Movie Box Office Gross Prediction Using Machine Learning

1.INTRODUCTION

Overview

The income of film industry comes from screening movie in the theater, which is called “Box-Office”. Film industry is a highly competitive industry. Many new movies queue up to be released each week, so a theater owner has to decide on which movie to be shown, based mainly on revenue. Regression analysis is a widely-used technique to predict revenue. This paper aims to compare accuracy among various types of regression analysis, with and without clustering techniques. Specifically, for regression analysis, we used three different types of regression to create prediction models, which are linear regression, polynomial regression, and support vector regression. As clustering into smaller groups of similar items may improve accuracy of the prediction models, we cluster the movie data and apply regression analysis on each cluster.

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.

Proposed System

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.. .

2. LITERATURE SURVEY

2.1 Existing Problem

Movies, in general, are products that have a long development stage until they reach final consumers and normally at a high cost level. We can describe a movie development process, in a broader way, as being composed of 4 (four) stages: Pre-production, production, post-production and distribution.The large growth in number of movies releasing over the past few decades Movie Prediction is necessary. The only way people can check whether the movie will be worth to watch is through applications, so this system would analyze the reviews posted by other users, as these reviews are large in number which the user cannot read and gets confused. Following are the aims and objectives suggested by our system.

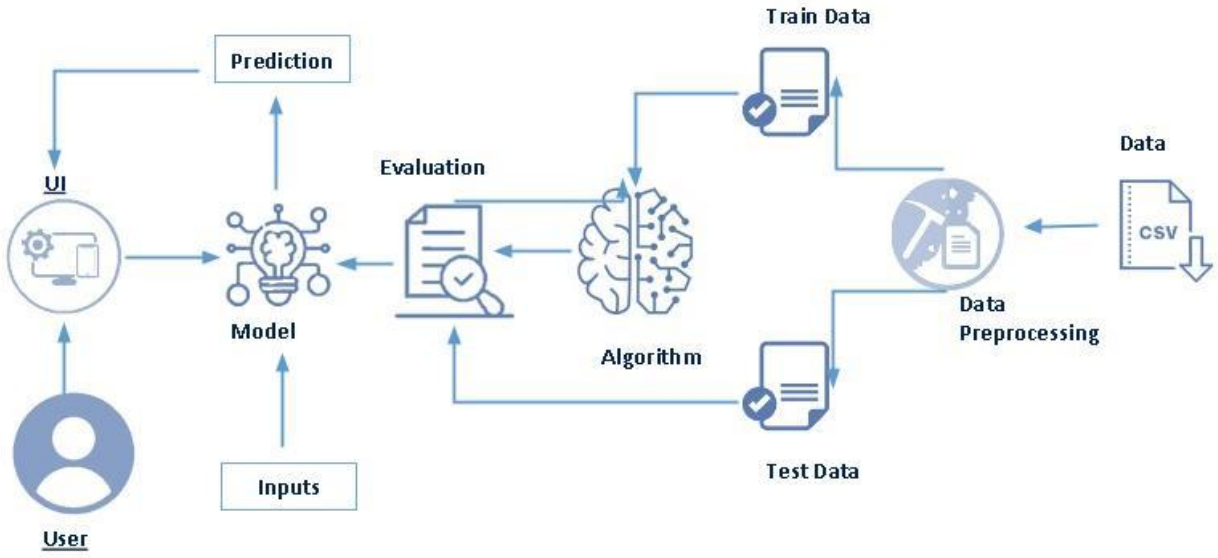
The first step is to identify a dataset of movie data which is suitable for analysis. Relevant attributes need to be selected from the movie data. Attributes can be general pre-production information regarding film productions such as movie title, sequel, genre, language and information about writers, actors, and directors. Similarly, the data must include some measure of success, such as user movie ratings. Secondly, the relevant dataset has to be prepared and structured in such a way that the data used is representative of the movie scene at large, as well as suitable for analysis by the relevant machine learning techniques and algorithms. Further, correlation is performed on relevant dataset to find the relationship between al the variable with each other. The important step in training our system is to apply classification model. There are many classifiers. Lastly, the prediction performance of the relevant machine learning algorithm has to be evaluated on the dataset in order to determine success and failure of movie accurately.

2.2 Proposed System

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.

3. THEORETICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware/Software Designing

Hardware requirements

- 2 GB ram or above
- Dual core processor or above
- Internet connection

Software requirements

- Anaconda Navigator
- Python packages
- Jupyter note book

4.EXPERIMENTAL INVESTIGATION

Here we are going to build a machine learning model that predicts the movie box office production with the help of natural language processing. Natural Language processing or NLP is a subset of Artificial Intelligence (AI), where it is basically responsible for the understanding of human language by a machine. It is done by creating a corpus to store prediction pre-processing the data and splitting them to test and train sets. From this a model is created then our box office prediction will predict accurately . Proposed system is to predict the success and failure of upcoming movie based on several attributes. The system will use a classification model to find correlation between several attributes like actor, movie rating etc. The system will predict success rating from the correlation between various movie criteria. This model can be used by movie watchers, producers.

5. FLOWCHART

- The user interacts with the UI (User Interface) to upload the input features.
- Uploaded features/input is analyzed by the model which is integrated.
- Once the model analyses the uploaded inputs, the prediction is showcased on the UI.

  To accomplish this, we have to complete all the activities and tasks listed below Tasks:

- Data Collection.
    - Collect the dataset or Create the dataset
- Data Preprocessing.
    - Import the Libraries.
    - Reading the dataset.
    - Exploratory Data Analysis
    - Converting json objects to strings
    - Checking for Null Values.
    - Data Visualization.
    - Dropping the columns
    - Label Encoding
    - Splitting the Dataset into Dependent and Independent variable.
    - Feature scaling
    - Splitting Data into Train and Test.
- Model Building
    - Training and testing the model
    - Evaluation of Model
    - Save the model
    - Predicting the output using the model
- Application Building
    - Create an HTML file
    - Build a Python Code
    - Run the app

6. RESULT

Movie Box Office Gross Prediction Using ML

Enter your details and get probability of your movie success

Enter budget 237

Action

Enter popularity 150

Enter runtime 162

Enter vote_average 7.2

Enter vote_count 11800

Clint Eastwood

Enter the month of release 12

Enter the week of the month 3

Predict

Movie Box Office Gross Prediction Using ML

The Revenue predicted is [1203.22932147] million $

It can perform both regression and classification tasks. A random forest produces good predictions that can be understood easily. It can handle large datasets efficiently. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.

7. ADVANTAGES AND DISADVANTAGES

Advantages

1. Predict the movie revenue.
2. Effective prediction technique
3. Beneficial for accurate prediction of which movies has to be released on priority basis.
4. Secure and efficient system
5. You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.

Disadvantages

1. country's GDP rate can be used as a feature to know if there is financial stability during the period when a movie is released
2. When using a random forest, more resources are required for computation. It consumes more time compared to a decision tree algorithm.

8. APPLICATIONS

- Cinematography
- Movie box office
- Box office tracking

9. CONCLUSION

The aim of this project was to investigate which factors affect the profitability of a film, and develop models to predict film performance in both pre-release and post-release settings. We have explored how to best feature engineer raw data and designed models to answer the Yes or No question about profitability, as well as make concrete predictions of box office performance before and after release.

We solved a classification problem to predict a film will be profitable using data available prior to release using a Logistic Regression and an XGBoost Model. Inspection of the coefficient terms of the logistic regression model showed general insight into which features had a negative or positive impact on profitability and. We performed extended analysis to obtain more informative Shapley values for both models and showed how values of certain features impact profitability. Comparing Shapley feature plots for the logistic regression and XGBoost mostly concurred on the importance and effects of features, with the XGBoost plot yielding more insight into the effects of magnitudes of values. We identified surprising trends identified during model evaluation and suggested reasons why they may be, recognising that can be other sources of revenue that we do not consider, that contribute to profitability.

10. FUTURE SCOPE

This program allows users to predict movie box office prediction . By the help of this prediction**,** 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.

The intent of this study of creating a way to predicting the success of movies using historical data was not a direct success, it shows that other features are necessary to accurately make a prediction. When trying to predict the rating of a movie a highest success rate of around 65% achieved when using 6 groups and 83% when using 4 groups. This could arguably be a good enough result and shows promise, but when it is compared to previous studies with similar methodologies the result equal in predictability.

11. BIBLIOGRAPHY

- **Data set:** **https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata**
- **https://medium.com/analytics-vidhya/how-to-use-machine-learning-approach-to-predict-movie-box-office-revenue-success-e2e688669972**
- **https://www.kaggle.com/c/tmdb-box-office-prediction**

12. APPENDIX

```
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'))#load
ing the saved model
scalar=pickle.load(open("scalar_movies.p
kl","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','popula
rity','runtime','vote_average','vote_count',
                  'director','release_month','relea
se_DOW']
    x_df=pd.DataFrame(features_values,col
umns=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",predictio
n_text=prediction[0])
if __name__ == "__main__":
    app.run(debug=False)
```