

# **MOVIE BOX OFFICE GROSS PREDICTION**

**Submitted To :**  
SmartBridge  
Applied Data Science

**Submitted By:**

MURRA ANIL KUMAR REDDY	-20BEC0299
ADDANKI KEERTHAN	-20BEC0620
C C VISHNU VARDHAN REDDY	-20BCT0273
KORLAPATI VAMSI SAI	-20BCE2556

# **1. INTRODUCTION**

## **1.1 Overview**

This project is to develop a machine learning-based solution using IBM Watson to predict the box office gross of movies. By analyzing various factors such as genre, cast, release date, and marketing budget, the system aims to provide insights into the potential success of a movie at the box office. This prediction can be valuable for movie studios, distributors, and investors in making informed decisions regarding the marketing, distribution, and investment in movies.

## **1.2 Purpose**

The purpose of this project is to leverage the capabilities of IBM Watson machine learning to build a predictive model that can estimate the box office gross of movies. This project aims to Enable movie studios and distributors to assess the potential success of their movies before release. Assist investors in making informed decisions about investing in movie projects. Optimize marketing strategies by identifying influential factors that contribute to box office success. Provide a reliable tool for analyzing the financial viability of movies.

## **2. LITERATURE SURVEY**

### **2.1 Existing Problem**

The problem of predicting movie box office gross is a well-studied area in the field of data science and machine learning. Several existing approaches have been developed to tackle this problem. Some common methods include regression models, ensemble techniques, and deep learning algorithms. These approaches utilize various features such as movie genre, production budget, release date, marketing expenditure, and historical box office performance.

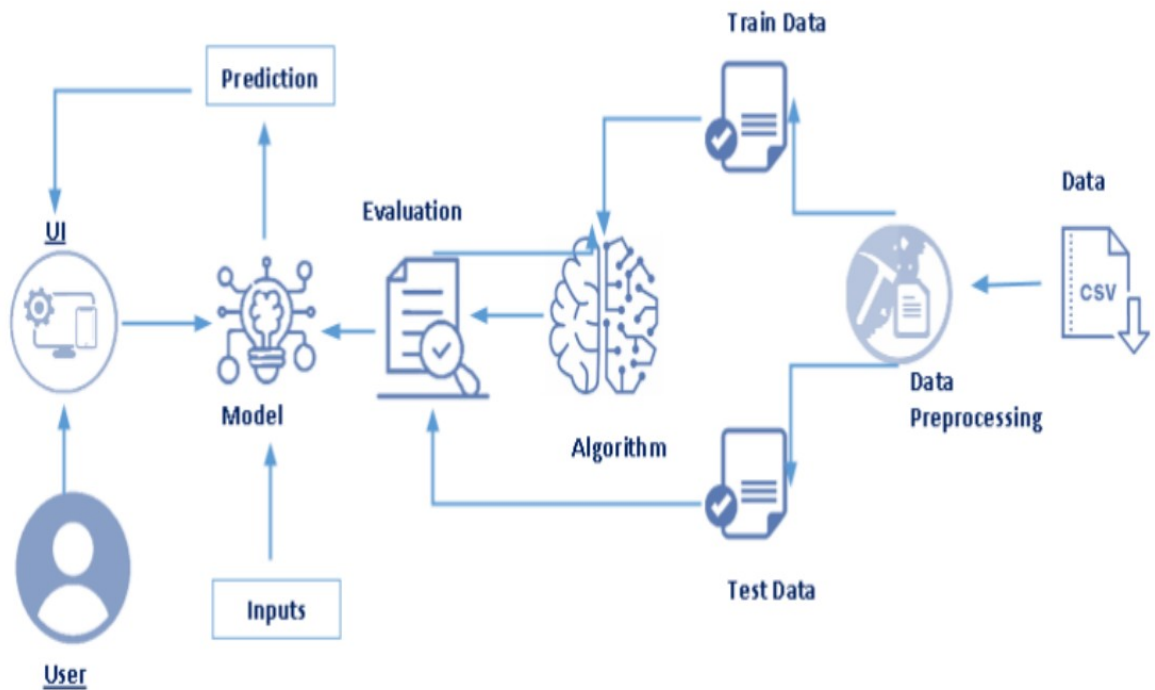
### **2.2 Proposed Solution**

In this project, we propose using IBM Watson machine learning capabilities to develop a predictive model for movie box office gross. The solution involves gathering relevant movie data, preprocessing and feature engineering, model training using machine learning algorithms, and evaluation of the model's performance. By utilizing IBM Watson's powerful machine learning tools and libraries, we aim to create an accurate and robust prediction system.

### 3. THEORETICAL ANALYSIS

#### 3.1 Block Diagram

The following block diagram provides an overview of the project:



## 3.2 Hardware / Software Designing

### Hardware Requirements:

- Computer system with sufficient processing power and memory
- Internet connection for accessing IBM Watson services

### Software Requirements:

- IBM Watson Machine Learning tools and libraries
- Python programming language
- Data preprocessing and analysis libraries (e.g., pandas, NumPy)
- Machine learning libraries (e.g., scikit-learn, TensorFlow)
- Integrated Development Environment (IDE) for Python development (e.g., Jupyter Notebook, PyCharm)

## 4. EXPERIMENTAL INVESTIGATIONS

During the course of this project, various investigations and analyses were conducted. These include:

**Data collection:** Gathering relevant movie data from reliable sources, such as IMDb, Box Office Mojo, or The Movie Database (TMDb).

**Data preprocessing:** Cleaning the collected data, handling missing values, encoding categorical variables, and normalizing numerical features.

**Feature engineering:** Extracting and creating relevant features from the movie dataset, such as genre indicators, release month, and production budget per genre.

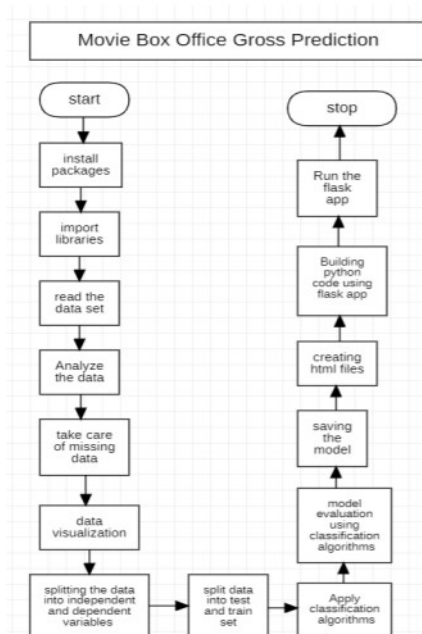
**Model training and selection:** Employing different machine learning algorithms and techniques (e.g., linear regression, random forest, gradient boosting) to train models on the preprocessed dataset. Evaluating and selecting the best-performing model based on appropriate metrics (e.g., mean squared error, R-squared).

**Hyperparameter tuning:** Optimizing the selected model by tuning hyperparameters using techniques like grid search or random search.

**Model evaluation:** Assessing the performance of the final model using various evaluation metrics and techniques, such as cross-validation, hold-out validation, or time-based validation.

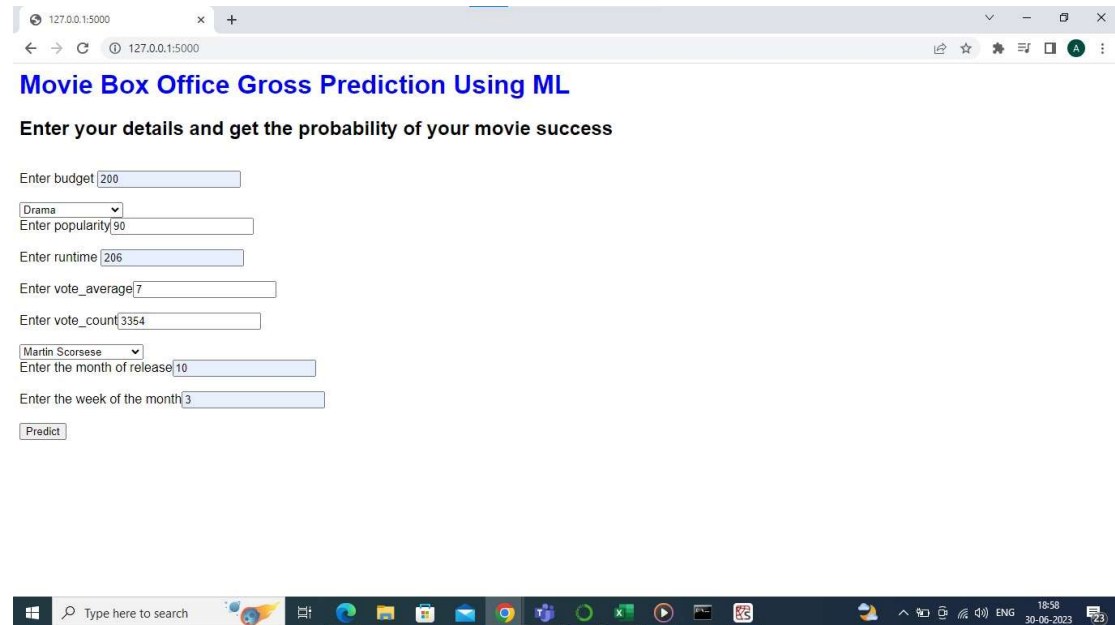
## 5. FLOWCHART

The flowchart below illustrates the control flow of the solution:



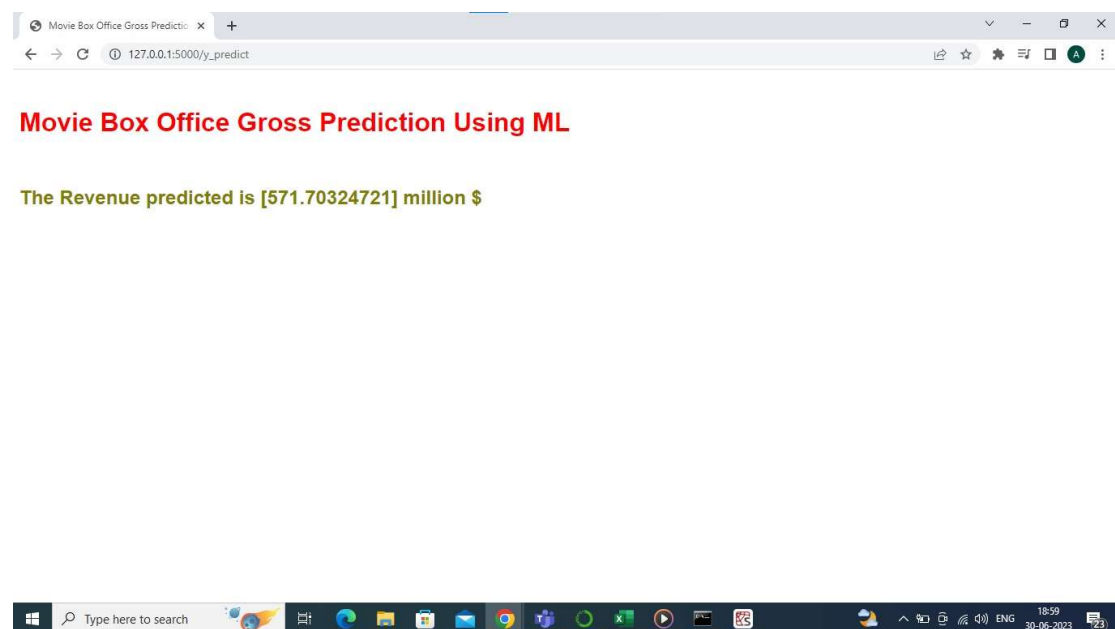
## 6. RESULT

The final output of the project is the prediction of the box office gross for a given movie.



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000". The page title is "Movie Box Office Gross Prediction Using ML". Below the title, there is a heading "Enter your details and get the probability of your movie success". The form contains several input fields and a "Predict" button. The inputs are: "Enter budget" (200), "Drama" (selected from a dropdown), "Enter popularity" (90), "Enter runtime" (206), "Enter vote\_average" (7), "Enter vote\_count" (3354), "Martin Scorsese" (selected from a dropdown), "Enter the month of release" (10), and "Enter the week of the month" (3). The "Predict" button is located at the bottom of the form.

## Predicted gross:



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/y\_predict". The page title is "Movie Box Office Gross Prediction Using ML". Below the title, there is a heading "The Revenue predicted is [571.70324721] million \$".

## **7. ADVANTAGES & DISADVANTAGES**

### **Advantages of the proposed solution:**

- Provides movie studios, distributors, and investors with a predictive tool to estimate box office gross.
- Enables data-driven decision-making in movie marketing, distribution, and investment.
- Utilizes IBM Watson machine learning capabilities, ensuring access to powerful tools and libraries.
- Can leverage historical data to identify patterns and trends that influence movie success.
- Offers potential for improving the accuracy of box office predictions compared to traditional methods.

### **Disadvantages and limitations:**

- Reliance on historical data may not fully account for unforeseen events or changes in consumer preferences.
- Accuracy of predictions can be affected by factors not included in the model, such as critical reception, competition, or external market conditions.
- Data availability and quality may vary, potentially affecting the model's performance.
- The model may require periodic retraining to account for evolving trends and preferences in the movie industry.



## **8. APPLICATIONS**

The proposed solution can find applications in various areas within the movie industry, including:

**Movie studios:** Assessing the potential success of their upcoming movies and optimizing marketing strategies.

**Movie distributors:** Identifying movies with high box office potential for distribution and targeting specific market.

**Investors:** Making informed decisions about investing in movie projects based on predicted box office performance.

**Market analysts:** Analyzing historical movie data to gain insights into box office trends and market dynamics.

**Film festivals and award shows:** Predicting the box office potential of films showcased or recognized at events.

## **9. CONCLUSION**

In conclusion, we developed a movie box office gross prediction system using IBM Watson machine learning. By analyzing various movie features and leveraging historical data, the system provides valuable insights into the potential success of movies at the box office. The project involved data collection, preprocessing, feature engineering, model training, evaluation, and prediction. While the solution offers advantages in terms of data-driven decision-making, it also has limitations due to the complexity of predicting movie success accurately. Nonetheless, the project demonstrates the potential of machine learning in assisting the movie industry in making informed decisions.

## **10. FUTURE SCOPE**

There are several avenues for future enhancements and expansions of this project, including:

- Incorporating sentiment analysis of movie reviews and social media data to capture audience reactions and trends.
- Integrating real-time data sources to adapt predictions to current market conditions and emerging trends.
- Exploring deep learning models, such as neural networks, to capture complex relationships between movie features and box office gross.
- Developing a user-friendly interface or web application to make the prediction system accessible to a wider audience.
- Collaborating with industry stakeholders to gather proprietary data and refine the model's accuracy and applicability.

By continuing to refine and expand the project, it has the potential to become a valuable tool for decision-making and analysis in the movie industry.

## **11.BIBILOGRAPHY**

[1] Kim, Taegu, Jungsik Hong, and Pilsung Kang. "Box office forecasting using machine learning algorithms based on SNS data." *International Journal of Forecasting* 31.2 (2015): 364-390.

[2] N. Quader, M. O. Gani, D. Chaki and M. H. Ali, "A machine learning approach to predict movie box-office success," 2017 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 2017, pp. 1-7, doi: 10.1109/ICCITECHN.2017.8281839.

[3] Sharda, R. and Delen, D. Predicting box-office success of motion pictures with neural networks. In *Expert Systems with Applications*, 2006.

# APPENDIX

## A. SOURCE CODE:

```
# Importing Libraries
import pandas as pd #data manipulation
import numpy as np #Numerical Analysis
import seaborn as sns #data visualization
import json #for reading json object
import matplotlib.pyplot as plt #data visualization
import pickle # For saving the model file
from wordcloud import WordCloud #to create word clouds
from ast import literal_eval #to evaluate the string as python expression
### Loading the data sets
credits=pd.read_csv("tmdb_5000_credits.csv")
movies_df=pd.read_csv("tmdb_5000_movies.csv")
### Exploring the data
credits.head()
credits.shape
movies_df.head()
movies_df.shape
print("credits:",credits.columns)
print("movies:",movies_df.columns)
### Merging the data sets
credits_column_renamed=credits.rename(index=str,columns={"movie_id":"id"})
movies=movies_df.merge(credits_column_renamed,on="id")
movies.head()
movies.shape
movies.info()
movies.describe()
### converting json to strings
# 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)
movies.head()
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) > 3:
            names = names[:3]
        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']
movies.head()
#corr() is to find the relationship between the columns
movies.corr()
#finding out the null values and dropping them
movies.isnull().any()
movies.isnull().sum()
sns.heatmap(movies.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```

#Dropping the null values
movies = movies.dropna(subset = ['runtime'])
movies.isnull().sum()
movies.head(5)
#Divide the revenue and budget columns by 1000000 to convert $ to million $
movies['revenue'] = movies['revenue'].floordiv(1000000)
movies['budget'] = movies['budget'].floordiv(1000000)
movies.head(5)
#As there cannot be any movie with budget as 0, let us remove the rows with budget as 0
movies = movies[movies['budget'] != 0]
movies.info()
#Let us create three new columns and extract date, month and Day of the week from the release date
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 visualisation
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);
#creating log transformation for revenue
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'])
#comparing distribution of revenue and log revenue 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');
#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)})
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
#let's create column called has_homepage and pass two values 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 a 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');
sns.jointplot(data=movies, x='budget', y='revenue');
sns.jointplot(data=movies, x='popularity', y='revenue');
sns.jointplot(data=movies, x='runtime', y='revenue');
plt.show()
plt.figure(figsize=(15,8))
sns.jointplot(data=movies, x='release_month', y='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',
                           'log_revenue', 'log_budget', 'has_homepage'], axis = 1)
movies_box.isnull().sum()
movies_box.dtypes
movies_box.head()
# 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)
movies_box.head()
#testing and training
#Independent Variables
x=movies_box.iloc[:,[0,1,2,4,5,6,7,8,9,10]]
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
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) #accuracy
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)
#outcomes
prediction
mr.score(x_test,y_test)

```

## FLASK APP CODE:

```

import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
import pandas as pd
app = Flask(__name__) # initializing the Flask app
filepath = "model_movies.pkl"
model = pickle.load(open("C:\\Users\\SREE\\Documents\\Python Scripts\\model_movies.pkl", 'rb')) # loading the
saved model

```

```

scalar = pickle.load(open("C:\\Users\\SREE\\Documents\\Python Scripts\\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', '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=False)

```

## HTML files:

### Demo2.html:

```

<html>
<style>
    body {
        background-image: url("cinema_strip_movie_film.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 the probability of your movie success <span class="label label-
default"></span></h2><br>
        <style>
            h1 { color: blue; }
            p { 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>
            </select>

```

```

        <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("cinema_strip_movie_film.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>
```