Performance prediction of movies in India based on quantitative parameters

Adittya Gupta [1] Gautam Sharma [1] Sparsh Mittal [1] Yash Raj Singh [1]

Abstract

With a market size of Rs 15,000 crore and 3,000 films produced annually(Wikipedia contributors, 2024) in India, predicting a movie's performance is crucial for stakeholders to tailor content and ensure profitability. This project aims to use machine learning to forecast a movie's success in the Indian market, aiding industry players in optimizing their offerings.

1. Introduction

The current Cinema industry cannot efficiently cater to the needs of its audience as while creating a movie, the makers' creativity doesn't take into account the market trends, which are reflective of how a movie would perform at the box office.

The entire industry's success is based on hit and trial or luck, which should not be ideal, given the industry's size and influence over the Indian GDP. If the makers of a film could see how their cinema would perform at the box office at the current time, then it would be a win-win situation for both the makers and the audience, as the audience would get a good (intended) experience in the theatres and the producers would be able to book good profits off the movie.

The motivation behind designing a movie performance predictor stems from the observation that certain demographics within specific geographic regions share similar entertainment preferences. Additionally, we hypothesize a correlation between a movie's release year and its success, reflecting the evolving tastes of cinema and its audience over different eras. For instance, the popularity of Bollywood romantic movies was higher in the early 2000s compared to the 2020s. Our goal is to develop a model that can capture these trends, while also considering factors such as the movie crew, to provide a comprehensive prediction of a movie's performance in the Indian market. We will be considering the entire Indian movie dataset for this purpose.

[1]Department of Computer Science and Engineering, Indian Institute of Technology, Guwahati, India. Correspondence to: Adittya Gupta <g.adittya@iitg.ac.in>, Gautam Sharma <gautam.sharma@iitg.ac.in>, Sparsh Mittal <m.sparsh@iitg.ac.in>, Yash Raj Singh <yash.singh@iitg.ac.in>.

Course Project, Instructor: Prof. Ashish Anand

2. Literature Review

Apala et al., had previously applied K-means on a dataset generated using social media for the given task (Apala et al., 2013), but they employed a pre-trained model, in the form of Weka to apply K-means. Abidi et al., had applied various models using a relatively small training set with very less predictors to solve the above problem (Abidi et al., 2020). The former group found the **Generalized Linear Model (GLM)** to be the best performer among all the methods used. Sharda and Delen applied neural networks with almost exhaustive inputs to solve the problem (Sharda & Delen, 2006). Earlier Sawhney and Eliashberg (Sawhney & Eliashberg, 1996) used the early box office collections of a movie to predict its lifetime performance at the box office.

2.1. Proposed Novelties

Our work extends on the previously done works, as we employ a bigger dataset, sensitive to Indian lingual cinema. We also plan to apply NLP on the movie's storyline, fetched from the dataset to further classify the movies.

3. Dataset Generation

We generate the dataset through web scrapping the IMDb website. We have included the following features in our dataset: Time of release, Genre, Cast, Directors, Production Company, Storyline, Original Language, Country of Origin, and Duration.

The code for dataset generation is provided here.

The raw data that was directly scraped contained 10,004 rows with 12 columns which acted as the features/predictors for each data point.

A histogram of the dataset for the IMDb rating is given in Figure 1.

4. Data Cleaning

We discovered the number of null values in each column upon generating the dataset.

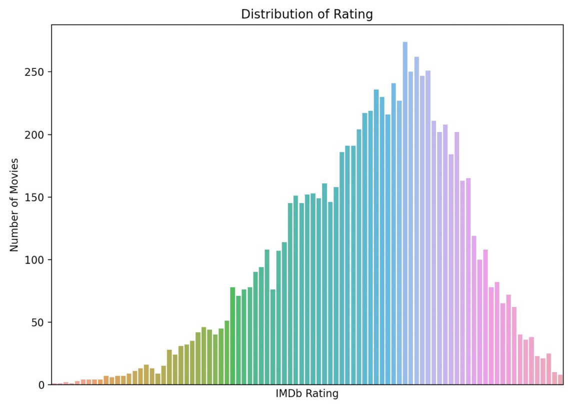

Figure 1. Histogram of the custom dataset for the IMDb rating

1. NaN values in Movie Name is 84 (0.84%)
2. NaN values in IMDb Rating is 1333 (13.32%)
3. **NaN values in Popularity is 9817 (98.13%)**
4. NaN values in Release Date is 3755 (37.53%)
5. **NaN values in Box Office is 7434 (74.31%)**
6. NaN values in Runtime is 2305 (23.04%)
7. Empty values in Genre is 337 (3.37%)
8. Empty values in Director is 2287 (22.86%)
9. Empty values in Cast is 818 (8.18%)
10. **Empty values in Production Company is 6581 (65.78%)**
11. NaN values in Country is 4006 (40.04%)
12. Empty values in Language is 284 (2.84%)

1. Since the popularity, box office, production company column is mostly empty, we decided to drop it.
2. Since the movie name can't have any significant effect on the model. We dropped that column. We also dropped Country because we are considering only Indian movies so it would be same for all
3. We also dropped those rows whose IMDb rating is not present.
4. We decided to drop the **Release Date** column because the exact release date won't help us in any way. Instead we used extracted a new column called **Release year** which was used in the prediction
5. Also the **Language** column was dropped and in place of it column **isHindi** was introduced which tells if movie contains Hindi as one of its languages in it.

4.1. Finding Outliers

Regarding Runtime, we identified **16** outliers by creating a z-score threshold of [-3, 3] for the non-outliers. Z-score is the amount of deviation a quantity has from the mean of the distribution and is calculated as $z - score = \frac{x-\mu}{\sigma}$, where μ is the mean of the distribution and σ is the standard deviation of the distribution.

5. Data Analysis

We analysed the dataset collected to extract the characteristics of the data in the dataset.

5.1. Top Languages

Distribution of the languages (figure 2) reveals that the top five movie languages were **Hindi**, followed by **Tamil**, **Telugu**, **Malayalam** and **English**.

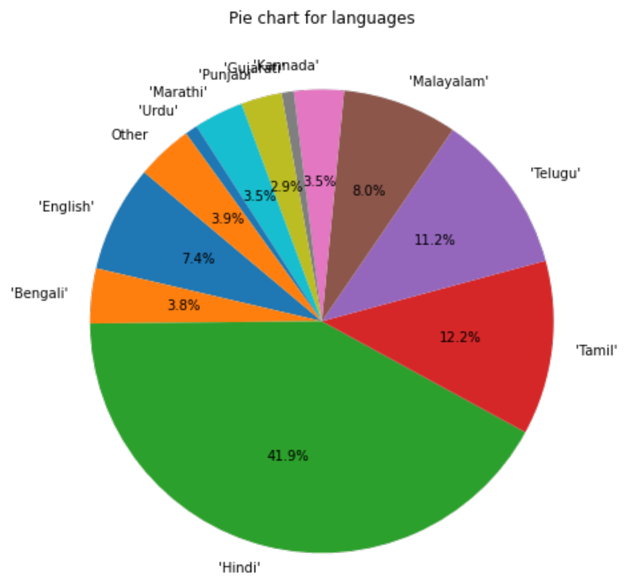

Figure 2. Distribution of the languages in which the movie was originally made

5.2. Number of movies released every year

We calculated the number of movies released every year (figure 3), and we can see that the number of films produced each year is growing exponentially.

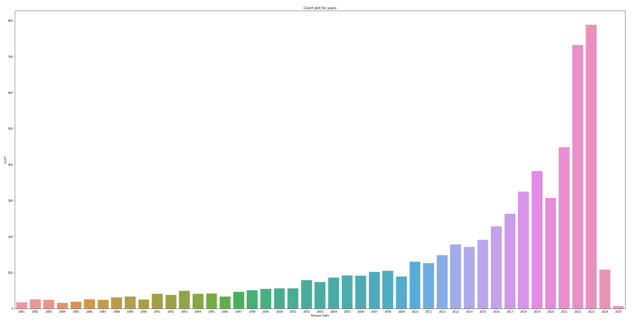

Figure 3. Count plot of the number of movies released every year

5.3. Average Rating of movies

We also calculated the average rating of movies released in a year (figure 4). There is no trend in the data, but on average, one can say that the quality of movies over the period has decreased drastically.

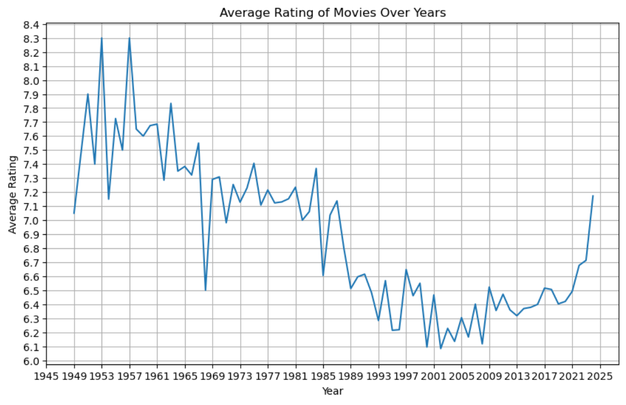

Figure 4. Average rating of movies released in a year

5.4. Top 10 years by rating

We also calculated the top 10 years by average rating of movies released in a year (figure 5). There is no trend in the data, but on average, one can say that the quality of movies over the period has decreased drastically.

5.5. Frequency of movie ratings

We also calculated the frequency of movie ratings being given (figure 6), and found out that most of the movies have been given a rating in the range of [6, 8].

5.6. Popularity of genres

We also found out the popularity of various genres in the dataset (figure 7), it revealed that **Thriller** genre was the most popular, followed by **War**, **Family**, **Horror** and **Crime**.

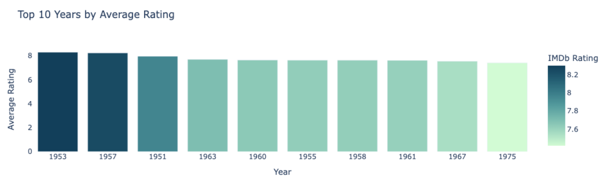

Figure 5. Top 10 years by average rating

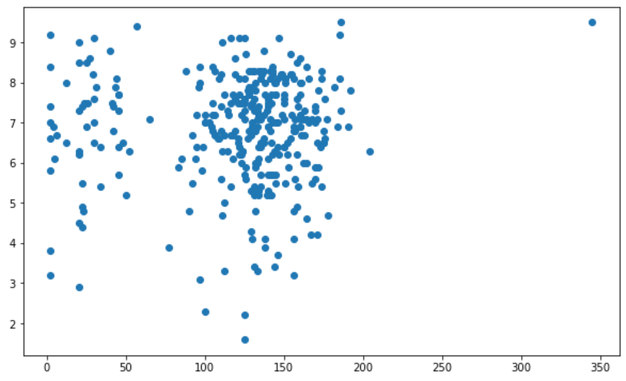

Figure 6. Scatter plot of film rating versus its frequency in the dataset

5.7. Average rating accross years for top 3 genres

We studied the trend of rating for the top 3 genres(by number of occurences) accross years (figure 8). This graph can be viewed as the taste of indian population accross various years.

5.8. Finding correlations

We also tried to find the correlation between the Popularity of a movie, Its IMDb Rating and its Runtime in minutes and found out a Correlation Matrix (table 1).

	Popularity	IMDb Rating	Runtime
Popularity	1.000000	0.199417	-0.340967
IMDb Rating	0.199417	1.000000	0.046358
Runtime	-0.340967	0.046358	1.000000

Table 1. Correlation between Popularity of a movie, its IMDb rating and its runtime in minutes

We also tried finding any correlation between the box office earnings of a movie and its IMDb rating (table 2), but found out that there is little to no correlation, as a movie's rating

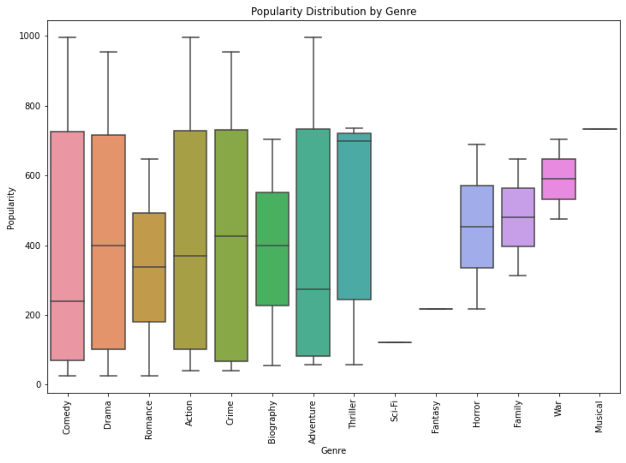

Figure 7. Trend Accross years for different genres

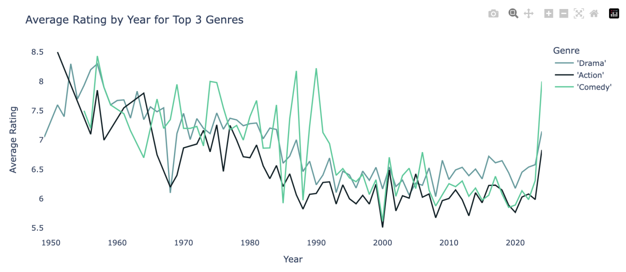

Figure 8. Popularity distribution by genre

depends on its storyline, cast and crew, not it's financial collections on the box office.

	Box Office	IMDb Rating
Box Office	1.000000	0.090164
IMDb Rating	0.090164	1.000000

Table 2. Correlation between Box office collection of a movie and its IMDb rating

6. Feature Engineering

Through common sense and prior knowledge we identified new features from the original ones and used them instead to make our Machine learning model learn better and faster. These are as follows:

- **Year of Release:** The original column date of release though containing more information than what we are turning it into but still we can assume that the preferences of common audience doesn't change drastically over a year and we only consider the year part of the release. This is also done because it is easier to use year than the exact date in the model
- isHindi: The dataset consists of movie with more than 100 languagages with many movies with unique languages. Since Hindi is the most common language in these movies it is better to consider it different from movies which doesn't contain Hindi at all.
- **Genre_mean_rating:** For each genre we calculate the average IMDb rating which signify the importance of that genre so it can be learned by the model. This is because the name of genre is not important but it's avg rating is.
- **Director_mean_rating:** For each list of Directors we take the first director and calculate the average IMDb rating which signify the importance of that director so it can be learned by the model.
- **Cast_mean_rating:** Similar thing as above is done for the cast column also

7. Modelling of problem

We modelled the problem as follows:

1. We defined **four** classes of classification based on IMDb rating, spanning from [0, 3), [3, 5), [5, 7), and [7, 10].
2. We used **six** models out of which we ourselves implemented four of them while 2 of them were from the sk-learn library. The models that we implemented were **Logistic Regression**, **Support Vector Machine**, **Gaussian Naive Bayes**, **Decision tree classifier**. While we used **Random forest Classifier** and **XGBoost Classifier** from the sk-learn library. One can see the accuracy on testing the models at 3
3. We used **six** features to train the models, these were, **Runtime of movies**, **isHindi**, **Year of Release**, **Genre_mean_rating**, **Director_mean_rating**, **Cast_mean_rating**.

Table 3. Model Accuracies

Model	Accuracy (%)
Logistic Regression	61.29
SVM	66.63
Gaussian Naive Bayes	71.49
xgBoost classifier	86.49
Random Forest classifier	88.24
Decision Tree classifier	89.59

7.1. Logistic Regression

7.1.1. Overview of the Algorithm

The core idea behind Logistic Regression is to model the probability that a given input belongs to a certain class. It does this by fitting a logistic function (also known as the sigmoid function) to the input features. The logistic function maps any input value to a value between 0 and 1, which can be interpreted as the probability of belonging to a particular class.

Given a set of input features $X = \{x_1, x_2, ..., x_n\}$, where x_i represents the i-th feature, and weights $\theta = \{\theta_0, \theta_1, ..., \theta_n\}$, the logistic regression model computes the logit z as:

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$$

The logit z is then passed through the logistic function (also known as the sigmoid function) to obtain a probability value between 0 and 1:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

where e is the base of the natural logarithm.

The logistic function converts the logit z into a probability $\hat{y}$, where:

$$\hat{y} = P(y = 1|X; \theta) = \sigma(z)$$

Here, $\hat{y}$ represents the predicted probability that the output y is 1 given the input features X and model parameters θ.

To train the logistic regression model, we use a loss function called cross-entropy, which measures the difference between the predicted probabilities and the actual class labels.

Given a set of training examples $\{(X^{(i)}, y^{(i)})\}$, where $X^{(i)}$ is the i-th input feature vector and $y^{(i)}$ is the corresponding class label (0 or 1), the cross-entropy loss $J(\theta)$ is defined as:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

where m is the number of training examples.

The goal of training the logistic regression model is to minimize the cross-entropy loss function $J(\theta)$ with respect to the model parameters θ. This is typically done using optimization techniques such as gradient descent.

The gradients of the loss function with respect to the model parameters are computed using the chain rule of calculus, and the parameters are updated iteratively in the direction that minimizes the loss.

Regularization techniques such as L1 (Lasso) and L2 (Ridge) regularization can be applied to logistic regression to prevent overfitting. These techniques add penalty terms to the loss function that discourage large parameter values.

Once the model is trained, predictions are made by passing new input features through the logistic regression model and converting the resulting logit into a probability using the logistic function. Typically, a threshold (e.g., 0.5) is applied to the predicted probability to determine the class label.

7.1.2. Model Training

The training was done using the following steps:

1. Convert the target labels into one-hot encoded format to handle multiclass classification.
2. Applying gradient descent and regularization to learn the biases and weights

7.1.3. Prediction

Once trained, the Logistic Regression model can be used to predict new data. Given a set of input features, the model computes the probability of each class using the learned parameters and assigns the input to the class with the highest probability.

7.1.4. Results

The model, which was implemented from scratch, achieved 62% accuracy on the testing dataset.

Algorithm 1 Predict function for Logistic Regression

```
Require: Test data features $X$, target labels $y$
1: Initialize the weights and the bias
2: for each iteration $i$ in $iterations$ do
3:     Predict the values of y using the updated parameters
       and compute the error.
4:     Compute the gradients.
5:     Add regularization, L1 or L2 depending on the inputs
       given by the user.
6:     Update parameters using learning rate.
7: end for
```

Algorithm 2 Predict function for Decision Tree classifier

```
Require: Testing data features $X$
1: for each training data $x$ in training dataset $X$ do
2:     Calculate the z value and find the softmax value of z.
3:     The class of $x$ will be this value.
4: end for
```

7.2. Support Vector Machine

SVM, or Support Vector Machine, is a powerful supervised machine learning algorithm used for classification and regression tasks.

Here are the key concepts and components of SVM:

7.2.1. Support Vectors

- In SVM, the algorithm tries to find the hyperplane that best separates the data into different classes. Support vectors are the data points closest to the hyperplane and influence the position and orientation of the hyperplane.
- The hyperplane is chosen so as to maximize the margin, which is the distance between the hyperplane and the nearest data point from either class.

7.2.2. Hyperplane

- In a two-dimensional space, a hyperplane is a line that linearly separates the classes. In higher-dimensional spaces, it becomes a hyperplane.
- However, in this problem, the One-vs Rest method is used to define the Hyperplane, since it is a multi-class problem.
  In this approach, a separate binary classifier is trained for each class, where each classifier distinguishes between one class and all other classes combined. Once all binary classifiers are trained, the class with the highest decision function score among all classifiers is predicted for a given input. The hyperplane for each binary classifier is defined similarly to binary classification, aiming to maximize the margin between the class it represents and all other classes.

7.2.3. C Parameter

- The C parameter controls the trade-off between maximizing the margin and minimizing the classification error.
- A smaller C value leads to a larger margin but may misclassify some points, while a larger C value may classify all points correctly but result in a smaller margin.

7.2.4. Regularization Parameter

- SVM uses a regularization parameter (C or λ) to prevent overfitting by penalizing large values of the weight vector.
- It helps in finding a balance between minimizing the classification error and keeping the model simple.

7.2.5. Optimization Objective

- The goal of SVM is to find the optimal hyperplane that separates the classes while maximizing the margin and minimizing the classification error.
- This is typically formulated as a convex optimization problem that can be solved using techniques like gradient descent or quadratic programming.

7.2.6. Pseudo Code

The algorithm's pseudocode is given in 3 (Fit function), and 4 (Predict function).

Algorithm 3 Fit function for Multiclass SVM

```
1: for each $class\_idx$ in range($n\_classes$) do
2:     $binary\_y \leftarrow$ Create binary labels for $class\_idx$
3:     Initialize SVM classifier for binary classification
4:     Train the binary SVM classifier on $X$ and $binary\_y$
5:     Store the trained classifier
6: end for
```

Algorithm 4 Predict function for Multiclass SVM

```
1: Initialize an empty array $class\_scores$ of shape $(len(X), n\_classes)$
2: for each classifier in the stored classifiers do
3:     Compute decision function scores for $X$ using the classifier
4:     Store the scores in the corresponding column of $class\_scores$
5: end for
6: return the class index with the highest score for each sample in $X$
```

7.2.7. Results

The model, which was implemented from scratch, achieved 66% accuracy on the testing dataset.

7.3. Gaussian Naive Bayes

Gaussian Naive Bayes is a classification algorithm based on Bayes' theorem with the assumption of independence between predictors (features) within the dataset. It is particularly useful for classification tasks when dealing with continuous features that can be assumed to follow a Gaussian (normal) distribution.

Let us denote the data points by $\mathbf{X}$ and their corresponding labels by $\mathbf{y}$ where $\mathbf{X}$ is a $n * m$ matrix. On the other hand $\mathbf{y}$ is a vector in $\mathbb{R}^n$. Here n is the number of data points and m is the number of predictors.

Now for a given $\mathbf{X}$ we seek $\arg\max(\mathbb{P}(y_i|\mathbf{X}))$ where i is over the number of classes k.

Now by Bayes' Theorem we have

$$\mathbb{P}(y_i|\mathbf{X}) \propto \mathbb{P}(\mathbf{X}|y_i) * \mathbb{P}(y_i) \; \forall \, i = 1 \dots k$$

Since probabilities can be very small we use log instead to prevent underflow errors

$$\log(\mathbb{P}(y_i|\mathbf{X})) \propto \log(\mathbb{P}(\mathbf{X}|y_i)) + \log(\mathbb{P}(y_i))$$

In Gaussian Naive Bayes we assume that each of these probability distributions are identical to the Normal Distribution with parameters μ and σ to be estimated. The PDF for Normal distribution is given by:

$$\mathbb{P}(x) = \frac{e^{\frac{-(x-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

Now for estimating $\mathbb{P}(y_i)$ which are prior probabilities of these classes we can directly write

$$\mathbb{P}(y_i) = \frac{\text{Number of samples with class i}}{\text{Total number of samples}}$$

For estimating the parameters of $\mathbb{P}(\mathbf{X}|y_i)$ we calculate the mean and variance by only finding the mean and variances of the data points whose label is i. This result is obtained by Maximum Likelihood Estimation(MLE) treatment of this problem.

We also added a smoothing value so as to ensure that these variances don't become zero due to underflow.

Algorithm 5 Fit function for Gaussian Naive Bayes classifier

```
Require: Training data features $X$, target labels $y$, smoothing value $smoothing\_value$
Ensure: Mean, variance, and prior probabilities for each class
 1: Initialize empty arrays for mean, variance, and prior probabilities
 2: Initialize classes as unique labels in $y$
 3: Initialize number of classes $n\_classes$
 4: Set $\epsilon = smoothing\_value \times$ maximum variance across features in $X$
 5: for each class $c$ in $classes$ do
 6:     Extract features $X_c$ corresponding to class $c$
 7:     Compute mean and variance of $X_c$ along each feature dimension
 8:     Add $\epsilon$ to variance to avoid division by zero
 9:     Compute prior probability of class $c$
10: end for
```

Algorithm 6 Predict function for Gaussian Naive Bayes classifier

```
Require: Test data features $X$
Ensure: Maximum probabilitic label for these features
 1: Initialize empty arrays posterior probabilities
 2: for each class $c$ in $classes$ do
 3:     Extract prior probability for this class
 4:     Compute the probability $\mathbb{P}(\mathbf{X}|y_c)$ using the normal pdf formula and plugging $\mu$ and $\sigma$ as calculated from earlier fit function
 5:     Add the sum of logarithm of these probs and push to posterior array
 6: end for
 7: Return the index of maimum value in posterior array
```

7.4. Decision Tree Classifier

A Decision Tree Classifier is a classification algorithm that creates a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Decision trees are constructed recursively by splitting the dataset based on feature values, aiming to create homogeneous subsets in terms of the target variable. This is done by choosing the best feature from the dataset to split it into two subsets (can be more) to maximize the homogeneity of the subsets regarding the target variable. The criterion that we have used for feature selection is Gini Impurity.

Gini Impurity : Gini impurity is a measure of how often a randomly chosen element from a set would be incorrectly classified if it were randomly labeled according to the distribution of labels in the subset. The Gini impurity of a node t in a decision tree is calculated as follows:

$$G(t) = 1 - \sum_{i=1}^{c} p(i/t)^2$$

where c is the number of classes and p(i/t) is the proportion of samples in node t that belong to class i. The goal is to choose the feature with the minimum sum of weighted gini impurity in the subsets it divides.

$$G_W = N_L/N_t * G_L(t) + N_R/N_t * G_R(t)$$

Here N_L and N_R are the number of samples in the left child and right child node and N_t is the total number of samples in the parent node.

In the final decision tree, at each internal node, there is a decision to be made. Leading to the leaf nodes which are labelled with classes. However, this model may contain high variance leading to overfitting. To tackle this, we have limited the maximum depth of the tree which helps reduce the variance and allows the model to perform better on unseen data.

The model, which was implemented from scratch, achieved 89.5% accuracy on the testing dataset.

Algorithm 7 Fit function for Decision Tree classifier

```
Require: Training data features $X$, target labels $y$
 1: If max depth is reached or all samples are of the same class then label this node with class with maximum samples. This will be the leaf node.
 2: for each feature $f$ in $features$ do
 3:     for each threshold point $tp$ of the feature $f$ do
 4:         Split the dataset into two subsets based on the threshold.
 5:         Calculate the weighted Gini Impurity.
 6:         Store the the feature, threshold and gini impurity if it is the smallest till now.
 7:     end for
 8: end for
 9: Separate the samples based on the feature and threshold corresponding to the minimum gini impurity.
10: Recursively call this function to grow the left and the right subtree of the decision tree.
```

Algorithm 8 Predict function for Decision Tree classifier

```
Require: Testing data features $X$
 1: for each training data $x$ in training dataset $X$ do
 2:     Traverse through the decision tree until leaf node is not reached.
 3:     The class of the leaf node will be the predicted class.
 4: end for
```

7.5. XGBoost

7.6. Random Forest Classifier

8. Future Extension

We plan to implement the following in the future:

1. Use Natural Language processing to extract the information from the reviews of the movie to get the information regarding the general opionion of the movie in the public. This can largely determine IMDb rating.

2. We can also scrape more data from the IMDb website by buying there premium plan. This would allow us to get more important information like Director's rank, cast rank, popularity, box office collection, number of votes etc. which can have a big impact on the accuracy

3. We can improve the models by using more advanced machine learning techniques like Hyperparameter tuning etc. which can have a huge impact on the final accuracy of the model.

References

Abidi, S. M. R., Xu, Y., Ni, J., Wang, X., and Zhang, W. Popularity prediction of movies: from statistical modeling to machine learning techniques. *Multimedia Tools and Applications*, 79(47):35583–35617, Dec 2020. ISSN 1573-7721. doi: 10.1007/s11042-019-08546-5. URL https://doi.org/10.1007/s11042-019-08546-5.

Apala, K. R., Jose, M., Motnam, S., Chan, C.-C., Liszka, K. J., and de Gregorio, F. Prediction of movies box office performance using social media. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pp. 1209–1214, 2013. doi: 10.1145/2492517.2500232.

Sawhney, M. S. and Eliashberg, J. A parsimonious model for forecasting gross box-office revenues of motion pictures. *Marketing Science*, 15(2):113–131, 2024/02/22/ 1996. URL http://www.jstor.org/stable/184189. Full publication date: 1996.

Sharda, R. and Delen, D. Predicting box-office success of motion pictures with neural networks. *Expert Systems with Applications*, 30(2):243–254, 2006. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2005.07.018. URL https://www.sciencedirect.com/science/article/pii/S0957417405001399.

Wikipedia contributors. Cinema of india — Wikipedia, the free encyclopedia, 2024. URL https://en.wikipedia.org/w/index.php?title=Cinema_of_India&oldid=1209395942. [Online; accessed 22-February-2024].