IMDB Movies

Tim Bialek, Adrienne Jennings , Jack Petrella , Stephanie LaForge,

Data

Introduction

Movies play a huge role in the entertainment business and the industry has been growing rapidly. We chose a Movies Database because we wanted to create a model that would be able to play a role in future castings, movie expectations from budget to revenue and if the data presented in the past can accurately predict a movie, revenue than their original budget.

Introduction

Data set from IMBD that contains information about the top movies and includes categories like years, production companies, and directors.

The key factors that we are going to analyze is to merge the data sets to get one clean file that will contain the needed information from The IMDB Movies CSV and the Numbers CSV. Our merged movie database we will analyze several key factors; actors, budget, & directors to help play a role in determining revenue.

In order to predict correctly we had to make tables and drop information that was not valuable to the equation. After keeping the columns we ran into some issues and had to make sure that there were no corruptions while importing our csv's to created merged data frames. After fixing a few errors we were able to upload, merge, and export to SQL.

TOur main focus will be to predict the revenue for future movies based off the actors, budget, and director.

After realizing there were only several aspects we wanted to focus on we chose to stick with main components of actors vs budget since future investors or movie companies would want to use when receiving data information from a large movie dataset. To predict if a movie is going to profitable and make more revenue than their original budget will be a big deal breaker in future choices.

The data is a combination of three data sources.

IMDB - IMDB Movies Dataset | Kaggle

Kaggle - EDA on IMDB Movies Dataset | Kaggle

Numbers - The Numbers - Movie Budgets (the-numbers.com)

Data from IMDB Movie

CSV

- PosterLink: Link of the poster that imdb using
- SeriesTitle: Name of the movie
- ReleasedYear: Year at which that movie released
- Certificate: Certificate earned by that movie
- Runtime: Total runtime of the movie
- Genre: Genre of the movie
- IMDB Rating: Rating of the movie at IMDB site
- Overview: mini story/ summary
- Meta_score: Score earned by the movie
- Director: Name of the Director
- Star1,Star2,Star3,Star4: Name of the Stars
- No of votes: Total number of votes
- Gross: Money earned by that movie

Database:

Data will be stored in AWS

The data was imported and cleaned through Jupyter Notebook

The next step was to create a database engine that will allow Pandas to communicate with PostgresSQL

The machine learning will be through Jupyter Notebook that will also have an engine to communicate with PostgresSQL.

In PostgresSQL the tables will be exported to cleaned CSV files.

The EDA code was created & ran in Jupyter Notebook

Data Tables - Cleaning the Data

Below is a excerpt from the code used to cleaned the Data from each CSV imported into Jupyter Notebook. After importing the CSV, it was cleaned by dropping columns that were not needed and creating data frames with the new columns free from any Nan's or abstract characters. As you can see below we ran into an issue from the Numbers budget having a "A" character in front of the dollar amounts. By catching this we were able to have clean imported data

```
#READ IN FILE FOR THE-NUMBERS
#numbers_df = pd.DataFrame(f'{file_dir}/TheNumbersData.csv',encoding='unicode_escape')
numbers_df = pd.read_csv("Resources2/TheNumbersData.csv", encoding= 'unicode_escape')
numbers_df = numbers_df.drop(['ï»¿Number', 'Release Date', 'Domestic Gross'], axis=1)

cols_to_check = ['Production Budget','Worldwide Gross']
numbers_df[cols_to_check] = numbers_df[cols_to_check].replace({'Â':''}, regex=True)

numbers_df.rename({'Movie':'original_title', 'Production Budget': 'production_budget', 'Worldwide Gross': 'worldwide_gross'},axis='columns',
```

```
# Set the DataFrames from the return statement equal to the file names above.
wiki_movies_df = wiki_file
movies_df = kaggle_file
numbers_df = numbers_file
```

JOIN

Numbers Data with the Movies Data using a full outer join from Jupyter Notebook

Query Editor | Query History

```
CREATE TABLE movies_data AS
    (SELECT numbers.index,
        numbers.original_title,
        numbers.production_budget,
        numbers.worldwide_gross,
        movies.imdb_id,
        movies.kaggle_id,
        movies.runtime,
        movies.budget,
        movies.revenue,
        movies.release_date,
        movies.popularity,
        movies.vote_average,
        movies.vote_count,
        movies.genres,
        movies.country,
        movies.production_companies,
        movies.production_countries,
        movies.distributor,
        movies.producers,
        movies.director,
        movies.starring,
        movies.cinematography,
        movies.editors,
        movies.writers,
        movies.composers
    FROM numbers
    FULL JOIN movies
    ON numbers.original_title = movies.original_title);
```

```
db_string = f"postgresql://postgres:{db_password}@127.0.0.1:5432/class_project"
engine = create_engine(db_string)
movies_df.to_sql(name='movies', con=engine, if_exists='replace')
```

75

```
db_string = f"postgresql://postgres:{db_password}@127.0.0.1:5432/class_project"
engine = create_engine(db_string)
numbers_df.to_sql(name='numbers', con=engine, if_exists='replace')
```

295

Data Tables

The first table was used with the Numbers data
Contained Title, Production Budget, & Worldwide Gross

The second table was with the Movies Data
Index ,original_title ,production_budget, worldwide_gross, imdb_id, kaggle_id, runtime, budget, revenue, release_date, popularity, vote_average, vote_count, genres, country, Production_companies, production_countries, distributor, producers, director, starring, cinematography, editors, Writers, composers

By merging these two CSV Files we were able to get the data needed in order to have a Budget Column that was accurate and updated combined with the Movie Data.

app.quickdatabasediagrams.com/#/

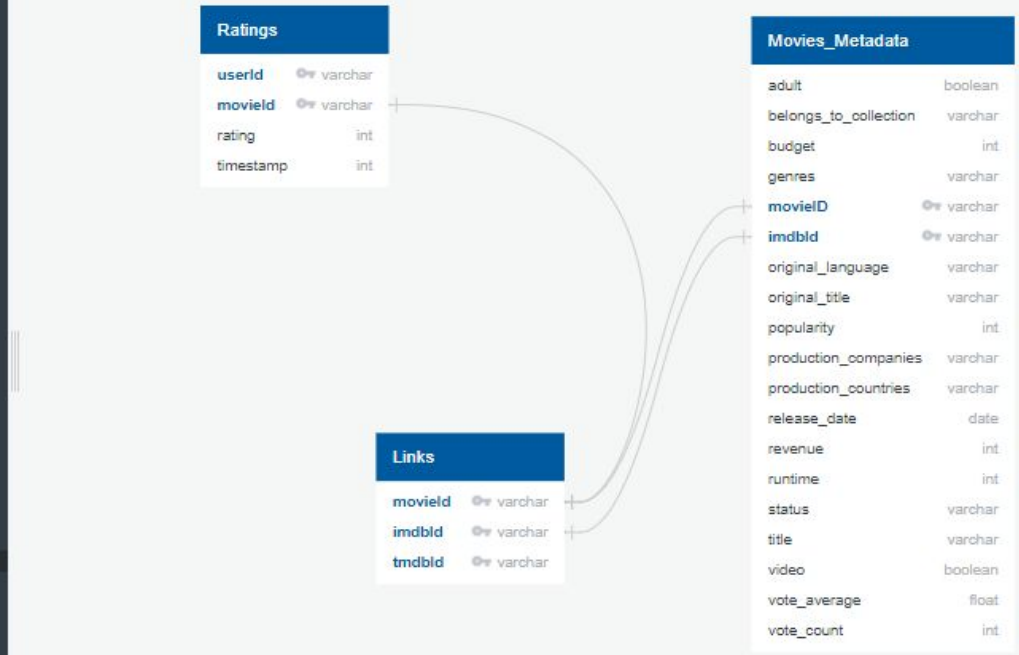
FILE EDIT EXPORT IMPORT DOCS SAVE

SQL

```
Ratings
-
userId varchar pk
movieId varchar pk fk - Links.movieId
rating int
timestamp int

Links
-
movieId varchar pk
imdbId varchar pk
tmdbId varchar pk

Movies_Metadata
-
adult boolean
belongs_to_collection varchar
budget int
genres varchar
movieID varchar pk fk - Links.movieId
imdbId varchar pk fk - Links.imdbId
original_language varchar
original_title varchar
popularity int
production_companies varchar
production_countries varchar
release_date date
revenue int
runtime int
status varchar
title varchar
video boolean
vote_average float
vote_count int
```



Machine Learning Model

Deliverables - Descriptions

The preliminary data preprocessing was to take the three data sets used and to clean the data so it can be ready to be used in data frames, tables, dropped columns, and removed empty spaces.

The feature engineering:

Created Target Variable “movie_success” for classification supervised ML predictions. Movie Success is determined if the gross revenue column was greater than the movie’s budget column (1). If the gross revenue was equal to or less than the budget, a 0 was used to indicate the movie was unsuccessful. Label Encoding was completed for the categorical features to prepare for the ML process. For the ML models practiced with that benefit scaling, the StandardScaler was implemented to scale and normalize the features.

Preliminary feature selection & decision making process:

The feature selection was determined by discussing which aspects of a movie would have an effect on determining if a movie would produce a profitable result. Past movie’s budget and revenue are key features for the model, and additional features that play a part in the movie’s success include genre, star actors/actresses, writers, directors, and runtime.

How data was split into training and testing sets:

The train_test_split module is used to split our features and target variable into training and test datasets.

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	240	11
Actual 1	20	712

Accuracy Score : 0.9684638860630722

Classification Report

	precision	recall	f1-score	support
0	0.92	0.96	0.94	251
1	0.98	0.97	0.98	732
accuracy			0.97	983
macro avg	0.95	0.96	0.96	983
weighted avg	0.97	0.97	0.97	983

The Model Choice -

Decision Tree Classifier, handles outliers the best

EDA

EDA

- Used to import pandas, matploblib.pyplot to create graphs needed.

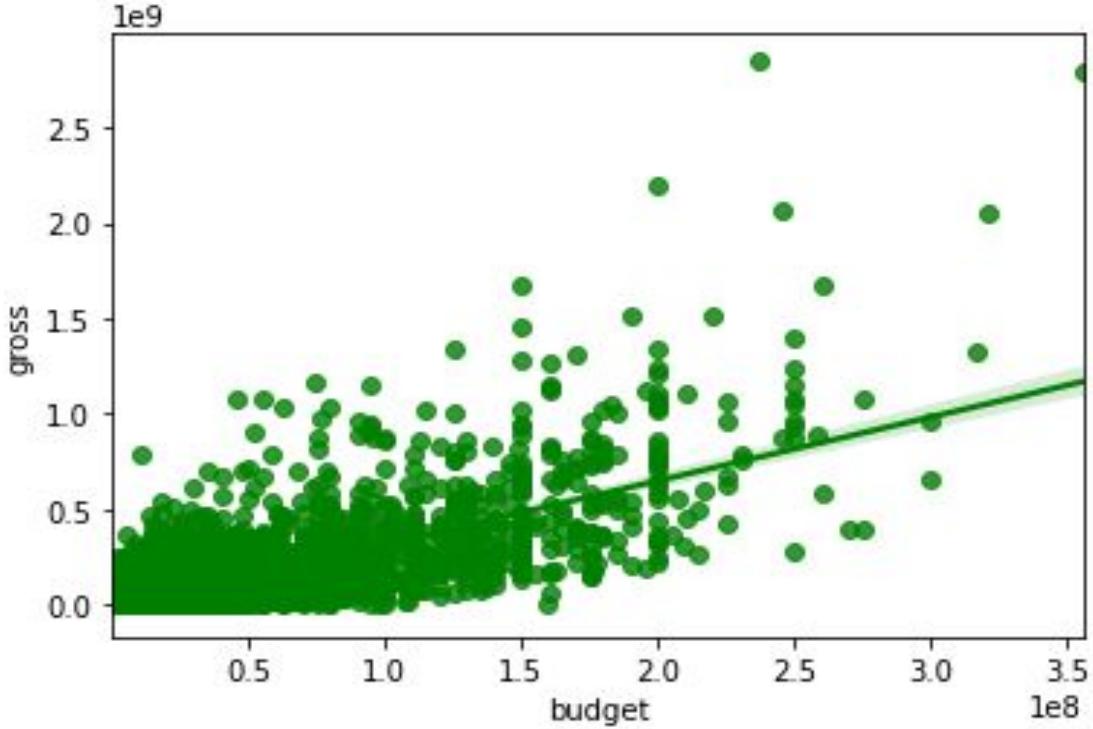
- After cleaning through the data and creating tables needed in order to gather the data needed that will play a role in our predictions.

- Performing EDA will will help give insight to the data, an understanding of the data types, and allow us to determine what features are important and the relationships between them. We can also test any underlying assumptions during the exploratory phase.
- It was important to find
- Started with Data Analysis, Correlation Analysis, Categorical Analysis, Visualizations, Data Relationships

After viewing the graphs in these visualizations we can start asking important questions:
One of the Gross Revenue Graphs is showing a increase in movies revenue from 1980 to 2020.
What is causing this? More Movie theaters? Increase in Rate? More people going to moves?

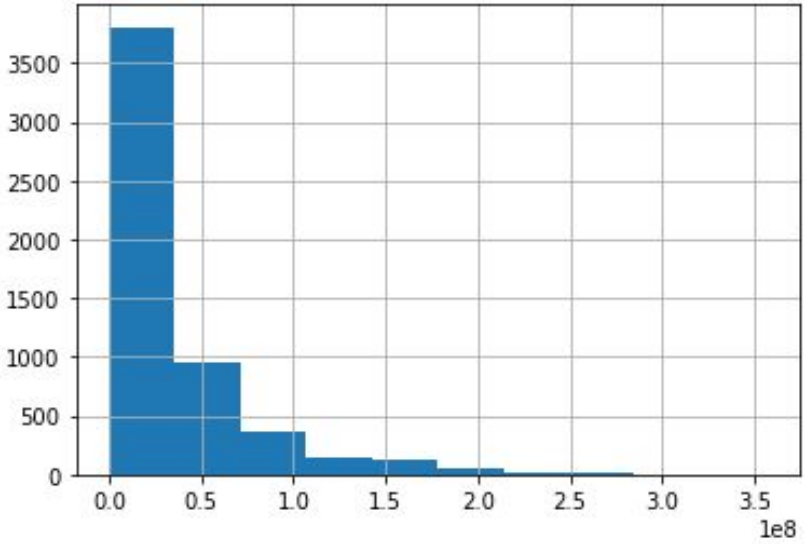
Graphs

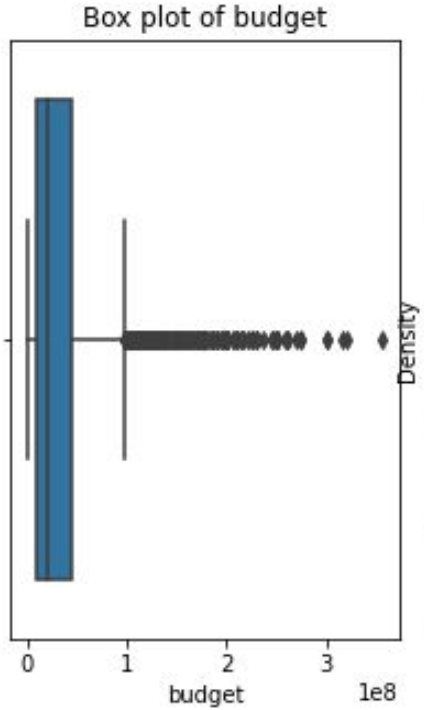
Scatter plot with regression line;

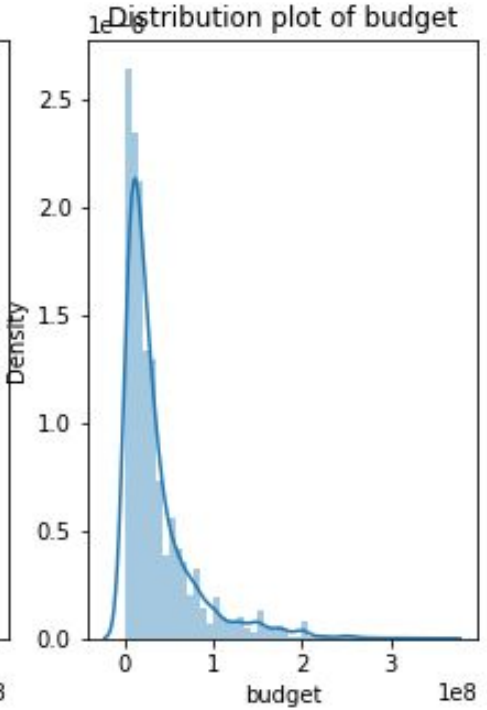


Budget Graphs

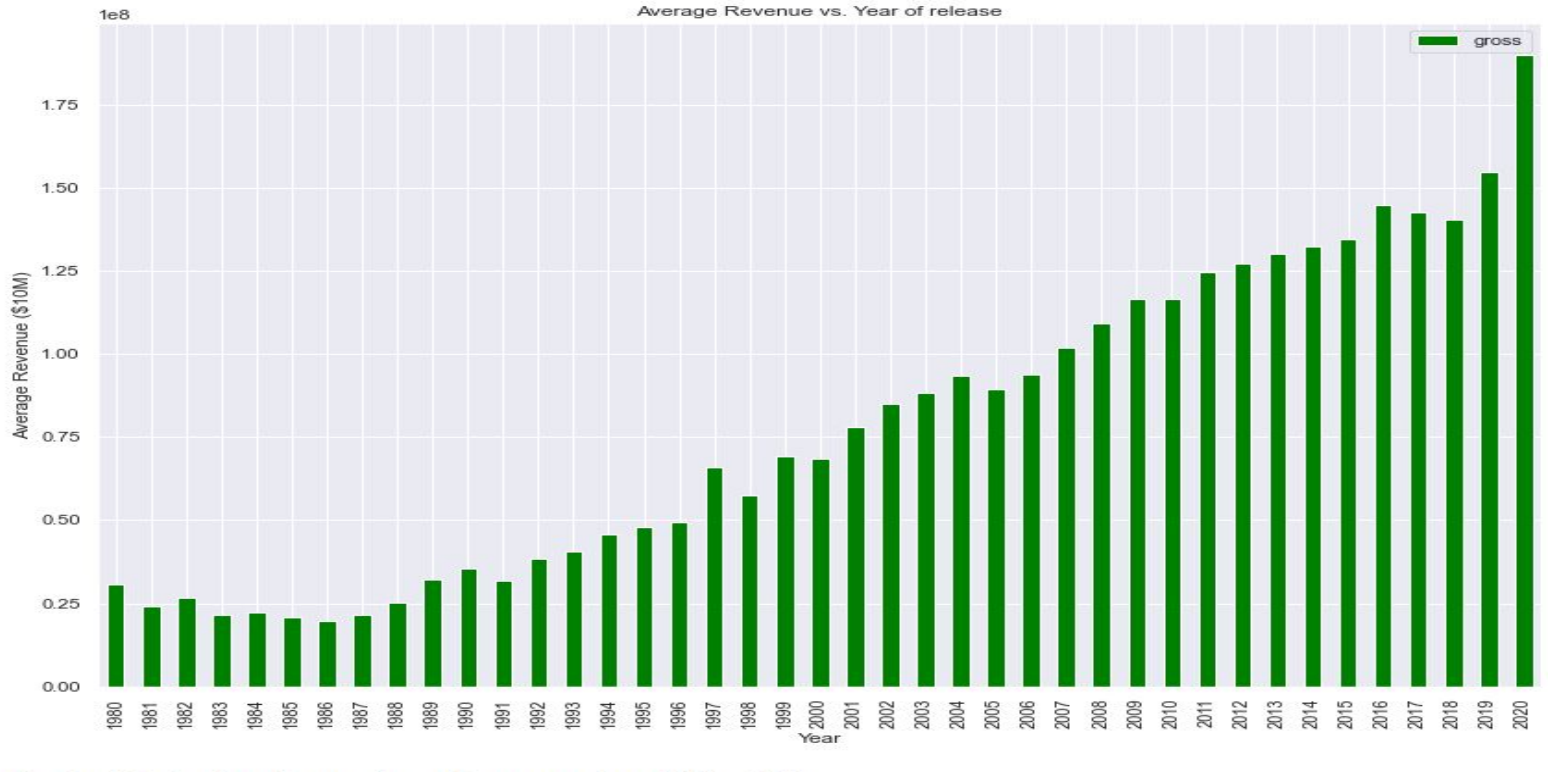
Histogram



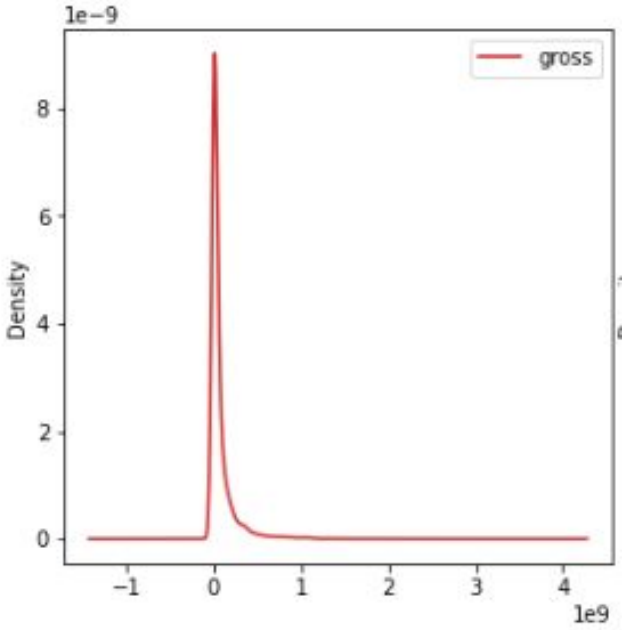


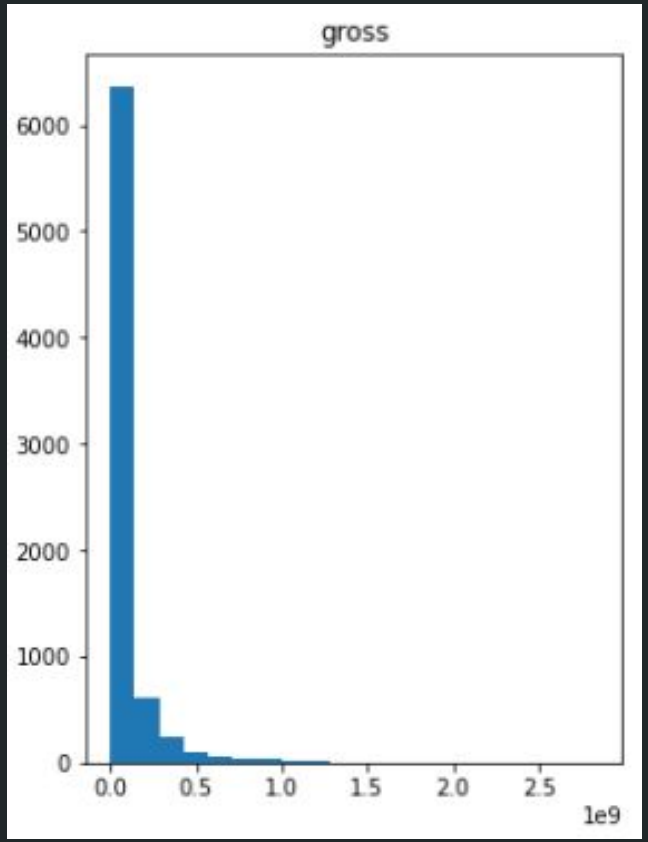


Gross Revenue Graphs



Gross Revenue Graphs Cont.





Technologies - Preliminary

The interactive dashboard would be hosted on GitPages we will be using JavaScript, HTML, CCS, Bootstrap 4

Dashboard


Average rounds/encounter by targetSelector : increase[15m]
15.0
12.5
10.0
7.5
5.0
2.5
16:00
20:00
00:00
04:00
08:00
12:00
avg
BiggestFirst 8.63
Overall 7.61
LowestHealth 7.56
HighestHealth 7.41
SmallestFirst 7.27
Random 7.05

Dashboard - Blue Print

IMDB MOVIES	
	Paragraph; By having a dashboard that take data based off the criteria in the past and make predictions to help future movie investors, production companies, or streaming platforms make calculated predictions. The importance of these predictions can help keep revenue above budget and predict how well a movie will sell based on the popularity or movie presence based off actors in their past roles.
Filters: Actors / Budget /Directors	Table of Movies results for Revenue

Tools - JavaScript , HTML
Interactive Elements - filters for director, actor, budget