KEY = "MHLG-wkplyQJ35N_1neKLLoGDxTrf2xMkncI1BSanE-F"

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": [array_of_input_fields], "values": [array_of_values_to_be_scored, another_array_of_values_to_be_scored]}]}


response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/510ef900-f7ba-43c7-bc9d-6373b31c786a/predictions?version=2022-04-15', json=payload_scoring,

 headers={'Authorization': 'Bearer ' + mltoken})

print("Scoring response")

print(response_scoring.json())


app = Flask(__name__) #initialising the flask app

filepath="model_movies.pkl"

model=pickle.load(open(filepath,'rb'))#loading the saved model

scalar=pickle.load(open("scalar_movies.pkl","rb"))#loading the saved scalar file


@app.route('/')
```

```
def home():
    return render_template('Demo2.html')


@app.route('/y_predict',methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML
    '''
    input_feature=[float(x) for x in request.form.values() ]
    features_values=[np.array(input_feature)]
    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)
     # predictions using the loaded model file
    prediction=model.predict(x)
    print("Prediction is:",prediction)
    return render_template("resultnew.html",prediction_text=prediction[0])
if __name__ == "__main__":
    app.run(debug=True)
```