

Early Success Prediction of Indian Movies using

Subtitles: A Document Vector Approach

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Vaddadi Sai Rahul

17BCE0230

Tejas M

17BCE2242

Under the guidance of

Prof. / Dr. Jothi K.R.

School of Computer Science and Engineering (SCOPE)
VIT, Vellore.



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

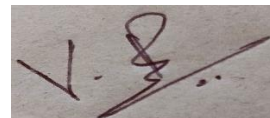
June, 2021

DECLARATION

We hereby declare that the thesis entitled “**Early Success Prediction of Indian Movies using Subtitles: A Document Vector Approach**” submitted by us, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by us under the supervision of **Prof. / Dr. Jothi K.R.**

We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore
Date : 04-06-2021



Signature of the Candidate

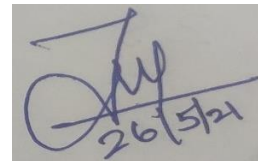
CERTIFICATE

This is to certify that the thesis entitled “**Early Success Prediction of Indian Movies using Subtitles: A Document Vector Approach**” submitted by **Vaddadi Sai Rahul & Reg. No 17BCE0230, Tejas M & Reg. No 17BCE2242, School of Computer Science and Engineering (SCOPE)**, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by him / her under my supervision during the period, 01. 12. 2020 to 04.06.2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 04-06-2021



Signature of the Guide

Internal Examiner

External Examiner

Head of the Department
Programme

ACKNOWLEDGEMENTS

It is my pleasure to express with deep sense of gratitude to Prof./Dr. Jothi K.R., Associate Professorship in Computational Intelligence, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Machine Learning.

I would like to express my gratitude to our Chancellor Dr. G. Viswanathan, Vice President Mr. G.V. Selvam, Vice Chancellor Dr. Rambabu Kodali, Pro Vice-Chancellor Dr. S. Narayanan, and Dean SCOPE Dr. K. Ramesh Babu for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. S. Vairamuthu, Head of Department (HOD) SCOPE, all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Vaddadi Sai Rahul
Tejas M

Executive Summary

Scientific studies of the elements that influence the box office performance of Indian films have generally concentrated on post-production elements, such as those discovered after a film has been completed or released, and notably for Bollywood films. Fewer studies have looked at regional film industries and pre-production factors, which are elements that are known before a decision to greenlight a film is made. This study looked at Indian films using natural language processing and machine learning approaches to see if they would be profitable in the pre-production stage. We extract Movie data and English subtitles (as an approximation to screenplay) for the top five Indian film industries based on their revenue contribution. Subtitle Vector (Sub2Vec), a Paragraph Vector model trained on English subtitles, was used to embed subtitle text into 50- and 100- dimensions. The proposed approach followed a 2-stage pipeline. In the first stage, ROI was calculated using aggregated subtitle embeddings and associated movie data. Classification models used the ROI calculated in the first step to predict a film's verdict in the second step. The optimal regressor-classifier pair was determined by evaluating classification models using F1-score and Cohen's Kappa scores on various hyperparameters.

	Page
	No.
Acknowledgements	iv
Executive Summary	5
Table of Contents	6
List of Figures	8
List of Tables	10
Abbreviations	11
Symbols and Notations	14
1 INTRODUCTION	15
1.1 Theoretical Background	15
1.2 Motivation	16
1.3 Aim of the Proposed Work	17
1.4 Objective(s) of the Proposed Work	17
2. Literature Survey	18
2.1. Survey of the Existing Models/Work	18
2.2. Summary/Gaps identified in the Survey	19
3. Overview of the Proposed System	29
3.1. Introduction and Related Concepts	29
3.2. Framework, Architecture or Module for the Proposed System(with explanation)	29
3.3. Proposed System Model (ER Diagram/UML Diagram/Mathematical Modeling)	57
4. Proposed System Analysis and Design	58
4.1. Introduction	58
4.2. Requirement Analysis	58
4.2.1.Functional Requirements	

4.2.1.1.	Product Perspective	
4.2.1.2.	Product features	
4.2.1.3.	User characteristics	
4.2.1.4.	Assumption & Dependencies	
4.2.1.5.	Domain Requirements	
4.2.1.6.	User Requirements	
4.2.2.	Non Functional Requirements	58
4.2.2.1.	Product Requirements	
4.2.2.1.1.	Efficiency (in terms of Time and Space)	
4.2.2.1.2.	Reliability	
4.2.2.1.3.	Portability	
4.2.2.1.4.	Usability	
4.2.2.2.	Organizational Requirements	58
4.2.2.2.1.	Implementation Requirements (in terms of deployment)	
4.2.2.2.2.	Engineering Standard Requirements	
4.2.2.3.	Operational Requirements (Explain the applicability for your work w.r.t the following operational requirement(s))	59
	<ul style="list-style-type: none"> • Economic • Environmental • Social • Political • Ethical • Health and Safety • Sustainability • Legality • Inspectability 	
4.2.3.	System Requirements	60
4.2.3.1.	H/W Requirements(details about Application Specific Hardware)	
4.2.3.2.	S/W Requirements(details about Application Specific Software)	
5.	Results and Discussion	61
6.	References	63

List of Figures

Figure No.	Title	Page No.
1	Architecture of Movie Success Prediction's System	29
2	Data Distribution of Indian Films.	30
3	Sentences per subtitle distribution.	31
4	Words per subtitle distribution.	32
5	Distributed Memory Architecture	33
6	Distributed Bag of Words Architecture	33
7	Paragraph Vector Architecture	33

8	Class Distribution of Train Data	38
9	Formula of Accuracy	53
10	Formulas for Precision, Recall and F1 score	54
11	Formula for Matthews Correlation Coefficient	55
12	Formula for Cohen's Kappa Statistic	55
13	UML Diagram for the proposed system	57

List of Tables

Table No.	Title	Page No.
1	Timeline of various approaches	18
2	F1 and Cohens Kappa Test data scores of optimal classifiers for different hyperparameter values	61
3	Optimal Regressors and Classifiers for different hyperparameter values	61

List of Abbreviations

ROI	Return on Investment
INR	Indian Rupees
CAGR	Compound Annual Growth Rate
W2V	Word 2 Vector
P2V	Paragraph 2 Vector
IMDB	Indian Movie Database
GDP	Gross Domestic Product
CART	Classification and Regression Tree
ANN	Artificial Neural Network
SVR	Support Vector Regressor
MIAS	Movie Investor Assurance System
APHR	Average Percentage Hit Rate
ELMO	Embeddings from Language Models
IPTV	Internet Protocol Television
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
DNN	Deep Neural Network
MLBP	Multi-Layer Back Propagation
MRPI	Fourth Generation
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
RAE	Relative Absolute Error
SVM	Support Vector Machine
KNN	K Nearest Neighbor
MLR	Multivariate Linear Regression

MLP-NN	Multi-layer Perceptron Neural Network
RF	Random Forest
SNS	Social Network Service
AIC	Akaike Information Criterion
GPR	Gaussian Process Regression
MLP	Multilayer Perceptron
SGD	Stochastic Gradient Descent
MPAA	Motion Picture Association of America
CBFC	Central Board of Film Certification
ISM	Independent Subspace Method
NLR	Non-Linear Regression
LSTM	Long Short-Term Memory
WOM	Word of Mouth
Doc2Vec	Document 2 Vector
PV-DM	Paragraph Vector–Distributed Memory
PV-DBOW	Paragraph Vector–Distributed Bag of Words
CBOW	Continuous Bag of Words
TMDB	The Movie Database
Sci-Fi	Science Fiction
U	Unrestricted
UA	Unrestricted with Caution
A	Adults only
MCC	Matthews Correlation Coefficient
CV	Cross Validation
MSE	Mean Squared Error

XGBR	Extreme Gradient Boosting Regressor
XGBC	Extreme Gradient Boosting Classifier
SVC	Support Vector Classifier
ABC	Adaptive Boosting Classifier
ABR	Adaptive Boosting Regressor
GBR	Gradient Boosting Regressor
GBC	Gradient Boosting Classifier

Symbols and Notations

ϵ	Epsilon (belongs to)
π	Pi (product)
Σ	Sigma (summation)
σ	Sigma (product)
\subset	Subset
α	Alpha (learning rate)
I	Identity Matrix

1. INTRODUCTION

1.1 Theoretical Background

With a market size of INR 18.6k crores (US\$ 2.7 billion) as of financial year 2019 [39], Indian film industry is one of the largest film industries worldwide. With a Compound Annual Growth Rate (CAGR) of 9%, the film industry's business is expected to reach INR 29.9k crores (US\$ 4.3 billion) in 2024 [39]. Although Indian film industry stands 4th in terms of global box office collection, it leads in the number of films produced and number of tickets sold every year. The number of Indian films certified has increased by about 89 percent in the last decade [39] and is expected to increase further. There are various sub-industries within the Indian film industry, segmented by languages. Bollywood, known as the Hindi language film industry, has the highest revenue share of 40%. Regional film industries together contribute 47% to the overall revenue followed by Hollywood films at 13%. Among the regional industries, Kollywood (Tamil language film industry) and Tollywood (Telugu language film industry) account for 14% and 10% of total revenue share respectively. Other languages' film industries, such as Mollywood (Malayalam), Sandalwood (Kannada), Bengali, Marathi, and others, are next in line in terms of box office receipts [39].

The Advertising industry for Indian cinema is projected to be INR 1,211 crores [39]. The fraction of films that generate good box office revenue and return on investment has been steadily increasing over the previous three to four years yet a considerable amount of films incur huge losses and fail to recover investment. These end up being categorized as Flop. The reason can be attributed to a report published by Microsoft [39] which says, because of the growing popularity of social media and messaging apps on smartphones, viewers' attention spans are dwindling. While movies are considered stress relievers, and customers have a plethora of options, it is critical to keep the audience focused in something worthwhile. A recent increase in content-based films suggests that films with compelling storylines have become more popular. Production businesses and studios are investing in content-driven films, while distributors and exhibitors are promoting their screening. To address this, On-demand availability of content-driven movies are being used by online video streaming platforms such as Netflix and Amazon Prime to synchronize with the changing viewer mindset.

For low-budget films or films that aren't suited for a theatrical distribution, the online streaming sectors serve as a risk-reducing choice. However, there has been discussion over how this would influence movie stars' brand value, because a movie is best enjoyed both visually and auditorily in a theatre.

Considering these factors, determining the performance of movies early on is a challenging task which certainly aids production studios in their decision making of whether or not to fund a film. The current research aims at estimating if a film will be 'profitable' (successful) or 'unprofitable' (unsuccessful) for the top 5 Indian film industries; Bollywood, Kollywood, Tollywood, Mollywood and Sandalwood at the point of green-lighting. Guided by textual knowledge from subtitles and user extracted content-based features we determine the ROI. This ROI is then, used to estimate a film's verdict i.e., to conclude if a film will be profitable or not.

The economic significance of our approach lies in its contribution towards both the Indian and International market. This is the first study to look at all of India's major film industries. Furthermore, only a few studies used screenplays, and those that did had a smaller sample size. In comparison to other Indian and international methods, our study uses English subtitles, that accurately resemble screenplay data, on a larger sample scale. We include ROI in our independent features list instead of calculating a continuous box office revenue parameter. This is thought to provide more direct details about the film's financial success. Instead of using word embeddings, we used the Subtitle Vector (Sub2Vec) paragraph vector model to scale through textual data, which is efficient and accurate.

1.2 Motivation

The Indian Cinema Industry spends INR 1,211 crores (US\$ 163 million) on advertising and is expected to reach INR 1,400 crores (US\$ 189 million) by the end of financial year 2021. Owing to the enormous number of movies (~2400) produced every year and changing viewer mindset, there has been an emphasis on early prediction of box office performance. It is necessary for the producers and/or the production houses to assess the economic potential of a movie at the point of green-lighting. Previous studies, concerning how to accurately predict the revenue sales of Indian films, focused primarily on films which either have completed shooting and/or released in local theaters; particularly more on Hindi films (Bollywood) and less on regional language films.

1.3 Aim of the proposed Work

To accurately predict whether an Indian movie will be a box office success or failure in the pre-production phase i.e., at the point of green-lighting a film.

1.4 Objective(s) of the proposed work

Specifically, the objectives include:

- Categorize Indian films as ‘successful’ or ‘unsuccessful’ based on the textual (subtitles) and content-based (movie data) features using machine learning and natural language processing.
- Apply the proposed methodology on the dataset.
- Evaluate the performance of regression and classification algorithms on the dataset by metrics such as mean-squared error, cross validation, f1-score and cohen’s kappa score.
- Select the optimal regressor-classifier pair for the given dataset.

2. Literature Survey

There are three sorts of methods for predicting box office performance, based on their timeline of prediction as shown in Table. 1.

- Early Predictions – predictions made before the actual shooting of movie has been started.
- Intermediate Predictions – predictions made after the completion of a movie's shooting and before its theatrical release.
- Late Predictions – predictions made after a movie's release in cinema theatres.

Table. 1. Prediction Timeline of Various Approaches

Early Predictions	Intermediate Predictions	Late Predictions
Kaur & Nidhi (2013)	Sharda & Delen (2006)	Neelamegham & Chintagunta (1999)
Eliashberg et al. (2014)	Mazurowski et al. (2006)	Simonoff & Sparrow (1999)
Chaudhari et al. (2016)	Zhang & Skiena (2009)	Lee & Chang (2009)
Hunter et al. (2016)	Reddy et al. (2012)	Kim et al. (2015)
Lash & Zhao (2016)	Parimi & Caragea (2013)	Chen et al. (2016)
Shreya et al. (2019)	Apala et al. (2013)	Hur M et al. (2016)
Kim et al. (2019)	Pangarker et al. (2013)	Magdum et al. (2017)
Reddy V G et al. (2020)	Mestyán et al. (2013)	Quader N et al. (2017)
Liao et al. (2020)	Selvaretnam et al. (2015)	Kim et al. (2017)
Hunter et al. (2014)	Taneja et al. (2016)	Dhir & Raj (2018)
	Subramaniaswamy et al. (2017)	Ru et al. (2018)
	Zhou et al. (2017)	
	Ruhrländer et al. (2018)	
	Yen & Zhou (2018)	
	Verma et al. (2019)	
	Ahmad et al. (2020)	
	Wang et al. (2020)	

I. Early Predictions

(Chaudhari et al.; 2016) obtained an accuracy of 94% using language and aspect ratio, in addition to Facebook likes of actor and actress and production budget for determining the success rate of films. Additional costs associated with a pre-production phase like hiring crew members and location scouting however, were not included. (Shreya et al.; 2019) predicted whether a movie will be financially successful before completion, using categorical data extracted from *IMDB* Bollywood movies from 1990 to 2018. Results of three classifiers were compared to ultimately decide the best algorithm for predicting success or failure. Since the proposed model's findings were not mentioned, outcome could not be evaluated. (Kaur & Nidhi; 2013) used neural networks as classifiers and trained them using the *Levenberg-Marquardt* algorithm to achieve an accuracy of 93.3% on Bollywood motion pictures. (Kaur & Nidhi; 2013) converted the attributes into equivalent numerical quantities by considering average statuses of each attribute's previous ten movies. However, the results were based on only 111 instances. Since an artificial neural network is a black box system, its reasonings are difficult to comprehend, making it difficult to convince a film's producer of its box office potential. (Hunter et al.; 2016) employed a regression model to study the effect of network text analysis on opening weekend's box office performance. The film scripts were represented as a map of interconnected components called as *multi-morphemic compounds (MMCs)* (Hunter; 2014), and the size of the largest cluster of mutually reachable nodes referred to as the *main component*, was used as a measure. A possible drawback is scripts having a larger size of the main component are preferable to those having a smaller size. If that were the case, writers would have introduced numerous MMCs in the script with the hopes of a higher box office take, which is not the case. (Eliashberg et al.; 2014) assessed the box office performance in terms of return on investment (ROI) using movie script and production budget. However, two experts in film study were required to read scripts to gather information for genre and content variables. A possible reason for low mean-squared error might be the small sample size of 300 film scripts, compared to the relatively large number of predictor variables. Due to the curse of dimensionality problem, it can lead to overfitting and hence, requires further data analysis. (Lash & Zhao; 2016) used text mining techniques and social network analysis to develop *Movie Investor Assurance System (MIAS)* that predicts the success of movies at the box office before the pre-production stage. The three main features considered include: 'who' was in the cast, 'what' the movie was all about and 'when' the movie was planned to release.

One possible drawback was that (Lash & Zhao; 2016) considered only the plot synopses rather than the entire film script. Another limitation is that it is practically impossible to predict the actual amount of money a movie makes as some movies make a lot of money before its release from merchandise sales. This aspect was totally ignored. (Reddy V G et al.; 2020) made several plots to understand the relationship between variables such as genre, budget, cast, crew etc. The models used include Linear Regression, Random Forest and XGBoost. Linear regression gave an RMS value of 2.4236, Random Forest Regression a value of 2.2127 and XGBoost a value of 2.3558. Random Forest turned out to be better predictor. (Liao et al.; 2020) focused on China which has the second largest box office in the world. The author only considered features that were directly relevant to the subject of the film and completely ignored features related to word-of-mouth data collected from various social media platforms. A stacking fusion model was developed by combining Random Forest, Extreme and light gradient boosting, KNN and stacking model fusion theory. The model's effectiveness was assessed using Average Percentage Hit Rate (APHR). 2 types of APHR i.e., Bingo & 1-Away were used to determine the model accuracy. With a Bingo accuracy of 69.16% and a 1-Away accuracy of 86.46%, the stacking model outperformed all single prediction models. According to the findings (Liao et al.; 2020), star influence played the most significant role in predicting a film's box office revenue. One disadvantage of this approach is that a film may be developed from an intellectual property with a large fan base. As a result, during the early stages of production, social media data becomes critical in projecting the box office of a film. (Kim et al.; 2019) proposed a deep learning approach to predict box office success of movies using *Embeddings from Language Models (ELMO)* embedding and the sentiment scores of sentences based only on the textual summary of a movie plot. By obtaining movie plot summaries and their accompanying review scores from the CMU Movie Summary Corpus, evaluation datasets were created to test the efficacy of the proposed technique. All the plot summaries were obtained from Wikipedia. ELMO embedding and two combined deep learning models, merged 1D CNN network and merged residual bidirectional LSTM network, were developed. The majority class baseline for the combination of audience and critics scores is F1 of 0.53 for “not successful,” and 0 for “successful”. For predicting ‘successful’ movies, the proposed model received a maximum F1 score of 0.68, while for predicting ‘not successful’ movies, it received 0.70. Forecasting films that were “not popular” or “not successful” performed better than predicting films that were “popular” or “successful”. Internet Protocol television (IPTV) content providers like Netflix can benefit from predicting failure films.

As a result, the proposed method can be utilized to filter out unappealing content to users. The only drawback with this model was that thriller and comedy genres were not considered.

II. Intermediate Predictions

(Verma et al.; 2019) used the number of screens, IMDB and music ratings for predicting the success of Bollywood movies, categorised as hit or flop. By using a logistic regression model, (Verma et al.; 2019) achieved an accuracy of around 80%. However, the dataset was confined to a small sample size of only 116 movies. One of the major predictors, the production costs (Pangarker & Smit; 2013), were overlooked. (Taneja et al.; 2016) determined the success quotient of movies using sales revenue. The categorical parameters used for estimating the box office performance were quantitatively approximated by *averaging* over the total sales revenues corresponding to each parameter. However, a pre-release rating was defined as a criterion for assessing box office performance, which is difficult to estimate before a film's release. External sources of revenue like television broadcasting and theatrical rights were not considered, which usually play a significant role in revenue estimation. Artificial neural networks were employed by (Sharda & Delen; 2006) to forecast the commercial success of movies. A film's financial success was determined by its box office receipts, which were classified into nine categories ranging from flop to blockbuster. To better understand the relationship between the inputs and outputs, sensitivity analysis was used. Several functions implemented by the neural network to reliably estimate training patterns are a drawback of this approach (Mazurowski & Szcwoka; 2006). Production costs, which play a major role in revenue estimation (Pangarker & Smit; 2013) [8], were not considered. (Ruhrländer et al.; 2018) introduced a *user-centered* approach for predicting box office results. A three-stage prediction pipeline was used. The first stage involved using matrix factorization to generate user and movie profiles, the product of which yielded a predicted rating of j^{th} movie by i^{th} user. The second stage, combined meta-data and predicted user ratings from the first stage to output the likeliness of a user watching a movie. The last stage aggregated the predictions obtained from the previous step to output estimated revenue. One shortcoming of this method is that since it depends on number of users, its computationally expensive. (Zhang & Skiena; 2009) were able to accurately predict movies gross using quantitative data from *online news articles*. (Zhang & Skiena; 2009) employed a regression model for low grossing movies and a K-nearest neighbor (KNN) classifier for high grossing movies. Nonetheless, it is stated that the proposed approach targeted high-grossing films due to increased visibility through movie articles and revenue generation.

For an unseen movie, there is a possibility of both the classifier and regressor to yield better results, leading to a contradiction of whether the movie would perform better or worse. (Subramaniaswamy, V., et al.; 2017) took into account the effect of variables such as trailer views, time of release and Wikipedia page views to determine the revenue of movie by using both Multiple Linear Regression and Support vector machine (SVM). SVM gave an accurate classification rate of 56.52%. One drawback with this model was the consideration of only fewer features as independent variables. (Parimi & Caragea; 2013) utilized graph networks between movies and the resulting movie network was used along with transductive algorithm to construct the features required for classification. Three alternative weighing schemes were used by (Parimi & Caragea; 2013): (i) a fixed weight of '1' (ii) radial basis function kernel (iii) custom weighing technique to describe the network with an adjacency matrix. In comparison to the other two weighing techniques, the bespoke method of producing weights offered significantly superior outcomes. The modelling was done using Logistic Regression and Random Forest. (Apala et al.; 2013) made use of data mining tools to predict the potential box office performance of a movie by collecting data from various social media handles like YouTube, Twitter and IMDb. The box office prediction was based on the movie database, Twitter followers count, analysis of the sentiments of YouTube comments. The author used 3 classes for prediction i.e., Hit, Neutral and Flop. *Weka's K-Means clustering* was used for prediction. A film's box office success is strongly influenced by the popularity of its leading lady; sequels are even more successful; and a film in an unpopular genre with an unpopular star is a sure-fire recipe for a box office flop. (Reddy et al.; 2012) analysed the tweets in Twitter and the hype surrounding a movie prior to its release to determine its performance at the box office. Using a web crawler, the initial step was to determine the amount of tweets connected to a movie. The total quantity of unique users and relevant tweets were then calculated hourly. The final step was to determine the hype and reach factors for calculating the final hype. All these tasks would be performed for 7 days prior to the release of a movie. (Mestyán et al.; 2013) observed the activity level of online users in the Wikipedia page corresponding to a particular movie before its release. The authors considered the following measures for their prediction: (i) V: number of views of the article page, (ii) U: number of users (Number of editors who had contributed to the article), (iii) E: number of edits made by the editors, (iv) R: collaborative rigor, (v) T: The number of theatres. The box office income of a film was predicted using a multivariate linear regression model. The box office revenues accumulated by the movies showed a Bimodal distribution.

The model was trained using different combinations of the predictor variables and $R^2(t)$ i.e., Coefficient of determination was calculated using the 10-fold cross validation. The model that used $\{T\}$ is generally seen as state-of-the-art model in the industry, but the model that used only $\{V\}$ performed same as the later. Also $\{U, T\}$ and $\{V, T\}$ performed better than the above two. On considering $\{V, U, R, E, T\}$, highest $R^2(t)$ of 0.77 (a month before the release of movie) was produced. (Mestyán et al.; 2013) also tried to use features such as time span between the creation of the article, release time of the article, length of the article, but these were not useful in predicting the box office result of the movie. The $R^2(t)$ value also reached 0.94 a few days before the release of a movie. The model gives more accurate prediction for highly successful movies and less accurate results for less successful movies. This is mainly due to the difference in the amount of data available for each of these classes of movies. The accuracy of this model can be improved by adding neural networks on parameters such as the controversy measure of article. (Zhou et al.; 2017) developed a system using multimodal deep neural network. The author built a CNN to extract features from movie posters. There were ten convolutional layers and four pooling layers in the CNN. The activation function used was ReLU. This multimodal DNN was compared with SVM, Random Forest, Multilayer BP Neural Network. DNN performed better mainly due to 2 reasons: network depth and dropout technique. The author concluded that the model's prediction performance improved by integrating movie poster attributes learned by CNN as input into the multimodal DNN. Rather than relying just on traditional data, Multimodal DNN made use of movie poster content. In addition, being able to visualize the learnt features of movie posters serves as a reference for future movie poster design. The only drawback is that this model does not take into account any movie audio or video data. (Yen & Zhou.; 2018) developed a system that make use of the features that are extracted from the movie posters by CNN along with other features to predict the box office revenue of a movie. Deep Neural Network (DNN) architecture was constructed by considering features such as genre, duration, star value, budget, movie poster etc. CNN was constructed to act as a feature extractor for movie posters. Genetic operators such as Genotype design, Population initialization, Crossover, Mutation, Evaluation and Selection were employed. Performance of the model was determined using APHR. DNN performed better than SVM, Random Forest, MLBP. The main drawback is high computation cost, applying this for other multimodal prediction problems and genetic operators designing. (Ahmad et al.; 2020) develops a model that predicts the box office revenue of a movie by extracting peoples trailer reviews from social media sites like YouTube. The main objective of this model was to determine the purchase intention using YouTube trailer reviews.

This model gave a relative error of only 29.65% as compared to three baseline models. The author proposed a new feature called the WLDratio, which stands for weighted like-to-dislike ratio, and is used to describe the YouTube like function. (Ahmad et al.; 2020) also proposed WPNratio i.e., Weighted Positive-to-Negative sentiment ratio and MRPI i.e., Movie Review Purchase Intention, to represent the sentiments feature and calculate the purchase intention correspondingly. There are 2 main tasks in the model. First is to extract the YouTube trailer reviews which is known as the MRPI approach and second is to predict the box office revenue. Budget, number of reviews, number of views, etc., comprise independent variables. Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Relative Absolute Error (RAE), and k-fold cross validation were used to determine the model's correctness. PI and Review count features achieved the lowest error result. The model's accuracy was determined using four prediction algorithms: SVM, MLP-NN, MLR, and RF. MLR gave the best results with the least error followed by SVM and RF. MLP-NN gave the worst result. The only drawback with this model is ignoring the reply feature in YouTube reviews which might have helped to increase the accuracy a bit more. To estimate the box office sales of movies using big data, (Wang et al.; 2020) proposed feature learning, prediction and ranking methods. The two types of feature learning models include: (i) a heterogenous network embedding model that learns the latent representation of directors, actors, companies. It collectively captures their cooperation relationship. (ii) deep neural network-based model that captures the high-level representation of the quality of a movie from their trailers. The metrics used to determine the performance of the model include accuracy, mean absolute percentage error, Normalized Discounted Cumulative Gain. All the methods gave accurate results when the box office revenue of a movie was more than \$12.3M. But when the revenue was less than \$12.3M it produced a relatively high error. The only drawback with this approach was that the textual data about movies were completely ignored. None of the research mentioned, tried to measure the box office success in terms of profit, which would be a better estimate compared to sales (Selvaretnam & Yang; 2015). Another apparent drawback in this study is that the entire shooting of a movie will have been finished by then, making it useless for determining whether or not to proceed with production.

III. Late Predictions

(Magdum et al.; 2017) predicted the success (categorical) and the sales performance (numerical) of movies by mining online reviews, tweets and past box office revenues. (Neelamegham & Chintagunta; 1999) employed a hierarchical Bayes formulation of the Poisson model to predict the first week viewership in domestic and international markets.

(Simonoff & Sparrow; 1999) performed a regression analysis on variables available post the first weekend release of films; such as initial weekend gross and opening screens, to predict the box office performance. By relating the size of audience as an equivalent measure for predicting gross revenue, (Lee & Chang; 2009) used Bayesian Belief networks to predict the box office outcome. (Chen et al.; 2016) performed a non-linear modelling of count-based and content-based features from microblog data to accurately estimate weekly box office performance. (Kim et al.; 2015) made use of Social Network Service (SNS) data and ML algorithms. Three sequential forecasting model were used for predicting the box office revenue of a movie i.e. (i) prior to the release (ii) one week after the release (iii) two weeks after the release. The SNS data include weekly number of total positive, negative, emotional mentions in various social networking sites. Three forecasting models proposed are as follows: (i) *Model R*: predicts box office revenue during first theatrical week. Uses 25 variables as input and 12 SNS related data was considered, (ii) *Model W1*: predicts box office revenue during second theatrical week. Uses 32 variables as input and 16 SNS related data was considered, (iii) *Model W2*: predicts box office revenue during third theatrical week. Uses 38 variables as input and 20 SNS related data was considered. Most important variables were selected using the *Akaike information criterion (AIC)*. The models were developed using four regression algorithms. The 4 regression algorithms used were: Multivariate Support Vector Regression (SVR), k-Nearest Neighbours (k-NN), Linear Regression (MLR), and Gaussian Process Regression (GPR). The performance of a linear algorithm like MLR was not at par with those of the other three non-linear algorithms. Also, a combination of GPR, k-NN and SVR was used to increase the accuracy. When making a forecast before a film's release, the total number of mentions is significant, but the number of emotional remarks becomes essential after the film has been released. The baseline model utilized in previous similar publications was Multivariate linear regression, which primarily employed screening-related variables. Best accuracy was obtained when all the 3 non-linear models were combined into a single predictive model and used both screening and SNS data. (Quader N et al.; 2017) used 7 machine learning algorithms: Gaussian Naïve Bayes, Stochastic Gradient Descent (SGD), Support Vector Machine (SVM), AdaBoost, Multilayer Perceptron (MLP), Logistic Regression, and Random Forest, to determine the box office revenue. Best accuracy in prediction was given by Multilayer perceptron neural network. Out of 755 movies, multilayer perceptron correctly classified 442 movies. The model was trained using both pre-release and post-release features, thus total 15 features were used.

The features that were considered include Motion Picture Association of America (MPAA) ratings, Rotten tomato, meta centric and IMDb ratings, the star power of both the actor and director, month of movie release, budget of movie and the number of screens. Logistic Regression is useful when there is small dataset. It did well with noisy data, had low variance, and reduced overfitting. SVM produced accurate results in case of multiclass classification. The main disadvantage of this model was that it frequently predicted one class lower than the true value. MLP gave an accuracy of 89.67%. Author suggests that including country GDP as an attribute would improve the efficiency of the model. (Dhir & Raj; 2018) devised a methodology for predicting the IMDb score of a movie which tells whether a movie will be a success at the box office or not. Out of all the algorithms used, Random Forest gave the best accuracy. Also, it has been noticed that the amount of Facebook likes, critics for reviews, voted users, movie's runtime, and gross collection of a movie strongly affected the IMDb score. The two most successful genres for success at the box office were Drama and Biopic. The various classifiers used include: K-Nearest Neighbours, AdaBoost, Random Forest, Support Vector Machine (SVM), Gradient Boost. The best accuracy was given by Random forest (61%), followed by Gradient Boost (56.68%). Ada boost gave an accuracy of 49%, SVM 45% and KNN the lowest accuracy. The author has noted that the number of audiences is one of the important features for a movie to be successful. (Hur M et al.; 2016) devised a methodology to get the sentiment of users from a movie review and use it as an input variable for feeding it into non-linear machine learning algorithms like Artificial Neural Network (ANN), Support Vector Regression (SVR), Classification and Regression Tree (CART). Also, an Independent Subspace method (ISM) is applied. Three input variables are defined by the author: Motion picture related variable, External variable, Audience related variable. Motion picture related variables include nationality, MPAA rating, director popularity, actor popularity, distributor power, sales power etc. External variables include screen ratio, seasonality etc. Audience related variables include average rating, reviews, number positive, neutral and negative opinion about the movie. (Hur M et al.; 2016) built 6 forecasting models. Model W01 forecasted the first week's box office. Used all of the information gathered during the pre-release period. Model W02 forecasted second-week box office revenue, while Model W03 forecasted third-week box office revenue. Model W12 (which can additionally take into account audience reviews and previous week's box office income) forecasted total box office income till the second week, and so on. CART was outperformed by ANN and SVR. (Quader N et al.; 2017) made use of SVM, Neural Network and NLP. In total 15 features were considered like Budget, Release month, Director star power etc.

The accuracy of the pre-released features was 84.1%, while the accuracy of the combined pre-released and post-released features was 89.27% as computed by Neural Network. SVM gave an accuracy of 83.44% for the pre-released features and an accuracy of 88.87% for all the features combined. According to the research, the most essential criteria that assist in determining the profit are the budget, IMDB votes and the number of screens. A drawback in their approach was that (Quader N et al.;2017) were unable to reduce the number of target classes, and the SVM suffers from segregating the data points precisely because of the problem of data overlapping. Thus, SVM was unable to find out the correct hyperplanes. Another possible drawback is the exclusion of features like Number of tickets sold, Political condition and economic stability of the country at the time of release of the movie, GDP rate. (Ru et al.; 2018) proposes a model called Deep-DBP which is an end-to-end deep learning model for predicting the box office revenue of a movie. The Deep-DBP consist of static characteristics component and the temporal component. The static component integrates the static characteristics to improve the prediction. The temporal component employs LSTM to investigate the temporal relationships between movie data points. The key benefit of utilizing Deep-DBP is that it solves the problems caused by ANN models and ARIMA such as non-linear relations and multi variable problem. Another benefit of employing this model is that it can solve problems with short time series prediction. It also gives successful results when dealing with multi-view data and multi-source data. The prediction error reduces by 7% when static characteristics component is added. Deep-DBP gave a prediction error of only 30.1%. Only 114 movies were considered from the Chinese film market. The effectiveness of the model was determined using MAPE. Deep-DBP was able to accurately predict the 21 days movie box office revenue accurately. The model gave an accuracy of 37% when only temporal components were considered but the accuracy increased by 7% when static characteristics component were also considered. (Kim et al.; 2017) developed a system considering the competition and word of mouth related factors along with machine learning based NLR algorithms to determine the success of movie in the Korean film market. When both the competition environment and the WOM effect were taken into account, the model's forecasting accuracy improved. The early phases of the box office score were discovered to be heavily affected by the competitive atmosphere. But while forecasting the total box office score WOM had a greater influence over the competition environment. Machine learning based NLR algorithms achieved higher accuracy as compared to traditional MLRs. The drawback with this approach is that the forecasting models were applied only to the Korean film market rather than markets in other countries also.

Thus, the trends displayed by this model can't be universally accepted. The fact that a film's opening week generated around 25% of overall box office earnings (Simonoff & Sparrow; 1999), making it too late to influence the decision to bankroll a film. This is a flaw in research that relied on forecasts after a film's release.

3. Overview of the Proposed System

3.1 Introduction and Related Concepts

Overall system involves four phases. The first phase begins with data collection. The next phase involves data pre-processing, followed by modelling and evaluation.

Related Concepts – Web Scrapping, Natural Language Processing, Machine Learning.

3.2 Framework, Architecture or Module for the Proposed System (with explanation)

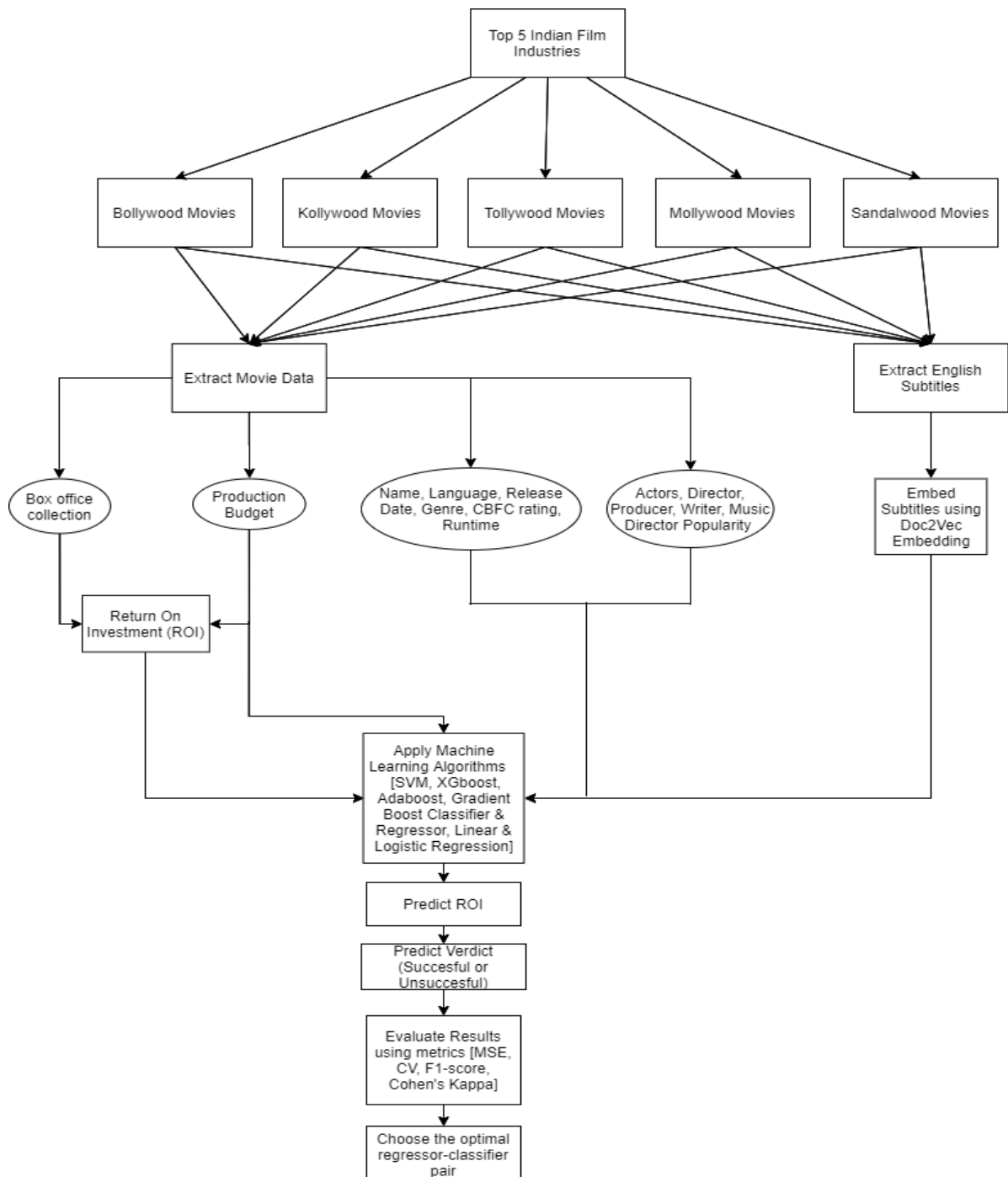


Fig. 1. Architecture of the Movie Success Prediction's System

The methodology utilized comprised of the following steps:

- Data Collection
- Data Transformation
- Application of models
- Evaluation of results

3.2.1 Data Collection

The data used for this research consisted of over 2700 instances of movie data and English subtitle files. The subtitle files in '.srt' format were collected from the websites *OpenSubtitles.org*, *Subscene.com*, *Subdl.com*, *Yifysubtitles.org*, *CinemaSubtitles.com*. The movie data collected consisted of 14 attributes namely movie name, language, runtime, budget, worldwide box office collection, release date, genre, actors popularity, music director popularity, movie director popularity, writer popularity, producer popularity, central board of film certification rating and verdict. Most of the movie data were scrapped while some had to be recorded manually from the websites *IMDb*, *TMDb*, *Wikipedia*, *YouTube*, *India.com*, *Sacnilk.com*, *Movieboxofficecollection.com*, *Boxofficemojo.com*, *Cinestaan.com*, *Mtwikiblog.com*, *Filmibeat.com*, *Behindwoods.com*.

Data Distribution Of Top 5 Languages In Indian Films

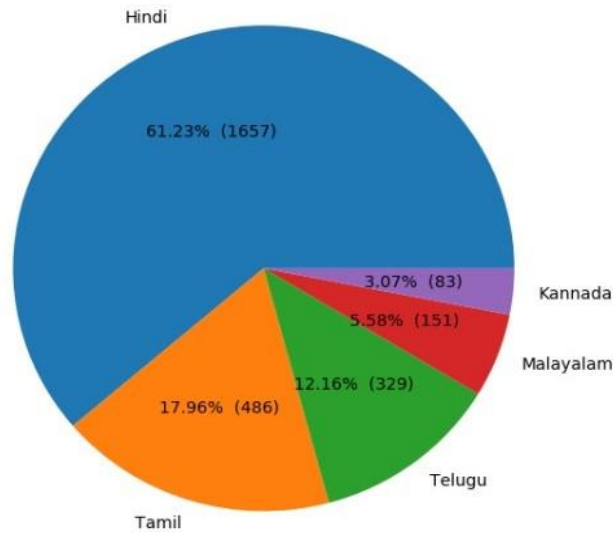


Fig. 2. Data Distribution of Indian Films

The distribution of data collected as depicted in Fig. 4. is as follows – Bollywood movie data accounts for ~61% (1657), Kollywood and Tollywood account for ~18% (486) and ~13% (329) respectively. Mollywood and Sandalwood account for ~6% (151) and ~3% (83) respectively.

3.2.2 Data Transformation

3.2.2.1. Subtitle file data

3.2.2.1.1. Pre-processing Phase

Over 2700 subtitles files were parsed into text using the *srt* library in Python. Subsequently the obtained text had to be converted into words. Standard NLP pre-processing techniques such as lowercase conversion, tokenization, stop words removal, non-English words removal and lemmatization were used to obtain the words list from the text. Lemmatization was preferred over stemming as it takes each word and reduces it to its simplest form, which is a word that can be found in the dictionary. For example, according to context the word “following” can either be a noun, verb or adjective. If it is used as an adjective or verb then lemmatization will return “following”, and if it is used as a verb then will return “follow”. Whereas Stemming will return “follow” in all cases by removing the “ing” suffix.

Next, to obtain unique words list for each subtitle, the words list was transformed into a set. The input for training the Subtitle Vector (Sub2Vec) model is a list of *TaggedDocument* objects. The *TaggedDocument* objects were generated by assigning a tag to each subtitle’s unique words list. We ended up with over 2400 subtitles after the initial pre-processing stage.

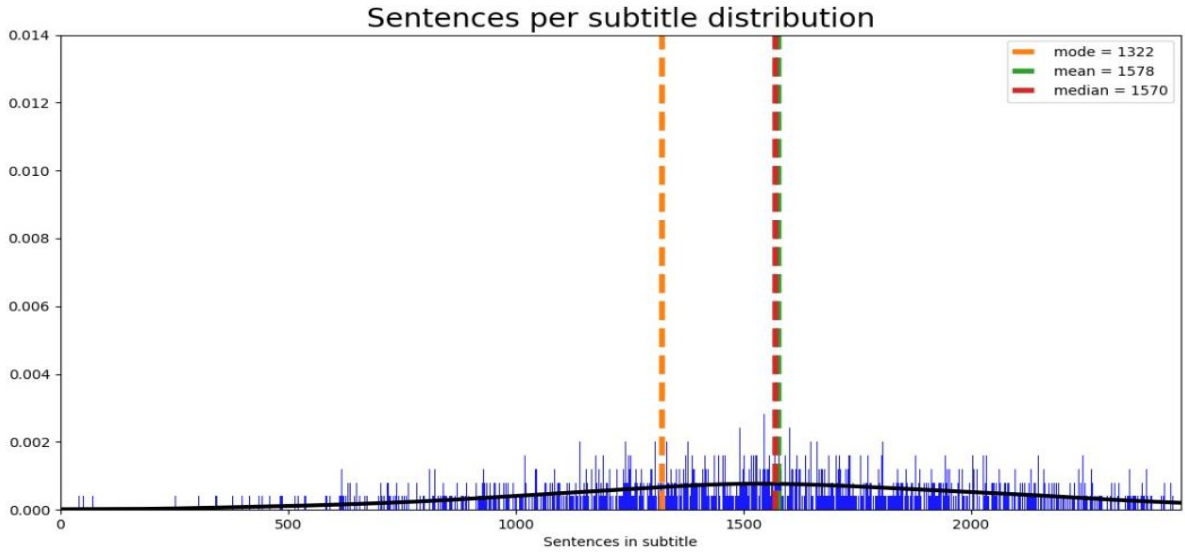


Fig. 3. Sentences per subtitle distribution

Fig. 5. represents the distribution of sentences per subtitle. In total we had around 4 million sentences, averaging over 1500 per subtitle.

Similarly, Fig. 6. represents the distribution of words per subtitle. We had a total of around 2.6 million unique words. The average number of words amounted to slightly over a 1000 per subtitle file.

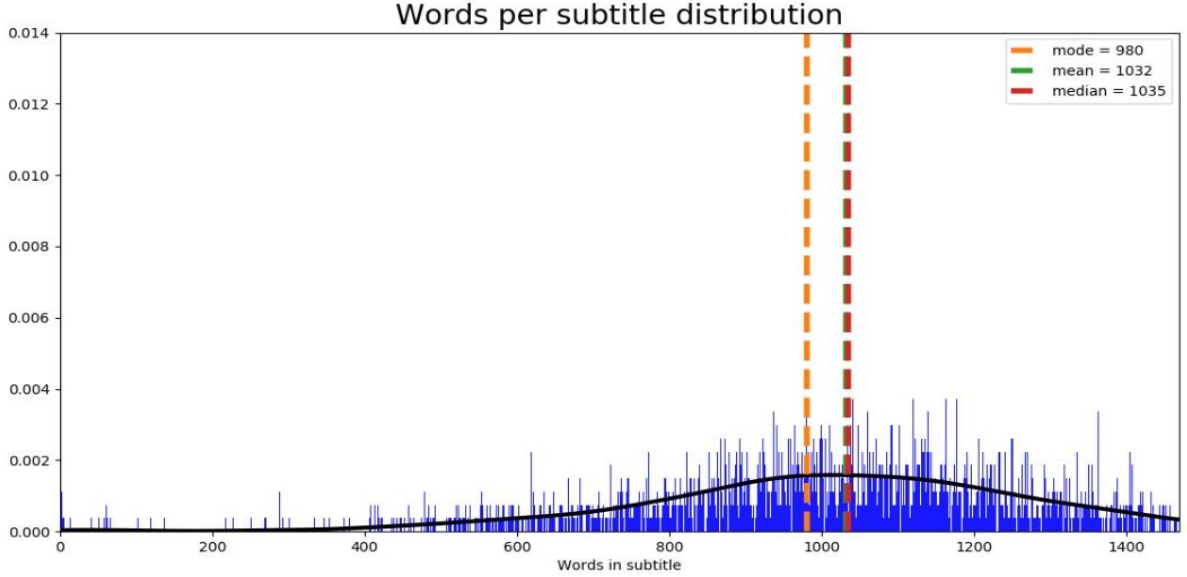


Fig. 4. Words per subtitle distribution

3.2.2.1.2. Training Phase

The Sub2Vec model was trained using the tagged unique words list. Training was performed for vector sizes of 50 and 100 for different epochs.

3.2.2.1.2.1 Paragraph Vector Model

Paragraph Vector (P2V/Doc2Vec) model is an unsupervised approach for learning distributed representations of text fragments. It's a modification of the standard Word2Vec (W2V) model. The aim of P2V is to learn a numerical representation of either documents or documents and words. P2V can be represented in two ways: Distributed Memory (PV-DM) and Distributed Bag of Words (PV-DBOW).

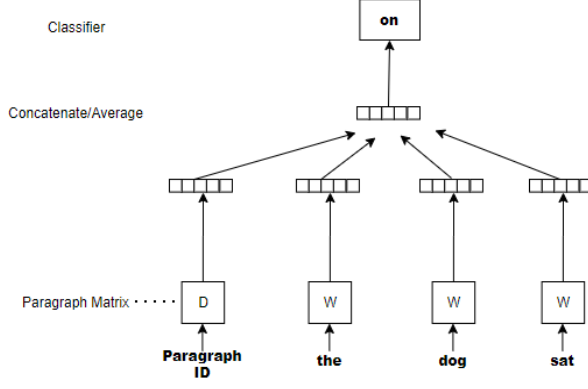


Fig. 5. Distributed Memory Architecture

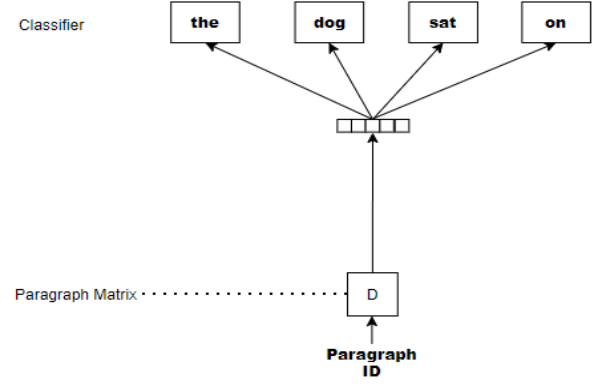


Fig. 6. Distributed Bag-of-Words Architecture

The Paragraph Vector model uses a single hidden layer neural network, similar to Word2Vec. Fig. 5. is the architecture for PV-DM. Numerical representations for word vectors and paragraph-id vectors are computed using Distributed Memory embedding. It's similar to W2V's Continuous Bag of Words (CBOW) model, in which the task is to predict the probability of a target word based on the context words around it. In PV-DM, the context entities consist of {'the','dog','sat'} and a paragraph-id ('D'); target word is ('on'). PV-DBOW architecture is illustrated in Fig. 6. Distributed Bag of Words embedding computes numerical representation only for documents. It functions similarly to W2V's Skip-Gram concept, with the goal of predicting context words {'the','dog','sat', 'on'} provided the target words. In PV-DBOW, a target paragraph ('D') is used in-place of target word. Compared to PV-DBOW, PV-DM is computationally expensive as even word embeddings are considered.

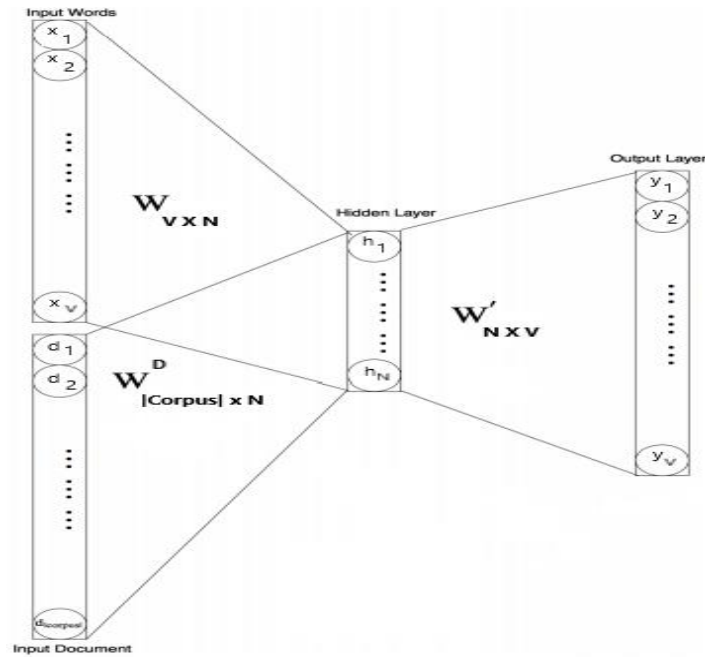


Fig. 7. Paragraph Vector Architecture

Assume a corpus of document ids D , where $D = \{d_1, d_2, \dots, d_i, \dots, d_{|Corpus|}\}$ are the input layer nodes of a document(paragraph) and $|D| = |Corpus|$. The dimension of D is $1 \times |Corpus|$. Let V be the size of vocabulary generated by aggregation of unique words from all documents present in the corpus D . Let $W_I = \{w_I^1, w_I^2, \dots, w_I^i, \dots, w_I^V\}$ and $W_O = \{w_O^1, w_O^2, \dots, w_O^i, \dots, w_O^V\}$ be the set of initial-input and output layer nodes. Let h denote the hidden layer vector, where $h \in R^N$. N denotes the size of hidden layer and, V denotes the sizes for the input and output layer respectively. W_I and W_O are one-hot encoded vectors of dimension $1 \times V$. Let's define two matrices W and W^D , where the former represents weight matrix between input-hidden layer and the latter represents weight matrix between document-hidden layer. W and W^D are of dimensions $V \times N$ and $|Corpus| \times N$ respectively. Performing the matrix dot multiplication between $\{W, W_I\}$ and $\{D, W^D\}$ produces two matrices of dimensions $1 \times N$, which are concatenated/averaged to generate a single matrix of similar size. Let $p_i \in R^N$ and $q_j \in R^N$ be the i^{th} input column vector and j^{th} output row vector of W and W_O respectively. Therefore, the final input data consists of a word vector and a document-id vector given by $W_{IF} = \{w_I^1, w_I^2, \dots, w_I^i, \dots, w_I^V, d_1, d_2, \dots, d_i, \dots, d_{|Corpus|}\}$.

The training data consists of a set of context entities – words and a document-id, used to predict a target entity at output layer. Let C and T be the set of context and target entities with $C \subset W_{IF}$ and $T \subset W_O$. For a given training instance, let $C = \{w_I^{C_1}, \dots, w_I^{C_{|C|}}, d_C\}$ represent the sequence of context entities and $w_O^{j^*}$ be the target entity. Similar to W2V, the objective in Paragraph-Vector model is to maximize the conditional probability of a target variable given its context words. While this probability is modelled as a softmax function in W2V, here we use sigmoid/logit function as in Negative sampling technique given by

$$\sigma(x) = \frac{1}{1 + e^{-x}} ; 0 < \sigma(x) < 1$$

The interesting property of sigmoid function is $\sigma(-x) = 1 - \sigma(x)$ which is useful when evaluating the gradient of objective function w.r.t input and output node weights. Negative Sampling avoids the complex softmax normalizations. Instead of considering all non-target (negative) words, only few negative words present in set N' are considered. The objective function now becomes

$$\max P(w_O^{j^*} | C) * \left(\pi_{j \in N'} (1 - P(w_O^j | C)) \right) - (1)$$

where N' is a randomly sampled subset of negative words and $P(w_o^j|C) = \frac{1}{1+e^{-q_j^T * h}}$.

The hidden vector h is constructed in a feed-forward manner as

$$h = \frac{1}{|C|} \sum_{i \in C} p_i$$

To compute update, we take a step in gradient i.e. perform gradient ascent in the direction of log of the objective function defined in (1). Therefore,

$$\begin{aligned} f(q_0, q_1, \dots, q_V, h) &= \log \left(P(w_o^{j^*}|C) * \left(\pi_{j \in N'} (1 - P(w_o^j|C)) \right) \right) \\ &= \log \left(\sigma(q_{j^*}^T * h) \right) + \sum_{j \in N'} \log \left(\sigma(-q_j^T * h) \right). \end{aligned}$$

Output node vector's gradient is

$$\begin{aligned} \frac{df}{dq_i} &= \left(I(i = j^*) - \sigma(q_j^T * h) \right) * h \\ &= \left(I(i = j^*) - P(w_o^i|C) \right) * h. \\ &= e_i * h; \forall i \in N' \cup j^* \\ &= 0; \text{otherwise} \end{aligned}$$

$e_i = \left(I(i = j^*) - P(w_o^i|C) \right)$ denotes error incurred at i^{th} output node.

Input node vector's gradient is

$$\begin{aligned} \frac{\partial f}{\partial p_i} &= \frac{\partial f}{\partial h} * \frac{\partial h}{\partial p_i} - (2) \\ \frac{\partial f}{\partial h} &= \sum_j^{N'} \left(I(j = j^*) - \sigma(q_j^T * h) \right) * q_j \\ &= \sum_j^{N'} e_j * q_j - (3) \end{aligned}$$

Substituting equation (3) in (2), we get

$$\begin{aligned} &= \frac{1}{|C|} * \sum_j^{N'} e_j * q_j; \text{ for } w_l^i \in C \\ &= 0; \text{otherwise} \end{aligned}$$

Update for output node vector thus is:

$$\begin{aligned} q_i^{t+1} &\leftarrow q_i^t + \alpha * e_i * h; \text{ if } i \in N' \cup j^* \\ q_i^{t+1} &\leftarrow q_i^t; \text{ otherwise} \end{aligned}$$

And, update for input node vector is:

$$p_i^{t+1} = p_i^t + \alpha \frac{\sum_j^{N'} e_j * q_j}{|C|} ; \text{if } w_i^i \in C$$

$$p_i^{t+1} = p_i^t ; \text{otherwise}$$

The above mathematical details concentrate on the training of a PV-DM model. In PV-DBOW, $C = \{d_C\}$ and $W_O = \{w_O^{11}, w_O^{12}, \dots, w_O^{1i}, \dots, w_O^{1V}, \dots, w_O^{nV}\}$ where ' n ' denotes number of context words. We adopted the PV-DBOW document embedding, trained on English subtitles and called it *Subtitle Vector (Sub2Vec)* model. PV-DBOW was chosen over PV-DM because the model ignores context words in the input when predicting arbitrarily selected words from the paragraph in the output to generate vectors. Moreover, the former has been found to be much effective and faster than the latter.

3.2.2.1.3. Embedding Phase

The generated Sub2Vec models were leveraged to infer vectors of 50 and 100 dimensions for each subtitle. Thus, creating matrix of dimensions 2495 X 50 and 2495 X 100 where 2495 is the total number of subtitles obtained after pre-processing and 50, 100 are the Sub2Vec embedding dimensions.

3.2.2.1.4 Feature Scaling

Min-Max Scaling was done prior to model fitting to guarantee exact same scale for all the features in the embedded vector. Only the embedded vectors of those subtitles were retained for which the complete movie data was available in the movie data file. The rest were dropped off. Finally leaving us with matrices of dimensions 2364 X 50 and 2364 X 100.

3.2.2.2. Movie data file

3.2.2.2.1. Pre-processing Phase

Over 1300 movies data comprising of categorical and numerical variables were pre-processed after the removal of movies containing null records.

3.2.2.2.2. Categorical Variables

The columns *language* and *cbfc rating* were label encoded. The languages 'Hindi', 'Tamil', 'Telugu', 'Malayalam' and 'Kannada' were encoded as 0, 1, 2, 3 and 4 respectively. Similarly, film certifications 'U', 'UA' and 'A' were encoded as 0, 1 and 2 respectively.

The *release day* and *release month* were extracted from the *release date* column to be added as two separate columns after label encoding, thus dropping the original *release date* column. A list of 20 unique genres comprising of ‘Action’, ‘Sci-Fi’, ‘Adventure’, ‘Animation’, ‘Short’, ‘Biography’, ‘Drama’, ‘Thriller’, ‘Comedy’, ‘Crime’, ‘Musical’, ‘Family’, ‘Fantasy’, ‘Romance’, ‘History’, ‘Horror’, ‘Mystery’, ‘Documentary’, ‘Sport’, and ‘War’ were generated and appended as columns in the dataset by removing the original *genre* column. These represented binary valued features where, the genre columns corresponding to a movie’s genre values were set to 1, leaving the rest to 0.

Based on the verdict of movies, two separate lists - successful and unsuccessful verdict list were generated. Verdicts such as ‘All Time Blockbuster’, ‘Blockbuster’, ‘Hit’, ‘Super Hit’, ‘Semi Hit’, ‘Above Average’ and ‘Average’ were placed in the successful list whereas ‘Flop’, ‘Below Average’ and ‘Disaster’ in the unsuccessful list. If a movie’s verdict lies in the successful list, then its *verdict* was set to 1, otherwise to 0.

3.2.2.2.3. Numerical Variables

The columns *runtime*, *budget* and *box office collection* were ensured to be of integer type. A new column *roi* (*return on investment*) was generated by dividing the *box office collection* with the *budget* thereby making it the second last column in the dataset, *verdict* being the last. This was done to ease the process of splitting the dataset.

For the columns *actor*, *music director*, *movie director*, *writer* and *producer popularity*, a popularity dictionary was created containing the popularity score of individuals from all five movie industries. *Actor popularity* was calculated by summation of popularity score of actors involved in the movie. Same method was adopted to get other popularity values.

A total of 1355 movies were obtained with columns *language*, *genre*, *release day*, *release month*, *cbfc rating*, *runtime*, *budget*, *box office collection*, *actor popularity*, *music director popularity*, *movie director popularity*, *writer popularity*, *producer popularity*, *roi* and *verdict*.

3.2.2.2.4. Feature Scaling

To ensure exact same scale, Min-Max scaling was done on the numerical variables alone. Due to the unavailability of box office collections during pre-production phase, the column *box office collection* was dropped from the dataset. As a result, the movie data had dimensions of 1355 X 33.

3.2.2.3 Merged data

The subtitle vectors of dimension 2364×50 and 2364×100 were intersected with movie data vectors of dimension 1355×33 to generate merged datasets of dimension 1200×83 and 1200×133 where 1200 is the final set of movies and 83, 133 are the total number of features generated by combining subtitle and movie data vectors corresponding to 50 dimensions and 100 dimensions respectively.

roi and *verdict* were the two target variables, so after excluding them from the merged dataset final datasets of dimension 1200×81 and 1200×131 were formed. Both *roi* and *verdict* served as 1200-dimensional column vectors.

Fig. 8. represents the class distribution of target variable *verdict*, for the train data. We observe that this distribution is found to be balanced.

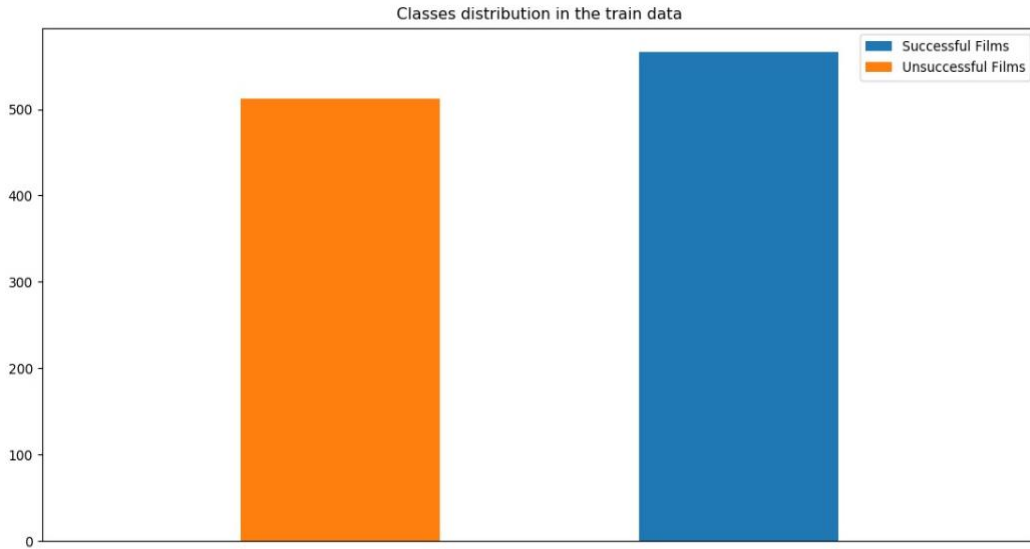


Fig. 8. Class distribution of train data

3.2.2.4 Application of models

The machine learning algorithms used in this research consisted of regression and classification models. Regression models include Linear Regression, XGB Regressor (XGBR), Support Vector Regressor (SVR), Gradient Boosting Regressor (GBR), and AdaBoost Regressor (ABR). Classification models include Logistic Regression, Support Vector Classifier (SVC), XGB Classifier (XGBC), Gradient Boosting Classifier (GBC), and AdaBoost Classifier (ABC). The regressors were trained using the final merged datasets of dimensions 1200×81 and 1200×131 as the predictors, with *roi* as the target. The classifiers were trained with *roi*, serving as the predictor variable and *verdict* as the final target variable.

The proposed model thus, followed a two-stage prediction pipeline. In the first stage, the merged datasets corresponding to Sub2Vec 50- and 100- dimensions mentioned above, were used to estimate *roi* using the trained regressors.

For the final stage, classifiers trained on the *roi* values, use the expected *roi* from the 1st stage to estimate the target variable *verdict*. Since the target variable was of continuous type in the 1st stage, it was a Regression problem. The estimation of final verdict in the 2nd stage is a Binary Classification problem with two categories – successful and unsuccessful.

a. Linear Regression

Linear regression forms/models a relationship between a set of explanatory (dependent) variables and a continuous (independent) variable. Linear Regression can be classified into Simple Linear Regression and Multiple Linear Regression. In simple linear regression, a single dependent variable is used in the estimation of continuous variable whereas in multiple linear regression, more than one dependent variable is used in the estimation of a continuous target. A linear function is modelled from the provided dataset by estimating unknown parameters of the function. Linear models are a model that corresponds to this description.

Given the values of the dependent attributes, the conditional mean of the target/independent variable is mostly considered to be an affinity function while the conditional median of the same is considered less commonly as an affinity function of the predictor instances. A conditional distribution of probability of target given the dependent attributes concerns a simple linear regression and a joint distribution of probability of target given dependent attributes is concerned with multivariate/multiple linear regression.

Despite the fact that various cutting-edge regression algorithms have been created over the years, linear regression was the first type of regression analysis to be thoroughly researched and implemented in practical systems. The fundamental reason for this was that the parameters of a linear hypothesis were easier to estimate than those of a non-linear hypothesis. Linear regression has a wide range of applications. Most of these categories fall either of two:

Through using observed data, i.e., labeling in each case, linear regression can construct a predictive model where the goal is to prevent, predict or reduce errors. This trained model can forecast the outcome based on these values for test data, i.e., if unknown values of attributes/features are supplied.

By analyzing a linear regression model, we can quantify the extent of dependency of an attribute to the response or target variable. These help us determine if these attributes form a non-linear relationship with the target or a linear one. As linear models aren't well suited for complex non-linear associations, we can rule out these corresponding variables to potentially improve the model's accuracy or predictive power.

There are various methods for accurately fitting a linear regression model. The "least squares method" is the most commonly used method. The total of the observed and actual errors is minimized using this strategy. The projection error of actual data points on the hypothesis line is used to calculate this. Ridge (L2) and Lasso (L1) regression are penalized variants of the linear regression model. These include a regularization parameter to decrease the extent to which dependent features on the target variable are overfitted. Ridge is the most commonly used of these.

A set of attribute instances (x) in a linear equation predicts the target value (y). Unlike ' x ,' which can be either numerical or categorical, ' y ' is a numerical quantity. The value of each input attribute is allocated a unique parameter or coefficient, represented by θ_1 . We have an extra coefficient known as an intercept or bias, which is denoted by θ_0 . The equation shown below is an algebraic representation of a straight-line equation for a single attribute.

$$y = \theta_0 + \theta_1 * x$$

When there are multiple dependent variables, the line equation above is transformed into a higher-dimensional plane. Then you have a unique coefficient for each dependent variable, such as 1, 2, 3, and so on. There has been a lot of discussion on how the coefficient values affect the model.

A coefficient value of 0 for an attribute means it has no effect on the linear model; this is equal to $(0 * x = 0)$. This is especially important when using regularization methods like Ridge and Lasso. Because the regularization parameter penalizes the magnitude of a coefficients by pushing them towards 0, an absence of relation of that variable with the target can cause issues.

b. Logistic Regression

One of the most popular Machine Learning algorithms in the Supervised Learning approach is logistic regression. Here a collection of independent variables are used to predict a dependent variable which is categorical in nature.

Logistic regression is used to predict the output of a variable of type categorical and dependent. Thus, the outcome must be either discrete or categorical. It could have been Yes/No, 0/1, true/false etc but instead of giving precise values such as 1 and 0, it delivers the values which are probabilistic and between the values 1 and 0.

Logistic Regression is extremely similar to Linear Regression in terms of how it is used. Linear Regression is used to solve regression problems, while Logistic Regression is used to solve classification problems.

The two maximum values of 0/1 are predicted by the “S” shaped logistic function , rather than fitting the regression line.

The probability of things like whether the cells are cancerous or not, based on its weight whether or not a mouse is obese etc are reflected in the logistic function's curve.

Logistic regression is a popular ML approach as it has the ability to generate probabilities & classify new data points utilizing both continuous & discrete datasets. It also has the ability to categorize observations which are based on data of different variants and also to identify all the prominent factors.

A mathematical formula for translating expected values into probabilities is the sigmoid function. It tries to convert any real number between 0 & 1 to another number. The value of the logistic regression must be between 1 & 0, further it must not exceed this limit, else a "S" curve will result. Thus, the Sigmoid function, popularly known by the name logistic function, a curve which is S-form.

The approach value, which indicates the likelihood of either 0 or 1, is used in logistic regression. For example, numbers above the threshold value will most likely be 1, whereas those below the approach value will most likely be 0. A categorical dependent variable is required. There should be no multi-collinearity in the dependent variable. The final equation for Logistic Regression is the equation below.

$$\ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k$$

LR can be divided into 3 types based on the categories:

- Binomial: Here, the dependent variables can only be one of 2 types, such as 0 or 1, Pass or Fail, and so on.
- Multinomial: Here, the dependent variable can be one of 3 or more unordered kinds, as "cat," "dogs," or "sheep."
- Ordinal: 3 or more possible depending variables, as "low," "medium", "high," may exist in ordinal logistic regression.

c. Support Vector Machines

SVM are models which are supervised in nature & used in machine learning to analyse data for classification and regression.

Vladimir Vapnik and colleagues created SVMs at AT&T Bell Laboratories. These were based on learning frameworks which were statistical in nature / VC theory. These theories were proposed by Chervonenkis & Vapnik. SVM training builds a model that categorises fresh instances into one of two groups. Thus it can also be termed as a linear classifier which is binary in nature and also is of the type non probabilistic. Lets say we have a set of examples for the purpose of training, all of which are marked so as to belong to any of the 2 given categories(But there are approaches for using SVM in a classification environment that is probabilistic in nature, such as Platt scaling). All the examples which are used for the purpose of training are mapped by SVM to a point in space. This is done in order to widen the distance between the 2 categories as much as possible. The same space is used for mapping the newly available examples. This is done so that the classifications fall only into those categories based on the side of the gap (However, there are methods for using SVM in a probabilistic classification situation, such as Platt scaling).

Kernel trick is utilized by SVMs to perform non-linear classification as well as linear classification, which involves completely translating into feature space which have a higher dimension from the given input.

The supervised learning approach is impossible when the data is unlabeled, hence unsupervised learning strategy is necessary, where the data is clustered naturally into groups and new data is mapped to groups. Also the support vector clustering technique was developed by Vladimir Vapnik & Hava Siegelmann to categorize unlabeled data using support vector statistics obtained in the support vector machines method. For industrial applications, it is one of the most widely used clustering methods. SVM, divides data points into two groups by using a hyperplane/linear decision surface to build a border between them. In order to produce the widest possible split between the two categories of data, the SVM seeks out the Maximum Margin Hyperplane, popularly known as MMH.

They are particularly ably adapted to binary classification issues, & they have been effectively applied to text classification. SVM can also be utilized as regression approach while retaining all of algorithm's fundamental characteristics (maximal margin).

For a few minor exceptions, the Support Vector Regression utilizes the similar principles as that of the SVM for classification. For starters, because output is a real number, predicting the information that we have at hand, has unlimited prospects, becomes extremely challenging. In regression, an approximation to the SVM is set with a tolerance margin (epsilon), which the problems would have already been requested.

However, in addition to this, a more complex motive why the algorithmic program must be taken more intricate. However, in order to minimize error, to individualize the hyperplane maximizing the margin, the principal notion is the same, taking into account that some of the mistake is accepted.

Our aim is to take into consideration the points within the decision-making line when we proceed with SVR. The hyperplane with the most points is the most suitable fit line for us. Before moving on, there are a few crucial SVM parameters to be aware of:

Kernel: It helps to identify a subspace or hyperplane in the larger dimension in the absence of raising the costs of computing. In general, the cost of calculations increases as the data dimension grows. This dimension increases when a separating hyperplane cannot be found in a particular dimension and moves to a dimension of higher value.

Hyperplane: Basically, a line that is between 2 SVM classes of data. This line is utilized for the purpose of predicting a continuous output in Support Vector Regression.

Decision limit: On the one hand, the decision limit can be considered a line (for simplicity) with positive instances and on the other hand negative examples. The decision limit may be considered a line.

d. Adaptive Boosting Gradient

Freund and Schapire devised AdaBoost, a statistical classification meta-algorithm for which they were awarded the Gödel Prize in 2003. Combining additional learning algorithms known as 'weak learners' improves the algorithm's performance. The final output of the AdaBoost algorithm is a weighted sum obtained by adding the outputs of all the weak learners. The term 'adaptive' in AdaBoost refers to how this method adjusts subsequent weak learners by considering misclassified cases from previous classifiers.

When compared to other learning algorithms, it has a lower risk of overfitting. Although the predictive power of "weak learners" is limited, if their combined performance is superior to random guessing, the final system will be a capable learner.

Tweaking parameter configurations assures that any learning algorithm achieves optimal performance on an arbitrary dataset, and for some issues, one algorithm performs better than the other for specific parameter values. AdaBoost is a cutting-edge classifier in which the weak learners are decision trees. The misclassified examples of previous decision trees are handed on to subsequent decision trees, making classification more difficult for subsequent decision trees at each stage.

AdaBoost is an ensemble method for training and deploying trees in a sequential order. AdaBoost increases the classification of samples that were misclassified by the preceding weak classifier by connecting a sequence of weak classifying devices. This combines a strong classifier to build a powerful classifier with weak classifiers.

Decision trees employed in boosting methods are nicknamed "stump," as every decision tree is not excessive, but biased models. A single tree has been taught to focus solely on the previous tree's flaws.

The weight of a sample misclassified by the prior tree will be increased, allowing the next tree to concentrate on accurately identifying the previously misclassified sample. When more weak classifiers are added to the model in series, the classification accuracy improves; nevertheless, this can result in significant overfitting and a loss of generalization capacity.

AdaBoost is suitable for uneven datasets but in the presence of noise underperforms. Slower for train is AdaBoost. Yoav Freund and Robert Schapire originally formulated Ada-Boost. In AdaBoost the different hyperparameters are:

- i. base estimator: This parameter specifies the type of base learners that can be utilized, as well as the type of weak learner that should be employed. It can be SVC, Logistic regression, or Decision Tree. It cannot be KNN, because in this paradigm the weight cannot be allocated. Default is DecisionTreeClassifier (max depth=1) as the basis estimator.
- ii. n estimators: The number of weak learners or base estimators we'd like to utilize in our dataset. The n estimator is set to 50 by default.
- iii. learning_rate: This option is used to reduce each classifier's contribution. It is given a value of 1 by default. rate of learning: This option is used to reduce each classifier's contribution. It is given a value of 1 by default.
- iv. algorithm: SAMME or SAMME.R are two options. The SAMME and SAMME.R algorithms are compared in terms of performance. SAMME.R updates the additive model using probability estimates, whereas SAMME just utilizes classifications.

e. Gradient Boosting

It is an ML methodology in which a prediction model is created as a set of prediction models which are very weak, often decision trees are utilized for regressions and classification problems. The resulting technique is called gradient boosted trees when a decision tree is the weak learner, and in most cases, it outperforms a random forest. It builds the model in the similar step-by-step fashion as all the other boosting methods. However, by permitting optimization of any loss function which is differentiable, it broadens the scope.

It has three components:

- i. A loss function that needs to be optimized
- ii. A weak predictor learner.

- iii. An additive model for the purpose of reducing the loss function by adding weak learners.

- i. Loss Function

The type of loss function that is used depends on the problem that is being solved. Must be differentiable, however there are numerous standard loss functions available, as well as the ability to define your own.

Let's take an example: a squared error in regression may be used, while in classification, a logarithmic loss may be used.

The gradient boosting framework has the advantage of not requiring the development of a new boosting algorithm for any loss function that may possibly be used; but, any loss function that is differentiable can be used because it is a generic framework.

- ii. Weak Learner

Decision trees are mainly utilized to boost gradients as the learner which is weak. Regression trees are utilized specifically because they deliver true split values & can be combined altogether, which allows for future model where their outputs can be added & “correct” the left overs in the at hand prediction task.

All these trees built gullibly, with the optimal split points chosen on the basis of pure values such as Gini or for the purpose of minimizing loss. At first, like as we see with AdaBoost, extremely small decision-making bodies, called the decision-making stump, were used. Larger trees with 4 to 8 tiers can usually be employed. Weak learners are usually restricted in certain ways, for example the maximum number of nodes, layers, splits or even leaf nodes. Thus, ensuring that all the learners are still weak & can be built greedily.

- iii. Additive Model

In the model all existing trees left unmodified, and one at a time the novice trees are inserted. When performing the summation of trees, a gradient descent approach is employed for the purpose of minimizing loss.

Gradient descent is traditionally utilized for the reduction of a number of parameters including regression coefficients or neural network weights. The weights are modified to reduce this inaccuracy after the calculation of error or loss.

We have weak learner substructures or decision trees in particular, rather than parameters. To complete the gradient descent methodology, we should take summation of a tree to the model so that it tries to reduce the loss after computing the loss. After computing the loss, we should add a tree to the model that reduces the loss to complete the gradient descent technique (i.e., follow the gradient). For this purpose, the tree is parameterized, modifying the tree's parameters & moving in the correct direction.

Here the method is known as functional gradient descent. The new tree's output is then blended with the previous tree sequence's output to correct or improve the model's final output. When loss reaches a tolerable level or training no longer improves even on a validation dataset that is external, a specified number of trees are added or training terminates.

It is a gullible method that can easily overfit a dataset used for training.

Regularization algorithms that penalise different parts of the algorithm and enhance overall performance by reducing overfitting can help.

In this part, 4 improvements to the basic gradient increase will be considered:

1. Shrinkage
2. Random sampling
3. Tree constraints
4. Penalized Learning

1. Weighted Updates

Each tree's predictions are gradually joined together.

An algorithm can weigh the contribution of each and every tree to the summation in order to reduce learning. Shrinkage or learning rate are known as this weighting. The value of "learning rate parameter v " simply scaled for each update.

The impact is learning has been slowed, and that more number of trees are added to model, and that they will take time to train and make the change between number of trees & the rate of learning possible. The reduction in ν enhances optimal M [number of trees] value.

Small values are commonly found in the range from 0.1 to 0.3 and values below 0.1. Shrinkage minimizes the influence of each and every leaves & tree room for the upcoming trees to enhance our model, similar to the rate of learning in stochastic optimisation.

2. Stochastic Gradient Boosting

The greatest breakthrough in random forest & bagging ensembles enabled a tree from sub-samples of training data to be greedily generated.

This same benefit may be used to lower the sequence correlation of gradient boost models between the trees.

Stochastic gradient boosting is named this kind of boosting. A subsample of the training data is randomly taken from the whole training data set (without replacement). Instead of the whole sample, the randomly selected subsample is utilized to fit the basic student.

There are a number different types that can be used:

- Sub-sample rows before the building of each and every tree.
- Before every tree a column is created
- Columns are sub-samples before each split is considered.

Generally speaking, aggressive sub-samples like the selection of only 50% of the data have shown useful.

3. Tree Constraints

It's vital that the struggling or the weak learners keep their skills. Constricting trees can be done in a lot of different ways.

The more restricted the tree form, the more will be the number of trees we require in model & in reverse, the fewer trees you need if you use less restricted individual trees.

Some limits that are imposed on development of DT are listed below:

- A number of trees can be quite slow to overfit if additional trees are added to the model. The recommendation is that trees continue to be added until further improvements are noted.
- The deeper the trees, the deeper the trees, the shorter the trees. Better results with four-eight levels are usually noticed.
- Number of leaves or nodes, such as depth, may restrict tree size, but is not limited to symmetrical construction when other restrictions are employed.
- A minimum restriction is placed so that before a split can be considered, there must be a sufficient amount of training data at training node.

4. Penalized Gradient Boosting

Aside from their structure, the parameterized tree can be subjected to additional constraints.

As weak learners, traditional decision trees such as CART are not employed; but, a reworked version known as regression tree is utilized, which contain numerical values at the terminal nodes. In some literature, the term weight is used to refer to all the values in leaves of tree.

Weight of leaf of tree can therefore be regularized. This is done so using popular regularization functions, for example:

- L_1 weight regularisation.
- L_2 weight regularisation.

Additional regularization period aids to reduce weights learned to ensure that they are not exceeded. Intuitively, a model using basic and predictive functions is used for the regularized objective.

f. Extreme Gradient Boosting Machines

XGBoost is a C++-based gradient-boosted library. Because of its performance and speed of execution, it is a go-to algorithm for many Kaggle tournaments. It has dominated a number of cutting-edge machine learning algorithms.

Each independent variable in XGBoost is allocated a weight factor. These factors are then fed into a decision tree as input.

The sequential organization of decision trees is used in this strategy. When a variable's classification is wrong, the weight associated with it is increased. This improperly classified variable, along with other variables, are subsequently forwarded to the next level, where they serve as input to a second decision tree. These decision tree predictors are integrated at a later stage to produce a powerful model. In addition, XGBoost can be used to solve regression, ranking, classification, and user-defined predictions.

There are three types of gradient boosting that are supported:

- Gradient Boosting
- Stochastic Gradient Boosting
- Regularized Gradient Boosting

Extreme Gradient Boosting (XGBoost) is an open-source package which implements the gradient boost method in an efficient and efficient way.

While the approach existed before XGBoost in other open-source implementations, the XGBoost release appears to release the power of this technology and the machine learning community notice a gradient boost in general.

XGBoost became the go-to strategy and often a critical element for successful solutions to the classification and regression issues in automated learning competitions shortly after its inception and initial releases.

Gradient improvement refers to a type of machine-learning technique that may be utilized for predictive modelling problems with classification or regression. Ensembles are built from models of decision tree. Trees are added to the ensemble one by one and fit to rectify previous model prediction mistakes. This is a kind of machine learning model known as boosting. Models are suitable employing every arbitrary differentiable loss function and methodology for gradient downward optimization. This gives the technique its name, "gradient boosting," because the loss rate is reduced as well as the fit model, similar to a neural network.

Extreme gradient boosting or short XGBoost means that the gradient boosting technique is efficient open-source implementation. XGBoost is therefore an algorithm, an open-source project and a library of Python.

It has been developed first by Tianqi Chen. In their article entitled "XGBoost: A Scalable Tree Boosting System" for 2016, Chen and Carlos Guestrin discussed it. It is designed to be computationally efficient (e.g. quick) and highly efficient and possibly more efficient than others.

The name XGBoost alludes in fact to the engineering objective of increasing the calculation resources limit for increased tree algorithms. That is why a lot of people utilize XGBoost. Execution speed and model performance are the two key reasons to employ XGBoost. XGBoost is usually quick compared to other gradient boosting solutions. With respect to tabular or structured, XGBoost performance exceeds other learning algorithms. This can be observed from the Data science competition in Kaggle whose winners used this algorithm

Some of the hyperparameters for the Gradient Boosting set and the impact they should have on model performance will be examined in further detail.

i. Explore Number of Trees

The number of decision trees employed in the ensemble is an important hyperparameter for the XGBoost ensemble technique. In order to correct and improve predictions produced by previous trees, the decision trees are gradually added to the model. More trees, therefore, are often preferable. You can set the number of trees to 100 with the option "n estimators."

ii. Explore Tree Depth

Another essential hyperparameter for gradiently boosting varies the depth of each tree contributed to the ensemble. The depth of the trees checks whether each tree is specialized in the training dataset: how generic or overfit.

Trees that are not too shallow and generic (like AdaBoost) or too deep and specialized (like AdaBoost) are desirable (like bootstrap aggregation). Gradient improvement usually works effectively with trees of modest depth, which strikes a balance between skill and generality. Tree depth is configurable by the option "max depth" and by default 6.

iii. Explore Learning Rate

The learning rate determines how much each model contributes to the ensemble forecast. Smaller rates may need the inclusion of more decision trees in the ensemble. The "eta" option controls the learning rate, which is set to 0.3 by default.

iv. Explore Number of Samples

It's possible to change the quantity of samples utilized to fit each tree. This means that each tree is trained on a subset of the training dataset chosen at random. Although using fewer samples creates greater variation for each tree, it can enhance the model's overall performance.

The “subsample” option specifies the amount of samples used to fit each tree and can be set to a percent of the training dataset size. It's set to 1.0 by default, which means it'll use the whole training dataset.

3.2.2.5 Evaluation of results

A brute force approach was adopted to determine the optimal regressor and classifier by evaluating the F1 and Cohen's Kappa score. For the optimal regressor and classifier, the Accuracy, Precision, Recall, Matthew's Correlation Coefficient (MCC) and 10-fold Cross Validation (CV) were also calculated.

a. Mean Squared Error

The MSE of a regression line reflects how close it is to a set of points. By squaring the distances between the points and the regression line, it achieves this. To eliminate any negative signals, squaring is required. Significant differences are also given more weight. It's called the MSE since you're calculating the average of a sequence of errors. The better the forecast, the lower the MSE.

$$\text{MSE} = (1/n) * \sum (\text{actual} - \text{forecast})^2$$

Where:

- n = number of samples,
- Σ = [notation](#) for summing,
- Actual = y-value (original or observed),
- Forecast = predicted regression's y-value.

Follow these procedures to calculate the MSE given a set of X and Y values:

1. locate the regression line.
2. To determine the new Y values (Y'), fill in the linear regression equation with the X values.
3. In order to determine the fault, subtract the newly obtained Y value from the original.
4. Make a square with the errors.
5. Add up the errors (summation notation is used in the formula).
6. Calculate the average.

b. Accuracy

The accuracy score function determines the accuracy of those predictions which are correct in the form of fraction, which is the default one or as a count i.e when normalize is equal to False.

In multilabel classification, the subset accuracy is returned by the function. The accuracy of subset is 1.0 if a sample's entire set of projected labels closely matches the set of labels which are true; otherwise, it is 0.0.

The percent of successful predictions over nsamples is defined as follows: The i-th sample's anticipated value is \hat{y}_i , and the matching true value is y_i .

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

Fig. 9. Formula of Accuracy

c. Precision, Recall and F1-score

Precision, on the surface, refers to the classifier's capacity to avoid labeling a sample which is negative as a positive one, while recall is the capacity of a classifier to locate all the samples which are positive.

When both precision and recall are subjected to their weighted harmonic mean, we obtain the F-measure (also known as the F_β and F1 measurements). A F_β measure has a prime value of 1 followed by an unfavourable value of 0. F_β & F1 are equal with $\beta=1$, and both precision and recall are important.

The words "positive" and "negative," in a binary classification task relate to a classifying prediction and the keywords "true" and "false" mean whether the forecast is the external (also referred to as a "observation") judgment.

Tp (true positive) = Correct result ; Fp (false positive) = Unexpected result

Fn (false negative) = Missing result; Tn (true negative) = Correct absence of result

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Fig. 10. Formulas for Precision, Recall and F1 score

d. Matthews Correlation Coefficient

MCC is a test to determine the quality of binary classification tasks. It was proposed by the famous biochemist namely Brian W. Matthews in the year 1975. This coefficient is heavily used in machine learning.

From its inception by Udny Yule in 1912, the MCC is defined exactly as the phi coefficient introduced by the reverent Karl Pearson. The MCC is popularly referred to as the Yule phi-coefficient. Although the term MCC is used many decades earlier than that of Matthews, it is commonly employed in the field of machine learnings and bioinformatics.

The factor takes genuine and false positive & negative into consideration and generally is considered a measure which is balanced that may be employed even in extremely dissimilar classes. The MCC is essentially a coefficient of correlation between the binary classifications observed and forecast and returns the value between -1 and +1.

A +1 factor is a flawless prediction, i.e 0 is not better than an arbitrary guess & -01 shows the entire discrepancies between forecast and the observation. If the value of MCC is not -1, 0%, +1, however, the likelihood of a predictor being randomly guessed does not depend on that dataset is reliable indication. MCC is in close contact with the 2-2 contingency table chi-square statistics.

While true and false positive and negative by a single number are not properly stated in the confusion matrix, the Matthews coefficient of correlation is widely considered to be one of the best of these metrics.

Additional measurements are not relevant if the two classes are of considerably different size, including the percentage of predictions which are accurate (popularly known by the name accuracy);

In particular, allocating each object to the bigger collection results in a soaring percentage of predictions which are right, but is not usually useful. Also, the MCC from the confusion matrix can be derived directly from the formula:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Fig. 11. Formula for Matthews Correlation Coefficient

e. Cohen's Kappa

Rather than a classifier vs. a basic fact, this measure is used to assess the etiquette of various human annotateurs.

The score ranges from -1 to 1. Good agreement is defined as a score of .8 or higher; no agreement is defined as a score of zero or lower (practically random labels).

For binary or multi class issues, but not for multi-label problems or for only two annotateurs, the Kappa scores can be determined (except by calculating manually a per-label score).

The agreement of two raters who categorize N things into categories which are mutually exclusive is measured by Cohen's kappa.

$$\kappa \equiv \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e}$$

Fig. 12. Formula for Cohen's Kappa statistic

f. Cross Validation

A resampling method for assessing machine learning models on a limited sample of data is cross-validation.

There is only one parameter in the method, k, which defines how many groups a given data sample should be divided into. Thus, k-fold cross-validation is a common name for the procedure.

When a precise value for k is supplied, it can be used to replace k in the reference model, for example, for 10-fold cross-validation the value of k=10.

Cross-validation is a machine learning approach that is used to measure the competence of a machine learning model on unknown data. That is, a small sample size is used to test how the model would perform in general when used to make predictions on data that was not used during training phase.

It's a common strategy since it's simple to understand and results in a less skewed or optimistic assessment of model competency than other methods, such as the basic train/test split.

The commonly followed procedure is as follows:

1. At first, the dataset is randomly shuffled
2. Make a total of k groupings out of the data.
3. For each one-of-a-kind group:
 1. The group is used as a reserve or a data collection for the purpose of testing.
 2. Use the data set of the groups which are remaining for training.
 3. On the training set, build a model, then test it on the set for testing.
 4. Evaluation score should be kept, but model should be discarded.
4. Summarize the ability of model using the sample model evaluation scores.

Especially, each and every observation in the data sample is allocated to a specific group and remains in that group throughout the procedure. This indicates that each sample can only be used once while training the model $k-1$ times.

It's also crucial that the data is prepared in the loop on the cross validation-related training dataset rather than the larger data set before the model is deployed, and not on the dataset as a whole. This is applicable to all hyperparameter settings as well.

Failure to complete these procedures within the loop could lead to data leakage and an overestimation of model competence.

The mean of the model skill scores is frequently used to describe the outcomes of a k -fold cross-validation run.

It's also a good idea to add a measure of the skill scores' variance, i.e. the standard deviation. The k-fold cross validation technique can be done in a variety of ways. The following are some of the most regularly used variations:

- Train/Test Split: If taken to its logical conclusion, k might be set to 2 instead of 1, yielding a single train/test split for model evaluation.
- LOOCV: Taken to its logical conclusion, the value of k could be set to the total number of observations in the dataset, allowing each observation a chance to be the dataset's last survivor. This is referred to as LOOCV (leave-one-out cross-validation).
- Stratified: Data can be divided into folds by criteria that ensure that each and every fold contains the exact same fraction or percentage of observations of a category value i.e. the result value of the class. This is known as cross-validation stratified.

3.3 Proposed System Model (ER Diagram/UML Diagram/Mathematical Modeling)

UML USE CASE DIAGRAM

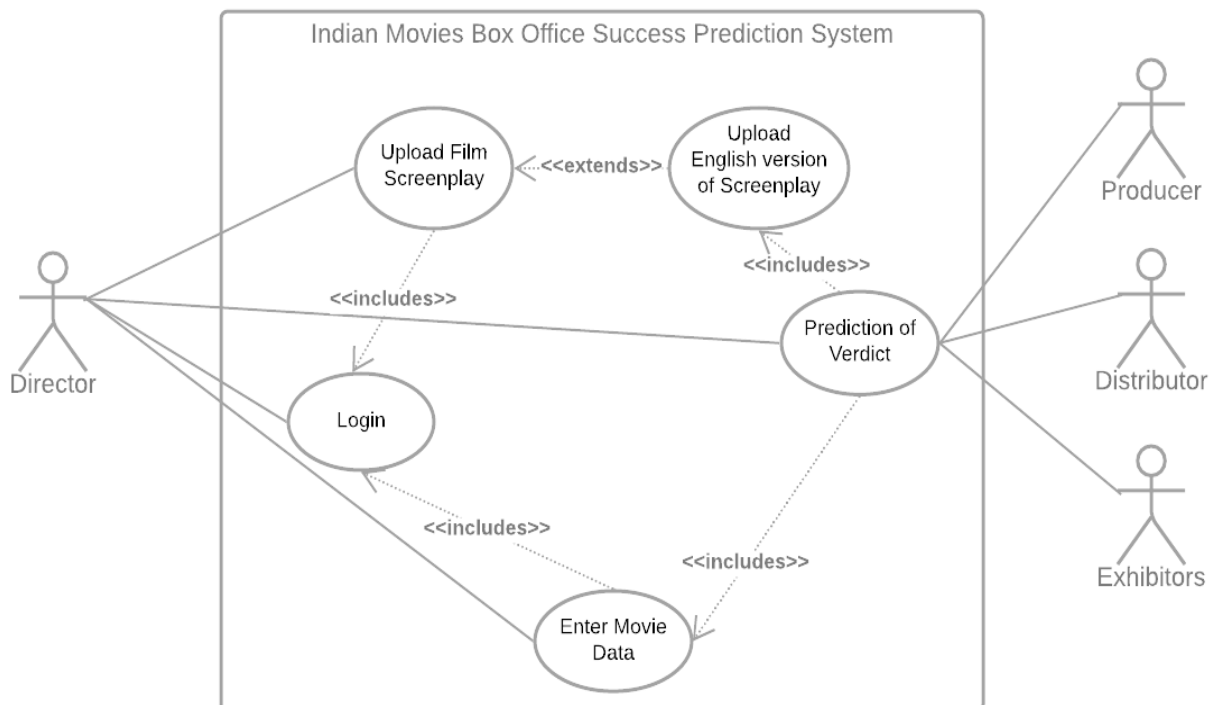


Fig. 13. UML Diagram for the proposed system

4. Proposed System Analysis and Design

4.1 Introduction

The proposed system model is shown in form of a use-case diagram. There are 3 actors, who constantly interact with the system – director, producer, distributors and exhibitors. The director logs into the system and provides required input i.e., movie data and screenplay. Using these use-cases as inputs, the system makes a prediction if movie will be successful or unsuccessful. The output use-case is then relayed to all the actors.

4.2 Requirement Analysis

4.2.1 Functional Requirements

4.2.1.1 Product Perspective

The Indian movies box office success prediction system estimates if an Indian film will be profitable or not in the pre-production phase, based on a set of pre-production factors.

4.2.1.2 Product features

The major features of the system include – early prediction of a movie's success.

4.2.1.3 User characteristics

The movie director and producer are the main users of the system. Other users include film distributors and exhibitors. Users should be able to know early on if their movie would turn out profitable, based on screenplay and movie data provided.

4.2.1.4 Assumption & Dependencies

Client/User has production cost/budget, writer, producer, actors, music directors and film script available for the movie.

4.2.1.5 User Requirements

The system must accurately predict if a movie will be successful or unsuccessful at the box office. Furthermore, the prediction must be explainable i.e., the system should say why a movie will succeed or fail at the box office.

4.2.2 Non-Functional Requirements

4.2.2.1 Product Requirements

4.2.2.1.1 Portability

- Supported Operating systems
 - Windows 7 or later
 - Mac OS X 10.1 or higher, 32/64-bit
 - Linux: RHEL 6/7, 64-bit (almost all libraries also work in Ubuntu)
- Network Specifics
 - Inbound HTTP: TCP 8080, 8443
 - Optional Inbound SSH: TCP 22 (SSH)
 - Optional Outbound HTTPS: TCP 443
 - Optional Outbound SMTP: TCP 25 (if not using AD/LDAP) email notifications
 - Optional Outbound LDAP(s): TCP 389/636 for authentication integration
- Browsers
 - Internet Explorer
 - Chrome
 - Firefox
- Hardware requirements (minimum)
 - x86 64-bit CPU (Intel / AMD architecture)
 - 4 GB RAM
 - 5 GB free disk space

4.2.2.2 Operational Requirements (Explain the applicability for your work w.r.t the following operational requirement(s))

- Economic – The proposed system assists directors and movie stakeholders i.e., producer, distributors and exhibitors to assess the economic potential of their film at quite an early stage.
- Environmental – None
- Social – In social aspect, system specifically assists the director and movie stakeholders.
- Political – None
- Ethical – None
- Health and Safety – None

- Sustainability – None
- Legality – None
- Inspectability – None

4.2.3 System Requirements

4.2.3.1 H/W Requirements (details about Application Specific Hardware)

- x86 64-bit CPU (Intel / AMD architecture)
- 4 GB RAM
- 5 GB free disk space

4.2.3.2 S/W Requirements (details about Application Specific Software)

- Python ≥ 2.6 (included all packages)

5. Results and Discussion

The *gensim library's Doc2Vec* function was used for training the Sub2Vec model. The hyperparameters tuned comprised of *vector_sizes* = [50,100] for 50- and 100-dimensional vector embeddings respectively and *epochs* = [10,20,50,100]. The other hyperparameter values were learning rate *alpha* = 0.025, *min_alpha* = 0.0001, threshold word frequency *min_count* = 5 and *dm* = 0 (PV-DBOW). Training-Testing split of 90%-10% was used. Table. 2. mentions the optimal regressor-classifier pair obtained for the hyperparameters mentioned above.

Table. 2 Optimal Regressors and Classifiers for different hyperparameter values

	Epochs							
	10		20		50		100	
	50-d	100-d	50-d	100-d	50-d	100-d	50-d	100-d
Regressor	XGBR	XGBR	XGBR	XGBR	XGBR	XGBR	XGBR	XGBR
Classifier	SVC	SVC	ABC	SVC	ABC	XGBC	SVC	GBC

Table. 3. F1 and Cohens Kappa Test data scores of optimal classifiers for different hyperparameter values

	Epochs							
	10		20		50		100	
	50-d	100-d	50-d	100-d	50-d	100-d	50-d	100-d
F1 score	0.766	0.81	0.792	0.753	0.77	0.764	0.753	0.775
Cohens Kappa	0.434	0.45	0.431	0.421	0.483	0.414	0.465	0.40

XGB Regressor performed best among the optimal regressors. SVC performed better for most predictions among the optimal classifiers. AdaBoost Classifier comes next, followed by XGB Classifier and Gradient Boost Classifier. The best optimal model consists of an XGB Regressor and SVC classifier corresponding to Sub2Vec of 100 dimensions and 10 epochs. From Table. 3., the best optimal model's classifier measured an F1 score of 0.81 and Cohen's kappa score of 0.45 on test data; other recorded performance measures include Accuracy = 0.75, Precision = 0.753, Recall = 0.876, MCC = 0.461 and 10-fold CV = 0.83.

As subtitles aren't available in pre-production stage, we can use screenplay instead. Owing to the scarcity of screenplay data for Indian films, we used subtitles as an approximation. A potential extension to this work would be the use of film scripts rather than subtitles.

The existing model does not take into account the essence of a scene because subtitles do not include it. If there were more dedicated websites and data available, particularly for regional language films, the predictions may have been more accurate. Ambiguity in data severely affects the model's prediction. In many websites, there is contradicting data for Indian movies, especially for box office collection, budget, genre and verdict.

Explainability of a prediction plays an important role. If model predicts that a movie will not be profitable (unsuccessful), it should say why is it going to be so; similarly for a profitable (successful) prediction. To justify predictions, we can incorporate Explainable AI techniques in future work, to describe these black box models. Using these techniques allows us to leverage advanced models like Neural Networks.

Piracy of a film also affects its financial recovery. Some people would prefer to watch a leaked film on the Internet rather than paying for the ticket and watching in theatres. This affects the size of audience and hence, can result in a huge loss to all stakeholders.

Furthermore, various intangible properties contribute to the success of a movie. A movie may be at the centre of some controversy if the messaging of the movie seems inappropriate to a particular class in the country. This could affect the box office performance of the film. Another drawback of predicting the success of movie based on subtitles is the large variation in how the movie would actually look and feel. The subtitle data won't convey any information regarding the parts played by the actors, so this could be a drawback.

6. References

- [1] Neelamegham, R., & Chintagunta, P. (1999). A Bayesian Model to Forecast New Product Performance in Domestic and International Markets. *Marketing Science*, 18, 115-136.
- [2] Simonoff, J., & Sparrow, I.R. (1999). Predicting Movie Grosses: Winners and Losers, Blockbusters and Sleepers. *CHANCE*, 13, 15 - 24.
- [3] Sharda, R., & Delen, D. (2006). Predicting box-office success of motion pictures with neural networks. *Expert Syst. Appl.*, 30, 243-254.
- [4] Mazurowski, M.A., & Szecowka, P. (2006). Limitations of sensitivity analysis for neural networks in cases with dependent inputs. *2006 IEEE International Conference on Computational Cybernetics*, 1-5.
- [5] Zhang, W., & Skiena, S. (2009). Improving Movie Gross Prediction through News Analysis. *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, 1, 301-304.
- [6] Lee, K., & Chang, W. (2009). Bayesian belief network for box-office performance: A case study on Korean movies. *Expert Syst. Appl.*, 36, 280-291.
- [7] Reddy, A., Kasat, P., & Jain, A. (2012). Box-Office Opening Prediction of Movies based on Hype Analysis through Data Mining. *International Journal of Computer Applications*, 56, 1-5.
- [8] Kaur, A., & Nidhi, A. (2013). Predicting Movie Success Using Neural Network.
- [9] Pangarker, N.A., & Smit, E. (2013). The determinants of box office performance in the film industry revisited. *South African Journal of Business Management*, 44, 47-58.
- [10] Parimi, R., & Caragea, D. (2013). Pre-release Box-Office Success Prediction for Motion Pictures. *MLDM*.
- [11] Apala, K.R., Jose, M., Motnam, S., Chan, C., Liszka, K.J., & Gregorio, F.D. (2013). Prediction of movies box office performance using social media. *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, 1209-1214.
- [12] Mestyán, M., Yasseri, T., & Kertész, J. (2013). Early Prediction of Movie Box Office Success Based on Wikipedia Activity Big Data. *PLoS ONE*, 8.
- [13] Hunter, S.D. (2014). A Novel Method of Network Text Analysis. *Open Journal of Modern Linguistics*, 4, 350-366.
- [14] Eliashberg, J., Hui, S.K., & Zhang, Z. (2014). Assessing Box Office Performance Using Movie Scripts: A Kernel-Based Approach. *IEEE Transactions on Knowledge and Data Engineering*, 26, 2639-2648.

- [15] Selvaretnam, G., & Yang, J. (2015). Factors Affecting the Financial Success of Motion Pictures: What is the Role of Star Power?
- [16] Kim, T., Hong, J., & Kang, P. (2015). Box office forecasting using machine learning algorithms based on SNS data. *International Journal of Forecasting*, 31, 364-390.
- [17] Taneja, H., Dewan, A., & Bhardhwaj, V. (2016). Pre-Release Success Quotient Prediction of Movies.
- [18] Chaudhari, N., Vardhajan, K., Shekhar, S., Thind, P., & Swarnalatha, P. (2016). A DATA MINING APPROACH TO LANGUAGE SUCCESS PREDICTION OF A FEATURE FILM.
- [19] Hunter, S., Smith, S., & Singh, S. (2016). Predicting Box Office from the Screenplay: A Text Analytical Approach. *Journal of Screenwriting*, 7, 135-154.
- [20] Chen, R., Xu, W., & Zhang, X. (2016). Dynamic box office forecasting based on microblog data. *Filomat*, 30, 4111-4124.
- [21] Lash, M.T., & Zhao, K. (2016). Early Predictions of Movie Success: The Who, What, and When of Profitability. *Journal of Management Information Systems*, 33, 874 - 903.
- [22] Hur, M., Kang, P., & Cho, S. (2016). Box-office forecasting based on sentiments of movie reviews and Independent subspace method. *Inf. Sci.*, 372, 608-624.
- [23] Magdum, S.S., & Megha, J. (2017). Mining online reviews and tweets for predicting sales performance and success of movies. *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, 334-339.
- [24] Subramaniaswamy, V., Vaibhav, M., Prasad, R.V., & Logesh, R. (2017). Predicting movie box office success using multiple regression and SVM. *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, 182-186.
- [25] Zhou, Y., Zhang, L., & Yi, Z. (2017). Predicting movie box-office revenues using deep neural networks. *Neural Computing and Applications*, 31, 1855-1865.
- [26] Quader, N., Gani, M.O., & Chaki, D. (2017). Performance evaluation of seven machine learning classification techniques for movie box office success prediction. *2017 3rd International Conference on Electrical Information and Communication Technology (EICT)*, 1-6.
- [27] Kim, T., Hong, J., & Kang, P. (2017). Box Office Forecasting considering Competitive Environment and Word-of-Mouth in Social Networks: A Case Study of Korean Film Market. *Computational Intelligence and Neuroscience*, 2017.
- [28] Ruhrländer, R.P., Boissier, M., & Uflacker, M. (2018). Improving Box Office Result Predictions for Movies Using Consumer-Centric Models. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

- [29] Zhou, Y., & Yen, G. (2018). Evolving Deep Neural Networks for Movie Box-Office Revenues Prediction. *2018 IEEE Congress on Evolutionary Computation (CEC)*, 1-8.
- [30] Dhir, R., & Raj, A. (2018). Movie Success Prediction using Machine Learning Algorithms and their Comparison. *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, 385-390.
- [31] Ru, Y., Li, B., Liu, J., & Chai, J. (2018). An effective daily box office prediction model based on deep neural networks. *Cognitive Systems Research*, 52, 182-191.
- [32] Verma, G., & Verma, H. (2019). Predicting Bollywood Movies Success Using Machine Learning Technique. *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 102-105.
- [33] Shreya Jayachandran, Sanika Tamhankar, Karishma Netake, Sayoojya Dinesan, and Dr. Sharvari Govilkar. (2019). Prediction of Movie Box-office success using NLP and Machine Learning Techniques, *JASC: Journal of Applied Science and Computations*.
- [34] Kim, Y.J., Cheong, Y.G., & Lee, J.H. (2019). Prediction of a Movie's Success From Plot Summaries Using Deep Learning Models.
- [35] Reddy, V. G., Reddy, K. R., Krishnamoorthy, A., Kannadasan, R., & Boominathan, P. (2020). Movie Prior Release Box Office Prediction A Machine Learning Based Approach. *European Journal of Molecular & Clinical Medicine*, 7(2), 5016-528.
- [36] Ahmad, I., Bakar, A.A., & Yaakub, M. (2020). Movie Revenue Prediction Based on Purchase Intention Mining Using YouTube Trailer Reviews. *Inf. Process. Manag.*, 57, 102278.
- [37] Wang, Z., Zhang, J., Ji, S., Meng, C., Li, T., & Zheng, Y. (2020). Predicting and ranking box office revenue of movies based on big data. *Inf. Fusion*, 60, 25-40.
- [38] Liao, Y., Peng, Y., Shi, S., Shi, V., & Yu, X. (2020). Early box office prediction in China's film market based on a stacking fusion model. *Annals of Operations Research*, 1 - 18.
- [39] <https://www2.deloitte.com/content/dam/Deloitte/in/Documents/about-deloitte/in-about-deloitte-economic-impact-of-the-film-television-and-osv-industry-noexp.pdf>