IMDB Ratings Model

https://github.com/YashM188/Data-Science-Final-Project

1. Project Definition

1.1 Problem Statement

Stakeholder's Problem: As we know in the film industry, studios and investors are used to making multi million dollar decisions but don't have clear insights into how a film will be received by the audiences. It is for this reason as to why accurately predicting a movie's IMDb rating based on its metadata can help stakeholders make more informed decisions regarding a movie's status. Such metadata can include the genre, cast, director, budget, and runtime. A predictive model could help us guide with investment, casting, and marketing strategies by understanding a film's reception before it even releases. This would aid in reducing the financial risk across numerous film projects. Aspiring Filmmaker's Problem: For directors, writers and producers, it is crucial to understand the factors that influence a movie's IMDb rating. It will allow them to create a roadmap in creating well received films. In this case, it is important to understand which metadata causes a high audience ratings. The model would provide valuable feedback early during the making process which would help them make decisions around casting and genre selection.

1.2 Connection to Course Material

This project will be related to the lectures since it would complete the entire checklist of a data science project. This would include data acquisition to predictive modeling. I will begin by collecting movie metadata from public sources such as the IMDb data. An important lesson learned in class is receiving raw data and then cleaning null or inconsistent values. I will also need to encode categorical variables like genre, cast, and director using one-hot encoding. I also structured the data into a tabular format, similar to the relational databases we discussed in class. There will be defined entities such as budget, runtime, and box office figures. Along with this, I also use Pandas and NumPy for tasks such as filtering and transforming a dataset, a crucial technique that was emphasized in class.

2. Novelty and Importance

2.1 Importance of the Project

This project is important as it goes over a few challenges that exist in the film industry. And these challenges are related to data driven decisions. In a time where budgets and marketing costs are rising, it has become more and more crucial to predict a film's reception before it actually releases. Studios and producers usually rely on past trends but they should rather utilize data backed analysis to predict the success of a film. This can lead to financial savings and better opportunities. I decided to useIMDb ratings since it is one of the most referenced measures of audience reception. I can use it to have a prediction model based on this metadata that can introduce the objectivity needed for the decision needed to be made. It would allow for smarter casting and marketing strategies which would help with return on investment.

2.2 Excitement

I have been a huge fan of movies along with the interest in how different elements (metadata) of a movie, whether it be the cast or director to genre and budget, could contribute to its success. Data from sources such as IMDb is where I often go to explore what's trending and how to rate my own favorites. This project is exciting for me as it would allow me to connect my personal interest in movies to the data science and machine learning skills I have developed throughout this class. I hope to make these insights accessible and easy to understand so I can provide a tool in an area that might be driven by subjective opinions.

Prior related work?
I am aware of a few attempts to predict IMDb ratings but only using limited features such as genre or budget. But there are few to none models that use a much broader data set of metadata and even fewer apply modern regression techniques with numerous feature analysis. I hope that this project builds on these previous ideas but goes further in depth and readability.

2.3 Related Work

There have been data science works in film analytics that have explored the relationship between metadas, such as budget, genre, or star power, to determine a movie's success. However, I feel that many of these public works focus just on basic correlation or use much smaller and outdated datasets. I feel that this does not reflect the current industry dynamics or audience preferences. The goal is to build a model that reduces this gap. It would be crucial to apply ensemble regression techniques, which would be the Random Forests in this case, to a greater variety of reliable metadata. This would offer better insights and add practical value to a wide audience and the film industry.

3. Implementation

3.1 Data Utilization

This dataset included real world information which would be interesting since it would require cleaning and preparing the data but it would provide advantages. This would be that we would work with authentic values that would better reflect the real user ratings, industry trends, and professional metadata. Because of this, the project does not need any synthetic or scraped data since it wouldn't fit the project.

This is supervised learning and the data I used came in the form of a tabular table. The dataset has the following key columns: Series_Title, Runtime, Gross, Genre, Director, Certificate, Meta_score, No_of_Votes, Star1, Start2, Star3, Star4, and the IMDB_Rating. The goal of this project was to predict the IMDb ratings using accurate metadata and not rely on subjective data that would include reviewed text or plot summaries. Using this metadata allowed me to focus the project on structured and realistic analysis with machine learning rather than use natural language processing.

One of the major initial challenges I encountered was dealing with improper formatted columns. For instance, the Runtime column included values like "142 min" as strings. However since it also contains a number, I would have to stripp of the non numeric characters and convert them into floats. Similarly, the Gross column included commas like "28,341,469" which means that it would have to be cleaned and converted into a numerical format that would allow for math related operations.

After the formatting issues were done, I moved on to handling missing values since a few columns had them. Rows with missing IMDB_Rating, for instance, were dropped since these rows could not train the supervised learning model. However, there were other columns also with missing data, such as Gross. In this case, I considered several options before choosing to ignore them depending on the context and their importance.

I also needed to better extract a few patterns so I decided to engineer some new features from the raw data:

- Budget_per_minute: I would calculate this by dividing the Gross by Runtime. This would allow me to see how much money was invested per minute of screen time.
- num_genres: I would extract the Genre column and the value would reflect how many genres each movie was classified under.
- cast_popularity: This would be calculated by counting how many of the top four actor columns (Star1 - Star4) were not null. This would allow me to see how complete the credited cast list was for each film. This will let us see the star power and casting decisions.

I added each of these features to the dataset as a new column to better train the model.

After all this, the next step was preparing the categorical variables for the machine learning model. I decided to apply one-hot encoding to the Genre, Certificate, and Director columns. However, in order to manage the sparsity of the Director field, I grouped all but the top 10 most frequently occurring directors into another category labeled “Other.” This allowed me to make sure that the model would not get overwhelmed by hundreds, and perhaps thousands, of low frequency categories that wouldn’t even contribute much to the predictive performance.

```
df['Runtime'] = df['Runtime'].str.replace(" min", "").astype(float)
df['Gross'] = df['Gross'].str.replace(",", "")
df['Gross'] = pd.to_numeric(df['Gross'], errors='coerce')
df['Budget_per_minute'] = df['Gross'] / df['Runtime']
df['num_genres'] = df['Genre'].apply(lambda x: len(str(x).split(',')))
df['cast_popularity'] = df[['Star1', 'Star2', 'Star3', 'Star4']].notna().sum(axis=1)

df.dropna(subset=["IMDB_Rating"], inplace=True)
```

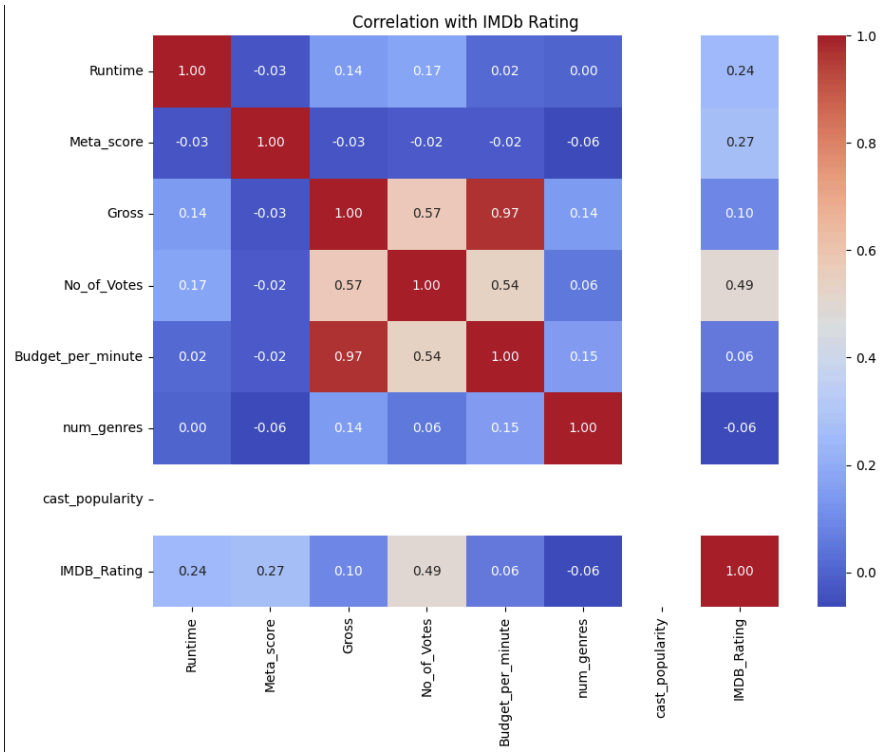
```
top_directors = df['Director'].value_counts().nlargest(10).index
df['Director'] = df['Director'].where(df['Director'].isin(top_directors), 'Other')
df_encoded = pd.get_dummies(df, columns=["Genre", "Certificate", "Director"], drop_first=True)
```

After these steps were done, I now had a cleaned reliable dataset that was ready for training. All the relevant columns were numeric and the missing values had been either dropped or handled. This final dataset had over 20 engineered and encoded features which would help a lot for the regression modeling.

3.2 Models, Techniques, and Algorithms

Now in order to predict the IMDb ratings effectively, I wanted the project to use standard supervised learning techniques that would analyze the features in this dataset. I wanted to identify which variables, either present or engineered, would most contribute to an accurate model.

My first step would involve exploratory data analysis (EDA) to understand the patterns and relationships. Using the python seaborn and matplotlib libraries, I visualized the correlations between numerical features and IMDb ratings. This helped me determine which features were worth keeping and which features might have to be transformed or removed. I saw that features like No_of_Votes, Meta_score, and Runtime were largely correlated with the IMDb ratings while other variables such as Gross and num_genres showed much weaker relationships.



We can see in the heatmap above with No_of_votes having 0.49, the Meta_score being 0.27, and the Runtime being 0.24 indicating the strong positive correlation with the IMDB ratings. This means that this side is moderately predictive. And with variables like Gross being 0.10 and num_genres being -0.06, we can see that it indicates a weak correlation. We can also see how Budget_per_minute and Gross are nearly collinear (0.97) which indicates the redundant information. All of this information will help me when prioritizing the model training.

I then defined the set for training. This included the original numerical values, engineered features (Budget_per_minute, num_genres, cast_popularity), and the one-hot encoded categorical features. All the missing numeric values were filled with 0 since I believed that the impact of these nulls will be minor after the transformation.

```
feature_cols = ['Runtime', 'Meta_score', 'Gross', 'No_of_Votes', 'Budget_per_minute', 'num_genres', 'cast_popularity'] + \
               [col for col in df_encoded.columns if col.startswith(('Genre_', 'Certificate_', 'Director_'))]
X = df_encoded[feature_cols].fillna(0)
y = df_encoded['IMDB_Rating']
```

```
from sklearn.model_selection import train_test_split
```

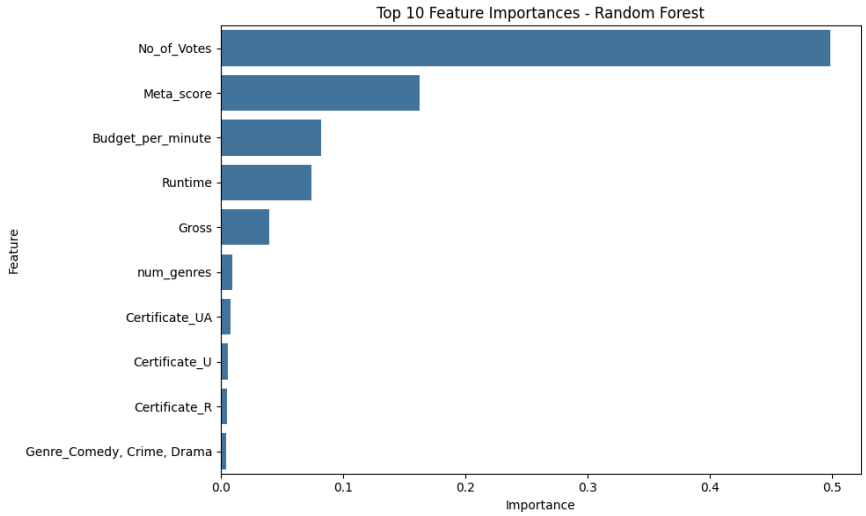
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

After testing the linear regression model which didn't work out well, I ended up selecting the Random Forest Regressor as the final approach. Linear regression, while it was somewhat interpretable, resulted in various unstable predictions. It was often outside the range of IMDb scores (1-10) and even gave negative R^2 scores. On the other hand, the Random Forest model was much more robust since it ended up giving a much more accurate power ($R^2 \approx 0.45$) which gave meaningful insight into the feature importance. So in the end, I chose the Random Forest Regressor since it had the ability to handle mixed feature types. I expect this approach to work much better compared to other existing methods since most other methods relied on genre or budget alone and using the Random Forest would capture the interaction between the numerous variables I was using very well.

As I said, the Random Forest Regressor handles non linear relationships and categorical splits much better. And this is crucial since that matches the mixed type of structure of my dataset as well. It does not assume any specific form for the input or output mapping and ranks feature contributions naturally.

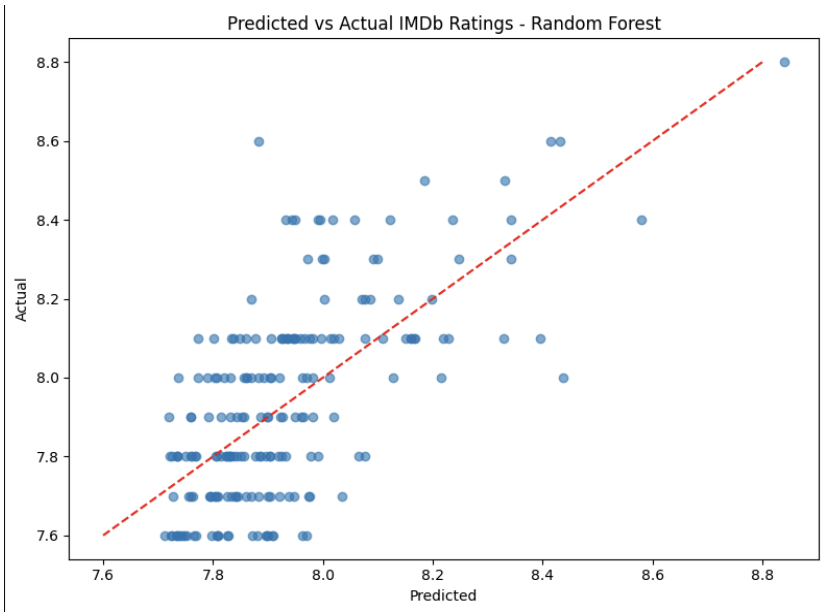
```
model = RandomForestRegressor(n_estimators=200, max_depth=10, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Once the model was trained, I explored the internal mechanics by understanding the feature importances. The top contenders were the No_of_Votes, Meta_score, and Budget_per_minute. This was close to what I originally predicted in that the audience engagement and critical reception would be the main drivers of IMDb ratings.



The last thing to do was generate a predicted vs. actual plot that would allow me to see how well the model matched its outputs to the real world ratings. When I put in the code, I saw that the plot showed a clear diagonal trend with most of the points tightly clustered along the prediction line which showed the model's predictive nature.

```
plt.figure(figsize=(8, 6))
plt.scatter(y_pred, y_test, alpha=0.6)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'r--')
plt.title("Predicted vs Actual IMDb Ratings - Random Forest")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```



After viewing the Random Forest along with the data pipeline, and the targeted feature engineering, I can see how the results showed a model that performed well under real world conditions and showed meaningful patterns behind the movie ratings.

3.3 Experimental Design

As we can see, the goal of the project was to predict the IMDb ratings for movies using reliable metadata and not relying on user reviews or any sort of subjective input. The working hypothesis I had was that the specific measurable features such as the number of votes, the meta scores, and gross revenue would be significantly correlated with IMDb ratings. From the start, I expected that the number of votes and meta scores would be highly predictive. This was since both metrics captured public and critic opinion and these two directly influence the rating of a film. It makes sense that movies with more votes are generally more widely seen and discussed which means that the extreme ratings are stabilized in the final score. Similarly, a film's meta score shows the critic's reviews which might help us understand the execution of the movie.

Another hypothesis I tested was whether financial data, such as gross earnings or the budget, through Budget per minute, could be tied to the quality. However, since the production budget is not always public in available datasets, I used gross revenue divided by runtime as a way to estimate this value. The assumption was that the higher the budget for the films, which would mean better stars, visual effects, and investments, the better it would perform with audiences.

I also explored engineered and structural attributes such as the number of genres to understand if broader themes appealed to the scores. The Cast completeness, through cast popularity, to see whether all the top cast members listed, indicated more prominent productions. And the Directors, which I included through one-hot encoding, for the creator's influence.

I applied correlation analysis in order to remove weak predictors to avoid weak features in the training process and make sure that the model relied on the most informative signals. The experimental design itself was based around a supervised learning framework, where the target variable was IMDB_Rating while all the others were treated as predictors.

The final evaluation would be based on a R^2 score and the Mean Squared Error (MSE). Both metrics that are standard for this model. I also had to rely on feature importance plots and visualizations to understand the model behavior and confirm to see if it matched my hypotheses.

3.4 Key Findings and Results

The results of the model were close to the initial hypotheses I had in terms of the performance and the variables that turned out to be the most predictive. Through the correlation analysis and model evaluation, I saw which features had the most influence in determining a film's IMDb rating. This validated that this modeling approach could generalize well across unseen data.

After training the Random Forest Regressor on an 80/20 train test split, the MSE was around 0.0354 with R^2 around 0.4478. This shows that the model was able to capture around 45% of the variance in IMDb ratings. Though this means that the model doesn't have that much of a high accuracy, it is still a strong result since there were limitations of the input data with the lack of subjective data like text reviews or general sentiments.

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Random Forest MSE: {mse:.4f}")
print(f"Random Forest R² Score: {r2:.4f}")
```

```
Random Forest MSE: 0.0354
Random Forest R² Score: 0.4478
```

In terms of feature importance, the model confirmed what I had early assumed. The most significant predictor was the No_of_Votes which again brought back the idea that public engagement strongly correlated with the rating scores given. It was then followed by Meta_score which meant that critics' opinions mattered. and Budget_per_minute which was to

understand the financial side of the study. Runtime and gross earnings also played a role but to a lesser degree. At the same time, some of the less correlated features such as the number of genres did not dominate the model's decisions.

3.5 Expectations Vs Results

I actually expected the model, initially, to achieve a much higher accuracy given the structured data of the IMDb metadata. But the actual results revealed it was far more complex than it seemed. The Random Forest Regressor achieved a R^2 score of around 0.45 which means that it was just under half of the variance in IMDb ratings. Due to the data, I thought of a R^2 score closer to 0.6 or above. It was done under the assumption that features like vote count, meta score, and runtime would be strongly predictive. However, the actual results ended up indicating that audience ratings might be just as influenced by subjective factors, such as public sentiment, marketing, release timing, and cultural trends as well. These sort of factors can not be easily captured through metadata alone. This gap between the expected and actual performance showed to me the limitations of using just structured data and the need for complex methods.

3.6 Advantages and Limitations

I think one of the advantages in this approach was that it relied a lot on real world data from IMDb rather than having to rely on synthetic information. This allowed me to add more credibility to the model since the insights were more related to actual audience and industry behavior. By using column metadata such as votes, critic scores, and genre information, I ended up avoiding the subjectivity that comes with text based reviews or sentiment analysis, though as noted above, was still important. But regardless, there were reasonable predictions. Another advantage in this approach was the simplicity. The model is built entirely on features that can be easily obtained for any movie. It means that it does not require any advanced scraping or third party datasets. This allowed me to make it more reproducible and easy to expand on. And using feature engineering on several columns allowed me to extract more information from the existing data.

However, the project also had several limitations that created some drawbacks. It can be considered that the IMDb ratings are inherently subjective which means that the dataset did not have as many human elements that could influence them. For instance, I did not include text reviews or audience demographics or the release timings. All of these might have been likely to influence the ratings but I couldn't since they were outside the scope of the dataset. And as explained before, while the model performed well (R^2 around 0.45), it still meant that more than half of the variance in the IMDb ratings remained unsolved. While I do believe that some of this may be due to missing variables, I believe that it also shows the challenges of modeling human opinion using structured data alone.

4. Challenges and Bottlenecks

One of the challenges that I faced was understanding the limitations of the IMDb dataset itself. Although it was organized, I found that many features I expected to be reliably populated such as gross earnings or meta scores were actually missing a large portion of entries. This meant that I would have to filter the dataset which meant that I would risk reducing the volume of data that was available for training. For instance, when calculating Budget_per_minute, I had to remove rows with missing gross or runtime values. And for the cast like cast_popularity, I had to handle null actor names through removal.

Feature engineering also became a time consuming process. Creating new columns like num_genres, Budget_per_minute, and categorical encodings for genre and director required me multiple trials and errors until it was done. For instance, director names had such high cardinality that I had to reduce them into just the top 10 and group the rest under an “Other” label. Another bottleneck was the visualization. Although matplotlib and seaborn were powerful tools, making sure that the plots showed useful insights and did not overwhelm the report also took trials and errors. For instance, an early version of the correlation heatmap had included too many features which made it unreadable. I had to test multiple combinations of features to balance the clarity.

5. Future Work

I might work on this project even further and I could move it to several directions which could increase its impact. One way to do this would be to add more to the dataset with additional features such as award data (Oscars) or sentiment scores from user reviews. This would allow me to better capture the subjective factors that influence audience perception. Something that I feel like will be needed for a higher R^2 score. I also want to add in Natural Language Processing (NLP) techniques to analyze the reviews which would also allow me to generate features that would reflect on the content quality. From the modeling side, I have been researching and might use even more advanced algorithms like XGBoost or deep learning approaches that could improve prediction accuracy. This would happen by understanding more complex patterns in the data. And for fun, turning this project into an interactive web application would allow others to also input movie metadata and receive the IMDb rating prediction based on the model.