# IMDB Rating Prediction

## Abstract

## Introduction and Data Description

In this work we are using the IMDB database to predict the average ratings of a content. The data is available here.

## Technical Approach

We split the technical work into 3 components:

1. Data wrangling, can be found in this notebook.
2. Exploratory data analysis, can be found in this notebook.
3. ML Modelling, can be found in this notebook.

### Main Hypothesis and Data Wrangling

Our main hypothesis is that the average rating of a content depends primarily on 3 factors:

1. The nature of the content, such as the type (tv show vs movie), the genre (horror vs comedy), etc.
2. The production value of the content. Since we don’t have information on the production budget of the content, we can infer the production value from the production crew information (e.g. director and writer).
3. For TV show, the performance of the past episode and the longevity of the show.

Building on top of these hypothesis, we calculated the following additional features as the predictors.

|  |  |  |
| --- | --- | --- |
| **Column** | **Type of data** | **Description** |
| parentTconst | String | Alphanumeric identifier of the parent TV Series |
| directors\_writers | String | Text data containing the identifier of the director(s) and writer(s) of the given title |
| max\_episodeNumber | Numerical | Maximum number of episodeNumber of the given title |
| max\_seasonNumber | Numerical | Maximum number of season of the given title |
| prefix 'unique\_' | Dummy indicator | Number of unique {language, ordering, region, title, types} the given title has |
| prefix 'listed\_' | Dummy indicator | Number of listed {actor, actress, archive\_footage, archive\_sound, cinematographer, composer, director, editor, producer, production\_designer, self, writer,total} for the given title |
| prefix 'type\_' | Dummy indicator | One-hot encoding whether the given title is of type {short, tv series, tv episode, movie, video, tv special, tv movie, video game, tv short, tv mini-series} |
| prefix 'genre\_' | Dummy indicator | One-hot encoding whether the given title is considered to be of {Adventure, Drama,Short, Comedy, Family,Game-Show, Music, Documentary, War ,Reality-TV, Action, Animation, History,Sci-Fi ,Fantasy, Horror,Mystery,Adult,Romance,Crime, Film-Noir, no\_genre, Talk-Show,Western, Thriller, Sport, Biography, Musical ,News} |
| total\_genres | Numerical | Total number of genre the given title is considered |
| number\_past\_episode | Numerical | For TV episode, the number of past episode before the given title |
| average\_show\_past\_rating | Numerical | For TV episode, the average rating of the past episodes before the given title |
| director\_median\_past\_rating | Numerical | The median rating of the past contents that the directors has worked on before directing the given title |
| director\_number\_past\_movies | Numerical | The number of past contents that the directors has worked on before directing the given title |

### 3.2 Exploratory Data Analysis

Our exploratory data analysis identified the following noteworthy insights:

1. TV Episode is the most common title type (47%). It also has the highest average rating compared to other title types.
2. As a given title can be classified into more than 1 genre, we performed difference in means analysis to study the relationship between a title genre and its rating. For a given genre, for example horror, we calculate the difference in means of rating between titles classified as horror (genre\_Horror ==1) and titles classified as non-horror (genre\_Horror ==0). We found that documentary, history, and crime are the genres with the most positive difference in average rating. We also found that horror, thriller, and sdult are the genres with most negative difference in average rating.
3. We found that ‘average\_show\_past\_rating’ and ‘director\_median\_past\_rating’ as the features with the highest correlation average rating.

### ML Modelling

Our ML modelling work consist of the following steps:

1. We split the data into training and testing set
2. We performed numerical data pre-processing on the training set, which consists of the following:
   1. We removed sparse features.
   2. We removed features with zero variance.
   3. We removed highly correlated features.
   4. We performed k-Nearest Neighbor (k-NN) imputation to impute missing values.
   5. We performed Recursive Feature Elimination (RFE) to select the best numerical features. We first performed RFE with Cross Validation (RFECV) to study the relationship between number of features on model performance. We then used the insight from RFECV to decide the number of features to be selected by the RFE procedure.
3. We performed custom tokenization and dimensionality reduction for the sparse text data (‘parentTconst’ and ‘directors\_writers’). We leveraged count vectorization and truncated SVD method. As the standard sklearn implementation does not support the treatment of missing values, we wrote our own custom implementation for this step.
4. We performed standard scaling on the numerical features and the tokenized sparse text features.
5. We trained the ML model on the pre-processed data and tuned the parameters using randomized search.

## Modelling Results

There are 4 models that we used in this work:

1. Baseline Model: Naïve prediction model which uses the mean of the average rating as the prediction.
2. Linear Model: Elastic Net model using on only the kNN-imputed numerical features.
3. XGB Model: XGBoost model using only the unimputed numerical features.
4. XGB + Tokenization Model: XGBoost model using the unimputed numerical features and the sparse text features.

The performance of the model is given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Training Performance** | **Testing Performance** | | |
| **R2 - Score** | **Mean Absolute Error** | **Mean Squared Error** | **R2 - Score** |
| Baseline Model | 0.000 | 1.074 | 1.933 | 0.000 |
| Linear Model | 0.223 | 0.917 | 1.512 | 0.218 |
| XGB Model | 0.347 | 0.821 | 1.271 | 0.342 |
| XGB + Tokenization | 0.365 | 0.792 | 1.215 | 0.371 |

## Potential Future Works