# Applying Supervised Learning Models to Movie Review Rating Prediction

**Naman, Jeb, Ikhoon, Yi Da**

National University of Singapore (NUS)

## Introduction

The report examines the efficacy of various machine learning models for predicting movie ratings based on review text sentiment. Rating prediction models based on text are crucial for developing commercial content recommendation algorithms. When ratings are explicitly provided, like on Netflix, companies may implement a collaborative filtering algorithm by analyzing user data, such as past reviewed movies (Kumar (2021)). For instances where no rating data is provided, trained models could help predict the sentiment of text in contexts where no sentiment is provided. For example, sentiment analysis models could be run on YouTube video comments, which determines whether the video should be recommended to viewers. Predictions could either be binary or multi-class representations.

Previous research have gradually improved the accuracy of movie rating predictions. **?** Miedema and Bhulai (2018) hypothesized that long-term memory models for movie review text ameliorates the overall accuracy. The research was conducted by implementing LSTM with IMDB movie reviews to predict their ratings in binary representation. Training with 15 epochs, it was concluded that LSTM predicts the sentiment with a high accuracy of 86.25%. However, this methodology lacks optimization processes for hyper-parameters and has poor time efficiency. To address these issues, modified supervised algorithms are devised. For instance, Rehman et al. (2019) implemented hybrid CNN-LSTM model on the IMDB review, which produces a 91% accuracy with better memory and time efficiency. Recent works strive to devise multi-class algorithms such as memory-based deep RNN and multi-class LSTM.

In this project, we apply generative Bayesian, TFIDF vectorized logistic, Vader weighted multi-layered LSTM, and random forest models to obtain sentiment scores and use them to perform binary and multi-class rating predictions. To construct them, we transform movie reviews to numerical vectors by employing multiple transformation methods including frequency count vectorizer, Term Frequency–Inverse Document Frequency (TFIDF) vectorizer and dense word embeddings. Fitted models are optimised for their architectural hyper-parameters through validation strategies. Finally, models are compared based on their prediction accuracy to evaluate which gives the best predictions.



## Dataset Description & Preliminary Analysis

The dataset used is the "Movie Review Dataset". The dataset consisted of training (25k) and testing data (25k), which were each divided into negative sentiment (rating 1-4) and positive sentiment (rating 7-10). Pre-processing the data required us to transform the data from a collection of segregated text files to a randomly shuffled tabular test and train datasets for supervised learning. The datasets were checked for null values, followed by deploying various visualization techniques to gain insight into their distribution, attributes and possible challenges. For the review text, we filtered stop words (filler words such as 'the' and 'is') as this gave the best accuracy results. Comparing different proportions of validation to training data size, smaller proportions of validation size led to over-fitting data. In comparison, larger proportions were not implemented given the large size of the training data. Thus, we performed a 20-80 split of validation to training data. The validation data provides an unbiased evaluation of the train data model fitting.

Constructing models on the data set was restricted by the challenges posed by the data set including but not limited to large dataset size and associated difficulty in implementing array-based tokenized representations, multi-class nature, and the possibility of neural nets being unable to recognize long-term dependencies owing to very long sentences for some reviews.

Three methods were used to encode sentences into numeric vectors that could be interpreted by the algorithm depending on the model used: Counter Encoding, TFIDF Vectorizers & Word Embeddings (check Appendix).

## Methods

### Bayesian Modelling

Bayesian modelling allows us to interpret the probability of a class conditional on the given data, which provides a principled way of combining prior information with data in a sound theoretical framework.

$$\log\left(P(y \mid x_1, \ldots, x_n)\right) \propto \log\left(P(y)\right) + \sum_{i=1}^{n} \log\left(P(x_i \mid y)\right)$$

Unlike many other time-consuming algorithms, Bayesian models usually work very efficiently on large and high-dimensional datasets. This is particularly useful given the large size of our data and the multiple dimensions owing to a count vectorizer transformation. While coming up with Bayesian models for the rating prediction, we optimised for the distribution of our dataset (identifying a suitable Bayesian model) and tuned the associated smoothing parameter. Gaussian, Multinomial and Bernoulli Bayesian models were implemented (which determined $P(x_i|y)$), and correspondingly the best model was tuned for its Laplace smoothing parameter ($\alpha$). The Bernoulli and Multinomial models gave the best performance for the binary classification and multi-classification problems respectively. Appendix fig. 1 shows the plot of variation of validation set error against the smoothing hyperparameter.

### Logistic Regression

A single perceptron model/logistic regressor is easily implementable on the TFIDF vectorized data. The regressor works by fitting a logistic function on the binary data (equivalently, the softmax function on the multi-class data) that allows us to extract probability values for the observation in a particular class. Unlike Naive Bayes, this model is discriminative and fits the data through the method of maximum likelihood estimation that seeks to minimize the Kullback-Leibler divergence between plausible probability distributions.

### Multi Layered LSTM Model

Exploratory data analysis (Appendix fig. 2) showed a high value of average sentence lengths for the data set. Higher lengths make simple recurrent neural nets incapable of accounting for long-term word dependencies, making it pivotal to incorporate LSTM frameworks. LSTM's quadruple linear interaction on a steady conveyor belt of information makes it a great fit for our model. Before configuring the LSTM model, we computed the VADER (Valence Aware Dictionary for sEntiment Reasoning, Hutto and Gilbert (2015)) scores for every sentence (including stop words as Vader considers both preceding and following words in computing the polarity score). VADER scores allow us to map lexical features to varying emotional intensities to obtain a normalized sentiment score for every review. It is computed as follows:

$$V_i = \frac{x_i}{\sqrt{x_i^2 + \alpha}}$$

where $x_i$ denotes the sum of valence scores (sentiment ratings) of constituent words, and $\alpha$ denotes the normalization constant. The Vader scores were accumulated with the Tokenized representation of sentences and fed to the LSTM model. Potential pitfalls of the model included overfitting (Appendix fig. 3) which required us to reduce model complexity (decreasing the LSTM cell number); additional dropout layer (to occasionally drop neurons during training); and early stopping criteria which would interrupt the training after a decline in validation accuracy. The model structures are in Appendix fig. 4. The LSTM models were built and trained for binary and multi-classification problems, both with and without Vader Scores metric. Distinctively, bidirectional LSTM architectures were also deployed to process information in both backward and forward directions.

### Random Forest Model

Random forests (Ali et al. (2012)) was our first attempt at building ensembling models to improve our prediction accuracy. These forests are constructed by aggregating the predictions from many decision tree classifiers, which impose linear hyperplane segregations on the vector space spanned by the encoding. Their decision boundaries are optimised to reduce prediction entropy, maximizing their potential.

$$H(X) = \mathbb{E}[-\log p(X)] = -\int_{x \in \chi} p(x) \log(p(x))\, dx$$

These models were further tuned for various hyperparameters (number of trees, max depth, max features, and minimum samples per leaf), optimising their performance (Appendix fig. 5).

## Results & Discussions

Model performances on binary classification were evaluated on validation accuracy. Multi-class predictions were evaluated using Root Mean Squared Error (RMSE) to give weight to predictions close to true output values. Multiple runs were carried out for Bayesian, Logistic and Random Forest models (to ensure model stability), but few for LSTM owing to its high time complexity. Fine-tuning was implemented by optimizing hyperparameter values, randomized data splitting and validation set creation. A summary of the model performances for binary and multi-class predictions has been shown in Appendix tables 2 and 3. For the Binary prediction, the Random Forest model performed best with 96.67% testing accuracy followed by the Logistic model at 92.52% accuracy. Their high performance was likely due to the usage of the TFIDF vectorizer. Alternate methods have a steep drop-off regarding testing accuracy, approximating 85.00%. The lower accuracy may result from non-ideal model training parameters (owing to the vast possibility of parameter combinations) and their inherent complexity/tendency to over-fit.

For the Multi-class prediction, the trend was similar: Random Forests performed best with 1.57 testing RMSE, followed by logistic regression performed best with 1.95 testing RMSE. Alternate methods have a steep drop-off regarding testing RMSE, ranging from 2.82 testing RMSE for Bayesian models to 3.45 for LSTM.

There are numerous possibilities for future work including experimenting with different training-testing-rations, cross-validation for hyperparameter tuning, employing algorithms such as Support Vector Machines, BERT (Hutto and Gilbert (2015)) or GPT-3 transformers (Devlin et al. (2018)), examining user-level features such as user rating patterns, context analysis of review, and weighted ensembling models to combine predictions from different models.

## References


Ali, J.; Khan, R.; Ahmad, N.; and Maqsood, I. 2012. Random Forests and Decision Trees. *International Journal of Computer Science Issues*, 9.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Hutto, C.; and Gilbert, E. 2015. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. In *Proceedings of the 8th International Conference on Weblogs and Social Media*.

Kumar, P. 2021. Netflix Movie Recommendation System.

Miedema, F.; and Bhulai, S. 2018. Sentiment Analysis with Long Short-Term Memory Networks. *arXiv*.

Rehman, A. U.; Malik, A.; Raza, B.; and Ali, W. 2019. A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis. *Multimedia Tools and Applications*, 78.