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

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 pyhton 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) > 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()
#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 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
#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 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)})
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(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>


        <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 }}
```

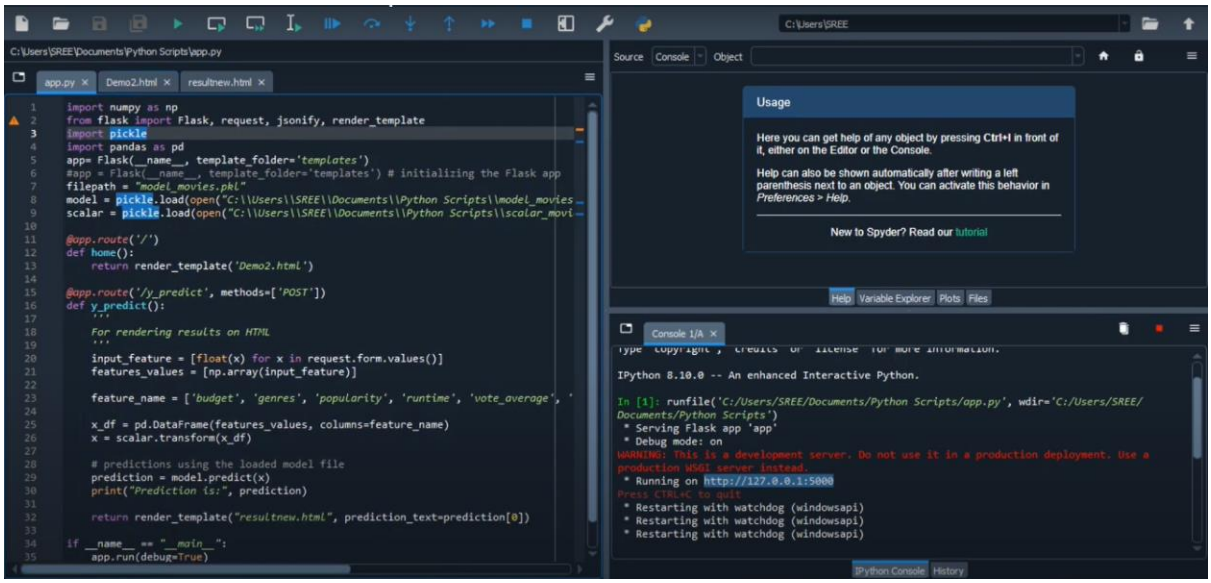
```
    </div>
  </body>
</html>
```

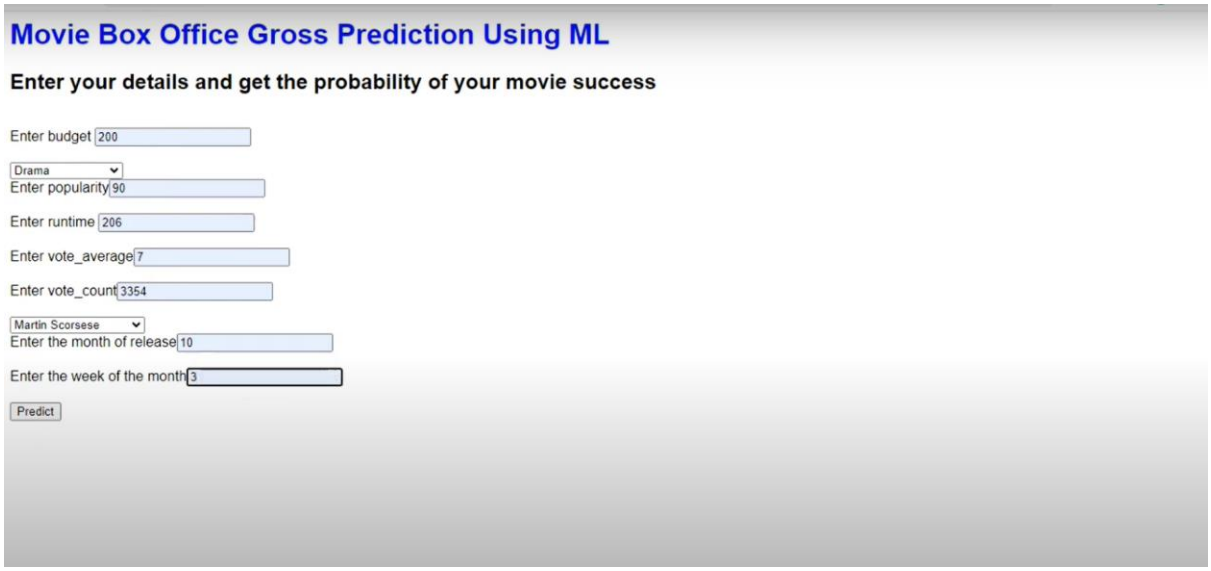
Resultnew.html:

```
<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>
```

Output Screenshots :-





Movie Box Office Gross Prediction Using ML

The Revenue predicted is [571.70324721] million $