RECOMMENDATION SYSTEM USING MACHINE LEARNING

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

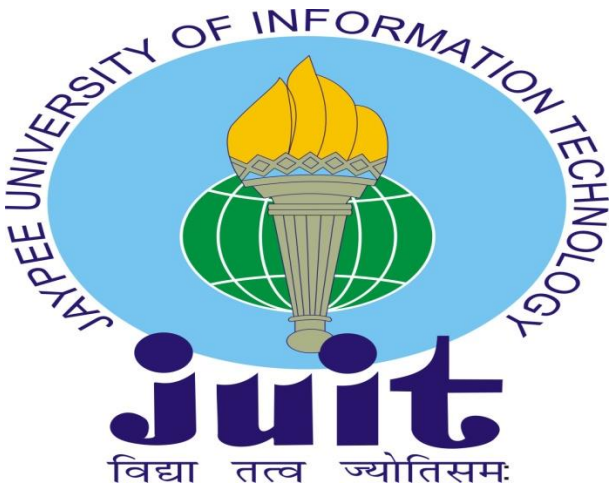
By

Harshita Sharma 181215
Hiteshwar Singh 181223

Under the supervision of

Dr. Jagpreet Sidhu

to



Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled " RECOMMENDATION SYSTEM USING MACHINE LEARNING " in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2022 to May 2022 under the supervision of Dr. Jagpreet Sidhu, Assistant Professor Senior Grade Computer Science and Engineering/ Information Department. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Hiteshwar Singh 181223

Harshita Sharma 181215

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Jagpreet Sidhu
Assistant Professor Senior Grade
Computer Science and Engineering/ Information Department
Dated:

ACKNOWLEDGEMENT

To begin, I would like to express my heartfelt gratitude to almighty God for his heavenly grace, which enabled us to successfully complete the project work.

I am extremely grateful and wish to express my deep gratitude to Supervisor **Dr. Jagpreet Sidhu, Asst. Prof. Senior Grade,** Department of CSE&IT Jaypee University of Information Technology, Waknaghat. His never-ending patience, intellectual direction, persistent encouragement, constant and vigorous supervision, constructive criticism, helpful suggestions, and reading numerous poor versions and revising them at all stages allowed this project to be completed. I would like to express my heartiest gratitude to Dr. Jagpreet Sidhu**,** Department of CSE&IT, for his kind help to finish my project.

I would also like to express my gratitude to everyone who has assisted me in making this project a success, whether directly or indirectly. In this unusual scenario, I'd like to express my gratitude to the different staff members, both teaching and non-teaching, who have provided me with valuable assistance and assisted my project. Finally, I must express my gratitude for my parents' unwavering support and patience.

Hiteshwar Singh (181223)

Harshita Sharma (181215)

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

At the present age, recommendation systems have become very imperative and have been extensively used in the present world to know the customers' opinions. Opinion mining has become an important tool for any industry to understand their user sentiments, current trends and user responsiveness towards their products and services. Recommendations help users decide whether the products and services are worth their money and time. There are various applications of the recommendation systems. The availability of all the reviews of the movie can be considered to help the users make their decisions by not making their time go futile reading and viewing all the ratings and the reviews. Movie-rating applications and websites are frequently used by the experts and the commentators to post comments and rate the films according to their perspectives which help the other viewers decide and recognise if the movie is worth watching or not. Different e-commerce websites help the customers decide what to buy. Hence, the task of comprehending if a movie is to be recommended to user or not can be fully converted and automated using the algorithms as the machine or the system learns through training and testing the data. By this project, we aim to rate the reviews using user based and item based collaborative filtering and content-based filtering. Using different features to get recommendation of the movies that are best suited for a particular user.

Chapter 01: INTRODUCTION

1.1 Introduction

Recommendation systems play a vital role for several online websites, applications and e-commerce services, like social-network, recommendation of items and entities like movies, music and articles. People in today's world have a pool of choices to choose from. It can be the choices of movies, songs, books etc. Also there has been an information explosion in the past few years. There has been a gradual increase in the amount of available digital information, electronic data in recent years. This information explosion makes it hard for people to manage their time. In order to help the people to come up with the humongous number of choices, recommendation systems have been developed and have been put to work. Recommendation systems can help the people know what is right by making available the things they will like easily based upon their preferences and likings. Performance of recommendation systems have a significant influence on the success of the commercial companies and industries on the basis of revenue generation and their customer satisfaction. Recommendation systems are distinct from other types of expert systems because these integrate an expert's subject knowledge with the customer's or the user's preferences to filter out the available data and information for a particular user.

Machine learning is a critical component for a recommendation system. It is a method for giving computers intelligence by simulating how the human brain works. It has been extensively used in the data mining and knowledge discovery fields. The two very basic approaches used to create the recommendation systems are the content-based filtering technique and the collaborative filtering technique.

Content-based and Collaborative filtering-based approaches are the basic approaches for Recommendation systems [11]. Collaborative filtering makes recommendations based on user and item similarity measurements. The algorithm suggests things that are popular among people in comparable groups. It is determined by the user's preference profile and the item description.

In Content based Filtering technique, keywords are employed in addition to the user's profile to highlight the user's preferred likes and dislikes. In other words, the content-based filtering

algorithm recommends goods that are similar to ones that were previously loved. It looks at previously rated things and suggests the best match.

The aim is to determine whether the textual information generated by the users convey their positive(good), negative(bad) or neutral opinions. It has various approaches including the lexicon and machine learning techniques. The reviews have to be categorised in positive, negative and neutral categories. Using content based and collaborative filtering methods like user-based and item-based filtering, recommendations can be done. These techniques help in building the recommender systems based on the likings and disliking of the similar users.

In this project we used various approaches to build different types of recommendation systems. Recommendation based on scores, overviews and genres were the major areas to work on. With the help of abundant data available we were able to build models that can recommend movies to the user based on which category of recommendation users want.

1.2 Problem Statement

Due to the excessive availability of the data and information on the internet, developments in customization, growing internet connectivity, and evolving technology, recommendation systems are effective in generating great suggestions. Sentiment analysis helps in devising the textual information in positive, negative and neutral categories.
The areas of Machine learning and Data Mining and these techniques have made several advancements in this domain. Many research studies have been conducted in the discipline of opinion mining and sentiment analysis using collaborative filtering and content based. The main focus of this project was to build an integrated system that can recommend movies which user is mostly likely to watch or choose for themselves.

1.3 Objective

Objectives of the project:

- To build a model using various collaborative algorithms which gives the highest accuracy in recommending the users based on the similar users' interests.
- To build a model using various content similarity algorithms which gives the highest accuracy in recommending the users based on the similar users' interests.

1.4 Methodology

For building the model, amazon dataset has been used. The dataset was present in .txt format has to be converted into 2-dimensional data. For this python language has been put into use. This dataset contains around 45000 different movies with 24 features. Dataset included some different genres for different movies which needed to be separated and cleaned. Individual genre for each movie was found and better clean dataset was formed by the team.

Sequences would be created out of the dataset and the processed input sequences are to be fed to the model. After fine tuning the model, the performance and computational metrics are checked. The content-based and collaborative filtering-based methodologies such as item-based and user-based methods are then applied to the processed dataset and the accuracy values are checked.

1.5 Technical Requirements

- Python IDLE
- Libraries including Scikit learn, Pandas, Numpy, Transformer, BERT model.

Chapter 02: LITERATURE SURVEY

2.1 Literature Survey

The internet in today's world has been overflowing with the enormous amount of textual form of data which is growing rapidly every minute. It has become very difficult to extract the exact information about a particular entity. As the Internet has grown in popularity, the need for individualized information systems has grown as well. The humongous amount of data has passed the limits of human capacity to search, organise and categorise it. In the past few years, there has been an evolution of the process of opinion gathering of the customers and the users. There are several websites, applications and even social media platforms which are gathering their users' reviews, likings and disliking. Different reviews contain different expressions, views and emotions which are hard to be categorised manually.

Sentiment Analysis helps in undercovering the reviews, opinions and other subjectivities of the users. It primarily uses the Machine learning techniques and the Natural Language Processing, NLP techniques to unveil the attitudes of the users on a particular subject by recognizing and categorising the textual patterns from the text available. Many researches and studies have been conducted in this field. Successful applications of these systems can help in the high revenue achievements for the companies. These are also helpful in increasing the satisfaction level of the customer base of different platforms. In recent years, recommender systems have grown in popularity and are now employed in a variety of online applications. Recommender Systems are the software systems that give users with recommendations based on their needs.

Content-based recommendation systems look at a set of documents and/or descriptions of items that have been previously evaluated by a user and build a model or profile of that person's interests based on the features of the objects that have been rated. A user's profile is a structured representation of their interests that is used to recommend new items to them. The main idea underlying the recommendation process is to match the properties of a content item to the qualities of a user profile. The end result is a relevance score that represents the user's level of interest in the item. When a profile accurately reflects user preferences, the information access process becomes substantially more efficient.

Collaborative Filtering approaches play a crucial part in the recommendation process, and as a result, Collaborative Filtering is the most often used methodology which is used to construct recommender systems now-a-days. In this method, each active user receives a suggestion or recommendation based on the preferences, disliking or likings of other users who have already rated any item or product similarly to the current user. Several researchers have conducted their studies on such recommendation systems and have proposed some models using various algorithms and methodologies.

Andrea Chiorrini et al. [1] carried out a study on the BERT model for both sentiment analysis and emotion recognition of Twitter data. They have compared the cased and uncased BERT
model for the twitter dataset and have got 90% accuracy for cased BERT and the uncased Bert has given the accuracy of 89%.

In the research conducted by Manish Munikar et al. [2] sentiment classification has been done using the BERT model and sentiment labels have been provided to the review text. After evaluation accuracy given by RNN on Stanford Sentiment Treebank (SST) is 86.1 and accuracy provided by the LSTM is 84.9%

The study performed by Bing Liu et al. [3] shows the usage of the BERT model as their base model and the post-training algorithm on the dataset of Amazon laptop reviews. Accuracy given by the proposed model has been achieved as 78.07%.

Kuat Yessenov [4] has developed an experimental model for improving the efficacy of the machine learning techniques in the field of sentiment analysis and has used technique like Naive Bayes, Bagging algorithms, Maximum-Entropy, and K-Means clustering with highest accuracy of 75% using the Maximum entropy and Naive Bayes algorithm with 45% accuracy.

M.Govindarajan [5] has proposed a hybrid model of naive bayes and genetic algorithm for the sentiment analysis of movie reviews. When individually applied, Naive bayes achieved 91.1% and Genetic algorithm achieved 91.25% accuracy. The hybrid system made the optimum use of the classifiers and performed 93.8% accurate.

V.K. Singh et al. [6] have done aspect level sentiment analysis for the movie reviews and experimented with various linguistic feature selection, aggregation, and weighing systems. SWN has given 78% accuracy while the Alchemy API has achieved 77% accuracy on the manually created moderate sized dataset.

B. Lakshmi Devi et al. [7] have also conducted the sentiment analysis study on movie reviews using two classifiers. Data set used in the model is a set of 500 IMDB movie reviews with half positive and half negative reviews. Naive Bayes and Decision trees have been used with the highest accuracy of 92.3% of naive bayes in the model.

Al-Rubaiee et al. [8] used two types of sentiment classification: polarity classification and rating classification. They used SVM, MNB, and BNB machine learning algorithms in their project. The accuracy of sentiment polarity classification was 90 percent, but there was still room for improvement in the second form, rating classification, where the accuracy was just 50% which is quite low in this case.

For sentiment analysis, deep learning has been frequently employed. Socher et al. [9] introduced a 91% accurate RNN (Recurrent neural network) based solution that is trained on a sentiment treebank and has greatly improved sentence-level sentiment analysis on English datasets. Sarah Al Humoud [10] utilised an LSTM CNN model with two imbalanced classes (positive and negative) among four types of ASTD: objective, subjective positive, subjective negative, and subjective mixed form. The model shows good levels of accuracy and precision using CNN.

Juan M Fernandez et al. [19] has examined the two very basic and traditional recommendation systems which are based on content based filtering and collaborative filtering. Due to several drawbacks and disadvantages in both the systems like the early rater problem, limited content analysis, the gray sheep problem, no recommendation for the unanticipated items, cold-start issue, over specialization problem, new user problem and the sparsity problem, they built a new system using two approaches based on the Bayesian network and collaborative filtering. The newly proposed system is highly efficient for the problem and gives probability distributions, helpful to make inferences from them. The proposed model of the recommendation system was tested on a pre-processed database, and its effectiveness has been compared and contrasted with the existing methods and was

proven in on-line experiments. Vladimir-Nicolae Dinu et al. [20] designed a Movie Recommender, in this system the movies are recommended by the model through the information that is available about the user. Several features present in the model contains the psychological profile of the user, the watching history of the users, the information including scores of the movies based on other websites. They are completely based on aggregated similarity calculation. This is the combination of content and collaborative filtering which removes the data particular to a specific user.

Geetha G et al. [21] proposed a model based on a hybrid approach. With the application of basic k-means clustering, collaborative and content-based filtering approaches, they have made a model which allows a user to choose or make his choices from a pre provided set of features and then recommend a movie list based on the aggregated or cumulative weight of different attributes. This clearly depicts the use of Pearson's correlation function in their model. James Salter et al. [24] proposed a cinema-based recommendation model that automatically generates recommendations for a random sample of 200 users using the combination of both the collaborative-based filtering and content-based filtering algorithms. This system generates a list of movies showing at the particular cinemas and extracts out the matching films from the complete record of the recommendations being generated. The dataset contained 100,000 film ratings by 943 users. This model makes output in the form of coverage of predictions I.e. Catalogue coverage, prediction coverage, standard coverage.

All the studies have been compared and comprehended. The table below depicts the comparison of the accuracy values of the different research studies done in the same field using Naive Bayes, RNN, BERT and other algorithms.

Table 1: Comparison of the different approaches on Recommender System.

Author	Algorithms	Accuracy
Manish Munikar	RNN	86.1%
	LSTM	84.9%
R Socher	RNN	91%
	KNN	83%
B. Lakshmi Devi	Naive Bayes	92.3%

M.Govindarajan	Genetic Algorithm	91.25%
	Naive Bayes	91.1%
	Hybrid approach	93.8%
Bing Liu	BERT	78.07%
Kuat Yessenov	Maximum entropy	75%
	Naive Bayes	45%
Andrea Chiorrini	Cased BERT	90%
	Uncased BERT	89%
Eyrun A.	SVM	78%
	K-Means Clustering	83.7%
Poonam B. Thorat	KNN	78.7%
	SVM	83%

Chapter 03: SYSTEM DEVELOPMENT

3.1 Algorithms

3.1.1 Collaborative Filtering

Collaborative filtering is one of the most used techniques in making recommendation systems. This technique of filtering, takes up the similarities between the users and the items at the same time and then it makes the recommendations [12][14]. A collaborative filtering model can recommend an item to a user, say user 'A' based on the interests, likings and disliking of a similar user, say user 'B'. This method makes recommendations by analysing correlations between users. Collaborative filtering needs the data or the historic information about the users regarding any entity like movies, music, products or services. There are two types of Collaborative Filtering techniques with a principal purpose of approximating target user's rating or review for the target entity, these techniques include User-based collaborative filtering (UbCF) and Item-based collaborative filtering (IbCF).

Table 2: Collaborative Filtering User-Movie Table

User	Movie1	Movie2	Movie3	Movie4
User1	5	4		5
User2	4		3	
User3		1		2
User4	1	2		

In this scenario, we can see that User 1 and User 2 give the movie nearly identical ratings, so we can conclude that Movie 3 will be averagely liked by User 1, but Movie 4 will be a good recommendation to User 2. We can also see that there are users who have opposing preferences, such as User 1 and User 3.Because User 3 and User 4 share a same interest in the film, we can predict that Movie 4 would be despised by User 4.This is Collaborative Filtering; we recommend users the items which are liked by the users of similar interest domain.

Collaborative Filtering models are considered models with significant accuracy among the basic fundamental recommender system models which generally use the target user-item interactional information such as reviews and ratings. There have been many efforts in the domain of recommender systems where the collaborative filtering methods are

combined with content-based filtering techniques to make a hybrid approach for the recommender systems [13]. In this method, surveys of similar users need to be done. This method has the tables of the users with different items and entities rated by them which they choose or like. Based on the similarities of the preferences, anticipations can be made of what the user might like, based on the likes, dislikes and the preferences of the similar users to the target user. The list in this method is then filtered and are matched accordingly to the users who used the similar items for comparison, suggestions and recommendations. The highest score will then be recommended to the target user.

There are several problems in Collaborative Filtering:

1. Sparseness that occurs due to the unavailability of the data or the sparsity of the rating values in the dataset.
2. Scalability issue.
3. Cold-Start problem that is it does not work with cold-start or new users because the dot product will be all 0s and hence it cannot recommend anything.

3.1.2 Content Based Filtering

A Content-Based Recommender system is entirely based on the information gathered directly from the ratings given by the users or intuitively the search phrases. Based on the data, a user profile is created, which is subsequently utilised to provide suggestions or recommendations to the user. The engine grows increasingly accurate and precise when the user provides new information or acts on the recommendations. The three main aspects of Content Based Filtering technique are User Profile, Item Profile, and Utility Matrix.

In the User Profile, vectors are made that define the user's preferences and likings. A single user profile is created using the utility matrix, which depicts the relationship between the person and the item. The closest conclusion that can be made based on this data is that a user favours a mix of those items.

The performers, director, release year, and genre are the most significant qualities of a film when converted into an item. We may also put the IMDB (Internet Movie Database) rating in the Item Profile. The Utility Matrix represents the user's preference for certain items. In the data obtained from the user, the utility matrix is utilised to find a link between the items that the user loves and those that he or she hates. Each user-item pair is assigned a numerical

number known as the degree of preference. We then develop a user matrix using the required elements to assess their preference connection.

Table 3: Content-Based Filtering User-Movie Table

User	Movie1	Movie2	Movie3
User1	3		1
User2	2	4	

Because we don't always receive complete information from the user, some of the columns in the matrix are empty with no value in them, and the goal of any recommendation system isn't to fill all of the columns which are there with no values, but to suggest or recommend a movie to the user that he or she would love. Based on this data, our recommender algorithm would not recommend Movie 3 to User 2 because their ratings in Movie 1 were nearly identical, and User 1 provided the lowest rating in Movie 3, making it highly probable that User 2 will detest it as well.

Recommending Items to User Based on Content:

Method1:

The cosine distance between the item's and the user's vectors can be used to establish the item's preference for the user. Consider a movie with a huge positive proportion of actors that the user likes and only a few actors that the user dislikes. As a result, the angle will be near to 0 and the cosine distance between the vectors will be tinyIf the cosine distance is big, it indicates that the user likes the movie; otherwise, we prefer to avoid the item from the suggestion.

Method2:

We may use a classification technique in recommendation systems as well, such as using the Decision Tree to determine if a user wants to watch a movie or not, and applying a condition at each level to refine our suggestion.

Apart from that, we employed various regression algorithms described in the methodologies and techniques portion of Chapter 4.

Chapter 04: PERFORMANCE ANALYSIS

4.1 Dataset

The dataset used in this model has been gathered by Kaggle. This dataset contains around 45000 different movies with 24 features which are mentioned below in the table. These features include some important feature that we will be using for our project work and also some new features will be derived from already existing feature with data pre-processing.

Table 4: List of all the features in dataset.

adult	belongs_to_collection	budget	genres
homepage	id	imdb_id	original_language
original_title	overview	popularity	poster_path
production_companies	production_countries	release_date	revenue
runtime	spoken_languages	status	tagline
title	video	vote_average	vote_count

4.2 Analysis

Analysing which feature plays what role in the dataset we can perform some data analytics techniques on the dataset to help us understand our data more. To start with we will let us plot a graph of most popular movies and top big budget movies. We will use the popularity, budget column from the dataset.

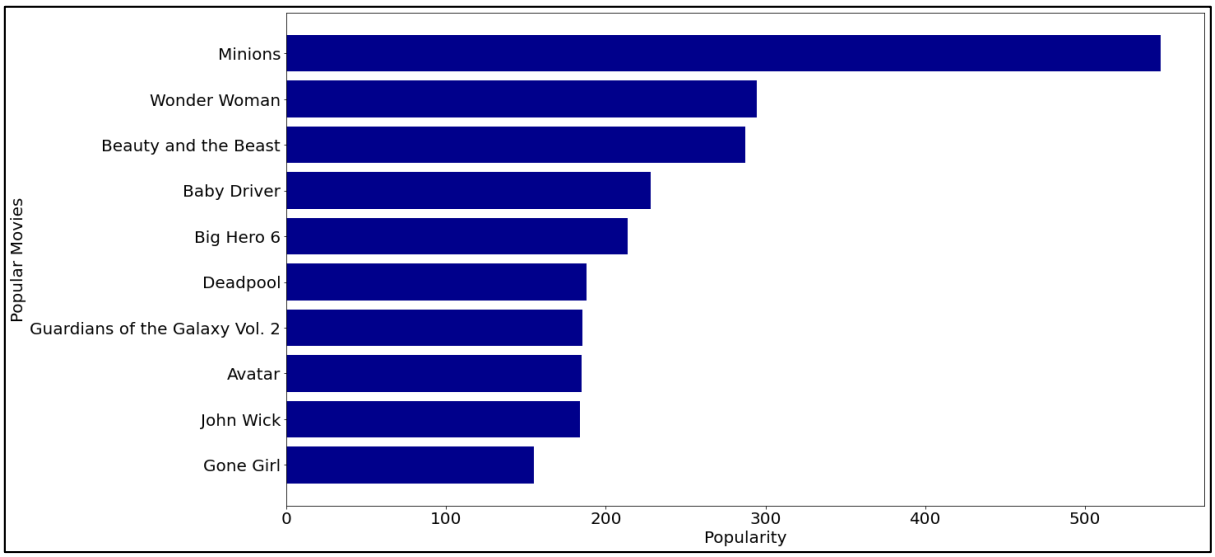


Figure 1: High Popularity Movies

Above graph showed us how different movies were ranked on the basis of their popularity index in the database. Top 10 highly ranked movies on the basis of the popularity were shown in the graph.

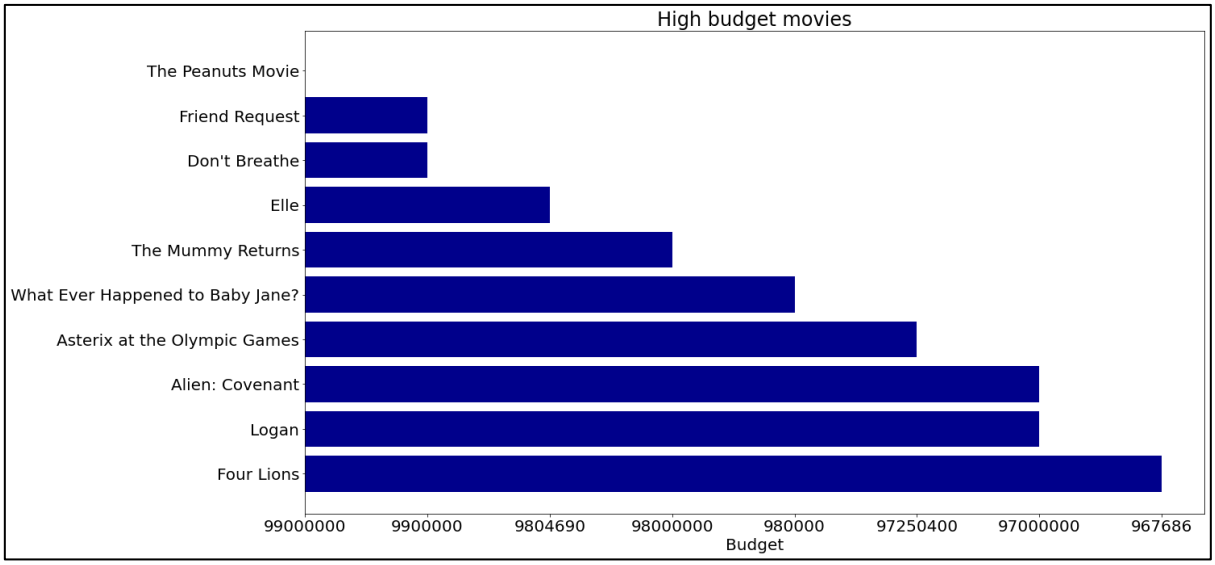


Figure 2: High Budget Movies

Here in the above graph, we saw which movies were high budgeted and now let us see which movies had the highest revenue and does any of the high budgeted movie made their way in top 10 highest revenue movies.

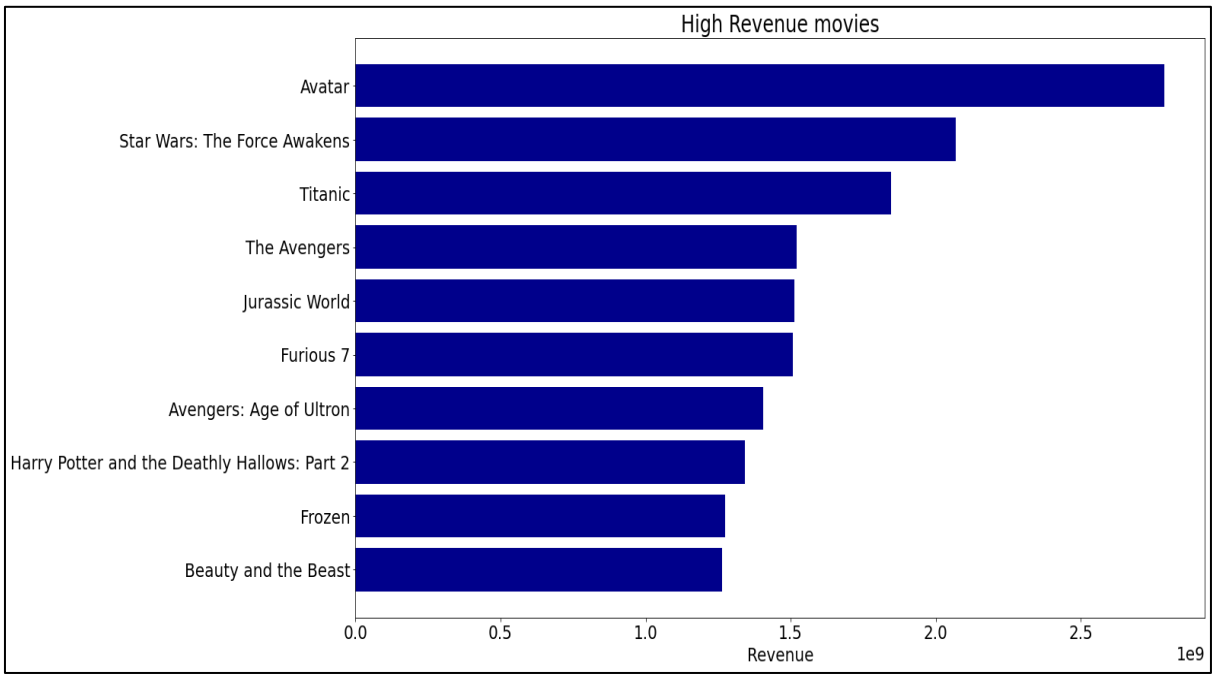


Figure 3: High Revenue Movies

By comparing both the graphs we can see that no high budgeted movies made to the top 10 for highest revenued movies.

Movies can also be liked or dislike on the basis of their length, in today's cinematic world most viewers prefer short length movies as they can have more content consumed in less time. So, we also have a graph that represents the runtime of top 10 movies with highest and lowest runtime.

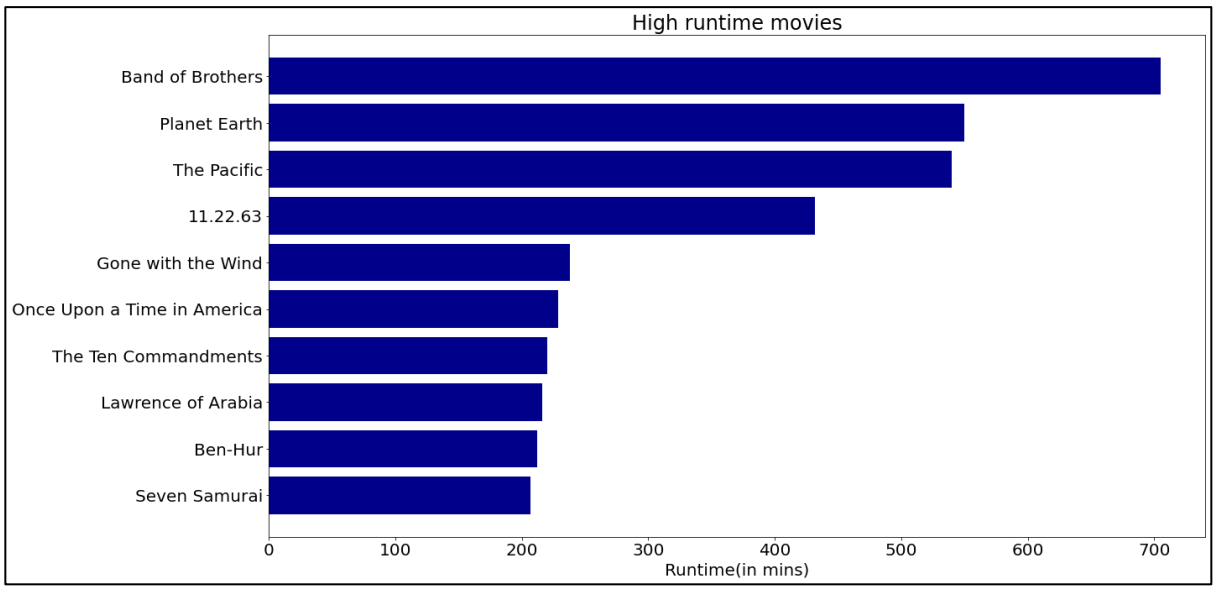


Figure 4: High Runtime Movies

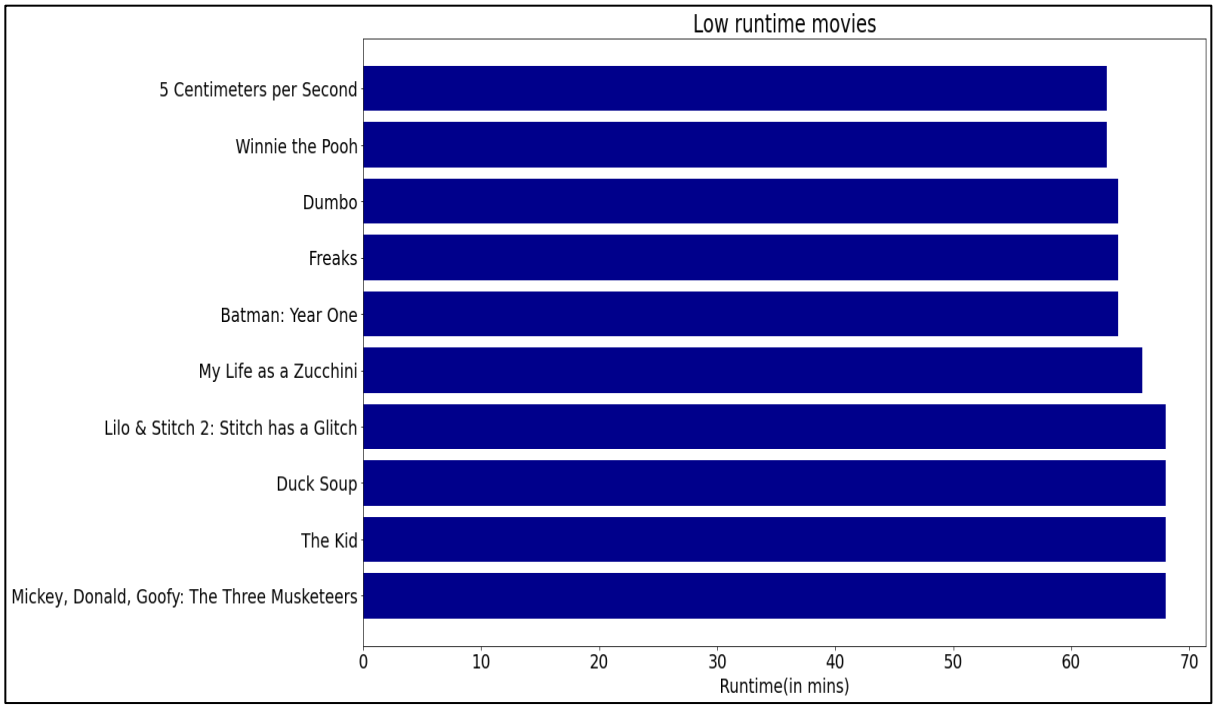


Figure 5: Low Runtime Movies

Now let us see how these movies varies with their orginal language i.e. how much percentage of movies were originally made in which langague and which language has the leading number of movies.

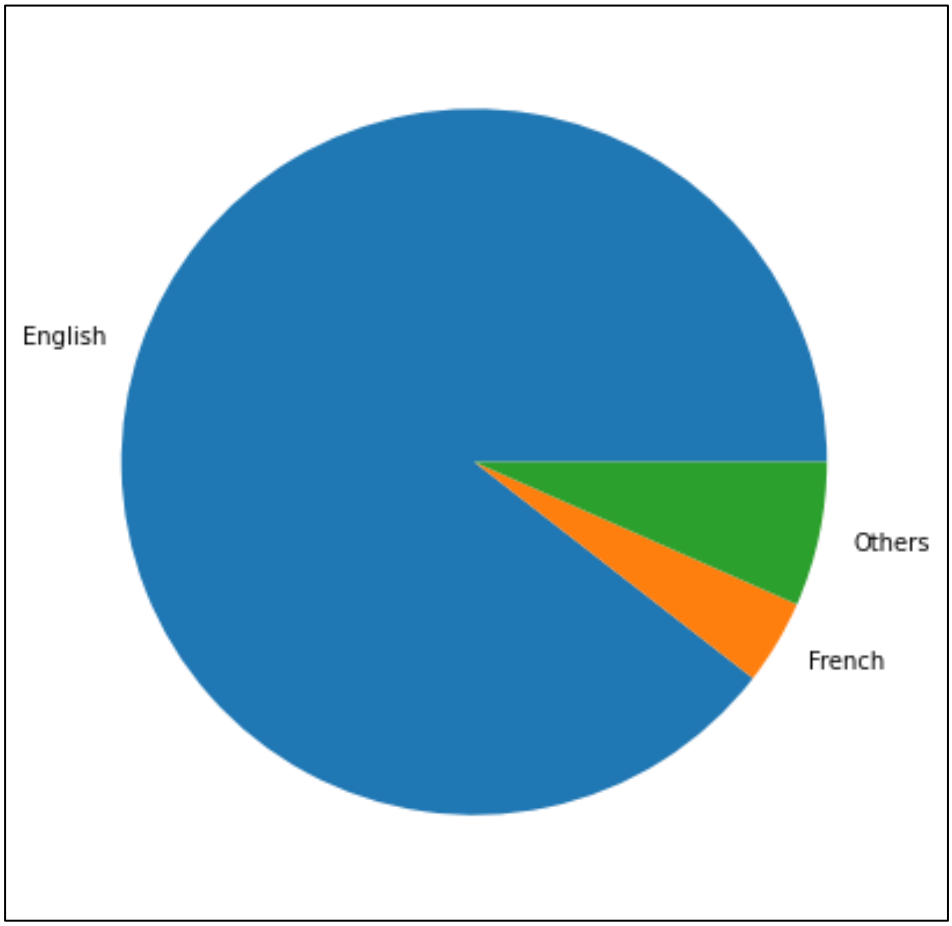


Figure 6: Language Classification of Movies

From the pie chart above we can se that most of the movies were made in "English" followed by "French" as the second highest language for movie makes.

4.3 Methods and Techniques

In this model we make three recommendation models first one based on the movie score which is calculated according to the IMBD formula, second one based on the content-based filtering of the overview of the movie and found the similarity based on the same and third we again used content-based filtering approach with the genres of the movies. But before moving on to creating a model for recommendation we need to pre-process our

dataset so that it will be clean and used properly. For this we removed all the null values from the dataset so that it wont cause any

4.3.1 Data pre-processing

The raw dataset of reviews gathered from IMDb was initially in the form of a plain *.text* file. We have used the python code to convert the whole dataset into 2-dimensional data. After processing the raw dataset, we filtered the only values we needed for our project. We also did separate data pre-processing for each specific model we needed without compromising the original integrity of the dataset. This dataset contained very detailed information of every movie that is mentioned in it.

4.3.2 Model 1: Score based recommendation model

In this model we tried a different approach for recommending movies to the user by taking the movie score for reference. User will use this model as a feature where the model will be provided with vote_average and vote_count for any particular movie and in return get the output score of the movies that match the particular score. To calculate the score we use the formula provided by IMDb (Internet Movie Database).

$$score = (v/(v+m) * R) + (m/(m+v) * C)$$

where,

v : vote_count of the movie

R : vote_average of the movie

m : 90% quantile of the vote_average

C : mean of the vote_count

	title	vote_count	vote_average	score
314	The Shawshank Redemption	8358.0	8.5	8.445869
834	The Godfather	6024.0	8.5	8.425439
10309	Dilwale Dulhania Le Jayenge	661.0	9.1	8.421453
12481	The Dark Knight	12269.0	8.3	8.265477
2843	Fight Club	9678.0	8.3	8.256385
292	Pulp Fiction	8670.0	8.3	8.251406
522	Schindler's List	4436.0	8.3	8.206639
23673	Whiplash	4376.0	8.3	8.205404
5481	Spirited Away	3968.0	8.3	8.196055
2211	Life Is Beautiful	3643.0	8.3	8.187171

Figure 7: Sorted Score by top 10 movies.

We chose quartile 90 because we want movies with more than 90% of votes to be included in the list, i.e. movies with more votes should be included. Now comes the part when we rate our movies based on their score to see which ones rank higher. The top 10 movies are shown in Figure 7.

After we have calculated scores for all the movies, we use it as our output vector and use v ote_average and vote_count as input vector. A pictorial diagram is also present to explain t he process.

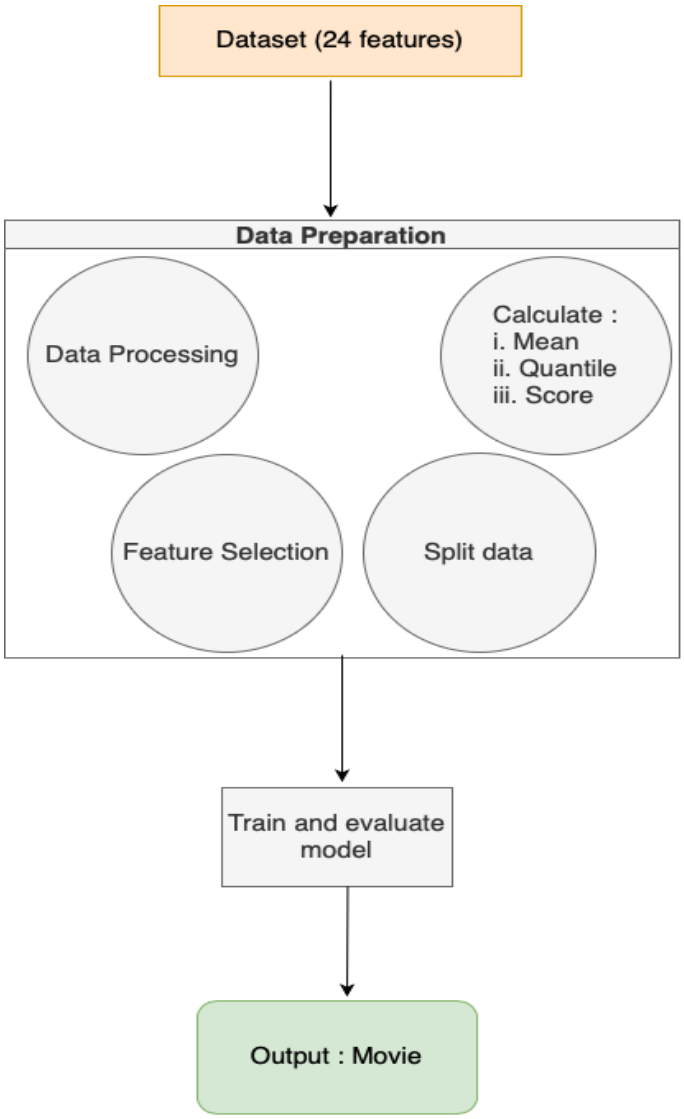


Figure 8: Flow Chart of Model 1.

To achieve the best results, we used four machine learning regression algorithm and compared them to have use the best one. The algorithms we used were Linear Regression, Decision Tree, Lasso Regression, Random Forest Regressor.

4.3.2.1 Linear Regression

For supervised learning, linear regression is a machine learning method. Linear regression is a technique for predicting a dependent variable (goal) from an independent variable (s). As a result, this regression approach determines if a dependent variable and the other independent variables have a linear connection. As a result, this algorithm's name is Linear Regression.

The independent variable is on the X-axis, while output is on the Y-axis in the diagram below.

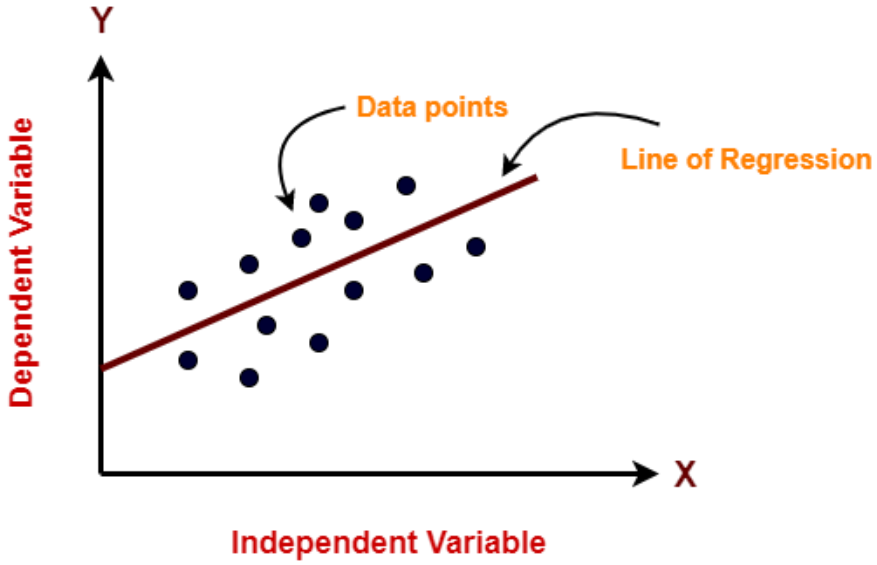


Figure 9: Linear Regression Graph

The regression line is the model's best fit line. And finding the greatest fit line is the major goal of this method.

Pros:

- Linear Regression algorithm is quite easy when it comes to the implementation.
- Also, linear regression is less complex than other techniques

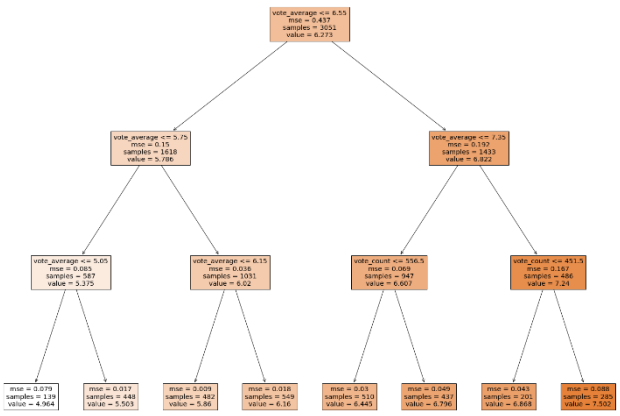
- Linear Regression can lead to the problem of overfitting but it can be avoided using some techniques and methods like dimensionality reduction, regularization techniques, cross-validation etc.

Cons:

- If there is the presence of the outliers, the performance can be affected very badly.
- This algorithm assumes a linear relationship between the variables for almost all the problems. Therefore, it is not advised to use linear regression technique for practical problems.

4.3.2.2 Decision Tree

The decision tree models may be used with any data that has both numerical and categorical properties. Non-linear interactions between features and the goal variable are well captured by decision trees. Because decision trees resemble human thought patterns,



understanding the data is simple.

Figure 10: Decision Tree representation

So, in a nutshell, a decision tree is basically a tree in which each node of the tree represents a feature, each branch of the tree depicts a decision, and each leaf of the tree portrays an outcome or an output which is generally a numerical value for regression cases.

Pros:

- Decision tree algorithm is very easy to understand and interpret visually as well.

- Working with numerical and categorical data is quite easy in the algorithm.
- This algorithm does not have any need for the pre-processing of the data like making dummy variables etc.

Cons:

- When built to its maximum height, it can lead to overfitting.
- Due to the creation of the tree structure, if there is any change in the data point, it can cause a big change in the whole structure of the tree.

4.3.2.3 Lasso Regression

LASSO is short for Least Absolute Selection Shrinkage Operator. A limitation on traits or parameters is defined as shrinkage. The approach works by identifying and imposing a constraint on model features that causes regression coefficients for some variables to drop toward zero. The model excludes variables having a regression coefficient of zero. So, lasso regression analysis is essentially a shrinkage and variable selection approach that aids in determining the most significant predictors.

Pros:

- Unlike decision trees, lasso tends to avoid overfitting.

Cons:

- From a set of correlated features or attributes, LASSO will choose just one feature.
- In this, the selected features can be highly biased.

4.3.2.4 Random Forest

Random Forests are a collection of decision trees that work together. It's a classification and regression algorithm based on Supervised Learning. Multiple decision trees are used to process the input data. Random forest starts executing by making a completely different number of decision trees at the time of training and making the

output class that is the mode of the classes (for classification) or mean or average prediction for regression use-cases of the individual trees.

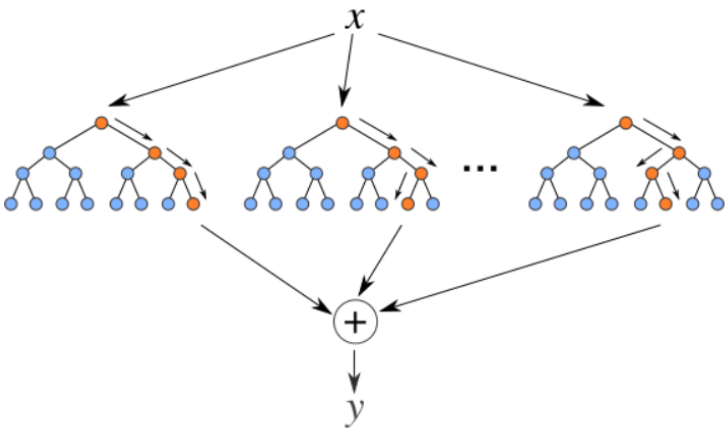


Figure 11: Random Forest diagram

Pros:

- For non-linear relationships and less complex problems, it works very well.
- Random forest is easy to understand and comprehend.

Cons:

- Due to some reasons, it can be prone to the case of overfitting.
- To obtain greater performance, larger random forest ensembles must be used, which slows down their pace and requires more memory.

After applying these five algorithms to our movies data we compared the results of all the algorithm in order to find the best of all. To compare these algorithms, we considered performance metrices like accuracy, recall, precision, fscore and ROC curve. The results are briefly discussed in section 4.2.3 Results and Analysis. With the help of the results we achieved we can determine which algorithm will be best suited to be used in the final model of our recommendation system.

4.3.3 Model 2: Content Based filtering based on Movie Overview

In content-based filtering we use the features of the movie or any item to determi ne its similarity with the other items present in the dataset. A pictorial diagram is also pres ent to explain the process.

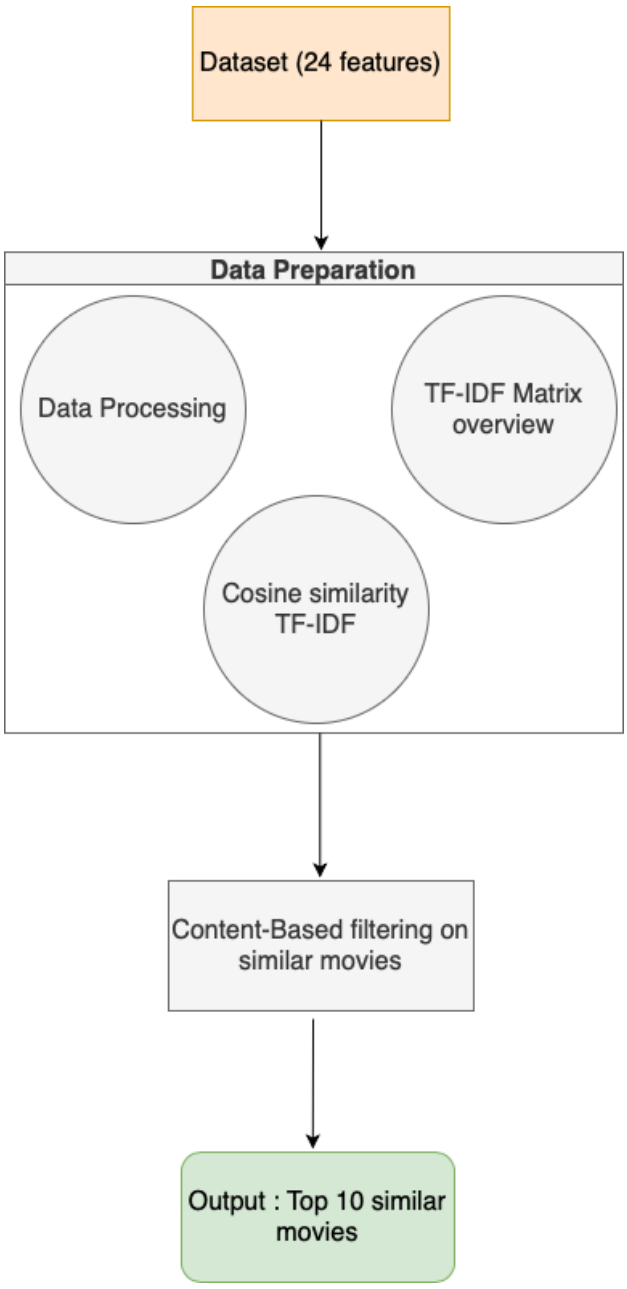


Figure 12: Flow Chart of Model 2

This similarity can be based on any feature of the item in this case say overview. To find th e similarity of the movies based on their overview we used some NLP (Natural Language P rocessing) methods. To begin with we made a database that only contained movie name an d overview. After that we defined a TF-IDF vectorizer object and removed all the English s top words such as ‘the’, ‘a’ etc. Then we replace all the NaN with an empty string. After re placing and pre-processing the overviews of the movies we constructed the required TF-ID F matrix by fitting and transforming the data. The matrix that we constructed was a sparse matrix of class float type with elements stored in Compressed Sparse Row format. After th is we computed the cosine similarity of TF-IDF matrix.

A get_recommendation function was made which returned the the top 10 similar movies of the input movie. To get recommendation get the pairwise similarity scores of all movies wi th in

put movie, sort the movies based on the similarity scores and get the scores of the 10 most similar movies.

```
get_recommendations('Forrest Gump')
```

```
4029                    Sweet November
782                Walking and Talking
797                Death in the Garden
1474    The Lost World: Jurassic Park
370                       Safe Passage
913      The Adventures of Robin Hood
3                   Waiting to Exhale
2624                 The Color Purple
4171                  A Knight's Tale
4461              Enemies: A Love Story
```

Figure 13: Output function for top 10 recommended movies for model 2

These top 10 recommendations are only based on the overview of the movies. These movies are similar according to their overviews.

4.3.4 Model 3: Content Based filtering based on Movie Genres

In content-based filtering we use the features of the movie or any item to determi ne its similarity with the other items present in the dataset. In this model we selected the ge nres feature of the movies to find the similarity of the movies. A pictorial diagram is also p resent to explain the process.

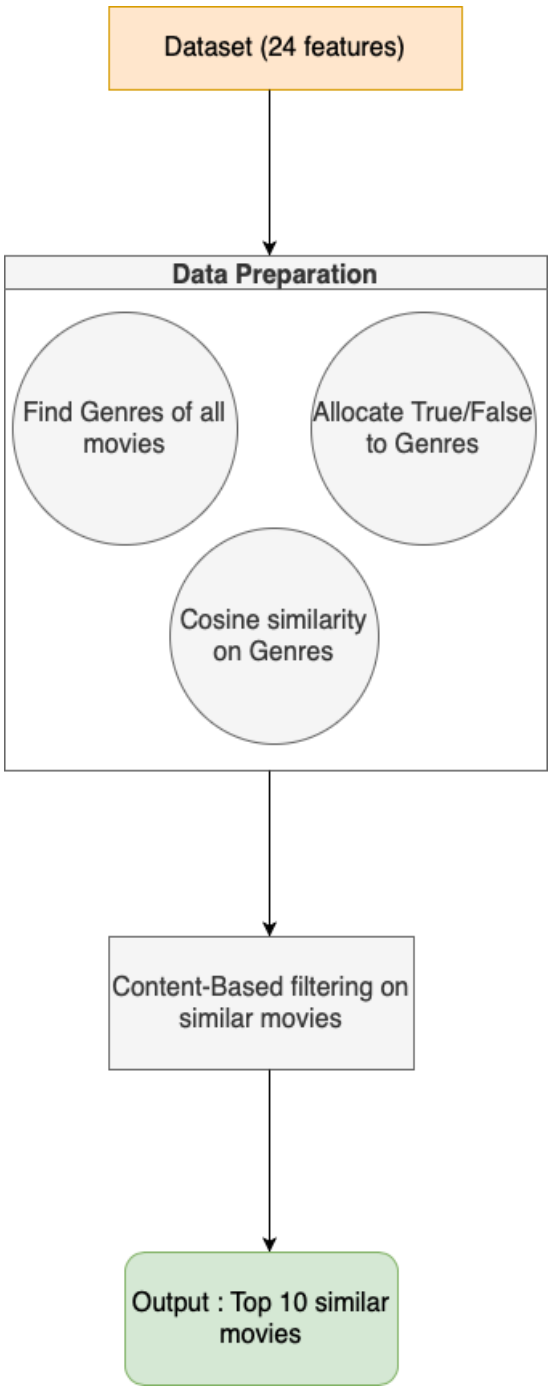


Figure 14: Flow Chart of Model 3

A particular movie can be of one single or multiple genres which will help us to determine how much similar it is to other movies. As discussed in Model 2 here also we create a separate dataset that contained movies and their genres marked as 0 if the movies don't belong to that particular genre and 1 if movie is of that genre. After pre-processing we had about 20 different genres in our dataset.

We computed cosine similarity of the genres of the movies and similarly as in Model 2 we created a get_recommendation function which gave an output of top 10 similar movies according to the genres of the input movie. An example of which is shown in the figure below.

Figure 15: Output function for top 10 recommended movies for model 2

All these different models can be improved further more if we include more features and integrate these models. For now, we have only focused on analysing how different features can get us different types of recommendation.

4.4 Result and Analysis

In this section we will discuss the results of our all the models that we created and also discuss the possibility of integrating all these models. Firstly, we will look into the performance metrices of model 1.

4.4.1

Mean Squared Error (MSE) :

The MSE is a measure of how near a fitted line is to data points. Take the vertical distance between each data point and the matching y value on the curve fit and square the value. MSE, or Mean Squared Error, is a commonly used regression error statistic. It's also an important loss function for algorithms that use the least squares framework to fit or optimise regression situations. The practise of decreasing the mean squared error between expected and anticipated values is known as "least squares."

$$MSE = 1/N * sumforitoN(y\_i - yhat\_i)^2$$

Here yi is the i'th anticipated value in the dataset and yhat_i is the i'th forecast value. The sign is eliminated when the difference between these two values is squared, resulting in a positive error value.

As a result of the squaring, large errors are also exaggerated or exacerbated. The squared positive error is proportional to the difference between the anticipated and expected numbers.

4.4.2 Root mean squared error (RMSE)

The square root of the mean square error is all it is. Because it uses the same units as the amount represented on the vertical axis, this statistic is perhaps the easiest to understand. The Root Mean Squared Error (RMSE) is a mean squared error variation. The square root of the error is determined, suggesting that the RMSE units are the same as the anticipated target value's original units.

The RMSE can be computed using the following formula:

$$RMSE = sqrt(1 / N * sum\ for\ i\ to\ N\ (y\_i - yhat\_i)^2)$$

Where y i is the dataset's i'th anticipated value, yhat i is the predicted value, and sqrt() is the square root function. In terms of the MSE, the RMSE may be rephrased as:

$$RMSE = sqrt(MSE)$$

The RMSE is a better indicator of goodness of fit than a correlation coefficient since it can be easily translated into measurement units. The RMSE can be compared to the variance in measurements of a typical point. For a good fit, the two should be similar.

4.4.3 Mean absolute error (MAE)

The Mean Absolute Error, or MAE, is a popular statistic because, like the RMSE, the error score's units match the expected target value's units. Unlike the RMSE, changes in MAE are linear and so visible. In other words, MSE and RMSE penalise larger errors more severely than smaller errors, inflating or magnifying the mean error score. Because the erroneous number is squared, this occurs. The MAE does not give different types of errors more or less weight; instead, the scores rise in a linear fashion as the error grows.The MAE does not give different types of errors more or less weight; instead, the scores rise in a linear fashion as the error grows. As the name implies, the MAE score is calculated as the average of absolute error values. Absolute, or abs(), is a predefined mathematical function that makes a number positive in nature. As a result of it, while the difference between an expected and projected number might be positive or negative, it must be positive when calculating the Mean absolute error.

The MAE may be determined using the following formula:

$$MAE = 1/N * sumforitoNabs(y\_i - yhat\_i)$$

The i'th anticipated value in the dataset is y i, the i'th forecast value is yhat i, and the absolute function is abs().

The table and graph also include these performance metrics and their values.

Table 5: Performance metrices for model 1

Algorithm	Accuracy (%)	MSE (%)	RMSE (%)	MAE (%)
Linear Regression	95.62	1.73	13.14	9.37
Decision Tree	91.59	3.32	18.21	15.25
Lasso Regression	92.28	3.04	17.45	13.2
Random Forest Regressor	94.84	3.04	17.45	13.2

From the above table we can say that Linear Regression performance was the best among all the algorithms we used followed by Random Forest Regressor. Linear Regression provides an accuracy of 95.6% with Mean Squared Error, Root Mean Squared Error, Mean Absolute Error as 1.73%, 13.14% and 9.37% respectively. We also have a graphical representation of our results to understand it in a better way.

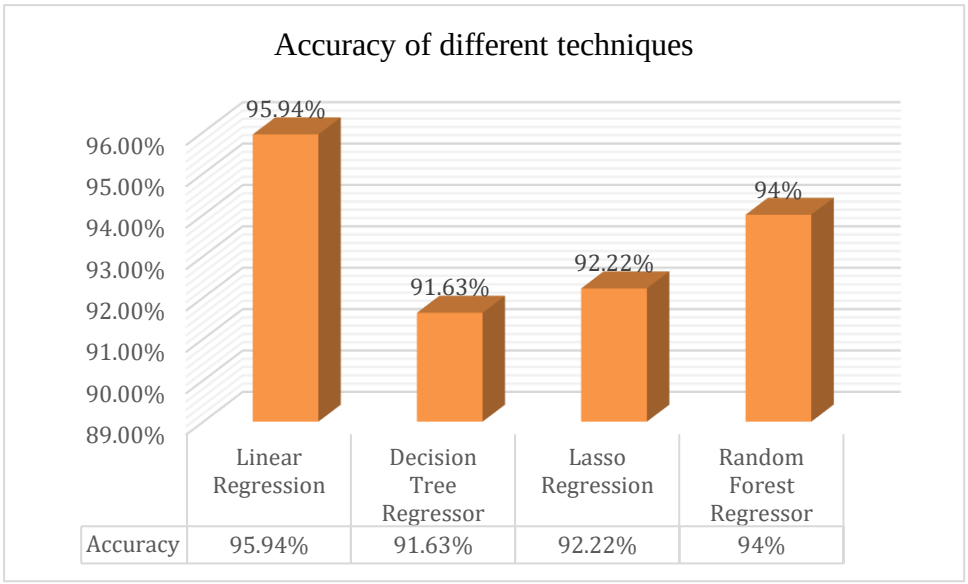


Figure 16: Accuracy graph for model 1 algorithms.

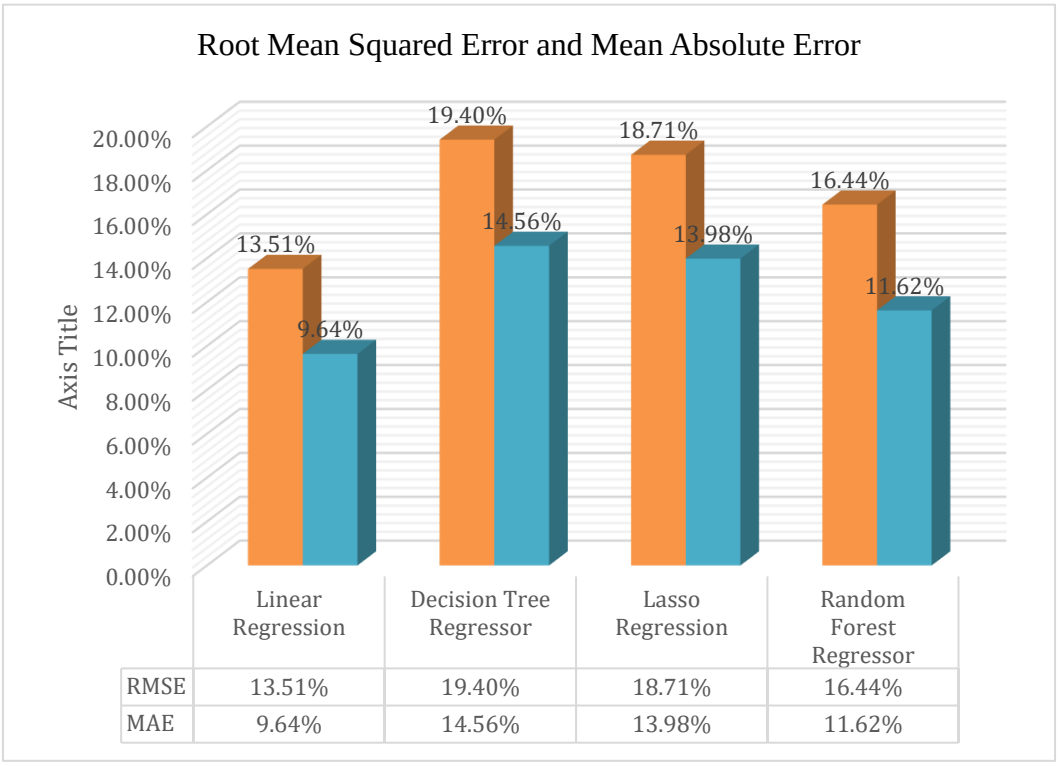


Figure 17: Root Mean Squared Error and Mean Absolute Error for model 1.

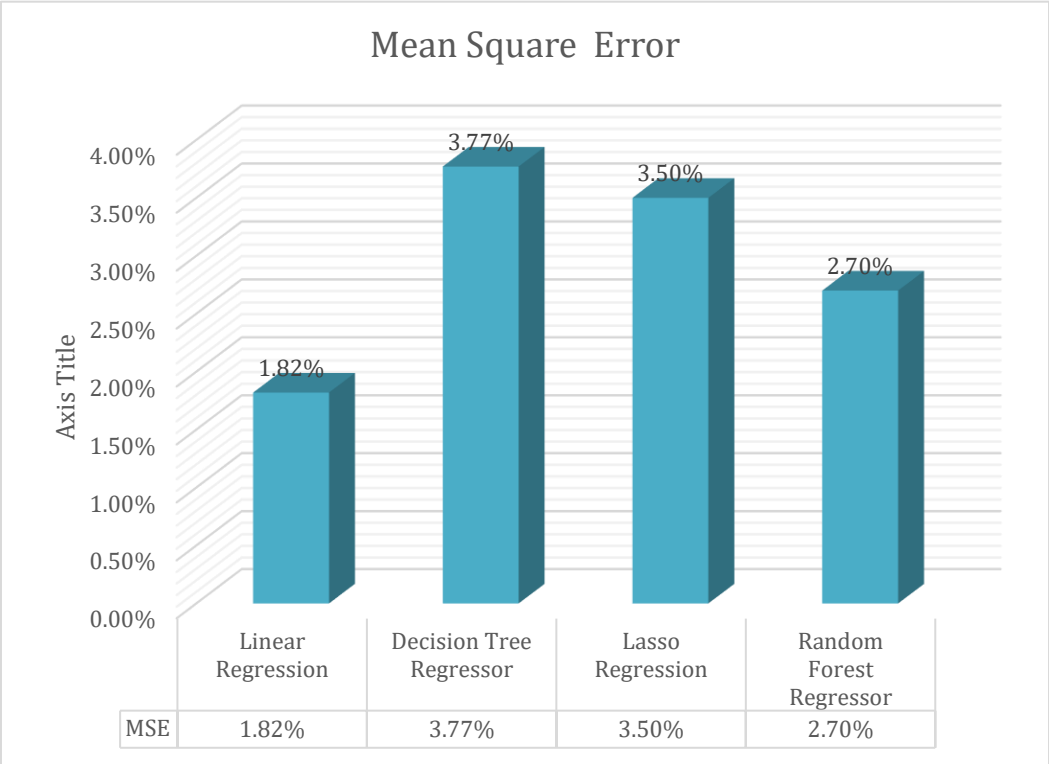


Figure 18: Mean Square Error for model 1.

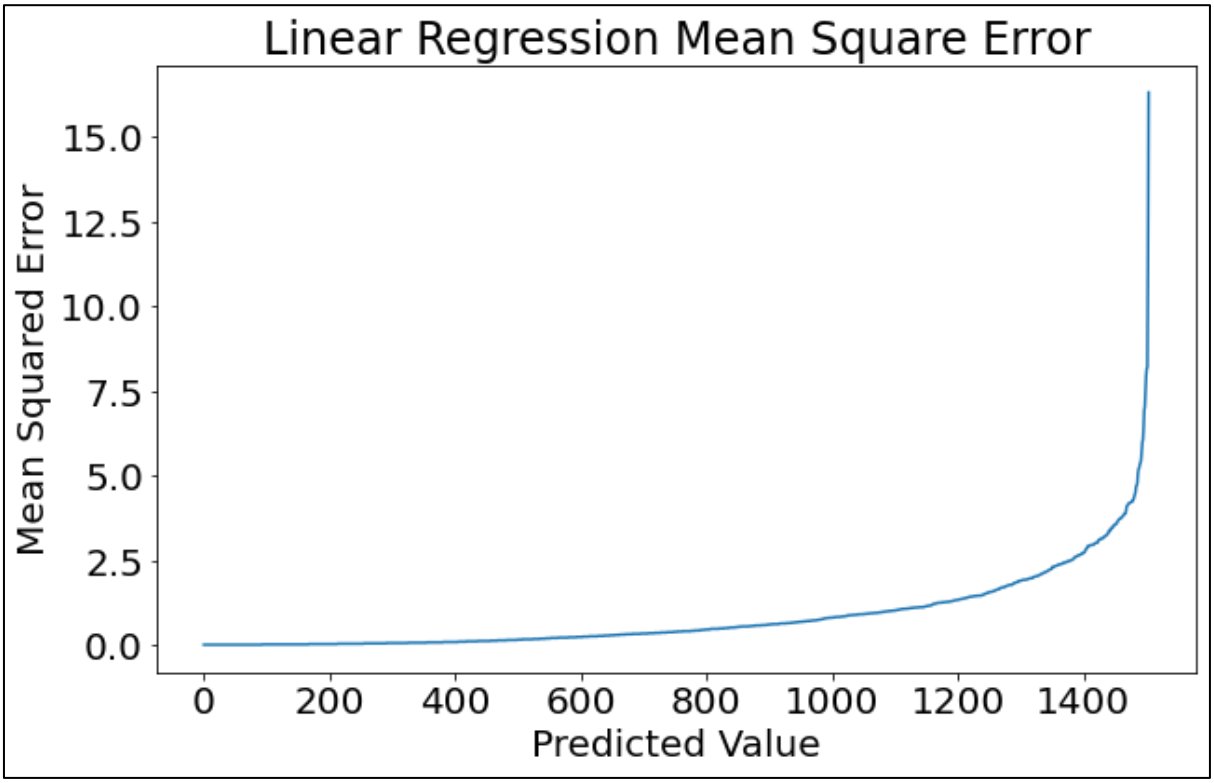


Figure 19: Mean Square Error graph for Linear Regression.

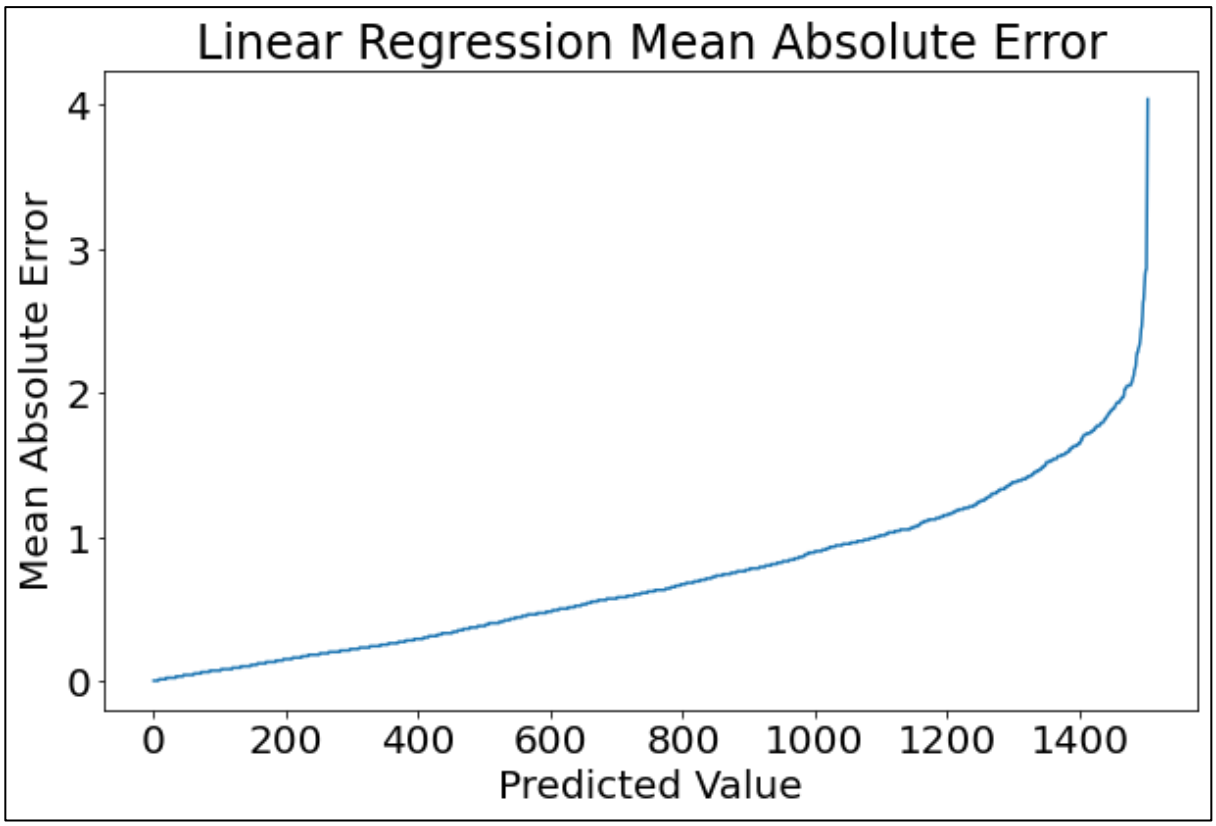


Figure 20: Mean Absolute Error graph for Linear Regression.

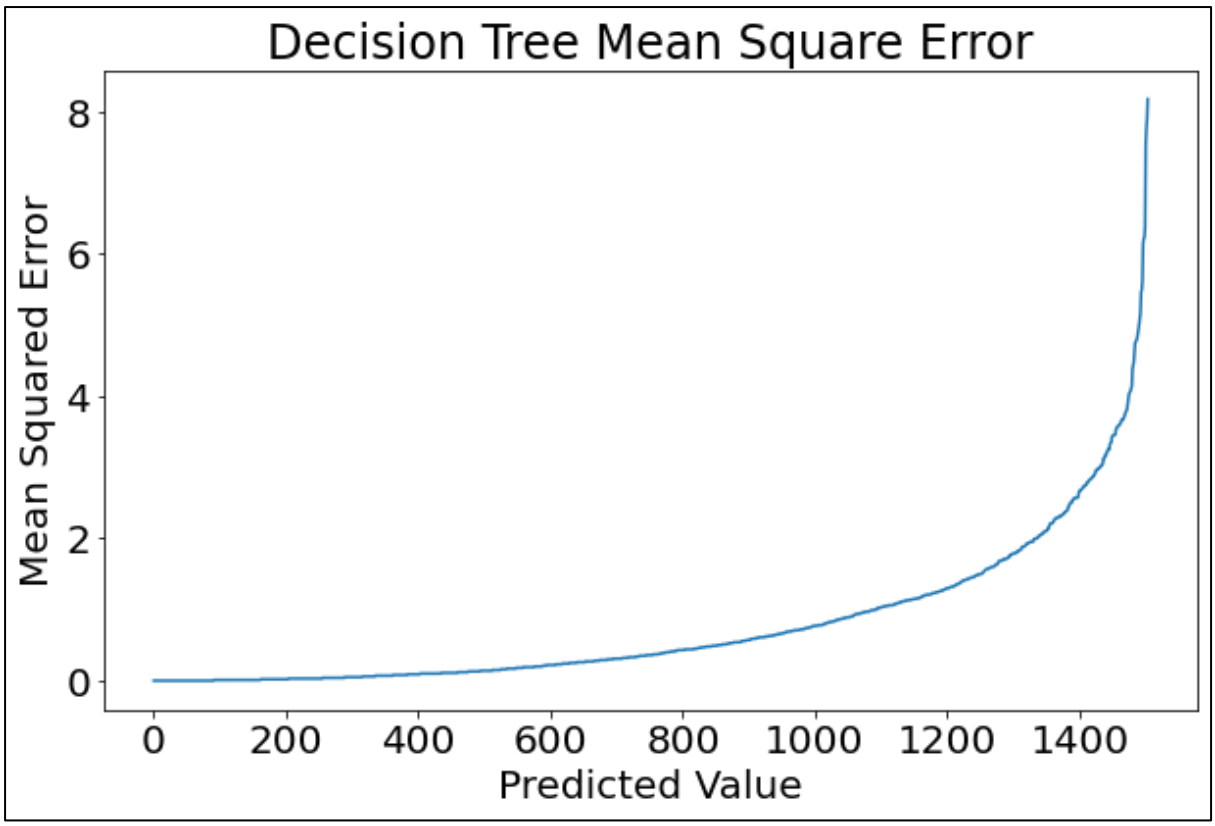


Figure 21: Mean Square Error graph for Decision Tree Regressor.

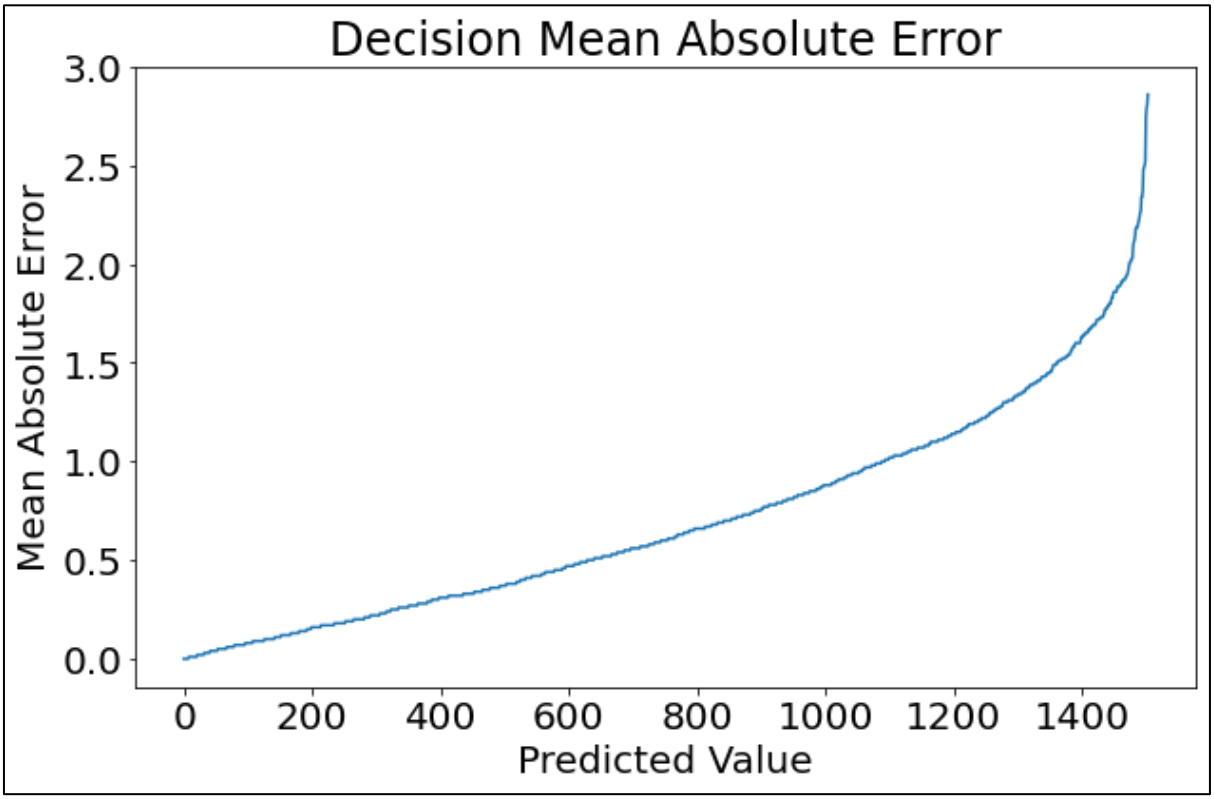


Figure 22: Mean Absolute Error graph for Decision Tree Regressor.

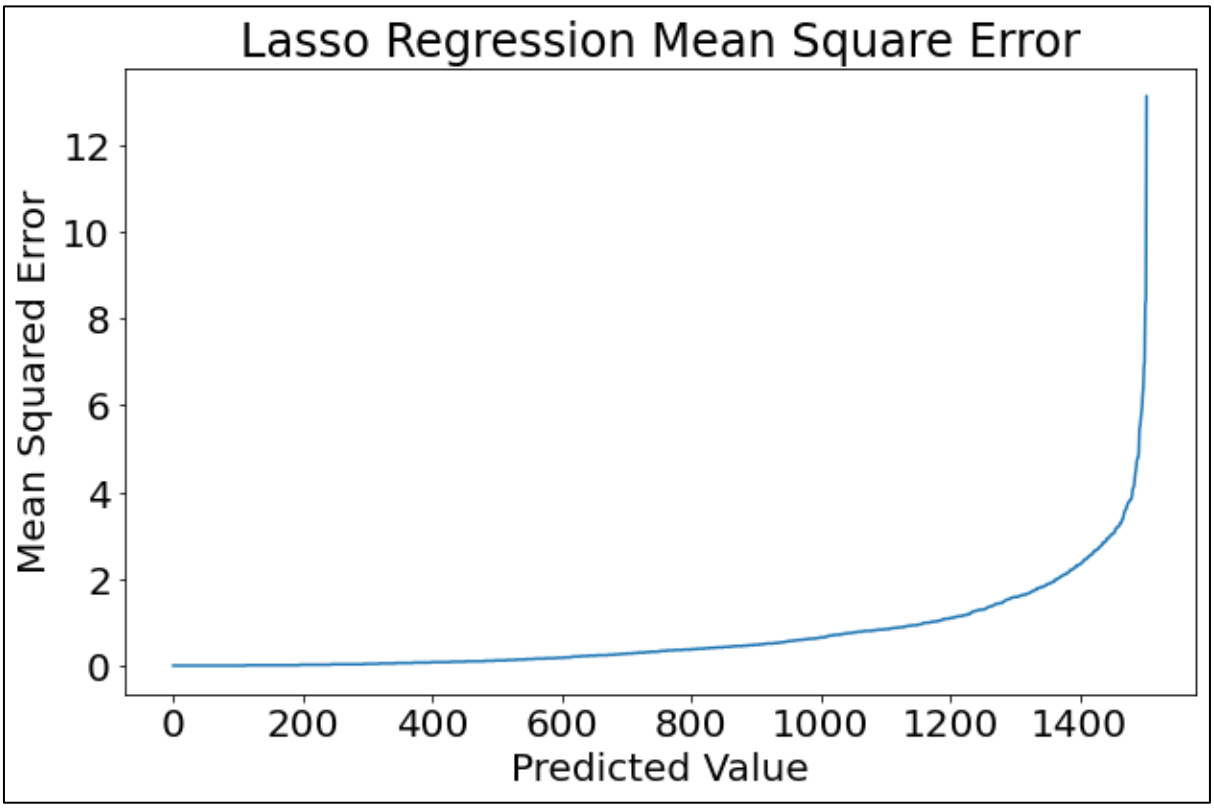


Figure 23: Mean Square Error graph for Lasso Regression

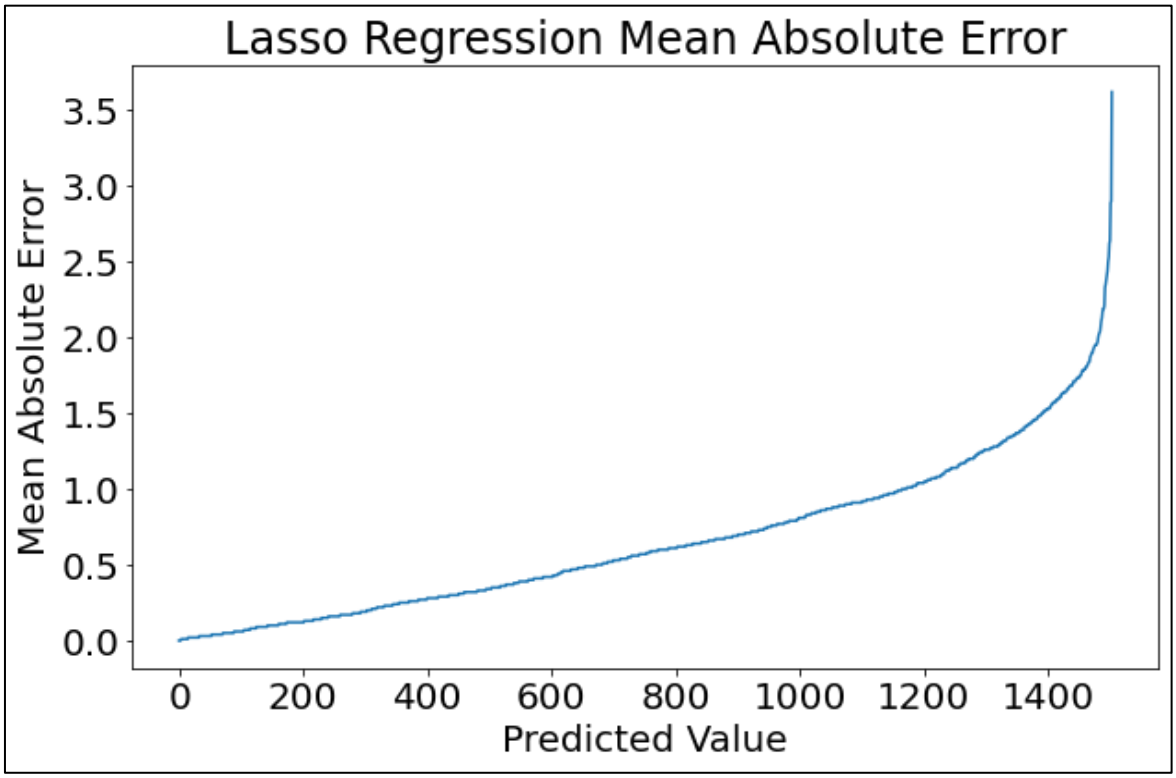


Figure 24: Mean Absolute Error graph for Lasso Regression

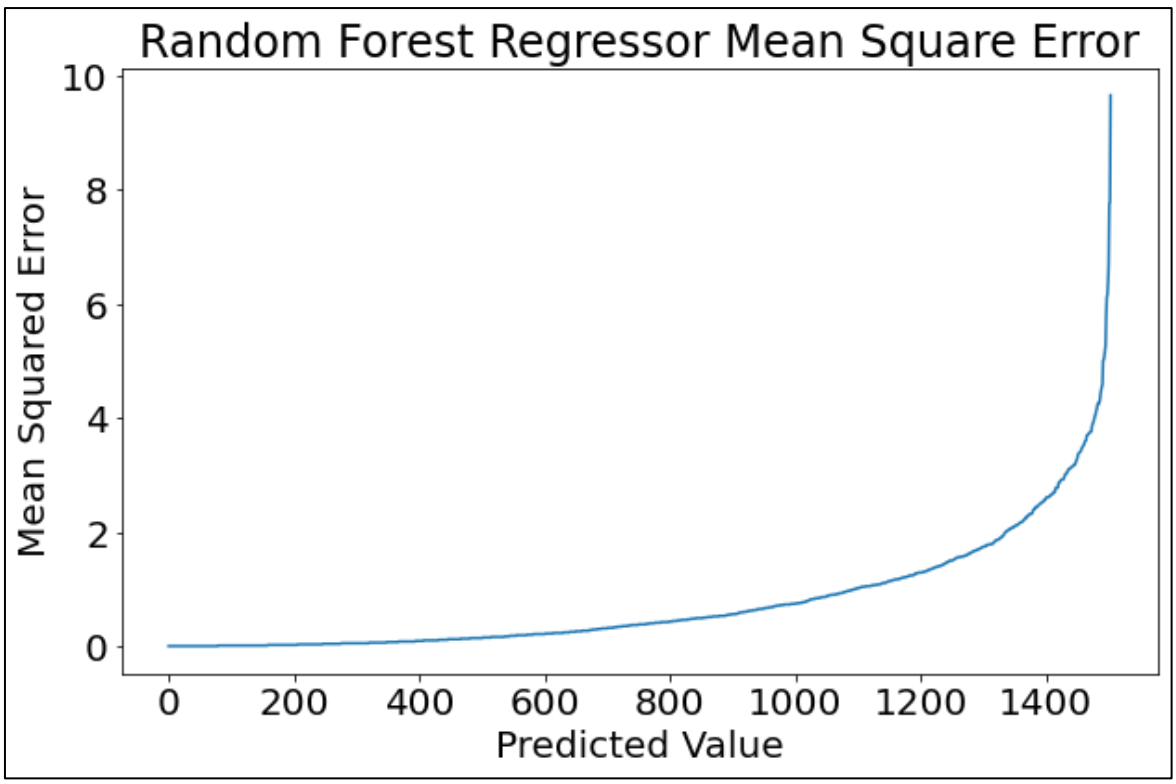


Figure 25: Mean Square Error graph for Random Forest Regressor

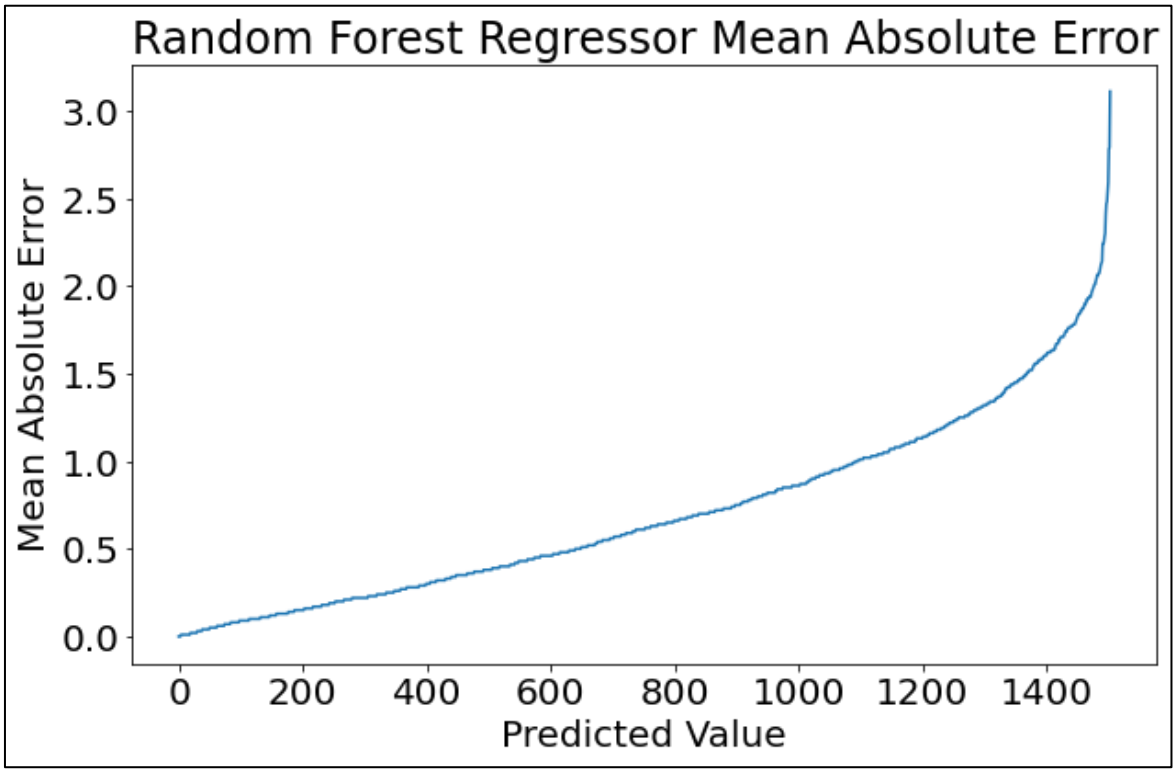


Figure 26: Mean Absolute Error graph for Random Forest Regressor

Chapter 05: CONCLUSIONS

5.1 Conclusion

After creating three different models for recommending movies to the user we observed that every feature of the movies plays a very significant and different roles in the recommendation of the movie. Where as our model 1 which used some regression algorithm to determine the best scored movie for the user displayed some very promising result using Linear Regression algorithm which gave us accuracy of 95.62%, at the same our model 2 and model 3 which used feature similarity with the content-based filtering algorithm to recommend the best suited movie for the user. Combined all the models provided very promising results for the recommendation system.

The modelling was done by using Python 3.8 and its necessary libraries. Code was performed by all the team members on their respective systems. More complex and advanced algorithms would be helpful in evaluating the performance of the model at a significant rate.

5.2 Application

There are several applications of the recommendation systems. Nowadays, there is a plethora of data that has been gathered on the web worldwide. Such huge levels of the textual information available are very necessary for the e-commerce platforms and other companies as well. This data is used for recommending entities to their users. Many applications like amazon, Netflix, and other social networking websites use recommendation systems. High accuracy levels for the predictions give high revenue generation for all these companies. Although this model will require a proper production and development, also to say an extensive level of testing before using it in real life, improvement can be done to make that possible.

5.3 Future Work

Till now we have used content-based filtering and some regression algorithm for the model and that has given quite high accuracy. In the future we are looking forward to evaluating the dataset using a different number of NLP algorithms to significantly improve the accuracy of the model by analysing the reviews and other important features of the

dataset. We would also try to train it using a much larger dataset having larger number of reviews, more textual subjectivities and more features

To get the optimum accuracy and precision of the model we will train the dataset on different machine learning and deep learning algorithms or hybridise algorithms. The less availability of the large data is also a barrier in this project. Using the various types of recommendation algorithms, we tend to make it a product website and deploy it as a working project.

References

[1] Andrea Chiorrini, Claudia Diamantini, Alex Mircoli, “Emotional and sentiment analysis of tweets using BERT”, 2021

[2] Manish Munikar, Sushil Shakya and Aakash Shrestha, “Fine-grained Sentiment Classification using BERT”, CoRR abs/1910.03474, 2019.

[3] Bing Liu, Hu Xu, Philip S. Yu and Lei Shu. “BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis”, arXiv:1904.02232, 2019.

[4] Kuat Yessenov, Sasa Misailovic, “Sentiment Analysis of Movie Review Comments”, 6.863 Spring, 2009.

[5] M.Govindarajan , “Sentiment Analysis of Movie Reviews using Hybrid Method of Naive Bayes and Genetic Algorithm”, International Journal of Advanced Computer Research. December-2013.

[6] V.K. Singh, R. Piryani, A. Uddin, P. Waila, “Sentiment Analysis of Movie Reviews”, In conference on IEEE-2013.

[7] B. Lakshmi Devi, V. Varaswathi Bai, Somula Ramasubbareddy and K. Govinda. “Sentiment Analysis of Movie Reviews”, SOET, 2018.

[8] Hamed AL-Rubaiee, Renxi Qiu, Khalid Alomar and Dayou Li, “Sentiment Analysis of Arabic Tweets in e-Learning”, uobrep 2016.

[9] Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075.

[10] Sarah Alhumoud, Asma Al Wazrah, “Sentiment Analysis Using Stacked Gated Recurrent Unit for Arabic Tweets”, IEEE Access, 9, 137176-137187,2021.

[11] Zainur Romadhon, Eko Sediyono, Catur Edi Widodo,”Various implementation of collaborative filtering-based approach on recommendation systems using similarity”,Kinetik, Vol. 45, No. 3, August 2020, Pp. 179-186.

[12] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," in IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734-749, June 2005, doi: 10.1109/TKDE.2005.99.

[[13] De Campos, L.M., Fernández-Luna, J.M., Huete, J.F., Rueda-Morales, M.A.: Combining content-based and collaborative recommendations: a hybrid approach based on bayesian net- works. Int. J. Approx. Reason. 51(7), 785–799, 2010.

[14] https://developers.google.com/machinelearning/recommendation/collaborative/basics

[15] Adomavičius,G.,Tuzhilin,A.:Toward Next Generation Recommender Systems:a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. 17(6), 734–749 (2005)

[16] https://searchenterpriseai.techtarget.com/definition/BERT-language-model

[17]https://towardsdatascience.com/sentiment-analysis-in-10-minutes-with-bert-and-hugging-face-294e8a04b671

[18] Thakkar P., Varma K., Ukani V., Mankad S., Tanwar S. (2019) Combining User-Based and Item-Based Collaborative Filtering Using Machine Learning. In: Satapathy S., Joshi A. (eds) Information and Communication Technology for Intelligent Systems. Smart Innovation, Systems and Technologies, vol 107. Springer, Singapore. https://doi.org/10.1007/978-981-13-1747-7_17

[19] Luis M. de Campos, Juan M. Fernández Luna, Juan F. Huete, Miguel A. Rueda-Morales "Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks", International Journal of Approximate Reasoning, revised 2010.

[20] Costin-Gabriel Chiru, Vladimir-Nicolae Dinu, Ctlina Preda, Matei Macri ; "Movie Recommender System Using the User's Psychological Profile" in IEEE International Conference on ICCP, 2015.

[21] Geetha G, Safa M, Fancy C, Saranya D; "A Hybrid Approach using Collaborative filtering and Content based Filtering for Recommender System", 2018.

[22] Rupali Hande, Ajinkya Gutti, Kevin Shah, Jeet Gandhi, Vrushal Kamtikar, "Moviemender - A movie recommender system", 2016.

[23] Eyrun A. Eyjolfsdottir, Gaurangi Tilak, Nan Li, "MovieGEN: A Movie Recommendation System", 2018.

[24] James Salter, Nick Antonopoulos; "CinemaScreen Recommender Agent: Combining Collaborative and Content-Based Filtering", University of Surrey, 2017.

Appendix

Calculate score for movies

```
m = metadata['vote_count'].quantile(0.90)
C = metadata['vote_average'].mean()
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    print(v,R,m,C)
    # Calculation based on the IMDB formula
    return (v/(v+m) * R) + (m/(m+v) * C)
# Define a new feature 'score' and calculate its value with `weighted_rating()`
q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
#Sort movies based on score calculated above
q_movies = q_movies.sort_values('score', ascending=False)
#Print the top 15 movies
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(20)
```

Linear Regression

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
X_new = q_movies_new[['vote_count', 'vote_average']]
Y_new = q_movies_new[['score']]
X_train, X_test, y_train, y_test = train_test_split(X_new, Y_new, test_size=0.33)
model = LinearRegression()
model.fit(X_train,y_train)
acc = model.score(X_test, y_test)
Y_pred = model.predict(X_test)
MSE = mean_squared_error(y_test, Y_pred)
RMSE = mean_squared_error(y_test, Y_pred, squared=False)
MAE = mean_absolute_error(y_test, Y_pred)
```

Decision Tree Regressor

```
from sklearn.tree import DecisionTreeRegressor
regr = DecisionTreeRegressor(max_depth=3)
```

```
regr.fit(X_train, y_train)
scoreacc = regr.score(X_test,y_test)
Y_pred = regr.predict(X_test)
MSE = mean_squared_error(y_test, Y_pred)
RMSE = mean_squared_error(y_test, Y_pred, squared=False)
MAE = mean_absolute_error(y_test, Y_pred)
```

Lasso Regression

```
from sklearn import linear_model
lassoReg = linear_model.Lasso(alpha=0.1)
lassoReg.fit(X_train,y_train)
scorelasso = lassoReg.score(X_test,y_test)
Y_pred = regr.predict(X_test)
MSE = mean_squared_error(y_test, Y_pred)
RMSE = mean_squared_error(y_test, Y_pred, squared=False)
MAE = mean_absolute_error(y_test, Y_pred)
```

Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
X, y = make_regression(n_features=4, n_informative=2, random_state=0,
shuffle=False)
rfr = RandomForestRegressor(max_depth=3)
rfr.fit(X_train, y_train)
accrf = rfr.score(X_test,y_test)
print(rfr.score(X_test,y_test))
Y_pred = rfr.predict(X_test)
MSE = mean_squared_error(y_test, Y_pred)
RMSE = mean_squared_error(y_test, Y_pred, squared=False)
MAE = mean_absolute_error(y_test, Y_pred)
```

Creating TF-IDF matrix of overviews

```
#Import TfIdfVectorizer from scikit-learn
from sklearn.feature_extraction.text import TfidfVectorizer
#Define a TF-IDF Vectorizer Object. Remove all english stop words such as
'the', 'a'
tfidf = TfidfVectorizer(stop_words='english')
#Replace NaN with an empty string
q_movies['overview'] = q_movies['overview'].fillna('')
#Construct the required TF-IDF matrix by fitting and transforming the data
tfidf_matrix = tfidf.fit_transform(q_movies['overview'])
```

```
#Output the shape of tfidf_matrix
tfidf_matrix.shape
```

Creating Cosine Similarity Table for TF-IDF matrix

```
from sklearn.metrics.pairwise import linear_kernel
# Compute the cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

Creating get recommendation function for Movie overviews

```
indices=pd.Series(q_movies.index,index=q_movies['title']).drop_duplicate
s()
def get_recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    idx = indices[title]
    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))
    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]
    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]
    # Return the top 10 most similar movies
    return metadata['title'].iloc[movie_indices]
```

Creating find genres function

```
cmn_list=[]
def find_genres(text):
    matches = re.finditer('name', text)
    matches_positions = [match.start() for match in matches]
    # print(matches_positions)
    matches_positions = [x+8 for x in matches_positions]
    # print(matches_positions)
    list_genre = []
    for i in matches_positions:
        idx = i
        gne = ""
        for j in text[idx:]:
            if(j=='}'):
                break
            gne=gne+j
```

```
        list_genre.append(gne[:-1])
        cmn_list.append(gne[:-1])
    return list_genre
idx_genres = []
for text in genere_specific.genres:
    idx_genres.append(find_genres(text))
cmn_list_new = set(cmn_list)
```

Marking true and false to genres in movie

```
genres = ['Fantasy', 'Romance', 'Music', 'Action', 'Adventure', 'TV
Movie', 'Horror', 'Science Fiction', 'Comedy', 'Documentary', 'Western',
'Foreign', 'Drama', 'Mystery', 'Crime', 'Animation', 'Family', 'Thriller',
'War', 'History']
for x in genres:
    test[x] = 0
drop_index = []
for ind in test.index:
    cat = test['new_gen'][ind]
    for x in cat:
        try:
            test[x][ind] = 1
        except:
            print(ind, " = ", test['id'][ind], " - ",
test['new_gen'][ind]," - ", cat, " - ",x)
            drop_index.append(ind)
drop_index = list(set(drop_index))
test.drop(drop_index, inplace = True)
```

Creating Cosine Similarity Table for Genre dataset

```
from sklearn.metrics.pairwise import linear_kernel
# Compute the cosine similarity matrix
cosine_sim = linear_kernel(dataset, dataset)
indices = pd.Series(data.index, index=data['title']).drop_duplicates()
```

Creating get_recommendation function for Movie Genres

```
indices=pd.Series(q_movies.index,index=q_movies['title']).drop_duplicate
s()
def get_recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    idx = indices[title]
```

```
# Get the pairwsie similarity scores of all movies with that movie
sim_scores = list(enumerate(cosine_sim[idx]))
# Sort the movies based on the similarity scores
sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
# Get the scores of the 10 most similar movies
sim_scores = sim_scores[1:11]
# Get the movie indices
movie_indices = [i[0] for i in sim_scores]
# Return the top 10 most similar movies
return metadata['title'].iloc[movie_indices]
```