MOVIE BOX OFFICE GROSS PREDICTION

1. INTRODUCTION

1.1 overview

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.

Can film studios 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.

This study marks as a decision support system for the movie investment sector using machine learning techniques. This project 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

The film industry has grown immensely over the past few decades generating billions of dollars of revenue for the stakeholders . Now people can watch movies online and offline on a variety of mobile devices during leisure or travel through Netflix, Youtube and downloads . A prediction system to assess the box office success of new movies can help the movie producers and directors make informed decisions when making the movie in order to increase the chance of profitability and box office gross success. New social media tools are constantly appearing which are enabling people to gather information on films and post comments about movies. These comments can influence the initial prediction about the box office gross success of a movie which some of the existing research do not take into account. Critic reviews often come out a few days before the film is released and may, therefore, help in prediction and at the same time influence the box office revenue.

2.LITERATURE SURVEY

2.1 Existing Problem (OR) Problem Statement

Given a two datasets containing various attributes, use the features available in the dataset and define a supervised classification algorithm which can identify whether the movie gross getting correct predicted values or not. This data set contains many movie gross records. The data set records was collected all over the world

2.2 Proposed Solution

This is a classic example of supervised learning. We have been provided with a fixed number of features for each data point, and our aim will be to train a variety of Supervised Learning algorithms on this data, so that when a new data point arises, our best performing classifier can be used to categorize the data point as a positive example or negative. Exact details of the number and types of algorithms used for training is included in the 'Algorithms and Techniques' sub-section of the 'Analysis' part.

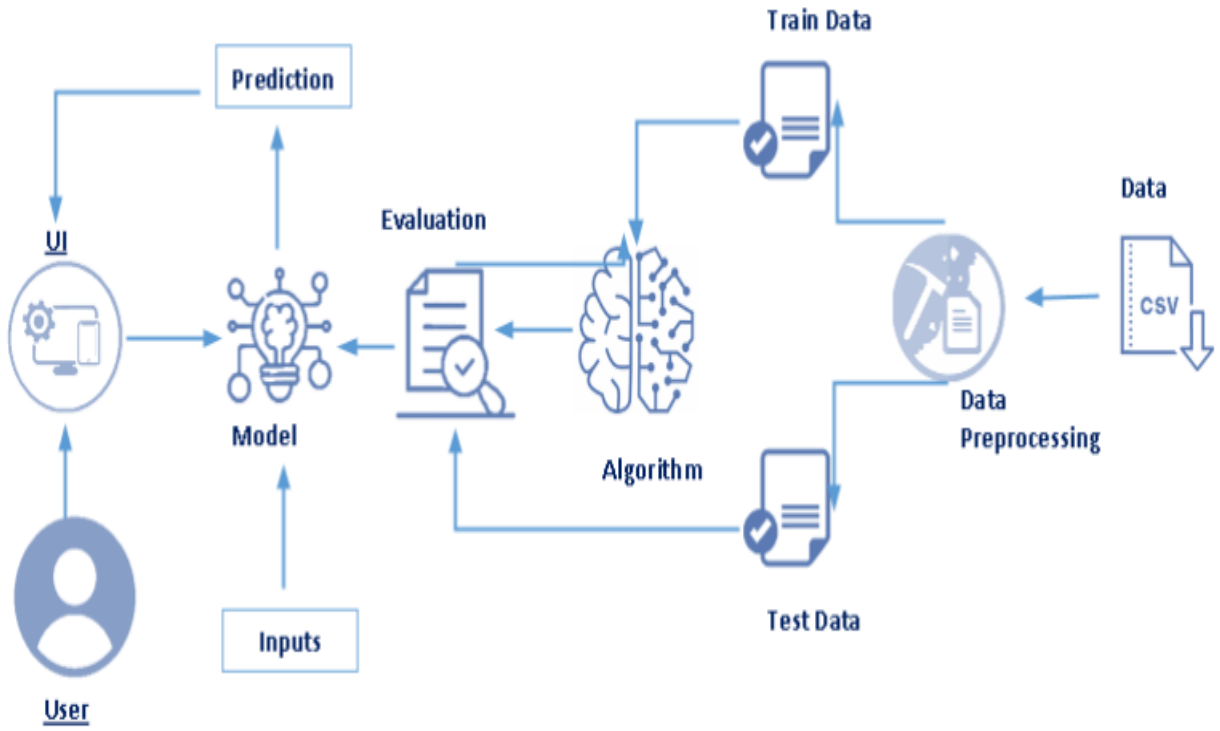
This project focuses on the related works of various movies to calculate box office gross prediction such that algorithms were implemented using Jupyter that is a machine learning software written in Python. Various attributes that are essential in the prediction of movie gross were examined and the dataset of movies were also evaluated. This project compares various classification algorithms such as Random Forest, Support Vector Machine and KNN classification Algorithm with an aim to identify the best technique.

Based on this study, Random Forest with the highest accuracy outperformed the other algorithms and can be further utilised in the prediction of movie box office gross recommended to the user.

Later by using Flask app create html files and create an user interface to display the movie box office gross prediction values.

3.THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware / Software designing

The following is the Hardware required to complete this project:

- Internet connection to download and activate
- Administration access to install and run Anaconda Navigator
- Minimum 10GB free disk space
- Windows 8.1 or 10 (64-bit or 32-bit version) OR Cloud: Get started free, *Cloud account required.

Minimum System Requirements To run Office Excel 2013, your computer needs to meet the following minimum hardware requirements:

- 500 megahertz (MHz)
- 256 megabytes (MB) RAM
- 1.5 gigabytes (GB) available space
- 1024x768 or higher resolution monitor

The following are the software s required for the project:

- Jupyter Notebook
- Spyder
- Microsoft Excel 2013

4.EXPERIMENTAL INVESTIGATIONS

Coming to analysis or investigations three supervised learning approaches are selected for this problem. Movies is taken that all these approaches are fundamentally different from each other, so that we can cover as wide an umbrella as possible in term of possible approaches.
For each algorithm, we will try out different values of a few hyper parameters to arrive at the best possible classifier. This will be carried out with the help of grid search cross validation technique.

There are several Machine learning algorithms to be used depending on the data you are going to process such images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have may be classification algorithms and Regression algorithms.

(1) Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. Support Vectors are simply the coordinates of individual observation. The goal of a support vector machine is not only to draw hyperplanes and divide data points, but to draw the hyperplane the separates data points with the largest margin, or with the most space between the dividing line and any given data point.

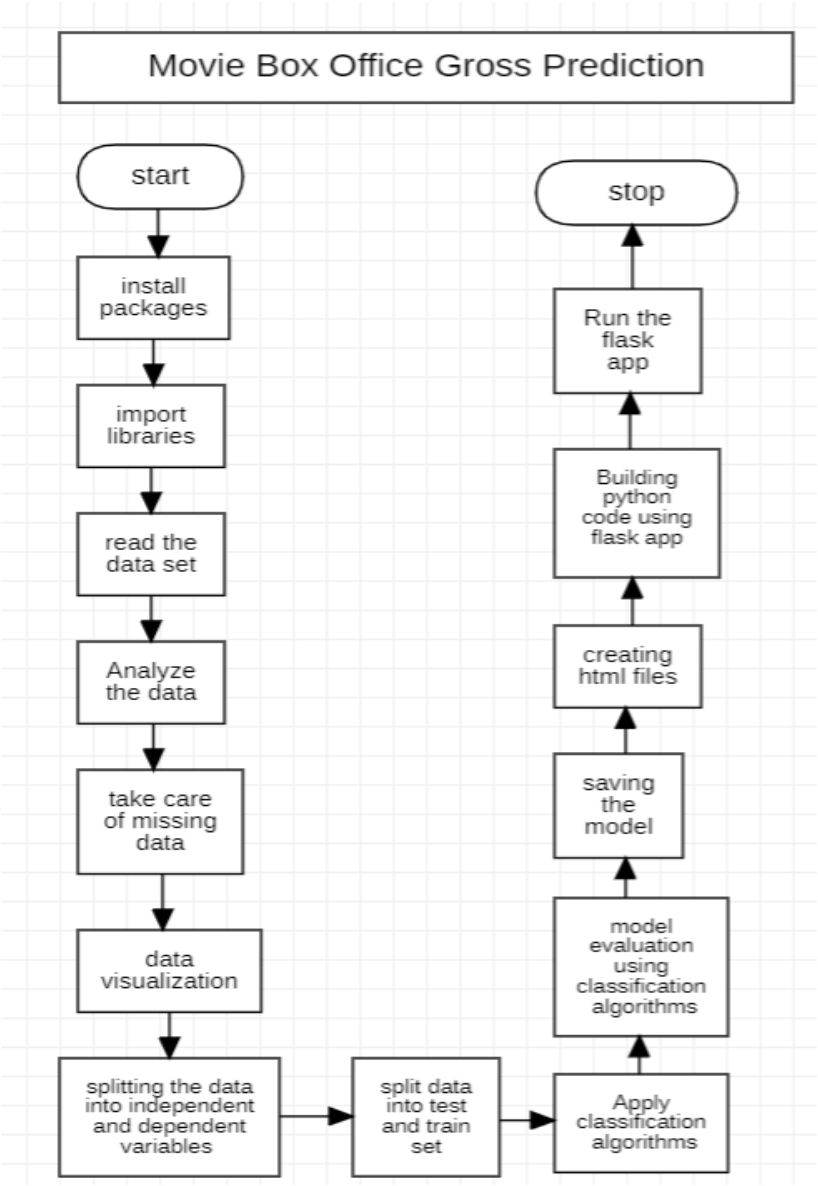
(2) Random Forest Classification

Random Forest or Random decision forests are an ensemble method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean/average prediction of the individual trees.

(3) KNN Classification algorithm or K-Nearest Neighbour algorithm

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

5.FLOW CHAR


Movie Box Office Gross Prediction
start
install packages
import libraries
read the data set
Analyze the data
take care of missing data
data visualization
splitting the data into independent and dependent variables
split data into test and train set
Apply classification algorithms
model evaluation using classification algorithms
saving the model
creating html files
Building python code using flask app
Run the flask app
stop

6.RESULT

As we see the predicted output is displayed on the User Interface

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- **Efficiency in workflow :** One of the first desires that probably comes to mind is efficiency. When building your website, you want to be able to reach as many people as you can. The system predicts an approximate success rate of a movie based on its profitability by analyzing data.

- **Reduce costs :** You don't need a large time to wait for the results with in a seconds the amount will be predicted .

- Using machine learning algorithms to predict movie box office gross in data sets. Various kinds of data sets, have to use this to train classifier algorithms to predict movie gross with good accuracy.

DISADVANTAGES:

- Any single error in data set can change the entire data.
- Correct accuracy must be needed while doing the project using supervised machine learning algorithms.

- Python code should be correct without any error.

8. APPLICATIONS:

This application can further be developed with more idea and implementation and by using different algorithms. The accuracy score of the model can be further improved by using decision tree and also by increasing the data set, K-Nearest Neighbours algorithm is also one of the pertinent methods which can be used to predict the movie gross accurately. It proposes to improve the accuracy further.

9.CONCLUSIONS

Prompt and timely accurate prediction of Movie box office gross plays a vital role in decreasing for producers and stakeholders. In this paper, an attempt is made to predict the presence of Movie gross using Support vector machine, Random Forest , K-NN classification methods of Machine Learning.

I developed a computational model for movie box office gross prediction using a combination of features extracted from movie database metadata, budget-revenue relationship graphs, popularity-revenue relationship graphs, and movie-Revenue relationship graphs. I demonstrated that by using features extracted from these runtime-movies and revenue-movie relationship graphs, we are able to create a more accurate model than using metadata features alone.

Among three ML classification methods, SVM and RF performed better than K-NN classification. Although, the accuracy levels for all three methods performed well based on the testing data set.

10.FUTURE SCOPE

There are a variety of extensions that could be made to the existing model proposed in this paper. One alternative would be to model movie gross as a continuous quantity rather than a discrete quantity. Another extension would be to introduce a temporal model for how movie genre and actor popularity change over time, which one might suspect would lead to more accurate gross predictions. Finally, perhaps the biggest improvement that could be made would be to acquire more Movie box office gross data.

11.BIBILOGRAPHY

We referred some books and surfed the internet for the better outcome of the project

[1] Simonoff, J. S. and Sparrow, I. R. Predicting movie grosses: Winners and losers, blockbusters and sleepers. In Chance, 2000.

[2] Joshi, M., Das, D., Gimpel, K., and Smith, N. A. Movie Reviews and Revenues: An Experiment in Text Regression. In Proceedings of the North American Chapter of the Association for Computational Linguistics Human Language Technologies Conference, 2010.

[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:

```
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 pyhton expression
#Reading the dataset by using pandas read_csv function
credits=pd.read_csv(r"D:\ML_training may 2020\Projects_50\Final\Movie Box Office Gross Prediction Using ML\dataset\tmdb_5000_credits.csv")


movies_df=pd.read_csv(r"D:\ML_training may 2020\Projects_50\Final\Movie Box Office Gross Prediction Using ML\dataset\tmdb_5000_movies.csv")
#head() gives us first  5 rows of the dataset
credits.head()
credits.tail()
movies_df.head()
#columns in the dataset
print("credits:",credits.columns)
print("movies_df:",movies_df.columns)
#Shape of the dataset
print("credits:",credits.shape)
print("movies_df:",movies_df.shape)
#Renaming the columns
credits_column_renamed=credits.rename(index=str,columns={"movie_id":"id"})
movies=movies_df.merge(credits_column_renamed,on="id")
movies.head()
movies.shape
#information about the datset
movies.info()
movies.describe()
```

```
# 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']
movies.head()
print("movies:",movies.shape)
#corr() is to find the relationship between the columns
movies.corr()
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()
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 o,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
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');
#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()

#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');

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.dtypes
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'])
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
movies_box.head(3)
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()
#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
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]

3.76955224*100000000

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)
```

FLASK APP CODE:

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

import pandas as pd

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','runt
ime','vote_average','vote_count',

'director','release_month','release_DOW']

x_df=pd.DataFrame(features_values,columns=f
eature_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_tex
t=prediction[0])
if __name__ == "__main__":
    app.run(debug=False)
```

HTML files:

Demo2.html:

```
<html>
<style>
body
{
background-image:url('../static/img/reel1.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;}
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>
<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/img/reel2.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>
```