IMDb Movie Review Sentiment Prediction

Daphne Hidley (UID: 105502680), Kaylin Lee (UID: 906036630),
Megha Velakacharla (UID: 705587841), Noor Benny (UID: 605624764),
Rachel Ki (UID: 005590842)

Department of Statistics and Data Science UCLA

Abstract

IMDb, short for the Internet Movie Database, is an online database containing information about popular movies and television shows, including user ratings, reviews, plot summaries, cast lists, and more. Our objective is to explore the relationship between movie reviews and their associated sentiments, aiming to devise a methodology for predicting review sentiments. We will implement classification methods such as logistic regression, KNN, LDA, QDA, and random forests to build models capable of predicting sentiment from the words present in each review.

⋆ ⋆ ⋆ ⋆ ⋆

1 Introduction

Seeking cinematic guidance? IMDb is an online database that houses statistics for over 625,000 movies and consists of written reviews contributed by users **[cite imdb thing]**. Sifting through nearly eight million reviews within this extensive collection presents an opportunity to explore diverse viewpoints. For those curious about films they have not yet seen, IMDb reviews prove invaluable in gauging potential enjoyment based on fellow users' written opinions. These reviews typically vary in length and detail, reflecting users' perspectives on whether they liked or disliked the movie. Our project involves analyzing a dataset of 50,000 IMDb reviews, each categorized by sentiment (negative or positive). Our objective is to explore the relationship between reviews and their sentiment. By evaluating different classification models, we aim to identify the model that optimizes the accuracy of predicting a review's sentiment based on its content.

2 Preprocessing Step

To prepare the dataset of 50,000 IMDb reviews for classification modeling, the data was cleaned and processed in Python. We started with one file containing two columns: one column contained the reviews, while the other indicated the true sentiment of each review.

Using the tidyverse and tidytext packages in R, we first applied a tokenization process to the data which separated each review into the individual words contained in it. We proceeded by removing common stop words such as "a", "the", "is", and "are" from the reviews to leave behind only the meaningful words. Additionally, any numbers, special characters, pronouns, and line breaks (specifically the "" tags left over from HTML) were removed while converting all text to lowercase for consistency. Finally, in order to refine the overall linguistic quality of the reviews, adverbs were transformed into adjectives using the udpipe package in R. For example, words like "happily" and "quickly" were changed to "happy" and "quick", respectively.

2.1 Descriptive Statistics

Once the data had been properly cleaned, we were able to begin our preliminary analysis and visualization of the reviews. This would help us to gain a better understanding of how the data looked.

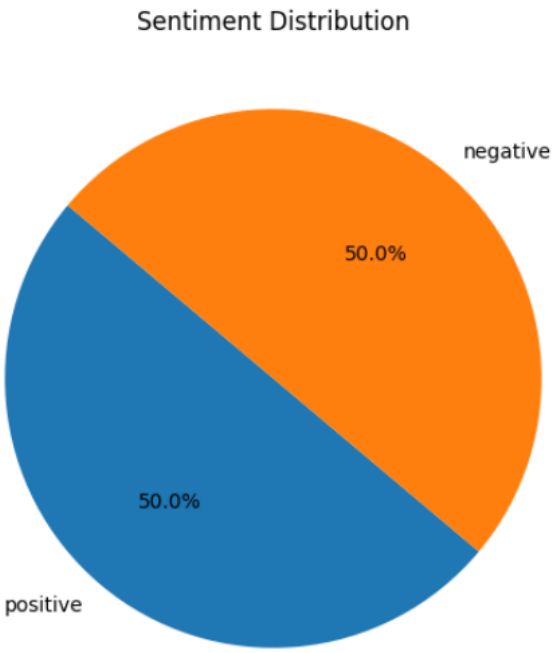


Figure 1: Sentiment Distribution

We began by tallying the count of reviews expressing positive and negative sentiments to assess the balance between each. Our analysis revealed an equal number of positive and negative reviews within the dataset, establishing a well-balanced foundation for sentiment analysis. This equilibrium contributes to ensuring fairness in accurately understanding and predicting sentiments.

Figure 2: Word Cloud

We also created a word cloud to quickly and intuitively visualize the most frequent or prominent words in the set of reviews. Commonly used words like 'movie' and 'film' appear frequently but seem less critical in predicting sentiment compared to more emotionally loaded words like 'excellent' and 'wonderful'. This suggests that focusing on emotionally charged words might be more effective in understanding sentiments accurately.

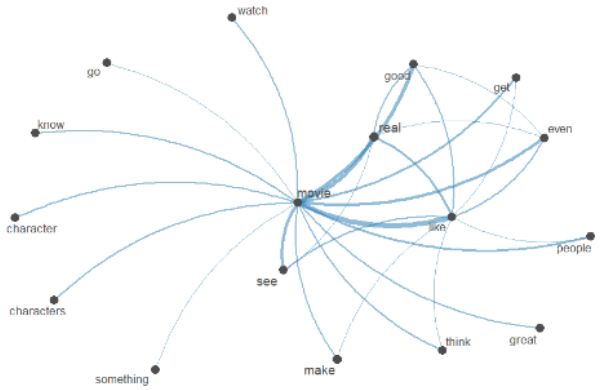


Figure 3: Word Association Network

Finally, we created a word association network to better understand the relationship between common words appearing in the set of reviews. The word association network displays connections between words, offering insights into how words relate to each other in reviews. Thicker connecting lines represent a stronger connection in how two words are used together. Additionally, words placed in the center of the network are most critical in connecting ideas. Here, we see that the strongest connections occur between common words like "movie", "like", and "real" which seem common to all reviews and less powerful in predicting sentiment. Going forward, we would want to find a method to disregard common and useless words in favor of using emotional and qualitative words to conduct analysis and prediction.

2.2 Feature Engineering: Sentiment Lexicon and PCA

Our next step in preparing the data for classification was to begin constructing the dictionary of words that the reviews would be compared to. In our approach to building the word dictionary, we employed the TF-IDF (Term Frequency-Inverse Document Frequency) approach to transform textual data into numerical representations. This technique evaluates word importance within reviews and across the entire dataset. The selection of words was based on

their TF-IDF scores falling within specified thresholds (min_df = 0.1 to max_df = 0.7). By placing these thresholds, we hoped to omit highly appearing yet useless words like "movie" and any rarely appearing words. We should add a sentence here that says how many words we got from TF-IDF.

We leveraged external resources encapsulated in the positive-words.txt and negative-words.txt files [**cite**]. These lists contained words classified as positive or negative based on predefined sentiments.

Principal Component Analysis (PCA) was employed to address high-dimensional feature spaces inherent in text data. We conducted PCA to mitigate the negative effects of dimensionality, aiming to reduce the computational complexity and potential noise within the dataset while preserving the most critical information. Initially, we transformed the textual features into a lower-dimensional space using PCA, a technique that identifies the principal components capturing the variance in the data. The choice of the reduction factor was determined by assessing the cumulative explained variance, ensuring a balance between dimensionality reduction and information retention, thereby enabling efficient sentiment analysis without compromising predictive power. To achieve 95% of variance of the input explained by the generated components, we chose to utilize 95 principle components.

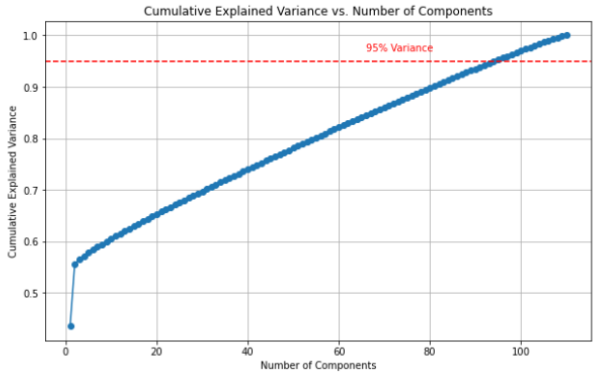


Figure 4: Principal Components by Cumulative Explained Variance

3 Experiment

In assessing the optimal prediction model, we employed four supervised learning algorithms: Logistic Regression, KNN, LDA/QDA, and Random Forests.

3.1 Logistic Regression

The first approach we employed was a logistic regression model. It served as our initial approach due to its relevance and efficacy in handling binary outcome predictions, aligning well with the sentiment analysis task. Logistic regression's simplicity, speed, and flexibility make it an ideal starting point for predictive analysis tasks. Moreover, its computational efficiency and ability to process large volumes of data swiftly provided us with a pragmatic approach to handle the substantial amount of textual information present in movie reviews. Its simplicity also granted us enhanced visibility into the model's internal processes, facilitating easier error identification and correction. Upon training logistic regression on a subset of our dataset (20% for testing purposes), the model demonstrated an accuracy rate of 78.51%, as shown below.

Table 1: Logistic Regression Testing Results

	Precision	**Recall**	**F1-Score**	**Support**
False	0.7840	0.7823	0.7832	4961
True	0.7861	0.7879	0.7870	5039
Accuracy	-	-	0.7851	10000
Macro Avg	0.7851	0.7851	0.7851	10000
Weighted Avg	0.7851	0.7851	0.7851	10000

Table 2: KNN Confusion Matrix

	Predicted False	**Predicted True**
Actual False	3881	1080
Actual True	1069	3970

Furthermore, the analysis of the Receiver Operating Characteristic (ROC) curve unveiled an area under the curve (AUC) of 0.86, as displayed below, affirming the model's robustness in discriminating between positive and negative sentiments.

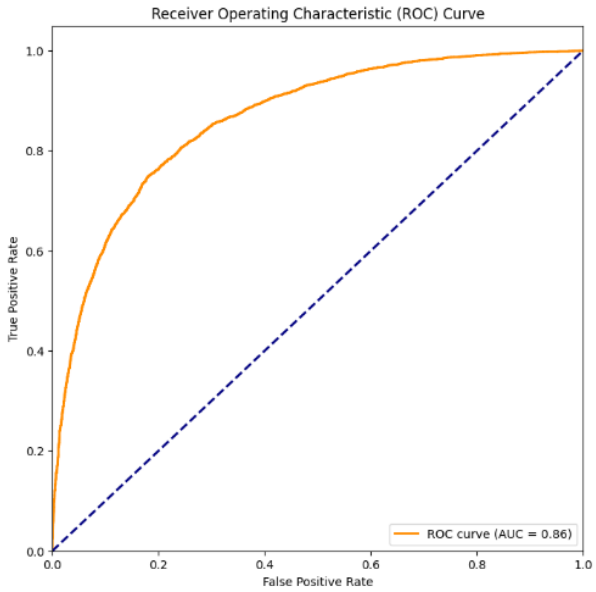


Figure 5: ROC Curve of the Logistic Regression Model

While logistic regression served as an initial benchmark, our study further delved into employing a spectrum of classification methods, including KNN, LDA, QDA, and random forests. Nevertheless, the initial success of logistic regression at 78.5% accuracy and an AUC of 0.86 underscored its importance as an effective starting point in our pursuit of training effective sentiment prediction models for movie reviews.

3.2 KNN

Next, we chose to experiment with the performance of K-Nearest Neighbors classification on the dataset. KNN is a supervised learning classification algorithm [IBM] that compares each testing data point to a training dataset to find a predetermined amount of data points (k) closest or most similar to it. From this group of k data points, the algorithm takes the average of their classes to make a prediction for the testing data point. One important quality of KNN is that it is non-parametric which means that it does not make any assumptions about the data.

When employing KNN and testing its performance, it is important to choose a good value of k because using an inadequate value can upset the balance between underfitting or overfitting of the model [IBM]. The exact tradeoff point for this balance, otherwise known as the optimal value of k, depends on the sample size of the data. For this reason, we decided to test multiple values of k to determine which would maximize the accuracy.

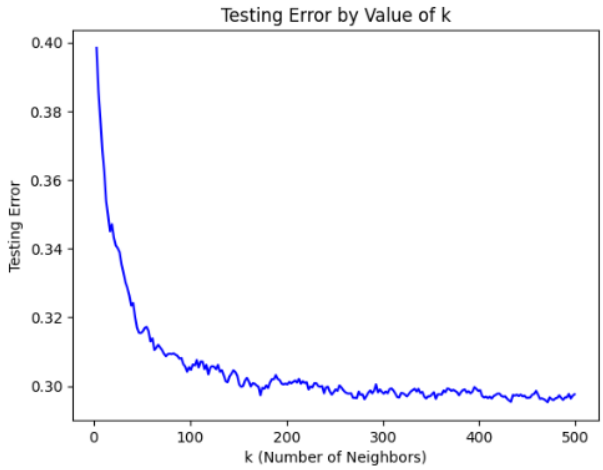


Figure 6: Testing Error by Value of K

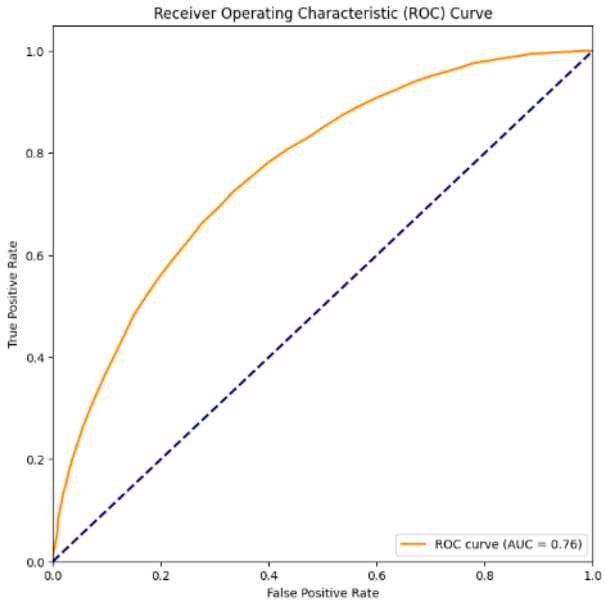
This plot compares the testing error and value of k. Ordinarily, we would expect for this plot to initially decrease and slowly start increasing as the value of k increases. The minimum point of the plot would be our optimal k. However, this plot seems to consistently decrease. It is possible that the sheer size of the dataset could be affecting the value of k. Therefore, we will need to employ a different technique to select our value of k. One common yet less precise method of determining an appropriate choice of k is to let k equal the square root of the sample size, around 223 in this case. To reduce the risk of overfitting, we would prefer to select a medium value of k. Using the scikit-learn library in Python, we fit a KNN model with our medium k = 101 to our data and we found the following results:

Table 3: KNN Testing Results

	Precision	**Recall**	**F1-Score**	**Support**
False	0.7044	0.6648	0.6840	4961
True	0.6873	0.7253	0.7058	5039
Accuracy	-	-	0.6953	10000
Macro Avg	0.6958	0.6951	0.6949	10000
Weighted Avg	0.6958	0.6953	0.6950	10000

Table 4: KNN Confusion Matrix

	Predicted False	Predicted True
Actual False	3298	1663
Actual True	1384	3655



Using the KNN algorithm, we achieved an accuracy of 69%. According to the ROC curve, the area under the curve for our KNN model is 0.76. Both of these figures reflect a performance that is not as accurate as we would have hoped, especially since KNN is supposed to be flexible and relies on the data instead of parameters to do prediction. In addition, due to the process of the KNN algorithm comparing each data point one by one, this classification method proved to be more computationally expensive. Due to these limitations and unsatisfactory accuracy score, we decided to experiment with alternative classification methods.

3.3 LDA/QDA

We also chose to explore the performance of LDA and QDA on our dataset. Unlike the other models that predict the estimate of the posterior class probabilities, LDA and QDA are generative models, which estimate class-conditional densities [Maugis (REF 1)].

LDA assumes that the features of each class have the same covariance matrix and differ only in their means, making it a linear classifier. In contrast, QDA relaxes this assumption and allows for different covariance matrices for each class, making it a more flexible quadratic classifier.

While LDA is simpler and more computationally efficient, it might not capture complex relationships between variables as effectively as QDA, which can better handle nonlinear boundaries between classes. However, QDA requires more data to estimate covariance matrices accurately and can be more prone to overfitting, especially with smaller datasets.

LDA and QDA offer parametric simplicity and explicit probabilistic interpretation compared to non-parametric models like KNN and random forests, especially in scenarios with smaller datasets. Their flexibility in modeling linear and quadratic decision boundaries equips them to better capture complex data relationships, offering valuable insights and computational efficiency in classification tasks.

INSERT TABLE AND ROC GRAPH HERE

Looking at the results, LDA displays higher precision and recall values classes compared to QDA. This implies that LDA has a better overall performance in correctly identifying instances of both classes. However, QDA, despite a slightly lower accuracy, maintains a comparable balance between precision and recall for the given classes. Overall, LDA seems to perform marginally better in precision and recall, while QDA shows a balanced performance with competitive metrics, even with a marginally lower accuracy.

3.4 Random Forests

Our last model, Random Forest, is a method that combines the capabilities of a multi-decision tree to improve predictive accuracy and control overfitting. When employing Random Forest, the balance between model complexity and overfitting is important in achieving optimal performance. The hyperparameters in Random Forest, such as the number of trees and their depth, play a crucial role in shaping the model.

Since our aim is not only merely to optimize accuracy but to understand how different configurations impact the model's interpretability and generalization across the data review, we focused into the depth of individual trees within the Random Forest.

When we trained the mode, we initially chose a practical approach using the default hyper-parameter values provided by the scikit-learn library. Specifically, our model was instantiated with 100 trees (n_estimators = 100) and various maximum tree depths (max_depths).

Our cleaned dataset of 50,000 reviews, each labeled with sentiments (negative or positive), served as the foundation for our experiments. Then, we conducted an exploration using different tree depths (5, 10, 15, and 20) to understand how the tree depth affects the performance of the Random Forest classifier. Based on the exploration, we identified the optimal tree depth that maximized accuracy. The chosen depth was then used to train a Random Forest classifier on the entire training dataset.

INSERT GRAPH ON THE TREE DEPTH ON RANDOM FOREST MODEL ACCURACY HERE

Figure 7: Impact of Tree Depth on Random Forest Model Accuracy

Figure 7 shows that the accuracy of our Random Forest model increased gradually with the tree depth. Starting at an accuracy of 0.82 for a maximum depth of 5, the model showed improvements with accuracies of 0.83 at depth 10 and same at depth 15. The highest accuracy, reaching 0.84, was achieved with a maximum tree depth of 20.

Table 0: Random Forest Test Results

INSERT RANDOM FOREST TEST RESULT TABLE HERE

Table 0: Random Forest Confusion Matrix

INSERT RANDOM FOREST CONFUSION MATRIX HERE

Figure 0: Random Forest ROC Curve

Overall, the model seems to be performing well, with a balanced trade-off between precision and recall. The values in the confusion matrix also suggest that the model is making reasonably accurate predictions for both classes. Also looking at the ROC curve, it is close to the top-left of the graph, which implies that the model exhibits strong predictive capabilities and achieves a good balance between true positive predictions and false positive predictions.

4 Conclusion and Summary

4.1 Results and Analysis

4.2 Final Remarks