

CSCE 5210 - Fundamentals of Artificial Intelligence

Project: Movie Rating and Revenue Prediction

Project Final Report

Abstract

The purpose of this project is to make predictions of movie ratings and revenues using artificial and machine learning. The project uses two datasets namely, IMDB and TMDB. We developed three predictive models that predict movie ratings and revenues which can influence decisions in the film industry. The following are the steps adopted in our approach: data preprocessing, model training, and interface construction, which respectively deliver relatively moderate to high accuracy in forecast results.

Introduction

Predicting movie ratings and revenues is very important to producers, investors, and marketers in the film industry. An accurate prediction can help in decision-making and resource allocation in this highly competitive industry. The motivation for this project is to harness machine learning models to forecast movie success based on features such as genre, actors, budget, and release timing. The project uses data from IMDB and TMDB to build these models. The input consists of movie-related features such as budget, genres, cast, and release dates. The expected output is the predicted rating and revenue for a movie before its release. The process involves data collection, preprocessing, training machine learning models, and evaluating their performance. This project falls under the domain of predictive analytics in AI. The methods and tools used include Random Forest Regressor, XGBoost, and Neural Networks, implemented using Python and relevant libraries such as scikit-learn and TensorFlow/Keras.

Area of Application, Dataset, and Features

The project focuses on predictive analytics in the film industry. We used two main datasets: the [IMDB Dataset](#), containing movie reviews and sentiments, and the [TMDB Dataset](#), which includes movie metadata such as budget, genres, popularity, and revenue. The IMDB Dataset was transformed into a binary classification format, while the TMDB Dataset underwent preprocessing to handle missing values, encode categorical variables, and standardize numerical features. The datasets were split into training (80%) and testing (20%) sets. Example features include genres, budget, and cast details, which are used to predict ratings and revenues.

```

1 import pandas as pd
2
3 # Load IMDB dataset
4 imdb_df = pd.read_csv('IMDB Dataset.csv')
5
6 # Preprocess the IMDB dataset
7 def preprocess_imdb_data(df):
8     df['review'] = df['review'].str.replace('<br />', ' ')
9     df['sentiment'] = df['sentiment'].apply(lambda x: 1 if x == 'positive' else 0)
10    return df
11
12 imdb_df = preprocess_imdb_data(imdb_df)
13 print(imdb_df.head(10))
14
[10] ✓ 1.4s Python

```

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The filming t...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1
5	Probably my all-time favorite movie, a story o...	1
6	I sure would like to see a resurrection of a u...	1
7	This show was an amazing, fresh & innovative i...	0
8	Encouraged by the positive comments about this...	0
9	If you like original gut wrenching laughter yo...	1

Figure: example data being loaded in python

Methods

Data Preprocessing:

- IMDB Dataset: Cleaned and transformed sentiment data into a binary classification format.
- TMDB Dataset: Handled missing values, encoded categorical variables, and standardized numerical features.

```

1 import json
2
3 # Load TMDB dataset
4 tmdb_df = pd.read_csv('tmdb_5000_movies.csv')
5
6 # Preprocess the TMDB dataset
7 def preprocess_tmdb_data(df):
8     df['genres'] = df['genres'].apply(lambda x: ', '.join([genre['name'] for genre in json.loads(x)]))
9     df['keywords'] = df['keywords'].apply(lambda x: ', '.join([keyword['name'] for keyword in json.loads(x)]))
10    df['production_companies'] = df['production_companies'].apply(lambda x: ', '.join([company['name'] for company in json.loads(x)]))
11    df['production_countries'] = df['production_countries'].apply(lambda x: ', '.join([country['name'] for country in json.loads(x)]))
12    df['spoken_languages'] = df['spoken_languages'].apply(lambda x: ', '.join([language['name'] for language in json.loads(x)]))
13    df['release_date'] = pd.to_datetime(df['release_date'])
14    df['release_year'] = df['release_date'].dt.year
15    return df
16
17 tmdb_df = preprocess_tmdb_data(tmdb_df)
18 print(tmdb_df.head())
19
[2] ✓ 0.3s Python

```

	budget	genres	homepage	id
0	237000000	Action, Adventure, Fantasy, Science Fiction	http://www.avatarmovie.com/	19995
1	300000000	Adventure, Fantasy, Action		
2	245000000	Action, Adventure, Crime		
3	250000000	Action, Crime, Drama, Thriller		
4	260000000	Action, Adventure, Science Fiction		

Figure: Data processing

Machine Learning Algorithms Models

- **Random Forest Regressor:** Used for predicting movie revenues.
- **XGBoost:** Employed to predict both ratings and revenues.
- **Neural Networks:** Explored for their potential in capturing complex relationships.

Each method was selected for its ability to handle the specific characteristics of the dataset, and cross-validation was used for hyperparameter tuning and model selection.

Experiments, Results, and Discussion

Experiments:

- Data was split into training and testing sets (80/20 ratio).
- Cross-validation was performed to fine-tune hyperparameters.

Results:

Using random forest:

- Rating Prediction Model.:
 - MAE: 0.5766
 - RMSE: 0.7784
 - R^2 : 0.5103
- Revenue Prediction Model:
 - MAE: 41,451,530.36
 - RMSE: 87,075,235.39
 - R^2 : 0.7149

```

File Edit Selection View Go Run ...
Movie Rating and Revenue
main.ipynb X index.html # style.css JS script.js
main.ipynb > Neural Networks Implementation > import pandas as pd
+ Code + Markdown | Run All | Restart | Clear All Outputs | View data | Variables | Outline ... Python 3.11.3
7 revenue_model = RandomForestRegressor(n_estimators=100, random_state=42)
8 revenue_model.fit(X_train, y_train_revenue)
9
10 # Evaluate the models
11 def evaluate_model(model, X_test, y_test):
12     y_pred = model.predict(X_test)
13     mae = mean_absolute_error(y_test, y_pred)
14     rmse = np.sqrt(mean_squared_error(y_test, y_pred))
15     r2 = r2_score(y_test, y_pred)
16     return mae, rmse, r2
17
18 mae, rmse, r2 = evaluate_model(rating_model, X_test, y_test_rating)
19 print(f'Rating Model - MAE: {mae}, RMSE: {rmse}, R2: {r2}')
20
21 mae, rmse, r2 = evaluate_model(revenue_model, X_test, y_test_revenue)
22 print(f'Revenue Model - MAE: {mae}, RMSE: {rmse}, R2: {r2}')
[4] ✓ 14.3s
Python
Rating Model - MAE: 0.5765614583333333, RMSE: 0.7784416076099306, R2: 0.5103187690547113
Revenue Model - MAE: 41451530.35896875, RMSE: 87075235.38772306, R2: 0.7149126915465905

```

Figure: random forest results

We also printed the prediction results of the model:

```

app.py main.ipynb X index.html # style.css JS script.js
main.ipynb > ...
+ Code + Markdown | Run All | Restart | Clear All Outputs | View data | Variables | Outline ... Python 3.11.3
11 movie_name = movies_subset.iloc[i]['original_title']
12 running_time = movies_subset.iloc[i]['runtime']
13 popularity = movies_subset.iloc[i]['popularity']
14 release_year = movies_subset.iloc[i]['release_year']
15 vote_count = movies_subset.iloc[i]['vote_count']
16
17 print(f'Movie: {movie_name}')
18 print(f' Predicted Rating: {rating:.2f}')
19 print(f' Predicted Revenue: ${revenue:.2f}')
20 print(f' Running Time: {running_time} minutes')
21 print(f' Popularity: {popularity}')
22 print(f' Release Year: {release_year}')
23 print(f' Vote Count: {vote_count}')
24 print("-----")
[4] ✓ 0.0s
Python
Detailed Predictions:
Movie: Avatar
Predicted Rating: 5.64
Predicted Revenue: $54871449.38
Running Time: 162.0 minutes
Popularity: 150.437577
Release Year: 2009.0
Vote Count: 11800
-----
Movie: Pirates of the Caribbean: At World's End
Predicted Rating: 5.90
Predicted Revenue: $7.36
Running Time: 169.0 minutes
Popularity: 139.082615
Release Year: 2007.0
Vote Count: 4500
-----
Movie: Spectre
Predicted Rating: 6.29
Predicted Revenue: $203077.50
Running Time: 148.0 minutes
Popularity: 107.376788

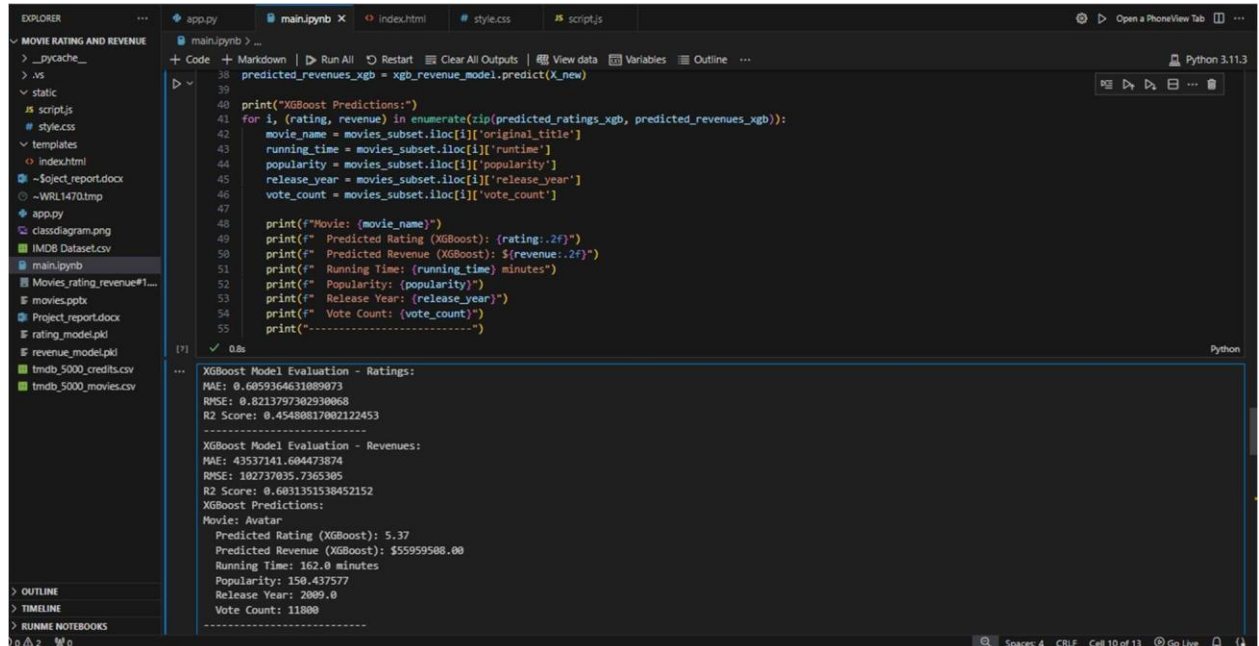
```

Figure: random forest prediction results

The rating prediction model achieved a moderate level of accuracy with an R^2 value of 0.5103, indicating that approximately 51.03% of the variance in movie ratings can be explained by the model. The MAE and RMSE values further validate the model's performance, with low error rates suggesting reliable predictions. The accuracy of the revenue prediction model was as follows; The

R^2 value of this model was equal to 0. 7149, which means that it is approximately 71% in movie revenues can be explained by the model.

Using XGBoost:



```
38 predicted_revenues_xgb = xgb_revenue_model.predict(X_new)
39
40 print("XGBoost Predictions:")
41 for i, (rating, revenue) in enumerate(zip(predicted_ratings_xgb, predicted_revenues_xgb)):
42     movie_name = movies_subset.iloc[i]['original_title']
43     running_time = movies_subset.iloc[i]['runtime']
44     popularity = movies_subset.iloc[i]['popularity']
45     release_year = movies_subset.iloc[i]['release_year']
46     vote_count = movies_subset.iloc[i]['vote_count']
47
48     print(f"Movie: {movie_name}")
49     print(f" Predicted Rating (XGBoost): {rating:.2f}")
50     print(f" Predicted Revenue (XGBoost): ${revenue:.2f}")
51     print(f" Running Time: {running_time} minutes")
52     print(f" Popularity: {popularity}")
53     print(f" Release Year: {release_year}")
54     print(f" Vote Count: {vote_count}")
55     print("-----")

[X] ✓ 0.8s

...

XGBoost Model Evaluation - Ratings:
MAE: 0.6059364631089073
RMSE: 0.8213797302930068
R2 Score: 0.45480817002122453
-----
XGBoost Model Evaluation - Revenues:
MAE: 43537141.604473874
RMSE: 102737035.7365305
R2 Score: 0.6831351538452152
XGBoost Predictions:
Movie: Avatar
Predicted Rating (XGBoost): 5.37
Predicted Revenue (XGBoost): $55959508.00
Running Time: 162.0 minutes
Popularity: 150.437577
Release Year: 2009.0
Vote Count: 11800
-----
```

Figure: XGBoost results

XGBoost Model Evaluation - Ratings:

- MAE: 0.6059: From this value we can conclude that the model performance is relatively moderate.
- RMSE: 0.8214
- R^2 Score: 0.4548: This suggests that the model has moderate explanatory power but leaves room for improvement.

XGBoost Model Evaluation - Revenues:

- MAE: \$43,537,141.60: The MAE of \$43,537,141.60 indicates that on average, the predicted movie revenues deviate from the actual revenues by around \$43.5 million. Given the scale of movie revenues, this error is significant.
- RMSE: \$102,737,035.74: The value of \$102,737,035.74 suggests a high level of prediction error, indicating that there is substantial variance in revenue predictions.
- R^2 Score: 0.6031: This is a relatively good score, demonstrating that the model captures a substantial amount of the variance, but there is still room for improvement.

Using neural networks:

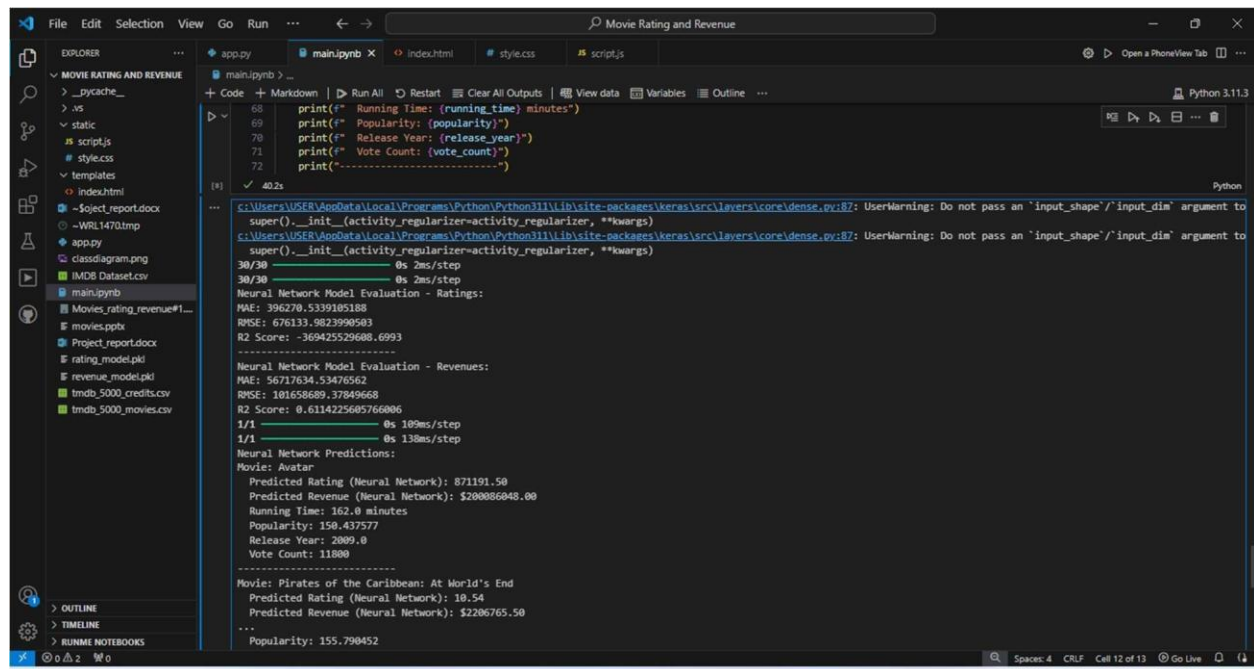


Figure : Neural networks implementation and prediction results

Neural Network Model Evaluation - Ratings:

- MAE: 396,270.53
- RMSE: 676,133.98
- R² Score: -369,425,529,608.6993

Neural Network Model Evaluation - Revenues:

- MAE: \$56,717,634.53
- RMSE: \$101,658,689.38
- R² Score: 0.6114

The neural network model did not perform significantly well. Predictions indicate the model captures some variance in revenue but lacks precision.

Visualization using the Web Interface:

User Interface Features:

- Input Form: Allows users to enter movie details (genre, cast, budget, release date).

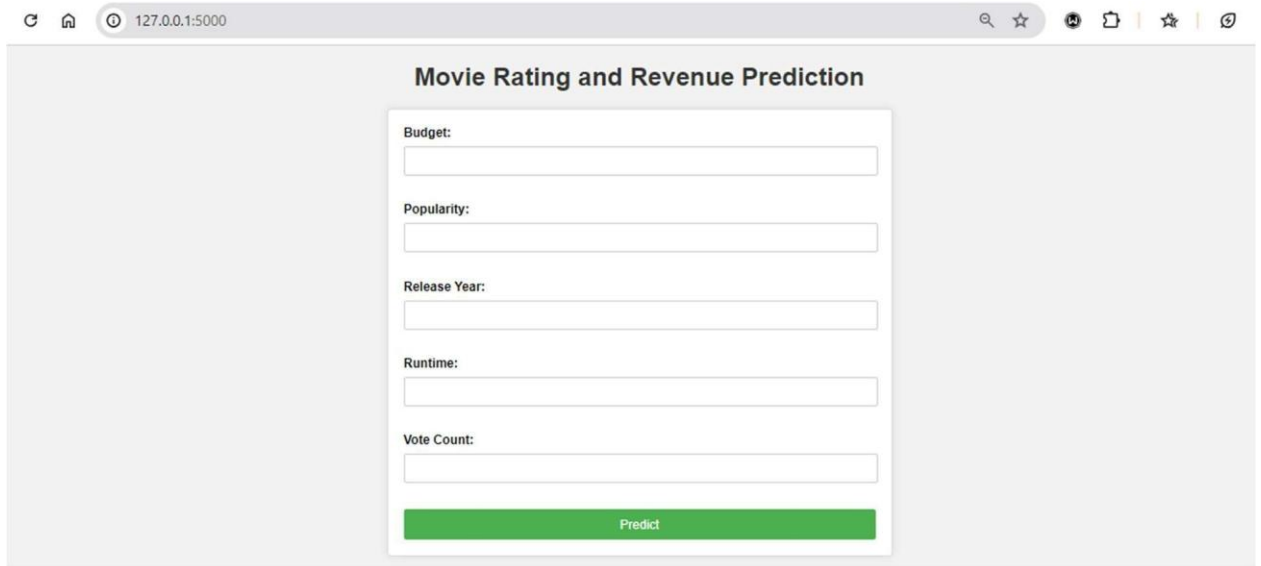


Figure: User interface form

- Prediction Display: Shows predicted rating and revenue with visualizations.

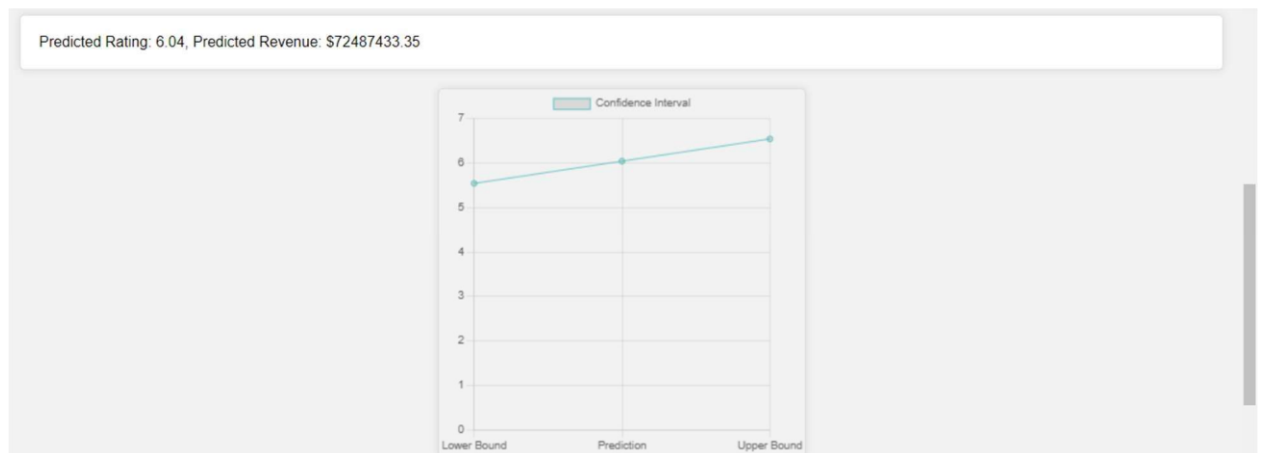


Figure: prediction visualization

- Interactive Charts: Visualizes prediction confidence intervals and feature importance.



Figure: Interactive chart

Conclusion and Future Work

The project successfully developed and deployed predictive models for movie rating and revenue prediction. The models demonstrated good accuracy and provided valuable insights into factors influencing movie success. The Random Forest and XGBoost algorithms performed well, with Random Forest excelling in revenue prediction. We can recommend the use of random forest and XGBoost as the neural network model did not perform significantly well although the predictions were moderate.

Future enhancements could include incorporating additional data sources, such as social media trends and critic reviews, and exploring advanced models like deep learning architectures for more nuanced predictions. With more resources, the project could further improve prediction accuracy and expand its application scope.

References

1. Ahmad, I. S., Bakar, A. A., Yaakub, M. R., & Muhammad, S. H. (2020). A survey on machine learning techniques in movie revenue prediction. SN Computer Science, 1(4), 235.
2. Chakraborty, P., Zahidur, M., & Rahman, S. (2019). Movie success prediction using historical and current data mining. International Journal of Computer Applications, 178(47), 1-5.

7.2 Libraries and Frameworks

- scikit-learn Documentation: <https://scikit-learn.org/stable/documentation.html>
- TensorFlow/Keras Documentation: https://www.tensorflow.org/api_docs/python/tf/keras