IJARIIT INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume3, Issue3)
Available online at www.ijariit.com

Prediction of Movie Success for Real World Movie Data Sets

Sanjai Pramod
VIT University, Tamil Nadu
sanjaipramod@gmail.com

Abhisht Joshi
VIT University, Tamil Nadu
abhishtjoshi46@gmail.com

Geetha Mary .A
VIT University, Tamil Nadu
geethamary@vit.ac.in

Abstract: Predicting the success of a movie before its release has far been a huge point of concern for directors and producers alike, especially after they factor in real world data and the occurrence of unforeseen circumstances. Since these entities invest massively in movies, they need some kind of reassurance that the movie will be successful and reap in massive profit in terms of financial returns and credibility. In this paper, we talk about a prediction engine we have implemented that uses classification and fuzzy logic to categorize a movie as successful or not as well as using our own algorithm to calculate a target variable which is the IMDB score of a movie based on various parameters. Our engine has proved to be successful from the technical unit testing performed on it with the use of multiple iterations of input values.

***Keywords**: Prediction, Data Mining, Classification, Fuzzy, IMDB, Normalization, Movie Dataset, Confusion Matrix.*

I. INTRODUCTION

There are vast movie data repositories available online which are very useful in the need to determine which upcoming movie will be popular and successful. However, the presence of these repositories haven't been exploited by movie Moghuls for the above-mentioned purpose i.e understanding the data and deriving valuable information from it before making or producing a movie. If this process of data mining was actually implemented by entities in the entertainment industry, it would be a crucial step toward ensuring the success of movies that they invest in. The movie success is predicted on the basis of ratings from millions of users available in a consolidated dataset that we have used, called the IMDB dataset. IMDB is a giant in the domain of rating and reviewing movies by using a Bayesian equation in which the ratings of multiple users is fed as the input for the movie, and the outcome is a generalized and accurate rating that depicts the user's view of the movie. One major assumption in this paper and project we have implemented is that the multiple user's consolidated reviews and rating of the movie are a clear depiction of whether the movie has performed well in the box office or not. Although we have enough evidence supporting this assumption, it is still worth noting that in future times, the features determining the success of a movie will change. We included various other factors that determine a movie's success as well including the entities involved such as actors and directors, the number of likes for the movie on social media sites like Facebook, and the total budget of the movies. To achieve these goals, we had to incorporate classification and fuzzy selection which are predominant methods used in most predictive engines. Through classification, we take various features like the entities present and classify each entity in the tuple's contribution to the success rating of the movie. Through fuzzy selection, we set 7 slabs of rating ranges and fuzzify the input into these ranges. Each of these ranges will depict a textual value to categorize the output. The inclusion of fuzzy into our implementation was for the purpose of removing a checking clause that will have three possibilities to rate a movie by. Now we dynamically categorize the movie into the fuzzy set without having to perform manual checks which improve the ability and efficiency of our engine. The final result of our implementation will contain the following items:

1) Original IMDB score
2) Algorithmically calculated IMDB score
3) Fuzzy categorization of the movie's success
4) Percentile of movies it falls under within that category
5) Accuracy range of our algorithm
6) Fuzzy Success.

II. LITERATURE SURVEY

Hassan and Hammad [20] talk about how they follow the functional steps of data extraction, data preprocessing, data integration and transformation, feature selection and finally classification like in [1]. They also used an IMDB dataset like in [2] and based on an algorithm designed by them, set parameters to classify the movie as a success or failure. Although their implementation has shown a high rate of accuracy in prediction, their algorithm has had drawbacks of bad time complexity, as the initial data retrieval takes a long time to create a training data set for even a few tuples of data. We plan on incorporating their idea and taking it ahead by adding our own algorithm to convert the string value of a classifying parameter, like 'actor name' or 'budget of movie' or 'language of movie' and so on, to a numerical value which will then be put into a broader formula in relation to all classifying parameters of the test data, and hence decide whether the movie will be successful or not. We will also incorporate a data set which contains real world historical data of movies so that we can operate our application in a real-time environment.

Zexi and Jiapeng [5] have developed an efficient method to predict whether a movie will be successful or not based on neural network algorithms. [3] Suggests neural network algorithms are a common approach to prediction applications. They have adapted some new ideas into their method by adding unique classification parameters like technical specifications, the screenplay of the movie and so on, which were not included in older predictive methods like in [4] and [5]. However, the biggest drawback of their approach was that their method was restricted only to the Chinese box office and hence the historical data being mined was a data set based only on Chinese movies. Due to this, their application will have to be further scaled to incorporate a global movie database and hence have a more international use case. This is what we will achieve in our project, as we are using a database of movies that are internationally acclaimed and not just restricted to one country. We have also added our own different classifying parameters that have not been used before, like years of experience of actor and also the year of release of the movie.

Hina [6] has explained how predictive algorithms are necessary in real world scenarios to prevent students from dropping out of college by strictly focusing on the hidden knowledge in the educational data system. She has used data from Indira Gandhi National Open University and talked about various data mining algorithms used for prediction. There is a clear specification on using feature selection on these data sets to reduce the data set to a specifically trained tuple and then perform classification algorithms on it [6]. We are implementing this idea in our project as we are also performing feature selection on our movie data set to produce numerical values of each classifying parameter, and only after that do we perform our classification algorithms. The basic idea behind a classification algorithm is to have strongly weighted classification parameters to guide the number of leaves generated by the classification tree. Hina has used 'weka' as her data mining tool, while we are using 'orange'. We have selected orange because of its multilayer architecture interface as well as its canvas that allows a drag and drop feature in terms of adding widgets and generating classifiers, dendrograms, and histograms [7].

Tom Snee [8] takes a look at a prediction system developed by Kang Zhao which introduces machine learning into its predictive techniques to analyse the factors that determine whether a movie is successful or not. A major drawback we've noticed is that they used the budget of the movie as a primary feature which can sometimes backfire on the success of a movie. Case in point: Batman Vs Superman had a $200 million budget [8] but failed to perform well with critics. Hence, we can come to a conclusion that although the budget of a movie is an important classifying feature in a movie's success, it cannot be a primary feature (i.e., a main deciding feature). Thus, we have taken this understanding and implemented it in our code by adding low weights to the budget factor so it doesn't have adverse effects on our output.

Manish Rama [9] talks about a method that uses Example-Based Machine Translation (EBMT) [9]. Example-based machine translation (EBMT) is a method of machine translation often characterized by its use of a bilingual corpus with parallel texts as its main knowledge base at runtime [9]. The input that is retrieved from the front-end is then tokenized using Natural Language Processing [10]. Natural language is a fundamental thing for human society to communicate and interact with one another. In this globalization era, we interact with different regional people as per our interest in the social, cultural, economic, educational and professional domain [10]. The system is capable of translating a string in one language to another already existing language. The application determines exact substrings from the user-input string and relates these substrings to the already existing substrings in of the target language. The fuzzy-wuzzy module of python is being utilized in order to determine the ratio of substrings in the existing language translating and translator languages.

Akshay Nagpal [12] focuses on providing a timeline in accordance to which career path should be followed by young individuals currently a little confused about what they should pursue. It takes the career goals [12] and current education [12] as the inputs and performs string matching with datasets to determine relevant work-positions [12] and career options as the next step. The datasets used have been prepared by extracting information from LinkedIn and online surveys performed by experts in their respective domains. The ratio percentage is calculated, this is the match percentage which has career goals and current education as input strings. Accordingly, the results are displayed.

III.THE SYSTEM MODEL

The proposed system is as shown below. The presentation layer is where the user interacts with the engine and gives his input values for the movie whose prediction we must do. The business/application layer is where the functionalities occur, the python script is called to perform the logic and trigger the algorithm, and data is read from the database layer and manipulated in the business layer and finally, the required output values are sent back to the presentation layer for the user to view.

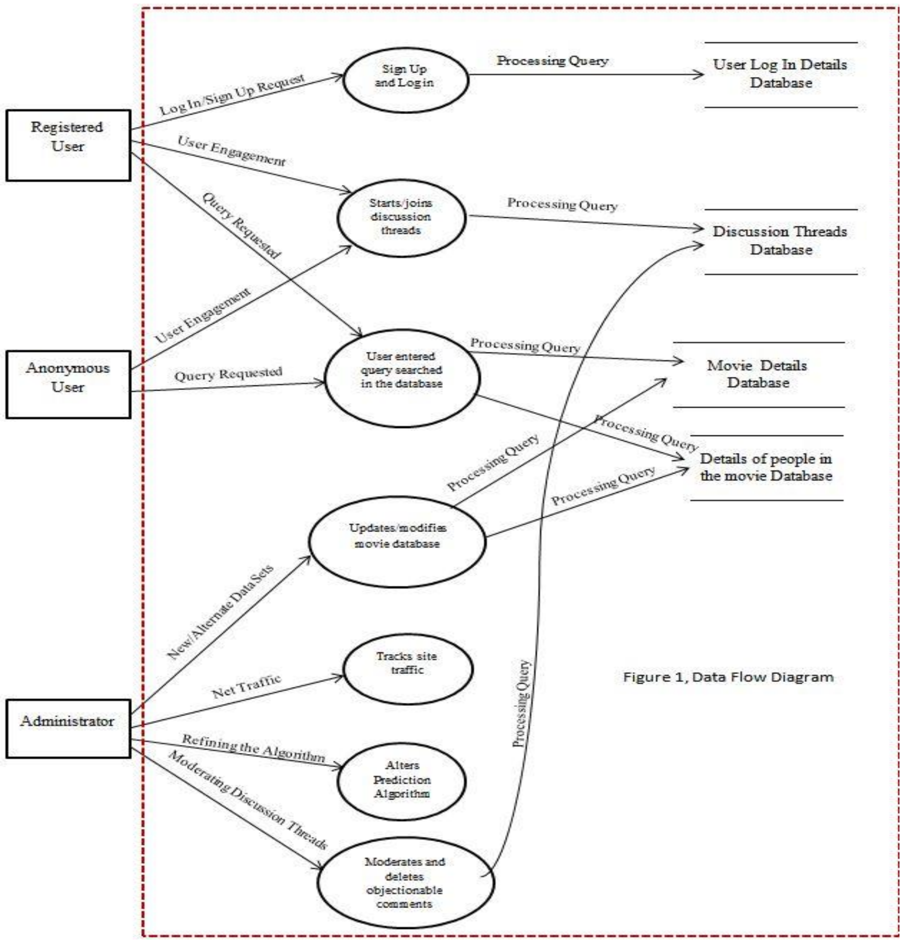


Fig. 1 System Architecture: Divided into the UI, back-end contains the Prediction Engine and PHP Dynamic website and database for both

The Prediction Engine is a module inside the website which functions independently and has a different data access and storage mechanism altogether.
Prediction Mechanism is divided into the following steps:

- Prompt the user to input the movie title, actor and director names.
- Save the file.
- Submit the form.
- The back-end interfacing saves it to the host computer.
- Executes the prediction algorithm.
- Prediction algorithm accesses the two data repositories: Historical Movie data set(IMDB Dataset), user input data
- The algorithm retrieves the information of other movies done by the actors and directors based on this calculates the weights. It also considers the budget, gross and profit of the movie, if the movie already exists in the database. If not it utilizes the other factors and predicts the success of a future movie.
- Ultimately the output generated by the algorithm contains various weight ages based on various factors and the final rating. If the movie exists in the database it compares the rating generated algorithmically with the IMDB score and showcases the accuracy as well. This accuracy is likely to never be beyond 95% as the IMDB score considers the user ratings while the prediction algorithm considers the user ratings in the form of IMDB weights for each actors and other major factors like gross, budget, profit, previous performance of the cast in their other movies.
- The output is then displayed to the user in a consolidated and categorical fashion in the front-end.

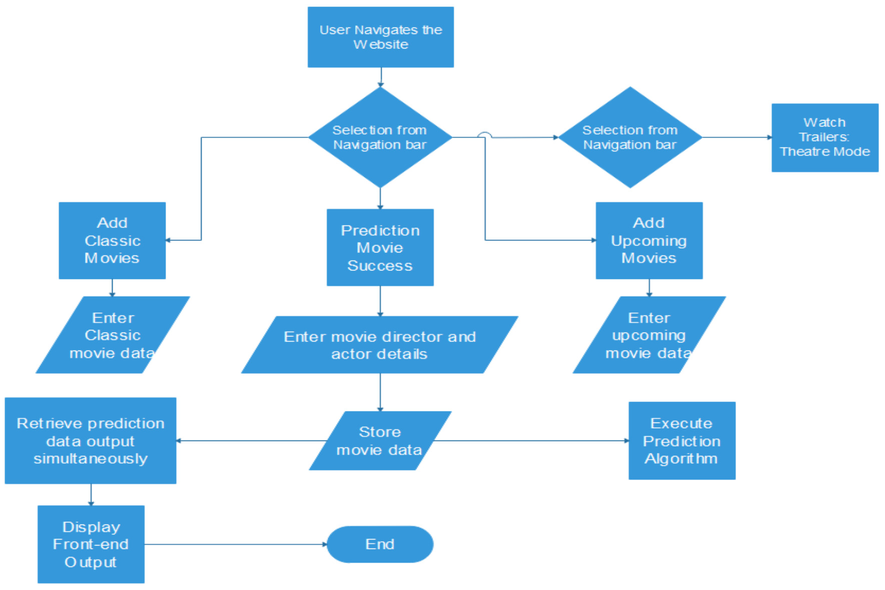

User Navigates the Website
Selection from Navigation bar
Selection from Navigation bar
Watch Trailers: Theatre Mode
Add Classic Movies
Prediction Movie Success
Add Upcoming Movies
Enter Classic movie data
Enter movie director and actor details
Enter upcoming movie data
Retrieve prediction data output simultaneously
Store movie data
Execute Prediction Algorithm
Display Front-end Output
End

Fig. 2 Control flow diagram: Adding movies and prediction modules have various functions and files linked to each module, from the navigation bar we can add classic movies, upcoming movies and predict the success of movies.

IV. EXISTING METHODS

A common approach to handling predictive requirements is by using neural networking algorithms. Backpropagation is one of these algorithms. In backpropagation, an input is taken from the user and propagated forward through each layer until it reaches the output. At the output phase, it is then compared with a loss function and the error is pushed back in the reverse direction of the layers such that now each weight added to the layer nodes will have an associated error value to it [15]. These error values will now help in improving the data that was trained so that when a new input vector comes, the loss in the output accuracy will keep minimizing. The backpropagation method is very effective in improving a dataset [16], however, we haven't used it in our tool, as the data is given by the user may or may not be real-time data. Due to this fact, we can't propagate the user input back into the training set until validation is then assuring that the movie data actually exists in real-time. This validation is unfeasible since we would have to scour through petabytes of data to find a hit, which is not time optimal.
Another method that exists is social network analysis which is also a highly effective mode of collecting data to predict the success of a movie. The idea of social network analysis (SNA) is that it will crawl through social networking platforms like Facebook, Twitter, and Instagram and so on and gather all information regarding the particular movie through a fuzzy search of keywords [17]. These keywords will be purely unique to the movie so that the algorithm doesn't collect irrelevant information. This data collected will then be analyzed by various functions to gather the hype around the movie, the 'commercial noise' that the movie generates and how eager real-world movie watchers are waiting for the movie's release. We have incorporated this idea mildly in our algorithm by taking 'Facebook likes' of the movie and its associated actors as a classifying feature and assigning weights to these values accordingly. This actually helps us understand the hype for the movie and as we have observed through our research, the more hype a movie has, the more destined it is to succeed.
Using decision trees are also a largely prevalent predictive method used today for data mining. Decision trees consist of a root node and subsequent child nodes, each of which has an associated path and predictive value/category [18]. This tree gets formed by analysis the data set and passing it through a classification tree function. This function will then create the tree node by node, as it decides which feature has the most dominance in assuring the success of a movie and which has the least. According to its decision, a hierarchical structure is formed based on multiple branches and leaves. Ultimately, every node in the decision tree will have some percentage in deciding the success of a movie. So we accordingly assign weights to each feature depending on which level it is in the decision tree. Supposing, feature A is at level 0 or the root level, and feature B is at level 4, that is it has a degree of separation from feature A as 3, we will assign feature A with a higher weight and feature B with a relatively lower weight. In this way, the decision tree will help us understand the dataset and how we can use each field to classify and predict our input.

V. TECHNIQUES AND CALCULATIONS

Fuzzy Logic: Developing and maintaining an understanding of technical subjects across a broad range of areas means that you will have more knowledge and skills to attack problems when they appear and formulate innovative solutions to those problems [11]. The result we obtain from our algorithm is in the form of a normalized rating between the ranges of 1 to 10. The perennial challenge faced is when a calculated value has to be expressed in terms of how an actual user would rate the movie i.e., a good movie or a pretty decent movie. The determination when a rating between 0 to 10 categories the movie into a good movie or it brings to the movie to a category of the pretty decent movie is where we intend to apply fuzzy string matching. In this context fuzzy string matching is used to determine an expressive linguistic review for a numerically calculated rating within the range of 0 to 10. The movie rating which is calculated using the various weights is now subjected to fuzzy string matching. Consider a set R, which gives the range according to the rating. R has a size of seven and contains humanly expressive terms to determine the movie success rate of a particular movie.

$$R_i : \{Flop, Bad, Watchable, Decent, Good, Great, Amazing\} \text{ where } i: 1,2, \dots 7 \qquad (1)$$

The success rate of a movie in the set *R* increases as we increment *i* in the set. The slab for each element in the set R is within a *numerical rating range*. The rating *Rate* calculated by the prediction algorithm is then compared to the set *R* and according to the rating, it will have the elements in the set *R* appends to *fuzzy_rate* string, where *fuzzy_rate: {i in R, i: 1, 2…7}*. The higher the movie rating, more number of elements from set *R* will be appended to the above-mentioned string. The descriptiveness of *fuzzy_rate* is connected to the movie success rate in the following manner. Let *k* be the length of the string *fuzzy_rate:*

$$k \; \alpha \; Successful \; Movie \qquad (2)$$

TABLE 1 List of variables used in fuzzy logic system

S.no	Range	Category
1	0.0-2.9	Flop
2	3.0-4.9	Bad
3	5.0-5.9	Watchable
4	6.0-7.7	Decent
5	7.8-8.5	Good
6	8.6-8.9	Great
7	9.0-10.0	Amazing

Normalization: The normalization formula used is shown below [19].

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \qquad (3)$$

$$where \; x = (x_1, x_2, \dots, x_n) \; and \; z_i \; is \; the \; i^{th} \; normalized \; value$$

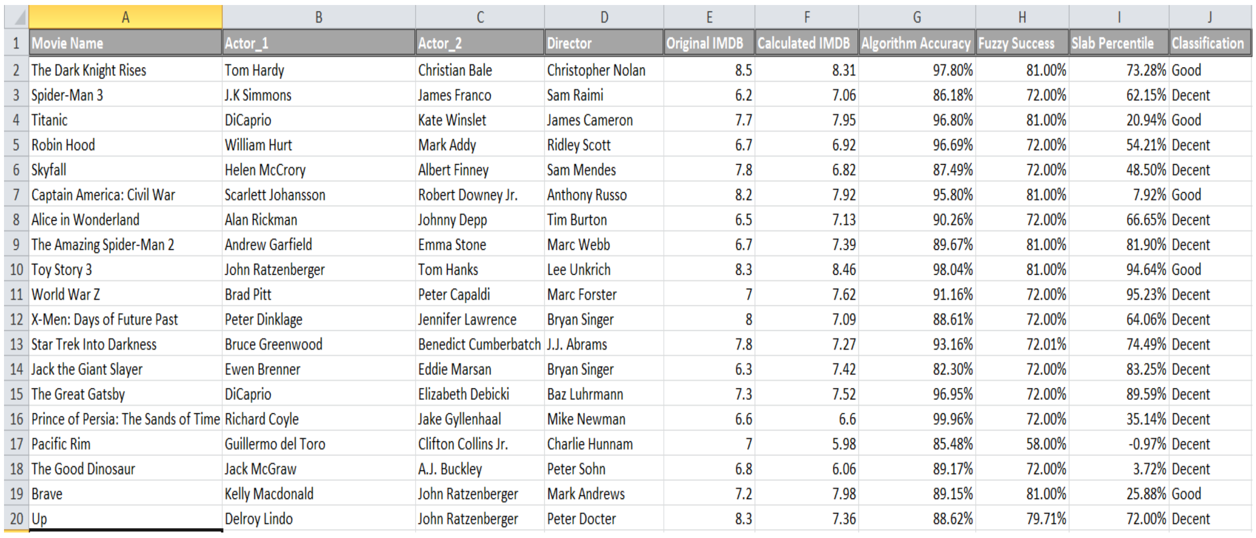
Pseudo Code: Prediction Algorithm

```
1:  Begin
2:  Function rate_movie: Calls the function get_user_input( ) and stores the user input
3:  Open the historical data set csv file
4:  Iterate through each row of the historical data set
5:  experience_factor( ) calculates the number of hits are equivalent to the number of movies done by cast
    so far
6:  dir_imdb_score, a1_imdb_score, a2_imdb_score are three lists which store the imdb scores of all the
    movies done by the actors and director of the movie title which needs success prediction
7:  dir_high_imdb_score, a1_high_imdb_score, a2_high_imdb_score are lists which store the imdb rates
    of the same director and actors respectively but the difference being the only those imdb scores are
    stored that are above 7.8. Thus great artist factor comes into play. imdb_weight( ) calculates the
    weights
8:  fame_rating( ) calculates the fam_factor by taking the number of voted users, number of critic reviews,
    number of user reviews, cast facebook likes, movie facebook likes together and normalizing it on a
    scale of {0.0...1.0}
9:  profit( ) calculates Profit = Gross - Budget, for each movie in the historical data set this is calculated
10: profit_scale ( ) scales down the profit to a range of {0.0...1.0}.
11: Set A= max(profit) , B=min(profit)
12: Set: profit=(0.1 + (profit-A)*(1.0-0.1)/(B-A))
13: Set imdb_factor for each actor and the directors to (average_score/10.0)*imdb_weight
14: Set imdb_rate to average of imdb factors of each artist
15: Set experience_rate to average to experience_factors of actors and director
16: movie_rating equals to average of all above calculated factors multiplied by respective weights and
    scaled up to a value between 1-10
17: Suggest ( ) generates suggestions according to the user input movie.
18: Display suggestions
19: Display movie_rating
```

```
20: Define
    $R_i: \{Bad, Flop, Watchable, Decent, Good, Great, Amazing\}$ where $i: 1,2,\ldots 7$ corresponding to
    $\{[0-2.9], [3-4.9], [5-5.9], [6-7.7], [7.8-8.5], [8.6-8.9], [9.0-10.0]\}$
21: Percentile of a particular movie rating in the particular slab range is calculated
22: Display Prediction i.e., one among the values in set R along with percentile of the movie in that
    particular slab range. Example 76% Good
23: if movie_rating is less than equal to ori_im
24: then accuracy equals (movie_rating/ori_im)*100
25: else accuracy equals 100.0-(((movie_rating-ori_im)/ori_im)*100)
26: Display accuracy
27: Success rate is calculated on the basis of fuzzy string matching method
28: Ideal success is matched with the success achieved by the movie entered into the prediction engine.
    This gives the success rate in percentage
29: Display success rate
30: End
```

VI. EXPERIMENTAL ANALYSIS

The input taken for data analysis was taken as movies from IMDB, rotten tomatoes and Wikipedia. As we needed data from sources to provide information about our classification features i.e Actor 1 name, Actor 2 name, Director Name and Movie Name, we required reliable websites and data sets of information. Multiple test cases were run to perform testing to verify the results with that of the IMDB and see how well our algorithm has fared. Below are some of the results we obtained.



	A	B	C	D	E	F	G	H	I	J
1	Movie Name	Actor_1	Actor_2	Director	Original IMDB	Calculated IMDB	Algorithm Accuracy	Fuzzy Success	Slab Percentile	Classification
2	The Dark Knight Rises	Tom Hardy	Christian Bale	Christopher Nolan	8.5	8.31	97.80%	81.00%	73.28%	Good
3	Spider-Man 3	J.K Simmons	James Franco	Sam Raimi	6.2	7.06	86.18%	72.00%	62.15%	Decent
4	Titanic	DiCaprio	Kate Winslet	James Cameron	7.7	7.95	96.80%	81.00%	20.94%	Good
5	Robin Hood	William Hurt	Mark Addy	Ridley Scott	6.7	6.92	96.69%	72.00%	54.21%	Decent
6	Skyfall	Helen McCrory	Albert Finney	Sam Mendes	7.8	6.82	87.49%	72.00%	48.50%	Decent
7	Captain America: Civil War	Scarlett Johansson	Robert Downey Jr.	Anthony Russo	8.2	7.92	95.80%	81.00%	7.92%	Good
8	Alice in Wonderland	Alan Rickman	Johnny Depp	Tim Burton	6.5	7.13	90.26%	72.00%	66.65%	Decent
9	The Amazing Spider-Man 2	Andrew Garfield	Emma Stone	Marc Webb	6.7	7.39	89.67%	81.00%	81.90%	Decent
10	Toy Story 3	John Ratzenberger	Tom Hanks	Lee Unkrich	8.3	8.46	98.04%	81.00%	94.64%	Good
11	World War Z	Brad Pitt	Peter Capaldi	Marc Forster	7	7.62	91.16%	72.00%	95.23%	Decent
12	X-Men: Days of Future Past	Peter Dinklage	Jennifer Lawrence	Bryan Singer	8	7.09	88.61%	72.00%	64.06%	Decent
13	Star Trek Into Darkness	Bruce Greenwood	Benedict Cumberbatch	J.J. Abrams	7.8	7.27	93.16%	72.01%	74.49%	Decent
14	Jack the Giant Slayer	Ewen Brenner	Eddie Marsan	Bryan Singer	6.3	7.42	82.30%	72.00%	83.25%	Decent
15	The Great Gatsby	DiCaprio	Elizabeth Debicki	Baz Luhrmann	7.3	7.52	96.95%	72.00%	89.59%	Decent
16	Prince of Persia: The Sands of Time	Richard Coyle	Jake Gyllenhaal	Mike Newman	6.6	6.6	99.96%	72.00%	35.14%	Decent
17	Pacific Rim	Guillermo del Toro	Clifton Collins Jr.	Charlie Hunnam	7	5.98	85.48%	58.00%	-0.97%	Decent
18	The Good Dinosaur	Jack McGraw	A.J. Buckley	Peter Sohn	6.8	6.06	89.17%	72.00%	3.72%	Decent
19	Brave	Kelly Macdonald	John Ratzenberger	Mark Andrews	7.2	7.98	89.15%	81.00%	25.88%	Good
20	Up	Delroy Lindo	John Ratzenberger	Peter Docter	8.3	7.36	88.62%	79.71%	72.00%	Decent

Fig. 3 A screenshot of an excel file containing test cases of the prediction engine. Each field is an entity in the output screen.

Error study: Our algorithm sometimes still manages to have anomalous values that do not match with real world ratings. These values although few, fall on the extreme lines of inaccuracy and hence we've observed these through a confusion matrix [13].

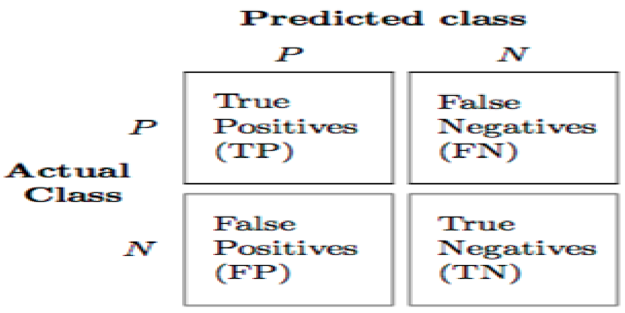


Fig. 4 A traditional approach to the confusion matrix showing an Actual Class vs Predicted Class matrix consisting of True Positive, False Negative, False Positive and True Negative.

What this means is, movies with 85%+ algorithm accuracy on high imdb scores fall under true positives, which are supposed to be successful and our engine predicts them as successful. False negatives [14] are movies that are supposed to be good but have been predicted as flops. Similarly, false positives are movies that are supposed to be flops but our engine predicts them as good and true negatives are movies that are supposed to be flops and our engine has predicted them as flops with an 85%+ algorithm accuracy. The reason for the presence of False Negatives and False Positives is the presence of unpredictability in real world scenarios. Even though through past data and facts, the movie on paper is supposed to be successful, a few unforeseen circumstances such as script, offensive content, poor execution and infamous publicity will make the movie produce false results.

CONCLUSION

From the experimental analysis, we could understand that the prediction of movie success is indeed possible with high percentages of accuracy. So by using our prediction engine, producers can evaluate beforehand if the movie is worth investing in and accordingly make their decisions. We also observe that fuzzy logic is an effective means of categorizing predictions and adds more accuracy and dynamicity to the engine. Although there are a few cases where the obtained result does not match with the expected result, they can be considered negligible when compared with the larger percentage of accurate results.

REFERENCES

[1] Mrs. Bharati M. Ramageri, "DATA MINING TECHNIQUES AND APPLICATIONS", Indian Journal of Computer Science and Engineering, Vol. 1 No. 4, pp.301-305, 2011

[2] Steven Yoo, Robert Kanter, David Cummings TA, Andrew Maas, "Predicting Movie Revenue from IMDb Data", pp.1-5, 2011

[3] Kanchan "Advanced Neural Network Algorithms for Prediction", International Journal of Science and Research (IJSR), Volume 3 Issue 12, pp. 441-444, December 2014

[4] Jeffrey Ericson and Jesse Goodman, "A Predictor for Movie Success", CS229, Stanford University", 2014

[5] Karl Persson ECTS, "Predicting movie ratings, A comparative study on random forests and support vector machines", pp.1-28, 2015

[6] Janez Demsar and co, "Orange: Data Mining Toolbox in Python", Journal of Machine Learning Research, Volume 14, pp.2349-2353, 2013

[7] K.Sutha, "A Review of Feature Selection", International Journal of Innovative Research in Computer and Communication Engineering, Vol. 7 No.6, pp.63-67, 2015

[8] Scott Mendelson, "Box Office: Batman V Superman: Dawn of Justice Was A $855M Wash", Forbes, pp. 1-2, 2016

[9] Manish Rama, Mohammad Atique, "Enhancing Bi-Lingual Example-Based Machine Translation Approach", International Journal of Advanced Engineering Research and Science (IJAERS) Vol-3, No. 10, 2016,

[10] Chandranath Adak, "A bilingual machine translation system: English & Bengali", IEEE, pp.1-4, 2014

[11] William Smith, Helen Sun, "Advanced Data Acquisition and Processing",
Pro Salesforce Analytics Cloud, pp.141-155, 2016

[12] Akshay Nagpal, Supriya P.Panda, "Career Path Suggestion Using String Matching and Decision Trees", International Journal of Computer Applications, Vol.117 No.7, 2015

[13] A.K Santra, "Genetic Algorithm and Confusion Matrix for Document Clustering", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 2, pp.322-328, 2012

[14] Maria Navin, "Performance Analysis of Neural Networks and Support Vector Machines using Confusion Matrix", IJARSET, Vol.3 No.5, pp.2106-2109, 2016

[15] Parminder Kaur, "A Comparative Study of Different Neural Networks Learning Algorithms for Forecasting Indian Gold Prices", IJARCCE, Vol.5 Issue 4, pp.1086-1090, 2016

[16] Yang Liu, "Parallelizing Backpropagation Neural Network using Map Reduce and Cascading Model", CIN, Vol.2016, pp.1-11, 2016

[17] Lule Ahmed, "A bimodal social network analysis to recommend points of interest to tourists", Springer, 2016

[18] Chung-ping Zhang, Hao Wu, "Research of Decision Tree Classification Algorithm in Data Mining", IJDTA, Vol.9 No.5, pp.1-8, 2016

[19] Navdeep Kaur, "Normalization Based K-means Data Analysis Algorithm", IJARCSSE, Vol.6, pp.455-457, 2016

[20] Hassan and Hammad, "Prediction of Movie Popularity Using Machine Learning Techniques", IJCSNS, Vol.16 No.8, pp.127-131, 2016